Numerieke Wiskunde (WISB251)

Tristan van Leeuwen

Inhoudsopgave

Hoofdstuk 1

Introductie

Wiskundige modellen worden op veel plekken gebruikt; denk bijvoorbeeld aan weermodellen, het genereren van driedimensionale plaatjes in een MRI-scanner of het ontwerpen van een brug. In al deze toepassingen wordt ten eerste een wiskundig model opgesteld dat (een bepaald aspect) van de werkelijkheid omzet in bijvoorbeeld een stelsel vergelijkingen, een integraal of een differentiaalvergelijking. In het modelvormingsproces worden bepaalde aannamen gedaan en hierbij worden *modelfouten* geïntroduceerd. De volgende stap in het proces is het oplossen van het model (stelsel vergelijkingen, integraal, differentiaalvergelijking,etc.). In sommige gevallen kan dit analytisch en kan het antwoord worden uitgedrukt in termen van bekende functies (veeltermen, trigonometrische functies, etc.), maar vaak is dit niet het geval. We zullen in zo'n geval de oplossing moeten benaderen met behulp van bekende functies. We maken in dit proces een *benaderingsfout*. Tot slot zullen we de oplossing ook moeten uitrekenen om er iets mee te kunnen. Hier krijgen we te maken met de beperkingen van ons rekenvermogen. We kunnen immers maar met een eindig aantal cijfers rekenen om een reëel getal weer te geven. We krijgen hier te maken met *afrondfouten*. Voor de tijd van de digitale computer werden zulke berekeningen grotendeels met hand gedaan, eventueel met behulp van mechanische rekenmachines en tabellenboeken voor functies als sin en log. Tegenwoordig gebruiken we hier natuurlijk computerprogramma's voor, maar ook daar zijn afrondfouten onvermijdelijk.

Voorbeeld 1.1. ***Röntgentomografie:*** *Als we willen weten hoe iets er vanbinnen uitziet kunnen we het in een Röntgenscanner leggen. Wanneer we een Röntgenfoto van één kant nemen krijgen we wel een aanzicht van het object, maar nog geen volledige informatie over hoe het object er in drie dimensies uit ziet. Hiervoor moeten we Röntgenfoto's van verschillende kanten nemen. Allereerst hebben we een model nodig voor de interactie tussen Röntgenstralen en materie. De* wet van Lambert-Beer *zegt dat de intensiteit van de Röntgenstraal vóór en na het passeren van het object aan elkaar gerelateerd zijn:*

$$I_1 = I_0 e^{-\int u(x(s),y(s))\mathrm{d}s},$$

waarbij $u(x,y)$ de absorptiecoëfficiënt is van het object in de scanner en de integraal een lijnintegraal is langs het pad van de Röntgenstraal. Nemen we vervolgens metingen van alle kanten dan kunnen we de metingen en de absorbtiecoefficient van het object aan elkaar relateren met de Radontransformatie

$$p(s,\theta) = \int_{-\infty}^{\infty} u\left(r\sin\theta + s\cos\theta, -r\cos\theta + s\sin\theta\right)\mathrm{d}r.$$

We hebben nu een geïdealiseerd wiskundig model. Om dit model geschikt te maken voor berekeningen moeten we een manier verzinnen om de integraal te kunnen berekenen voor willekeurige (en mogelijk ingewikkelde) functies u. Een mogelijkheid is om de functie u weer te geven op een rooster van pixels. Doen we hetzelfde met de metingen p dan krijgen we een lineare relatie tussen de pixelwaarden p_i en u_j:

$$p_i = \sum_{j=1}^{n} w_{ij} u_j,$$

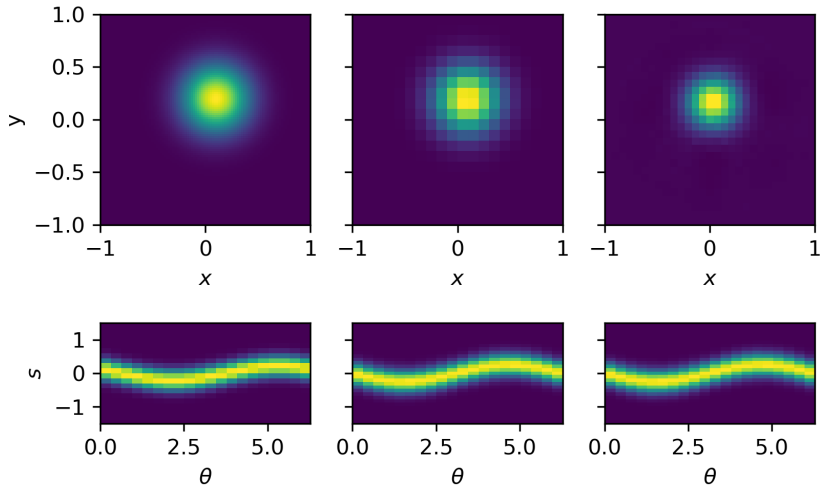
waarbij w_{ij} de lengte is van lijnsegment dat door de $j-$de pixel naar de i-de detectorpixel loopt. We hebben hier een benaderingsfout gemaakt door de functie weer te geven op een rooster van pixels. Voor gegeven metingen p_i kunnen we nu het stelsel vergelijkingen $p = Wu$ oplossen. Zoals je je kunt voorstellen kan dit een groot stelsel zijn, in de praktijk wel tot een miljoen onbekenden! De resultaten zullen niet allemaal mooie gehele getallen of breuken zijn zodat afrondfouten bij het oplossen onvermijdelijk zijn.
Een voorbeeld is te zien in figuur 1.1. Hier gaan we uit van

$$u(x,y) = \exp(-\alpha((x-x_0)^2 + (y-y_0)^2)),$$

met $\alpha = 10$, $x_0 = 0.1$, $y_0 = -0.2$ en rekenen daarvoor de Radontranformatie uit:

$$p(s,\theta) = \exp(-\alpha(s - x_0\cos\theta - y_0\sin\theta)).$$

De numerieke benadering van u op een grid van pixels en de bijhorende meting p zijn ook te zien. Tot slot kunnen het stelsel vergelijkingen oplossen om zo een benadering van de oplossing te krijgen. De totale fout tussen de oplossing en het oorspronkelijke beeld bevat dus twee bijdragen; de benadering van de Radontransformatie en de benadering van de oplossing van het stelsel vergelijkingen.



Figuur 1.1: Voorbeeld van tomografische reconstructie. Links de functie u en de Radontransformatie ervan. In het midden de numerieke benadering van de Radonsformatie en tot slot, rechts de oplossing van het stelsel vergelijkingen.

1.1 Wat is numerieke wiskunde

In dit vak leren we de basis van het numeriek oplossen van (differentiaal-)vergelijking en het benaderen van functies afgeleiden en integralen. Zulke technieken worden vaak gebruikt (zelfs in je rekenmachine) en we kunnen onze basiskennis van infinitesimaalrekening, lineaire algebra en analyse gebruiken om ze te analyseren. Centraal staan telkens de vragen: Hoe groot is de benaderingsfout? Wat is de invloed van afrondfouten op de berekening? Hoe efficiënt is de berekening?
Abstract gezien kunnen we alle vraagstukken die we gaan behandelen als volgt formuleren:

Gegeven een afbeelding, G, en een invoer f, geef een benadering voor de uitvoer $u = G(f)$.

Wannneer we bijvoorbeeld de integraal van een functie f willen benaderen hebben we $G(f) = \int_a^b f(x)\mathrm{d}x$. Als we een stelsel vergelijkingen $Au = f$ willen oplossen dan is $G(f) = A^{-1}f$.
De nadruk van dit vak ligt op het begrip van de behandelde methoden, de basisbegrippen: benaderingsfout, stabiliteit, afrondfout, efficiëntie. Er komen weinig nieuwe wiskundige begrippen aan bod; we bouwen voort op wat je hebt geleerd bij infinitesemaalrekening, lineaire algebra en analyse. Verder gaan we in dit vak gebruik maken van numerieke experimenten om methoden te testen. We kunnen op die manier stellingen verifiëren en bijvoorbeeld zien hoe goed de afgeschatte bovengrenzen zijn. Ook kunnen de resultaten van numerieke experimenten aanleiding geven tot meer gedetailleerde analyse van een methode.
Hieronder een kort overzicht van een aantal basisbegrippen die we nodig zullen hebben.

1.1.1 Hoe groot is de fout?

We gaan in dit vak getallen, vectoren en functies benaderen. De objecten die we gaan benaderen vormen in ons geval een *vectorruimte* (over $\mathbb{R}$). In alle toepassingen die we gaan zien kunnen we ook een norm, $\|\cdot\|$ en vaak ook een inproduct, $\langle\cdot,\cdot\rangle$ definiëren. Nu kunnen we de fout in een benadering $\widetilde{u}$ van een vector u meten door simpelweg de norm van het verschil te nemen. We noemen dit de *absolute fout*:

$$\|u - \widetilde{u}\|.$$

Omdat de grootte van deze fout niet direct iets zegt over de nauwkeurigheid (denk aan $u = 0.001$ en $\widetilde{u} = 0.002$) introduceren we de *relatieve fout:*

$$\frac{\|u - \widetilde{u}\|}{\|u\|}.$$

Voorbeeld 1.2. $\mathbb{R}^n$*: We kennen de definities nog van lineaire algebra. Voor twee vectoren* $u = (u_1, u_2, \ldots, u_n)$ *en* $v = (v_1, v_2, \ldots, v_n)$ *hebben we*

- *het in-product:* $\langle u, v\rangle = \sum_{k=1}^{n} u_k v_k$
- *de Euclidische* $2-$*norm:* $\|u\|_2 = \sqrt{\sum_{k=1}^{n} u_k^2}$.

Naast de gebruikelijke Euclidische norm zijn er nog twee andere normen die soms van pas komen:

- *de* $1-$*norm:* $\|u\|_1 = \sum_{k=1}^{n} |u|$
- *de supremum-norm:* $\|u\|_\infty = \max_k |u_k|$

We kunnen makkelijk nagaan dat deze twee inderdaad voldoen aan de eisen voor een norm. Bovendien kunnen we de volgende nuttige relaties afleiden:

- $\|u\|_\infty \leq \|u\|_2 \leq \|u\|_1 \leq \sqrt{n}\|u\|_2 \leq n\|u\|_\infty$.

Hieruit volgt ook dat deze normen equivalent *zijn.*

Voorbeeld 1.3. ***Continue functies:*** *De normen uit het vorige voorbeeld laten zich op een natuurlijk manier vertalen naar normen voor continue functies op een eindig interval (*$f \in C[a, b]$*)*

$$\|f\|_p = \left(\int_a^b |f(x)|^p \mathrm{d}x\right)^{1/p}.$$

In het bijzonder hebben we $\|f\|_\infty = \sup_{x\in[a,b]} |f(x)|$*. Merk op dat deze normen niet langer equivalent zijn! We kunnen bovendien het volgende inproduct definieren:*

$$\langle f, g\rangle = \int_a^b f(x)g(x)\mathrm{d}x,$$

zodat inderdaad $\|f\|_2 = \sqrt{\langle f, f\rangle}$.

We kunnen het concept van een norm ook uitbreiden voor lineaire operaties. Voor een lineaire operatie $G : V \to W$, waarbij V, W genormeerde vectorruimten zijn is de norm van G gedefiniëerd als

$$\sup_{\|u\|} \frac{\|Gu\|}{\|u\|},$$

waarbij $\|\cdot\|$ een gekozen vectornorm is. Afhankelijk van welke vectornorm we kiezen, kunnen we dus een bijbehorende operatornorm definiëren.
Verder kunnen voor lineaire operators de *spectrale radius* van G definiëren:

$$\rho(A) = \lim_{k\infty} \|G^k\|^{1/k}.$$

Hiervoor geldt dat $\rho(G) \leq \|G\|$ voor *iedere* vectornorm $\|\cdot\|$.

Voorbeeld 1.4. ***Geïnduceerde matrixnorm:*** *We kunnen een matrixnorm definiëren op basis van een vectornorm:*

$$\|A\| = \max_u \frac{\|Au\|}{\|u\|}.$$

Drie veelgebruikte matrixnormen zijn:

$$\|A\|_1 = \max_i \sum_j |a_{ij}|,$$

$$\|A\|_\infty = \max_j \sum_i |a_{ij}|,$$

en

$$\|A\|_2 = \max_i \sqrt{\lambda_i},$$

waarbij λ_i een eigenwaarde van A^TA is.
Uit de definitie zien we ook direct dat

$$\|Au\| \leq \|A\|\|u\|.$$

Voorbeeld 1.5. ***Spectrale radius van een vierkante matrix:*** *De spectrale radius van een vierkante matrix A met eigenwaarden $\lambda_1, \lambda_2, \ldots, \lambda_n$ is gedefiniëerd als*

$$\rho(A) = \max_i |\lambda_i|.$$

Bovendien hebben we

$$\rho(A) \leq \|A\|,$$

voor iedere geïnduceerde matrixnorm.

Voorbeeld 1.6. ***Frobeniusnorm:*** *Er zijn ook matrixnormen die niet direct volgen uit de bijbehorende vectornorm. Een bekende is de* Frobeniusnorm*:*

$$\|A\|_F = \sqrt{\sum_{ij} a_{ij}^2}.$$

1.1.2 Foutanalyse, stabiliteit en efficiëntie

We willen $u = G(f)$ voor een gegeven afbeelding G en invoer f benaderen. We gebruiken hiervoor een benadering $\widetilde{G}$. De *voorwaartse fout* is dan gegeven door $\|\widetilde{u} - u\|$ met $\widetilde{u} = \widetilde{G}(f)$. De *terugwaartse fout* is gedefinieerd als $\|\widetilde{f} - f\|$ met $\widetilde{u} = G(\widetilde{f})$. We zoeken hier dus naar een invoer $\widetilde{f}$ waarvoor de benadering $\widetilde{u}$ een *exacte* oplossing is. In het volgende voorbeeld wordt duidelijk waarom we vaak beide nodig hebben.

Voorbeeld 1.7. ***Terugwaartse foutanalyse:*** *We willen het stelsel vergelijkingen $Ax = b$ oplossen met*

$$A = \begin{pmatrix} 99 & 98 \\ 100 & 99 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

De oplossing is $x = (1, -1)^T$.
Stel, we krijgen een benadering van de oplossing $\widetilde{x} = (2.97, -2.99)^T$. Dit lijkt een slechte benadering, maar het is de oplossing van het stelsel met rechterlid $\widetilde{b} = (1.01, 0.99)^T$.
Bekijken we de benadering $\widetilde{x} = (1.01, -0.99)^T$ dan lijkt deze goed te zijn, maar we zien dat dit de oplossing is van het stelsel met rechterlid $\widetilde{b} = (-1.97, -1.97)^T$; een heel ander stelsel vergelijkingen dus!

De twee fouten zijn aan elkaar gerelateerd via het *conditiegetal* van het probleem

$$\frac{\|\widetilde{u} - u\|}{\|u\|} \leq \kappa(G) \frac{\|\widetilde{f} - f\|}{\|f\|}.$$

Dit zegt iets over de gevoeligheid van de berekening voor kleine verstoringen in de invoer; als het probleem een groot conditiegetal heeft is het zeer gevoelig voor kleine verstoringen.

