Katholieke Universiteit Leuven
Faculteit Ingenieurswetenschappen

Inleiding tot de numerieke wiskunde

Illustraties

Prof. Dr. A. BULTHEEL

Juni 2006

Bij het handboek A. Bultheel "Inleiding tot de numerieke wiskunde", Acco, Leuven, 2006 horen een aantal illustraties die te vinden zijn op de webstek

`http://www.cs.kuleuven.ac.be/~ade/WWW/NW/`.

Deze webpagina's kunnen ook afgehaald worden en lokaal geïnstalleerd op je eigen pc zodat je daar alle lokale bestanden off-line kan bekijken. Deze tekst hoort bij deze pagina's maar bevat niets "extra". Het bespaart je enkel de moeite om de pagina's van het web of je lokale kopie af te drukken. De tekst is ook via Acco-cursusdienst te verkrijgen. De elektronische versie staat onder `NW/ILLST/NW3.pdf`.

Er bestaan twee versies van deze tekst: de papieren (zwart-wit) versie, en een elektronische (kleuren) versie in pdf.

De opdrachten en vragen die kunnen uitgevoerd worden zijn speciaal aangegeven in het rood als het over de elektronische versie gaat of met een speciale aanduiding zoals dit stukje als het over deze tekst gaat.

Het is belangrijk dat je over deze vragen nadenkt en dat je experimenteert met de programma's die beschikbaar gesteld worden. Enkel door ondervinding kan je een feeling opbouwen voor wat numerieke wiskunde is en welke problemen erbij komen kijken en hoe men die moet oplossen. Men zal zo ook ervaring opdoen in verband met het interpreteren van resultaten.

De paragrafen die als "extra" worden aangegeven zijn moeilijker of minder essentieel en kunnen dus in eerste lezing worden overgeslagen.

In de tekst zijn regelmatig verwijzingen opgenomen naar URL's. Sommige zijn extern (`http://...`*) anderen verwijzen naar lokale bestanden. Opdat de lokale verwijzingen correct zouden werken bij aanklikken in de elektronische versie moeten de pdf en de html bestanden op een vaste plaats staan op jouw pc. Daarom zal je de hele webstek moeten downloaden en installeren op een windows pc onder*

`C:\Documents and Settings`.

Staat in de tekst een externe link zoals op lijn drie van deze bladzijde, dan zal je die moeten intikken in je browser als je de papieren versie hebt, of je zal hem kunnen aanklikken in de elektronische versie en je krijgt verbinding met de gerefereerde pagina op voorwaarde dat je on-line bent.

Staat in de tekst een lokale link zoals

`NW/NW/cdvgl/cd/index.html`

dan zijn er verschillende mogelijkheden. Lees je dat in de papieren versie, dan kan je via je browser surfen naar

`http://www.cs.kuleuven.ac.be/~ade/WWW/NW/NW/cdvgl/cd/index.html`

als je on-line bent, of, als je de webstek lokaal hebt geïnstalleerd, dan open je het bestand `C:\Documents and Settings\NW\NW\cdvgl\cd\index.html` *met je*

favoriete browser (of een analoog bestand als je het op een andere plaats hebt geïnstalleerd). Maar als je de webstek lokaal hebt geïnstalleerd, dan heb je meteen ook de elektronische versie en dan zul je door de link aan te klikken meteen op de gewenste lokale pagina komen (maar dit zal enkel werken als je het wel degelijk onder `C:\Documents and Settings` *hebt gezet).*

Een tip: Open de elektronische versie rechtstreeks met een Acrobat Reader en niet in je browser, zodat de pdf niet weg is als je een link opent.

Als je off-line bent, zullen de externe links natuurlijk niet kunnen gelegd worden. Daarom zijn op de webstek alle externe links aangegeven door er [vierkante haken rond te zetten]. Zo weet je dat deze niet bereikbaar zijn als je niet met het Internet verbonden bent.

Inhoudsopgave

Deel III

Illustraties

Hoofdstuk 1

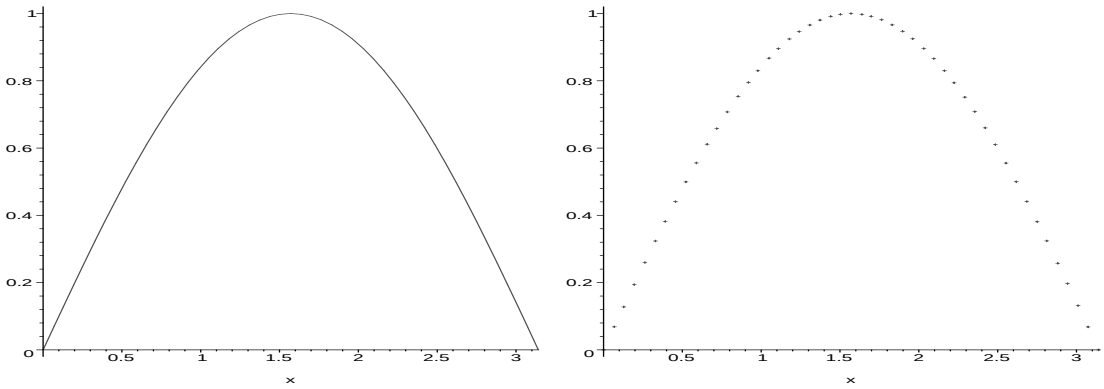
Wat is numerieke wiskunde

1.1 Allemaal digitaal

Men kan zich indenken dat men een figuur maakt in maple. Bijvoorbeeld:

```
> plot(sin(x),x=0..Pi);
```

Het resultaat is de linkse figuur.



Het lijkt alsof de sinus getekend is als een continue lijn die verloopt als een echte sinus. Dit is niet wat er in realiteit gebeurt. Als men de '`help`'-functie gebruikt voor het commando '`plot`', dan kan men vinden onder '`plot[options]`' dat er een parameter '`numpoints`' is die kan ingesteld worden. De standaardwaarde is 50. Dit wil zeggen dat er minstens 50 punten berekend worden van de te tekenen functie in het opgegeven interval, maar, zo zegt de documentatie, maple werkt adaptief, wat betekent dat daar waar er een onregelmatiger verloop is van de functie, er meer punten zullen berekend worden. Hoe dan ook, er zullen nooit oneindig veel punten berekend worden. Met het volgende commando zie je welke punten er berekend worden:

```
> plot(sin(x),x=0..Pi,style=point);
```

Het resultaat staat op de vorige figuur rechts. Het zijn inderdaad ongeveer 50 punten. Met het commando

```
> plot(sin(x),x=0..Pi,numpoints=100,style=point);
```

zullen er ongeveer 100 punten getekend worden en als je `numpoints=1000` stelt zal je een dikke lijn zien die continu lijkt te verlopen, maar die eigenlijk uit 1000 punten bestaat die vlak naast elkaar liggen.

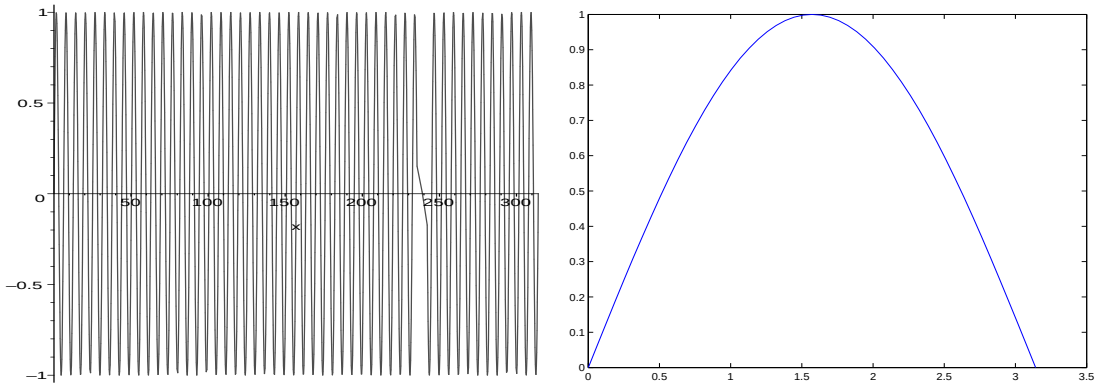
Het is zelfs niet nodig om 1000 of meer punten te tekenen om de indruk van een vloeiende kromme te krijgen. Zelfs met 50 punten kan dat. Eens die 50 punten gekend, zal de figuur

getekend worden door die punten te verbinden met rechte lijnen (veeltermen van de eerste graad) of door er veeltermen door te trekken van graad 2 of 3 die dan op een zachte manier aaneensluiten (splines).

Soms zijn er te weinig punten gegenereerd. Bijvoorbeeld

```
> plot(sin(x),x=0..100*Pi);
```

geeft de volgende figuur links. In de buurt van $x = 240$ is een anomalie te zien. Dit komt omdat daar te weinig punten zijn berekend.



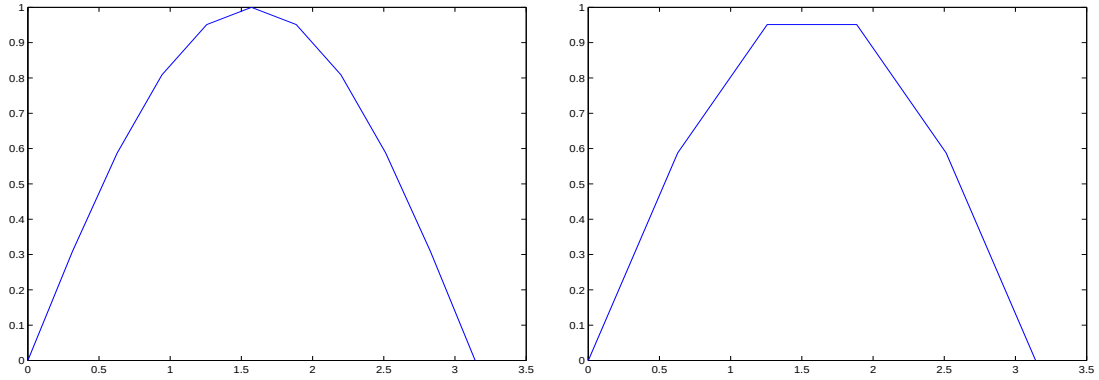
In matlab komt dit nog beter tot uiting omdat je daar zelf expliciet moet opgeven in welke punten je de functie evalueert. Bijvoorbeeld:

```
> x=[0:pi/50:pi]; plot(x,sin(x));
```

geeft de bovenstaande figuur rechts. Met

```
> x=[0:pi/10:pi]; plot(x,sin(x)); x=[0:pi/5:pi]; plot(x,sin(x));
```

krijg je figuren zoals hieronder



Dus als de oplossing van een probleem een continue functie is, dan zal men, als men numerieke waarden voor die oplossing wil hebben niet de functie berekenen in oneindig veel punten, maar slechts in eindig veel punten, en men zal dan een benadering van de functie opstellen, bijvoorbeeld door die punten met rechte lijnen te verbinden. Men kan dan die stukjes veelterm uitrekenen in om het even welk punt.

1.2 Alles is eindig

Zelfs al zou men de figuur maken door oneindig veel punten uit te rekenen, dan nog zou men niet al die punten kunnen tekenen. Een computerscherm heeft maar een eindige resolutie:

men zal horizontaal en verticaal maar een eindig aantal pixels hebben. De (x, y) coördinaten van de pixels die tot de grafiek behoren moeten we ook niet met oneindige nauwkeurigheid kennen. Als er horizontaal voor de tekening maar 100 pixels zijn, dan is het voldoende om de x-coördinaat slechts met een relatieve nauwkeurigheid van 1% te kennen. Analoog voor de y-coördinaat.

Maar op een blad papier waar ik een curve teken zijn er toch oneindig veel punten die oneindig dicht bij elkaar liggen kan men tegenwerpen, maar ook dat is niet waar want als je met een potlood een lijn op een blad papier zet, dan bestaat die uit een aantal stofdeeltjes die op het papier zijn blijven hangen en dat zou onder een elektronenmicroscoop helemaal niet continu zijn. Bovendien zou zo een lijn eigenlijk oneindig dun moeten zijn, geen dikte hebben, en dan zouden we ze niet zien natuurlijk.

Als we de waarde van een sinus met oneindige nauwkeurigheid willen kennen, dan zouden we daar oneindig veel bewerkingen voor moeten maken. De sinus is een transcendente functie, wat betekent dat ze niet met eindig veel elementaire bewerkingen kan berekend worden. Ze is bijvoorbeeld bepaald door de McLaurinreeks

$$\sin(x) = \sum_{k=0}^{\infty} \frac{x^{2k+1}}{(2k+1)!}$$

en oneindig veel termen uitrekenen vraagt oneindig veel tijd. Maar zelfs al zouden we dat kunnen, dan zouden we het resultaat niet eens kunnen stockeren, want dat zou oneindig veel bits vergen die in een oneindig groot geheugen zouden moeten opgeslagen worden, wat oneindig veel energie zou vergen,...

1.3 Benaderingen

Begrippen als reëel getal, continue functie, en alles wat met het oneindig kleine of het oneindig grote te maken heeft zijn dus abstracte begrippen die we concreet nooit zullen tegenkomen. We zullen ons praktisch moeten tevreden stellen met *benaderingen* die voldoende goed moeten zijn voor de doeleinden waarvoor ze moeten dienen. Zo bleek het in ons voorbeeld voldoende te zijn om de functie $\sin(x)$ in $[0, \pi]$ slechts in 50 punten uit te rekenen om een vloeiende kromme te verkrijgen op ons beeldscherm. Maar als in een digitale computer geen echt continue functies bestaan, maar slechts functies die men in een aantal discrete punten heeft berekend, dan heeft het ook geen zin om over de afgeleide of de integraal van zo een functie te spreken. Als men de functie slechts bij benadering kent in een aantal punten $\{(x_i, y_i) : i = 0, \ldots, n\}$, dan zal men de waarde van $f(x)$ voor een $x \notin \{x_0, \ldots, x_n\}$ vinden door interpolatie. Als $x \in (x_i, x_{i+1})$, dan kan men bijvoorbeeld een rechte lijn trekken tussen de punten (x_i, y_i) en (x_{i+1}, y_{i+1}) en daaruit de waarde in x berekenen:

$$f(x) \approx y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i).$$

Dit is een lineaire interpolatie. De afgeleide van de functie zou men in (x_i, x_{i+1}) kunnen benaderen door de richtingscoëfficiënt van de interpolerende rechte. Men zou echter ook een veelterm van hogere graad kunnen gebruiken. Bijvoorbeeld een veelterm van graad n die door alle $n+1$ punten van de tabel gaat, en die evalueren in x om een benadering voor $f(x)$ te krijgen. Dit is het onderwerp van *veeltermінterpolatie*. Dit is niet altijd de beste manier

om een functie te benaderen. We kunnen bijvoorbeeld beschikken over een tabel van een ongekende functie die we hebben bekomen door experimenten uit te voeren waarbij er fouten op de metingen zitten. In dat geval is het veel beter om veeltermen te zoeken die slechts "zo goed mogelijk" door de gemeten punten lopen. Bijvoorbeeld door het criterium der kleinste kwadraten toe te passen. Dat wil zeggen we bepalen de veelterm van graad $m \ll n$

$$p_m(x) = a_0 + a_1 x + \cdots + a_m x^m$$

door te eisen dat de vector met waarden $p_m(x_i)$ zo dicht mogelijk bij de vector van de gegeven waarden y_i ligt. Bijvoorbeeld door het overgedetermineerde stelsel

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^m \\ 1 & x_1 & & x_1^m \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{1.1}$$

op te lossen in de zin der kleinste kwadraten, d.w.z. zodanig dat

$$\sum_{i=0}^{n} (y_i - p_m(x_i))^2$$

zo klein mogelijk is.

Eenmaal de benadering (veelterm of niet, interpolerend of niet) van de functie gevonden, is het dikwijls gemakkelijk om de benadering voor de afgeleide of de integraal van de functie te vinden door de benadering af te leiden of te integreren. Op die manier wordt het dus mogelijk om een discrete functie toch te differentiëren en te integreren.

Men kan eenzelfde techniek toepassen om het nulpunt te vinden van een transcendente functie. In de buurt van waar men het nulpunt vermoedt, zal men de functie in bijvoorbeeld 2 punten evalueren, en door die punten een rechte lijn trekken. Het snijpunt van de rechte met de x-as levert een nieuw punt op dat hopelijk wat dichter bij het echte nulpunt ligt dan de 2 startpunten. Men kan daar weer de functie evalueren en samen met één van de vorige punten, weer een rechte bepalen die de x-as snijdt in een nieuw punt enzovoort. Eenzelfde strategie kan men volgen als men een minimum van een functie zoekt. Men vertrekt van 3 punten waardoor men een parabool trekt waarvan men een minimum zoekt wat een nieuw punt oplevert enzovoort.

Hoe goed men het probleem ook benadert, als het theoretisch oneindig veel bewerkingen vraagt om het resultaat te berekenen, dan zal men steeds een *benaderingsfout* moeten maken door de berekeningen af te breken na eindig veel rekentijd. Men spreekt in dat geval ook van *afbrekingsfout*. Indien de benadering ontstaat doordat men een continu probleem maar gaat berekenen in een eindig aantal discrete punten, dan spreekt men soms van *discretisatiefout*.

1.4 Eindige nauwkeurigheid

Niet alle numerieke problemen moeten noodzakelijk oneindig veel bewerkingen of oneindig veel geheugen vergen, maar zelfs dan zal men toch een oplossing berekenen die maar bij benadering juist is. Neem bijvoorbeeld het stelsel (1.1) dat we hiervoor beschreven, en stel dat $m = n$. We hebben dan een vierkant stelsel en omdat de matrix van het stelsel een matrix

van Vandermonde is, heeft het stelsel juist één oplossing (als alle x_i onderling verschillen). Stel even dat de gegeven x_i en y_i allemaal exact zijn. Dit is in het algemeen niet waar want het zijn punten die misschien meetpunten zijn of misschien berekende waarden die dus maar benaderingen zijn. Ook al zou men ze exact kennen (bijvoorbeeld $x_n = \pi$), dan kan men dat nog niet exact in de computer inbrengen want π bevat oneindig veel cijfers en zal dus een oneindig geheugen vragen. Voor elk reëel getal is er in de computer maar een beperkt en vooraf vastgelegd aantal bits voorzien om het op te slaan. Stel dat de computer 32 bits voorziet om een reëel getal te bewaren. Met die 32 bits kunnen hoe dan ook maar eindig veel verschillende reële getallen voorgesteld worden. Als men een getal in de computer invoert dan zullen we ervan uitgaan dat de computer, als hij dat getal niet exact kan voorstellen, in de plaats daarvan het getal zal opslaan dat, onder alle getallen die hij wel kan voorstellen, zo dicht mogelijk bij het ingegeven getal zit. Daarbij zullen al berekeningen moeten gebeuren want wij geven het getal waarschijnlijk in in zijn decimale vorm en de computer bewaart dat in binaire of hexadecimale vorm. Laten we van dit laatste nog even abstractie maken, en veronderstel dat de computer maar plaats reserveert om 6 decimale cijfers na de komma te stockeren. Als x_i uit 6 decimale cijfers bestaat, dan zal x_i^5 normaal uit 30 decimale cijfers bestaan. Vermits dat resultaat ergens in het geheugen van de computer moet opgeslagen worden zal hij 24 cijfers van de 30 moeten weggooien. We veronderstellen daarbij dat hij op de juiste manier afrondt en echt het dichtstbijzijnde getal van 6 cijfers bewaart. Zo gaat dat voor alle getallen in de matrix van het stelsel. Nadien zal men op het stelsel bijvoorbeeld de methode van Gauss toepassen om het op te lossen, maar daar moeten weer $O(n^3)$ bewerkingen gebeuren, en voor elke bewerking zal de computer het resultaat van die bewerking moeten afronden om het te kunnen stockeren. Het gevolg is dat de computer in numerieke berekeningen praktisch in elke stap fouten zal maken. Dit zijn *afrondingsfouten.*

1.5 Fouten, fouten en nog eens fouten

Het besluit van de vorige paragrafen is dat een computer in numerieke berekeningen niets anders maakt dan fouten: benaderingsfouten als men het probleem niet met eindig veel bewerkingen kan oplossen en afrondingsfouten omdat getallen in de computer maar met eindig veel bits worden opgeslagen met als gevolg dat er praktisch geen enkel getal kan ingevoerd worden en geen enkele bewerking uitgevoerd worden zonder dat er daarbij fouten ontstaan. Die fouten ontstaan op elk moment maar ze kunnen zich ook accumuleren. Fouten helemaal in het begin van de berekeningen gemaakt kunnen meegesleept worden door alle berekeningen en uiteindelijk zwaar wegen op het resultaat.

Bij dit alles maken we abstractie van "menselijke fouten" omdat een methode verkeerd geprogrammeerd is, of per vergissing een komma verkeerd geplaatst wordt bij het invoeren, of doordat een stel gegevens voorkomt waaraan de programmeur niet gedacht had bij het schrijven van het programma.

1.6 Numeriek versus analytisch

Men kan zich bij al deze bedenkingen afvragen of het dan wel zinvol is de berekeningen numeriek te doen. Maple kan toch symbolisch rekenen, waarom lossen we dan niet alle problemen op met een symbolisch pakket? Daar zijn natuurlijk ook beperkingen bij. Maple kan maar problemen symbolisch oplossen in zoverre dat er een symbolische oplossing bestaat. Vraag

aan maple simpelweg de waarde van de sinus in $x = 0.1$, en dit zal numeriek moeten berekend worden. De oplossing van een stelsel lineaire vergelijkingen met rationale coëfficiënten kan men exact oplossen met eindig veel bewerkingen en de oplossing zal rationaal zijn. Als het stelsel wat groot is, zullen teller en noemer van die rationale uitdrukkingen zeer groot worden en heel onoverzichtelijk. Bovendien zullen bewerkingen met dergelijke grote getallen en tijdens de berekeningen steeds maar groeiende getallen, steeds meer rekenwerk vragen en tenslotte onaanvaardbaar lang duren. Het berekenen van de grootste gemene deler van die getallen om ze te vereenvoudigen vraagt veel tijd, ook al zijn daar snelle algoritmes voor. Maar zelfs al zou men dit probleem nog willen aannemen, wat doet men dan als die getallen reëel zijn of als het over een niet lineair stelsel gaat. Voor een beperkte klasse van differentiaalvergelijkingen kan men analytische oplossingen berekenen, maar dikwijls zijn de differentiaalvergelijkingen die men moet oplossen niet van dat type.

Natuurlijk is een symbolisch pakket als maple niet overbodig. Het kan in zeer veel gevallen zeer nuttig zijn, maar soms is het onmogelijk om een analytische oplossing te vinden, en dan moet men wel zijn toevlucht nemen tot een numerieke techniek. Daarover gaat deze cursus.

Hoofdstuk 2

Convectie-diffusievergelijking

Laten we meteen met vuurwerk beginnen en een moeilijk numeriek probleem formuleren en laten we nagaan hoe daarvoor een numerieke oplossing kan gevonden worden.

2.1 Definitie en relevantie

Convectie-diffusievergelijkingen zijn partiële differentiaalvergelijkingen van de vorm

$$\frac{\partial u(x,t)}{\partial t} - \alpha \frac{\partial^2 u(x,t)}{\partial x^2} + c(x,t)\frac{\partial u(x,t)}{\partial x} = f(x,t).$$

Hierin is $\alpha\frac{\partial^2 u}{\partial x^2}$ de zg. *diffusieterm* en $c\frac{\partial u}{\partial x}$ de *convectieterm*. De vergelijking heeft

Beginvoorwaarde: $u(x,0) = w(x)$, $0 \leq x \leq L$

Randvoorwaarden: $u(0,t) = u_0$, $u(L,t) = u_L$

Het probleem is $u(x,t)$ te zoeken voor $0 \leq x \leq L$ en $t > 0$.

Dit soort vraagstukken komt voor in de stromingsleer of vloeistofmechanica. Dit is belangrijk voor het modelleren van een vliegtuigcockpit of een auto waar de lucht rondstroomt. Ook voor warmtetransport in bv. metalen onderdelen van een machine enz. Het beschrijft bijvoorbeeld de diffusie van een stof (of temperatuur) in een stromende vloeistof. In zo een geval stelt u de concentratie van de stof (of de temperatuur) voor. Als je op $t = 0$ en $x = 0$ een hoeveelheid massa van de stof inbrengt beschrijft de vergelijking hoe die stof zich zal verspreiden in functie van de afstand en de tijd. De oplossing is een steeds breder wordende Gauss-curve.

De diffusieterm beschrijft hoe de stof zal uitdeinen in de vloeistof (zelfs als er geen stroming is) en de convectieterm heeft te maken met de verplaatsing van de stof doordat de vloeistof stroomt. De coëfficiënt $c(x,t)$ is de snelheid van de vloeistof op de plaats x en het ogenblik t. Bijvoorbeeld warmte verplaatst zich vanwege de luchtverplaatsing boven een verwarmingselement (convectie), maar er zal ook warmte uitstralen tot op een zekere afstand van het element omdat er diffusie optreedt. Analoog zal een druppel kleurstof in stromend water niet alleen meegesleurd worden (convectie) maar ook uitdeinen (diffusie).

Men kan zich voorstellen dat het analytisch oplossen van dergelijke problemen enorm moeilijk zal zijn, als het al mogelijk is. Numeriek zal men echter toch een oplossing moeten vinden.

Hoe moet zoiets gedaan worden?

2.2 Een voorbeeld: de warmtevergelijking

Stel dat we een staaf beschouwen die aan de linkerkant ($x = 0$) tegen een warm medium wordt gehouden en aan de rechterkant ($x = L$) op een constante temperatuur $u = 0$ wordt gehouden.

Het is de bedoeling om de temperatuursverdeling $u(x, t)$ in de staaf ($0 \leq x \leq L$) in de loop van de tijd ($t \geq 0$) te berekenen.

De temperatuur $u(x, t)$ voldoet aan de warmtevergelijking (zie cursus natuurkunde 1e kan.)

$$\alpha \frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}.$$

Hierin hangt $\alpha = K/(C\rho)$ af van K de warmtegeleidingscoëfficiënt, ρ het soortelijk gewicht, en C de soortelijke warmte. Men noemt het de diffusiviteit van het medium (in dit geval van staal).

Vermits er hier geen sprake is van een stromende vloeistof is er dan ook geen convectieterm in de vergelijking.

Beginvoorwaarde: We veronderstellen dat op het ogenblik $t = 0$ de temperatuur in de ganse staaf 0°C is en dat de temperatuur in $x = 0$ plots van 0 naar 100°C gebracht wordt: $u(x, 0) = 0$ voor $x > 0$ en $u(0, 0) = 100$.

Randvoorwaarden:

1. De temperatuur aan het uiteinde $x = L$ wordt vast gehouden op 0°C: $u(L, t) = 0$
2. De warmteuitwisseling bij het contactpunt wordt beschreven door

$$\left.\frac{\partial u(x,t)}{\partial x}\right|_{x=0} = \frac{H}{K}(u(0,t) - T_{\text{omgeving}})$$

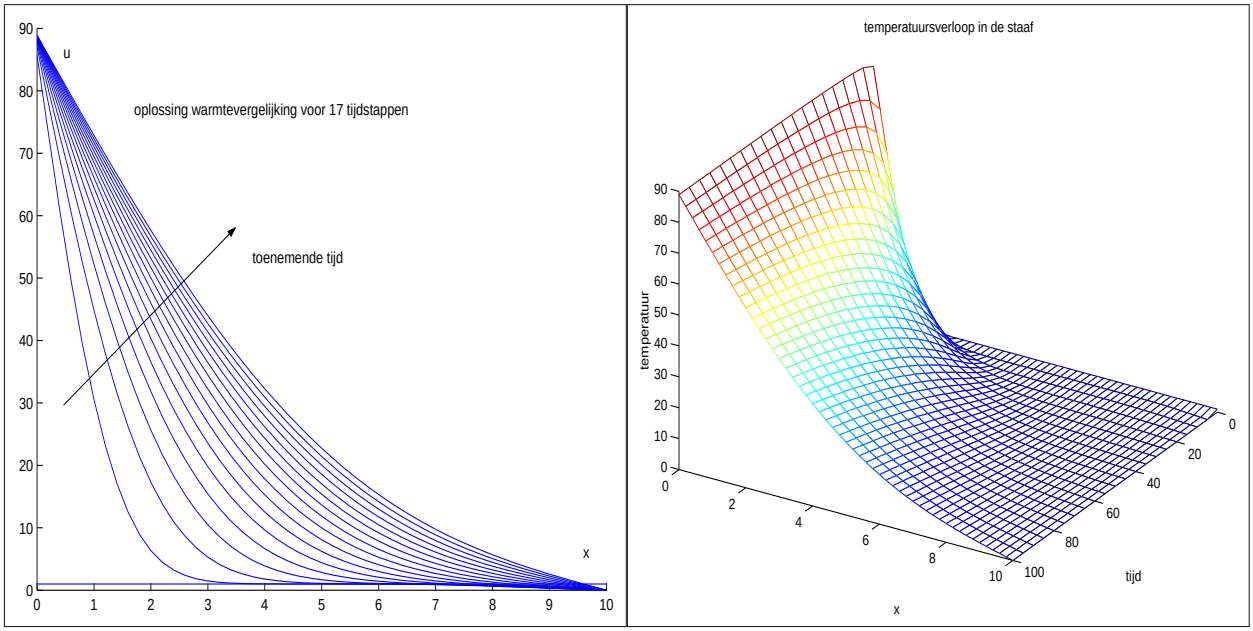
Hierin is H de convectieve warmteoverdrachtscoëfficiënt en K de warmtegeleidingscoëfficiënt.

Men ziet dat dit een CD-vergelijking is met $c = 0$ en $f = 0$.

Het probleem werd opgelost via een matlab programma en het resultaat is op de onderstaande figuur weergegeven.

De gegevens waren

naam	parameter	waarde	eenheden
warmteoverdrachtscoëfficiënt	H	500	J/m^2 s K
warmtegeleidingscoëfficiënt	K	45	J/m s K
lengte staaf	L	10	m
soortelijke warmte	C		J/kg K
massadichtheid	ρ		kg/m^3
diffusiviteit $\alpha = \frac{K}{C\rho}$	α	0.1	m^2/s
temperatuur in $x = L$	u_L	0	°C
omgevingstemperatuur in $x = 0$	T_{omgeving}	100	°C



Er worden 50 temperatuursverdelingen getekend met tussenstappen in de tijd van $h_t = 0.5$. Merk op hoe de temperatuur verandert van een discontinue verdeling (overal 0, behalve in $x = 0$ waar de temperatuur $u = 100$ is) naar een bijna lineaire verdeling tussen $u(0, t) = 100$ en $u(L, t) = 0$.

Het is duidelijk dat hier de oplossing berekend is voor discrete tijdstappen.

Weet je hoe men voor een bepaalde tijd (bv $t = t_j$) de functie $u(x, t_j)$ tekent in matlab?

Als je het antwoord op deze vraag weet, dan zul je ook weten dat men de oplossing zal berekenen op een rooster $u(x_i, t_j)$, met $x = (x_1, \ldots, x_n) =$ `0:hx:L` en $t = (t_1, \ldots, t_m) =$ `0:ht:Tmax`. We noteren $u(x_i, t_j) = u_i^j$.

We zullen verder uitvoeriger terugkomen op deze vergelijking.

2.3 De transportvergelijking

Om te laten zien hoe we zo een probleem aanpakken zullen we een eenvoudiger vorm van het oorspronkelijk probleem kiezen. Stel dat de diffusieterm niet optreedt. We zoeken dan de oplossing van

$$\frac{\partial u(x,t)}{\partial t} + c_0 \frac{\partial u(x,t)}{\partial x} = 0.$$

We veronderstellen c_0 constant. (Ga na dat het de dimensie van een snelheid heeft, nl. $[c_0] =$ m/s.) Dit is een heel eenvoudig probleem. Men veronderstelt geen diffusie. Een olieverfdruppel in stromend water zal als druppel meegenomen worden met de stroming.

De vergelijking heeft

Beginvoorwaarde: $u(x, 0) = w(x)$, $0 \leq x \leq L$

Randvoorwaarden: $u(x, t) = u(L + x, t)$ (cyclische randvoorwaarden)

Het probleem is nog steeds $u(x, t)$ te zoeken voor $0 \leq x \leq L$ en $t > 0$.

Men kan eenvoudig analytisch afleiden (en dit is ook fysisch zeer plausibel) dat de oplossing van dit probleem gegeven wordt door

$$u(x,t) = w(x - c_0 t),$$

waarin $w(x)$ de beginvoorwaarde is. Dit betekent dat de beginoplossing in de loop van de tijd gewoon opschuift in de x-richting.

Nu we de exacte oplossing kennen is het eenvoudig om na te gaan of de berekende oplossing wel correct is.

Uit het vorige weten we dat we slechts een oplossing moeten berekenen op een discreet rooster $u(x_i, t_j)$ met $x_i = ih_x$ en $t_j = jh_t$. De h_x en h_t zijn stapjes die men doet in de x en de t-richting respectievelijk. We stellen $u_i^j = u(x_i, t_j)$.

De afgeleide naar x in het punt (x_i, t_j) als men enkel de functie in de roosterpunten kent zou men kunnen benaderen door

$$\left.\frac{\partial u(x,t)}{\partial x}\right|_{(x,t)=(x_i,t_j)} = \frac{u_{i+1}^j - u_i^j}{h_x}. \tag{2.1}$$

En analoog is de afgeleide naar t benaderd door

$$\left.\frac{\partial u(x,t)}{\partial t}\right|_{(x,t)=(x_i,t_j)} = \frac{u_i^{j+1} - u_i^j}{h_t}.$$

We kunnen de continue vergelijking dus benaderen door

$$\frac{u_i^{j+1} - u_i^j}{h_t} + c_0 \frac{u_{i+1}^j - u_i^j}{h_x} = 0.$$

Of nog, met $\lambda = c_0 h_t / h_x$:

$$u_i^{j+1} = (1+\lambda) u_i^j - \lambda u_{i+1}^j.$$

Vermits we $u_i^0 = w_i$ kennen, kunnen we hieruit de u_i^j op alle volgende tijdstippen j berekenen voor $i = 1, \ldots, n$ (wegens de cyclische randvoorwaarden stellen we $u_0^j = u_n^j$ en $u_{n+1}^j = u_1^j$).

In (2.1) doen we een voorwaartse stap om de afgeleide te benaderen. We hadden ook een achterwaartse stap kunnen nemen:

$$\left.\frac{\partial u(x,t)}{\partial x}\right|_{(x,t)=(x_i,t_j)} = \frac{u_i^j - u_{i-1}^j}{h_x} \tag{2.2}$$

of een gemiddelde van de twee:

$$\left.\frac{\partial u(x,t)}{\partial x}\right|_{(x,t)=(x_i,t_j)} = \frac{u_{i+1}^j - u_{i-1}^j}{2h_x}. \tag{2.3}$$

Ga zelf na hoe men voor deze benaderingen de oplossing u_i^{j+1} op het ogenblik $j+1$ kan berekenen vanuit de oplossing u_i^j op het ogenblik j.

Indien we het schema

$$\left.\frac{\partial u(x,t)}{\partial t}\right|_{(x,t)=(x_i,t_j)} = \frac{u_i^j - u_i^{j-1}}{h_t} \tag{2.4}$$

gebruiken voor de afgeleide naar de tijd, dan kan men niet meer rechtstreeks u_i^{j+1} berekenen uit u_i^j, maar bijvoorbeeld de combinatie (2.4) en (2.1) levert

$$(1-\lambda)u_i^j + \lambda u_{i+1}^j = u_i^{j-1}.$$

Hier moet dus een stelsel opgelost worden om u_i^j te vinden in functie van u_i^{j-1}. Namelijk

$$\begin{bmatrix} 1-\lambda & \lambda & 0 & \cdots & \cdots & 0 \\ 0 & 1-\lambda & \lambda & \ddots & \ddots & \vdots \\ 0 & \ddots & 1-\lambda & \lambda & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1-\lambda & \lambda \\ \lambda & 0 & \cdots & & 0 & 1-\lambda \end{bmatrix} \begin{bmatrix} u_1^j \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ u_n^j \end{bmatrix} = \begin{bmatrix} u_1^{j-1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ u_n^{j-1} \end{bmatrix}$$

Ga zelf na hoe men de formule (2.4) kan combineren met (2.2) en (2.3).

2.4 Matlab experimenten

Programmeer de methodes in matlab, of gebruik de voorgeprogrammeerde methodes om een aantal experimenten te doen.

1. Combineer de verschillende mogelijkheden voor het benaderen van de afgeleiden in x en t richting. Welke geeft de beste resultaten?
2. Heeft het wijzigen van h_x of h_t een invloed op de nauwkeurigheid van de berekeningen?
3. Zijn er bepaalde waarden van de fysische parameter c_0 waarvoor men een betere numerieke oplossing krijgt dan voor andere?
4. Heeft een scalering in de x- of de t-richting een invloed op de nauwkeurigheid? Bv. x in cm i.p.v. meter uitdrukken en/of t in ms i.p.v. s. Is het voldoende om h_x en/of h_t te wijzigen als je dit wil testen?
5. Een relatief goed werkende methode is de methode van Lax. Hierbij gebruikt men voorwaartse differenties in de tijd en centrale differenties in de ruimte. Bovendien vervangt men de u_i^j uit de tijdsdifferentiatie door $(u_{i+1}^j + u_{i-1}^j)/2$. Ga na hoe de matrix dan moet opgesteld worden. (Dit correspondeert met invoer (4,2) in de matlab code `tr_0` en dus met `case 8` in deze code).
6. Probeer voor de verschillende waarden van de parameters en voor de verschillende discretisatieschemas eens de gevallen $h_t < h_x/c_0$ en $h_t > h_x/c_0$. Welk verschil merk je? Kan je daaruit iets besluiten over de stabiliteit van de gebruikte methode?

2.5 Java experimenten

Voor experimenten met een java applet zie `/NW/NW/cdvgl/cd/index.html`

Hoofdstuk 3

Fouten in numerieke wiskunde

3.1 Afrondingsfouten

3.1.1 Berekenen van lim $(1 - \cos(x))/x^2 = 1/2$

We berekenen met maple de waarde van $(1 - \cos(x))/x^2$ in $x = 1.2e - 5$ met 10 decimale cijfers:

```
> Digits:=10;
> x=1.2e-5;
> (1-cos(x))/x^2;
```

Dit geeft helemaal niet het verwachte resultaat!

Ga na. Herneem de berekening met `Digits:=30;`. Krijg je nu het verwachte resultaat?

Opmerking: In principe rekent maple intern met decimale cijfers. Om de snelheid op te drijven worden echter bewerkingen met matrices en vectoren vanaf versie 7 in hardware uitgevoerd, wat betekent dat dat binair zal zijn. Indien men `Digits` groter maakt dan `evalhf(Digits)` (wat normaal 15 is op 32 bit machines) dan gebeuren ook de berekeningen in software (dus exact en decimaal). Je kan een bewerking in hardware van `expr` forceren door `evalhf(expr)` te schrijven.