Voorbeeld 1.8. ***Conditiegetal van een functie:*** *Wanneer we een functie evalueren $y = g(x)$ op het inverval $[a, b]$ kunnen we ons afvragen hoe gevoelig g is voor kleine veranderingen in de invoer. Met behulp van de Middelwaardestelling vinden we*

$$|y - y'| \leq \sup_{\xi \in [a,b]} |g'(\xi)||x - x'|.$$

Voor de relatieve fout krijgen we dan

$$\frac{|y - y'|}{|y|} \leq \sup_{\xi \in [a,b]} \left| \frac{\xi g'(\xi)}{g(\xi)} \right| \frac{|x - x'|}{|x|},$$

waarbij de factor

$$\kappa = \sup_{\xi \in [a,b]} \left| \frac{\xi g'(\xi)}{g(\xi)} \right|$$

het conditiegetal van g is. Voor de functie $g(x) = x^2$ op het interval $[0, \infty)$ vinden bijvoorbeeld dat $\kappa = 2$.

Behalve de *nauwkeurigheid* gaan we ook kijken naar de *stabiliteit* van de benadering. We willen dan weten welke invloed kleine verstoringen in de invoer hebben op de benadering. Zulke verstoring worden bijvoorbeeld veroorzaakt door afrondfouten. Zo kan het zijn dat een benadering erg nauwkeurig is in *exacte aritmetiek* maar dat de afrondfouten de benadering waardeloos maken.
Concreet gezien, willen we weten hoe goed we $u = G(f)$ benaderen met $\widetilde{u} = \widetilde{G}(\widetilde{f})$, waarbij $\|f - \widetilde{f}\| \leq \eta \|f\|$. We krijgen dan:

$$\|\widetilde{u} - u\| \leq \|G(f) - \widetilde{G}(f)\| + \|\widetilde{G}(f) - \widetilde{G}(\widetilde{f})\|.$$

De eerste term is de benaderingsfout en de tweede term hangt samen met het condititiegetal van $\widetilde{G}$:

$$\frac{\|\widetilde{G}(f) - \widetilde{G}(\widetilde{f})\|}{\|\widetilde{G}(f)\|} \leq \kappa(\widetilde{G})\eta.$$

Het valt niet altijd te voorkomen dat de benadering $\widetilde{G}$ een groot conditiegetal heeft. Een goede benadering zal zich immers net zo moeten gedragen als het oorspronkelijke probleem, en dus ongeveer hetzelfde conditiegetal hebben. Wat we willen voorkomen is dat de benadering een veel groter conditiegetal heeft dan het oorspronkelijke probleem. In dat geval kan de afrondfout wel eens groter zijn dan de benaderingsfout! De samenhang tussen stabiliteit en voorwaartse en terugwaartse fout wordt duidelijk in het volgende voorbeeld.

Voorbeeld 1.9. ***Stabiliteit:*** *Wanneer de functie $g(x) = \exp(-x)$ willen benaderen op het interval $[0, 1]$ kunnen we een eenvoudige benadering gebruiken:*

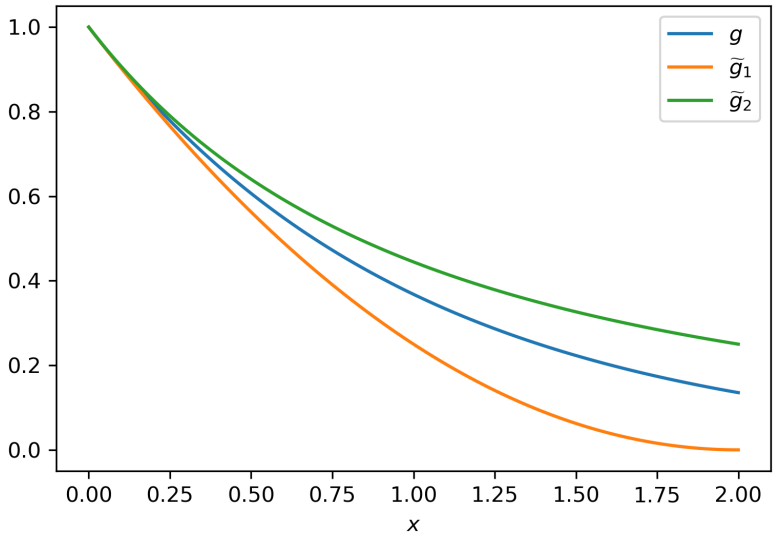
$$\widetilde{g}_1(x) = (1 - x/2)^2.$$

Een andere benadering is

$$\widetilde{g}_2(x) = (1 + x/2)^{-2}.$$

In figuur 1.2 zien we een grafiek van deze benaderingen. De conditiegetallen van g en de benaderingen zijn respectievelijk 1, 2 en $2/3$.

Tot slot kijken we ook naar de manier waarop we een benadering kunnen construeren. Dit gebeurt altijd door middel van een concreet algoritme dat na een eindig aantal stappen een benadering $\widetilde{u}$ teruggeeft. We willen dan graag weten hoe *efficiënt* het algoritme is. We kunnen dit bijvoorbeeld uitdrukken in de complexiteit (het aantal elementaire berekeningen) van het algoritme als functie van de benaderingsfout of de dimensie van de invoer.



Figuur 1.2: Two different approximations of $g(x) = e^{-x}$.

1.1.3 A-priori en a-posteriori foutanalyse

We zullen grofweg twee soorten foutafschattingen maken; *a-priori* en *a-posteriori* afschattingen. Een a-priori afschatting hangt in de regel van de echte oplossing af, en kan dus in de praktijk niet worden berekend. Zulke afschattingen zijn wel nuttig om het gedrag van de fout theoretische te bestuderen. Willen we de fout in de praktijk schatten, dan hebben we alleen de berekenden benadering tot onze beschikking. Een foutschatting die alleen alleen gebruik maakt van de benaderde oplossing, noemen we een a-posteriori foutschatting.

1.1.4 Numerieke experimenten

De nauwkeurigheid van een benadering en de efficiëntie van het bijbehorende algoritme zijn vaak goed te analyseren. Zo'n analyse gaat echter vaak uit van het slechtste geval en geeft dus een *bovengrens* voor de fout of de *asymptotische complexiteit*. Om beter te onderzoeken hoe een algoritme zich gedraagt op een bepaalde klasse van problemen gebruiken we *numerieke experimenten*. Zulke experimenten bieden ook uitkomst in het geval dat analyse te moeilijk is. Om gedegen conclusies te kunnen trekken uit zulke experimenten zullen we zorgvuldig te werk moeten gaan. Observaties uit numerieke experimenten zijn vaak aanleiding voor verdere analyse.

Tijdens dit vak gaan jullie zelf met Python aan de slag om verschillende methoden te testen. Door tabellen of grafieken van de resultaten te analyseren kun je een theoretische analyse verifiëren of geeft onverwacht gedrag juist aanleiding tot meer gedetailleerde analyse. Het is vaak zinnig om eerst eens te kijken of iets werkt; een hypothese op te stellen en vervolgens proberen deze wiskundig te bewijzen.

1.2 Onderwerpen

Hieronder een kort overzicht van de onderwerpen die we in dit vak gaan behandelen aan de hand van een aantal voorbeelden.

Table 1.1. Opeenvolgende benaderingen van $\sqrt{2}$

k	x_k
0	1.0
1	1.5
2	1.4166666666666665
3	1.4142156862745097
4	1.4142135623746899
5	1.414213562373095
6	1.414213562373095
7	1.414213562373095
8	1.414213562373095
9	1.414213562373095

1.2.1 Afrondfouten (H2)

In de praktijk kunnen we alleen maar rekenen met een eindig aantal cijfers. Dit levert problemen op voor irrationale getallen, maar ook voor breuken ($\frac{1}{3} = 0.3333\ldots$). De fout die we maken door een reëel getal te benaderen met een eindig aantal cijfers noemen we de *afrondfout*. Met name het effect van het herhaaldelijk afronden bij lange berekeningen kan rampzalig zijn.

Voorbeeld 1.10. ***Afrondfouten in het nieuws***
Een aantal voorbeelden van het rampzalige gevolg van onvoorziene afrondfouten is te vinden op deze Blog.

1.2.2 Vergelijkingen oplossen (H3, H4)

We zijn al bekend met stelsels lineaire vergelijkingen en polynoomvergelijkingen in één variabele. Als het stelsel niet te groot is (minder dan 10 variabelen) of het polynoom van lage orde, dan kunnen we deze met de hand oplossen. We moeten in ons achterhoofd houden dat we uiteindelijk numerieke waarden willen hebben. In plaats van $x = \sqrt{2}$ als oplossing van de vergelijking $x^2 = 2$ worden we dus genoodzaakt met een benadering te werken $x = 1.41\ldots$. Als we meerdere berekeningen achterelkaar doen dan moeten we ons zorgen maken over de invloed van deze benaderingen op het eindresultaat. Een paar voorbeelden:

Voorbeeld 1.11. ***Benaderen van*** $\sqrt{2}$***:*** *De Babyloniërs hadden een praktische manier om de wortel van* 2 *te benaderen door middel van een recursieve berekening:*

$$x_{k+1} = (x_k + 2/x_k)/2,$$

waarbij x_0 *een beginschatting is. Waarom werkt deze benadering? Hoe groot is de fout* $|x_k - \sqrt{2}|$*? Convergeert de reeks* $\{x_k\}$ *naar* $\sqrt{2}$ *als* $k \to \infty$ *voor alle* x_0*? In tabel 1.1 zien we het resultaat voor* $x_0 = 1$*.*

Voorbeeld 1.12. ***Gauss eliminatie:*** *Een bekende methode om een stelsel vergelijkingen op te lossen is Gauss-eliminatie. Laten we dit eens toepassen op het stelsel* $Ax = b$ *met* $b = (2, 1, 0, -1, -2, \ldots, -n + 2)$ *en*

$$A = \begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 & 1 \\ -1 & 1 & 0 & \ldots & 0 & 1 \\ -1 & -1 & 1 & \ddots & \vdots & 1 \\ \vdots & \ddots & -1 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 \\ -1 & \ldots & \ldots & \ldots & -1 & 1 \end{pmatrix}.$$

Table 1.2. Gauss eliminatie

n	relatieve fout
20	0.0
40	0.0
60	0.3162277660168379
80	0.5700877125495689

De oplossing is in dit geval $x = (1, 1, \ldots, 1)$. In tabel 1.2 zien we de relatieve fout in de numerieke oplossing voor verschillende n. Voor kleine n lijkt er dus niks aan de hand, maar als we n iets groter kiezen gaat er iets flink mis!

Bij het oplossen van vergelijkingen moeten we ons eerst afvragen of er wel een oplossing is en zo ja, of er ook meerdere oplossingen mogelijk zijn. Tot slot zullen we zien dat sommige vergelijkingen moeilijker zijn op te lossen dan andere. Het kan bijvoorbeeld zo zijn dat kleine afrondfouten een grote invloed hebben op de benadering van de oplossing. Vergelijkingen die niet aan deze eisen voldoen (er is een unieke, stabiele oplossing) noemen we *slecht gesteld*.

Voorbeeld 1.13. 2×2 ***matrix:*** *We willen stelsel oplossen met de volgende matrix:*

$$A = \begin{pmatrix} 1+\epsilon & 1 \\ 2 & 2 \end{pmatrix},$$

met $\epsilon > 0$. We weten dat deze matrix inverteerbaar is omdat $\det(A) = 2\epsilon \neq 0$. Voor een gegeven b is er dus een unieke oplossing $x = A^{-1}b$. Voor kleine ϵ kunnen we echter ook oneindig veel bijna-oplossingen *vinden, gegeven door*

$$\widetilde{x} = x + \alpha \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Voor deze oplossingen geld dat $\|A\widetilde{x} - b\|_2 = \alpha\epsilon$. We zien echter ook dat de oplossingen voor een gegeven fout $\delta = \alpha\epsilon$ willekeurig ver van de echte oplossing x kunnen liggen: $\|x - \widetilde{x}\|_2 = \sqrt{2}\delta\epsilon^{-1}$. De oplossing is in dit geval niet stabiel; een kleine verandering in het stelsel vergelijkingen kan een grote verandering in de oplossing teweegbrengen. Dit komt in dit geval doordat de matrix A slecht geconditioneerd is voor kleine ϵ.

1.2.3 Functies benaderen (H5, H6)

Het benaderen van functies is het volgende onderwerp. Hoe doel hier is om een (ingewikkelde) functie $f \in C[a, b]$ te benaderen met een (eenvoudige) functie $\widetilde{f}$. Het kan zijn dat f zich niet goed leent voor numerieke berekeningen (denk aan $\sin x$) of dat f het resultaat is van een complexe berekening en zich niet laat uitdrukken in bekende functies. We gaan ervan uit dat we f (en wellicht de afgeleide(n) van f) wel kunnen evalueren op gegeven punten $x_i \in [a, b]$.

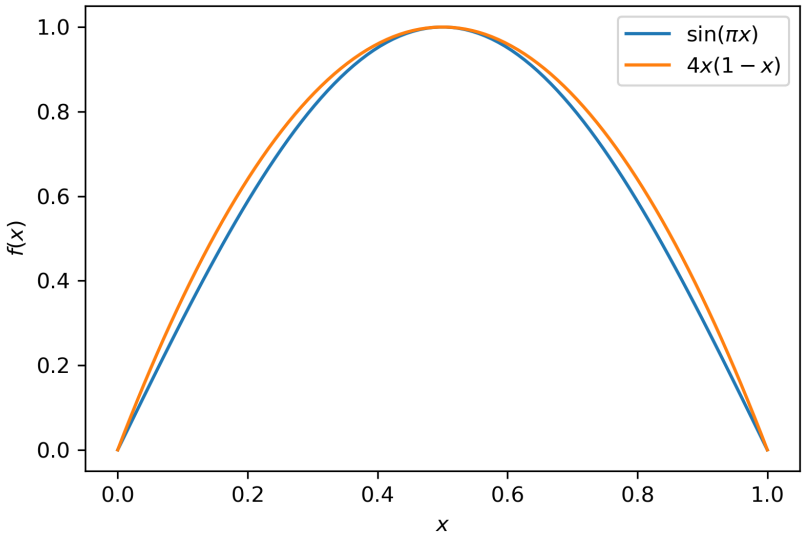
Voorbeeld 1.14. ***Benaderen van*** $\sin(\pi x)$***:*** *Als we de functie $f(x) = \sin(\pi x)$ willen benaderen op het interval $[0, 1]$ kunnen we uitgaan van een aantal bekende waarden, zoals $f(0) = 0$, $f(1/2) = 1$, $f(1) = 0$. Een benadering die voor de hand ligt is*

$$\widetilde{f}(x) = 4x(1 - x).$$

Zoals te zien in figuur 1.3 is dit een vrij goede benadering. Kunnen we ook uitrekenen hoe goed deze benadering is?

Vragen die we gaan behandelen tijdens dit vak zijn

- Hoe construeren we de benadering $\widetilde{f}$ uit de gegeven waarden $f(x_i)$
- Hoe groot is de fout $\|f - \widetilde{f}\|$ voor geveven steunpunten x_i
- Hoe kiezen we de steunpunten zo dat de fout zo klein mogelijk is



Figuur 1.3: De functie $\sin(\pi x)$ en een kwadratische benadering $\widetilde{f}(x) = 4x(1-x)$.

Als we eenmaal een gegeven functie kunnen benaderen, kunnen we ook de afgeleide en integralen van de functie benaderen als

$$f'(x) \approx \widetilde{f}'(x),$$

en

$$\int_a^b f(x)\mathrm{d}x \approx \int_a^b \widetilde{f}(x)\mathrm{d}x.$$

Uiteraard willen we ook hier weten hoe groot de fout is. Een ander aspect dat hier (ook) aan bod komt is het schatten van de fout. Dit is niet zo triviaal, we kunnen $\|f - \widetilde{f}\|$ immers niet uitrekenen omdat we de echte waarde f niet weten!

Voorbeeld 1.15. ***De raaklijn:*** *Als we de afgeleide van een functie willen benaderen kunnen we uitgaan van de formele definitie*

$$f'(x) = \lim_{h \downarrow 0} \frac{f(x+h) - f(x)}{h},$$

en simpelweg een kleine h kiezen. Passen we dit toe op de functie $f(x) = x^2$ op het punt $x = 1$ dan zien we in tabel 1.3 dat de afgeleide eerst steeds beter wordt benaderd voor kleinere h tot de fout ineens weer groter wordt.

Voorbeeld 1.16. ***De Riemannsom:*** *Een manier om de een integraal te benaderen is via een Riemannsom:*

$$\int_a^b f(x)\mathrm{d}x \approx h \sum_{k=1}^{n} f_k,$$

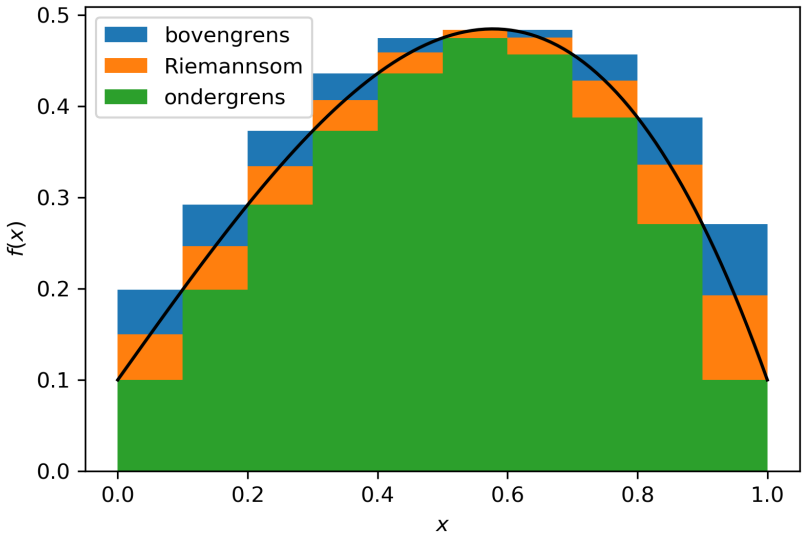
met $h = (b-a)/n$ en $f_k = f(h/2 + kh)$. We weten dat de fout naar nul gaat als $h \downarrow 0$ (aangenomen dat f Riemann-integreerbaar is). Hoe kunnen we de fout schatten voor een gegeven h? Een manier om door met boven- en ondersommen te rekenen en zo een boven- en ondergrens voor de integraal af te leiden:

$$h \sum_{k=1}^{n} f_k^- \leq \int_a^b f(x)\mathrm{d}x \leq h \sum_{k=1}^{n} f_k^+,$$

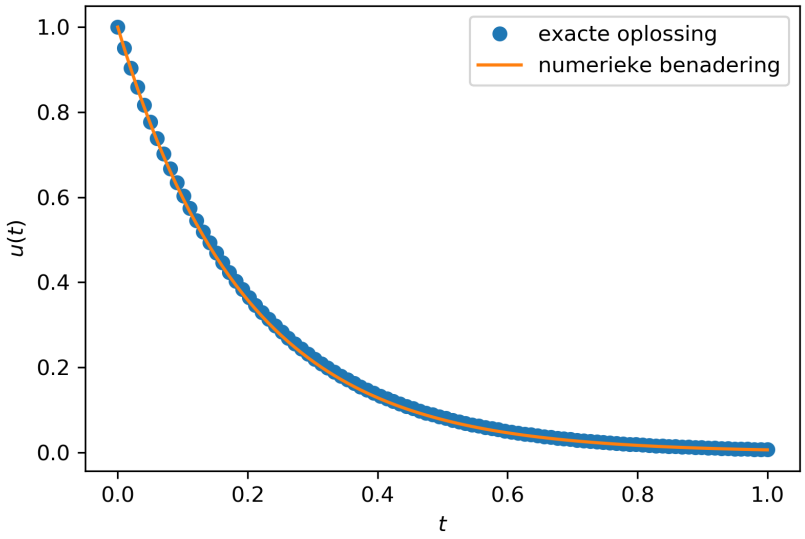
zoals geïllustreerd in figuur 1.4. We weten nu dat de fout niet groter kan zijn dan het verschil tussen de Riemannsom en de boven- en ondergrenzen.

Table 1.3. Benadering van de afgeleide van x^2 rond $x = 1$

h	$f'(x)$
1.0	3.0
0.1	2.100000000000002
0.01	2.0100000000000007
0.001	2.0009999999996975
0.0001	2.000099999999172
1e-05	2.00001000001393
1e-06	2.0000009999243673
1e-07	2.0000001010878066
1e-08	1.999999987845058
1e-09	2.000000165480742
1e-10	2.000000165480742
1e-11	2.000000165480742
1e-12	2.000177801164682
1e-13	1.9984014443252818
1e-14	1.9984014443252818
1e-15	2.220446049250313



Figuur 1.4: Riemannsom en boven- en ondergrens voor de functie $f(x) = 0.1 + x - x^3$ met $n = 10$.



Figuur 1.5: Numerieke oplossing van een eenvoudig beginwaardeprobleem.

1.2.4 Differentiaalvergelijkingen (H7)

Tot slot gaan we alles toepassen op het oplossen van (stelsels) differentiaalvergelijkingen. In het bijzonder gaan we kijken naar beginwaardeproblemen:

$$u'(t) = f(t, u(t)), \quad u(0) = u_0,$$

waarbij $u : \mathbb{R} \to \mathbb{R}^n$ en $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$.
In bepaalde gevallen kunnen we zulke differentiaalvergelijkingen met de hand oplossen. Voor $n = 1$ en $f(t, u) = a(t)u + g(t)$ krijgen we

$$u(t) = e^{A(t)} \left(C + \int e^{-A(s)} g(s) \mathrm{d}s \right),$$

waarbij A een primitieve is van a en C zo wordt gekozen dat aan de beginvoorwaarde wordt voldaan. In veel gevallen kunnen we vergelijking niet zo makkelijk oplossen en zullen we de oplossing moeten benaderen. Ook hier willen we graag weten hoe groot de fout is en of we de oplossing ook efficiënt kunnen uitrekenen.

Voorbeeld 1.17. ***Een kopje koffie:*** *We kunnen modelleren hoe een kopje koffie afkoelt met het volgende beginwaardeprobleem*

$$u'(t) = -\alpha u(t), \quad u(0) = u_0,$$

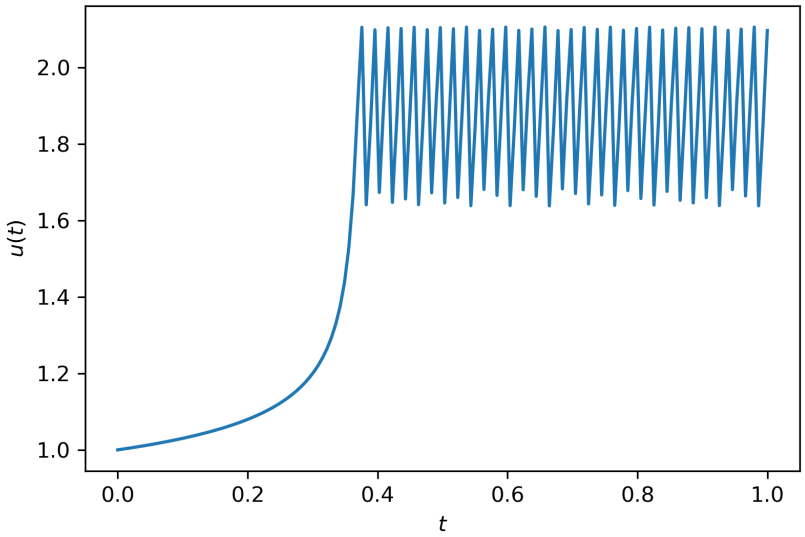
waarbij $\alpha > 0$ bepaalt hoe snel de koffie afkoelt. We weten natuurlijk dat de oplossing $u(t) = u_0 e^{-\alpha t}$ is, maar wat als we deze numeriek willen benaderen? Het ligt voor de hand om de afgeleide te benaderen als $u'(t) \approx (u(t+h) - u(t))/h$ en daaruit de volgende methode af te leiden om de oplossing te benaderen:

$$u(t+h) = u(t) - \alpha h u(t).$$

Op deze manier kunnen we vanaf $t = 0$ met stapjes van grootte h naar het eindtijdstip wandelen. Een voorbeeld is te zien in figuur 7.4. Deze simpele aanpak lijkt dus erg goed te werken!

Voorbeeld 1.18. ***Een chemische reactie:*** *De temperatuur van een bepaalde chemische reactie wordt bepaald door:*

$$u'(t) = \alpha(2 - u(t)) \exp\left(\beta - \beta/u(t)\right), \quad u(0) = 1,$$



Figuur 1.6: Numerieke oplossing van een niet-lineaire differentiaalvergelijking.

waarbij α, β constanten zijn. Wanneer we dezelfde benadering gebruiken als in het vorige voorbeelde krijgen we de oplossing uit figuur 7.2. De oplossing ziet er aardig uit, maar kunnen we deze ook vertrouwen?

Voorbeeld 1.19. ***Een slinger:*** *We kunnen een slinger (bij benadering) beschrijven met een tweede-orde differentiaalvergelijking*

$$u''(t) + \alpha^2 u(t) = 0,$$

met beginvoorwaarden $u(0) = u_0$ en $u'(0) = v_0$. We kunnen deze tweedeordevergelijking omschrijven naar een stelsel van eersteordedifferentiaalvergelijkingen door een nieuwe functie $v(t) = u'(t)$ in te voeren. We krijgen dan

$$w'(t) = Aw(t),$$

met $w(t) = (u(t), v(t))$, $w(0) = (u_0, v_0)$ en

$$A = \begin{pmatrix} 0 & 1 \\ -\alpha^2 & 0 \end{pmatrix}.$$

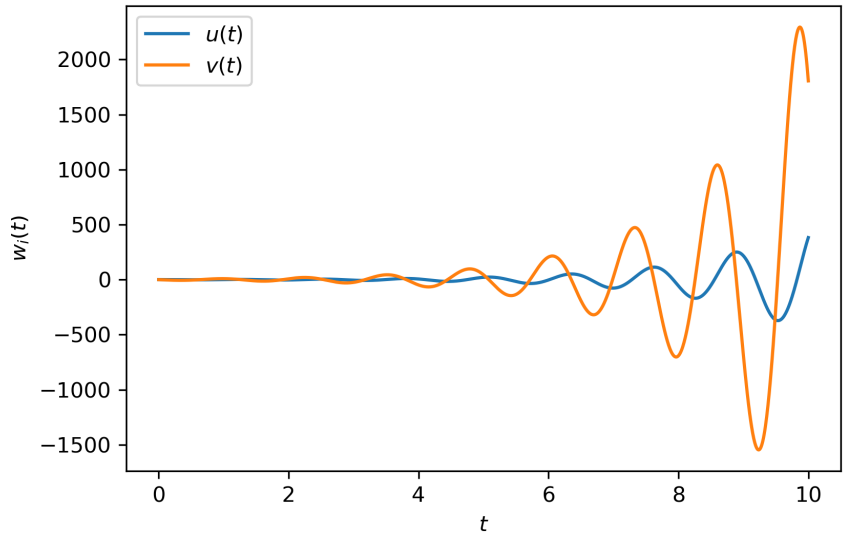
Wanneer we dezelfde benadering gebruiken als in het vorige voorbeeld zien we iets onverwachts (figuur 1.7), namelijk dat de uitslag van de slinger steeds groter wordt en dat deze steeds sneller gaat slingeren!

1.3 Opdrachten

Opdrachten met een 🖳-icoon zijn (deels) computeropdrachten. Vaak kun je code uit het hoofdstuk hergebruiken. Kijk daarvoor in het betreffende Jypter Notebook.

🖳 **Opdracht 1.1.** ***Wortel 2:*** *Onderzoek doormiddel van numerieke experimenten hoe gevoelig de methode van de Babyloniërs is voor de beginschatting x_0.*

🖳 **Opdracht 1.2.** ***Gauss eliminatie:*** *We zagen in het voorbeeld dat Gauss-eliminatie flink mis kan gaan. Onderzoek wat er gebeurt wanneer we in het stelsel uit het voorbeeld de laatste kolom op de tweede plek zetten en de rest doorschuiven.*



Figuur 1.7: Numerieke oplossing van een stelsel differentiaalvergelijkingen.

Opdracht 1.3. ***Taylor:*** *Op het interval* $[0, 2\pi]$ *kunnen we* $u(x) = \sin(x)$ *ook benaderen met een* n*-de orde Taylorbenadering* $\widetilde{u}_n(x)$ *rond* $x = 0$*. Geef een uitdrukking voor de restterm* $u - \widetilde{u}_n$ *in termen van de* $n + 1$*-ste afgeleide van* u*. Gebruik dit vervolgens om een bovengrens te vinden voor* $\|u - \widetilde{u}_n\|_\infty$*. Verifieer je bovengrens door middel van een numeriek experiment.*

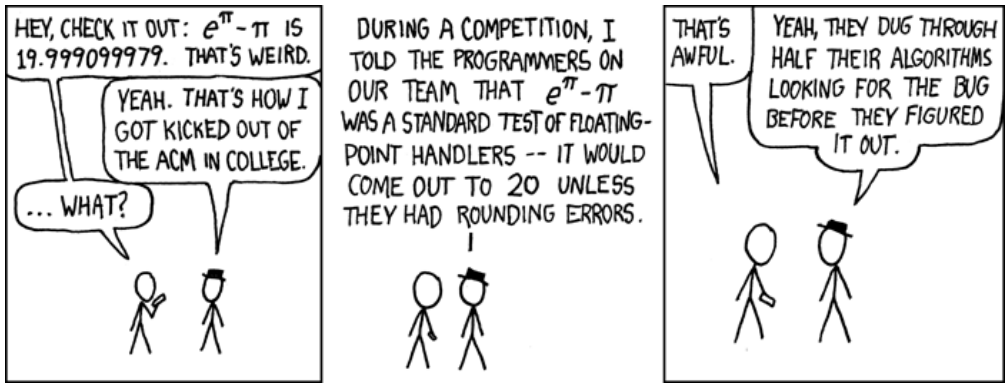
Opdracht 1.4. ***Koffie:*** *Onderzoek wat er gebeurt wanneer de koffie erg snel laten afkoelen (*α *heel groot). Werkt de benadering uit het voorbeeld beter voor grote of kleine* α*? Hoe hangt de fout af van de gekozen stapgrootte* h*?*

Opdracht 1.5. ***Chemische reactie:*** *Hoe zou je onderzoeken of de oplosmethode die in het voorbeeld wordt gebruikt betrouwbaar is?*

Opdracht 1.6. ***Slinger:*** *Onderzoek of je een goede oplossing voor de slinger kunt krijgen door een kleinere stapgrootte te kiezen.*

Hoofdstuk 2

Eindige precisie



Als we berekeningen doen op de computer moeten we benaderingen gebruiken. We weten namelijk dat niet ieder getal een eindige representatie heeft. Denk bijvoorbeeld aan $\sqrt{2} = 1.41\ldots$ en $\pi = 3.1415\ldots$. De vraag die we in dit hoofdstuk gaan bekijken is de volgende:

Hoe benaderen we een reëel getal $x \in \mathbb{R}$ met een eindig aantal cijfers?