Vervang `(1-cos(x))/x^2;` in de vorige code door `evalhf((1-cos(x))/x^2);`. Welke uitkomst krijg je nu? Kan je dat verklaren?

3.1.2 Berekenen van lim $(1 + 1/x)^x = e$

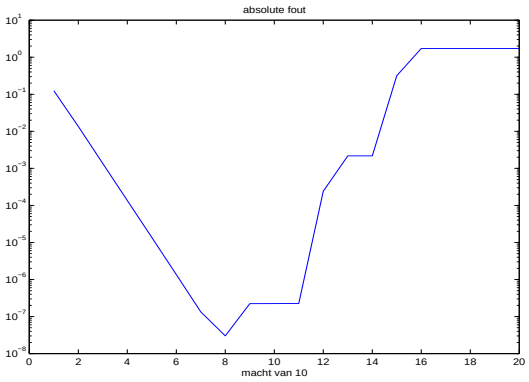
We weten dat

$$\lim_{x\to\infty} (1 + 1/x)^x = e.$$

Wanneer we de limiet proberen te benaderen door $(1 + 1/x)^x$ uit te rekenen in matlab (d.w.z. met ongeveer 16 decimale cijfers) voor $x = 10^1, 10^2, \ldots, 10^{20}$ krijgen we voor de lagere machten van 10 een telkens betere benadering maar daarna wordt de absolute fout

terug groter.

```
for i=1:20
  x=10.0^i;
  ae(i)=(1+1/x)^x;
end;
% plot van de absolute fout
figure(1)
semilogy(abs(ae-exp(1)))
title('absolute fout')
```



Kan je precies verklaren waarom de figuur verloopt zoals ze verloopt?
Waarom is er op het einde van de figuur een horizontaal stuk? Waarom ontstaat dit stuk bij $x = 10^{16}$?
Waarom daalt de figuur tot ongeveer 10^{-7} à 10^{-8} en waarom gebeurt dit bij $x = 10^{8}$?

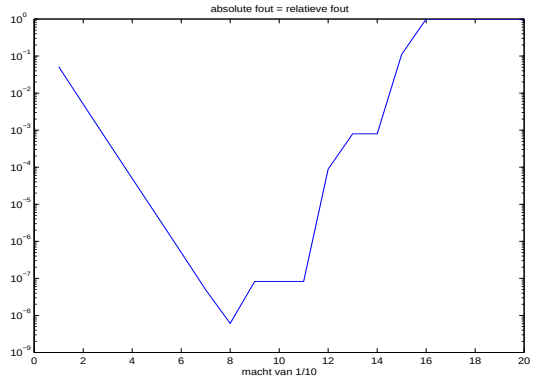
Opmerking: Matlab zal intern de IEEE standaard gebruiken. Dit wil zeggen dat getallen binair voorgesteld worden met dubbele IEEE nauwkeurigheid, wat betekent ongeveer 16 à 17 decimale cijfers.

3.1.3 Berekenen van lim $(e^x - 1)/x = 1$

Hetzelfde doet zich voor wanneer we de volgende limiet willen benaderen in matlab.

$$\lim_{x \to 0} \frac{e^x - 1}{x} = 1.$$

```
for i=1:20
  x=10^(-i);
  res(i)=(exp(x)-1)/x;
end;
figure(1)
semilogy(abs(res-1))
title('absolute fout = relatieve fout')
```



Probeer analoge vragen als hierboven te beantwoorden.
Kan je de opvallende gelijkenis met de vorige figuur verklaren?

3.1.4 Oplossen van een vierkantsvergelijking

Het benaderen van de kleinste oplossing in absolute waarde van een kwadratische vergelijking volgens de klassieke formule kan een groot verlies aan nauwkeurigheid opleveren.

```
% kwadratische vergelijking a*x^2+b*x+c=0
clear; format long e
xklein = 10e-8; xgroot = 10e+7; exacte_wortels = [xgroot xklein]

a = 1;
b = -(10e7+10e-8);  %-(xklein+xgroot)
c = 10e7*10e-8;     %xklein*xgroot
coef_van_vkv = [a b c]

D = b^2-4*a*c; x1 = (-b+sqrt(D))/(2*a); x2 = (-b-sqrt(D))/(2*a);
klassieke_formule = [x1 x2]
% de kleinste wortel in absolute waarde kan nauwkeuriger berekend worden
if b<0, x1n= (-b+sqrt(D))/(2*a); else x1n= (-b-sqrt(D))/(2*a); end
x2n = c/x1;
nauwkeuriger = [x1n x2n]
relatieve_fout_klassiek    = [(x1 -xgroot)/xgroot (x2 -xklein)/xklein]
relatieve_fout_alternatief = [(x1n-xgroot)/xgroot (x2n-xklein)/xklein]
```

Ga na waarom de alternatieve formule nauwkeuriger is dan de klassieke.

3.1.5 Eenvoudige berekeningen

Raar maar waar: de optelling is numeriek gezien niet meer associatief.

Speel deze matlab sessie eens na. Enig idee wat hier aan de hand is?

```
>> x = 0.1
x =
    0.1000
>> x - x
ans =
     0
>> x + x - x - x
ans =
     0
>> x + x + x - x - x - x
ans =
   2.7756e-17
>> x + x - x + x - x - x
ans =
     0
```

```
>> x = 0.01
x =
    0.0100
>> x - x
ans =
     0
>> x + x - x - x
ans =
     0
>> x + x + x - x - x - x
ans =
  -3.4694e-18
>> x + x - x + x - x - x
ans =
     0
```

3.1.6 Praktijkvoorbeelden

Een aantal voorbeelden waar afrondingsfouten desastreuze gevolgen hadden:

- Op 25 januari 1991 gedurende de golfoorlog werd een Iraakse Scud raket afgevuurd op de geallieerde troepen in Dharan, Saudi Arabië. De afweer bestond in het lanceren van

een Patriot raket die de Scud moest onderscheppen. De Patriot miste de Scud zodat deze laatste op een troepenverblijf viel en 28 doden maakte.

De oorzaak: afrondingsfouten.

- Op 4 juni 1996 werd in Frans Guiana de eerste van de reeks Ariane 5 raketten gelanceerd door de ESA. Alles ging goed de eerste 36 seconden. Daarna ging de raket uit haar koers en vernietigde zichzelf. De ontwikkeling had 7 miljard dollar gekost. De kostprijs van de raket zelf was ongeveer 500 miljoen dollar.

  De oorzaak: overloop (de getallen werden te groot).

- In 1982 installeerde de Beurs van Vancouver een nieuwe index op de waarde van 1000. Deze index werd aangepast na iedere verrichting. Twintig maanden later was de index gezakt tot 520. De echte waarde was 1098.892.

  De oorzaak: afrondingsfouten

3.2 Stabiliteit van een algoritme

We hebben in vorige paragraaf gezien dat afrondingsfouten in soms zeer eenvoudige berekeningen tamelijk grote gevolgen kunnen hebben. De mate waarin de benaderingsfouten en afrondingsfouten het resultaat van een algoritme beïnvloeden wordt aangegeven door de stabiliteit van het algoritme.

3.2.1 Evalueren van een veelterm

We illustreren het begrip stabiliteit met de evaluatie van een veelterm. Je kan een veelterm op veel manieren voorstellen en evalueren. Stel bijvoorbeeld dat x_i, $i = 1, 2, \ldots, n$ de wortels zijn van de veelterm met multipliciteit m_i en dat $\sum_{i=1}^{n} m_i = m \leq 10$.

Zij $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ de veelterm die deze x_i als nulpunten heeft.

Met de applet die je kan vinden op `NW/NW/stab/java/index.html` kan je deze veelterm tekenen (dit wil zeggen hem evalueren in een groot aantal punten en die coördinaten uitzetten in grafiek).

Daarbij kan je de veelterm evalueren op 3 verschillende manieren:

1. Door de factoren te vermenigvuldigen (rode curve)

$$f(x) = \prod_{i=1}^{n} (x - x_i)^{m_i}$$

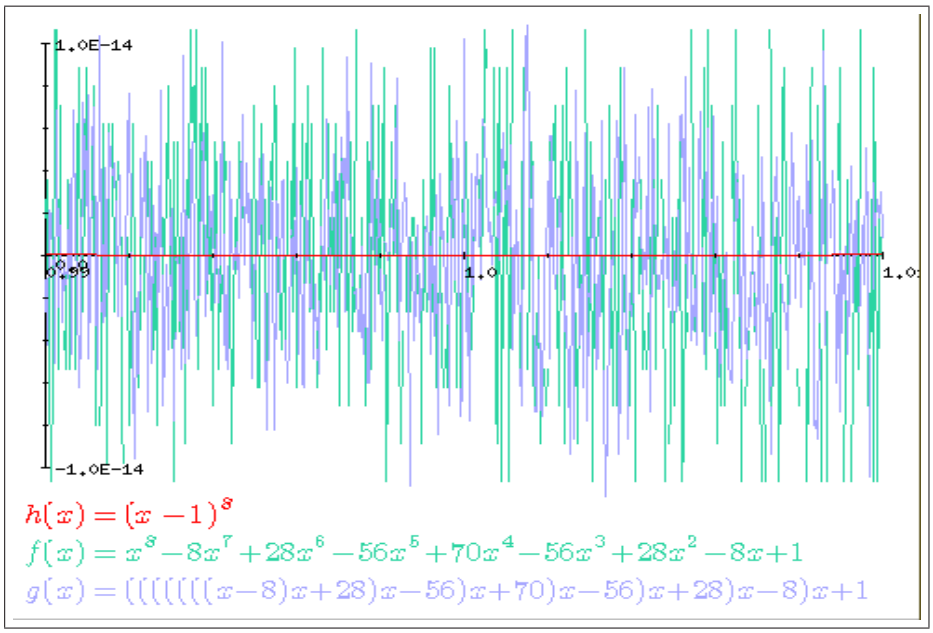
2. Door eerst de coëfficiënten a_k en dan de machten uit te rekenen en op te tellen (groene curve)

$$g(x) = \sum_{k=0}^{m} a_k x^k$$

3. Door eerst de coëfficiënten a_k en dan de geneste vermenigvuldiging uit te rekenen (Hornerschema = blauwe curve)

$$h(x) = ((\cdots(a_m x + a_{m-1})x + a_{m-2})x + \cdots + a_1)x + a_0)$$

Om de kleuren te kunnen waarnemen zul je de applet moeten gebruiken want op de zwart-wit figuur hiernaast is dit niet te zien.
Als we enkel gehele getallen toelaten als wortels, dan kan de omzetting naar de coëfficiënten a_k exact gebeuren. De afrondingsfouten zijn dus enkel afkomstig van de evaluatie van de uitdrukkingen.



Opgaven en experimenten

- Door het bereik van de x en de y te wijzigen kan je inzoomen op een stuk van de grafiek. De berekende punten zouden een gladde curve moeten vormen. Door afrondingsfouten wordt dit vervangen door ruis. Is deze ruis uniform verdeeld?
- De illustratie laat de ruis zien in de buurt van een 8-voudig nulpunt. Is het fenomeen even erg als je niet in de buurt van een nulpunt zit?
- Heeft het feit dat het een nulpunt is met een hoge meervoudigheid er iets mee te maken? Probeer eens met allemaal verschillende nulpunten.
- Van waar komen die afrondingsfouten? Heeft het iets te maken met de omzetting naar de coëfficiënten a_k in de laatste twee gevallen (groen en blauw)?

Extra

Waarom is de rode curve nauwkeuriger dan de groene of de blauwe?
Kun je hier een verklaring voor geven aan de hand van een gedetailleerde foutenanalyse?

3.2.2 Berekenen van een recursiebetrekking

De recursie

Beschouw de recursiebetrekking

$$x_{n+1} = ax_n + bx_{n-1}$$

met

$$a = \alpha_1 + \alpha_2 \quad \text{en} \quad b = -\alpha_1\alpha_2$$

Dus zijn α_1 en α_2 oplossingen van de vergelijking $\alpha^2 = a\alpha + b$, en daarom is de oplossing van de recursiebetrekking te schrijven als

$$x_n = c_1\alpha_1^n + c_2\alpha_2^n.$$

Kiest men $x_0 = 1$ en $x_1 = \alpha_i$, dan is de oplossing $x_n = (\alpha_i)^n$.

De applet

In de applet worden α_1 en α_2 aan de gebruiker gevraagd. Daaruit worden a en b berekend en dan worden de oplossingen $(\alpha_2)^n$ en $(\alpha_2)^n$ berekend volgens de recursiebetrekking voor $0 \leq n \leq 20$.

Men krijgt uit de applet de overeenkomstige fout op verschillende manieren afgebeeld.

1. de absolute fout $|x_n - \alpha_i^n|$.
2. de relatieve fout $|x_n - \alpha_i^n|/|\alpha_i^n|$.
3. de log van de absolute fout $\log|x_n - \alpha_i^n|$.
4. de log van de relatieve fout $\log(|x_n - \alpha_i^n|/|\alpha_i^n|)$.

De fout voor α_1 wordt in blauw getekend, die voor α_2 in het rood.

Opgaven en experimenten

De applet is hier te vinden: `NW/NW/stab/java/index.html`.

- Probeer de applet voor een aantal verschillende waarden van α_1 en α_2. Bekijk de verschillende fouten. Op welke van de 2 oplossingen zit de grootste fout, op de kleinste of op de grootste?
- Als de logaritmische fout in functie van n ongeveer een rechte lijn vormt, kun je dan in het algemeen zeggen hoe de fout van n afhangt?
- Welke invloed heeft het feit dat een/beide oplossingen groeit/daalt dus dat een/beide alfa's groter/kleiner is dan 1?
- Welke oplossingen zijn er als de twee alfa's aan elkaar gelijk zijn?
- Van waar komen die fouten? Zijn dit afrondingsfouten?

Extra

Kun je een verklaring geven van het geobserveerde aan de hand van een gedetailleerde foutenanalyse?

Besluit

Als de 2 parameters in de applet verschillen dan is er een "dominante" en een "minimale" oplossing. Door afrondingsfouten zal de dominante oplossing binnensluipen in de berekening van de minimale oplossing. Het is alsof er een perturbatie aan de juiste beginwaarden voor de minimale gegeven wordt, hierdoor krijgt deze een (kleine) component van de dominante mee en die zal gaan overheersen. Daardoor stijgt de fout exponentieel.

3.2.3 Recursief berekenen van integralen

De integralen

Beschouw de volgende 4 integralen

Integraal 1

$$I_n = \int_0^1 x^n e^x dx$$

Deze kan men uitrekenen door partiële integratie. Dit levert

$$I_0 = e - 1 \quad \text{en voor} \quad n = 1, 2, ... \quad I_n = e - nI_{n-1}$$

Integraal 2

$$I_n = \int_0^1 \frac{x^n}{x+5} dx$$

Deze kan men uitrekenen door partiële integratie. Dit levert

$$I_0 = \ln 1.2 \quad \text{en voor} \quad n = 1, 2, ... \quad I_n = \frac{1}{n} - 5I_{n-1}$$

Integraal 3

$$I_n = \int_0^{\pi/2} \cos^n(x) dx$$

Deze kan men uitrekenen door partiële integratie. Dit levert

$$I_0 = \frac{\pi}{2}, \quad I_1 = 1, \quad \text{en voor} \quad n = 2, 3, ... \quad I_n = \frac{n-1}{n} I_{n-2}$$

Integraal 4

$$I_n = \int_{\pi/2}^{\pi} \frac{\sin(x)}{x^n} dx$$

Deze kan men uitrekenen door partiële integratie. Dit levert

$$I_0 = 1, \quad I_1 \approx 0.481174884, \quad I_2 \approx 0.238287033,$$

$$\text{en voor} \quad n = 3, 4, ... \quad I_n = \frac{1}{(n-1)(\frac{\pi}{2})^{n-1}} + \frac{1}{(n-1)(n-2)\pi^{n-2}} - \frac{1}{(n-1)(n-2)} I_{n-2}$$

Verschillende fouten

We vergelijken de resultaten van de recursieve berekening met de echte waarde van de integralen.

De bijhorende fout wordt op verschillende manieren weergegeven:

1. de fout $I_n - \overline{I}_n$.
2. de absolute fout $|I_n - \overline{I}_n|$.

3. de relatieve fout $|I_n - \overline{I}_n|/|I_n|$.
4. de log van de absolute fout $\log|I_n - \overline{I}_n|$.
5. de log van de relatieve fout $\log(|I_n - \overline{I}_n|/|I_n|)$.
6. de verandering van de fout $|I_n - \overline{I}_n|/|I_{n-1} - \overline{I}_{n-1}|$

De applet

Men kan de verschillende bovenstaande integralen kiezen en opgeven hoeveel stappen in de recursie moeten uitgevoerd worden. Men krijgt dan de verschillende fouten te zien.

Voor applet en beschrijving zie `NW/NW/stab/java/index.html`.

Opgaven en experimenten

- Probeer de verschillende mogelijkheden van de applet eens uit. Bekijk de verschillende fouten. Is er een karakteristiek gedrag van de fout (relatieve, absolute,...)?
- Met de wijziging van de beginvoorwaarden kan je kleine perturbaties aanbrengen op deze startwaarden. Hoe gevoelig is de oplossing aan wijzigingen op deze startwaarden?
- Is de recursiebetrekking dan een goed gesteld probleem? of: Is het recursief uitrekenen van de integraal een stabiele manier?
- Is de gevoeligheid aan de beginvoorwaarden dezelfde voor alle 4 de integralen?
- Vergelijk met het andere voorbeeld van een eenvoudige recursiebetrekking die je hier kan vinden. Wat is het verschil met de recursiebetrekkingen die hier gebruikt worden?

Extra

Kun je een verklaring geven van het geobserveerde aan de hand van een gedetailleerde foutenanalyse?

Hoofdstuk 4

Stelsels lineaire vergelijkingen

4.1 De methode van Gauss

Voor het oplossen van stelsels programmeren we de methode van Gauss in matlab.

```
n=4, del=1.e-15;                    % afmeting stelsel + parameter del
a_exact=genmat(n,del), a=a_exact;   % genereer random matrix
x_exact=ones(n,1), x=x_exact;       % de exacte oplossing
b_exact=a_exact*x_exact, b=b_exact; % genereer rechterlid
ab=[a,b]                            % de uitgebreide matrix

% de eliminatie van gauss
for i=1:n
  for j=i+1:n
    disp(['[i,j]= [',int2str(i),',',int2str(j),']'])
    l(j,i)=ab(j,i)/ab(i,i),
    ab(j,:)=ab(j,:)-l(j,i)*ab(i,:),
  end
end
l(n,n)=0; l=l+diag(ones(n,1))

AB_min_LR=[a_exact,b_exact]-l*ab    % [a b] = L * R

% terugsubstitutie
ab(n,n+1)=ab(n,n+1)/ab(n,n)
for i=n-1:-1:1
  ab(i,n+1) = ( ab(i,n+1)-ab(i,i+1:n)*ab(i+1:n,n+1) ) / ab(i,i)
end

% relatieve fout op resultaat
norm(ab(:,n+1)-x_exact)/norm(x_exact)
```

Je hebt er wel een hulpprogramma `genmat.m` bij nodig dat een willekeurige matrix genereert. Het programma "kiest" een oplossingsvector $x = [1, 1, \ldots, 1]^T$ en genereert een bijhorend rechterlid.

```
function a=genmat(n,del)
a=rand(n);
if (del > 0), a(1,2)=a(2,2)*a(1,1)/a(2,1)*(1+del); end
```

Daarna wordt het stelsel opgelost.

4.1.1 Opgaven

De matlabprogramma's kan je vinden op `NW/NW/gauss/matlab/index.html`.

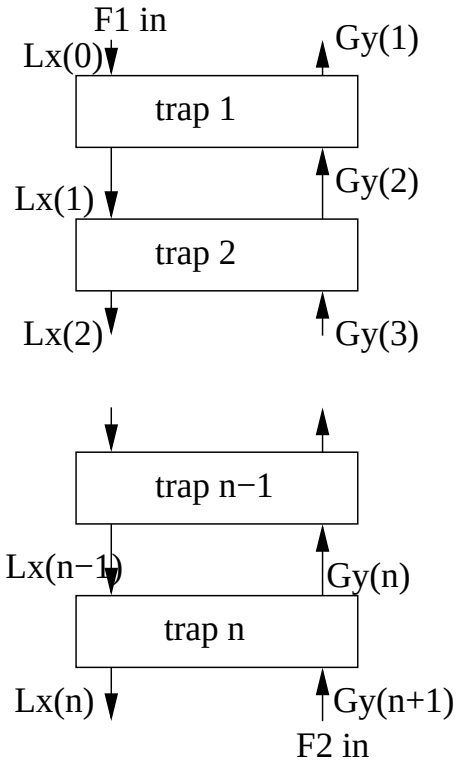
- Laat dit programma lopen met `del=0`. Geeft dit bevredigende resultaten?
- Hoe hebben we experimenteel de nauwkeurigheid van de berekende LR-ontbinding van de matrix getest?
- Laat het programma nu lopen met `del=1.e-15`. Hoe zijn nu de resultaten?
- Heb je hiervoor een verklaring? Kijk naar de grootte van de spilelementen.
- Ga na wat er in het programma `genmat.m` gebeurt.
- Kan je de grootte van de fout op `x` voorspellen aan de hand van de grootte van `del`?
- Programmeer nu de methode van Gauss met optimale pivotering. Probeer meteen ook de multiplicatoren op de nul-posities van de matrix te bewaren in plaats van in een afzonderlijke matrix.
- Geeft het programma nu bevredigende resultaten?
- Hoe bewaar je best de permutatiematrix?
- Waarom mag men nu de eliminatie niet op de ganse rij van de uitgebreide matrix toepassen?
- Er is een matlabprogramma `gaussp.m` gegeven.
- Hoe hebben we het vorige in het programma gerealiseerd?
- Hoe hebben we nu experimenteel nagegaan dat $PA = LR$?
- Hoe hebben we de matrices L en R uit de uitgebreide matrix $[A|B]$ gehaald? (Gebruik eventueel de matlab `help`.)
- Is er nu nog een verband tussen `del` en de relatieve fout op het resultaat?
- Pas het programma aan zodat het ook de determinant van de matrix A berekent.
- Pas het programma aan voor meerdere rechterleden (bv. om de inverse van A te berekenen).
- Pas het programma aan zodat het aantal bewerkingen geteld wordt. Klopt dit aantal met het theoretisch berekend aantal in de cursus? Zoniet, hoe is dit dan te verklaren?
- Het programma `gaussh.m` is een copie van het vorige, zonder het uitschrijven van tussenresultaten. De gegenereerde matrix is echter een hilbertmatrix (gemaakt met het matlab commando `hilb(n)` met $n = 10$).
- Voer het programma `gaussh.m` uit. Hoe nauwkeurig zijn de resultaten nu? Is dit nog steeds door een instabiliteit in de methode van Gauss veroorzaakt, of is er nu een andere reden?
  Doe `help cond` en bereken het conditiegetal van de matrix A.
- Verklaart dit de nauwkeurigheid van de resultaten?
- Is er een verband tussen `cond(A)` en de relatieve fout op de oplossing?
- Voor experimenten met het conditiegetal van de Hilbertmatrix is er ook een applet gemaakt zie `NW/NW/gauss/index.html`.

4.2 Voorbeeld van massascheiding

Voor speciale stelsels kan men de klassieke methoden aanpassen. Het volgende voorbeeld geeft aanleiding tot een tridiagonaal stelsel.

4.2.1 Beschrijving van het model

In een fluïdum F_1 is een stof S opgelost (bv. nicotine opgelost in water). Om de stof eruit te halen wordt een ander fluïdum F_2 in tegenstroom gebracht. Die tegenstroom neemt een deel van S op uit F_1 zodat F_1 gezuiverd wordt. Schematisch kan men dit voorstellen als een opeenvolging van verschillende trappen waarin de uitwisseling gebeurt.



In elke trap is er een zeker evenwich: Bijvoorbeeld in trap i is de concentratie in F_1 gegeven door x_i en de concentratie in F_2 door y_i. Evenwicht in trap i betekent dat er een constante K is zodat $y_i = Kx_i$. Zij verder L het debiet van F_1 en G het debiet van F_2. De hoeveelheid van S die trap i uitstroomt via F_1 is Lx_i en de hoeveelheid S die via F_2 trap i uitstroomt is Gy_i. Dus als hx_i en gy_i de hoeveelheden zijn die in trap i aanwezig zijn in de stroom F_1 en F_2 respectievelijk, dan moet

$$h\frac{dx_i}{dt} + g\frac{dy_i}{dt} = Lx_{i-1} + Gy_{i+1} - Lx_i - Gy_i, \quad i = 1, \ldots, n.$$

Als we stellen $y_i = Kx_i$, dan hangt de vergelijking enkel van de x_i af, namelijk

$$\frac{d\mathbf{x}}{d\tau} = A\mathbf{x} + \mathbf{b},$$

met $\tau = t/(h + gK)$ en $\mathbf{x} = [x_1, \ldots, x_n]^T$, en $\mathbf{b} = [Lx_0, 0, \ldots, 0, Gy_{n+1}]^T$ en

$$A = \begin{bmatrix} -(L+GK) & GK & & & \\ L & -(L+GK) & GK & & \\ & L & \ddots & \ddots & \\ & & \ddots & \ddots & GK \\ & & & L & -(L+GK) \end{bmatrix}$$

In stationaire toestand is $\frac{d\mathbf{x}}{d\tau} = 0$ en is dus $A\mathbf{x} = -\mathbf{b}$.

4.2.2 Mogelijke problemen om op te lossen

- Als x_n opgelegd wordt, hoeveel trappen heeft men dan nodig?
- Welke zijn de waarden voor x_i en y_i in stationaire toestand?
- Is de stationaire toestand stabiel? (eigenwaarden van A kleiner dan 1)
- Welk is het overgangsregime?
- Kan er een cyclus optreden?

4.2.3 Voorbeeld van gegevens

Beschouwen we het voorbeeld van nicotine in water en laten we als tegenstroom kerosine gebruiken. Dan gelden volgende waarden:

Voor het water geldt $L = 126$ kg/h,
Voor de kerosine geldt $G = 144$ kg/h
Stel er is 1% nicotine aanwezig in het water.
Op kamertemperatuur is de verdunningscoëfficiënt

$$K = 0.876 = \frac{\text{gewichtsfractie nicotine in kerosine}}{\text{gewichtsfractie nicotine in water}}.$$

Laten we ons als doel stellen om $x_n < 0.001$ te krijgen.
Hoe groot is dan n?

4.2.4 Opgaven

Een matlabprogramms kan je vinden op `NW/NW/absorb/matlab/index.html`

- Hoeveel trappen zijn er nodig?
- Wat kun je besluiten uit de berekende eigenwaarden van de matrix A?
- Hoe wordt in het programma het stelsel opgelost?
  Kan je een efficiëntere methode bedenken?
  Bijvoorbeeld door enkel de 3 diagonalen te bewaren van de matrix.
  Zelfs dat is niet echt nodig: enkel de 3 getallen die op de diagonalen staan zijn nodig.
  Schrijf een aangepaste versie om dit speciale tridiagonale stelsel op te lossen. Hoeveel geheugenplaatsen (vectoren) heb je daarbij nodig?

4.2.5 Extra

- Kan je ook de andere hierboven gesuggereerde problemen in verband met dit model oplossen?

Opmerking: Voor beperkte problemen (orde 10 bv.) heeft deze efficiënte implementatie niet veel zin. Er zijn echter problemen die aanleiding geven tot gelijkaardige stelsels. Bijvoorbeeld bij omkeerbare anionische polimerizatie. De variabelen hebben een andere betekenis, maar de vorm van de vergelijkingen en dus van de matrix A is gelijkaardig, alleen zijn het nu in principe oneindig veel vergelijkingen. Men zal dat dan benaderen door er heel veel te nemen, en dan wordt de efficiëntie wel belangrijk.

Hoofdstuk 5

Veeltermbenadering

5.1 Kleinste-kwadratenbenadering

5.1.1 Probleemstelling

We beschouwen het volgende probleem:
Gegeven is een aantal meetpunten (x_i, f_i), $i = 0, \ldots, N$.
Gevraagd is "zo goed als mogelijk" een veelterm van graad n door deze punten te trekken.

Stel de veelterm is

$$y_n(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_n.$$

Als hij door alle punten moet gaan, zal hij moeten voldoen aan

$$y_n(x_i) \equiv a_0 x_i^n + a_1 x_i^{n-1} + \cdots + a_n = f_i, \quad i = 0, \ldots, N$$

Indien $N = n$, dan is dit een vierkant stelsel $AX = B$ met

$$A = \begin{bmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_N^n & x_N^{n-1} & \cdots & x_N & 1 \end{bmatrix}, \quad X = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad B = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{bmatrix}.$$

De matrix van het stelsel is een matrix van Vandermonde, en is dus inverteerbaar en dus heeft het stelsel juist één oplossing. Dit is **de** interpolerende veelterm.

In de meeste praktische toepassingen echter is $N \gg n$.

Bovendien zullen er op de gegevens f_i (kleine) meetfouten zitten.

Het kan dan niet de bedoeling zijn om een veelterm van een zeer hoge graad N door al deze foutieve punten te trekken. Men zal dan een kleinste-kwadratenoplossing zoeken.

5.1.2 Kleinste-kwadratenoplossing

Uit de cursus algebra van de 1e kan weten we dat dit op verschillende manieren kan gevonden worden

1. Los het normaalstelsel op: $A^T A\ X = A^T\ B$.
2. Bepaal (met Gram-Schmidt) de QR ontbinding van A ($A = QR$) en los een klein driehoekig stelsel $RX = Q^T B$ op.

3. Gebruik de SWO: $A = U\Sigma V^T$ en stel $X = A^+B = V\Sigma^+U^TB$.

De eerste twee methoden werken goed als de kolommen van A onafhankelijk zijn. D.w.z. A is van volle (= maximale) rang(waarom?).

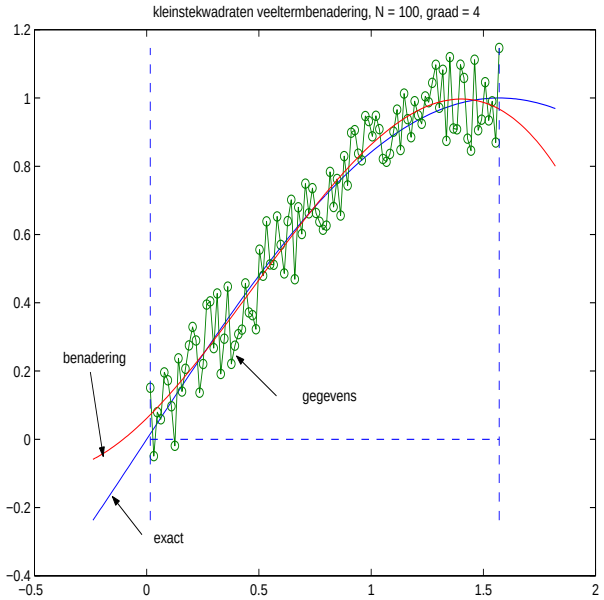
Een numeriek aspect:

Toon aan dat het conditiegetal $\kappa(A^TA) = \kappa(A)^2$ maar $\kappa(A) = \kappa(R)$.

Opmerking: het conditiegetal van een niet vierkante matrix A wordt gedefinieerd als $\kappa(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$, waarbij $\sigma_{\max}$ en $\sigma_{\min}$ de maximale en minimale singuliere waarde van een matrix zijn.
Dus, als $\kappa(A) = 10^8$, dan is $\kappa(A^TA) = 10^{16}$. Het eerste betekent dat men kan verwachten dat men bij de oplossing ongeveer 8 juiste beduidende cijfers zal verliezen; in het tweede geval 16.

Hoeveel cijfers houdt men dan nog over als men in IEEE dubbele nauwkeurigheid rekent?

Een voorbeeld van $f_i = \sin(\frac{\pi i}{2N}) + u_i$, $i = 1, \ldots, N = 100$ is hieronder gegeven. Hierin is u_i een willekeurig uniform verdeelde fout uit $[-0.15, 0.15]$. De exacte functie en de kleinste-kwadratenveeltermbenadering van graad $n = 4$ zijn ook getekend.



In dit voorbeeld is $\kappa(A^TA) = 2.5492e + 05$ en $\kappa(A) = 504.8937$.

Bemerk dat de benadering goed is in het benaderingsinterval, maar dat de fout sterk toeneemt buiten het interval.

5.1.3 Experimenten

Je kan in matlab het commando `census` intikken. Dit geeft als resultaat een voorbeeldprogramma waarbij gegevens over de bevolkingsaantallen in de VS worden benaderd door een veelterm (of een andere functie). Merk op hoe slecht de interpolerende veelterm voorspelt!

5.1.4 Extra

Schrijf zelf een matlab programma dat met SWO `[U,S,V]=svd(A)` of met QR ontbinding `[Q,R]=qr(A)`, of via het normaalstelsel, de kleinste-kwadratenveeltermbenadering opstelt. De matrix A kan je opstellen met

```
A=ones(N,1);
for i=1:n-1
 A=[x'.^i,A];
end
```

Bereken $\kappa(A^TA)$ en $\kappa(A)$: `cond(A'*A), cond(A)`
Voor de veeltermevaluatie gebruik je: `polyval(p,x)`.

5.2 Veelterminterpolatie met Lagrange

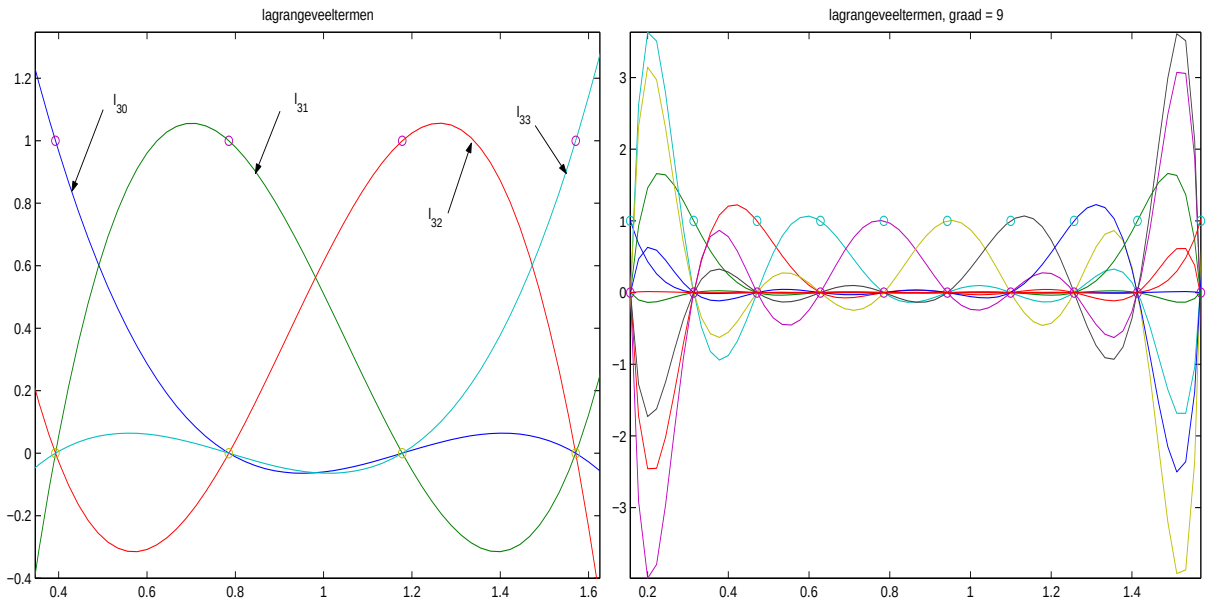
5.2.1 De veelterm

Men schrijft de interpolerende veelterm voor $(x_i, f_i)_{i=0}^n$ als

$$y_n(x) = \sum_{k=0}^{n} f_k \ell_{nk}(x)$$

met ℓ_{nk} de Lagrangeveeltermen die voldoen aan

$$\ell_{nk}(x_i) = \delta_{ki}, \quad i, k = 0, \ldots, n.$$



5.2.2 Experimenten

De matlab programma's op `NW/NW/veelterm/matlab/index.html` laten toe wat te experimenteren:

- Voer het programma uit voor verschillende waarden van de interpolatiegraad n. Bekijk het gedrag van de fout binnen en buiten het interpolatieinterval. Wat valt er op? Opmerking: Met de zoom-knop van het figuur-venster kan je op een tekening inzoemen.
- Probeer ook eens met andere functiewaarden en andere x-waarden. Is het gedrag steeds hetzelfde? Welke zijn de typische kenmerken van de fout? Vergelijk met de bespreking van de fout in de cursus.
- Bekijk ook eens het verloop van de Lagrangeveeltermen ℓ_{nl}. Hoe is hun verloop binnen en buiten het interval? Wat gebeurt er voor hoge graad n? Dit impliceert dat voor grote n, deze basis erg onstabiel wordt. Hoe verklaar je dat?

5.2.3 Extra

- Denk je dat het mogelijk is om door de interpolatiepunten niet equidistant te nemen, de fout aan de randen van het interval te verkleinen? Probeer voor het interval $[a, b]$ eens de punten $x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(t_i)$ met t_i equidistant in $[0, \pi]$. Men noemt dit de Chebyshevpunten.

5.2.4 Een java applet

- Je kan een java applet gebruiken van de universiteit van Utrecht. Te vinden op http://www.cut-the-knot.org/Curriculum/Calculus/LagrangeInterpolation.shtml. Hierbij kan je een aantal punten verplaatsen en de Lagrange interpolerende veelterm wordt berekend en getekend. Zie ook

  http://www.gris.uni-tuebingen.de/projects/grdev/applets/lagrange/EnglishApplet.html.

  voor een andere applet voor Lagrange interoplatie. Je kan zelf op zoek gaan naar nog andere.