Laten we eens kijken hoe we zo efficiënt mogelijk reële getallen kunnen weergeven als we slechts 2 cijfers van $0-9$ willen gebruiken. We kunnen natuurlijk werken met breuken, en een getal weergeven als

$$x = \frac{d_0}{d_1},$$

waarbij $d_i \in \{0,1,2,3,4,5,6,7,8,9\}$. Om berekeningen te doen is dit echter niet erg aantrekkelijk. Een andere optie is om een vast aantal plekken voor en achter de komma vast te leggen en ieder reëel getal weer te geven als bijvoorbeeld

$$x = d_0 + d_1 \cdot 10^{-1}.$$

Op deze manier kunnen we reële getallen benaderen met een vaste *absolute fout* van $\sim 5 \cdot 10^{-2}$. Het nadeel hiervan is dat we kleine getallen met minder nauwkeurigheid kunnen weergeven dan grotere. Het probleem hier is natuurlijk dat we de reële as nu hebben opgedeeld in gelijke stukjes. We zoeken dus een opdeling van de reële as die een uniforme *relatieve fout* oplevert. Zo'n indeling is gegeven door de *zwevendekommagetallen*. Met precies hetzelfde aantal cijfers als het voorgaande voorbeeld kunnen we een getal ook weergeven als

$$x = d_0 \cdot 10^{d_1 - 5}.$$

In dit geval hebben we de reële as opgedeeld in stukjes die kleiner worden naarmate we dichter bij 0 komen. Op deze manier bereiken we een constante relatieve fout van $\sim 5 \cdot 10^{-1}$.

Voorbeeld 2.1. ***Opdelen van de reële as.*** *We kunnen de opdeling van de reële as visualiseren door alle mogelijke getallen in bovengenoemde talstelsels te genereren. Op die manier kunnen ook de maximale relatieve fout uitrekenen. Een voorbeeld is te zien in figuur 2.1.*

2.1 Zwevendekommagetallen

De formele definitie van een zwevendekommagetal is als volgt:

Definitie 2.1. ***Zwevendekommagetal.*** *De zwevendekommarepresentatie van $x \in \mathbb{R}$ heeft de vorm*

$$fl(x) = \pm \sum_{k=1}^{n} d_k \beta^{e-k},$$

waarbij $\beta \in \mathbb{N}$ de basis *is, $e \in \mathbb{Z}$ de* exponent *en $d_k \in \{0, 1, \ldots, \beta - 1\}$ de cijfers. Een zwevendekommatalstelsel wordt gedefinieerd door (β, L, U, n) waar bij L en U de onder- en bovengrens van de exponent zijn. Om te zorgen dat ieder getal een unieke representatie heeft, eisen we dat $d_0 \neq 0$. Het kleinste reële getal dat we dan kunnen weergeven is β^{L-1}; het grootste (in absolute zin) is $\beta^U(1 - \beta^{-n})$.*

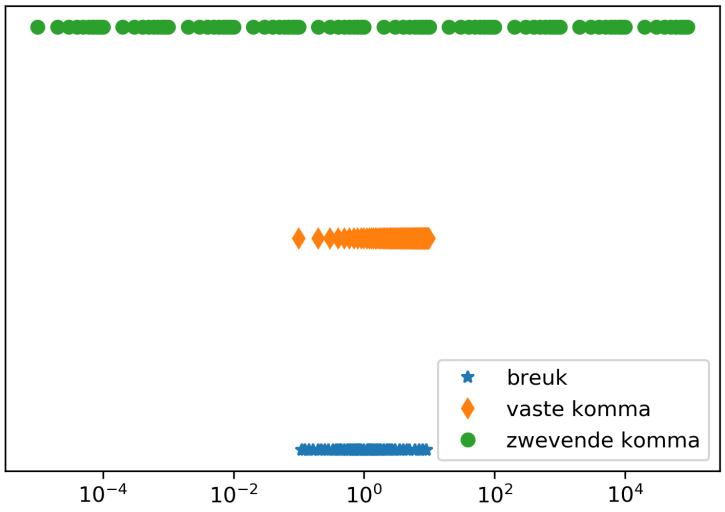
We kunnen op deze manier alle reële getallen tussen de onder en bovengrens benaderen met dezelfde relatieve fout:

Stelling 2.1. ***Representatiefout.*** *Voor een reëel getal x in het bereik van het zwevendekommatalstelsel geldt*

$$\frac{|x - fl(x)|}{|x|} \leq \eta.$$

waarbij $\eta = \frac{1}{2}\beta^{-n}$ ook wel de afrondeenheid *wordt genoemd. Dit impliceert dat $\exists \epsilon$ met $|\epsilon| \leq \eta$ zodat*

$$fl(x) = x(1 + \epsilon).$$



Figuur 2.1: Opdeling van de reële as voor verschillende talstelsels

Stelling 2.2. ***Alternatieve vorm van de representatiefout:*** *Gegeven x in het bereik van het zwevendekommatalstelsel kunnen $\exists\delta$ met $|\delta| \leq \eta$ zodat*

$$fl(x) = x(1+\delta)^{-1}.$$

Een veelgebruikt zwevendekommatalstelsel is de *IEEE double precision* standaard met $(\beta, L, U, n) = (2, -1023, 1024, 52)$. Voor één getal zijn 64 bits nodig (52 voor de cijfers, 11 voor de exponent en 1 voor het teken). Speciale waarden zijn gereserveerd voor getallen die te groot of te klein zijn (`inf`, `0`) en niet gedefinieerde berekeningen (`nan`). In Python is deze beschikbaar als het type `float`. De afrondeenheid voor dit stelsel is $\eta \approx \frac{1}{2} \cdot 10^{-16}$.
Er is ook een minder nauwkeurige variant, *single precision*, met $(2, -127, 128, 23)$ die 32 bits in beslag neemt. Deze is in Python (via `numpy`) beschikbaar als het type `float32`.
Wanneer geheugenbesparing cruciaal is wordt er soms ook wel gerekend met *half precision* met $(2, -15, 14, 10)$ die slechts 16 bits in beslag neemt. Deze is in Python beschikbaar als (je raadt het al) `float16`.

Voorbeeld 2.2. ***Representatiefout.*** *We kunnen nu onderzoeken hoe goed we bijvoorbeeld de breuk $\frac{1}{3}$ kunnen benaderen als `float`, `float32` of `float16`.*

```
float64(1/3) =  0.33333333333333331483


float32(1/3) =  0.33333334326744079590


float16(1/3) =  0.33325195312500000000
```

2.2 Rekenen met zwevendekommagetallen

Nu willen we gaan rekenen met zwevendekommagetallen. Helaas is de som van 2 zwevendekommagetallen in het algemeen geen zwevendekommagetal! De zwevendekommagetallen vormen niet eens een lichaam. Zo zal dus onder meer de volgorde van de berekeningen uitmaken en moeten we ons zorgen maken

over de nauwkeurigheid van elementaire berekeningen zoals optellen, aftrekken, delen en vermenigvuldigen. We kunnen door herhaaldelijke berekeningen gemakkelijk nauwkeurigheid verliezen. Om dit zoveel mogelijk te voorkomen worden er in iedere implementatie van de elementaire berekeningen $(+, -, /, *)$ extra bits meegenomen om zo te garanderen dat de relatieve fout na iedere elementaire berekening binnen de perken blijft. In het bijzonder hebben we dan de volgende garantie:

Stelling 2.3. ***Exacte afronding.*** *De relatieve fout na optellen, aftrekken, delen of vermenigvuldigen van twee zwevendekommagetallen is begrensd door η. We hebben dus*

$$fl(fl(x) \circ fl(y)) = (fl(x) \circ fl(y))(1+\epsilon),$$

waarbij $\circ$ staat voor één van de elemtaire berekeningen $+, -, /, *$ en $|\epsilon| \leq \eta$.

Hoewel de relatieve afrondfout na iedere individuele bewerking wel binnen de perken blijft, kunnen fouten zich toch nog ophopen. Omdat iedere elementaire berekening een afrondfout met zich meebrengt, verwachten we dat de totale afrondfout na n elementaire berekeningen in het ergste geval op z'n minst van $\mathcal{O}(n)$ is. Het enige wat we kunnen doen is de volgorde van de berekeningen zo te kiezen dat de constante klein blijft.
De volgende stellingen komen goed van pas wanneer we de invloed van afrondfouten willen analyseren.

Stelling 2.4. ***Ophopen van afrondfouten:*** *Voor gegeven $\{\epsilon_i\}_{i=1}^n$ met $|\epsilon_i| \leq \eta$ en $n\eta < 1$, $\exists \theta_n$ met $|\theta_n| \leq \frac{n\eta}{1-n\eta}$ zdd*

$$\prod_{i=1}^{n}(1+\epsilon_i) = (1+\theta_n).$$

Voorbeeld 2.3. ***Ophopen van afrondfouten.*** *Wanneer we bijvoorbeeld 2 optellingen achter elkaar doen*

$$x = x_0 + x_1 + x_2,$$

waarbij x_i allemaal zwevendekommagetallen zijn, dan is de benadering daarvan

$$fl(x) = fl\left(fl(x_0+x_1)+x_2\right) = \left((x_0+x_1)(1+\epsilon_0)+x_2\right)(1+\epsilon_1).$$

De relatieve fout kunnen we vervolgens afschatten als

$$\frac{|fl(x)-x|}{|x|} \leq \frac{|x_0|+|x_1|}{|x_0+x_1+x_2|}\gamma_2 + \frac{|x_2|}{|x_0+x_1+x_2|}\gamma_1,$$

waarbij $\gamma_n = \frac{n\eta}{1-n\eta}$. Hoewel dit een zeer grove afschatting is, zijn er wel getallen te verzinnen waarvoor de bovengrens bereikt wordt. Aangezien $\gamma_2 > \gamma_1$ willen we graag $|x_0|+|x_1|$ zo klein mogelijk hebben. De volgorde van de optelling heeft dus invloed op de totale afrondfout! Tot slot zien we dat deze berekening niet voorwaarts stabiel *is; wanneer $x_0+x_1+x_2 \approx 0$, dan kan de relatieve fout zeer groot zijn. De berekening is wel* terugwaarts stabiel *omdat de benadering de exacte optelling is van licht verstoorde invoer $x_0(1+\theta_2), x_1(1+\theta_2'), x_2(1+\theta_1)$.*

Voorbeeld 2.4. ***Het tensorproduct:*** *Gegeven twee vectoren $u, v \in \mathbb{R}^n$, geef het tensorproduct $Y = u \otimes v$ een matrix met elementen $y_{ij} = u_i v_j$. De relatieve (afrond)fout is*

$$\frac{y_{ij} - fl(y_{ij})}{|y_{ij}|} = \epsilon_{ij}, \quad |\epsilon_{ij}| \leq \eta.$$

Deze berekening is wel voorwaarts stabiel *omdat de voorwaartse fout klein is. De berekening is echter niet* terugwaarts stabiel*. Het tensorproduct geeft namelijk een matrix met rang 1 (alle kolommen zijn immers een geschaalde versie van u). De benadering na afrondfouten, $\widetilde{Y}$, zal echter in het algemeen geen matrix van rang 1 zijn omdat de afrondfouten ϵ_{ij} onafhankelijk zijn en* niet *te schrijven zijn als $\epsilon_{ij} = \epsilon_i \epsilon_j$.*

Voorbeeld 2.5. ***Cancellatiefout:*** *We willen de kwadratische vergelijking $ax^2 + bx + c$ met $a = 1, b = 200, c = -0.000015$ oplossen. De oplossingen zijn $x_0 = -200.0000000000000075\ldots$ en $x_1 = 0.0000000000000075\ldots$. We zien dat een simpele numerieke berekening wel een goede benadering van x_0 geeft, maar een erg slechte van x_1. Dat komt omdat we bij het berekenen ervan twee bijna even grote getallen van elkaar af hebben getrokken.*

```
x_0 =  1.4210854715202003717 4e-14
x_1 = -2.0000000000000000000e+02
```

Voorbeeld 2.6. ***Verlies van significantie.*** *Wanneer we een klein getal optellen bij een groot getal, $z = x + y$ met $|y| \ll |x|$, dan verliezen we nauwkeurigheid. Wanneer we bijvoorbeeld $c = (a + b) - a$ met $a = 1 \cdot 10^{10}$ en $b = 1.234 \cdot 10^{-5}$ uitrekenen dan krijgen we*

```
c =  1.1444091796875000000 0e-05
```

Voorbeeld 2.7. ***Speciale waarden.*** *Het resultaat van een berekening is te groot of te klein om weer te geven als zwevendekommagetal of de berekening is niet gedefinieerd.*

```
1/1e-320 =  inf
1e-324   =  0.0
inf/inf  =  nan
```

2.3 Opdrachten

Opdrachten met een 🖳-icoon zijn (deels) computeropdrachten. Vaak kun je code uit het hoofdstuk hergebruiken. Kijk daarvoor in het betreffende Jypter Notebook.

🖳 **Opdracht 2.1.** ***Representatiefout:*** *We hebben niet alleen last van afrondfouten met berekeningen; veel cijfers hebben ook geen exacte representatie als zwevendekommagetal. Onderzoek of relatieve fout in de volgende berekeningen binnen de theoretische grenzen ligt. Kun je verklaren waarom de fout soms veel kleiner is?*

1. `0.1 + 0.2`
2. `1.0 - 0.5`
3. `1.0/0.4`
4. `0.5 / 3.0`

Hint: zoek uit met welke precisie het Python type `float` werkt. Je kunt zelf bepalen hoeveel cijfers achter de komma worden weergegeven met `%precision`, zoals in onderstaand voorbeeld.

```
%precision 20
1.0 + 2.0
3.00000000000000000000
```

🖳 **Opdracht 2.2.** ***Nulpunten.*** *Bedenk een manier om de nulpunten van de kwadratische vergelijking $ax^2 + bx + c$ met $a = 1, b = 200, c = -0.000015$ zo nauwkeurig mogelijk te berekenen. Probeer het ook uit en vergelijk met het echte antwoord.*

🖳 **Opdracht 2.3.** ***π:*** *We kunnen π benaderen door de eenheidscirkel op te delen in steeds kleinere veelhoeken. We krijgen dan*

$$\pi \approx 2^{i+1}\sqrt{t_i},$$

met

$$t_{i+1} = 2 - \sqrt{4 - t_i}, \quad t_0 = 2.$$

1. *Probeer π op deze manier te benaderen tot 16 cijfers achter de komma; lukt dat? Waarom wel of niet?*
2. *Kun je een alternatieve recursie bedenken waarvoor je een betere benadering krijgt?*

Opdracht 2.4. ***Sommen.*** *We willen een rij getallen bij elkaar optellen:*

$$S_n = \sum_{k=1}^{n} a_k.$$

We bekijken hier twee algoritmen om de som te berekenen, `lineair` *en* `paarsgewijs`*:*

```
def lineair(a):
    som = a[0]
    n = len(a)
    for k in range(1,n):
        som = som + a[k]
    return som

def paarsgewijs(a):
    n = len(a)
    if n == 1:
        return a[0]
    else:
        m = int(n / 2)
        return paarsgewijs(a[0:m]) + paarsgewijs(a[m:])
```

1. *Laat zien dat de totale afrondfout is begrensd door*

$$|S_n - \widetilde{S}_n| \leq C_n \sum_{k=1}^{n} |a_k|,$$

met $C_n = \left(\frac{\eta(n-1)}{1-\eta(n-1)}\right)$ *voor* `lineair` *en* $C_n = \left(\frac{\eta \log_2(n)}{1-\eta \log_2 n}\right)$ *voor* `paarsgewijs`*. Je mag hierbij aannemen dat de* a_k*'s zwevendekommagetallen zijn.*

2. *Test beide algoritmen op de reeks* $a_k = k^3$ *, in dat geval hebben we* $S_n = (n(n+1)/2)^2$ *als exacte antwoord.*

3. *Verzin een reeks die zeer moeilijk nauwkeurig op is te tellen met de genoemde algoritmen*

Opdracht 2.5. ***Evalueren van functies.*** *Afrondfouten spelen ook een rol wanneer we functies als* $\sqrt{\cdot}$ *en* $\tan(\cdot)$ *willen evalueren. We willen weten in hoeverre een afrondfout in* x *het resultaat* $f(x)$ *beinvloedt. We kijken dus naar*

$$|f(x+\epsilon) - f(x)|/|f(x)|,$$

voor kleine ϵ*. Bestudeer de fout voor* $f(x) = \sqrt{x}$ *met* $x = 1$ *en* $f(x) = \tan(x)$ *voor* $x = 1.5$*. Wat valt je op? Verklaar het verschil.*

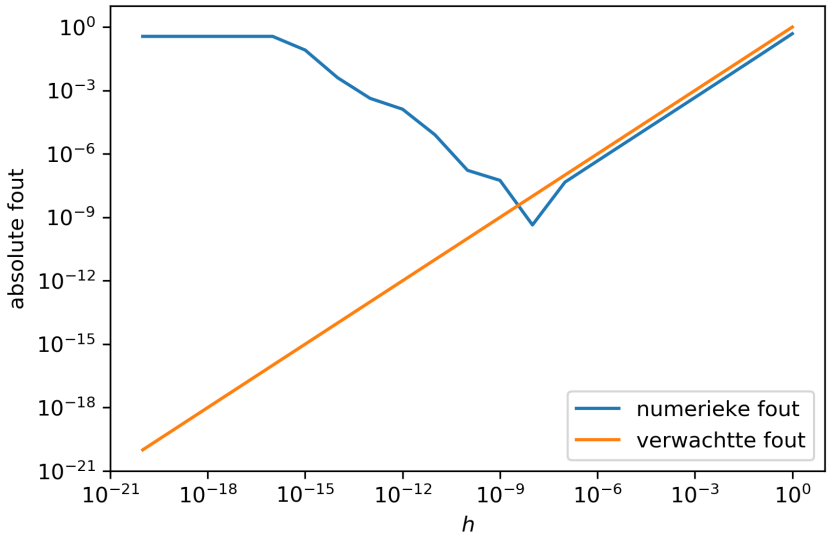
Opdracht 2.6. ***Polynomen.*** *We bekijken hier twee equivalente uitdrukkingen voor hetzelfde polynoom, namelijk*

$$f(x) = (x-1)(x-2)\ldots(x-20),$$

en

$$f(x) = x^{20} - 210 \cdot x^{19} + 20615 \cdot x^{18} - 1256850 \cdot x^{17} + 53327946 \cdot x^{16} - 1672280820 \cdot x^{15} + 40171771630 \cdot x^{14}$$
$$-756111184500 \cdot x^{13} + 11310276995381 \cdot x^{12} - 135585182899530 \cdot x^{11} + 1307535010540395 \cdot x^{10} - 10142299865511450 \cdot x^{9}$$
$$+63030812099294896 \cdot x^{8} - 311333643161390640 \cdot x^{7} + 1206647803780373360 \cdot x^{6} - 3599979517947607200 \cdot x^{5}$$
$$+8037811822645051776 \cdot x^{4} - 12870931245150988800 \cdot x^{3} + 13803759753640704000 \cdot x^{2} - 8752948036761600000 \cdot x$$
$$+2432902008176640000.$$

Maak een grafiek van beide functies en bestudeer het gedrag rond de nulpunten. Kun je dit gedrag verklaren?



Figuur 2.2: Fout bij het benaderen van de afgeleide van $\sin(x)$ in het punt $x = 1.2$

Opdracht 2.7. ***Eindige differentie.*** *We kunnen de afgeleiden van een functie f op het punt x als volgt benaderen*

$$f'(x) \approx \widetilde{g}_h(x) = \frac{f(x+h) - f(x)}{h}.$$

We verwachten dat de benadering beter wordt naarmate h kleiner wordt. In figuur 2.2 zien we een voorbeeld voor $f(x) = \sin(x)$ en $x = 1.3$. We zien dat de fout voor niet al te kleine h van $\mathcal{O}(h)$ is maar dat de fout daarna weer groeit en vervolgens constant is.

1. *Laat zien dat voor een vaste x de fout $|f'(x) - \widetilde{g}_h(x)|$ is begrensd door*

$$|f'(x) - \widetilde{g}_h(x)| \leq C_1 h + C_2 h^{-1},$$

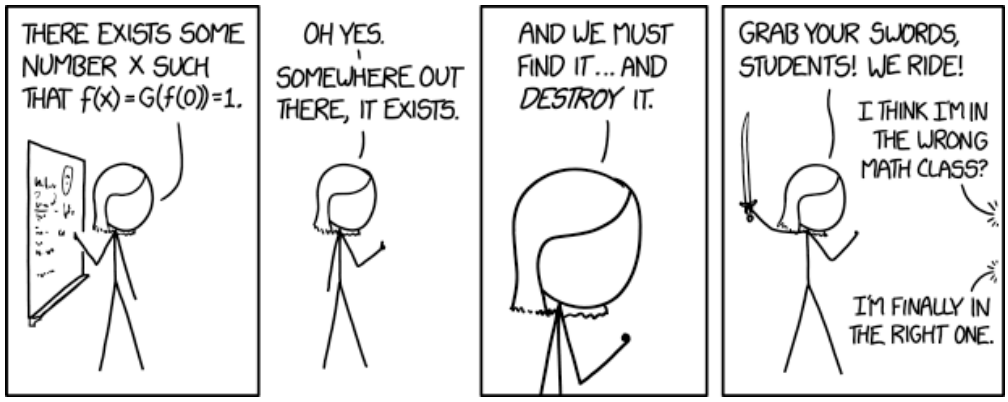
   en geef uitdrukkingen voor C_1, C_2. Je mag hierbij aannemen dat de relatieve afrondfout in het evalueren van f is begrensd door $\mathcal{E}$:

$$fl(f(x)) = f(x)(1+\epsilon), \quad |\epsilon| \leq \mathcal{E}.$$

2. *Gebruik deze bovengrens om het gedrag van de fout in de grafiek te verklaren.*

Hoofdstuk 3

Scalaire vergelijkingen



In dit hoofdstuk gaan we methoden bespreken om niet-lineaire vergelijkingen in één variabele op te lossen. Een toepassing hiervan is bijvoorbeeld het benaderen van $\sqrt{2}$ door het nulpunt te vinden van de functie $f(x) = x^2 - 2$ op het interval $[0, 2]$. Het centrale vraagstuk is

Gegeven een continue functie $f : \mathbb{R} \to \mathbb{R}$ en een interval $[a, b]$, vind (een benadering van) een nulpunt $x_* \in [a, b]$ waarvoor $f(x_*) = 0$.

Bij het bespreken van methoden om een nulpunt te vinden houden we steeds in ons achterhoofd dat we slechts in eindige precisie kunnen rekenen en dat we de functie f wellicht alleen maar kunnen bestuderen door er een x in te voeren en te zien wat eruit komt. De functie f kan bijvoorbeeld geen uitdrukking hebben in termen van machten van x of bekende algebraische of transcedente functies zoals $\sqrt{x}$ of $\sin x$. In zo'n geval kunnen we ervan uitgaan dat er wel een algoritme is om de waarde van de functie voor een gegeven x uit te rekenen. We willen in dit geval dus ook het aantal aanroepen van dit algoritme beperken in onze zoektocht naar een nulpunt, de berekening van f voor een gegeven x zou immers wel eens lang kunnen duren.
In de komende secties gaan we een aantal methoden bespreken om een nulpunt van een functie op een interval $[a, b]$ te vinden. We gaan er in het vervolg van uit dat er zich precies één nulpunt in dit interval bevindt. In de praktijk kunnen we zo'n interval bijvoorbeeld afbakenen door een grafiek van de functie te tekenen. Het algoritme wordt dan ingezet om een initiële schatting van een nulpunt te verfijnen.

3.1 Existentie, uniciteit en gevoeligheid

Het eerste wat we ons afvragen voor een gegeven functie f is of deze wel een nulpunt heeft op het interval $[a, b]$. Als de functie van teken wisselt op het interval $[a, b]$ dan heeft de functie tenminste één nulpunt op het interval. Een nulpunt is uniek op het interval $[a, b]$ als de afgeleide van f er niet van teken wisselt. Deze voorwaarden zijn *voldoende*, maar niet *noodzakelijk*. Denk bijvoorbeeld aan $f(x) = x^2$.
Tot slot bekijken we de gevoeligheid van het probleem. Stel dat we een x_δ vinden waarvoor $f(x_\delta) = \delta$, hoe groot is dan de fout $|x_\delta - x_*|$? Dit is relevant wanneer er bijvoorbeeld geen x is waarvoor f precies gelijk is aan nul omdat we alleen met zwevendekommagetallen rekenen (denk bijvoorbeeld aan $3x - 1$). Ook kan het zijn dat we een x vinden waarvoor onze berekening van f welliswaar precies 0 oplevert, maar dat door afrondfouten de *echte* waarde van f daar niet nul is. Om deze vraag te beantwoorden kijken we naar een Taylorbenadering van f rond het echte nulpunt x_0

$$f(x_\delta) = f'(\xi)(x_\delta - x_*),$$

met $\xi \in [x_*, x_\delta]$. Hieruit kunnen we afleiden dat

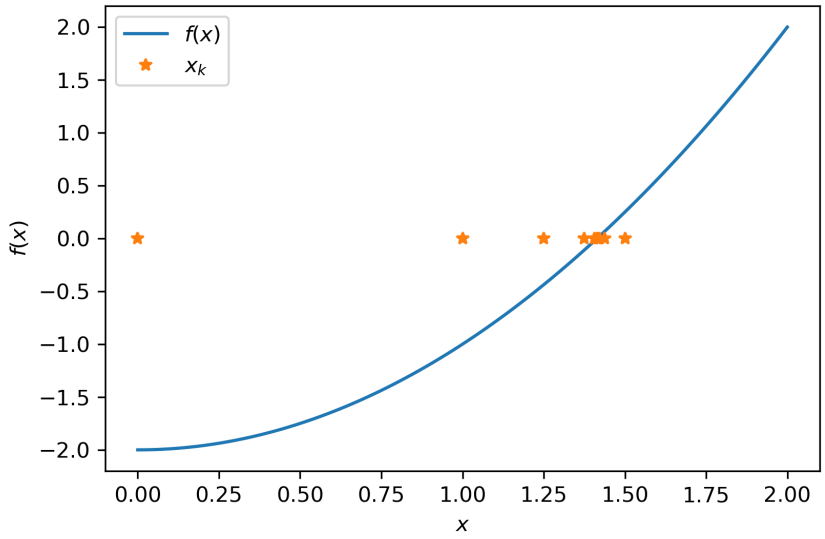
$$|x_\delta - x_*| = |\delta||f'(\xi)|^{-1},$$

en dus dat de fout in het gevonden nulpunt willekeurig groot kan zijn als de afgeleide van f klein is rond het nulpunt. We zeggen in zo'n geval dat het probleem *slecht gesteld* is. We kunnen niet verwachten dat we een robuust algoritme gaan vinden om een slecht gesteld op te lossen.

3.2 Numerieke methoden

3.2.1 De bisectiemethode

In de bisectiemethode gaan we er van uit dat de functie één keer van teken wisselt op het interval $[a, b]$ zo dat $f(a)f(b) < 0$. We kunnen nu als schatting van het nulpunt simpelweg het middelpunt van het interval nemen, $m = (a + b)/2$. Als toevalling $f(m) = 0$ dan zijn we klaar, en anders weten we dat de fout met echte nulpunt kleiner is dan $(b - a)/2$. Om een beter schatting te krijgen kunnen we kijken op welke helft van het interval het nulpunt zich bevindt door naar het teken dan $f(m)$ te kijken; als $f(a)f(m) < 0$ dan



Figuur 3.1: Bisectiemethode toegepast op de vergelijking $x^2 = 2$

gaan we verder op het interval $[a, m]$ en anders gaan we door op het interval $(m, b]$. Door op deze manier het interval telkens in tweeen te delen, halveren we de fout ook in iedere stap.
Een voordeel van de methode is dat we zeer weinig van de functie vragen; we willen alleen het teken weten. Daardoor is deze methode zeer robuust wanneer er afrondfouten worden gemaakt. Een nadeel is dat de fout slechts halveert in iedere stap en we dus 4 extra stappen moeten doen om de fout met een factor 10 te verkleinen. Een groter nadeel wellicht is dat deze methode niet werkt als de functie een dubbel nulpunt heeft, zoals bijvoorbeeld $f(x) = x^2$.
Er zijn verschillende implementaties mogelijk van de bisectiemethode. Hieronder een voorbeeld:

```
def bisectie(f,x0,n):
    x = np.zeros(n)
    x[0] = x0[0]
    d = (x0[1]-x0[0])/2
    for k in range(n-1):
        x[k+1] = x[k] + d
        d = d*np.sign(f(x[k+1])*f(x[k]))/2

    return x
```

Voorbeeld 3.1. ***De bisectiemethode in actie:*** *Laten we de bisectiemethode eens toepassen op de vergelijking $x^2 = 2$. We zien de successieve benaderingen x_k in figuur 3.1.*

3.2.2 Vastepuntiteratie

De vastepuntiteratie is niet zozeer een methode als wel een klasse van methoden. Er komt direct een aantal belangrijke concepten aan bod die we later ook nog nodig gaan hebben. Het idee achter de vastepuntiteratie is dat we de vergelijking $f(x) = 0$ gaan schrijven als $g(x) = x$ en dat we een vast punt van g vinden door middel van de iteratie

$$x_{k+1} = g(x_k),$$

beginnend vanaf een gegeven x_0. Als deze iteratie convergeert en we vinden een $x_k = x_{k-1}$ dan is x_k een vast punt van g. Dan resten ons twee belangrijke vragen: *i)* hoe construeren we een functie g voor een gegeven f? en *ii)* onder welke voorwaarden convergeert de vastepuntiteratie naar een vast punt van g?.