5.3 Veeltermінterpolatie met Newton

5.3.1 De veelterm

Men schrijft de interpolerende veelterm voor $(x_i, f_i)_{i=0}^n$ als

$$y_n(x) = \sum_{k=0}^{n} b_k \phi_k(x)$$

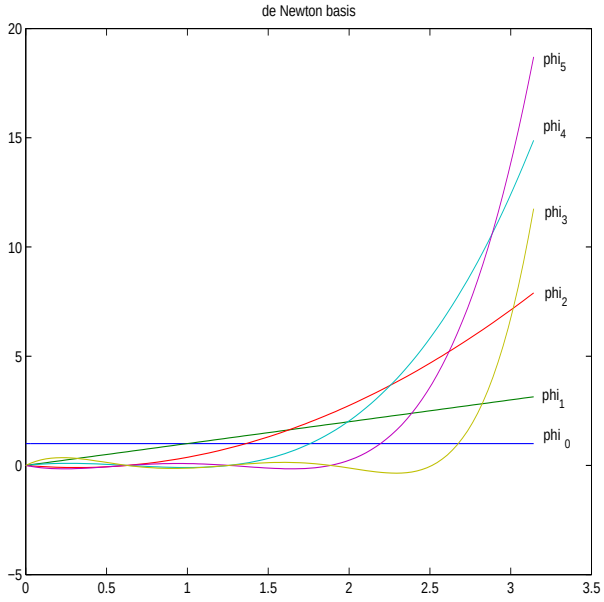
met ϕ_k de Newtonveeltermen die gegeven zijn door

$$\phi_k(x) = (x - x_0) \cdots (x - x_{k-1}).$$

De coëfficiënten b_k zijn gedeelde differenties: $b_k = f[x_0, \ldots, x_k]$ met

$$f[x_k] = f_k; \quad \text{en}$$

$$f[x_0, \ldots, x_{k-1}, x_k, x_j] = \frac{f[x_0, \ldots, x_{k-1}, x_j] - f[x_0, \ldots, x_{k-1}, x_k]}{x_j - x_k}.$$



5.3.2 Experimenten

De matlab programma's op `NW/NW/veelterm/matlab/index.html` laten toe wat te experimenteren:

- Het matlabprogramma voor Newton interpolatie is zodanig geschreven dat voor één interpolatiegraad een grafiek verschijnt van de functie en de interpolant, van de interpolatiefout en van de Newton-basis.
  Heeft de Newton-basis een gedrag dat analoog is aan dat van de Lagrange-basis? Is de Newton-basis even onstabiel als de Lagrange-basis?
  Indien men verscheidene graden opgeeft (bv. `2:30`), dan wordt de norm van de relatieve fout uitgezet in functie van de graad.
- Kijk na wat er gebeurt voor de functie `atan` (de boogtangens) met als interval `[-10,10]`. Wat valt er op?
  Herhaal dit voor het interval `[-1,1]`.
  Wat is het verschil met het vorige interval? Welke verklaring wordt daarvoor in de cursus gegeven?
- Zal een veelterm die interpoleert in equidistante punten convergeren naar de echte functie als het aantal interpolatiepunten toeneemt? Voer een paar experimenten uit

op zacht-verlopende functies om dit experimenteel na te gaan. Bv. `sin` in het interval `[0,pi/2]`. Bewijs dat voor dit voorbeeld de interpolatiefout theoretisch naar nul convergeert als de interpolatiegraad naar oneindig gaat.

- Lost dit programma het coëfficiëntenprobleem of het waardeprobleem op? Is het de meest efficiënte implementatie?
  Welk schema is er gebruikt om de gedeelde differenties te berekenen?

5.3.3 Extra

- De coëfficiënten van de interpolerende veelterm hangen af van de gekozen basis. Dus, de conditie van het coëfficiëntenprobleem moet afhangen van de gekozen basis. Wat zou het conditiegetal zijn voor het coëfficiëntenprobleem als je de basis $\{1, x, x^2, \ldots, x^n\}$ neemt? Wat wordt dit voor de Newton-basis, wat voor de Lagrange-basis, en voor een willekeurige basis?
  Als de interpolatiepunten zeer dicht bij elkaar liggen is de interpolerende veelterm niet goed bepaald. Kan je dat ook zien aan het conditiegetal?
- Zou men iets kunnen zeggen over de stabiliteit van de methode van Newton of Lagrange voor het waardeprobleem van interpolatie?

Hoofdstuk 6

Numerieke differentiatie

6.1 Theorie en praktijk

We weten dat in theorie

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} = \lim_{h\to 0} \frac{f(x+h) - f(x-h)}{2h} = \lim_{h\to 0} \frac{f(x) - f(x-h)}{h}.$$

De tellers van de uitdrukkingen waarvan men de limiet neemt komen overeen met voorwaartse differenties, centrale differenties en achterwaartse differenties.

Hoewel dit in theorie allemaal dezelfde limiet oplevert zal men in de praktijk een eindige benadering moeten nemen en h klein laten worden maar nooit helemaal 0. Daarbij ontstaan afbrekingsfouten (de differentiatiefout) en afrondingsfouten.

We berekenen de afgeleide van de functie $f(x) = \sin(x)$ in het punt $a = 1$. We doen dit door voorwaartse differenties en centrale differenties uit te rekenen.

De voorwaartse differentie is:

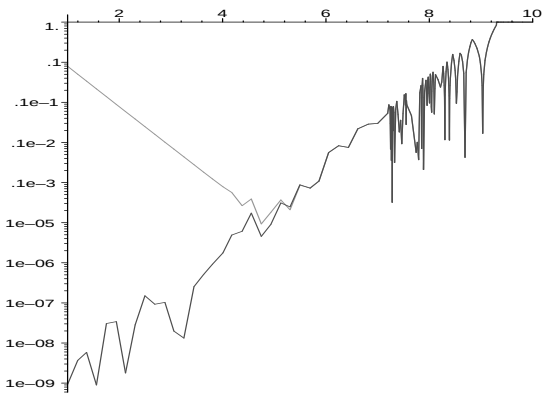
```
> vdiff:=(f,a,h)->(f(a+h)-f(a))/h;
```

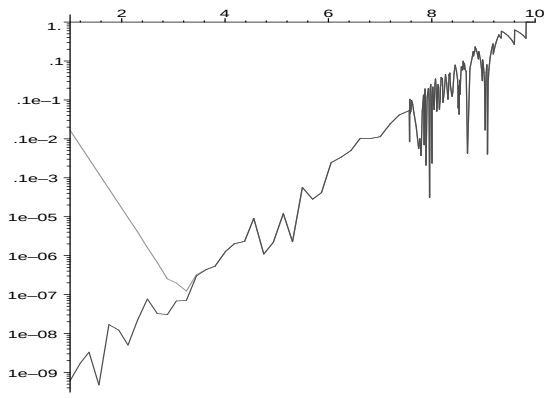
De centrale differentie is:

```
> cdiff:=(f,a,h)->(f(a+h)-f(a-h))/(2*h);
```

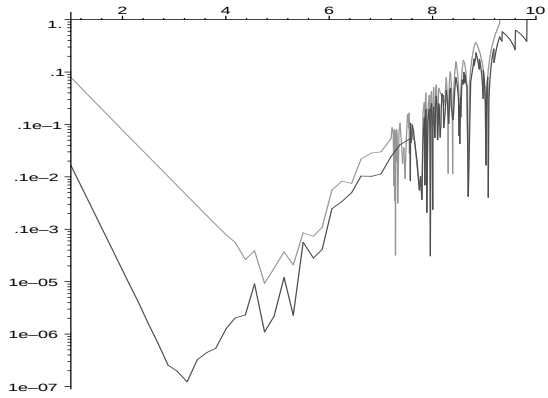
We berekenen dit met 10 cijfers nauwkeurig.
De volgende figuur links geeft in functie van i als $h = 10^{-i}$:

- de afrondingsfout bij het berekenen van de voorwaartse differenties (onderste curve)
- en de totale fout (afronding+discretisatie= exacte waarde − de in lage nauwkeurigheid berekende waarde van de afgeleide) (bovenste curve)





Rechts staat dezelfde figuur maar dan met centrale differenties. De volgende figuur tekent de twee onderste curven van hierboven op één tekening:



De bovenste is de totale fout voor voorwaartse differenties.
De onderste is de totale fout voor de centrale differenties.

6.1.1 Opgaven en experimenten

- Hoe bereken je de afrondingsfout en de discretisatiefout afzonderlijk? (Zie maple code.)
- Kan je precies verklaren waarom op de twee figuren de fout verloopt zoals ze verloopt? Dit wil zeggen, welke is de absolute/relatieve nauwkeurigheid die maximaal bereikt wordt?
- Kan je verklaren waarom dit de maximale nauwkeurigheid is die kan bereikt worden?
- Waarom wordt die bereikt bij die waarde van i?
  Je kan eventueel zelf experimenteren met andere functies en andere nauwkeurigheden door met het maple programma te spelen. Je kan dit vinden op
  `NW/NW/numdif/maple/index.html`.
  Denk erom, maple werkt (onder zekere omstandigheden) intern met decimale getallen.

Hoofdstuk 7

De warmtevergelijking

We laten zie hoe met eenvoudige differentieschema's en een methode voor het oplossen van een stelsel de 1-dimensionale warmtevergelijking kan opgelost worden.

7.1 Beschrijving van het probleem

We hebben dit probleem reeds vroeger beschouwd als een bijzonder geval van een convectie-diffusievergelijking.

Stel dat we een staaf beschouwen die aan de linkerkant ($x = 0$) tegen een warm medium wordt gehouden en aan de rechterkant ($x = L$) op een constante temperatuur $T = 0$ wordt gehouden.

Het is de bedoeling om de temperatuursverdeling $T(x, t)$ in de staaf ($0 \leq x \leq L$) in de loop van de tijd ($t \geq 0$) te berekenen.

De temperatuur $T(x, t)$ voldoet aan de warmtevergelijking

$$\alpha \frac{\partial^2 T(x,t)}{\partial x^2} = \frac{\partial T(x,t)}{\partial t}.$$

Beginvoorwaarde: Op het ogenblik $t = 0$ veronderstellen we dat de temperatuur van de gehele staaf gelijk is aan 0: $T(x, 0) = 0$

Randvoorwaarden:

1. De temperatuur aan het uiteinde is constant gelijk aan 0: $T(L, t) = 0$
2. De warmteuitwisseling bij het contactpunt wordt beschreven door

$$\frac{\partial T}{\partial x}(0, t) = \frac{H}{K}(T(0, t) - T_{\text{omgeving}})$$

Hierin is H de convectieve warmteoverdrachtscoëfficiënt en K de warmtegeleidingscoëfficiënt. α is de diffusiviteit.

Voor meer uitleg zie vroeger.

7.2 Discretisatie

We willen nu de temperatuur berekenen in een aantal discrete punten van een (x,t)-rooster. Stel dat we $T(x,t)$ berekenen voor $x = x_i = ih_x$, voor $i = 0, 1, \ldots, n$, waarbij $L = nh_x$ en h_x is een constante pas in de x-richting. In de tijd nemen we een constant tijdsinterval h_t en berekenen we de temperatuur voor $t = t_j = jh_t$ met $j = 0, 1, 2, \ldots$.

Stellen we nu $T(x_i, t_j) = u_i^j$.

We hebben bij het voorbeeld van de transportvergelijking reeds een aantal naïeve manieren van discretisatie gezien (voor- of achterwaartse differenties). Na onze opgedane kennis weten we dat we beter centrale differenties gebruiken omdat die nauwkeuriger zijn. Daarbij hebben we functiewaarden nodig die niet op het rooster vallen, maar die kunnen we dan berekenen door een eenvoudige interpolatie.

7.2.1 Discretisatie van de afgeleide naar t

De afgeleide in het rechterlid in het roosterpunt $(i, j + 1/2)$ benaderen we met behulp van een centrale differentie:

$$\left(\frac{\partial u}{\partial t}\right)_i^{j+1/2} = \frac{u_i^{j+1} - u_i^j}{h_t}.$$

7.2.2 Discretisatie van de afgeleide naar x

Voor de partiële afgeleide in het linkerlid benaderen we eveneens met behulp van een tweede orde centrale differentie.

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i^{j+1/2} = \frac{u_{i+1}^{j+1/2} - 2u_i^{j+1/2} + u_{i-1}^{j+1/2}}{h_x^2}.$$

7.2.3 Interpolatie

We hebben de halve index $j + 1/2$ genomen omdat dan in de eerste formule rechts gehele j's voorkomen. Voor het rechterlid van de tweede formule hebben we de waarden met halve j-index nog steeds. Die benaderen we door lineaire interpolatie als het gemiddelde:

$$u_i^{j+1/2} = \frac{1}{2}(u_i^{j+1} + u_i^j).$$

7.2.4 Gediscretiseerde differentiaalvergelijking

Door nu de laatste formule in de voorlaatste in te vullen en dan alles als benadering in de differentiaalvergelijking in te vullen verkrijgen we

$$\alpha\left(\frac{u_{i+1}^{j+1} + u_{i+1}^j - 2(u_i^{j+1} + u_i^j) + u_{i-1}^{j+1} + u_{i-1}^j}{2h_x^2}\right) = \frac{u_i^{j+1} - u_i^j}{h_t}.$$

Als we dit vermenigvuldigen met h_t en $\lambda = \frac{h_t}{h_x^2}$ stellen, dan wordt dit

$$-\frac{\alpha\lambda}{2}u_{i-1}^{j+1} + (1 + \alpha\lambda)u_i^{j+1} - \frac{\alpha\lambda}{2}u_{i+1}^{j+1} = \frac{\alpha\lambda}{2}u_{i-1}^j + (1 - \alpha\lambda)u_i^j + \frac{\alpha\lambda}{2}u_{i+1}^j$$

We moeten dit bekijken voor $i = 0, 1, \ldots, n-1$ (voor $i = n$ is de temperatuur gegeven), en voor $j = 0, 1, 2, \ldots$

Merk op dat er een probleem is voor $i = 0$ want dan komt u_{-1}^j voor. We lossen dit op met de randvoorwaarde (zie onder).

7.2.5 Progressie in de tijd

Als de temperatuur voor $t = t_j$ gekend is (d.w.z., we kennen u_i^j), kunnen we met deze betrekkingen de temperatuur op tijdstip $t = t_{j+1}$ (d.w.z., u_i^{j+1}) berekenen. Dus we veronderstellen alles met index j gekend, en berekenen alles met index $j+1$ door een (tridiagonaal) stelsel op te lossen.

7.2.6 Beginvoorwaarde

Op het ogenblik $t = 0$ is de temperatuur in de ganse staaf $T = 0$, dus voor $j = 0$ en voor alle $i = 0, 1, \ldots, n$ hebben we $u_i^0 = 0$.

7.2.7 Randvoorwaarden

Einde van de staaf: Vermits we voor $i = n$ (op het einde van de staaf) de temperatuur kennen (die is 0), zullen we daarmee rekening moeten houden in deze betrekkingen.
Begin van de staaf: De convectieve randvoorwaarde moet ook gediscretiseerd worden. We gebruiken hiervoor weer centrale differenties:

$$\left(\frac{\partial u}{\partial x}\right)_0^j = \frac{1}{2h_x}(u_1^j - u_{-1}^j)$$

Dus

$$u_{-1}^j = u_1^j - 2h_x p(u_0^j - T_{\text{omgeving}}), \quad p = \frac{H}{K}.$$

Hieruit halen we u_{-1}^j die we nodig hebben in de bovenstaande vergelijking voor $i = 0$. Dit levert namelijk als eerste vergelijking

$$(1 + \alpha\lambda(1 + ph_x))u_0^{j+1} - \alpha\lambda u_1^{j+1} = (1 - \alpha\lambda - \alpha p h_x \lambda)u_0^j + \alpha\lambda u_1^j + 2\alpha\lambda h_x p T_{\text{omgeving}}.$$

7.3 Stelsels

Uiteindelijk zal de overgang van u_i^j naar u_i^{j+1} gebeuren door het oplossen van een stelsel $AX = B$ waarin

$$A = \begin{bmatrix} 1+\alpha\lambda(1+ph_x) & -\alpha\lambda & & & & \\ -\frac{\alpha\lambda}{2} & 1+\alpha\lambda & -\frac{\alpha\lambda}{2} & & & \\ & -\frac{\alpha\lambda}{2} & 1+\alpha\lambda & -\frac{\alpha\lambda}{2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & -\frac{\alpha\lambda}{2} \\ & & & & -\frac{\alpha\lambda}{2} & 1+\alpha\lambda \end{bmatrix}, \quad X = \begin{bmatrix} u_0^{j+1} \\ u_1^{j+1} \\ u_2^{j+1} \\ \vdots \\ u_{n-2}^{j+1} \\ u_{n-1}^{j+1} \end{bmatrix}$$

$$B = \begin{bmatrix} (1-\alpha\lambda-\alpha p h_x\lambda)u_0^j + \alpha\lambda u_1^j + 2\alpha\lambda h_x p T_{\text{omgeving}} \\ \frac{\alpha\lambda}{2}u_0^j + (1-\alpha\lambda)u_1^j + \frac{\alpha\lambda}{2}u_2^j \\ \frac{\alpha\lambda}{2}u_1^j + (1-\alpha\lambda)u_2^j + \frac{\alpha\lambda}{2}u_3^j \\ \vdots \\ \frac{\alpha\lambda}{2}u_{n-2}^j + (1-\alpha\lambda)u_{n-1}^j + \alpha\lambda u_n^j \end{bmatrix}$$

Dit is een probleem waarbij we verschillende stelsels moeten oplossen, allemaal met dezelfde matrix A, maar voor verschillende rechterleden B, (B hangt immers af van j), en we kennen niet alle rechterleden vooraf. We moeten immers het vorige stelsel oplossen naar de $u_{i,j}$ om het nieuwe rechterlid te berekenen.

Zoals we gezien hebben, kan dit het meest efficiënt opgelost worden door één keer de LU ontbinding van de matrix te maken en dan telkens twee driehoekige stelsels op te lossen in elke iteratiestap voor j.

De matrix is niet symmetrisch door de convectieve randvoorwaarde. Als men als randvoorwaarde opgeeft dat voor $i = 0$ (in het begin van de staaf) de temperatuur vast op 100 graden gehouden wordt, is het stelsel wel symmetrisch.

7.4 Matlab experimenten (extra)

Programmeer de methodes in matlab, of gebruik de voorgeprogrammeerde methodes van `NW/NW/wrmvgl/matlab/index.html`.
om een aantal experimenten te doen.

- Als je de voorgeprogrammeerde code gebruikt probeer dan daarin te herkennen wat hiervoor is uitgelegd. Is in het programma rekening gehouden met de speciale tridiagonale structuur van de stelsels?
- Ga na, voor het symmetrische geval, of het gebruik van Cholesky ipv gewone Gauss een tijdwinst oplevert.
- Ga na dat de matrix van het symmetrische stelsel positief definiet is. In dit geval werkt Cholesky gegarandeerd. Er is dan ook geen pivotering nodig.
- Heeft het wijzigen van h_x of h_t een invloed op de nauwkeurigheid van de berekeningen? Voor kleine h_x worden de stelsels heel groot. Hoe lang wil je op de oplossing wachten?
- Wijzig de parameters van het probleem (H, K, T_{omgeving}, L, α). Kan je de oplossing voorspellen voor het programma uitgevoerd wordt? Klopt de oplossing die berekend wordt ook met je intuïtie?

Op het web is een java-applet (zie www.cs.utah.edu/~zachary/isp/applets/Plotter/Plotter.html) te vinden die hoort bij het boek van J.L. Zachary *Introduction to Scientific Programming* waar ongeveer hetzelfde probleem gedemonstreerd wordt.

Hoofdstuk 8

Numerieke integratie

8.1 Integratieregels

8.1.1 Enkelvoudige regels

De enkelvoudige regels ontstaan door een interpolerende veelterm te integreren.
De eenvoudigste zijn de Newton-Cotes formules die met equidistante punten werken.
De voornaamste kenmerken zijn:

1. Voor een toenemend aantal punten groeien de gewichten in absolute waarde (numerieke onstabiliteit)

2. De integratiefout is evenredig met

   - $f^{(n+1)}$ of $f^{(n+2)}$.
   - $[(b-a)/n]^{n+2}$ of $[(b-a)/n]^{n+3}$ vermenigvuldigd met $1/[n(\log n)^2]$.
     De tweede factor gaat traag naar nul. De eerste factor gaat snel naar nul als n toeneemt, tenzij $(b-a)$ groot is. Het is dus best mogelijk dat daarom de fout niet naar nul gaat.

3. De regels zijn niet erg geschikt voor automatische integratie.

8.1.2 Samengestelde regels

Deze ontstaan door het interval op te delen in deelintervallen en in elk deelinterval een eenvoudige Newton-Cotes formule te gebruiken.
De eenvoudigste zijn de middelpuntsregel, trapeziumregel en Simpsonregel.
De voornaamste kenmerken zijn :

1. De gewichten zijn klein en van hetzelfde teken (betere stabiliteit)

2. De integratiefout is evenredig met

   - f'' of $f^{(4)}$, zodat dit niet afhangt van n en dus zal dit de convergentie niet beletten.
   - $1/n^2$ of $1/n^4$ waardoor het gegarandeerd zal convergeren (op afrondingsfouten na). Dit is traag, maar snel genoeg.

3. Deze regels lenen zich wel goed tot het schrijven van automatische integratoren.

8.2 Computer experimenten en opgaven

8.2.1 Maple

Maple worksheets voor het opstellen van Newton-Cotes formules en hun fout en voor het vergelijken van de fout bij verschillende integratieregels zijn te vinden op `NW/NW/numint/maple/index.html`.

Hoe wordt de onstabiliteit van de Newton-Cotes formules geïllustreerd?

8.2.2 Matlab

Matlab code kan je vinden op `NW/NW/numint/matlab/index.html`.

Je kan de trapeziumregel en de Simpsonregel vergelijken. Er wordt ook een visualisatie gegeven van de benaderingen. Er is ook een programma voor automatische integratie met trapeziumregel en Simpsonregel. De convergentiesnelheid wordt ook vergeleken met de trapeziumregel en de regel van Simpson.

Kan je de theoretische convergentie ook terugvinden in het verloop van de fout?

8.2.3 Java applet

Een java applet is beschikbaar op de webpagina's van de cursus. Je kan er de functie opgeven, het aantal deelintervallen (tussen 1 en 10) en het aantal interpolatiepunten (tussen 2 en 10) per deelinterval.

Er zijn echter heel wat applets over integratie te vinden op het web.

Zoek eens een aantal applets door in een zoekmachine te zoeken met de sleutelwoorden "numerieke integratie applet" of "numerical quadrature applet".

8.2.4 Opgaven

- Welk stopcriterium wordt er in de automatische integratoren gebruikt?
- Is er een verschil in rekentijd voor de verschillende integratoren als men steeds eenzelfde aantal functie-evaluaties doet?
- Hoe zou een automatische integrator voor de middelpuntsregel moeten geschreven worden?
- In het maple werkblad worden alle integralen berekend voor n=2:2:N. Waarom doet men dat met stappen van 2?
- De rekentijd voor het maple werkblad kan groot zijn als men N groot kiest. Zou men dezelfde berekeningen efficiënter kunnen doen?
- Bereken de 2-norm van de gewichten-vector die hoort bij de n-de Newton-Cotes formule.
- Hoe verloopt die norm in functie van n?
- Wat kun je hieruit besluiten voor de numerieke stabiliteit van deze methode?

- Probeer eens andere functies en andere intervallen te kiezen in het maple werkblad.
- Hoe goed of slecht is de convergentie van de verschillende methoden?
- Zal de trapeziumregel en de Simpsonregel ook last krijgen van afrondingsfouten als het aantal punten (n) zeer groot wordt?
- Zal het zo zijn dat voor n te groot deze regels een divergerend gedrag gaan vertonen zoals de Newton-Cotes regels dat reeds voor lage n doen?
- Probeer eens een sterk oscillerende integrand te kiezen of een integrand die een singulariteit heeft in een randpunt.
- Hoe gedragen de verschillende regels zich in dit geval? Kun je dit verklaren?

8.3 De Randles-Sevcik functie

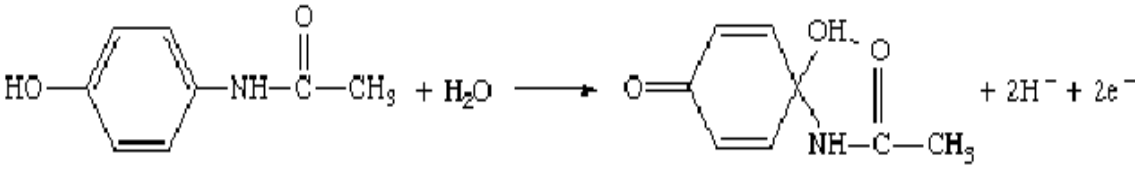
We berekenen een benadering van de Randles-Sevcik functie als een toepassing van numerieke integratie en numerieke interpolatie.

8.3.1 Oorsprong van het probleem

In elektrochemische toepassingen is men geïnteresseerd in de Randles-Sevcik functie. Deze functie geeft het verband aan tussen de elektrische stroom in functie van de cel-spanning.

We geven ter illustratie volgende toepassing.

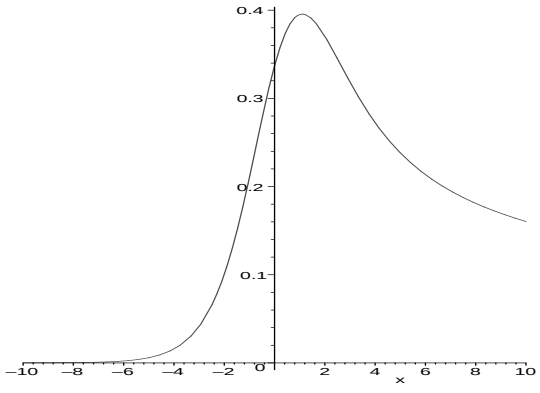
Acetaminophen (AM) is een veel gebruikt actief ingrediënt in vele pijnstillers. Het wordt bijvoorbeeld veel gebruikt in medicatie voor kinderen omdat aspirine dikwijls met het Reye syndroom wordt geassocieerd. Het geneesmiddel is meestal verpakt in tabletten, maar het kan ook als een drankje ingenomen worden. Naast allerlei kleur- en smaakstoffen die elektrochemisch inactief zijn bevat zo een drankje AM dat wel een elektrochemisch actief element is. Cyclische voltammetrie (CV) wordt gebruikt om de concentratie van acetaminophen in het drankje te bepalen. Men legt daarom twee elektroden in het drankje en laat de spanning lineair variëren tussen twee waarden en men meet de resulterende stroom. Tijdens de voorwaartse (toenemende) spanningsverandering zal elke molecule van het acetaminophen 2 elektronen verliezen. Men kan dus aan de elektrische stroom zien hoeveel acetaminophen moleculen er aanwezig zijn.



acetaminophen N-acetyl-4-quinoneimine

Dit is maar een voorbeeld, men kan op een dergelijke manier ook de concentratie van andere elektrochemisch actieve stoffen in een vloeistof bepalen. Bijvoorbeeld de hoeveelheid metalen in drinkwater enz.

Het verloop van de stroom in functie van de spanning heeft een verloop dat er ongeveer als volgt uitziet (zie hiernaast).
De functie heeft juist één piek, gedraagt zich voor $x \to -\infty$ als e^x en voor $x \to +\infty$ als $1/\sqrt{x}$. Het is voornamelijk de piek die bepalend is voor de problemen hierboven geschetst. Men heeft dan ook experimenteel formules opgesteld voor de positie van deze piek (de zg. Randles-Sevcik betrekkingen)



$$i_p = (2.687 \times 10^5) n^{2/3} \nu^{1/2} D^{1/2} AC.$$

Hierbij is i_p de piek stroom (uitgedrukt in Ampères); n is het aantal elektronen in de reactie, ν is de snelheid waarmee de spanning verandert (V/s); D is de diffusie-coëfficiënt van het analytisch specimen (cm^2/s); A is de oppervlakte van de elektroden (cm^2); C is de concentratie van het specimen (mol/cm^3).

Men kan dan een benadering vinden van deze functie en die vergelijken met een gekalibreerde curve om te weten wat uiteindelijk de concentratie is van de te onderzoeken stof in de vloeistof.

8.3.2 De Randles-Sevcik functie

Hoewel de vorm van de functie varieert met een aantal parameters, kan men een typisch verloop beschrijven door een genormaliseerde vorm die de Randles-Sevcik functie heet. Ze is gedefinieerd als

$$f(x) = \frac{d}{dx} \int_{-\infty}^{x} \frac{g(y)dy}{\sqrt{x-y}}, \quad g(y) = \frac{1}{1+e^{-y}}.$$

Mits enig rekenwerk kan men de afgeleide wegwerken en door dan nog een verandering van variabelen in te voeren kan men aantonen dat

$$f(x) = 2 \int_{0}^{\infty} \frac{e^{x-t^2} dt}{(1+e^{x-t^2})^2}.$$

Vermits de piek van de functie in de buurt van 0.45 ligt, en men slechts in de omgeving van die piek is geïnteresseerd, zullen we de functie benaderen in het interval $[-10, 10]$.

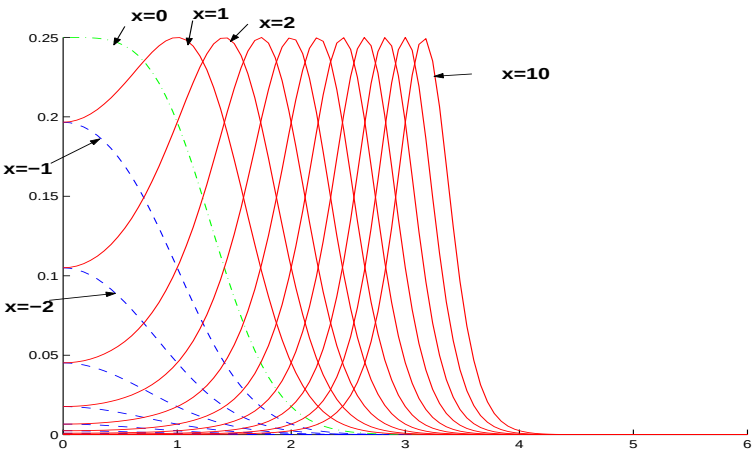
Voor de benadering kiezen we veelterminterpolatie in een aantal punten waarin we de integraal uitrekenen. We kiezen de Chebyshevpunten voor $[a, b] = [-10, 10]$ die gegeven zijn door

$$x_i = \frac{a+b}{2} - \frac{b-a}{2} \cos \frac{(2i+1)\pi}{2n+2}, \quad i = 0, 1, \ldots, n.$$

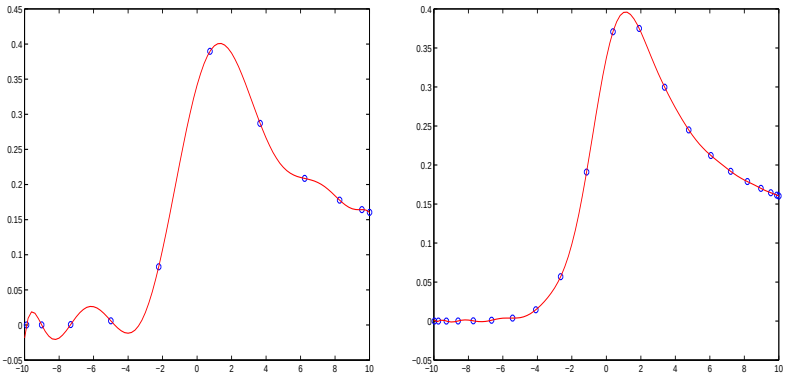
Voor de integratiemethode kiezen we de Simpsonregel.

Als we de integrand tekenen voor verschillende waarden van $x \in [-10, 10]$ dan zien we een verloop als op de volgende figuur:

De curves in stippellijn corresponderen met negatieve x. Voor $x = 0$ krijg je de curve die bij $t = 0$ helemaal bovenaan vertrekt en er wordt voor toenemende x een piek gevormd die langzaam naar rechts opschuift. Voor alle relevante waarden van x is de integrand buiten $[0, 4]$ ongeveer nul. Daarom zullen we de integraal die in feite van 0 tot ∞ loopt slechts uitrekenen van 0 tot 6.



We voeren daarna een veelterminterpolatie uit met het Newton-algoritme. Als we slechts 10 punten opgeven is het slingeren van de veelterm nog te fel om een goede benadering te krijgen (onder links). Bij 20 punten is de benadering behoorlijk (onder rechts).



8.4 Experimenten

De matlab programma's laten toe om wat te experimenteren: `NW/NW/elec/matlab/index.html`.

- Kun je het gedrag van de benadering verklaren als je te weinig (bv. 10) punten neemt om te interpoleren?
- Hoe ziet de benadering eruit als je equidistante punten neemt in plaats van de Chebyshev punten?
- We hebben de integralen (dus de functiewaarden) uitgerekend met een relatieve nauwkeurigheid `reps = 1.0e-5`. Is dit niet te hoog? Spaar je veel functie-evaluaties (en rekentijd) uit indien je dit verlaagt? Hoeveel? Kan je dit verklaren?
- We hebben de integralen met de Simpsonregel berekend. Is dit hier aangewezen? Zou de trapeziumregel niet beter zijn? Zou een adaptieve regel hier niet meer op zijn plaats zijn? Waarom?

Hoofdstuk 9

Differentiaalvergelijkingen

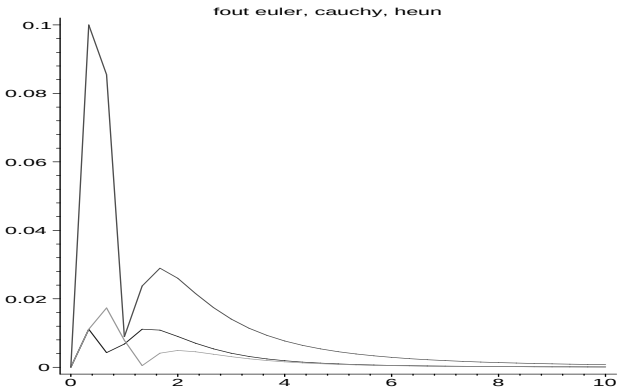
9.1 De orde van een methode

We beschouwen de vergelijking in $y(x)$

$$y' = -2xy^2, \quad x \in [a, b]$$

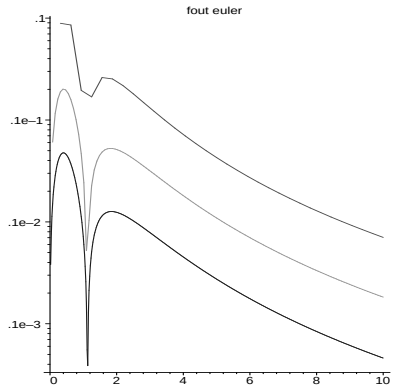
met als oplossing $y_{\text{exact}}(x) = (1 + x^2)^{-1}$.

We kunnen deze vergelijking oplossen met Euler, Heun, of Cauchy in het interval $[a, b]$ als we de beginvoorwaarde $y(a) = y_{\text{exact}}(a)$ kennen. We schrijven een maple code hiervoor en vergelijken de gevonden waarde van de 3 methodes met de waarde van de exacte oplossing. We krijgen voor het geval $[a, b] = [0, 10]$ en als we 30 stappen gebruiken als absolute waarde van de fout:

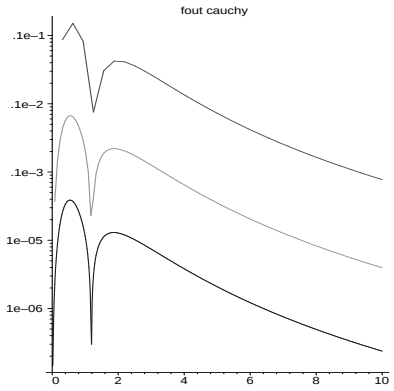


Bekijk de code en voer ze uit. Weet je nu welke curve bij welke methode hoort?

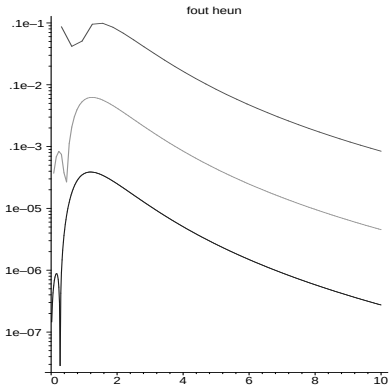
Teken de absolute waarde van de absolute fout voor een stap $h = (b - a)/n$ waarbij $n = 2^k$ met $k = 5, 7, 9$. Het resultaat is op de volgende figuren te zien.



Euler



Cauchy



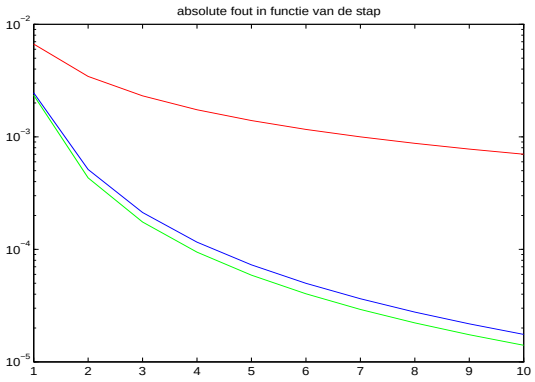
Heun

9.1.1 Experimenten

De maple- en matlabprogramma's kan je vinden op `NW/NW/dvgl/maple/index.html` en `NW/NW/dvgl/matlab/index.html`

- Als men rekening houdt met de logaritmische schaal, hoe kan men dan aan deze figuren zien dat de fout evenredig is met h^p?
- Kan je p schatten uit de bovenstaande figuren voor dit voorbeeld?
- Is dit ook zo voor andere voorbeelden? Het is een constante voor de methode. Men noemt dit de orde van de methode.
- Welke orde heeft de methode van Euler, Cauchy en Heun?
- Probeer met een ander rechterlid waarvoor je ook een exacte oplossing kent. Krijg je dezelfde resultaten?

Wat de orde van een methode betekent kan je zien als je de fout berekent voor meerdere waarden van n (het aantal deelintervallen). Hiernaast zie je de figuur voor de fout in het punt $b = 5$ voor $n = 10k$ punten in functie van k.



Welke curve hoort bij welke methode?

Probeer ook met het matlabprogramma andere voorbeelden uit. Krijg je ook hier dezelfde resultaten? Probeer eens voor verschillende waarden van a, b, $y(a)$, en aantal stappen. Vergelijk de nauwkeurigheid van de verschillende methoden.

9.2 Het opstellen van enkele methoden

9.2.1 BDF formules

Dit zijn impliciete formules.

Deze formules ontstaan door de achterwaartse differenties te gebruiken om $y'(x)$ te benaderen (vandaar de naam). Dit steunt op $D = -\log(I - \nabla)$ en geeft een benadering voor y'_{k+1} in functie van y_{k+1}, y_k, y_{k-1},... Men gebruikt dit in het linkerlid van de vergelijking $y'_{k+1} = f_{k+1}$ en lost dan de vergelijking op naar y_{k+1}.

9.2.2 De Adams-Bashforth formules

Dit is een expliciete methode.

Men gebruikt de interpolerende veelterm van de functie f volgens Lagrange voor de punten x_k, x_{k-1}, x_{k-2}, ... Men veronderstelt equidistante punten en gaat over op de functies in $u = (x - x_k)/h$. Men berekent y_{k+1} als y_k + de integraal van x_k tot x_{k+1} van de zopas

opgestelde interpolerende veelterm. De functiewaarden zijn f_k, f_{k-1}, f_{k-2},... en de gewichten zijn de integralen van de Lagrange basisveeltermen. Merk op dat men de interpolerende veelterm gebruikt om een integraal te berekenen buiten het interpolatieinterval.

9.2.3 De Adams-Moulton formules

Dit is een impliciete methode. De techniek is dezelfde als bij Adams-Bashforth, maar men neemt nu ook het punt x_{k+1} mee in de berekeningen.