Het is makkelijk om een g te vinden die een vast punt heeft op het nulpunt van f, neem bijvoorbeeld $g(x) = x + f(x)$. Dit is niet de enige, we kunnen ook $g(x) = x - 5f(x)$ nemen. Voor specifieke f zijn er vaak nog meer mogelijkheden. Voor $f(x) = x^2 - x$ ligt $g(x) = x^2$ voor de hand. Welke keuze voor g geschikt is komen we te weten door de vastpuntiteratie te bestuderen.
Allereerst zouden we graag willen dat wanneer f maar één nulpunt heeft op het interval $[a, b]$, dat g dan ook maar één vast punt heeft. Daarnaast willen we weten onder welke voorwaarden de vastepuntiteratie naar zo'n vast punt convergeert.

Stelling 3.1. ***Vastepuntstelling:*** *Een functie $g \in C[a, b]$ heeft teminste één vast punt op het interval $[a, b]$ wanneer $a \leq g(x) \leq b$ voor $x \in [a, b]$. Als bovendien $|g'(x)| < 1$ voor $x \in [a, b]$ dan is er precies één vast punt.*

Stelling 3.2. ***Convergentie van de vastepuntiteratie:*** *De vastepuntiteratie $x_{k+1} = g(x_k)$ convergeert naar een vast punt, x_*, van g (dat wil zeggen dat $\lim_{k\to\infty} x_k = x_*$) voor alle $x_0 \in [a, b]$ als $|g'(x)| < 1$ voor alle $x \in [a, b]$.*

Uit het bewijs van de voorgaande stelling volgt ook dat de fout $e_k = |x_k - x_*|$ in iedere iteratie *minstens* met een factor $\rho < 1$ afneemt:

$$e_k \leq \rho e_{k-1} \leq \rho^k e_0.$$

We noemen dit *lineaire convergentie*. We kunnen hieruit ook een schatting maken van het aantal iteraties dat we nodig hebben om een absolute fout van ϵ te behalen. We hebben $e_k \leq \rho^k e_0$. Om $|e_k| \leq \epsilon$ te bereiken hebben we dus $k \geq \rho^{-1} \log |e_0|/\epsilon$ iteraties nodig. Formeel gezien definieren we lineaire convergentie als volgt:

Definitie 3.1. ***Lineaire convergentie:*** *Een reeks $\{x_k\}_{k=0}^{\infty}$ convergeert lineair naar x_* als*

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|} = \rho.$$

We kunnen de vastepuntiteratie gemakkelijk implementeren in Python:

```
def vastepuntiteratie(g,x0,n):
    x = np.zeros(n)
    x[0] = x0
    for k in range(n-1):
        x[k+1] = g(x[k])

    return x
```

Voorbeeld 3.2. ***De vastepuntiteratie in actie:*** *We passen de vastepuntiteratie toe op twee functies die hetzelfde vaste punt $x_* = 1/2$ hebben $g_1(x) = (x + 1/2)/2$ en $g_2(x) = 1/2 - (x - 1/2)^3$. In figuur 3.2 zien we een grafische weergaven van de vastepuntiteratie en de convertentie van de fout $|x_k - x_*|$. We zien dat de iteratie voor g_2 veel sneller convergeert dan de verwachte lineaire convergentie.*

In de voorgaande voorbeelden zagen we dat de afschatting in de praktijk wat pessimistisch kan zijn; de fout daalde voor g_2 veel sneller dan de verwachte lineaire convergentie! Dit valt te verklaren met behulp van de volgende stelling:

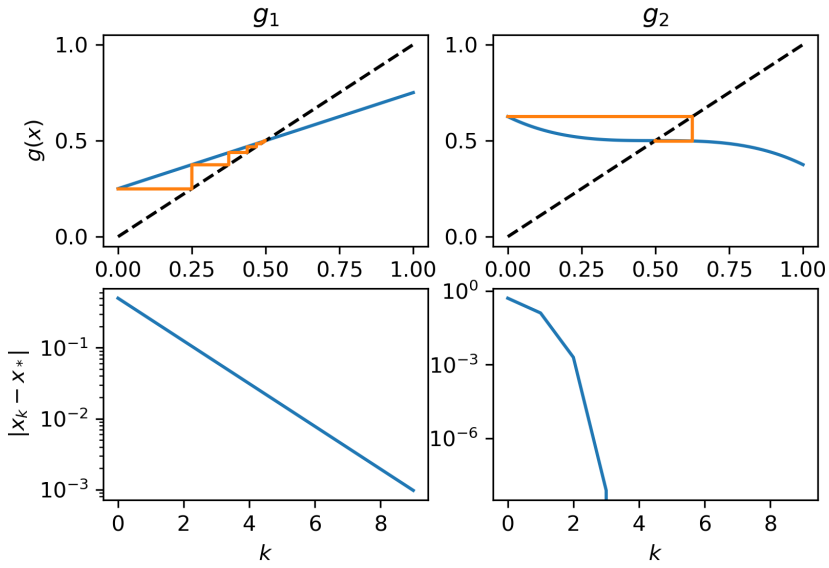
Stelling 3.3. ***Kwadratische convergentie:*** *Als de vastepuntiteratie $x_{k+1} = g(x_k)$ convergeert en bovendien $g'(x_*) = 0$ dan convergeert de iteratie* kwadratisch*, d.w.z.*

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|^2} = M.$$

waarbij $M = \sup_{\xi} |g''(\xi)|$.

Dan kunnen we ons nu afvragen hoe we een functie g vinden die voldoet aan de volgende eisen

- een vast punt van g is een nulpunt van f, d.w.z. $g(x_*) = x_* \iff f(x_*) = 0$
- $|g'(x)| < 1$ voor $x \in [x_* - \delta, x_* + \delta]$ met $\delta > 0$ (liefst zo groot mogelijk).
- indien mogelijk: $|g'(x_*)| = 0$.



Figuur 3.2: De Vastepuntniteratie voor twee verschillende functies. Boven zien we een grafische weergave van de vastepuntiteratie. De onderste grafieken laten de fout $|x_k - x_*|$ zien

3.2.3 Newtons methode

We kunnen Newtons methode zien als een vastepuntiteratie waarbij

$$g(x) = f(x) - \frac{f(x)}{f'(x)}.$$

We kunnen gemakkelijk verifiëren dat g inderdaad een vast punt heeft op een nulpunt van f en dat $g'(x_*) = 0$ (als $f'(x_*) \neq 0$). Newtons methode heeft ook een inzichtelijke interpretatie. We kunnen de functie f rond het punt x_0 benaderen als

$$f(x) \approx \ell(x; x_0) = f(x_0) + (x - x_0)f'(x_0)$$

Het nulpunt van $\ell(x; x_0)$ is $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$. Newtons method is zeer geschikt om snel een hoge nauwkeurigheid te behalen omdat de methode (onder bepaalde voorwaarden) kwadratisch convergeert:

Stelling 3.4. ***Convergentie van Newtons methode:*** *Er is een $\delta > 0$ zodat Newtons methode convergeert voor $x_0 \in [x_* - \delta, x_* + \delta]$. De convergentie is op zijn minst lineair en als bovendien $f'(x_*) \neq 0$ dan convergeert Newtons methode kwadratisch.*

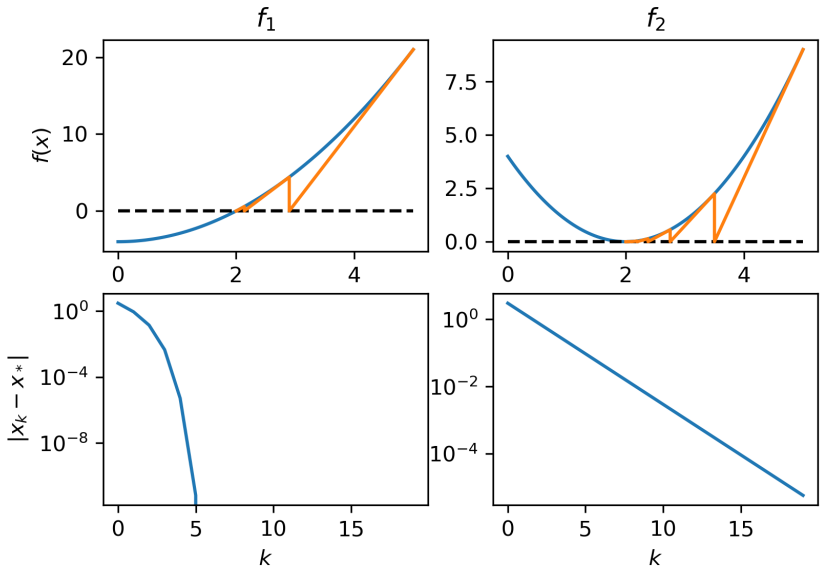
We kunnen Newtons methode bijvoorbeeld als volgt implementeren:

```
def newton(f,x0,n):
    x = np.zeros(n)
    x[0] = x0

    for k in range(n-1):
        x[k+1] = x[k] - f[0](x[k])/f[1](x[k])

    return x
```

Voorbeeld 3.3. ***Newton in actie:*** *We passen Newtons methode toe op twee functies met hetzelfde nulpunt $f_1(x) = x^2 - 4$ en $f_2(x) = (x-2)^2$. In figuur 3.3 zien we convergentie van de fout. Het valt op dat Newtons methode niet*



Figuur 3.3: Convergentie van Newtons methode voor twee verschillende functies.

altijd kwadratisch converteert. Dit is te verklaren uit het feit dat f_2 een dubbel *nulpunt heeft ($f'(x_*) = 0$). We zouden verwachten dat we dan in de problemen raken omdat we in de iteratie door nul delen. Echter, in de limiet naar het vaste punt toe zal f harder dalen dan f' zodat de iteratie toch convergeert, zij het slechts lineair.*

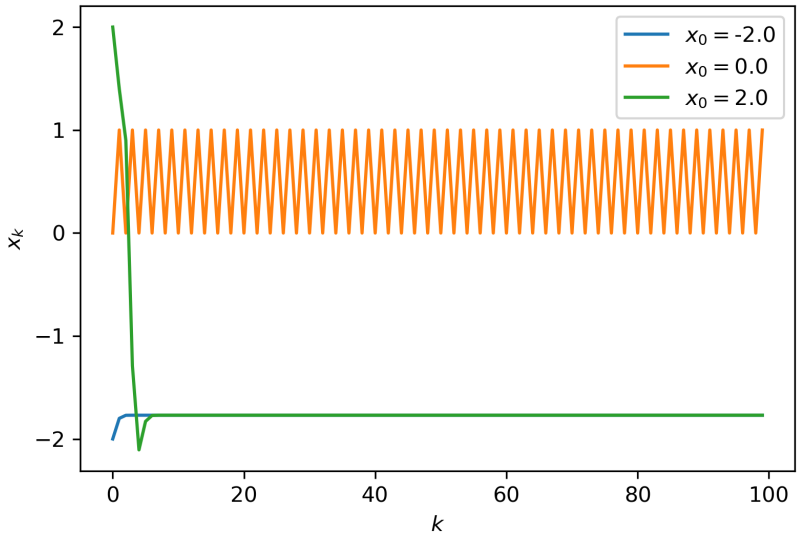
Voorbeeld 3.4. ***Heen en weer:*** *Zoals de stelling doet vermoeden convergeert Newtons iteratie alleen wanneer we in de buurt van een nulpunt beginnen. Maar wat betekent dat in de praktijk? En wat gebeurt er als we niet in de buurt van een nulpunt beginnen? In figuur 3.4 zien we de iteraties voor verschillende x_0 voor $f(x) = x^3 - 2x + 2$. We zien dat er naast convergentie naar het nulpunt $x_* = -1.76\ldots$ ook een oscillatie tussen 0 en 1 mogelijk is.*

Voorbeeld 3.5. ***Een complex voorbeeld:*** *We kunnen Newtons methode ook gebruiken om een* complex *nulpunt te vinden. Neem bijvoorbeeld de vergelijking $z^p + 1 = 0$. Wanneer we Newtons methode hierop toepassen krijgen we de volgende vastepuntiteratie:*

$$z_{k+1} = z_k - \frac{z_k^p + 1}{p z_k^{p-1}}.$$

Beginnen we bijvoorbeeld op $x_0 = 1 + 0.1\imath$ dan convergeert de methode (voor $p = 2$) binnen 10 iteraties naar de oplossing $x_* = \imath$.

```
0 (0.00495049504950506+0.09950495049504951j)
1 (-0.24690131107874452+5.062221303177699j)
2 (-0.11864470543503591+2.6296471190295643j)
3 (-0.050761042172708604+1.504576869843514j)
4 (-0.014181560304378317+1.0842299522278152j)
5 (-0.0010599523907402243+1.0031928797682195j)
6 (-3.3687486575046006e-06+1.0000045246152327j)
7 (-1.5242207158022237e-11+1.000000000004562j)
8 (-6.953487024273889e-23+1j)
9 1j
```



Figuur 3.4: Iteraties vanaf verschillende x_0 voor de functie $f(x) = x^3 - 2x + 2$

3.2.4 De secantmethode

Om Newtons methode te gebruiken hebben we extra informatie over de functie nodig, namelijk de afgeleide. In het geval dat deze niet beschikbaar is, moeten we iets anders verzinnen. Kunnen we afgeleide niet benaderen? Een elegante manier om informatie die we toch al hebben uitgerekend te hergebruiken is om de afgeleide te benaderen met een raaklijn

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

We krijgen dan de secantiteratie

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}.$$

In Python kunnen we deze als volgt implementeren:

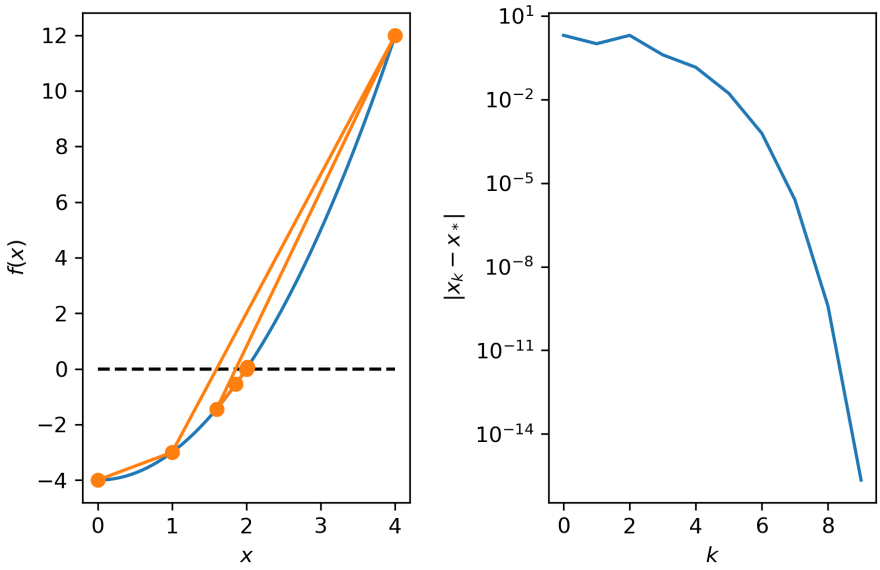
```
def secant(f,x0,n):
    x = np.zeros(n)
    x[0] = x0[0]
    x[1] = x0[1]

    for k in range(1,n-1):
        x[k+1] = x[k] - f(x[k])*(x[k] - x[k-1])/(f(x[k]) - f(x[k-1]))
    return x
```