Men gebruikt de interpolerende veelterm van de functie f volgens Lagrange voor de punten x_{k+1}, x_k, x_{k-1}, x_{k-2},... Men veronderstelt equidistante punten en gaat over op de functies in $u = (x - x_k)/h$. Men berekent y_{k+1} als y_k + de integraal van x_k tot x_{k+1} van de zopas opgestelde interpolerende veelterm. De functiewaarden zijn f_{k+1}, f_k, f_{k-1}, f_{k-2},... en de gewichten zijn de integralen van de Lagrange basisveeltermen. Merk op dat men de interpolerende veelterm gebruikt om een integraal te berekenen in slechts een stukje van het interpolatieinterval.

De corresponderende maple code kan je hier vinden: `NW/NW/dvgl/maple/index.html`

9.2.4 Vragen

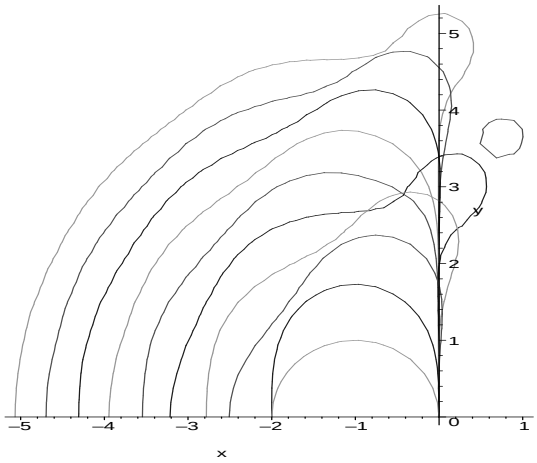
- Ga na hoe de verschillende methodes zijn geprogrammeerd in maple. Versta je alle commando's? Gebruik desnoods de maple help.
- Wat weet je over de stabiliteitsgebieden van de BDF formules?
- Worden de coëfficiënten in de lineaire combinatie die moet uitgerekend worden groot en van een verschillend teken voor hoge orde methoden? Dit is belangrijk voor afrondingsfouten.
- Wat weet je over de orde van al deze methoden? Heeft de orde iets te maken met het aantal punten of het aantal functieevaluaties dat men gebruikt?
- Kan je de lokale fout eenvoudig schatten bij methoden van het type predictor-corrector zoals Adams-Bashforth-Moulton? Ga dit na voor een paar voorbeelden.
- Als predictor-corrector methoden een eenvoudige foutenschatting geven, kunnen ze gemakkelijk voorspellen of de stap moet vergroot of verkleind worden. Zijn ze dan ook eenvoudig te gebruiken in adaptieve methoden?
- Zijn de ABM formules zelfstartend?

9.2.5 Extra

- De BDF formules zijn impliciet. Heeft het hier zin om een expliciete formule op te stellen op deze manier, namelijk door $y'_{k+1} = f_k$ te benaderen met achterwaartse differenties?
- Werkt dit even goed als de impliciete methode? Programmeer en test uit op een voorbeeld.

9.3 Stabiliteitsgebieden van expliciete methoden

De stabiliteitsgebieden zijn in dit geval de punten in het complexe vlak waarvoor de functie $f_p(z) = 1+z/1!+z^2/2!+ \ldots + z^p/p!$ in modulus gelijk is aan 1. Hiernaast is de figuur getekend voor $p = 1, 2, \ldots, 10$.
De figuur is maar getekend voor Im$z >$ 0. Natuurlijk hangt er aan de onderkant een symmetrisch stuk aan. Je kan zelf experimenteren met het maple werkblad `BDF.mws` dat je hier kan vinden: `NW/NW/dvgl/maple/index.html`



9.3.1 Experimenten en vragen

- Weet je voor welke waarde van p de kleine cirkel in het rechterhalfvlak getekend wordt?
- Waarom liggen die gebieden (grotendeels) in het linker halfvlak?
- Is het gebied dat overeenkomt met $p = 1$ een cirkel of lijkt dit alleen maar zo?
- Wat kan je zeggen over de stabiliteit van de methoden als p toeneemt?
- Voor welke methoden ziet de functie f eruit zoals hierboven beschreven? Of is dit voor alle methoden zo?
- Waarom is p de orde van deze methode?
- Leg uit waarom de hierboven berekende stabiliteitsgebieden overeenstemmen met die van de Runge-Kutta methoden.
- Wat hebben deze tekeningen te maken met het stabiliteitsinterval van een methode? Heeft dit iets te maken met het begrip “numerieke stabiliteit van een methode” zoals we dat gezien hebben?

9.3.2 Extra

Het maple werkblad bevat ook technieken om de stabiliteitsgebieden te berekenen voor andere methoden zoals BDF, AB en AM.

- Hoe moet je voor andere methoden de stabiliteitsgebieden berekenen?
- Zijn de stabiliteitsgebieden voor AB groter of kleiner dan die voor AM?
- Op de grafiek voor de stabiliteitsgebieden voor de AM(n) methoden is de tekening voor $n = 1$ niet terug te vinden. Waarom niet?
- Kan je uit de tekeningen voor de stabiliteitsgebieden voor BDF formules ook de stabiliteitsintervallen aflezen?
- Tot welke orde correspondeert dit met de ganse negatieve reële as?

9.4 Inherente onstabiliteit

Sommige differentiaalvergelijkingen zijn inherent onstabiel.

Neem de vergelijking

$$y'(x) = K[y(x) - f(x)] + f'(x).$$

De algemene oplossing van de vergelijking is

$$y(x) = [y(x_0) - f(x_0)] \exp[K(x - x_0)] + f(x).$$

Als men de oplossing $y(x) = f(x)$ wil hebben, dan moet men de startwaarde $y(x_0) = f(x_0)$ gebruiken.

De kleinste afrondingsfout die men maakt zal er echter voor zorgen dat $y(x_0) - f(x_0)$ niet exact nul is zodat er een component $\exp[K(x - x_0)]$ gaat meespelen. Als K positief is zal deze component sterk stijgen. Als de gezochte oplossing $y(x) = f(x)$ begrensd blijft voor toenemende x, dan zal al gauw de dominante exponentiële component gaan overheersen. Vermits elke numerieke berekening afrondingsfouten zal hebben, zal uiteindelijk elke numerieke methode de mist in gaan.

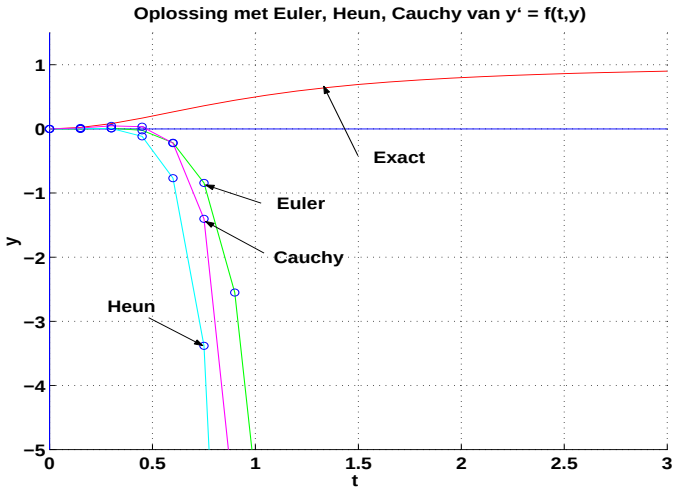
Ter illustratie nemen we het voorbeeld waarbij $K = 10$, en

$$f(x) = x^2/(1 + x^2),$$

met

$$f'(x) = 2x/(1 + x^2)^2.$$

De echte oplossing zal dus stijgen vanaf nul naar de horizontale asymptoot $y = 1$. De exponentiële component overheerst zeer vlug zoals het volgende voorbeeld illustreert.



De matlab code hiervoor kan je hier ophalen: `NW/NW/matlab/dvgl/index.html`

9.4.1 Experimenten

- Probeer de matlab code ook eens voor andere functies $f(x)$ die begrensd blijven. Merk je hetzelfde gedrag op?
- Helpt het als men de stap kleiner maakt om de lokale discretisatiefout te verkleinen? Blijft de berekende oplossing dan langer tegen de exacte oplossing "plakken"?
- Is in het algemeen de afwijking voor een methode van lagere orde (Euler) erger of minder erg dan voor een methode van hogere orde (Cauchy, Heun)?

9.4.2 Extra

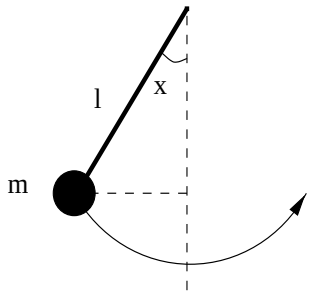
- Probeer ook eens een andere methode te programmeren (bv RK of ABM). Is het resultaat net hetzelfde als voor Euler, Cauchy, Heun?
- Welke methoden zitten er in matlab ingebakken voor het oplossen van differentiaalvergelijkingen? (doe `help funfun` in matlab).

9.5 Een voorbeeld: de slinger

We gebruiken de Runge-Kutta methodes om de bewegingsvergelijking voor een enkelvoudige slinger op te lossen.

9.5.1 Beschrijving van het probleem

We beschouwen het probleem van een eenvoudige slinger:



De bewegingsvergelijking van een slinger wordt gegeven door een tweede orde gewone differentiaalvergelijking.

$$ml\frac{d^2x}{dt^2} + c\frac{dx}{dt} + mg\sin x = 0.$$

Hierin is

symbool	omschrijving	eenheden	waarde
x	de uitwijkingshoek	–	
g	de gravitatieconstante	m/sec^2	9.81
l	de lengte van de slinger	m	1
m	de massa van de slingerbol	m	1
c	een constante (demping)	kg m/sec	0.1

We kunnen dit omvormen tot een stelsel van eerste orde differentiaalvergelijkingen

$$\begin{cases} \dfrac{dx}{dt} &= y \\ \dfrac{dy}{dt} &= -\alpha\sin x - \beta y \end{cases} \quad \text{waarin} \quad \begin{cases} \alpha &= \dfrac{g}{l} \\ \beta &= \dfrac{c}{ml} \end{cases}$$

Stellen we $\mathbf{x} = (x, y)$, dan is dit

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}) = \begin{bmatrix} y \\ -\alpha\sin x - \beta y \end{bmatrix}$$

De beginvoorwaarden zijn (x_0, y_0). Daarbij is x_0 de hoek op het ogenblik $t = 0$ (we stellen $x_0 = \pi/2$) en y_0 is de beginsnelheid (van de hoek) die aan de slinger meegegeven wordt.

9.5.2 De Runge-Kutta formules

De Runge-Kutta formules van verschillende orde worden gegeven door:

Runge-Kutta van orde 1 (RK1)

$$\mathbf{x}_{k+1} = \mathbf{x}_k + hf(\mathbf{x}_k) \equiv \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + h \begin{bmatrix} y_k \\ -\alpha \sin x_k - \beta y_k \end{bmatrix}$$

Runge-Kutta van orde 2 (RK2)

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{x}_k) = \begin{bmatrix} y_k \\ -\alpha \sin x_k - \beta y_k \end{bmatrix} \\ \mathbf{k}_2 &= f(\mathbf{x}_k + \frac{h}{2} f(\mathbf{x}_k)) = f(\mathbf{x}_k + \frac{h}{2}\mathbf{k}_1) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + h\mathbf{k}_2 \end{aligned}$$

Runge-Kutta van orde 3 (RK3)

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{x}_k) \\ \mathbf{k}_2 &= f(\mathbf{x}_k + \frac{h}{2}\mathbf{k}_1) \\ \mathbf{k}_3 &= f(\mathbf{x}_k + h(-\mathbf{k}_1 + 2\mathbf{k}_2)) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \frac{h}{6}(\mathbf{k}_1 + 4\mathbf{k}_2 + \mathbf{k}_3) \end{aligned}$$

Runge-Kutta van orde 4 (RK4)

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{x}_k) \\ \mathbf{k}_2 &= f(\mathbf{x}_k + \frac{h}{2}\mathbf{k}_1) \\ \mathbf{k}_3 &= f(\mathbf{x}_k + \frac{h}{2}\mathbf{k}_2) \\ \mathbf{k}_4 &= f(\mathbf{x}_k + h\mathbf{k}_3) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \end{aligned}$$

9.5.3 Evenwichtstoestanden

De evenwichtstoestanden van het systeem zijn

Het geval $(x^*, y^*) = (n\pi, 0)$ met n oneven

In dit geval staat de slinger rechtop stil (na een aantal omwentelingen). Hij zal natuurlijk bijna nooit in deze toestand eindigen. Het is een onstabiel evenwicht (een zadelpunt).

Het geval $(x^*, y^*) = (n\pi, 0)$ met n even

Hier hangt de slinger stil onderaan (na een aantal omwentelingen). Dit is de normale eindtoestand. Het is een stabiele toestand.

9.5.4 Het fasevlak

Als we in het (x, y)-vlak de beweging van de slinger uitzetten, dan zal naar de stabiele toestand geëvolueerd worden doordat het pad zich spiraalvormig naar dit punt zal convergeren. We noemen de stabiele toestanden daarom ook soms spiraalpunten.

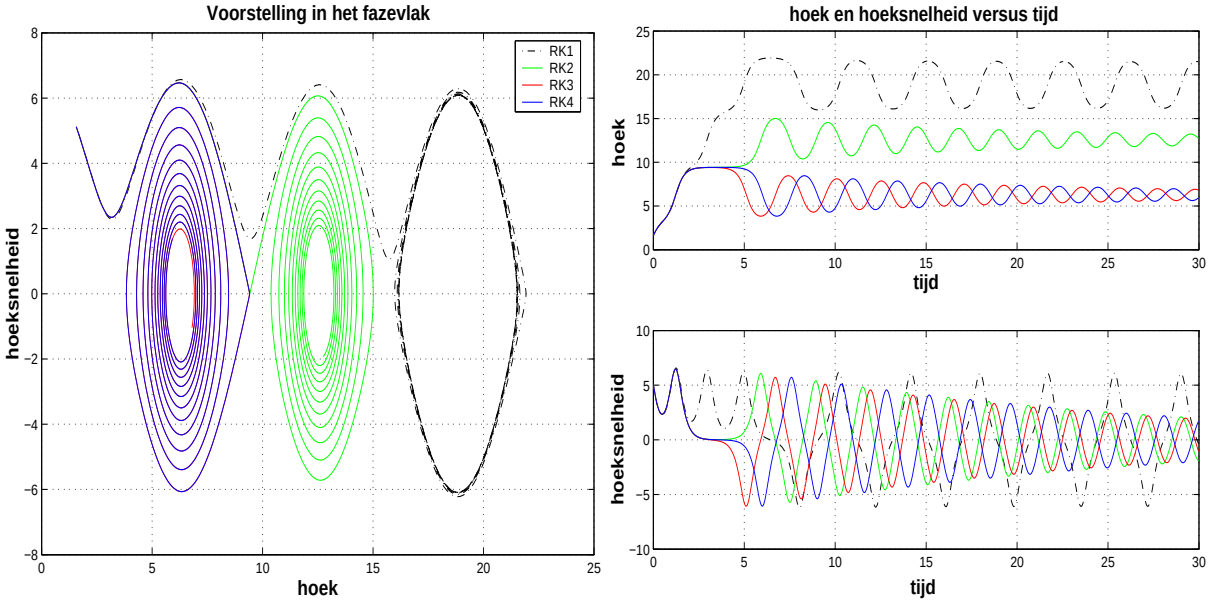
9.5.5 Experimenten

- Men kan de gevoeligheid van de berekende oplossing tov de beginvoorwaarden laten narekenen: Het moet blijken dat de berekende oplossing voor bepaalde beginvoorwaarden, zeer sterk afhankelijk is van perturbaties op die beginvoorwaarden. (In het programma kan men enkel de beginhoeksnelheid opgeven, de beginuitwijking is steeds $\pi/2$ gekozen.)
  Bijvoorbeeld
  - $y(0) = 5.12$ en $h = 0.05$
    - RK1 gaat naar een evenwichtspunt
    - RK2 gaat naar een verkeerd evenwichtspunt
    - RK3 is ongeveer RK4, maar toch, bij de tijdsevolutiegrafiek merkt men een faseverschuiving op.
  - $y(0) = 5.12055$ en $h = 0.025$
    - RK1 gaat naar een verkeerd evenwichtspunt
    - RK2 gaat naar een ander verkeerd evenwichtspunt
    - RK3 is ongeveer RK4, maar toch, bij de tijdsevolutiegrafiek merkt men een faseverschuiving op.
  - $y(0) = 5.1206$ en $h = 0.025$
    - RK1 gaat naar een verkeerd evenwichtspunt
    - RK2 gaat naar het juiste evenwichtspunt (maar met faseverschuiving)
    - RK3 gaat naar een verkeerd evenwichtspunt

  Opmerking: voor het ontwerpen van een controlesysteem is de faseverschuiving (door numerieke berekeningen) niet aanvaardbaar en kan catastrofale gevolgen hebben.

- Zoek de beginvoorwaarden die nodig zijn om de slinger net te doen terugvallen na een (bijna) volledige omwenteling. Dit kan door de beginsnelheid op te geven. Zoek ook waar men in het programma de beginuitwijking kan aanpassen. Probeer hetzelfde probleem op te lossen voor een andere beginuitwijking.
- Is het mogelijk om een onstabiele oplossing te berekenen met de numerieke methoden die hier gebruikt zijn?
- Welke van de eenvoudige methoden Euler, Cauchy en Heun zijn opgenomen in dit matlab programma?

In onderstaande figuren werd gebruikt: $h = 0.025$, $m = 30$, $y(0) = 5.12055$.



9.5.6 Extra

- Reken na of de methoden correct geprogrammeerd zijn in matlab.
- Stel dat men een stelsel wil oplossen maar met een andere $f(\mathbf{x})$, hoe zou men dan de matlabcode moeten aanpassen?
  Neem bijvoorbeeld een gedempt massa/veer systeem dat beschreven wordt door

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = 0$$

  met

symbool	omschrijving	eenheden	waarden
x	de uitwijking	m	
m	de massa	kg	1
k	de veerconstante	kg/sec^2	1
b	dempingsfactor	kg/sec	2

  Voor welke waarde van b is de demping kritisch? Komt dit ook tot uiting in de numerieke berekeningen?
- Zal elk stelsel van de vorm $\dot{\mathbf{x}} = f(t, \mathbf{x})$ waarbij $\mathbf{x} = (x, y)$ afkomstig zijn van een tweede orde differentiaalvergelijking in een scalaire variabele?

De matlab-code kan je vinden op `NW/NW/slinger/matlab/index.html`.

9.6 Java experimenten

Voor een java applet voor dit probleem zie:
`NW/NW/slinger/java/index.html`.

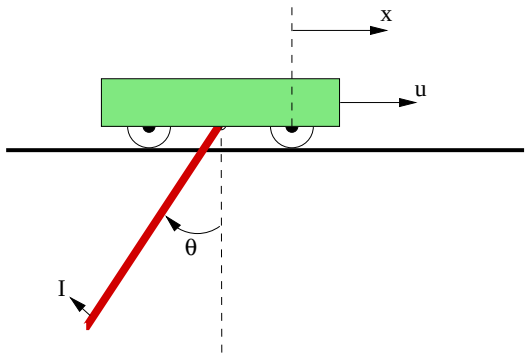
9.7 Slinger met bewegend ophangpunt

We bekijken een wat ingewikkelder probleem.

9.7.1 Opstelling van wagentje en slinger

Tabel 9.1: Parameters

l	=	halve lengte staaf	m
m_p	=	massa staaf	kg
θ	=	hoek	rad
ω	=	hoeksnelheid	rad/s
I	=	traagheidsmoment	kg.m^2
m_t	=	massa wagentje	kg
u	=	aandrijving wagentje	N
x	=	positie wagentje	m
v	=	snelheid wagentje	m/s
b	=	weerstandscoëf.	kg



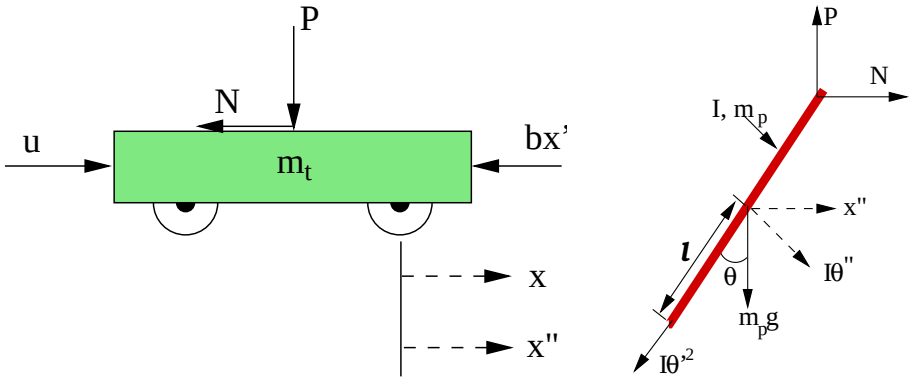
De opstelling bestaat uit een wagentje met 4 wielen en een massa m_t dat zich horizontaal kan voortbewegen, zowel voorwaarts als achterwaarts. Het wagentje bevindt zich op horizontale positie x, heeft een snelheid v en een aandrijving u. In het midden onderaan het wagentje hangt een rechte massieve staaf die een halve lengte l en een massa m_p heeft. De staaf vormt een hoek θ met een vertikale lijn en heeft een hoeksnelheid ω. Het traagheidsmoment van de slinger is I. Er is een weerstand die evenredig is met de snelheid: $b\dot{x}$.

9.7.2 Bewegingsvergelijkingen

Door de parameters in bovenstaande tabel juist te kiezen kunnen we een opstelling construeren waarbij de slinger en het wagentje zullen bewegen en waardoor een aantal van de parameters uit de tabel zullen veranderen. Willen we die bewegingen simuleren, moeten we telkens na een bepaald tijdsinterval die variabele parameters zien te achterhalen. Eerst moeten we het fysisch model omzetten naar een wiskundig model. We doen dit door het systeem op te splitsen en afzonderlijk te kijken naar de uitgeoefende krachten op het wagentje en de slinger. De figuren illustreren dit. Als we de krachten die in de horizontale richting uitgeoefend worden op het wagentje sommeren, levert dit ons deze vergelijking op

$$m_t\ddot{x} + b\dot{x} = u - N \tag{9.1}$$

Figuur 9.1: Krachten op het wagentje en de slinger



Zo sommeren we ook de krachten die in de horizontale richting uitgeoefend worden op de slinger en bekomen we deze vergelijking

$$N = m_p\ddot{x} + m_p l\ddot{\theta}\cos\theta - m_p l\dot{\theta}^2\sin\theta \tag{9.2}$$

De som van de krachten loodrecht op de slinger resulteert in de volgende vergelijking

$$P\sin\theta + N\cos\theta - m_p g\sin\theta = m_p l\ddot{\theta} + m_p\ddot{x}\cos\theta \tag{9.3}$$

Tenslotte hebben we nog de som van de momenten rond het midden van de slinger

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \tag{9.4}$$

Aan de hand van de vorige vier vergelijkingen bekomen we nu twee bewegingsvergelijkingen

$$\text{de eerste 2} \Rightarrow (m_t + m_p)\ddot{x} + b\dot{x} + m_p l\ddot{\theta}\cos\theta - m_p l\dot{\theta}^2\sin\theta = u \tag{9.5}$$

$$\text{de laatste 2} \Rightarrow (I + m_p l^2)\ddot{\theta} + m_p g l\sin\theta = -m_p l\ddot{x}\cos\theta \tag{9.6}$$

Verder hebben we:

$$\boxed{\begin{array}{ll} \dot{\theta} = \frac{d\theta}{dt} = \omega, & \ddot{\theta} = \frac{d\omega}{dt} \\ \dot{x} = \frac{dx}{dt} = v, & \ddot{x} = \frac{dv}{dt} \end{array}} \tag{9.7}$$

We hebben dus twee differentiaalvergelijkingen van de tweede orde in functie van de tijd t

$$(m_t + m_p)\frac{dv}{dt} + bv + m_p l\frac{d\omega}{dt}\cos\theta - m_p l\omega^2\sin\theta = u \tag{9.8}$$

$$(I + m_p l^2)\frac{d\omega}{dt} + m_p g l\sin\theta = -m_p l\frac{dv}{dt}\cos\theta \tag{9.9}$$

De differentiaalvergelijkingen (9.8) en (9.9) brengen we vervolgens terug tot een stelsel differentiaalvergelijkingen van de eerste orde. Hiervoor berekenen we uit (9.8) en (9.9) $\frac{d\omega}{dt}$ en $\frac{dv}{dt}$:

$$\frac{d\omega}{dt} = \frac{(u - bv + m_p l\omega^2 \sin\theta) + (m_t + m_p)(gtg\theta)}{\frac{(m_t+m_p)(I+m_p l^2)}{-m_p l\cos\theta} + m_p l\cos\theta} \tag{9.10}$$

$$= \frac{(-m_p l\cos\theta)(u - bv + m_p l\omega^2 \sin\theta) - (m_t + m_p)(m_p gl\sin\theta)}{(I + m_p l^2)(m_t + m_p) - (m_p l\cos\theta)^2} \tag{9.11}$$

$$\frac{dv}{dt} = \frac{(I + m_p l^2)(u - bv + m_p l\omega^2 sin\theta) + (m_p l)^2 g\sin\theta\cos\theta}{(I + m_p l^2)(m_t + m_p) - (m_p l\cos\theta)^2} \tag{9.12}$$

Noot: Verder maken we gebruik van (9.11) i.p.v. (9.10) om een deling door 0 te vermijden en omdat de noemer van (9.11) herbruikt kan worden als de noemer van (9.12).
Alles bij elkaar genomen, bekomen we het volgende stelsel differentiaalvergelijkingen van de eerste orde

$$\begin{cases} \frac{d\theta}{dt} = \omega \\ \frac{dx}{dt} = v \\ \frac{d\omega}{dt} = \text{voorlaatste formule} \\ \frac{dv}{dt} = \text{laatste formule} \end{cases}$$

Indien we nu $\boldsymbol{x} = (\theta, x, \omega, v)$ stellen en $\boldsymbol{x}$ afleiden naar de tijd krijgen we

$$f(\boldsymbol{x}) = \frac{d\boldsymbol{x}}{dt} = \begin{bmatrix} \omega \\ v \\ \frac{d\omega}{dt} \\ \frac{dv}{dt} \end{bmatrix} \tag{9.13}$$

9.7.3 Oplossing van de vergelijkingen

De bekomen vergelijkingen kunnen opgelost worden met een van de vele methoden voor het oplossen van dergelijke problemen. Sommige zullen goed werken, anderen zullen een slechte numerieke benadering geven.

Stel zelf een aantal methodes op. Je kan op het web een aantal resultaten vinden.

9.7.4 Java of matlab experimenten

Voor experimenten met een java applet zie:
`NW/NW/wagenslinger/java/index.html`.
De bijhorende matlab programma's vind je hier:
`NW/NW/wagenslinger/matlab/index.html`.

- Er is een verschil tussen de BDFx en de BDFxopt methodes. De eerste werken op de klassieke predictor-corrector filosofie. De tweede lossen de impliciet gedefinieerde waarde y_{k+1} op door een Newton-Raphson methode te gebruiken om het stelsel op te lossen. Om de goede stabiliteitseigenschappen van de BDF methodes te krijgen moet men de oplossing iteratief berekenen. De stabiliteitsgebieden zijn in het maple programma ook zo berekend.

- Men kan een zeer stijf probleem krijgen door de waarde van b zeer groot te kiezen (bv. $b = 100$). Waarom is dit een stijf probleem?
- De klassieke methodes geven geen goede oplossing voor een stijf probleem. Leg uit waarom. Ga het verschil na tussen BDF en BDFopt methodes. Kan je verklaren waarom dit een zo groot verschil geeft?
- Vanaf welke waarde van b ($b > 1$) krijg je stabiliteitsproblemen? Is dit voor alle methodes hetzelfde?
- Zou je de andere predictor-corrector methodes (andere dan BDF) ook stabieler kunnen maken door iteratief de impliciet gedefinieerde waarde voor y_{k+1} op te lossen?
- Bij de BDF formules moet je x_{k+1} oplossen uit een formule van de vorm

$$F(x_{k+1}) \equiv x_{k+1} + a_m x_{k-m} + a_{m-1} x_{k-m+1} + \cdots + a_0 x_k + bhf(x_{k+1}) = 0.$$

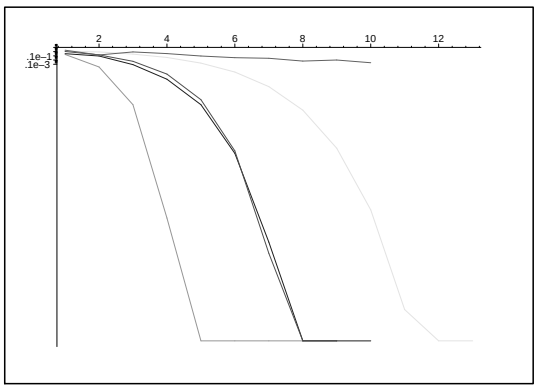
  Om dit niet-lineair stelsel in de onbekende vector x_{k+1} op te lossen gebruiken we de methode van Newton. Daarvoor moet men de Jacobiaan berekenen. Die is van de vorm $J = I + bh(df/dx)$ waarbij df/dx de Jacobiaan is van f. Bereken deze df/dx en ga na in het matlabprogramma hoe die daar effectief berekend werd.

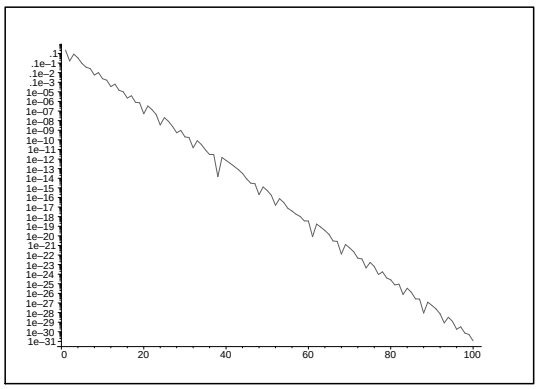
Hoofdstuk 10

Berekenen van nulpunten

10.1 De orde van convergentie

We zoeken het nulpunt van de functie $f(x) = 4x^4 - 5x - 3$ in het interval $[a, b] = [1, 2]$.
De eerste en tweede afgeleide zijn $f'(x) = 16x^3 - 5$, en $f''(x) = 48x^2$.
Het nulpunt is ongeveer
1.22977930747949339627995463156823314106696641713985800831821488238100328420183458020451855 63
Hieronder links zie je de relatieve fout in functie van de iteratiestap voor Bissectie, Secant, Muller, Newton, Halley (in deze volgorde is de convergentiesnelheid toenemend).





Merk op dat de bissectie lineair is (de bovenste curve is een rechte lijn op log schaal) en dat de andere methoden superlineair zijn (kromme op log schaal).

Als we de relatieve fout voor de bissectiemethode uitzetten wordt de lineariteit nog duidelijker. Zie bovenstaande figuur rechts.

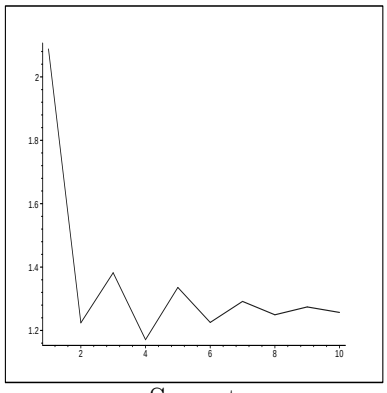
Laten we ook eens nagaan of voor dit voorbeeld de verschillende methoden van de gepaste orde zijn. Daarvoor moet de limiet

$$\lim_{k\to\infty} \frac{\epsilon_{k+1}}{\epsilon_k^p}$$

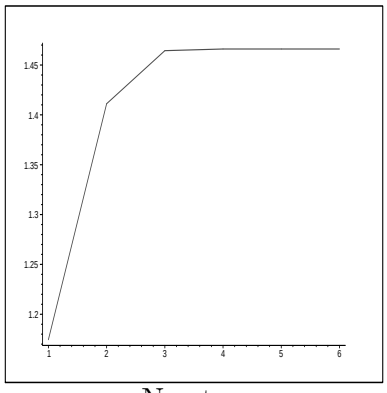
bestaan en niet 0 en niet oneindig zijn als p de orde is en ϵ_k de fout in de k-de stap.

We weten dat voor een enkelvoudige wortel zoals hier
$\text{orde}_{\text{Secant}} = 1.618...$
$\text{orde}_{\text{Newton}} = 2$
$\text{orde}_{\text{Muller}} = 1.839...$
$\text{orde}_{\text{Halley}} = 3$

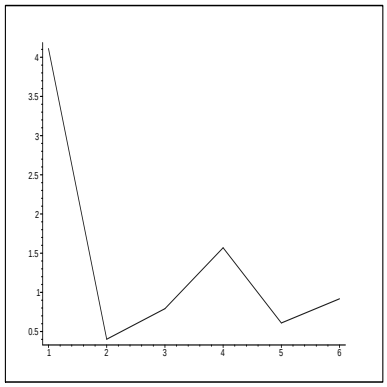
De figuren hieronder geven de waarden van $\epsilon_{k+1}/\epsilon_k^p$ voor enkele waarden van k. Voor de juiste p moet dit naar een constante streven.



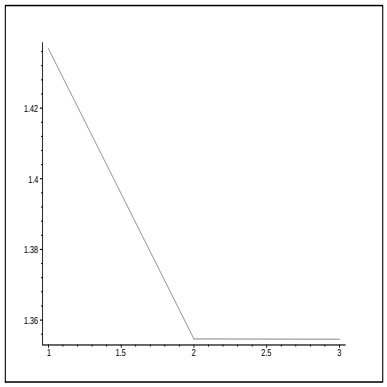
Secant



Newton



Muller



Halley

10.1.1 De orde van convergentie en het aantal juiste cijfers

Men kan aan de resultaten zien welke de orde van convergentie is: Als de orde p is, dan wordt het aantal juiste cijfers met p vermenigvuldigd.

We illustreren dit voor de methode van Newton ($p = 2$) en de methode van Halley ($p = 3$) op het voorbeeld $f(x) = x^2 - 4$, de startwaarde is $a = 4$.

Berekening met Newton

2.500000000000000000000000000000000000000000000000000000000000000000000000000000000000000,
2.050000000000000000000000000000000000000000000000000000000000000000000000000000000000000,
2.000609756097560975609756097560975609756097560975609756097560975609756097560975609756,
2.000000092922294660313259639758848060897555028582897837512358665189821663532087 9,
2.000000000000021586381109417172262212869091611099918033920872653754835313551091,
2.000000000000000000000000000011649296235025051145733475124733614073734306014401,
2.000000000000000000000000000000000000000000000000000000000000033926525692842207 92,
2.000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

Berekening met Halley

2.1538461538461538461538461538461538461538461538461538461538461538461538461538462,
2.0002032313789249060054872472309724621481556752362564780002032313789249060054873,
2.0000000000000524549060958905790823212416262425497512038423172183541749635185417 1,
2.0000000000000000000000000000000000000090206609349138355553519590375073488339885,
2.0000000000000000000000000000000000000000000000000000000000000000000000000000000

10.1.2 Experimenten en vragen

Het maple werkblad kan je hier vinden: `NW/NW/nulpt/maple/index.html`.

- Probeer ook eens om de berekeningen te doen met andere functies. Krijgt men steeds dezelfde orde van convergentie?
- Probeer ook eens met een functie die een meervoudig nulpunt heeft. Wat merk je voor de verschillende ordes?
- Convergeren de methoden altijd? Voor alle functies? Voor alle startwaarden? En als er convergentie is, is het dan altijd naar de juiste wortel? Probeer eens met een functie die meerdere wortels heeft.

10.1.3 Extra

- Als je in het werkblad probeert het nulpunt te bepalen van $\cos(x)$, wat merk je dan op. Kan je dit verklaren?

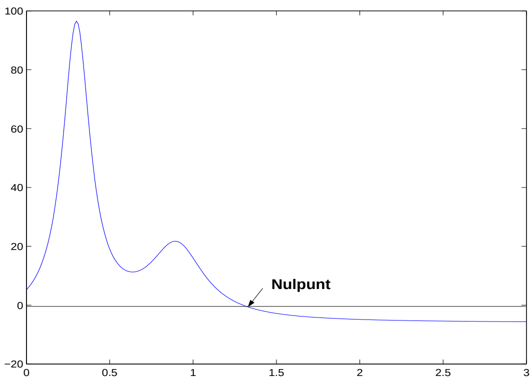
10.2 Dekker-Brent en Muller

10.2.1 De functie

We beschouwen het voorbeeld van de functie `humps` (een ingebouwde functie in matlab).

```
humps(x) = 1 / ((x-3)2 + 0.01) + 1 / ((x-0.9)2 + 0.04) - 6
```

De grafiek en het nulpunt x, ongeveer gelijk aan 1.29954968258482, kan je zien op bijgaande figuur.



10.2.2 De methode van Dekker en Brent

De Dekker-Brent-methode is een combinatie van de secant- en de bissectie-methode waarbij we steeds een onzekerheidsinterval zullen verkleinen.
Het onzekerheidsinterval is het interval waarvan we weten dat het een nulpunt (minstens 1) moet bevatten.
Het principe is:
gebruik steeds de secant-methode (want die is het snelst)
behalve als de secant-methode geen goed punt oplevert
gebruik in dat geval bissectie voor het onzekerheidsinterval
Wanneer levert secant geen goed iteratiepunt?
Als het secant-punt buiten het onzekerheidsinterval ligt of
als 1 van de eindpunten van het onzekerheidsinterval te lang (bv. 2 of 3 stappen)
op dezelfde plaats blijft liggen.
Het matlab programma `dbdemo.m` is hier te vinden: `NW/NW/nulpt/matlab/index.html`.
Je krijgt per iteratiestap te zien
het onzekerheidsinterval
de secantbenadering
de gebruikte methode en de reden waarom die gebruikt werd
de x waarde en de bijhorende functiewaarde
En op het einde de absolute fout in functie van de iteratiestap.

10.2.3 Experimenten en vragen

- Voer het programma dbdemo uit in matlab.
- Hoe snel is de convergentie van het Dekker-Brent algoritme hier?
- Is de bissectie-methode gebruikt? Hoe dikwijls en om welke reden?
- Hoe berekent men vanuit de twee startwaarden het eerste iteratiepunt?
- Hoe gaat men testen of het secant-punt geen goed punt is?
- Welk stopcriterium wordt gebruikt?
- Hoe wordt de uiteindelijke waarde van x bepaald?
- Na hoeveel iteratiestappen is er convergentie?

In matlab zit ook een Dekker-Brent algoritme ingebouwd. Het heet `fzero`.

- Voer de demo-routine `zerodemo` uit in matlab.
- Doe `type fzero` om een listing van `fzero` te krijgen.

10.2.4 De methode van Muller

De methode van Muller gebruikt kwadratische interpolatie.
Het matlab programma `muldemo.m` is hier te vinden: `NW/NW/nulpt/matlab/index.html`.
Je ziet de methode aan het werk voor
de functie humps van hierboven
de functie hmps1 = humps - 20 (er zijn nu 3 nulpunten)
de functie `fdmul1(x) = exp(x)*cos(x+3)+(x+1)*sin(x)`
Je krijgt per iteratiestap te zien

de interpolerende parabool met de gebruikte punten
de waarde van het iteratiepunt en de functiewaarde
En op het einde de absolute fout in functie van de iteratiestap.

10.2.5 Experimenten en vragen

- Voer het programma `muldemo` uit in matlab.
- Hoe snel is de convergentie van de methode van Muller?
- Kan de methode van Muller ook complexe nulpunten vinden?
- Hoe berekent men vanuit de twee startwaarden het derde punt om de parabolische interpolatie de eerste keer te kunnen uitvoeren?
- Hoe gaat men bepalen welk van de 2 nulpunten van de parabool het goede is?
- Levert de interpolatie steeds 2 reële nulpunten op in de voorbeelden?
- Wat gebeurt er als die nulpunten complex worden?
- Kan de methode dan nog convergeren naar een reële oplossing?
- Welk stopcriterium wordt gebruikt?
- Na hoeveel iteratiestappen is er convergentie in de voorbeelden?
- Hoe zou de regula-falsi werken op het laatste voorbeeld?

10.2.6 Extra

- Ga na in welke mate de methode van `fzero` essentieel verschilt van de methode in `db`(afgezien van de behandeling van het uitvoeren van tussenresultaten).
- Pas de vorige methoden toe op problemen met een dubbel nulpunt.
- Wat gebeurt er met de methoden? Is er convergentie? Is die even snel als voor een enkelvoudig nulpunt?
- De routines `db` en `muller`, zonder het uitschrijven van tussenresultaten zijn hier te vinden: `NW/NW/nulpt/matlab/index.html`.
- Schrijf een routine voor de regula falsi in matlab en pas die toe op de derde testfunctie fdmul1 van `muldemo`.
- Hoe brengt deze methode het ervan af op dit voorbeeld?

10.3 Over stabiliteit

Voor de code zie hier: `NW/NW/nulpt/matlab/index.html`.
Opgelet matlab gebruikt de IEEE standaard dubbele precisie berekeningen.
We beschouwen het voorbeeld van de veelterm

```
pol(x) = 0.5 x^3 - 1.5 x^2 - 0.4999 x + 1.4999
```

Men kan eenvoudig nagaan dat er een exact nulpunt $x = 1$ is.
Als men met als startwaarde $x_0 = 0.5$ de Newton-Raphson methode toepast, dan krijgt men als iteratiestappen

0.50000000000000
1.07692781094221
0.99977134862919
1.00000000000598
1.00000000000000
1.00000000000000

en alle 16 berekende cijfers zijn juist.
Gebruiken we echter de eenvoudige substitutiemethode

$$x(k+1) = x(k) + f(x(k))$$

dan is de convergentie veel trager en dan krijgen we dat na een aantal stappen de methode niet meer convergeert. Ze geraakt in een oneindige lus.
Er zijn op deze manier slechts 12 cijfers van het resultaat te vinden omdat het algoritme al vlug in een oneindige lus geraakt.
We hebben met een startwaarde $x_0 = 0.99999999999914$ bijvoorbeeld voor de iteratiestappen 9990:10000

1.00000000000083
0.99999999999917
1.00000000000083
0.99999999999917
1.00000000000083
0.99999999999917
1.00000000000083
0.99999999999917
1.00000000000083

Als we de versnelling van Aitken toepassen op de trage methode, dan kan dit op 2 manieren. Stel in dit geval

$$F(x) = x + f(x); \quad x0 = x(k); \quad x1 = F(x0); \quad x2 = F(x1).$$

Ofwel kunnen we de formule

$$x(k+1) = (x2 * x0 - x1 * x1)/(x0 - 2 * x1 + x2)$$

ofwel de formule

$$x(k+1) = x2 - (x2 - x1)^2/(x0 - 2 * x1 + x2)$$

gebruiken.

Deze nieuwe rij $x(k)$ zal sneller convergeren (volgens de stelling van Steffenson is de convergentie kwadratisch, en dus zo snel als Newton-Raphson).

De eerste manier is gevoeliger voor afrondingsfouten. Weerom zal de exacte oplossing niet bereikt worden maar blijft de rij oscilleren rond de exacte oplossing.

De tweede formule berekent een correctie op het vorige iteratiepunt en dit blijkt naukeuriger te gaan. Na slechts enkele stappen (vergelijkbaar met Newton-Raphson) wordt de exacte oplossing bereikt. De fout is (binnen de nauwkeurigheidsgrenzen van matlab) dan nul en er treedt geen oscillatie op.

10.3.1 Experimenten en vragen

Het matlab programma `stab.m` is hier te vinden: `NW/NW/nulpt/matlab/index.html`.

- Voer het programma uit in matlab.
- Hoe snel is de convergentie van Newton-Raphson hier?
- De convergentie van de substitutiemethode is lineair. Wat is de convergentiefactor?
- Vanaf welke iteratiestap geraakt de substitutiemethode vast in een oneindige lus?
- Als matlab met IEEE standaard werkt, wat is dan de machinenauwkeurigheid?
- Verklaar waarom de substitutiemethode slechts 12 cijfers nauwkeurigheid kan halen.
- Kun je ook experimenteel de convergentiefactor berekenen voor de substitutiemethode? Kan je die op de figuur van de fouten aflezen?
- Na hoeveel stappen heeft de versnelling volgens Aitken de maximale nauwkeurigheid bereikt (in de 2 gevallen)?
- Welke is de maximale nauwkeurigheid voor de eerste formule van Aitken?
- Kun je verklaren waarom het niet verder convergeert, terwijl de tweede formule toch de exacte oplossing vindt?

10.4 Conditie van nulpunten

Voor de matlab code zie hier: `NW/NW/nulpt/matlab/index.html`.
Opgelet matlab gebruikt de IEEE standaard dubbele precisie berekeningen.
We beschouwen het voorbeeld van de veelterm

```
pol(x) = (x - 3)^6
```

met een 6-voudig nulpunt $x = 3$.
We gebruiken de matlab routines

- `poly`: met als invoer een vector van nulpunten geeft dit als resultaat een vector van de coëfficiënten van de veelterm waarvan dit de nulpunten zijn.
- `roots`: dit is de inverse functie van `poly`: met als invoer een vector van coëfficiënten van een veelterm geeft dit als uitvoer de vector van nulpunten van deze veelterm.

Vertrekt men van een vector `v = [3,3,3,3,3,3]`, en doet men `poly(v)` dan zal men een exact resultaat krijgen omdat het over reële getallen gaat.
Past men op dat resultaat terug `roots` toe, dan zal men niet v terugkrijgen omwille van de afrondingsfouten. Men krijgt complexe wortels op een cirkel rond 3:

3.00888640598399
3.00444024498049 + 0.00769596696125i
3.00444024498049 − 0.00769596696125i
2.99555660203936 + 0.00769050371569i
2.99555660203936 − 0.00769050371569i
2.99111989997631

Men zou dit kunnen verklaren door de onstabiliteit van de methode die in `roots` is geïmplementeerd.
Het blijkt echter dat dit in feite conditie is. Inderdaad, als men met maple hetzelfde uitvoert (werkt standaard met 10 decimale cijfers), dan krijgt men:

```
> restart:with(plots):
> p:=(z-3)^6;
```

$$p := (z-3)^6$$

```
> p:=expand(p);
```

$$p := z^6 - 18z^5 + 135z^4 - 540z^3 + 1215z^2 - 1458z + 729$$

```
> zerosp:=fsolve(p,z,complex);
```

$$zerosp := 3., 3., 3., 3., 3., 3.$$

```
> q:=p+10.0^(-6); # kleine wijziging op de constante coefficient
```

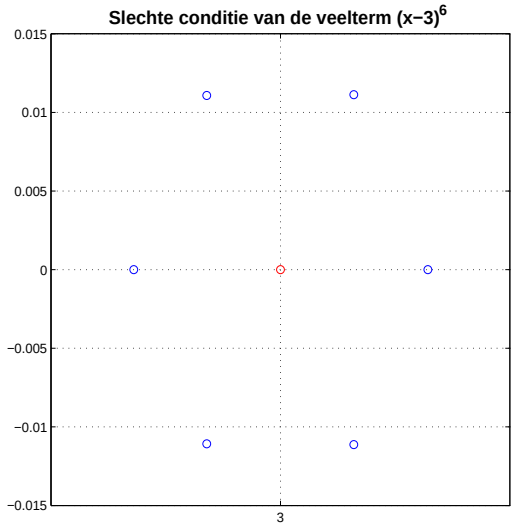
$$q := z^6 - 18 * z^5 + 135 * z^4 - 540 * z^3 + 1215 * z^2 - 1458 * z + 729.0000010$$

```
> zerosq:=fsolve(q,z,complex);
```

$$\begin{aligned} \text{zerosq} := \quad & 2.913397460 - .05000000000\ \text{I}, \\ & 2.913397460 + .05000000000\ \text{I}, \quad 3.000000000 - .10000000000\ \text{I}, \\ & 3.000000000 + .10000000000\ \text{I}, \quad 3.086602540 - .05000000000\ \text{I}, \\ & 3.086602540 + .05000000000\ \text{I} \end{aligned}$$

```
> complexplot([zerosq],style=point,symbol=circle);
```

Men krijgt dan een tekening die men ook in matlab kan genereren:



Het is wel degelijk een probleem van conditie.

De Wilkinson veelterm

Deze veelterm is de veelterm van graad 20 met als nulpunten $x = 1, 2, \ldots, 19, 20$.
De bovenstaande matlab-test is heel eenvoudig te schrijven als `roots(poly(1:20))`.

10.4.1 Experimenten en vragen

Het matlab programma `condvlt.m` is hier te vinden: `NW/NW/nulpt/matlab/index.html`.
Het maple programma staat hier: `NW/NW/nulpt/maple/index.html`.

- Welke methode gebruikt roots om alle nulpunten van een veelterm te vinden?
- Welk verband is er tussen de perturbatie die hierboven in maple wordt opgelegd aan de coëfficiënten en de perturbatie op de wortels. Met andere woorden hoe groot is het conditiegetal? Kan je dat verklaren?
- Is dit een constante voor alle perturbaties?
- Ga de conditie na van een dubbele wortel.
- Waarom is de Wilkinson-veelterm zo slecht geconditioneerd?
- Doe de Wilkinson-veelterm test ook eens in maple.

10.5 Complexe nulpunten van een reële veelterm

10.5.1 Methode van Newton

Voor het zoeken van nulpunten van een veelterm kan men Newton-Raphson gebruiken waarbij men de functiewaarde en de waarde van de afgeleide met een Hornerschema kan berekenen. Een Hornerschema vergt $nO + nV$ bewerkingen.

Hornerschema's

Een reële veelterm kan echter complexe nulpunten hebben. Men moet dan een complexe startwaarde nemen en complex rekenen. Een complexe vermenigvuldiging vraagt 4 reële V + 2 reële O. Een totaal van $4nV + 2nO$ per Hornerschema. Dit kan veel efficiënter met het dubbelrijïg Hornerschema: $2nV + 2nO$.

$$\begin{aligned} p_n(x) &= q_{n-2}(x)(x_2 - \rho x - \mu) + b_{n-1}(x - \rho) + b_n \\ p_n(x) &= a_0 x_n + a_1 x_{n-1} + \cdots + a_{n-1}x + a_n \\ q_{n-2}(x) &= b_0 x_{n-2} + b_1 x_{n-3} + \cdots + b_{n-3}x + b_{n-2} \end{aligned}$$

$$\begin{array}{c|cccccccc} & a_0 & a_1 & a_2 & a_3 & \cdots & \cdots & a_{n-1} & a_n \\ \rho & & \rho b_0 & \rho b_1 & \rho b_2 & \cdots & \cdots & \rho b_{n-2} & \rho b_{n-1} \\ \mu & & & \mu b_0 & \mu b_1 & \cdots & \cdots & \mu b_{n-3} & \mu b_{n-2} \\ \hline & b_0 & b_1 & b_2 & b_3 & \cdots & \cdots & b_{n-1} & b_n \end{array}$$

$$b_k = a_k + \rho b_{k-1} + \mu b_{k-2}, \quad k = 0, 1, \ldots, n; \quad b_{-1} = b_{-2} = 0.$$

10.5.2 Methode van Bairstow

In de methode van Bairstow probeert men het stelsel

$$b_{n-1}(\rho, \mu) = 0, \quad b_n(\rho, \mu) = 0$$

op te lossen. Dit kan met de methode van Newton voor stelsels.
De Jacobiaan vereist de partiële afgeleiden van b_{n-1} en b_n naar ρ en μ. Deze kan men bekomen door de recursie

$$b_k = a_k + \rho b_{k-1} + \mu b_{k-2}, \quad k = 0, 1, \ldots, n$$

af te leiden naar ρ en μ, en dit geeft een recursie voor de afgeleiden. Dit vereist een tweede dubbelrijïg Hornerschema.

10.5.3 Experimenten en vragen

De matlab code voor `brstw` kan je hier vinden `NW/NW/nulpt/matlab/index.html`.

- Reken de recursieformule na voor de partiële afgeleiden van b_{n-1} en b_n naar ρ en μ. Toon aan dat de partiële afgeleide van b_k naar ρ gelijk is aan c_{k-1} en de partiële afgeleide van b_k naar μ gelijk is aan c_{k-2}, waarbij

  $$c_k = b_k + \rho c_{k-1} + \mu c_{k-2}, \quad k = 0, 1, \ldots, n, \quad \text{met } c_{-2} = c_{-1} = 0.$$

  De determinant van de Jacobiaanmatrix is bijgevolg

  $$D = c_{n-2}^2 - c_{n-1} c_{n-3}.$$

  Reken nu de iteratieformules na voor de methode van Bairstow die in het programma zijn gebruikt.
- Hoeveel werk vraagt een iteratiestap van Bairstow ongeveer?
- Hoe snel convergeert de methode van Bairstow?
- Welk stopcriterium is er gebruikt in de code?
- Welke startwaarde wordt er voor r en u gebruikt? Kan je ook de r=0.9r en u=0.9u verklaren?
- Wat doet het matlabcommando `brstw(poly(1:10))`? Is het resultaat zoals verwacht? Wat krijg je voor `brstw(poly(1:16))`? Is dit ok?
- Zijn de laatst berekende wortels even nauwkeurig als de eerste? Kan je hier iets aan doen?
- Worden de wortels in stijgende volgorde van grootte gevonden?

10.5.4 Extra

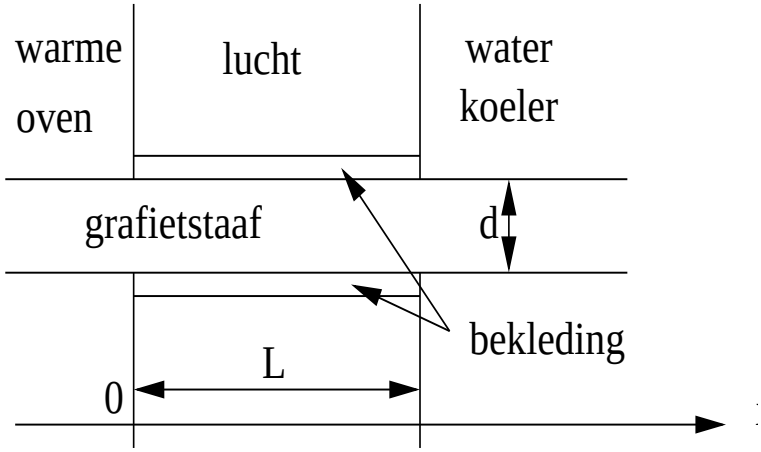
- Ga na dat Newton-Raphson met het dubbelrijïg Hornerschema om complexe wortels te vinden van een reële veelterm eigenlijk equivalent is met de methode van Bairstow, zodanig dat de uiteindelijke optimale implementatie voor beiden nagenoeg hetzelfde is.

10.6 Koeling grafietelektrode

We beschrijven een randwaardeprobleem waarbij een oplossing gevonden wordt door het zoeken van een nulpunt van een transcendente functie.

10.6.1 Beschrijving van het probleem

We beschouwen een grafietelektrode zoals aangegeven op bijgaande figuur.



Links steekt een grafietstaaf in een hete oven. Rechts wordt ze gekoeld door water. Tussen de oven en de koeler is ze geïsoleerd maar gaat ze over een afstand L door lucht op kamertemperatuur.

We brengen een x-as aan van links naar rechts en leggen de oorsprong daar waar de staaf de oven verlaat. De temperatuur in de staaf is een functie $T(x)$ van x.

Wegens de isolerende bekleding in de lucht met temperatuur T_o is het warmteverlies daar degeven door $\beta(T - T_0)$. Hierin is $\beta = 4U/d$ met d de diameter van de staaf en U de warmteoverdrachtscoëfficiënt tussen de lucht en de elektrode.

De temperatuursafhakelijke geleidbaarheid van grafiet is $k_0 - \alpha T$,

De temperatuur T zal als functie van x aan volgende differentiaalvergelijking voldoen.

$$(k_0 - \alpha T)\frac{d^2T}{dx^2} - \alpha\left(\frac{dT}{dx}\right)^2 - \beta(T - T_o) = 0.$$

Het debiet aan warmte afgestaan in de koeler wordt gegeven door

$$Q = -(k_0 - \alpha T)\left(\frac{\pi d}{4}\right)^2 \frac{dT}{dx} \quad \text{in } x = L.$$

Gevraagd wordt hoe groot Q is.

De gegevens zijn

$U = 1.7$ W°/C m^2
$k_0 = 152.6$ W°/C m^2
$\alpha = 0.056$ W(°/C)^{2}m^2
$d = 10$ cm
$L = 25$ cm
$T_o = 20$°C
$T(0) = 1600$°C
$T(L) = 160$°C

10.6.2 Oplossing

Een mogelijke manier om het probleem op te lossen gaat als volgt. We stellen $u = dT/dx$, dan is (formeel) $1/dx = u/dT$, zodat

$$\frac{d^2T}{dx^2} = \frac{d}{dx}\left(\frac{dT}{dx}\right) = \frac{du}{dx} = u\frac{du}{dT}.$$

Zodat de warmtevergelijking wordt

$$\frac{1}{2\alpha}(k_0 - \alpha T)\frac{d(u^2)}{dT} - u^2 - \frac{\beta}{\alpha}(T - T_o) = 0$$

of

$$\frac{d(u^2)}{dT} - \frac{2\alpha}{k_0 - \alpha T}u^2 = 2\beta\frac{T - T_o}{k_0 - \alpha T}$$

of nog

$$\frac{d}{dT}\left((k_0 - \alpha T)^2u^2\right) = 2\beta(k_0 - \alpha T)(T - T_o),$$

zodat na integratie

$$(k_0 - \alpha T)^2u^2 = \beta(k_0 - \alpha T)(T - T_o)^2 + \frac{\alpha\beta}{3}((T - T_o)^3 + C$$

waarin C een integratieconstante is. We lossen op naar $1/u = dx/dT$ en dat levert na integratie

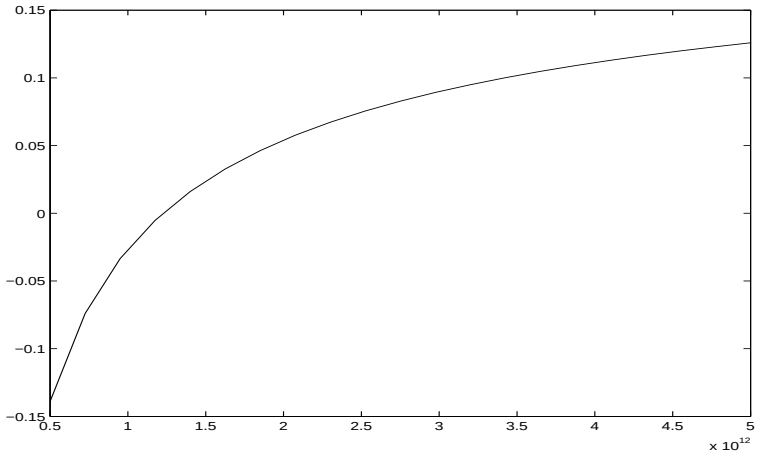
$$L = \int_0^L dx = -\int_{T(0)}^{T(L)} \frac{k_0 - \alpha T}{\sqrt{C + \beta(k_0 - \alpha T)(T - T_o)^2 + \frac{\alpha\beta}{3}(T - T_o)^3}}dT.$$

(Merk op dat de integrand positief is, wegens $T(0) > T(L)$ is de integraal negatief. Vandaar het minteken want L is natuurlijk positief.) We kunnen dus hieruit C berekenen door een nulpunt te zoeken van de functie

$$F(C) = L + \int_{T(0)}^{T(L)} \frac{k_0 - \alpha T}{\sqrt{C + \beta(k_0 - \alpha T)(T - T_o)^2 + \frac{\alpha\beta}{3}(T - T_o)^3}}dT.$$

Als we deze functie willen evalueren moeten we eerst de integraal uitrekenen. Voor een vaste C kunnen we dat bv. doen met de trapeziumregel. Het berekenen van de afgeleide $F'(C)$ is hier om duidelijke redenen niet aan te raden. We kiezen daarom voor het Dekker-Brent algoritme. We moeten ook een startinterval hebben. Gezien de grootte van de getallen die onder het wortelteken in de noemer voorkomen ongeveer 10^{10} is, heeft het niet veel zin om kleine waarden van C te gaan proberen. We gebruiken als startinterval voor C het voldoende grote interval $[0, 10^{15}]$. Het Dekker-Brent algoritme vindt hiermee zonder problemen het nulpunt. De waarde is voor onze gegevens $C = 1.2244e + 12$.

Als we de functie in de buurt van het nulpunt tekenen, dan zie je dat er ook geen problemen te verwachten zijn.



Nu de constante C gevonden is, kunnen we u berekenen als

$$u(T) = -\frac{\sqrt{C + \beta(k_0 - \alpha T)(T - T_o)^2 + \frac{\alpha\beta}{3}(T - T_o)^3}}{k_0 - \alpha T}.$$

Merk op dat we een minteken gebruiken omdat we weten dat u negatief moet zijn. De gevonden waarde van C levert $u = -7.7041e + 03$.

Dit kunnen we uiteindelijk gebruiken om Q te berekenen:

$$Q = -(k_0 - \alpha T(L)) \left(\frac{\pi d}{4}\right)^2 u(T(L)).$$

De uiteindelijke uitkomst is $Q = 6.8262e + 03$.

10.6.3 Matlab experimenten

Het matlabprogramma is te vinden op `NW/NW/grafiet/matlab/index.html`

- Waarom zou het Dekker-Brent algoritme zo goed werken op dit voorbeeld?
- In matlab zit ook een routine `fsolve`. Doe `help fsolve` en ga na hoe je dit moet gebruiken. Probeer ook eens het probleem op te lossen met behulp van fsolve. Levert dit ook zo een vlugge convergentie?

10.7 Stabiliteit van een lineair systeem

10.7.1 Beschrijving van het probleem

Bij een digitaal systeem is er een invoer u(1), u(2),... die wordt omgezet in een uitvoer y(1), y(2),...

Als we als invoer een impuls geven: u(1)=1, u(k)=0, k=1,2,... dan noemen we de uitvoer h(1)=y(1), h(2)=y(2), ... de impuls responsie van het systeem.

We veronderstellen dat het systeem lineair is en stationair. In dat geval kan je in het algemeen de uitvoer beschrijven voor een willekeurige invoer als y = h*u (de rij y(k) is de convolutie van de rijen u(k) en h(k)). Het is echter veel eenvoudiger om de rijen om te zetten in reeksen via de z-transformatie:

$$H(z) = \sum_{k=0}^{\infty} h(k)z^{-k}, \quad U(z) = \sum_{k=0}^{\infty} u(k)z^{-k}, \quad Y(z) = \sum_{k=0}^{\infty} y(k)z^{-k}$$

en dan wordt de convolutie omgezet in een product: Y(z)=H(z)U(z).

Meestal neemt men een rationale benadering voor H(z). Een rationale functie heeft echter polen die voor een onstabiliteit kunnen zorgen. Bijvoorbeeld een transferfunctie

$$H(z) = 1/(z-a) = \frac{1}{z} + \frac{a}{z^2} + \frac{a^2}{z^3} + \cdots$$

heeft een pool $z = a$, en een impuls respons $1, a, a^2, a^3, \ldots$. Als $|a| < 1$ dan gaat dit naar nul, zoals men van elk praktisch systeem kan verwachten. Als $|a| > 1$, dan groeit de respons in de loop van de tijd en ontploft het systeem tenslotte. Een dergelijk model voor ons systeem willen we natuurlijk vermijden. We willen enkel stabiele systemen.

Algemeen is een dergelijk systeem met rationale transferfunctie stabiel als alle polen in absolute waarde kleiner zijn dan 1.

10.7.2 Een voorbeeld

We hebben voor een robotarm een benadering van de transferfunctie opgesteld die aanleiding geeft tot een rationale functie van graad 6. $H(z) = B(z)/A(z)$ met $B(z)$ en $A(z)$ veeltermen van graad 6. Stel dat de coëfficiënten van A zijn $a_0, a_1, \ldots, a_6$ en die van B zijn $b_0, b_1, \ldots, b_6$, dan kan men de uitvoer y berekenen als functie van de invoer en de vorige uitvoer.

$$a_0 y(n) = b_0 u(n) + b_1 u(n-1) + ... + b_6 u(n-6) - a_1 y(n-1) - \cdots - a_6 y(n-6).$$

Dit is hetzelfde als $A(z)Y(z) = B(z)U(z)$ of $Y(z) = [B(z)/A(z)]U(z)$. Er is een functie `filter(B,A,u)` in de matlab signal processing toolbox die deze berekeningen maakt. (Doe `help filter` om te zien of je deze functie ter beschikking hebt.)

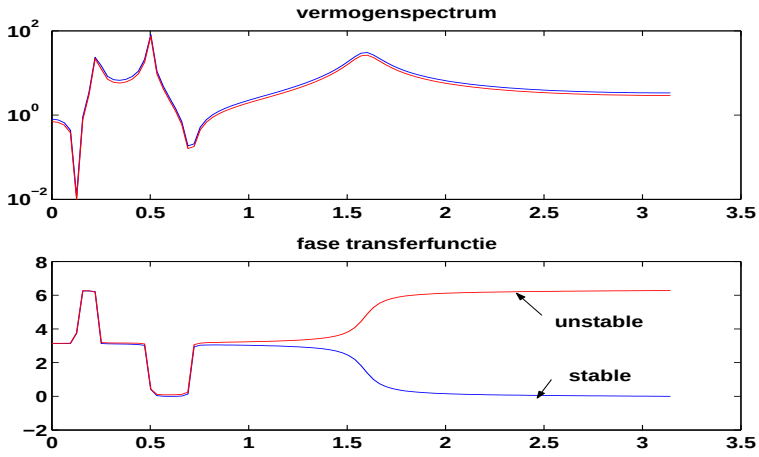
Merk op dat de orde van het systeen, dat is de graad van de rationale transferfunctie, bepaalt hoeveel van de vorige in- en uitvoer men in het geheugen moet bewaren om de volgende uitvoer te kunnen bepalen. Voor deze simulatie in matlab heeft dat niet veel belang maar als men dit model in hardware gaat uitvoeren is dit wel belangrijk.

Met een benaderingstechniek hebben we een teller en noemer gevonden die uit de volgende getallen bestaan

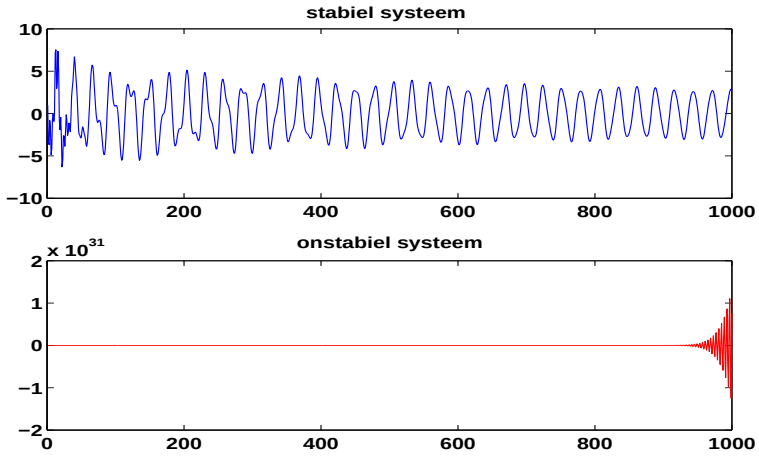
```
A = [1.0000   -3.6601    6.4109   -7.7328    7.0727   -4.2078    1.1447];
B = [1.0000   -8.0361   21.8663  -29.6238   21.7272   -7.9265    0.9736];
```

De transferfunctie $H(z)$ met z een complex getal dat op de complexe eenheidscirkel ligt is

een complexe functie, die hierbij geplot wordt (modulus en argument) als een rode curve.



Als we echter voor een input die een stap functie is (u(k)=1 voor k=1:10 en u(k)=0 voor k=11:1000), dan blijkt dat een onstbiel systeem te zijn.



Als we de polen van het systeem berekenen dan zien we dat die inderdaad bestaan uit paren complex toegevoegde getallen, waarvan 1 paar buiten de eenheidscirkel ligt.

```
poles = [1.0730    1.0730    0.9980    0.9980    0.9991    0.9991]
```

Als we dat paar vervangen door zijn inverse (a vervangen door $1/a$) dan is de modulus van de transferfunctie hetzelfde, de fase is nagenoeg hetzelfde, maar de uitvoer is nu (lichtjes) uitstervend, in ieder geval stabiel. Alle resultaten voor de stabiele vorm zijn in het blauw getekend op de vorige figuren.

10.7.3 Matlab experimenten

Het matlabprogramma is te vinden op `NW/NW/systeem/matlab/index.html`

- Beschik je over de routine **filter**? Zoniet, kan je ze dan zelf schrijven?
- Probeer eens andere systemen om te zien wat onstabiliteiten als uitvoer veroorzaken.
- Hoe zou je de impulsresponsie van het systeem kunnen laten tekenen?
- Hoe verandert de invoer het gedrag van de uitvoer?

- Hier wordt de methode van Bairstow gebruikt om de polen te zoeken. Is dit een goed idee, of zou je een andere methode voorstellen?
- Probeer de nultpunten ook eens met Newton te bepalen. Zijn ze even nauwkeurig? Gaat dit even efficiënt?
- Geeft de ingebouwde routine van matlab **roots** dezelfde resultaten?
- Welke methode wordt er in **roots** gebruikt?
- Wat doet de functie **unwrap** die hier gebruikt is?

Hoofdstuk 11

Stelsels vergelijkingen

11.1 Niet-lineaire vergelijkingen

We beschouwen het volgende stelsel

$$\begin{aligned} u(x,y) &= xy - 1 \\ v(x,y) &= x^2 + y^2 - 4 \end{aligned}$$

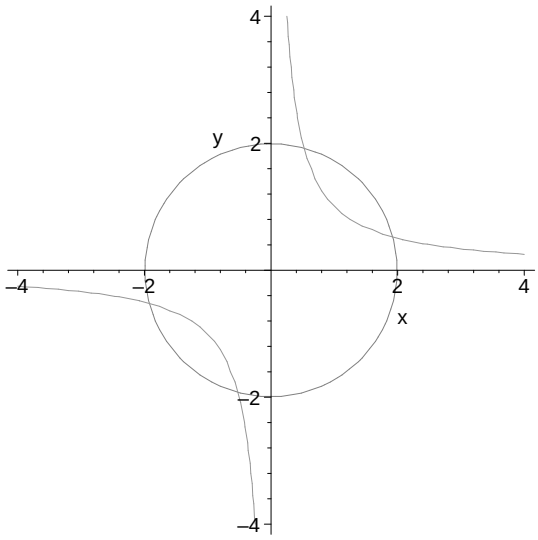
De Jacobiaan is

$$J = \begin{bmatrix} y & x \\ 2x & 2y \end{bmatrix}.$$

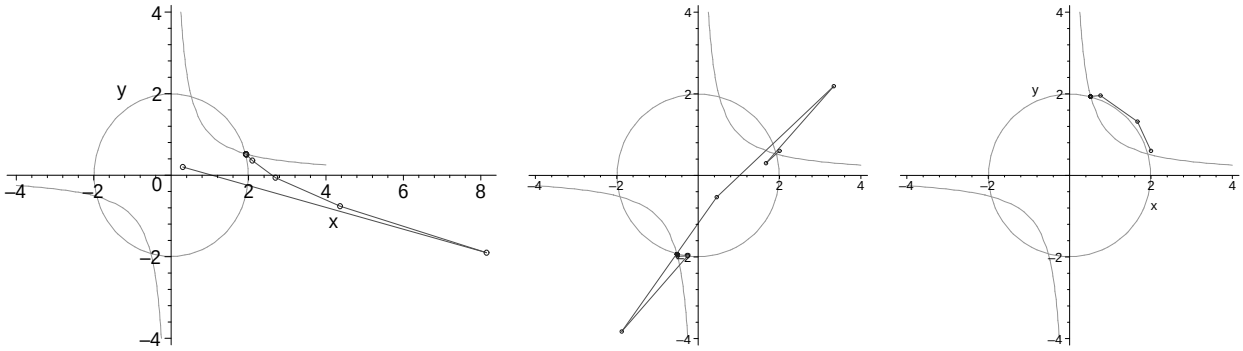
De oplossingen zijn $(+x_e, +y_e)$, $(-x_e, -y_e)$, $(+y_e, +x_e)$ en $(-y_e, -x_e)$ met

$$\begin{aligned} x_e &= 0.51763809020504152469779767 5248 \\ y_e &= 1.93185165257813657349948639946 \end{aligned}$$

Dit is weergegeven op de volgende figuur:



$u(x,y) = 0$ is de hyperbool
$v(x,y) = 0$ is de cirkel.
Er zijn 4 snijpunten (oplossingen van het stelsel).



11.1.1 Newton-Raphson

We gebruiken Newton-Raphson met als startwaarden

$$(x_0, y_0) = (0.3, 0.2)$$

De vorige figuur links toont het resultaat. Na ongeveer 6 stappen is de methode in het punt (y_e, x_e) aanbeland.

11.1.2 Vereenvoudigde Newton-Raphson methodes

Totale stap

We gebruiken nu de totale stap methode met een startwaarde

$$(x_0, y_0) = (2, 0.6)$$

die dicht ligt bij de oplossing $(+y_e, +x_e)$.
Dit blijkt een afstotingspunt te zijn en er is convergentie naar de oplossing $(-x_e, -y_e)$. Zie vorige figuur midden.

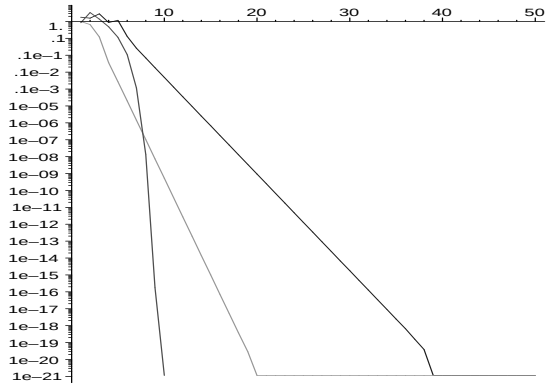
Enkelvoudige stap

Gebruiken we de enkelvoudige stap methode met dezelfde startwaarde dan blijkt $(+y_e, +x_e)$ weer een afstotingspunt te zijn maar nu is er convergentie naar $(+x_e, +y_e)$ zoals de bovenstaande figuur rechts weergeeft.

Vergelijking van de 3 methoden

Als we de 3 methoden vergelijken dan merken we voor Newton-Raphson een snelle (kwadratische) convergentie, terwijl voor de vereenvoudigde methoden er slechts lineaire convergentie is.
Op de onderstaande figuur is de relatieve fout gegeven voor

Newton-Raphson (de gekromde curve)
totale stap methode (traagst convergerende rechte lijn)
enkelvoudige stap (snelst convergerende rechte lijn)
op logaritmische schaal in functie van het aantal iteratiestappen.



11.1.3 Experimenten

De maple code kan je hier vinden: `NW/NW/itstelsel/maple/index.html`.

- Probeer eens met andere startwaarden. Zal Newton-Raphson een voorkeur hebben voor 1 van de vier nulpunten?
- Is dit zo voor de vereenvoudigde methoden?
- Is de enkelvoudige stap methode steeds vlugger dan de totale stap methode?
- Kan je de convergentiefactor aflezen uit de tekening voor de relatieve fout van de vereenvoudigde methoden?
- Is die constant voor een bepaalde methode of is die probleemafhankelijk?

11.1.4 Extra

Matlab code `nrst.m` is te vinden op: `NW/NW/itstelsel/matlab/index.html`.

- Zou je zelf de 3 methoden in matlab ook zo implementeren? Zijn al de methoden wel efficiënt geïmplementeerd?
- Doe tijdsmetingen en tel het aantal flops. Wat is je besluit over de efficiëntie van de verschillende methoden?
- Pas de methoden ook toe op stelsels met meer dan 2 vergelijkingen. Welke conclusies kan je hier trekken?

Voor een java applet voor het voorbeeld van hierboven zie: `NW/NW/itstelsel/java/index.html`.

11.2 Complexe vergelijkingen

Complexe vergelijkingen vormen een bijzonder geval van het vorige.

Om een complexe vergelijking $f(z) = 0$ op te lossen kunnen we z als $z = x + iy$ schrijven en het reële en imaginaire deel van f met $u(x, y)$ en $v(x, y)$ aangeven.

Om de (complexe) nulpunten van f te vinden moeten we dus het stelsel

$$u(x,y) = 0$$
$$v(x,y) = 0$$

oplossen.

Dit zijn 2 niet-lineaire vergelijkingen met 2 onbekenden. Dit kunnen we met Newton-Raphson oplossen.