Voorbeeld 3.6. ***Secant in de praktijk:*** *Wanneer we de secantmethode toepassen op de functie $f(x) = x^2 - 2$, zien we in figuur 3.5 dat de convergentie sneller is dan lineair, maar wellicht wat langzamer dan kwadratisch. Dit lijkt ook wel logisch, we gebruiken immers een benadering van de afgeleide. De benadering wordt wel steeds beter naarmate we dichter bij de oplossing komen.*

3.2.5 Stopcriteria

Tot nu toe hebben we alleen gekeken naar convergentie in de limiet $k \to \infty$. In de praktijk willen we natuurlijk zo min mogelijk iteraties doen om een vereiste nauwkeurigheid te behalen. We willen dus stoppen



Figuur 3.5: Convergentie van de secantiteratie voor $f(x) = x^2 - 2$

zodra $|x_k - x_*| \leq \epsilon$.
We kunnen de waarde van f in de gaten houden en stoppen wanneer deze klein is. We hadden echter al gezien dat dit niet garandeert dat de fout $|x_k - x_*|$ klein is waneer $|f(x_k)|$ klein is als het probleem slecht gesteld is. Niettemin kunnen we proberen de fout te schatten met behulp van een Taylorexpansie. We krijgen dan

$$|x_k - x_*| = f(x_k)/f'(\xi_k),$$

met $\xi_k \in [x_*, x_k]$. Mochten we een globale boven- en ondergrens voor de afgeleide van f hebben, dan kunnen we die gebruiken om een onder- en bovengrens voor de fout te vinden. Hebben we dat niet, dan kunnen we $f'(\xi_k)$ bijvoorbeeld benaderen met $f'(x_k)$.
We kunnen ook de iteraties x_k zelf in de gaten houden. We kunnen de afstand tussen opeenvolgende iteraties als volgt uitdrukken

$$|x_{k+1} - x_k| = |(x_{k+1} - x_*) - (x_k - x_*)| \leq |x_{k+1} - x_*| + |x_k - x_*|.$$

Als de methode convergeert, dan wordt de fout in iedere iteratie kleiner en kunnen we deze afschatten als $|x_{k+1} - x_k| < 2|x_k - x_*|$. Het enige wat we hieruit kunnen bepalen is dat we minstens moeten doorgaan tot $|x_{k+1} - x_k| < 2\epsilon$ maar zelfs dan kan de fout natuurlijk nog veel groter zijn. Toch is niet een onredelijk criterium, wanneer de fout bijvoorbeeld iedere stap van teken wisselt weten we bijvoorbeeld dat $x_* \in [x_k, x_{k+1}]$. In dat geval hebben we dus $|x_* - x_k| \leq |x_{k+1} - x_k|$.
In het volgende voorbeeld zien we hoe goed deze benaderingen het in de praktijk doen.

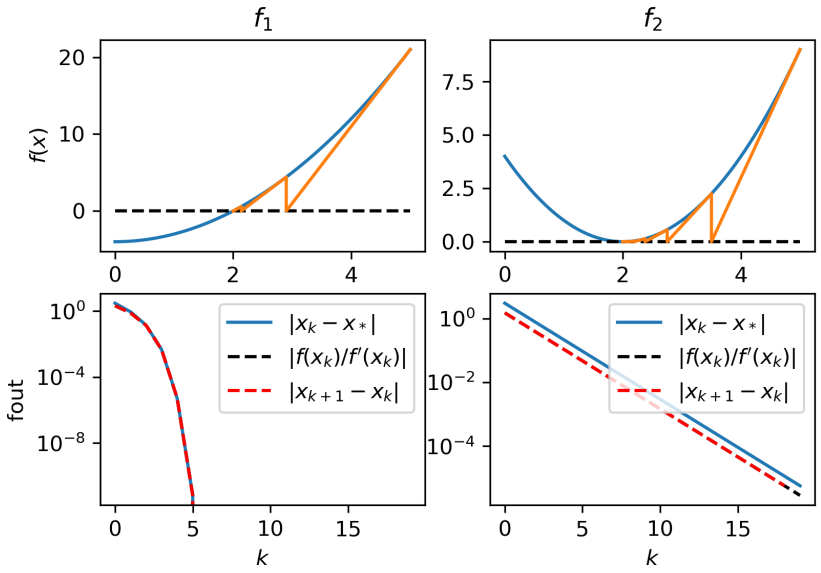
Voorbeeld 3.7. ***Het schatten van de fout:*** *We zien hier hoe twee schattingen van de fout een goed beeld kunnen geven van de echte fout in de praktijk:*

$$|x_k - x_*| \approx f(x_k)/f'(x_k),$$

en

$$|x_k - x_*| \approx |x_{k+1} - x_k|.$$

Het resultaat is te zien in figuur 3.6.



Figuur 3.6: Twee schattingen van de fout

3.2.6 Meerdere nulpunten*

Het vinden van meerdere nulpunten is in het algemeen niet eenvoudig. De enige manier lijkt om eerst doormiddel van een grafiek grofweg te bepalen waar de nulpunten zich bevinden en vervolgens de beginschattingen te verfijnen doormiddel van een paar Newtoniteraties. De moeilijkheid hiervan is dat we niet goed weten hoe dicht we bij de nulpunten moeten beginnen om ervoor te zorgen dat de iteratie er ook naar toe convergeert. Zoals uit het volgende voorbeeld blijkt kunnen de aantrekkingsdomeinen van de nulpunten er erg ingewikkeld uitzien.

Voorbeeld 3.8. ***Een nog complexer voorbeeld:*** *We hebben eerder al gezien dat we Newtons methode ook in het complexe vlak kunnen gebruiken. Ook hier kunnen we ons afvragen wat het aantrekkingsdomein is van de verschillende nulpunten.*
We kunnen dit gemakkelijk numeriek bekijken door de volgende functie te bekijken:

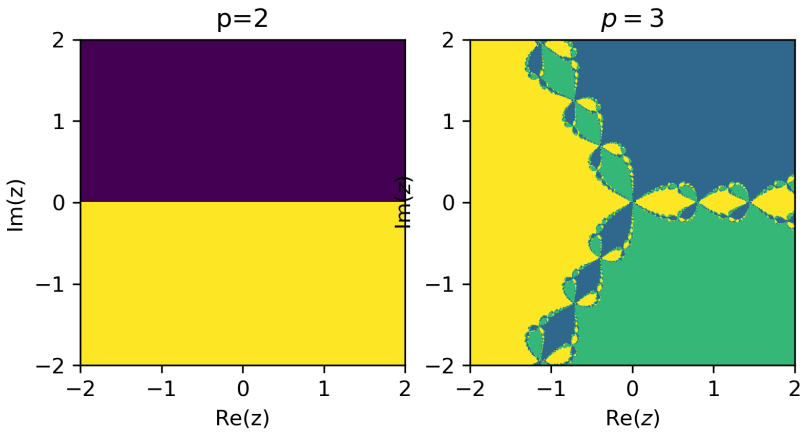
$$G_k(x_0) = \underbrace{g \circ g \circ \ldots \circ g}_{k \text{ keer}}(z_0).$$

Het aantrekkingsgebied van een nulpunt z_* is dan gegeven door alle z_0 waarvoor $\lim_{k\to\infty} G_k(z_0) = z_*$. Voor een numerieke benadering nemen we natuurlijk k groot. Verder kunnen we voor dit voorbeeld ieder nulpunt identificeren door naar het complexe argument te kijken. Zo krijgen we een functie van $\mathbb{C} \to \mathbb{R}$ waar we een grafiek van kunnen maken.
Voor $p = 2$ zien in figuur 3.7 we weining verrassends; als we in het linkerdeel van complexe vlak beginnen dan convergeren we naar $x_* = -\imath$ en wanneer we in het rechterdeel beginnen naar $x_* = +\imath$. Voor $p = 3$ is de sitatie erg verrassend! Dit is een voorbeeld van een zogenaamde Newton fractal.

Voor *polynomen* zijn er meer manieren beschikbaar om alle nulpunten te vinden. Voor polynomen van graad groter dan 4 zijn er in het algemeen geen algebraïsche uitdrukkingen beschikbaar voor de oplossingen en zullen we dus bijvoorbeeld Newtons methode moeten gebruiken. Een elegante manier om Newtons methode in te zetten om de nulpunten van een polynoom te vinden is *Horners methode*:

Voorbeeld 3.9. ***Horners methode:*** *We willen alle nulpunten van het polynoom*

$$f_0(x) = x^3 - 6x^2 + 11x - 6$$



Figuur 3.7: Aantrekkingsdomein van complexe nulpunten van $z^p + 1$ voor $p = 2$ en $p = 3$.

vinden. Als eerste stap gebruiken we Newtons methode om een nulpunt te vinden, zeg x_0. Vervolgens delen we dit nulpunt uit het polynoom en definiëren

$$f_1(x) = f_0(x)/(x - x_0).$$

We kunnen deze deling efficiënt doen door middel van een soort staartdeling *(`numpy.polydiv` in Python). Vervolgens zoeken we een nulpunt van f_1 etc. Op deze manier vinden we $x_0 = 3$, $x_1 = 2$ en $x_2 = 1$. In figuur 3.8 zien we de resulterende polynomen.*

Een tweede manier om alle nulpunten van een polynoom te benaderen is door het vraagstuk te vertalen naar lineaire algebre. Bij een polynoom $p(x) = x^n + c_{n-1}x^{n-1} + \ldots + c_0$ hoort een begeleidende matrix P waarvan de eigenwaarden precies de nulpunten van p zijn. Als we de eigenwaarden numeriek kunnen benaderen hebben we daarmee ook de nulpunten van het polynoom benaderd.

Voorbeeld 3.10. ***Begeleidende matrix:*** *Het polynoom $p(x) = (x-1)(x+1) = x^2 - 1$ heeft de volgende begeleidende matrix:*

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

De eigenwaarden van deze matrix zijn inderdaad $1, -1$ zoals beloofd.

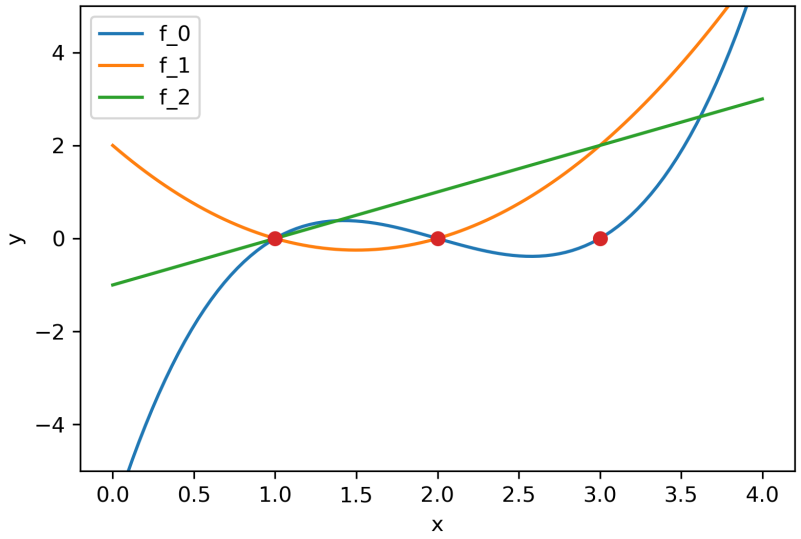
3.3 Opdrachten

Opdracht 3.1. ***Vastepuntiteratie:*** *Een eenvoudige methode om een nulpunt van een functie op het interval $[a, b]$ te vinden is de volgende vastepuntiteratie*

$$x_{k+1} = x_k - \alpha f(x_k).$$

1. *Laat zien dat de methode convergeert voor $\alpha \in (0, 2/M]$ wanneer $0 < f'(x) < M$ voor $x \in [a, b]$.*
2. *Kun je ook iets zeggen over de optimale keuze voor α?*
3. *Test de methode door middel van numerieke experimenten.*

Opdracht 3.2. ***Invloed van afrondfouten:*** *Onderzoek de invloed van afrondfouten op de vastepuntiteratie $x_{k+1} = g(x_k)$; in plaats van $g(x_k)$ je rekent met $\widetilde{g}(x_k) = g(x_k) + \epsilon_k$. Je mag aannamen dat $|g'(x)| \leq \rho < 1$ en dat $|\epsilon_k| \leq \mathcal{E}$.*



Figuur 3.8: Opeenvolgende polynomen van Horner's methode toegepast op $f_0(x) = x^3 - 6x^2 + 11x - 6$

1. *Convergeert de methode nog steeds? Zo ja, waarnaartoe?*
2. *Bedenk een functie g waarbij de afrondfouten een hele grote invloed hebben en een waarbij de afrondfouten een kleine invloed hebben.*
3. *Test dit ook numeriek door in de vastepuntiteratie telkens een (klein) willekeurig getal op te tellen (gebruik de module* `random`*)*

Opdracht 3.3. ***Kubische convergentie:*** *De methode van Halley vindt een nulpunt door middel van de volgende iteratie*

$$x_{k+1} = x_k - \frac{2f(x_k)f'(x_k)}{2(f'(x_k))^2 - f(x_k)f''(x_k)}.$$

1. *Laat zien dat deze methode* kubisch *convergeert naar een* enkelvoudig nulpunt *van f (d.w.z. dat $f^{(k)}(x_*) \neq 0$) indien we daar in de buurt van beginnen*
2. *Onderzoek door middel van een numeriek experiment wat er gebeurt er voor m-*voudige nulpunten *(d.w.z. dat $f^{(k)}(x_*) = 0$ voor $k < m$)*

Opdracht 3.4. ***Meervoudige nulpunten:*** *Laat zien dat de aangepaste versie van Newtons methode*

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)}$$

kwadratisch convergeert naar een $m-$voudig nulpunt van f. Een nadeel van deze methode is dat we vantevoren moeten weten wat m is. Wat gebeurt er wanneer we per ongeluk m te groot of te klein kiezen?

Opdracht 3.5. ***Alle nulpunten vinden:*** *Vind de nulpunten van de volgende functies tot 6 cijfers achter de komma nauwkeurig:*

1. $x^2 - 5x + 3$
2. $x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512.$

Je mag zelf weten welke methode je gebruikt. Probeer zo min mogelijk iteraties te doen om de gewenste nauwkeurigheid te bereiken; kies dus een goed stopcriterium.

Opdracht 3.6. ***Superlineaire convergentie:*** *De reeks $\{x_k\}$ convergeert superlineair naar x_* wanneer:*

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|} = \rho_k,$$

waarbij $\rho_k \to 0$ als $k \to \infty$.

1. *Laat zien dat de fout $e_k = x_k - x_*$ voor de secantmethode kan worden geschreven als*

$$e_{k+1} = \left(1 - \frac{f'(\xi_k)}{f'(\eta_k)}\right) e_k,$$

   met $\xi_k \in [x_*, x_k]$ en $\eta_k \in [x_{k-1}, x_k]$.

Om netjes te bewijzen dat dit de methode convergeert is niet zo gemakkelijk (probeer maar eens). We zien wel dat als *de methode convergeert, dat de factor ρ_k dan steeds kleiner wordt en we dus* superlineaire *convergentie hebben.*

2. *Onderzoek numeriek hoe snel de methode convergeert. Je kunt bijvoorbeeld kijken of je een macht $\alpha > 1$ kan vinden zodat de fout zicht gedraagt als $|e_{k+1}| = C \cdot |e_k|^\alpha$.*

Hoofdstuk 4

Stelsels vergelijkingen

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

In dit hoofdstuk kijken we naar *stelsels* van m vergelijkingen in n onbekenden. Dit kunnen *lineaire* of *niet-lineaire* vergelijkingen zijn. Vragen die aan bod komen zijn

1. Heeft het stelsel een oplossing?
2. Is deze oplossing uniek?
3. Hoe gevoelig is de oplossing voor verstoringen?
4. Hoe kunnen de oplossing efficiënt numeriek benaderen?