De formule is

$$\begin{aligned} x(k+1) &= x(k) - [u(x(k),y(k))u'(x(k),y(k)) + v(x(k),y(k)))v'(x(k),y(k))]/D \\ y(k+1) &= y(k) - [v(x(k),y(k))u'(x(k),y(k)) - u(x(k),y(k))v'(x(k),y(k))]/D \\ D &= [u'(x(k),y(k))]^2 + [v'(x(k),y(k))]^2. \end{aligned}$$

Beschouw nu de vergelijking

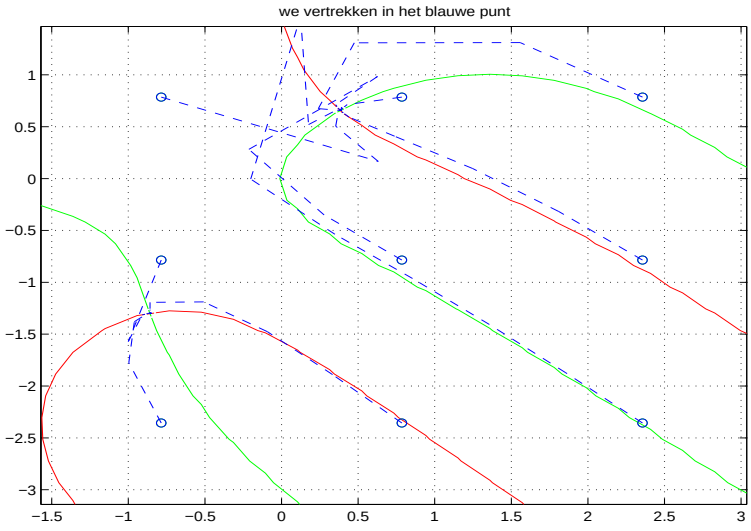
$$f(z) = z - e^{az}$$

Indien $z = x + iy$ en $a = c + id$, dan krijgen we

$$\begin{aligned} u(x,y) &= x - e^{(cx-dy)}\cos(dx+cy) \\ v(x,y) &= y - e^{(cx-dy)}\sin(dx+cy) \end{aligned}$$

Voor $a = 1 + i$ krijgen we de volgende figuur als we de impliciet gedefinieerde krommen $u(x,y) = 0$ en $v(x,y) = 0$ tekenen.

De reële en imaginaire delen van de wortels corresponderen met de snijpunten van de twee krommen op de figuur.



De demonstratie laat zien hoe er convergentie is naar de verschillende nulpunten vanuit negen verschillende startwaarden.

11.2.1 Experimenten en vragen

De matlab code kan je hier vinden: `NW/NW/itstelsel/matlab/index.html`.

- Probeer nog eens met andere startwaarden.
- Het blijkt dat elke wortel een gebied heeft van waaruit de iteratie aangetrokken wordt. Zou je een idee kunnen geven over het aantrekkingsgebied van de onderste wortel?
- Experimenteer ook eens met andere waarden van a (je kan die opgeven). Verandert de waarde van a veel aan de figuur?
- Men kan een complexe vergelijking oplossen met Newton op verschillende manieren.
  - Ofwel gebruikt men de complexe Newton-Raphson formule. Daarbij moet men complex rekenen.
  - Of men kan deze complexe iteratie splitsen in een reëel en een imaginair stuk.
  - Of men kan de Newton-Raphson methode gebruiken voor het stelsel $(u(x,y), v(x,y)) = (0,0)$.

  Toon aan dat de tweede en de derde manier eigenlijk dezelfde iteratie oplevert.
- Programmeer ook de "gewone" complexe Newton-Raphson methode om het probleem op te lossen.

```
for i = 1:nmax
  za=abs(z); dz=f(z)/df(z); z=z-dz;
  if abs(dz) < eps*za, return, end
end
```

  (zie `cmpnr.m` voor de matlab-code). Is de uitvoering in de praktijk even snel en even efficiënt als het opsplitsen in een stelsel?
- Probeer hier ook eens de enkelvoudige stap methode. In welke volgorde moet men de vergelijkingen schrijven opdat er convergentie zou zijn naar het eerste of het tweede nulpunt?
- Los op analoge manier de complexe vergelijking $f(z) = 0$ op met

$$f(z) = 2z^2 + z + 1.$$

  Splits in reëel en imaginair deel en programmeer dit in matlab.

11.2.2 Extra

- Wie was Raphson?
- Om reële of complexe wortels van een reële veelterm te vinden kan men gebruik maken van
  - De Newton-Raphson methode met het dubbelrijïg Hornerschema
  - De methode van Bairstow

  Toon aan dat als men de bedenking 2 pag. 92 in rekening brengt en de waarde van de afgeleide ook met een dubbelrijïg Hornerschema berekent, dan de methode van Newton en van Bairstow samenvallen. De matlab code `brst.m` voor de methode van Bairstow kan je ook vinden op `NW/NW/itstelsel/matlab/index.html`.
- Controleer het algoritme. Hoe komt men aan de formules voor de Jacobiaan?

- Door `brst(poly(1:n))` kan men de nulpunten zoeken van een veelterm met als nulpunten $1, 2, \ldots, n$. Hoe groot mag n zijn opdat het algoritme nog zou werken? Weet je ook waarom?
- Wie was Bairstow?

11.3 Lineaire stelsels iteratief

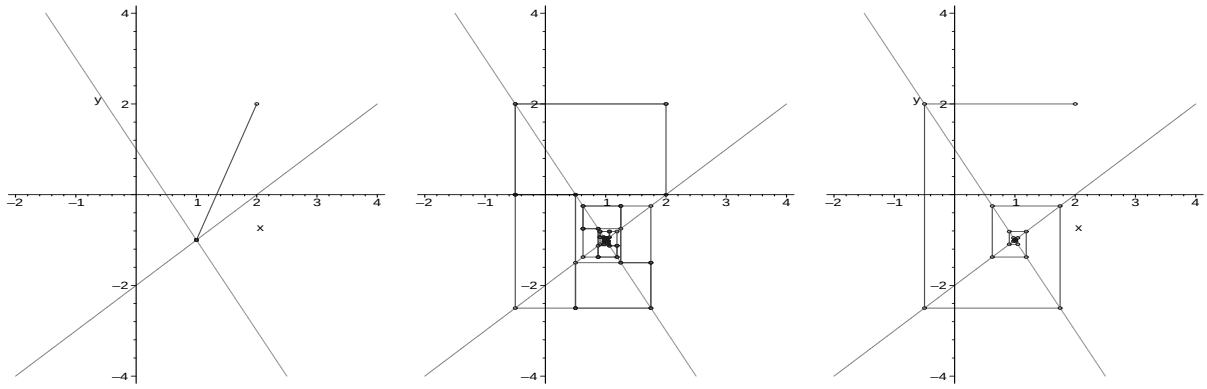
We beschouwen het volgende stelsel

$$\begin{aligned} 2x + y &= 1 \\ x - y &= 2 \end{aligned}$$

De exacte oplossing is $x_e = 1$ en $y_e = -1$.

11.3.1 De methode van Newton-Raphson

Met de Newton-Raphson methode krijgt men in één stap de oplossing omdat de lineaire benadering van een lineaire afbeelding exact is. Om de correctie te berekenen moet men echter het stelsel oplossen dat men wil oplossen. Dit helpt ons dus geen stap vooruit.
Onderstaande figuur links toont die stap, vertrekkende uit de startwaarde $(x_0, y_0) = (2, 2)$.



11.3.2 De vereenvoudigde Newton methoden

Totale stap = Jacobi

Gebruiken we de totale stap methode met dezelfde startwaarde dan blijkt er convergentie te zijn.

De opeenvolgende punten zijn

$$(2, 2), (-0.5, 0), (0.5, -2.5), (1.75, -1.5), (1.25, -0.25), (0.625, -0.75), \ldots$$

zoals aangegeven op bovenstaande figuur in het midden.

Enkelvoudige stap = Gauss-Seidel

Gebruiken we de enkelvoudige stap methode met dezelfde startwaarde dan blijkt er convergentie te zijn.

De opeenvolgende punten zijn

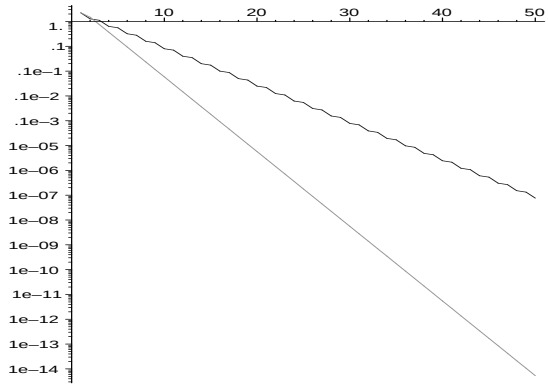
$$(2,2),(-0.5,2),(-0.5,-2.5),(1.75,-2.5),(1.75,-0.25),(0.625,-0.25),\ldots$$

zoals aangegeven op bovenstaande figuur rechts.

Vergelijking van de 2 vereenvoudigde methoden

Als we de 2 laatste methoden vergelijken dan merken we voor dit voorbeeld een lineaire convergentie. De Gauss-Seidel methode heeft een kleinere convergentiefactor.

Op de onderstaande figuur is de relatieve fout voor Jacobi (traagste convergentie) en Gauss-Seidel (snelste convergentie)



op logaritmische schaal in functie van het aantal iteratiestappen weergegeven.

11.3.3 Experimenten

De matlab code `itldemo.m` kan je hier vinden: `NW/NW/itstelsel/matlab/index.html`.
Voor een java applet voor Jacobi en Gauss-Seidel zie: `NW/NW/itstelsel/java/index.html`.

- Probeer eens met andere startwaarden. Is er steeds convergentie?
- Wordt de convergentie bepaald door de helling van de 2 rechten? Hoe? Voor welke configuratie is de convergentie het snelste?
- Probeer eens wat er gebeurt als je de 2 vergelijkingen van plaats verwisselt. Heb je altijd dit convergentiegedrag volgens een "vierkante spiraal"'? Wanneer heb je "monotone convergentie", waarbij x en/of y monotoon naar de oplossing gaat?
- Welke lijnen zijn in de routines `jacobi` en `gaussei` enkel toegevoegd om louter demonstratieredenen en kunnen voor een praktisch gebruik dus weg?
- De routines `jacobi` en `gaussei` bevatten geen stopcriterium (tenzij het bereiken van `kmax` stappen). Bedenk een goed stopcriterium en test het uit.
- Probeer wat er gebeurt als je de 2 vergelijkingen van plaats verwisselt.

11.3.4 Extra

- Pas de methode ook eens toe op een groter stelsel.
- Voeg zelf de code toe om de relatieve fout in functie van de iteratiestap te tekenen op logaritmische schaal. Wat kan je afleiden over de convergentiesnelheid? Probeer dit ook voor een groter stelsel. Blijft het besluit over de snelheid dan hetzelfde?
- Doe tijdsmetingen en tel het aantal flops. Wat is je besluit over de efficiëntie van de beide methoden?

11.3.5 De SOR methode

Voor de Jacobimethode gebruikt men

$$x(k+1) = -D^{-1}[(L+U)x(k) - b]$$

en voor de Gauss-Seidelmethode

$$x(k+1) = -(L+D)^{-1}[Ux(k) - b].$$

Men kan dit laatste omrekenen tot $(Ax^* = b)$

$$\begin{aligned} x(k+1) - x^* &= x(k) - x^* - (L+D)^{-1}[(L+D+U)x(k) - b] \\ e(k+1) &= e(k) - (L+D)^{-1}[Ax(k) - Ax^*] \\ e(k+1) &= [I - (L+D)^{-1}A]e(k) = (L+D)^{-1}Ue(k) \end{aligned}$$

Dus is de convergentiesnelheid voor GS bepaald door $\|G\|$ met $G = (L+D)^{-1}U$. Analoog is de convergentiesnelheid voor Jacobi bepaald door $\|G\|$ met $G = D^{-1}(L+U)$. In de SOR methode voert men een relaxatieparameter ω in zodat

$$x(k+1) = x(k) - \omega(L+D)^{-1}r(k) = (1-\omega)x(k) - \omega(L+D)^{-1}[Ux(k) - b]$$

Herhalen we de vorige berekeningen, dan vinden we

$$e(k+1) = [I - \omega(L+D)^{-1}A]e(k) = [(1-\omega)I - \omega(L+D)^{-1}U]e(k)$$

De convergentiesnelheid voor SOR wordt dan bepaald door $\|G(w)\|$ met $G(w) = [(1-\omega)I - \omega(L+D)^{-1}U]$. Een keuze voor ω die deze norm minimaliseert zal dus een optimale snelheid geven. De berekening van de optimale ω is niet eenvoudig en kan slechts in een paar gevallen expliciet gebeuren. Een voorbeeld is de oplossing van de Laplacevergelijking (zie onder) waar de SOR met optimale parameter duidelijk voor een versnelling van de convergentie zorgt.

11.4 Een toepassing: Laplace vergelijking

11.4.1 Beschrijving van het probleem

We beschouwen een rechthoekige dunne metalen plaat G. Als we de temperatuursverdeling g op de rand ∂G van de plaat kennen, dan wordt de warmteverdeling binnen de plaat gegeven

door de Laplacevergelijking

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad (x,y) \in G$$
$$u\Big|_{\partial G} = g$$

op te lossen.

11.4.2 Discretisatie

We gebruiken de klassieke vijfpuntsmolecule voor discretisatie van de tweede orde partiële afgeleiden. Daardoor wordt de vergelijking

$$u_{ij} - \theta_x(u_{i+1,j} + u_{i-1,j}) - \theta_y(u_{i,j+1} + u_{i,j-1}) = 0, \quad 1 \le i \le n-1, \quad 1 \le j \le m-1$$

Hierin is

$$\theta_x = \frac{k^2}{2(h^2+k^2)} \quad \text{en} \quad \theta_y = \frac{h^2}{2(h^2+k^2)}$$

waarin h de roosterafstand in de x-richting en k de roosterafstand in de y-richting voorstelt. In de onderstaande figuur is het kader de plaat en geeft het koppel de coördinaten aan van het punt dat er links-onder staat.

(0,m)	(1,m)	(2,m)	$\cdots$	(n-2,m)	(n-1,m)	(n,m)
(0,m-1)	(1,m-1)	(2,m-1)		(n-2,m-1)	(n-1,m-1)	(n,m-1)
(0,m-2)	(1,m-2)	(2,m-2)		(n-2,m-2)	(n-1,m-2)	(n,m-2)
$\vdots$	$\vdots$				$\vdots$	$\vdots$
(0,2)	(1,2)				(n-1,2)	(n,2)
(0,1)	(1,1)	(2,1)		(n-2,1)	(n-1,1)	(n,1)
(0,0)	(1,0)	(2,0)	$\cdots$	(n-2,0)	(n-1,0)	(n,0)

De randpunten hebben coördinaten (0,0),..., (n,0), (n,1),..., (n,m), (n-1,m),..., (0,m), (0,m-1),..., (0,0). Daar is de temperatuur gegeven en dit zijn dus bekenden. De onbekenden zijn alle waarden $u_{i,j}$ met $1 \le i \le n-1$ en $1 \le j \le m-1$.

We zullen verder veronderstellen dat $h = k$, zodat $\theta_x = \theta_y = 1/4$.

Deelt men alle vergelijkingen door $\theta_x = \theta_y$, dan worden alle onbekenden als we die in een vector X steken in de lexicografische volgorde $u_{11}, u_{21}, \ldots u_{n-1,1}, u_{12}, \ldots, u_{n-1,m-1}$, gevonden door een stelsel op te lossen van de vorm $AX = F$ waarbij

$$A = \begin{bmatrix} T & -I_{n-1} & & & \\ -I_{n-1} & T - I_{n-1} & & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I_{n-1} \\ & & & -I_{n-1} & T \end{bmatrix} \quad \text{met} \quad T = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{bmatrix}.$$

en het rechterlid F is gedefinieerd door

$$F = \begin{bmatrix} u_{01} + u_{10} \\ u_{20} \\ \vdots \\ u_{n1} + u_{n-1,0} \\ \hline u_{02} \\ 0 \\ \vdots \\ 0 \\ u_{n2} \\ \hline \vdots \\ \hline u_{0,m-1} + u_{1,m} \\ u_{2,m} \\ \vdots \\ u_{n-2,m} \\ u_{n,m-1} + u_{n-1,m} \end{bmatrix}.$$

11.4.3 Oplossen van dit stelsel

Als $n = m = 101$, dan gaat het hier om een stelsel van 10^4 onbekenden. Dit is zeer groot, maar gelukkig is er een zeer sterke structuur in te herkennen en zijn er veel nullen. Dit wijst in de richting van een iteratieve methode om deze lineaire stelsels op te lossen. We gebruiken als iteratieve methode Gauss-Seidel en SOR. Van elk zijn 2 versies voorzien: één die tussenresultaten uitschrijft en één zonder tussenresultaten.

Omdat dit soort matrix wel eens meer voorkomt is de eigenstructuur ervan reeds grondig onderzocht en kan men daardoor een optimale SOR parameter bepalen. In dit geval is die $\omega = 1/(1 + \sqrt{1 - \mu^2})$ waarbij $\mu = 0.5(\cos\frac{\pi}{n+1} + \cos\frac{\pi}{m+1})$.

Als stopcriterium is gekozen voor het eenvoudige testje: norm(correctie)/norm(oplossing) kleiner dan de opgegeven relatieve nauwkeurigheid.

11.4.4 Randvoorwaarden

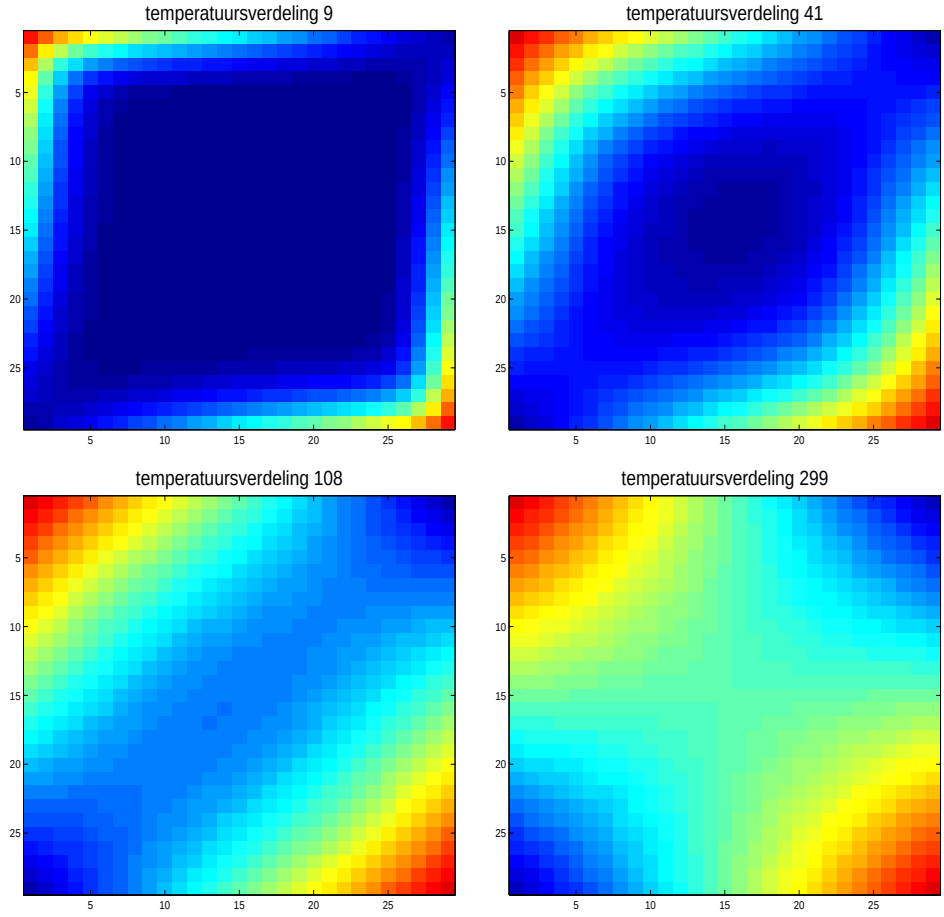
Als mogelijke randvoorwaarden kan men kiezen uit

soort	RVW			
1	$u_{i0} = 0,$	$u_{i,m} = 1,$	$u_{0,j} = j/m,$	$u_{n,j} = j/m$
2	$u_{i0} = i/n,$	$u_{i,m} = 1 - i/n,$	$u_{0,j} = j/m,$	$u_{n,j} = 1 - j/m$
3	$u_{i0} = 1 - i/n,$	$u_{i,m} = 0,$	$u_{0,j} = 1 - j/m,$	$u_{n,j} = 0$

11.4.5 Resultaten

We geven hieronder de oplossing weer in kleur (rood is warm, blauw is koud) voor de randvoorwaarde 2, waarbij de oplossing berekend is na respectievelijk 9, 41, 108, en 299 stappen. De methode is 3 (d.w.z. Gauss-Seidel). Er werden 30 discretisatiepunten genomen in elke

richting.



11.4.6 Experimenten en vragen

De matlabcode kan je hier ophalen: `NW/NW/laplace/matlab/index.html`.

- Welk is die "klassieke vijfpuntsmolecule" waarover sprake.
- Reken de vergelijkingen voor de u_{ij} eens na.
- Is er een verschil in tijd tussen de methode met en zonder overrelaxatie?
- Als in het voorbeeld 30 punten in elke richting genomen werden, hoeveel onbekenden zijn er dan in het stelsel dat men in elke iteratiestap moet oplossen?
- Hoeveel werk vraagt het ongeveer om 1 iteratiestap uit te voeren? (Kijk naar de implementatie in matlab.)
- Is de temperatuursverdeling bij stap 299 de definitieve verdeling? Kan je voorspellen welke de uiteindelijke verdeling moet zijn voor de andere randvoorwaarden? Laat het programma narekenen of jouw verwachtingen kloppen.

11.4.7 Extra

- Zoek op in de literatuur of reken zelf na dat de optimale omega voor relaxatie inderdaad door de opgegeven formule wordt gegeven.

Hoofdstuk 12

Eigenwaarden

12.1 Methode van de machten

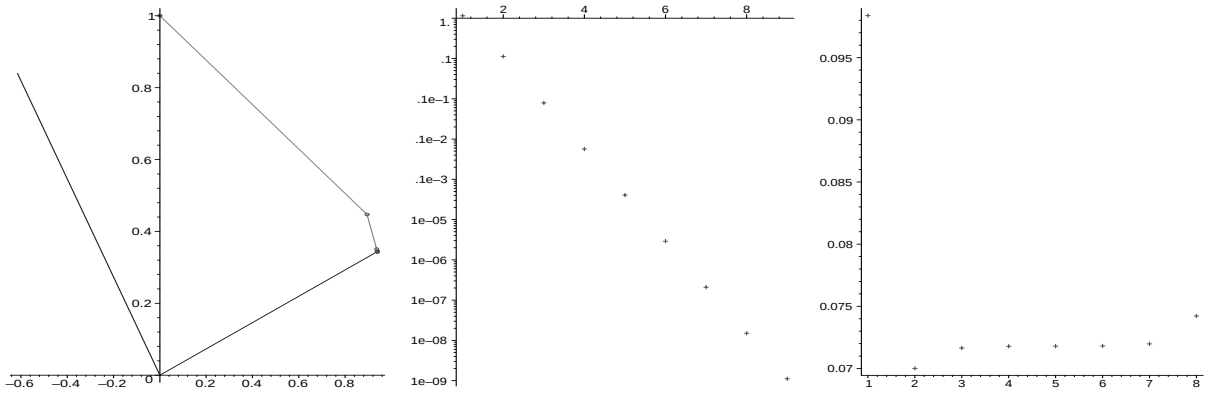
Deze methode wordt ook wel de methode van Von Mises genoemd. Om de methode grafisch te kunnen voorstellen, gebruiken we een 2x2 matrix.

$$A = \begin{bmatrix} 3 & 2 \\ 1 & 1 \end{bmatrix}.$$

De eigenwaarden s en bijhorende eigenvectoren V zijn

$$\begin{aligned} s &:= [3.73205080, 0.2679491925] \\ V_1 &:= [0.9390708016, 0.3437237693]' \\ V_2 &:= [-0.6148101593, 0.8398462963]' \end{aligned}$$

We kiezen een startvector $X_0 = [0\ \ 1]'$ en passen de methode van de machten toe, waarbij we steeds de nieuwe vector normalizeren.
Op de volgende figuur links zie je hoe de opeenvolgende X_k van X_0 naar V_1 (eigenvector horend bij de grootste eigenwaarde) convergeert.



Berekenen we de relatieve fout t.o.v. de exacte eigenvector V_1, en zetten we die uit op een logaritmische schaal, dan zien we duidelijk dat er lineaire convergentie is (zie bovenstaande figuur, midden).

De convergentiefactor is abs(kleinste/grootste eigenwaarde). In dit geval is dat $\rho = 0.071796$. Als we de verhouding van 2 opeenvolgende (relatieve) fouten uitzetten in grafiek,

dan zien we dat lineaire convergentie met deze ρ inderdaad optreedt (zie bovenstaande figuur rechts).

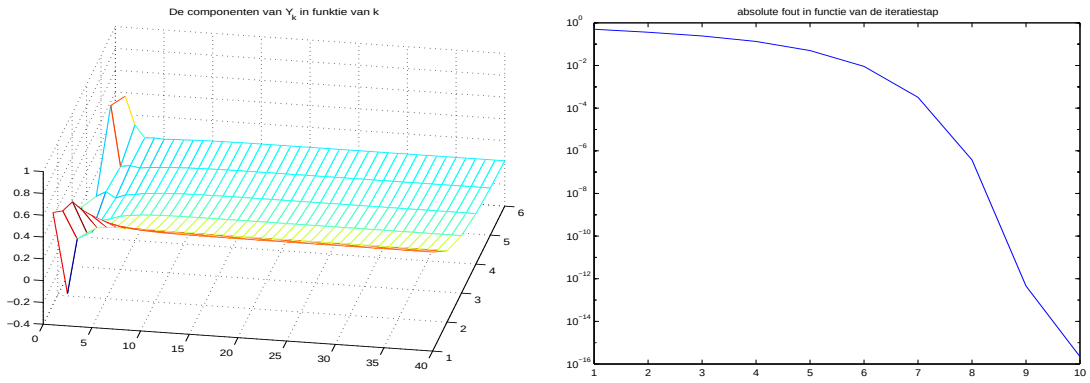
Indien we het aantal stappen opdrijven, dan is er geen verdere convergentie meer als we de standaardnauwkeurigheid (10 decimale cijfers) van maple gebruiken.

Voor een matrix van grotere afmetingen moeten we andere visualisaties bedenken.

Kies een stel van eigenwaarden `s = [2 -1 1 1 1 1]`. Met `poly(s)` kan men de coëfficiënten van een veelterm vinden met als wortels s, en dan is het eenvoudig om de companion-matrix op te stellen. Dat is een Hessenberg matrix met als eigenwaarden de elementen van s. De eerste rij bevat de coëfficiënten van de (monische) veelterm, en de subdiagonaal betaat uit eentjes. In dit geval is dat de matrix

```
h = [ 5    -8     2     7    -7     2;
      1     0     0     0     0     0;
      0     1     0     0     0     0;
      0     0     1     0     0     0;
      0     0     0     1     0     0;
      0     0     0     0     1     0]
```

We kiezen lukraak een startvector en itereren met de methode van de machten. Na 40 iteratiestappen is er nog geen convergentie. De gevonden schatting voor de grootste eigenwaarde (nl. 2) is $s = 2.00000000483753$.
De convergentie is heel traag. De convergentiefactor is 0.5. Er komt slechts 1 cijfer per 3 iteratiestappen bij. Het verloop van de bijhorende (eigen)vector is op de onderstaande figuur links weergegeven.



Van links naar rechts ziet men voor de 40 iteratiestappen de 6 componenten van de vector X_k uitgezet.

12.2 De methode van de inverse machten

Deze methode wordt ook wel eens de methode van Wielandt genoemd.
Met de methode van de inverse machten passen we eenzelfde strategie toe. Nu zijn de gekozen eigenwaarden `s = [ 0.5 -1 1 1 1 1]`.
We gebruiken de berekende norm van de iteratievector ook als shift voor de volgende iteratiestap. Daardoor wordt de methode kwadratisch convergent. We vinden reeds na 10 iteratiestappen dat er convergentie is naar $s = 0.5$. De kwadratische convergentie wordt geïllustreerd door bovenstaande figuur rechts.

12.2.1 Experimenten en vragen

Het maple werkblad kan je hier vinden: `NW/NW/eigwaarden/maple/index.html`.

- Voer het werkblad uit voor meer iteratiestappen. Wat is de maximale nauwkeurigheid die men kan bereiken. Kun je dit verklaren?
- Probeer ook een matrix waarbij de eigenwaarden complex worden. Wat gebeurt er met de iteratiestappen?
- Kan je een stopcriterium voor het algoritme bedenken?

Er zijn ook demos met matlab: de code `eigmdemo.m` (methode van de machten) en `eigidemo.m` (inverse machten met shift) kan je hier vinden: `NW/NW/eigwaarden/matlab/index.html`.

- Welk stopcriterium is hier gebruikt?
- Is er altijd convergentie?
- Welke reden wordt er opgegeven als er geen convergentie is?
- Betekent dit dan dat de eigenwaarde niet gevonden is?
- Als er in de methode van de inverse machten geen shift gebruikt wordt, is dan de methode ook zo snel?
- Wat gebeurt er als er twee eigenwaarden zijn die allebei gelijk zijn aan de grootste absolute waarde van de eigenwaarden?
- Hoe zie je aan de grafiek van de fouten dat er lineaire of kwadratische convergentie is?

12.2.2 Extra

- Kan je de code aanpassen om een andere dan de grootste of de kleinste eigenwaarde te vinden?
- Wie was Von Mises en wie was Wielandt?

12.3 Alternatieve technieken en deflatie

Er zijn alternatieven voor de methode van de machten of de inverse machten om eigenwaarden te berekenen. Dikwijls zijn ze numeriek gezien beter omdat ze op orthogonaliteit en/of op orthogonale transformaties steunen.

Toch steunen ze altijd op gelijkvormigheidstransformaties al dan niet in samenspel met het idee van de machten-methode, namelijk als men maar voldoende dikwijls met de matrix vermenigvuldigt, dan zal uiteindelijk de dominante eigenwaarde gaan overheersen.

12.3.1 Andere manieren om eigenwaarden te berekenen

- De belangrijkste is de QR methode. (Hier is Q orthogonaal en R bovendriehoeks.) Het idee is dat men de matrix schrijft als $A = Q_0 R_0$ door de kolommen van A te orthogonaliseren met Gram-Schmidt en daarna berekent men een nieuwe matrix $A_1 = R_0 Q_0$,

die men dan weer ontbindt als $A_1 = Q_1 R_1$ enz. Merk op dat $A_{k+1} = R_k Q_k = Q_k^T A_k Q_k$. Men voert dus een orthogonale gelijkvormigheidstransformatie uit in elke stap. Men kan aantonen dat normaal de matrix A_k naar een diagonaalmatrix convergeert, die dus noodzakelijk de diagonaal der eigenwaarden moet zijn.

- De deelruimtemethode is bijzonder geschikt als men slechts enkele (bv. n) eigenwaarden wil berekenen voor een heel grote matrix A. We willen helemaal niet op de details ingaan. Een zeer ruwe schets is als volgt te geven. Men stelt dan n vectoren op van de vorm $X_k = A^k X_0$ voor een startvector X_0. Hierin herkent men het idee van de methode van de machten. Men zal de n vectoren X_k orthogonaliseren zodat men eenvoudig het probleem kan projecteren op de ruimte voortgebracht door deze vectoren. Men moet dan slechts een klein eigenwaardeprobleem oplossen (van de orde n). Dan gaat men nieuwe informatie aan de ruimte toevoegen op basis van X_{k+1}, terwijl men de minst relevante informatie weggooit. Men blijft daardoor een (kleine) deelruimte van dimensie n behouden waarin men het probleem telkens moet oplossen. Aldus evolueert die deelruimte naar de deelruimte opgespannen door de eigenvectoren die horen bij de grootste eigenwaarden. Door inverse machten toe te passen kan men zo ook de kleinste eigenwaarden berekenen.

12.3.2 Over deflatie:

Als men bv. met de methode van de inverse machten een eigenwaarde gevonden heeft, dan moet men deze eigenwaarde op een of andere manier uit de matrix verwijderen zodat men de overige eigenwaarden kan gaan zoeken in wat overblijft.

De deflatie van Householder is het best gekend. Stel dat men de eigenvector V gevonden heeft voor de matrix A, dan zal men de matrix A projecteren op het orthogonaal complement van V. Dit gebeurt door het construeren van de vector

$$W = \frac{\text{sign}(V_1)}{\sqrt{2(1+|V_1|)}}(V + [1\ 0\ ...\ 0]^T)$$

Dan maakt men een orthogonale projectiematrix (Householderspiegeling)

$$H = I - 2WW^T$$

waarmee men de gelijkvormigheidstransformatie $A' = HAH$ uitvoert. Door in A' de eerste rij en de eerste kolom weg te laten krijgt men een kleinere matrix waarin de eigenwaarde die hoorde bij V is weggefilterd.

12.4 Chemische bellenbadreactor

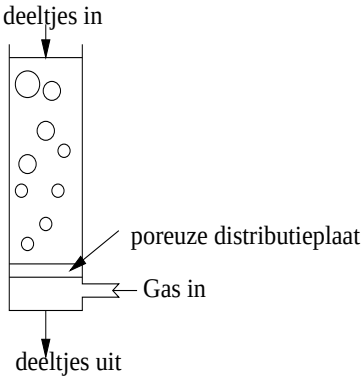
We geven een chemische interpretatie aan de methode van de machten.

12.4.1 Beschrijving van het probleem

We beschrijven een transportprobleem in een chemische reactor. In de reactor wordt onderaan een gas ingepompt dat door een poreuze distributieplaat met hoge snelheid in de reactor wordt gepompt en dat dan in een poederbad in de vorm van bellen opstijgt. Deze bellen

worden steeds groter naarmate ze stijgen omdat ze samensmelten tot een grotere bel. Men kan dit best vergelijken met bellen in kokend water.

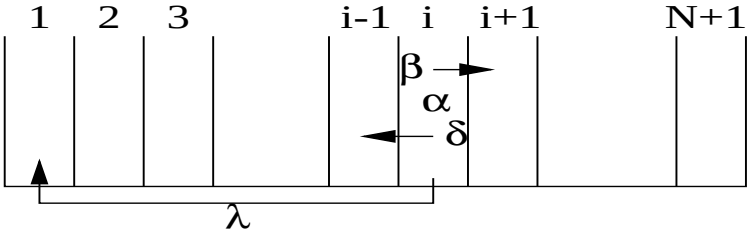
De deeltjes in het poederbad gedragen zich dan ook als een vloeistof. Bij het opstijgen nemen deze bellen in hun zog poederdeeltjes mee naar boven. De deeltjes van het poeder zullen voor de rest zowel naar boven als naar beneden kunnen bewegen, maar dan op een veel tragere manier dan de bellen er doorheen schieten. De deeltjes kunnen dalen, doordat er onderaan poederdeeltjes verdwijnen (naar buiten toe of meegezogen met de gasbellen) of ze kunnen stijgen als gevolg van de turbulentie veroorzaakt door de gasbellen.



Gegeven zekere wetten volgens dewelke de poederdeeltjes bewegen is het de bedoeling een schatting te maken van de tijd die een poederdeeltje in de reactor verblijft. Dit is meteen een beschrijving van het debiet waarmee deeltjes de reactor verlaten. Anderzijds wil men ook de dichtheid van de verdeling van de deeltjes in de reactor kennen. Dit is hetzelfde als het antwoord op de vraag met hoeveel kans een deeltje zich op een bepaalde plaats in de reactor bevindt.

12.4.2 Het Markov-keten model

We zullen de beweging beschrijven essentieel als een (discreet) geboorte-sterfte proces. Dit wil het volgende zeggen. We beginnen met de reactor in N lagen van gelijke dikte te verdelen. Laag nummer 1 is bovenaan en laag nummer N is onderaan. Laag $N+1$ correspondeert met de positie van een deeltje dat langs onder het reactorvat verlaten heeft.
De tijd is eveneens discreet en wordt aangeduid door de index $n = 1, 2, \ldots$.
De kans dat een deeltje op ogenblik n overgaat van laag i naar laag $i-1$ is δ_i.
De kans dat een deeltje op ogenblik n overgaat van laag i naar laag $i+1$ is β_i.
De kans dat een deeltje op een willekeurig ogenblik n van laag i in laag i blijft is α_i.
Normaal zijn dit de enige mogelijkheden en geldt dat $\alpha_i + \beta_i + \delta_i = 1$.
In dit geval is er echter ook een kans dat een deeltje met een gasbel meegezogen werd en met een relatief hoge snelheid (van het ene discrete tijdsmoment op het volgende) terugstijgt naar de oppervlakte (niveau 1). Daar wordt het achtergelaten omdat de gasbellen ontsnappen, maar de deeltjes aan de oppervlakte achterblijven. Noem λ_i de kans dat een deeltje in laag i op een willekeurig ogenblik meegezogen wordt naar niveau 1.



Noem X_n de afstand tot de top (nummer van de laag) waar een zeker deeltje zich bevindt op het ogenblik n. De mogelijke waarden zijn $1, 2, \ldots, N+1$.

Noem $p(n) = (p(n,1), \ldots, p(n, N+1))$ de verdelingsdichtheid van de deeltjes op het ogenblik n. Dit is equivalent met de kansverdeling dat een deeltje zich op ogenblik n in de

laag j bevindt. Dit is gelijk aan $p(n,j)$. Met andere woorden $p(n,j) = P(X_n = j)$.

De dynamica van het systeem wordt dan beschreven door

$$p(n,j) = \sum_{i=1}^{N+1} p(n-1,i)p_{i,j}$$

waarbij $p_{i,j}$ de kans is dat een deeltje van laag i naar laag j gaat. Vermits we het regime stationair veronderstellen is deze kans onafhankelijk van het ogenblik n. Dus

$$p_{i,j} = P(X_{n+1} = j | X_n = i)$$

Dus in het algemeen is voor $i = 2, 3, \ldots, N$

$$p_{i,i-1} = (1-\lambda_i)\delta_i, \quad p_{i,i} = (1-\lambda_i)\alpha_i, \quad p_{i,i+1} = (1-\lambda_i)\beta_i, \quad p_{i,1} = \lambda_i$$

en alle andere $p_{i,j}$ zijn nul.

Eenmaal het deeltje het vat verlaten heeft (onderaan), komt het nooit meer terug. Dus $p_{N+1,N+1} = 1$ en alle andere $p_{N+1,j} = 0$. De onderkant is volledig absorberend.

Een deeltje kan het vat langs boven nooit verlaten. Dus de randvoorwaarde aan de bovenkant kan men schrijven als

$$p_{1,2} = (1-\lambda_1)\beta_1, \quad p_{1,1} = 1 - (1-\lambda_1)\beta_1.$$

De bovenkant is volledig reflecterend.

De dynamica kan men dus samenvatten door een transitiematrix P:

$$p(n) = p(n-1)P = \cdots = p(0)P^n,$$

$$P = \begin{bmatrix} 1-(1-\lambda_1)\beta_1 & (1-\lambda_1)\beta_1 & 0 & & & & 0 \\ \lambda_2 + (1-\lambda_2)\delta_2 & (1-\lambda_2)\alpha_2 & (1-\lambda_2)\beta_2 & 0 & & & 0 \\ \lambda_3 & (1-\lambda_3)\delta_3 & (1-\lambda_3)\alpha_3 & (1-\lambda_3)\beta_3 & 0 & & 0 \\ \lambda_4 & 0 & (1-\lambda_4)\delta_4 & (1-\lambda_4)\alpha_4 & & 0 & 0 \\ \vdots & & & & & & \vdots \\ \lambda_N & 0 & & & & (1-\lambda_N)\alpha_N & (1-\lambda_N)\beta_N \\ 0 & 0 & & & & & 1 \end{bmatrix}$$

Vertrekkend van een begintoestand $p(0)$ kan men zo de deeltjesdichtheid in de verschillende lagen vinden voor een willekeurig ogenblik n.

12.4.3 Verband met eigenwaarden

Merk op dat $p(n) = p(0)P^n$ eigenlijk (op een transpose na) de methode van de machten is. Men vertrekt met een startvector $p(0)$ en men vermenigvuldigt voor elke tijdstap met de matrix P. Met andere woorden, dit is de methode van de machten om een dominante eigenwaarde en eigenvector te berekenen voor de matrix P^T. In dit geval kan men nagaan dat 1 een eigenwaarde is, want $[0\ 0\ \ldots\ 0\ 1]P = [0\ 0\ \ldots\ 0\ 1]$. De matrix P is een stochastische matrix (de som van de rijen is steeds gelijk aan 1). Hiermee kan men aantonen dat 1 ook de grootste eigenwaarde is. Dus $p(n)$ voor $n \to \infty$ zal convergeren naar $[0\ 0\ \ldots\ 0\ 1]$: uiteindelijk zullen alle deeltjes het vat verlaten.

Merk op dat hier in de methode van de machten geen scalering van de vector nodig is omdat de dominante eigenwaarde 1 is en dus de lengte van de vectoren niet verandert.

12.4.4 Gemiddelde verblijftijd

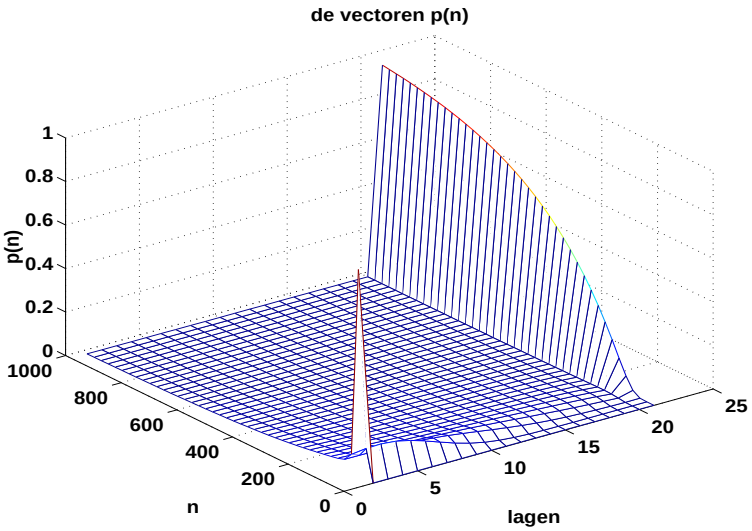
Vermits een deeltje dat het vat verlaat er nooit meer terugkomt is de verblijftijd gelijk aan de tijd nodig om in laag $N+1$ te geraken. Dus $\inf_{n\geq 0}\{X_n = N+1\}$. Men kan de VT verdeling (verdeling van de verblijftijd; d.i. de verdeling van de tijd die een deeltje in de reactor verblijft alvorens die te verlaten) berekenen als $F(n) = p(n, N+1)$. Dit geeft de cumulatieve verdeling van de VT.

Uiteindelijk is men dus in 2 componenten geïnteresseerd:

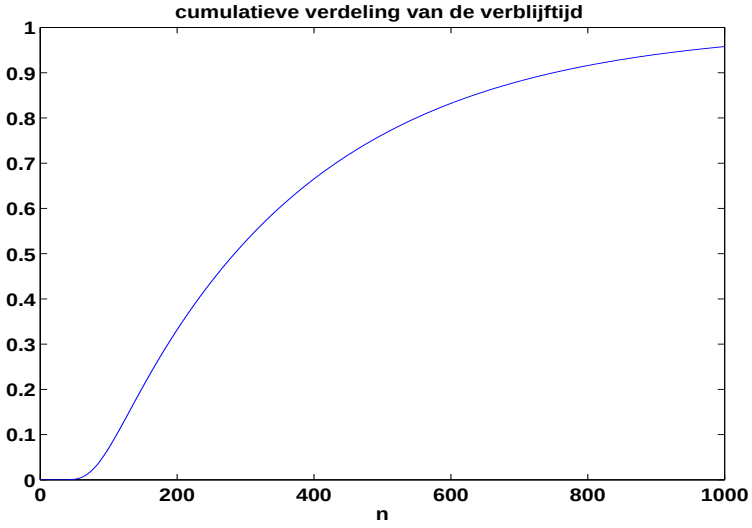
1. de verdeling van de VT, die zegt meteen met welk debiet de deeltjes de reactor verlaten (laag $N+1$)
2. de deeltjesdichtheid in de reactor, dus de verdeling gegeven door $p(n, 1:N)$.

12.4.5 Resultaten

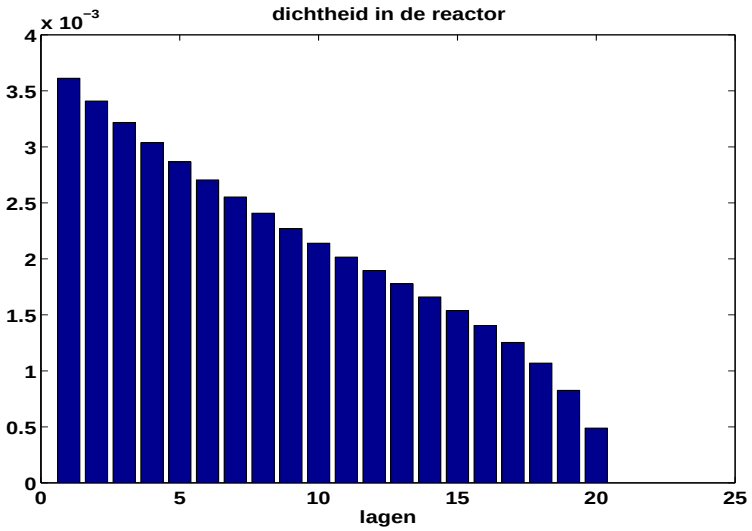
De gegevens zijn: $N = 20$, $\alpha_i = 0.5$, $\beta_i = 0.2$, $\delta_i = 0.3$, $\lambda = 0.01$.
We starten met een vector $p(0) = [1\ 0\ \ldots\ 0]$ en berekenen $p(n) = p(n-1)P$ voor $n = 1:1:its$. Het resultaat is uitgezet:



Bij dit voorbeeld is gestart bij $n = 0$ met een verdeling $p(0) = [1\ 0\ \ldots\ 0]$ en men ziet in deze grafiek het verloop van de methode van de machten: men ziet hoe de vector $p(0)$ geleidelijk evolueert naar de eigenvector $[0\ \ldots\ 0\ 1]$. De curve die je ziet bij $N = 21$ als functie van de tijd n geeft aan hoeveel deeltjes de reactor verlaten hebben in de loop van de tijd. Het is dus de cumulatieve verdelingsfunctie van de VT. We tekenen ze hieronder nog eens afzonderlijk.



Bij $n = 1000$ is reeds $p(n, N+1) = 0.9578$. De overige ongeveer 4% is verdeeld over $p(n, 1:N)$. Dit geeft aan hoe de deeltjes na lange tijd verdeeld zijn in de reactor. Dit wordt weergegeven in de volgende grafiek.



12.4.6 Matlab experimenten

Programmeer de methodes in matlab, of gebruik de voorgeprogrammeerde methodes: `NW/NW/react/matlab/index.html`.

- Uiteindelijk zal de vector $p(n)$ convergeren naar de dominante eigenvector, maar hoe ziet de verdeling binnenin de reactor ($p(n, 1:N)$) eruit voor grote n? Probeer dit voor $\lambda_i = 0.01$ en voor $\lambda_i = 0$. Is er een verschil in verdeling.
- Probeer ook eens met een startvector $p(0) = [1\ 0\ \ldots\ 0]$. Dit betekent dat men alle deeltjes op ogenblik 0 bovenaan in de reactor inbrengt. Bekijk hoe deze evolueert naar $[0\ 0\ \ldots\ 0\ 1]$.
- Probeer eens met andere parameters α_i, β_i en δ_i. Let op hun som moet steeds 1 zijn! Wat gebeurt er als men δ_i groter neemt dan β_i? Wat gebeurt er als men deze parameters laat afhangen van i?

12.4.7 Extra

- Toon aan dat de cumulatieve verdeling $F(n)$ voor de verblijftijd in de reactor zoals die benaderd wordt in het programma op ogenblik n, gegeven wordt door:

$$1 - F(n) = c\mu_1^n + O(\mu_2^n)$$

  waarbij c een constante is en $\mu_0 = 1, \mu_1, \mu_2, \ldots$ de geordende eigenwaarden zijn van P.
- Toon aan dat naast de eigenwaarde 1, alle andere eigenwaarden van P strikt kleiner zijn dan 1. (Gebruik dat de som van de elementen op een rij van P steeds gelijk aan 1 is).
- Bereken de eigenwaarden van P. Is er een groot verschil tussen μ_2 en 1? Verklaart dit de trage convergentie?

12.5 Toepassing: Modale analyse van een gebouw

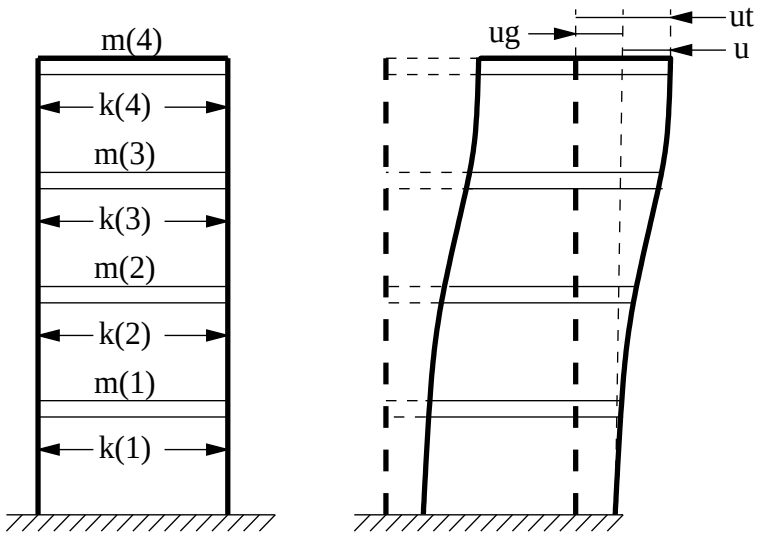
12.5.1 Het probleem

We bespreken hier enkel het eenvoudige model van een "shear building" waarbij enkel een laterale trilling wordt toegestaan. De vloeren worden oneindig stijf verondersteld in het vlak en alle flexibiliteit zit in de kolommen die de vloeren ondersteunen.

Beschouwen we een gebouw met 4 verdiepingen. De massa van de scheiding tussen elk verdiep wordt aangegeven door m_i, $i = 1, \ldots, 4$, te beginnen vanaf het dak.

We willen bestuderen hoe het gebouw vervormt als er een aardbeving plaats heeft.

Stel dat we de uitwijkingen van de vloer i voorstellen door u_i die we in een vector stoppen.



De totale verplaatsing u^t bestaat uit de verplaatsing van het hele gebouw omdat de grond beweegt (voorgesteld door u_g) en de verplaatsing u relatief ten opzichte van deze grondverplaatsing:

$$u^t = u + u_g.$$

De inertiekrachten worden gegeven volgens de wet van Newton: "massa x versnelling":

$$m_i \ddot{u}_i^t = m_i(\ddot{u}_i + \ddot{u}_g).$$

Deze krachten worden in evenwicht gehouden door stijfheidskrachten en dempingskrachten.

De muren boven en onder vloer n zullen de beweging van deze vloer tegengaan. Dit zijn krachten die worden uitgeoefend op de vloer n. Die worden gekarakteriseerd door de *stijfheden* k_i en zijn tegengesteld aan de verplaatsing. Bijvoorbeeld op vloer i werkt een kracht $-2k_{i+1}(u_i - u_{i+1})$ vanwege de twee muren die erbovenop staan, en een kracht $-2k_i(u_i - u_{i-1})$ vanwege de 2 muren die eronder staan. Samen dus

$$-[-2k_i u_{i-1} + 2(k_i + k_{i+1})u_i - 2k_{i+1}u_{i+1}].$$

Voor de bovenste wordt dit aangepast als

$$-[2k_4 u_4 - 2k_4 u_3].$$

De stijfheden worden voor elke ondersteunende muur of pilaar gegeven door $12EI/h^3$ waarbij EI de *buigingsstijfheid* is en h de hoogte van de muur of pilaar. De stijfheden worden uitgedrukt in $[kg/s^2]$.

We bekomen aldus de vergelijking

$$-K\mathbf{u}(t) = M\ddot{\mathbf{u}}(t) + C\dot{\mathbf{u}}(t) + M\mathbf{1}\ddot{u}_g$$

waarin

$$K = \begin{bmatrix} 2(k_1+k_2) & -2k_2 & 0 & 0 \\ -2k_2 & 2(k_2+k_3) & -2k_3 & 0 \\ 0 & -2k_3 & 2(k_3+k_4) & -2k_4 \\ 0 & 0 & -2k_4 & 2k_4 \end{bmatrix},$$

en $M = \text{diag}(m_1, \ldots, m_4)$, en $\mathbf{u} = [u_1 \;\; u_2 \;\; u_3 \;\; u_4]^T$, $\mathbf{1} = [1 \;\; 1 \;\; 1 \;\; 1]^T$.

De matrix K noemt men de (laterale) *stijfheidsmatrix* $[kg/s^2]$.
De matrix M noemt men de *massamatrix* $[kg]$.
De matrix C noemt men de *dempingsmatrix* $[kg/s]$.
De vector $\mathbf{u}$ noemt men de *verplaatsingsvector* $[m]$,
De vector $\mathbf{1}$ noemt men de *invloedsvector*.

De dempingsmatrix C is moeilijk te quantificeren. Een mogelijke proportionele demping is $C = \alpha M + \beta K$, waarbij α en β dan getuned worden volgens dempingen die men observeert bij een paar eigenmodes. Een andere dikwijls gebruikte methode is veronderstellen dat de dempingskrachten evenredig zijn met het verschil in verplaatsingssnelheid tussen nabije vloeren. Men kan dan redeneren zoals bij het opstellen van de stijfheidsmatrix en de dempingskrachten beschrijven met

$$-[-2c_i\dot{u}_{i-1} + 2(c_i + c_{i+1})\dot{u}_i - 2c_{i+1}\dot{u}_{i+1}].$$

De dempingsmatrix heeft dan dezelfde vorm als de stijfheidsmarix.

12.5.2 Vrije ongedempte trilling

Als we veronderstellen dat de grond niet trilt en dat er geen demping is, dan zal de oplossing van de vorm zijn

$$\mathbf{u}(t) = \mathbf{e}q(t), \quad \text{met} \quad q(t) = C_1 \cos\omega t + C_2 \sin\omega t.$$

Invullen in de vergelijking geeft

$$-K\mathbf{u}(t) = M\ddot{\mathbf{u}}(t) = -\omega^2 M\mathbf{u}(t).$$

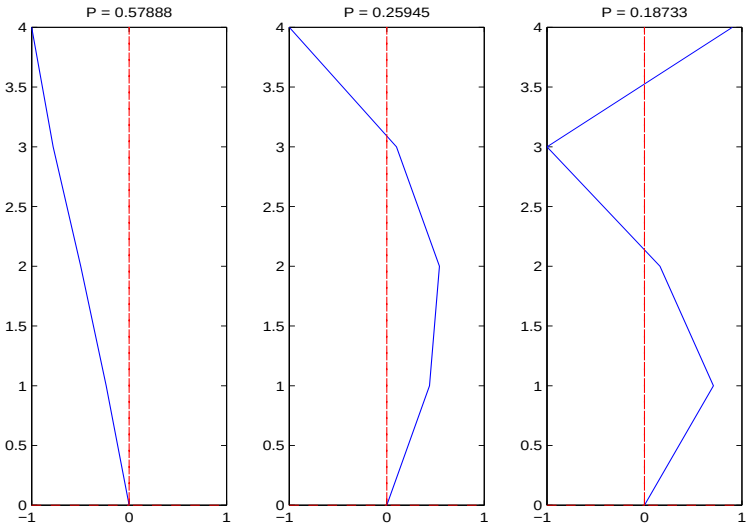
of nog $(K - \omega^2 M)\mathbf{e} = 0$, waaruit blijkt dat $\lambda = \omega^2$ een eigenwaarde en $\mathbf{e}$ de bijhorende eigenvector is voor het veralgemeend eigenwaardeprobleem $K\mathbf{e} = \omega^2 M\mathbf{e}$. Dit is natuurlijk om te zetten in een gewoon eigenwaardeprobleem $(A - \omega^2 I)\mathbf{e} = 0$ met $A = M^{-1}K$.

Stel dat we een aantal eigenvectoren $\mathbf{e}_n$ en de bijhorende eigenwaarden λ_n berekenen, en dat we die in een matrix E stoppen: $E_n = [\mathbf{e}_1 \; \mathbf{e}_2 \ldots \mathbf{e}_n]$. Stel $\Lambda_n = \text{diag}(\lambda_1, \ldots, \lambda_n)$.

De waarden $\omega_j = \sqrt{\lambda_j}$ noemt men de *eigenfrequenties* $[s^{-1}]$
De eigenvectoren $\mathbf{e}_j$ noemt men de *eigenmodes* $[m]$
De getallen $1/\omega_j = 1/\sqrt{\lambda_j}$ noemt men de *eigenperiodes* $[s]$

Merk op dat we niet alle eigenvectoren moeten berekenen. Als we enkel de eigenruimte berekenen die hoort bij de kleinste eigenwaarden, dan hebben we de belangrijkste dynamiek van het systeem wel te pakken. Voor ons eenvoudig model maakt het natuurlijk niet veel uit als we slechts 3 eigenvectoren gebruiken in plaats van 4 of 10. In het algemeen zal men een gebouw of een stuwdam met eindige elementen modeleren en dan heeft men duizenden vrijheidsgraden (variabelen). Als men de berekeningen dan kan herleiden tot berekeningen in een 3-dimensionale ruimte zal men wel enorm veel rekentijd besparen.

Op de onderstaande figuur zijn bijvoorbeeld de eigenvectoren uitgezet die horen bij de drie belangrijkste eigenwaarden (grootste periode). Ze geven de verplaatsingen aan van de 4 vloeren. Ze zijn genormalizeerd zodanig dat de maximale uitwijking 1 is.



P is de periode en is verbonden met de bijhorende eigenwaarde via $P = 1/\sqrt{\lambda}$.

12.5.3 Ontkoppeling van het systeem

Deze eigenvectoren hebben een soort gewogen orthogonaliteitseigenschap. Inderdaad, het eigenwaardeprobleem is

$$(M^{-1}K - \lambda I)E = 0 \quad \text{of} \quad (M^{-1/2}KM^{-1/2} - \lambda I)M^{1/2}E = 0.$$

Dus bevat $\hat{E} = M^{1/2}E$ de eigenvectoren van de symmetrische matrix $M^{-1/2}KM^{-1/2}$. Daarom is het een orthogonale matrix: $\hat{E}^T\hat{E} = D$ met D diagonaal. Dus $\hat{E}^T\hat{E} = E^TME$ is diagonaal.

Anderzijds is $M^{-1}KE = E\Lambda$ of $KE = ME\Lambda$, zodat $E^TKE = \hat{E}^T\hat{E}\Lambda$ ook een diagonaalmatrix is.

We kunnen nu de eigenvectoren gebruiken om het systeem te ontkoppelen. Stel

$$\tilde{K}_n = E_n^TKE_n, \quad \tilde{M}_n = E_n^TME_n, \quad \text{en} \quad \tilde{C}_n = E_n^TCE_n$$

dan zijn de eerste twee diagonaal. De dempingsmatrix resulteert in het algemeen niet in een diagonaalmatrix $\tilde{C}$. In klassieke modale analyse veronderstelt men voor de eenvoud dat de oorspronkelijke dempingsmatrix C zodaing is dat ook de matrix $\tilde{C}$ diagonaal is. In dat geval is het stelsel volledig ontkoppeld.

Het systeem kan dan beschreven worden door (we stellen $\mathbf{u} = E_n\mathbf{q}_n$)

$$-\tilde{K}_n\mathbf{q}_n = \tilde{M}_n\ddot{\mathbf{q}}_n + \tilde{C}_n\dot{\mathbf{q}}_n + \tilde{\mathbf{l}}_n\ddot{u}_g$$

waarbij $\tilde{\mathbf{l}}_n = E_n^T M\mathbf{1}$. Of nog (na vermenigvuldiging met $\tilde{M}^{-1}$)

$$-\Lambda\mathbf{q}_n = \ddot{\mathbf{q}}_n + \hat{C}_n\dot{\mathbf{q}}_n + \mathbf{y}_n\ddot{u}_g$$

met $\hat{C}_n = \tilde{M}_n^{-1}\tilde{C}_n$ en $\mathbf{y}_n = \tilde{M}_n^{-1}\tilde{\mathbf{l}}_n$.

De matrix $\tilde{K}_n$ noemt men de *veralgemeende stijfheidsmatrix*
De matrix $\tilde{M}_n$ noemt men de *veralgemeende massamatrix*
De matrix $\tilde{C}_n$ noemt men de *veralgemeende dempingsmatrix*
De vector $\tilde{l}_n$ noemt men de *veralgemeende belasting*
De vector q_n noemt men de *veralgemeende verplaatsingsvector*

Er geldt $\tilde{K}_n = \tilde{M}_n\Lambda_n$.

12.5.4 Het systeem met belasting

Doordat het systeem volledig ontkoppeld is, kan men de scalaire vergelijkingen afzonderlijk oplossen.

Zo een scalaire vergelijking heeft de vorm

$$-\omega_j^2 q_j = \ddot{q}_j + \hat{c}_j\dot{q}_j + y_j\ddot{u}_g.$$

Stellen we nu

$$u_g(t) = A\sin\omega t \quad \text{en} \quad q_j(t) = Ay_j\omega^2 z_j(t)$$

dan wordt dit

$$-\omega_j^2 z_j = \ddot{z}_j + \hat{c}_j\dot{z}_j + \sin\omega t.$$

Het is de gewoonte om $\hat{c}_j$ te schrijven als $\hat{c}_j = 2\omega_j\zeta_j$. Deze ζ_j geeft dus een modale demping weer en wordt dikwijls uitgedrukt in percenten. Als $\zeta_j = 0.02$ zegt men dat de eigenfrequentie ω_j 2% gedempt is. Merk op dat dit ook een manier is om de demping te beschrijven. Men geeft per ω_i een ζ_i op en na terugtransformatie krijgt men dan een dempingsmatrix voor het oorsprinkelijke gebouw. Maar keren we terug naar de oplossingstechniek. De oplossing voor deze laatste vergelijking bestaat uit een particuliere oplossing gegeven door

$$z_j^p(t) = C_{1j}\sin\omega t + C_{2j}\cos\omega t$$

met

$$C_{1j} = \frac{1}{\omega_j^2}\frac{1-(\omega/\omega_j)^2}{[1-(\omega/\omega_j)^2]^2 + [2\zeta_j(\omega/\omega_j)]^2} \quad \text{en} \quad C_{2j} = \frac{1}{\omega_j^2}\frac{-2\zeta_j\omega/\omega_j}{[1-(\omega/\omega_j)^2]^2 + [2\zeta_j(\omega/\omega_j)]^2}$$

en een homogene oplossing gegeven door

$$z_j^h(t) = e^{-\zeta_j\omega_j t}(C_{3j}\cos\omega_{Dj}t + C_{4j}\sin\omega_{Dj}t)$$

met $\omega_{Dj} = \omega_j\sqrt{1-\zeta_j^2}$ en de coëfficiënten C_{3j} en C_{4j} bepaald door de beginvoorwaarden. Indien die $z_j(0) = \dot{z}_j(0) = 0$ zijn, dan voldoen ze aan

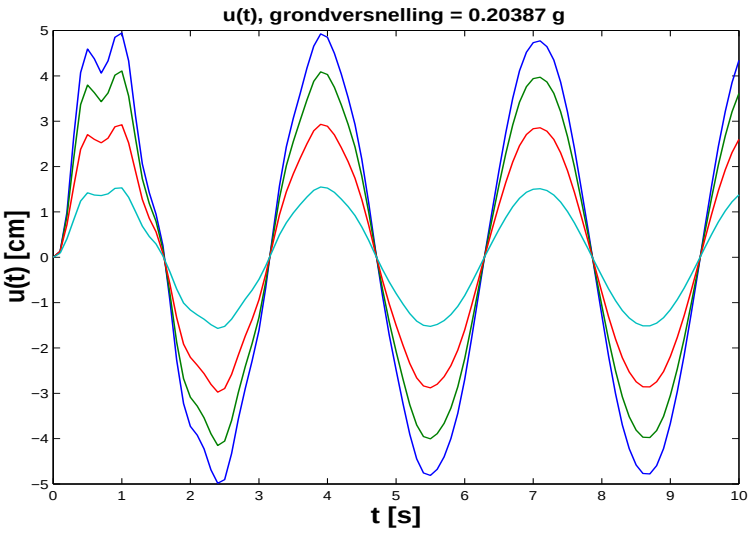
$$C_{3j} = -C_{2j} \quad \text{en} \quad C_{4j} = -(\zeta_j\omega_j C_{2j} + C_{1j}\omega)/\omega_{Dj}.$$

De algemene oplossing is dan $z_j(t) = z_j^p(t) + z_j^h(t)$.

Eenmaal al deze oplossingen berekend, dan kan men de uiteindelijke oplossing reconstrueren als

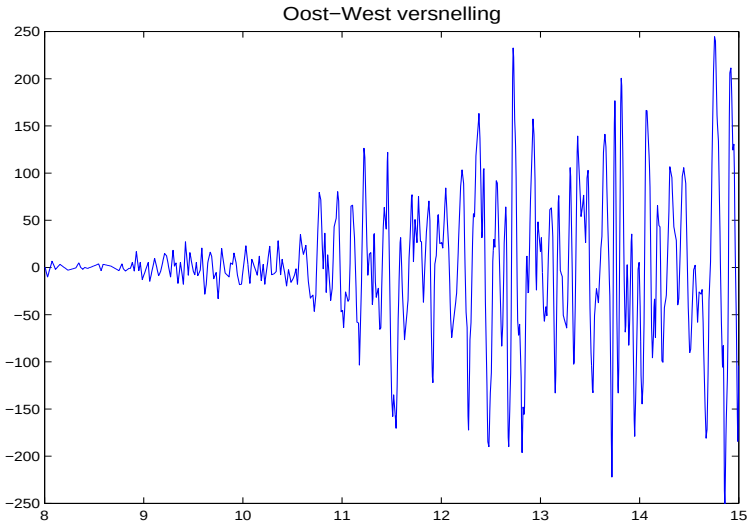
$$\mathbf{u}(t) = E_n \mathbf{q}_n(t) = E_n A\omega^2 [y_1 z_1(t)\ y_2 z_2(t)\ \ldots\ y_n z_n(t)]^T.$$

Het resultaat ziet er uit als volgt: Bemerk de overgangsverschijnselen in het begin. De oplossing is berekend met een amplitude van de grondbeweging gelijk aan 0.5m en een $\omega = 2$. Daardoor heeft men een piekversnelling die gelijk is aan $A\omega^2 = 200cm/s^2 \approx 0.2g$ waarbij $g = 981cm/s^2$ de gravitatieversnelling is.

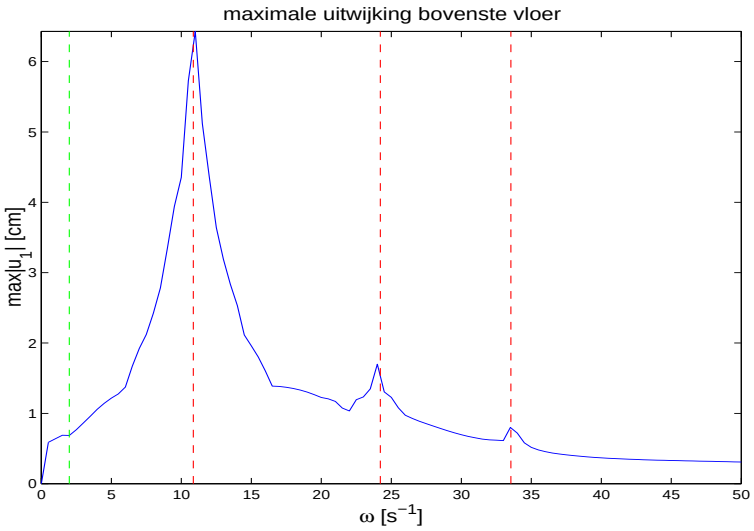


12.5.5 Eigenfrequenties

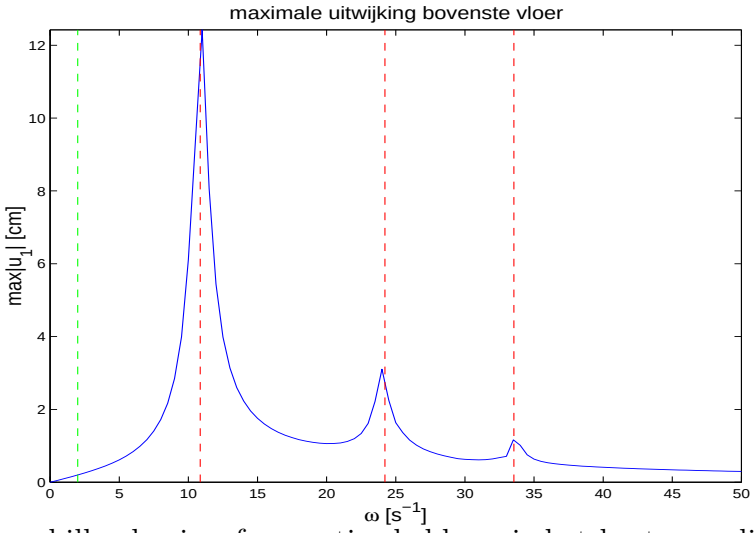
De vorige berekende oplossing $\mathbf{u}(t)$ hangt af van de frequentie ω waarmee de aarde trilt. Praktisch is de trilling van de aarde natuurlijk geen puur harmonische trilling. Men krijgt een grillig verlopende functie. Hiernaast is een voorbeeld van een versnelling opgemeten door een accellerometer gedurende de eerste seconde van een aardschok in Californië in 1989.



Men kan echter een algemene functie door een Fourier-analizer in een Fourierreeks ontwikkelen en de belangrijkste termen overhouden en voor elke term lost men dan op de vorige manier de vergelijkingen op. Als we een eenvoudige harmonische trilling van de aarde veronderstellen en voor de hoogste vloer de maximale uitwijking uitzetten in functie van de aangelegde aardbevingsfrequentie ω, dan krijgt men een grafiek als hiernaast.



Men merkt duidelijk de pieken die optreden als ω gelijk is aan een eigenfrequentie van het gebouw (aangeduid met stippellijn). De meest linkse stippellijn geeft de frequentie aan van de aardbeving waarvoor de vorige oplossing is berekend. Het effect bij eigenfrequenties van het gebouw wordt nog duidelijker als we de overgangsverschijnselen uitsluiten en enkel de homogene oplossing tekenen (zie hiernaast).



Vermits verschillende gebouwen verschillende eigenfrequenties hebben, is het best mogelijk dat na een aardbeving er sommige gebouwen overblijven die bijna onbeschadigd zijn, terwijl een er vlak naast staand gebouw totaal is ingestort.

12.6 Heuristiek

Het is numeriek niet zo eenvoudig om de pieken in de laatste grafiek te berekenen. Dit is trouwens een eerder slecht geconditioneerd probleem. Dikwijls gebruikt men daarom heuristieken om de piek-krachten in een gebouw te berekenen. Bij het ontwerp van een gebouw zal men vooral interesse hebben voor de maximale uitwijkingen en de maximale krachten. Een overschatting is niet erg want dat zal een veiligheidsmarge inbouwen bij de constructie.

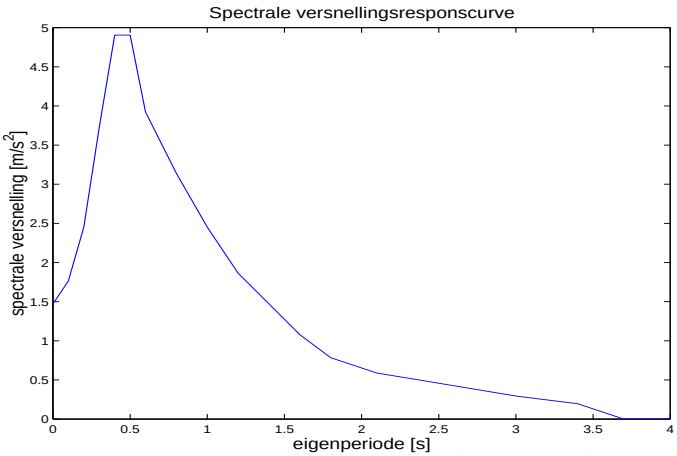
Beschouwen we het dynamische systeem

$$K\mathbf{u}(t) = M\ddot{\mathbf{u}}(t) + C\dot{\mathbf{u}}(t) + M\mathbf{1}\ddot{u}_g.$$

Men kan de uitwijkingen ten gevolge van de dynamica ook beschrijven als gevolg van een equivalente statische kracht. Die equivalente statische kracht is van de vorm $K\mathbf{d}$. Hierin

is K de stijfheidsmatrix en $\mathbf{d}$ de vervorming ten gevolge van de aardschok. Deze kracht komt overeen met een virtuele dynamische kracht $M\mathbf{a}$ waarbij $\mathbf{a}$ een vector van pseudo-versnellingen voorstelt (die heeft inderdaad de dimensie van een versnelling).

Welnu, er bestaan grafieken/tabellen waarbij men in functie van de natuurlijke frequentie of periode de maximale versnelling uitzet. Men noemt dit de *spectrale versnellingsresponsie-curve*. De bijgaande grafiek geeft een voorbeeld. Ze wordt bepaald door 19 meetpunten.



Zo kan men een curve krijgen per waarde van ζ. Men zegt er meestal bij voor hoeveel percent demping dat de curve geldt. Vermits we deze versnellingen enkel kennen in functie van de eigenperiode, zullen we de verplaatsingsvector $\mathbf{d}$ schijven als een combinatie van eigenvectoren: $\mathbf{d} = E_n\tilde{\mathbf{d}}_n$ en analoog voor de versnellingen: $\mathbf{a} = E_n\tilde{\mathbf{a}}_n$. Daardoor kunnen we de vergelijking $K\mathbf{d} = M\mathbf{a}$ omzetten in $\tilde{K}_n\tilde{\mathbf{d}}_n = \tilde{M}_n\tilde{\mathbf{a}}_n$ of $\tilde{\mathbf{d}}_n = \Lambda_n^{-1}\tilde{\mathbf{a}}_n$.

Voor de 3 eigenmodes kunnen we de versnelling uit bovenstaande grafiek aflezen door *lineaire interpolatie*. Dit levert de piek-uitwijkingen voor de verschillende modes. Als we de eigenvectoren genormalizeerd hadden volgens de regel dat de grootste component gelijk is aan 1 in absolute waarde, dan zal de maximale uitwijking voor de verschillende eigenmodes gegeven worden door de kolommen van E_nA_n met A_n de diagonaalmatrix met als diagonaal de elementen van $\Lambda_n^{-1}\tilde{\mathbf{a}}_n$, dus a_j/ω_j^2 met a_j de spectrale versnellingsrespons voor eigenwaarde $\lambda_j = \omega_j^2$.

We gaan hier niet verder in op de details, maar uit deze versnellingen en de eigenmodes van het gebouw kan men dan een schatting maken van de maximale verplaatsingen als gevolg van de eigentrillingen. Die worden dan opgeteld en daaruit kan men dan weer de maximale krachten (spanningen) afleiden die in het gebouw zullen optreden.

12.6.1 Matlab experimenten

Programmeer de methodes in matlab, of gebruik de code op `NW/NW/quake/matlab/index.html` om een aantal experimenten te doen.

- Controleer dat de veralgemeende stijfheid, massa en dempingsmatrices diagonaal zijn.
- Welke routine is gebruikt om de eigenwaarden en eigenvectoren van deze matrix te berekenen?
- Welke eigenwaarden en bijhorende eigenvectoren leveren de grootste bijdrage in de verplaatsingen van de vloeren? Zijn dit altijd de grootste of de kleinste eigenwaarden? Hangt het af van ω?
- Als we alleen de stationaire trilling willen berekenen, en niet de overgangstoestand, welke coëfficiënten moeten dan nul gesteld worden?

- Voor het gegeven voorbeeld, hoeveel bedragen de percentages demping per eigenwaarde? Wijzig de coëfficiënten in C. Wat is de invloed op de uiteindelijke trilling?

12.6.2 Java experimenten

Voor een java applet voor dit probleem zie: `NW/NW/quake/java/index.html`

12.6.3 Extra

- **Over aardbevingen:**

  Opgelet, dit zijn slechts zeer eenvoudige technieken en benaderingen. We hebben het probleem in 2 dimensies beschreven en opgelost. Natuurlijk is de realiteit 3-dimensionaal en kan een gebouw torsiekrachten ondergaan.

  De demonstratie-toolbox van matlab bevat een demo-programma `quake.m`. Door `quake` uit te voeren in matlab zie je hoe voor de gegevens van een aardbeving in 1989 in Loma Prieta (Californië) de aarde bewoog in 3 dimensies. Wat men opmeet zijn de versnellingen van de aarde. De baan vindt men door twee keer te integreren.

  Via een eenvoudige zoekrobot kan men duizenden web-sites vinden op het web met gegevens over aardbevingen.

- **Andere manieren om een dynamisch systeem te berekenen:**

  Men zou ook een programma kunnen gebruiken uit de matlab controle toolbox. Het programma `lsim` laat toe om een eerste orde stelsel van differentiaalvergelijkingen met een willekeurige invoer op te lossen. Mits de orde te verdubbelen kan men het tweede ordesysteem omzetten in een systeem van eerste orde. De vector van onbekenden bestaat dan uit de verplaatsingsvector $\mathbf{u}$, gevolgd door de vector der snelheden $\dot{\mathbf{u}}$: $\mathbf{y}^T = [\mathbf{u}^T\ \dot{\mathbf{u}}^T]$. Met de matrices K, M, C kan men dan een dynamisch systeem met invoer $\mathbf{w}$ van de vorm

  $$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{w}$$
  $$\mathbf{y} = C\mathbf{x} + D\mathbf{w}$$

  maken en laten oplossen.

- Wie meer over dit probleem wil lezen kan het volgende book nakijken: Anil K. Chopra, *Dynamics of structures, theory and applications to earthquake engineering*, Prentice Hall, 1995.

12.7 Schrödinger vergelijking

12.7.1 Het probleem

De Schrödingervergelijking is een partiële differentiaalvergelijking die fundamenteel is in de quantummechanica. Ze heeft de vorm

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi(x,t)}{\partial x^2} + V(x,t)\psi(x,t) = i\hbar\frac{\partial\psi(x,t)}{\partial t}.$$

Hierin is

- $\psi(x,t)$ de golffunctie,
- x is de positie,
- t is de tijd,
- m is de massa van het deeltje
- $V(x,t)$ is de opgelegde potentiaal,
- $\hbar = 6.6262 \times 10^{-27}$ erg s $= 6.6262 \times 10^{-34}$ J s, is de constante van Planck.

In het geval de potentiaal niet afhangt van de tijd wordt de oplossing $\psi(x,t) = \psi(x)T(t)$ waarbij $T(t)$ oplossing is van

$$i\hbar \frac{dT(t)}{dt} = ET(t)$$

met als triviale oplossing $T(t) = Ce^{-iEt/\hbar}$, (C een constante) en $\psi(x)$ oplossing is van de tijdsonafhankelijk Schrödinger vergelijking

$$H\psi(x) = E\psi(x), \quad H = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x).$$

Men noemt H de Hamiltoniaan en E de scheidingsconstante.

12.8 Oplossing van de tijdsonafhankelijke Schrödinger vergelijking

Merk op dat we hier een eigenwaardeprobleem hebben: E is een eigenwaarde voor de Hamiltoniaan operator en $\psi(x)$ is de bijhorende eigenfunctie.

Men kan dit door discretisatie eenvoudig terugbrengen tot een eigenwaardeprobleem uit de lineaire algebra. Daartoe discretiseren we de functie $\psi(x)$ en stellen $x_k = x_0 + k\Delta x$ en noteren $\psi(x_k) = \psi_k$. Vermits

$$\left.\frac{d^2\psi(x)}{dx^2}\right|_{x=x_k} \approx \frac{\psi_{k-1} - 2\psi_k + \psi_{k+1}}{(\Delta x)^2}$$

kunnen we stellen dat

$$H_{k,k-1}\psi_{k-1} + H_{k,k}\psi_k + H_{k,k+1}\psi_{k+1} = E\psi_k$$

met

$$H_{k,k-1} = H_{k,k+1} = -\frac{\hbar^2}{2m(\Delta x)^2}, \quad H_{k,k} = \frac{\hbar^2}{m(\Delta x)^2} + V(x_k).$$

We krijgen dus een eigenwaardeprobleem voor een tridiagonale matrix H.

In principe moeten we dit oplossen voor $x \in (-\infty, +\infty)$ maar om fysische redenen moet de oplossing $\psi(x)$ naar nul gaan voor $x \to \pm\infty$. We kunnen dus bij benadering $\psi(x)$ nul stellen voor waarden van x die ver weg liggen van het gebied dat ons interesseert. De oplossing die hoort bij de kleinste eigenwaarde noemt men de grondtoestand.

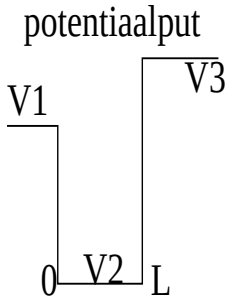
12.8.1 Gebonden toestanden

Soms is de potentiaalfunctie zodanig dat de oplossing beperkt blijft tot een bepaald gebied. Men spreekt dan van gebonden toestanden. Bijvoorbeeld als men een potentiaalput opgegeven krijgt.

Dit wil zeggen dat bijvoorbeeld

$$V(x) = \begin{cases} V1, & x \leq x_1 \\ V2, & x_1 < x \leq x_2 \\ V3, & x_2 < x \end{cases}$$

met bijvoorbeeld $V2 < V1 < V3$.



De eigenfuncties met eigenwaarde $V2 < E < V1$ zullen slechts essentieel verschillen van nul in de put tussen 0 en L. De eigenfuncties met eigenwaarde $V1 < E < V3$ zullen slechts verschillen van nul voor waarden $x < L$. Men kan zich ingewikkelder potentiaalfuncties indenken, maar de veralgemening is eenvoudig te maken.

12.8.2 Tijdsafhankelijke oplossing

Om de oplossing te krijgen afhankelijk van x en van t moeten we de oplossing berekenen $\psi(x)T(t)$. Deze hangt af van opgegeven beginvoorwaarden, namelijk de functie $\psi(x,0) = \psi(x)T(0) = \psi(x)$ moet gegeven zijn; bv. als $B(x)$. Stel dat we de oplossing kunnen schrijven als een lineaire combinatie van de eigenfuncties:

$$\psi(x,t) = \sum_{k=1}^{n} a_k e_k(x) T_k(t)$$

waarbij de $e_k(x)$ de eigenfuncties zijn met eigenwaarden E_k. De functies $T_k(t)$ vinden we door deze vorm in te vullen in de vergelijking

$$i\hbar \frac{\partial \psi(x,t)}{\partial t} = H\psi(x,t).$$

Zo vinden we dat $T_k(t) = e^{-iE_k t/\hbar}$. De coëfficiënten a_k bepalen we door te eisen dat in alle punten x_j $B(x_j) = \sum_{k=1}^{n} a_k e_k(x_j)$. Vermits er meestal veel meer punten x_j zijn dan termen in de som, is dit een lineair kleinstekwadratenprobleem. Dus als V de matrix is met eigenfuncties: $V_{jk} = e_k(x_j)$, dan lossen we dit in matlab op door te stellen: $a = V \backslash B$

Dus in matrixnotatie

$$\psi(x,t) = VTa$$

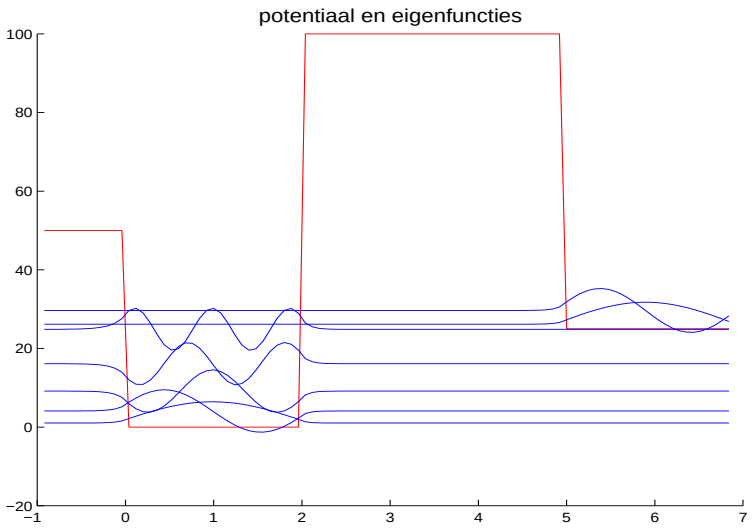
met

- $\psi(t)$ de vector van functiewaarden op het ogenblik t
- V de matrix met $V_{j,k} = e_k(x_j)$
- T de diagonaalmatrix $T = \mathrm{diag}(T_1(t), \ldots, T_n(t))$
- a de vector der coëfficiënten a_k

Om het reëel of imaginair deel van de oplossing te krijgen neemt men $T_k(t) = \cos(t/\hbar)$ resp. $T_k(t) = \sin(t/\hbar)$; de energie bekomt men door de absolute waarde te nemen.

12.9 Voorbeeld

Op de volgende figuur is de potentiaalfunctie ($V1 = 50$ voor $x < 0$, $V2 = 0$ voor $0 < x < 2$, $V3 = 100$ voor $2 < x < 5$, $V4 = 30$ voor $x > 5$) getekend en de eerste 7 eigenfuncties.
De eerste 5 hebben een eigenwaarde kleiner dan V4, de laatste 2 eigenwaarden zijn groter dan V4, maar kleiner dan V1.



12.9.1 Matlab experimenten

Programmeer de methodes of gebruik de code op `NW/NW/schroedinger/matlab/index.html` om een aantal experimenten te doen.

- Hoe en waar werd in de code de tridiagonale matrix die de Hamiltoniaan voorstelt opgesteld?
- Welke routine is gebruikt om de eigenwaarden en eigenvectoren van deze matrix te berekenen?
- Probeer verschillende potentialen om het effect op de eigenfuncties te zien.
- Bekijk ook voor een gegeven begintoestand de evolutie in de loop van de tijd.
- Hoe gevoelig is de berekende oplossing aan de lengte van het gekozen interval van de x-waarden? (De oplossing moet nul zijn op oneindig.)
- Hoe klein of groot moet de stap Δt zijn om een vloeiend verloop van de golffunctie te zien te krijgen?

- Door de parameter `ncoef` te wijzigen bepaalt men het aantal termen in de ontwikkeling van de oplossing in eigenfuncties. Welke invloed heeft de grootte van deze parameter op de berekende tijdsafhankelijke oplossing?

Opgelet, dit zijn slechts zeer eenvoudige technieken en benaderingen. Wil men algemenere gevallen oplossen of moeilijker problemen toch nog nauwkeurig berekenen, dan moet men zijn toevlucht nemen tot meer geavanceerde methoden die eventueel ook vluggere berekeningen toelaten.

Voor applets zie
http://www.brainflux.org/java/classes/Schrodinger1D.html.
of
http://www.neti.no/java/sgi_java/WaveSim.html.
Een matlab toolbox is te vinden op

http://www.theophys.kth.se/mathphys/schrodinger.html

Deze gebruikt numerieke technieken die we niet gezien hebben.

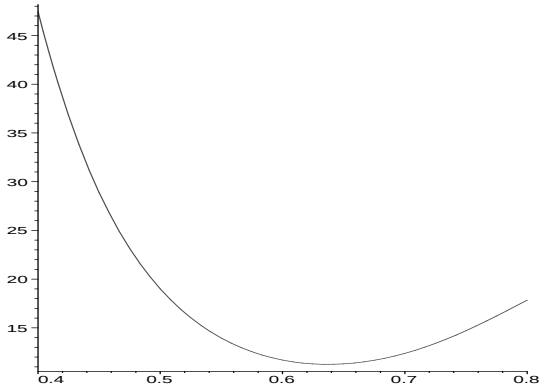
Hoofdstuk 13

Optimalisatie

13.1 Iteratieve methoden voor het berekenen van een minimum

We zoeken het minimum van de `humps` functie:

$$f(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6.$$



De eerste en tweede afgeleide zijn:

$$f'(x) = -\frac{2x-0.6}{(x-0.3)^2+0.01)^2} - \frac{2x-1.8}{(x-0.9)^2+0.04)^2}$$

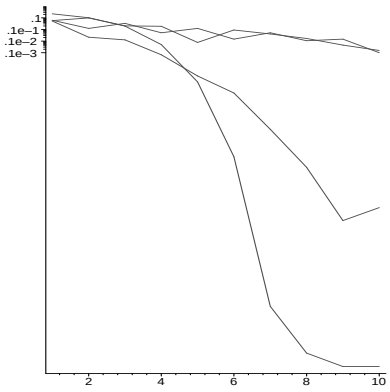
$$f''(x) = \frac{2(2x-0.6)^2}{(x-0.3)^2+0.01)^2} - \frac{2}{(x-0.3)^2+0.01)^2} + \frac{2(2x-1.8)^2}{(x-0.9)^2+0.04)^2} - \frac{2}{(x-0.9)^2+0.04)^2}.$$

Het minimum wordt bereikt voor
x = 0.637008984714047809980745955320934413645905401883345024326350240799520899649

13.1.1 Vergelijking van de verschillende methoden

Hiernaast zie je de relatieve fout in functie van de iteratiestap voor
Bissectie
Gulden snede
Kwadratische interpolatie
Newton

Voer het programma uit zodat je ziet welke curve bij welke methode hoort.



Merk op dat de bissectie en gulden snede methoden lineair zijn (rechte lijn op log schaal) en dat de andere methoden superlineair zijn (kromme op log schaal).

We weten dat voor een enkelvoudig minimum zoals hier

ordebissectie = 1
ordeGulden snede = 1
ordekwadratische interpolatie = 1.32...
ordeNewton = 2

13.1.2 Experimenten en vragen

Het maple werkblad kan je hier vinden: `NW/NW/min1/maple/index.html`

- Met het werkblad kan je proberen de orde van de methode van kwadratische interpolatie te bepalen. Men kan daarom bij het oproepen van `ofactor` het getal waarvan je denkt dat het de orde is invullen. Als je de juist waarde invult, dan zou het resultaat een grafiek moeten zijn die convergeert naar een constante. Waaraan is die gelijk volgens uw experimenten.
- Waarom is de Newton methode ook voor het zoeken van een minimum "normaal gesproken" kwadratisch convergent? Wanneer is dat niet meer waar?
- Probeer ook eens een andere dan de opgegeven functie. Zijn de resultaten hiervoor hetzelfde? Is er altijd convergentie naar het minimum? Zou er ook convergentie naar een maximum kunnen zijn? Voor welke methoden?
- Vanaf wanneer krijgen we problemen met afrondingsfouten? Je kan met de nauwkeurigheid spelen door hp en lp te variëren.
- Als je met 30 cijfers werkt, tot hoe ver kan je dan het minimum berekenen? Is dit zo voor alle methoden of zijn sommige methoden meer onderhevig aan afrondingsfouten dan anderen? Heeft dit dan iets te maken met de manier van implementeren?
- Welk stopcriterium zou je voor de verschillende methoden gebruiken?
- Hoe heeft men nagegaan dat de orde van convergentie voor de kwadratische interpolatie ongeveer 1.32 is? (Men kan bewijzen dat de exacte orde een wortel is van de vergelijking $x^3 = x + 1$.)
- Probeer ook eens `ofactor` voor de kwadratische interpolatie op te roepen met een "verkeerde orde", bv. 3.12 of 1.6 of 1.2. Hoe gedraagt zich dan de bekomen grafiek?
- Wat gebeurt er als je de orde te groot schat en wat gebeurt er als je ze te klein schat?

13.1.3 Extra

- Bij de gulden snede methode kan men expliciet de u en v berekenen met de gulden snede factor, of zoals hier is gedaan door het symmetrisch punt te nemen. Zijn beide methoden even goed wat betreft afrondingsfouten?
- Zoek alle wortels van de vergelijking $x^3 = x + 1$. Welke van de wortels geeft de orde van convergentie van de kwadratische interpolatiemethode?

Er zijn ook illustratieve demo's in matlab te vinden voor de gulden snede en het Dekker-Brent algoritme: `NW/NW/min1/matlab/index.html`.

13.2 Het Dekker-Brent algoritme

We zoeken zoals in de vorige paragraaf het minimum van de `humps` functie (ingebouwd in matlab) en van de functie $f_2(x) = 1 - x * \exp(-x^2)$.

De twee methoden die we gebruiken zijn gulden snede en het Dekker-Brent algoritme.

De gulden snede

De gulden snede neemt 2 punten in het interval (minimum 2 nodig om voor een unimodale functie het interval te kunnen herleiden).

De gulden snede vraagt enkel om het eerste punt vast te leggen, de andere punten gedurende het hele proces kan men als een symmetrisch punt bepalen. Het symmetrische punt van het overgebleven punt ten opzichte van het midden van het overblijvende interval.

De methode is lineair en de convergentiefactor is ongeveer 0.6 (de gulden snede). Dit is zeer traag. Trager dan de bissectiemethode, maar ze is maar half zo duur per iteratiestap.

Dekker-Brent algoritme

Dit algoritme combineert de gulden snede en de kwadratische interpolatie.

Kwadratische interpolatie convergeert superlineair maar is niet zo robuust als de gulden snede.

De vuistregel is: gebruik steeds kwadratische interpolatie, behalve als dit geen punt oplevert dat in het onzekerheidsinterval ligt of als het te dicht tegen een punt ligt dat al geëvalueerd is. In dat geval gebruik gulden snede voor het onzekerheidsinterval.

De routine geeft aan welke methode in elke stap is gebruikt.

De gulden snede is normaal slechts in het begin nodig. Uiteindelijk wordt enkel parabolische interpolatie gedaan en convergeert de methode dus superlineair.

13.2.1 Experimenten en vragen

De matlab code kan je hier vinden.: `NW/NW/min1/matlab/index.html`.

- Ga na hoe het algoritme van Dekker-Brent zoals het hier geprogrammeerd is, test of de parabolische interpolatie een goed punt oplevert.
- Hoe dikwijls wordt parabolische interpolatie vervangen door de gulden snede? Gebeurt dit in de eerste of in de laatste stappen?
- Probeer ook eens een andere dan de opgegeven functie. Zijn de resultaten hiervoor hetzelfde? Is er altijd convergentie naar het minimum? Zou er ook convergentie naar een maximum kunnen zijn?
- Hoe belangrijk zijn de startwaarden? Wat gebeurt er als je een te groot interval opgeeft zodat de functie er niet meer unimodaal is?
- Vanaf wanneer krijgen we problemen met afrondingsfouten?
- Welk stopcriterium zou je voor de verschillende methoden gebruiken?

13.2.2 Extra

- De methode waarbij men een minimum zoekt van een functie door een veelterminterpolatie te berekenen van de derde graad, en dan een nulpunt van de afgeleide neemt als nieuw iteratiepunt heeft een orde 1.46... Als je weet dat die orde een wortel is van de vergelijking $x^4 = x^2 + x + 1$ bereken dan de volgende cijfers in het getal 1.46...
  Men kan aantonen dat de orde van de secant methode om nulpunten te zoeken een wortel is van de vergelijking $x^2 = x+1$. Bereken de orde dan met 10 cijfers nauwkeurig.
- Merk op dat de orde van de methode van Muller voor nulpunten gelijk is aan 1.83... Dit getal is een wortel van $x^3 = x^2 + x + 1$. Bereken ook hier een aantal cijfers meer.

13.3 De Rosenbrock functie

13.3.1 De functie

We zoeken het minimum van de functie: $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$. Ze heeft een enig minimum gelijk aan nul bij (1,1), zoals men direct kan verifiëren. Als startpunt kiezen we telkens $(-1.9, 2)$.
De gradiënt van $f(x, y)$ heeft als componenten grad $= [-400x(y-x^2)-2(1-x), 200(y-x^2)]$.
De Hessiaan-matrix is

$$\begin{bmatrix} 2 - 400(y - 3x^2) & -400x \\ -400x & 200 \end{bmatrix}$$

Deze functie is een som van kwadraten, en kan dus ook dienen voor het testen van kleinste-kwadraten-methoden.

We beschouwen dan een stelsel van de vergelijkingen

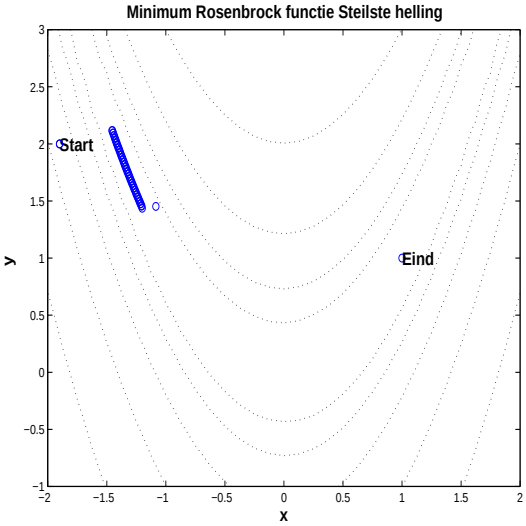
$$\begin{aligned} u(x, y) &= 10(y - x^2) \\ v(x, y) &= 1 - x \end{aligned}$$

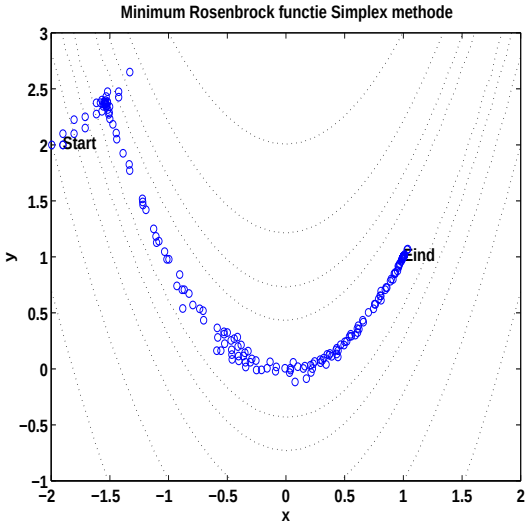
en we moeten zowel $u(x, y)$ als $v(x, y)$ nul krijgen.
De Jacobiaan matrix van dit stelsel heeft als elementen

$$\begin{bmatrix} -20x & 10 \\ -1 & 0 \end{bmatrix}$$

Op dit probleem hebben we methoden losgelaten voor algemene optimalisatieproblemen.

Hierna ziet men de resultaten voor de methode van de steilste helling en de simplexmethode.

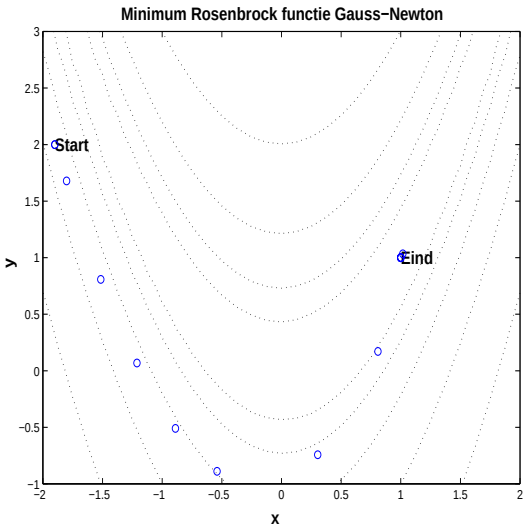


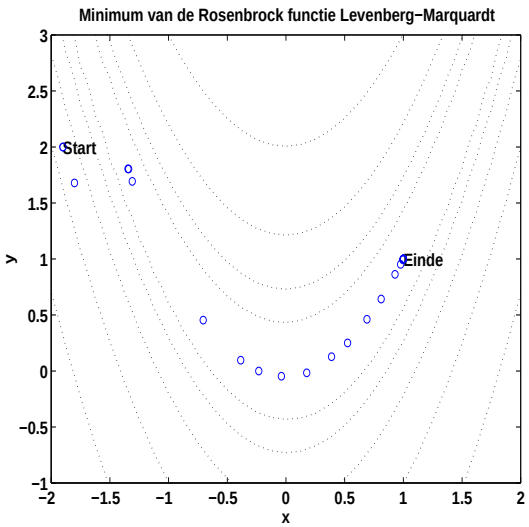


Steilste helling
Aantal iteraties = 68
Aantal functieevaluaties = 302
Geen convergentie.

Simplex methode
Aantal iteraties = 109
Aantal functieevaluaties = 201
Convergentie.

Men kan dit natuurlijk ook als een niet-lineair kleinste-kwadraten-probleem aanpakken. Hieronder ziet men de resultaten voor de methode van Gauss-Newton en de Levenberg-Marquardt methode. (De Levenberg-Marquardt routine van matlab is speciaal voor kleinste-kwadraten-problemen.)





Gauss-Newton
Aantal iteraties = 13
Aantal functieevaluaties = 55
Convergentie.

Levenberg-Marquardt
Aantal iteraties = 21
Aantal functieevaluaties = 92
Convergentie.

Deze resultaten werden bekomen met de routines uit de optimization toolbox van matlab. De 1D lijnzoekroutine (linesearch) die in elke stap gebruikt wordt gebruikt een vorm van veelterminterpolatie.

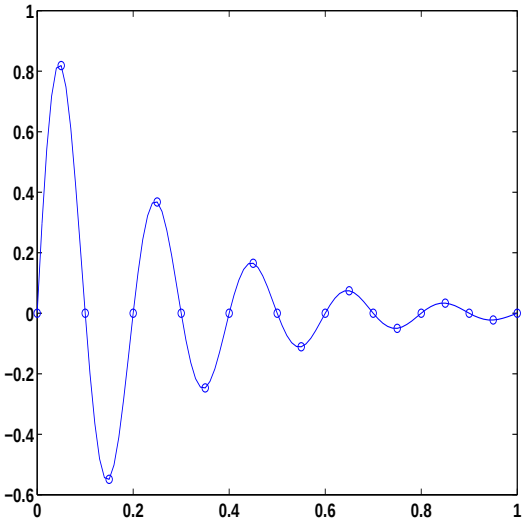
Deze optimalisatie toolbox is niet vrij beschikbaar zodat men hiermee niet kan experimenteren, tenzij men een licentie heeft.

13.3.2 De simplexmethode

Dit noemt men ook wel de methode van Nelder en Mead.
Deze methode kan men illustreren aan de hand van een 2D-probleem omdat men dan duidelijk de opeenvolgende reflecties, contracties, enz. van de simplices (driehoeken) kan voorstellen.

13.3.3 Data fitting

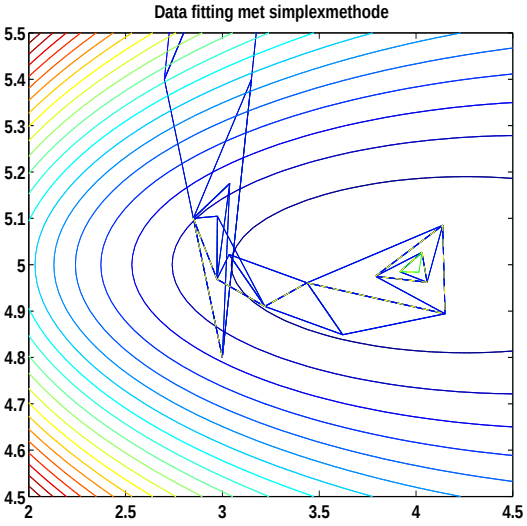
We beschouwen het volgende probleem. Gegeven zijn een aantal meetpunten voor de functie $\text{decay}(a, b, t) = e^{-at}\sin(2\pi bt)$. De meetpunten zijn voor $t = 0 : .05 : 1$ (21 equidistante punten tussen 0 en 1).
Gevraagd zijn de parameters a en b.
De functie en de meetpunten zijn op nevenstaande figuur weergegeven. In dit voorbeeld zijn de optimale waarden $a = 4$ en $b = 5$.



Noem de meetpunten $(t_k, y_k), k = 1, \ldots, N$ met $N = 21$. De functie die we moeten minimaliseren is :

$$S(a, b) = \sum_{k=1}^{N} |y_k - \text{decay}(a, b, t_k)|^2$$

De volgende figuur laat zien hoe de simplexmethode hiervan het minimum vindt. De opeenvolgende simplices (driehoeken) van de verschillende stappen zijn getekend.



13.3.4 Experimenten en vragen

De matlab code kan je hier vinden: `NW/NW/minn/matlab/index.html`.

- Voor de Rosenbrock functie kan je de gradiënt en de Hessiaan berekenen. Doe dat door rechtstreeks af te leiden. Ga na dat als men de formule voor de Hessiaan zoals die voor kleinste-kwadraten-problemen is afgeleid in de cursus tot hetzelfde resultaat aanleiding geeft.
- Welk stuk wordt er verwaarloosd in de methode van Gauss-Newton? Is dit een belangrijk stuk? Zal dit de convergentiesnelheid van de Gauss-Newton methode beïnvloeden?
- Kan je verklaren waarom de methode van de steilste afdaling niet tot een oplossing komt en vroegtijdig stopt?
- Welke van de methoden proberen eerst naar de bodem van de vallei af te dalen om daarna de bodem van die vallei te volgen, en welke methoden doen dit niet?
- Hoeveel werk (functies, afgeleiden,...) zijn er nodig per iteratiestap voor de verschillende methoden? Welke vraagt het minste werk per stap?
- Welke convergeert het vlugste?
- Is er bij het data-fitting probleem een verschil tussen de oorspronkelijke functie en de functie die gevonden werd? Hoe gevoelig is de functie voor wijzigingen op de parameters a en b? Is het probleem dan goed of slecht geconditioneerd?
- Wat weet je over de conditie van het minimum voor de Rosenbrock functie?
- Welke soort bewerkingen worden op de simplices uitgevoerd tijdens de opeenvolgende stappen van het data-fitting probleem? Zal deze methode altijd convergeren?
- Hoe ziet de gradiënt en de Jacobiaan er uit voor het data-fitting probleem?
- Bereken (met matlab) de eigenwaarden (en eigenvectoren) van de Hessiaan van de Rosenbrock functie in het minimum (1,1). Wat kan je daaruit besluiten in verband met de conditie van dat minimum?
- Hoe wordt in de eenvoudige code rosen.m de staplengte bepaald in de methode van de steilste afdaling? Heb je een beter voorstel?

13.3.5 Extra

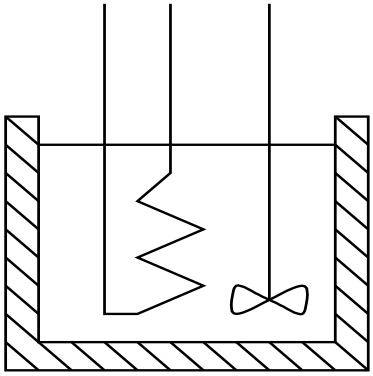
- Probeer het data-fitting probleem ook eens op te lossen als men op de gegevens ook nog een kleine willekeurige fout zet van bv p percent.
- Probeer het data-fitting probleem ook eens voor een andere functie.
- In het data-fitting probleem is daar het equidistant zijn van de meetpunten belangrijk voor de conditie van het minimalisatieprobleem? Zullen bijvoorbeeld twee meetpunten die dicht bij elkaar liggen een slechte conditie veroorzaken, of zal dit juist de conditie verbeteren?

13.4 Opwarming van een vloeistof

13.4.1 Beschrijving van het probleem

Een verwarmingselement wordt in een vat met vloeistof gestoken. Als het element “aan” staat, levert het een vermogen dat de vloeistof opwarmt. Met behulp van een roersysteem

wordt verondersteld dat de temperatuur van de vloeistof uniform is. De temperatuur van de omgeving wordt constant verondersteld.



We willen algemeen het verloop van de temperatuur van de vloeistof kennen in functie van de tijd, en met als parameter het moment waarop het verwarmingselement wordt in- of uitgeschakeld. Wat is de maximale temperatuur en op welk ogenblik na het uitschakelen van het verwarmingselement wordt die bereikt? Kun je de uitschakeltijd berekenen zodat een gegeven maximale temperatuur niet wordt overschreden?

Het verwarmingselement

Het model voor het verwarmingselement is als volgt: Zolang dit element aan staat levert dit een vermogen dat naar het maximum Q streeft

$$q(t) = Q(1 - e^{-\tau(t-t_0)})$$

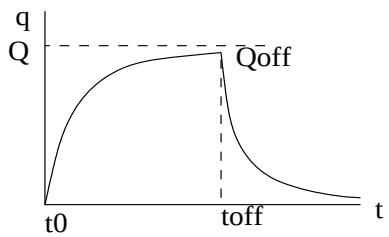
Van zodra het uitgeschakeld wordt zal dit exponentieel afnemen

$$q(t) = Q_{\text{off}} e^{-\tau(t-t_{\text{off}})}.$$

Hierin is

- τ een dimensieloze constante
- Q het maximaal vermogen [W]
- t de tijd [s]
- t_{off} het tijdstip waarop het element wordt uitgeschakeld [s]
- $Q_{\text{off}} = q(t_{\text{off}})$, de waarde van q op het ogenblik t_{off}
- t_0, tijdstip waarop het element wordt ingeschakeld

Een typisch verloop is als volgt



Temperatuurverloop

De temperatuur in het vat loopt op volgens de toegevoegde warmte door het element. Tegelijk is er warmteverlies aan de oppervlakte. De vergelijking is

$$\frac{\partial T(t)}{\partial t} = -\frac{kA}{Mc}(T(t) - T_u) + \frac{q(t)}{Mc}.$$

De eerste term geeft een verlies aan warmte, de tweede de toename van de warmte. De betekenis van de parameters is als volgt:

- k: warmteoverdrachtscoëfficiënt $\left[\frac{\mathrm{W}}{\mathrm{m^2K}}\right]$
- A: de vrije oppervlakte [m^2]
- M: de massa [kg]
- c: de soortelijke warmte van de vloeistof $\left[\frac{\mathrm{J}}{\mathrm{kg\ K}}\right]$
- T_0: temperatuur vloeistof op ogenblik t_0, [K]
- T_u: omgevingstemperatuur, [K]

Het probleem

De volgende vraagstukken moeten worden opgelost

- We willen het verloop kennen van de temperatuur in functie van de tijd.
- In het bijzonder willen we de maximale temperatuur kennen.
- We willen het verloop kennen van de maximale temperatuur in functie van de verwarmingstijd (dus in functie van t_{off}).
- Wat is de verwarmingstijd als we een bepaalde maximum temperatuur T_m willen bereiken zonder die te overschrijden.

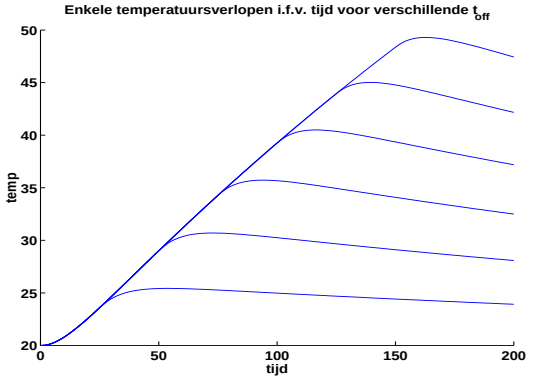
13.5 Numerieke gegevens

We lossen het probleem op voor de volgende numerieke waarden
$A = 0.5$, $M = 1$, $c = 4182$, $T_0 = 20$, $T_u = 20$, $k = 20$, $t_0 = 0$, $\tau = 0.1$, $Q = 1000$.

13.5.1 Oplossing

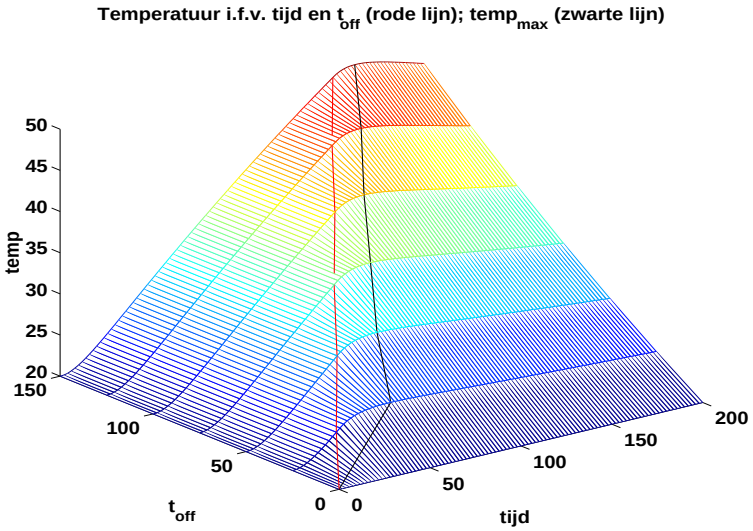
Temperatuur als functie van de tijd

Hiervoor moeten we een differentiaalvergelijking oplossen. We kunnen daarbij een kleine constante stap nemen en een eenvoudige methode gebruiken. Er is geen speciale moeilijkheid met de vergelijking zodat dit wel voldoende nauwkeurig is. We lossen de vergelijking op voor $t = t_0 : h : t_e$. We hebben daarbij een waarde voor t_{off} nodig. We kiezen er een aantal, bijvoorbeeld equidistant.



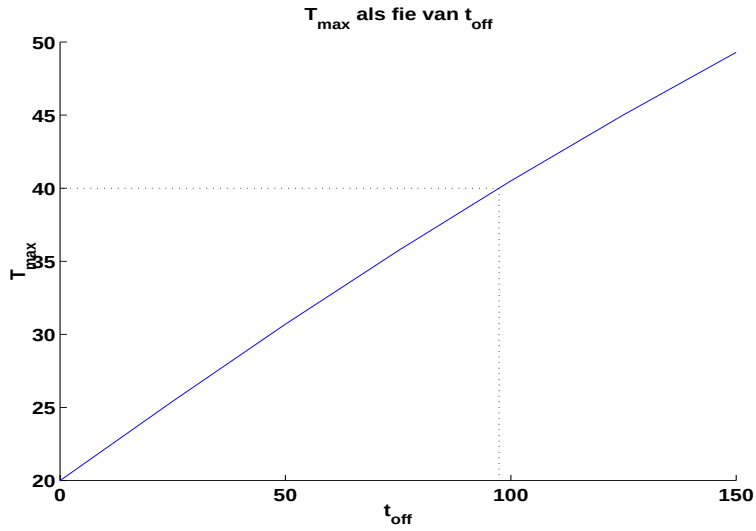
Maximale temperatuur

Vermits we de temperatuur in functie van de tijd in veel punten hebben berekend kunnen we in matlab eenvoudig de maximale waarde bepalen.



Maximale temperatuur als functie van t_{off}

We hebben voor verschillende waarden de maximale temperatuur bepaald. Als we dit tekenen zien we dat dit ongeveer een lineair verloop heeft.



Bepalen van t_{off} voor een maximaal te bereiken temperatuur

Stel dat de maximaal te bereiken temperatuur 40 is. Vermits het om een bijna lineair verloop gaat kunnen we inverse interpolatie toepassen en de tabelwaarden $(T_{\max}, t_{\text{off}})$ interpoleren. De bekomen veelterm kunnen we evalueren voor $T_{\max} = 40$ en de bekomen waarde is de tijd t_{off} waarvoor $T_{\max} = 40$.

13.5.2 Matlab experimenten

Programmeer de methodes in matlab, of gebruik `NW/NW/wamax/matlab/index.html` om een aantal experimenten te doen.

- Hoeveel keer moet men de differentiaalvergelijking oplossen om een goed idee te krijgen over het verloop van $T_{\max}$ als functie van t_{off}?
- Als de begintemperatuur van de vloeistof T_0 niet gelijk is aan de omgevingstemperatuur, heeft dit dan grote veranderingen als gevolg? Door $t_{\text{off}} = t_0$ te kiezen kan men

het verloop van de temperatuur van de vloeistof door gewone afkoeling/opwarming krijgen.

- Zou het zin hebben om $T_{\max}$ te bepalen met behulp van een routine die een 1-dimensionaal optimalisatieprobleem oplost (met of zonder afgeleiden)?
- Neem bijvoorbeeld $T_0 = 20$ en $T_u = 30$. Op welk ogenblik wordt dan de maximale temperatuur bereikt als men een kleine t_{off} opgeeft? Kun je dan de t_{off} bepalen die zo vlug mogelijk de vloeistof op een temperatuur van 30 graden brengt, zonder daar bovenuit te stijgen.
- Stel dat je de temperatuur van de vloeistof binnen de grenzen van 40 en 50 graden moet houden. Kun je dan een aan/uit strategie berekenen voor het verwarmingselement die een minimum aan omschakelingen vereist?