Eerst bespreken we verschillende methoden om een stelsel met een inverteerbare matrix op te lossen. Daarna bespreken we (kort) hoe we ook voor niet-inverteerbare matrices een oplossing kunnen bepalen door middel van de *pseudo-inverse*. Voor niet-lineaire vergelijkingen kijken we naar generalisaties van Netwon's methode.

4.1 Stelsels lineaire vergelijkingen

Wanneer we n lineaire vergelijkingen in n onbekenden hebben kunnen we deze weergeven in matrixvorm

$$Ax = b.$$

Voorlopig gaan we er van uit dat $\det(A) \neq 0$, ofwel dat het stelsel een unieke oplossing heeft. In dat geval bestaat er een inverse van A waarvoor $A^{-1}A = AA^{-1} = I$. In dat geval is $x = A^{-1}b$ de oplossing is van het stelsel vergelijkingen.
Om te weten of een stelsel vergelijkingen *goed gesteld* is willen we tot slot weten hoe gevoelig de oplossing is voor verstoringen. We kijken hiervoor naar de oplossingen van twee stelsels $Ax = b$ en $Ax' = b'$. We kunnen het verschil van deze oplossingen als volgt afschatten

$$\|x - x'\| = \|A^{-1}(b - b')\| \leq \|A^{-1}\|\|b - b'\|.$$

Daarnaast weten we dat $\|b\| = \|Ax\| \leq \|A\|\|x\|$, zodat we de *relatieve* verstoring kunnen afschatten als

$$\frac{\|x - x'\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|b - b'\|}{\|b\|}.$$

De factor $\|A\|\|A^{-1}\|$ wordt het *conditiegetal* van A genoemd en aangeduid met $\kappa(A)$.
We hebben geleerd om zo'n stelsel op te lossen door middel van een Gauss-Jordan eliminatie. Hier vegen we eerst van boven naar beneden om een bovendriehoeksstelsel te krijgen. Vervolgens vegen we van beneden naar boven om een diagonaal stelsel te krijgen. Een voorbeeld zien we hieronder.

Voorbeeld 4.1. ***Gauss-Jordan eliminatie:*** *We beginnen met het stelsel*

$$\begin{pmatrix} 1 & 3 & 1 \\ 1 & 1 & -1 \\ 3 & 11 & 6 \end{pmatrix} x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

Naar beneden vegen geeft

$$\begin{pmatrix} 1 & 3 & 1 \\ 0 & -2 & -2 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Vervolgens weer naar boven vegen geeft

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 4.5 \\ 3 \\ 1 \end{pmatrix},$$

waaruit we de oplossing af kunnen lezen: $x = (4.5, -1.5, 1)$.

In Python kunnen we deze stappen als volgt implementeren:

```
def GaussJordanHeen(A,b):
    n = len(b)
    L = np.zeros((n,n))
    for k in range(n-1):
        for i in range(k+1,n):
            L[i,k] = A[i,k]/A[k,k]
            for j in range(k,n):
                A[i,j] = A[i,j] - L[i,k]*A[k,j]
            b[i] = b[i] - L[i,k]*b[k]

def GaussJordanTerug(A,b):
    n = len(b)
    U = np.zeros((n,n))
    for k in range(n-1,0,-1):
        for i in range(k-1,-1,-1):
            U[i,k] = A[i,k]/A[k,k]
            for j in range(n-1,k-1,-1):
                A[i,j] = A[i,j] - U[i,k]*A[k,j]
            b[i] = b[i] - U[i,k]*b[k]
```

4.1.1 LU-decompositie

Zoals we in het vorige voorbeeld zagen, kunnen we de eliminatieprocedure opdelen in 2 stappen; *i)* Veeg van boven naar beneden om een bovendriehoeksstelsel $Ux = \widetilde{b}$ te krijgen, *ii)* Los dit stelsel op door middel van terugwaartse substitutie.
We kunnen deze stappen uitdrukken als een serie matrix-matrix vermenigvuldigingen $U = M_{n-1} \dots M_2 M_1 A$ en $\widetilde{b} = M_{n-1} \dots M_2 M_1 b$, waarbij M_k een *Gauss-transformatie* is. Zo'n transformatie zet alle elementen van een matrix onder de diagonaal in een bepaalde kolom op nul, bijvoorbeeld:

$$M_1 = \begin{pmatrix} 1 & & & \\ -\frac{a_{21}}{a_{11}} & 1 & & \\ -\frac{a_{31}}{a_{11}} & 0 & 1 & \\ \vdots & & & \ddots \end{pmatrix}.$$

Deze matrices blijken inverteerbaar te zijn:

$$M_1^{-1} = \begin{pmatrix} 1 & & & \\ \frac{a_{21}}{a_{11}} & 1 & & \\ \frac{a_{31}}{a_{11}} & 0 & 1 & \\ \vdots & & & \ddots \end{pmatrix}.$$

Op soortgelijke wijze kunnen we $M_2, M_3, \dots$ defini{ë}ren. Merk wel op dat M_2 niet direct de elementen van A bevat, maar die van $A_1 = M_1 A$. Zo kunnen we M_k pas defini{ë}ren nadat we M_{k-1} hebben toegepast.
We kunnen A nu schrijven als

$$A = M_1^{-1} \dots M_{n-2}^{-1} M_{n-1}^{-1} U.$$

Dit geeft een decompositie van de matrix A in een benedendriehoeksmatrix $L = M_1^{-1} \dots M_{n-2}^{-1} M_{n-1}^{-1}$ en een bovendriehoeksmatrix U:

Stelling 4.1. ***LU-decompositie:*** *Een matrix $A \in \mathbb{R}^{n \times n}$ met inverteerbare submatrices $(a_{ij})_{i \geq k, j \geq k}$ kan worden geschreven als het product van een benedendriehoeksmatrix L en een bovendriehoeksmatrix U: $A = LU$. Wanneer we eisen dat $l_{ii} = 1$ dan is deze decompositie uniek.*

In de praktijk willen we liever niet alle matrices M_k expliciet genereren en opslaan. Een mogelijk implementatie is *Doolittle's* methode. Op stap k rekenen we de elementen van L en U als volgt uit:

$$u_{kj} = a_{kj} - \sum_{i=1}^{k-1} l_{ki}u_{ij}, \quad j = k, \ldots, n,$$

$$l_{ik} = \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij}u_{jk} \right) / u_{kk}, \quad i = k+1, \ldots, m.$$

Merk op dat we l_{kk} niet aanpassen omdat we $l_{kk} = 1$ aannemen.
Een andere implementatie is gebaseerd op het recursief toepassen van de volgende decompositie

$$\begin{pmatrix} a & b^T \\ c & D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ a^{-1}c & I \end{pmatrix} \begin{pmatrix} a & b^T \\ 0 & D - a^{-1}cb^T \end{pmatrix},$$

waarbij $a \in \mathbb{R}$, $b, c \in \mathbb{R}^{n-1}$ en $D \in \mathbb{R}^{(n-1)\times(n-1)}$. We noemen het element a het *pivotelement*, omdat we met deze de bijbehorende kolom op nul gaan zetten.
We kunnen de vector $a^{-1}c$ handig opslaan in de eerste kolom van de matrix omdat deze na vegen toch alleen maar nullen bevat. Na één stap krijgen we dus

$$\begin{pmatrix} a & b^T \\ a^{-1}c & D - a^{-1}cb^T \end{pmatrix}.$$

Nu passen dit proces weer toe op de submatrix $A_{22} = D - a^{-1}cb^T$. Na $n-1$ stappen vinden de elementen van de matrix U in het bovendriehoeksdeel van de matrix en die van de matrix L onder de diagonaal.
In Python kunnen we dit bijvoorbeeld als volgt implementeren:

```
def LUDecompositie(A):
    n = A.shape[0]
    B = np.array(A)
    for k in range(n-1):
        B[k+1:,k] = B[k+1:,k]/B[k,k]
        B[k+1:,k+1:] = B[k+1:,k+1:] - np.outer(B[k+1:,k],B[k,k+1:])
        print(B)
    return B
```

Voorbeeld 4.2. *Gegeven de matrix*

$$A = \begin{pmatrix} 1 & 2 & 5 \\ 2 & 3 & 2 \\ 5 & 1 & 1 \end{pmatrix},$$

krijgen we de volgende tussenstappen:

$$\begin{pmatrix} 1 & 2 & 5 \\ 2 & -1 & -9 \\ 5 & -9 & -24 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 5 \\ 2 & -1 & -8 \\ 5 & 9 & 48 \end{pmatrix}.$$

Hieruit lezen de volgende decompositie af

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 5 & 9 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & 2 & 5 \\ 0 & -1 & -8 \\ 0 & 0 & 48 \end{pmatrix}.$$

Voorbeeld 4.3. *We kunnen nu het voorbeeld uit de introductie nog eens bekijken met*

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}.$$

We krijgen dan

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 8 \end{pmatrix}.$$

Wat opvalt is dat de elementen van U hier exponentieel groeien.

Voorbeeld 4.4. *Een ander bijzonder geval zien we wanneer we de volgende matrix ontbinden:*

$$A = \begin{pmatrix} 0.1 & 2.001 \\ 10 & 200 \end{pmatrix}.$$

We krijgen dan

$$L = \begin{pmatrix} 1 & 0 \\ 100 & 1 \end{pmatrix}, U = \begin{pmatrix} 0.1 & 2.001 \\ 0 & -0.1 \end{pmatrix}.$$

Hier zien we juist dat de elementen in L sterk kunnen groeien wanneer de pivotelementen klein zijn.

Nu kunnen we ons afvragen of we op deze manier ook gegarandeerd een goede oplossing krijgen wanneer we in eindige precisie rekenen. In hoofdstuk 1 zagen we al dat Gauss eliminatie niet altijd goed werkt voor grote matrices. We kunnen dit verklaren met behulp van de volgende stellingen:

Stelling 4.2. ***Stabiliteit van de LU-decompositie:*** *In eindige precisie krijgen we (met Doolittle's algoritme) een LU-decompositie van een verstoorde matrix:*

$$\widetilde{L}\widetilde{U} = \widetilde{A},$$

met absolute elementsgewijze fout $|\widetilde{A} - A| \leq \gamma_n |\widetilde{L}||\widetilde{U}|$ *met* $\gamma_n = n\eta/(1 - n\eta)$*. Hier is* $|G|$ *een matrix met elementen* $|g_{ij}|$*.*
Omdat de elementen in $\widetilde{U}$ *exponentieel kunnen groeien is de LU-decompositie niet terugwaarts stabiel.*

De fout $|\widetilde{A} - A|$ wordt dus beinvloed door de grootte van de elementen van $\widetilde{L}$ en $\widetilde{U}$. Zoals we hebben gezien groeien de elementen van U in het ergste geval exponentieel in n. Daarnaast kunnen de elementen in L erg groot worden wanneer de diagonaal elementen die we gebruiken om te vegen erg klein worden. Wanneer we een stelsel vergelijkingen oplossen met behulp van een LU-decompositie krijgen we te maken met deze fout. Hoe groot is dan de uiteindelijke fout in de oplossing? We kijken hiervoor naar $Ax = b$ en het verstoorde stelsel $(A + E)\widetilde{x} = b$ en vinden

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \kappa(A) \frac{\|E\|\|\widetilde{x}\|}{\|\widetilde{b}\|}.$$

Een slecht geconditioneerde matrix kan dus zelfs kleine foutjes in de LU-decompositie versterken.
Zoals we hebben gezien in de vorige stelling, is de LU-decompositie niet terugwaarts stabiel. Met name liepen we tegen de volgende problemen aan:

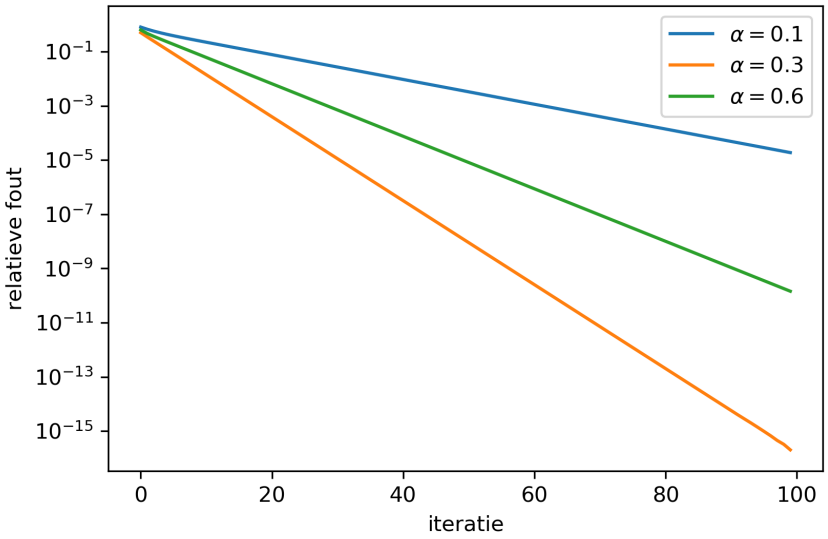
- de afrondfout wordt versterkt wanneer we een klein element op de diagonaal tegenkomen. In het ergste geval hebben we $a = 0$ en kunnen we helemaal niet verder vegen.
- de elementen van U groeien (in het ergste geval) exponentieel, wat tot problemen leidt wanneer we met eindige precisie rekenen.

Om deze problemen te voorkomen kijken we naar twee aangepaste versies van de LU-decompositie.

Stelling 4.3. ***LU-decompositie met gedeeltelijke pivoting:*** *Voor een gegeven matrix* $A \in \mathbb{R}^{n \times n}$ *krijgen we*

$$PA = LU,$$

waarbij P een permutatiematrix *is. Deze wordt zo gekozen dat in iedere stap het grootste pivotelement naar boven wordt gehaald om mee te vegen. Op die manier garanderen we dat* $|l_{ij}| \leq 1$*.*



Figuur 4.1: Richardsoniteratie voor verschillende stapgroottes.

Stelling 4.4. ***LU-decompositie met volledige pivoting:*** *Voor een gegeven matrix $A \in \mathbb{R}^{n\times n}$ krijgen we*

$$PAQ = LU,$$

waarbij P en Q permutatiematrices *zijn. De matrix Q wordt zo gekozen dat exponentiele groei van de elementen van U wordt voorkomen.*

Volledige pivoting is lastig te implementeren; het vinden van de optimale kolompermutatie is een NP-compleet probleem! In de praktijk worden vaak heuristische algoritmen gebruikt om een goede kolompermutatie te vinden.

4.1.2 Stationaire iteratieve methoden

We kunnen een stelsel vergelijkingen ook oplossen door middel van een vastepuntiteratie. We zoeken dan een functie $g : \mathbb{R}^n \to \mathbb{R}^n$ waarvan de oplossing van het stelsel een vast punt is. De eenvoudigste is wellicht de *Richardsoniteratie:*

$$x_{k+1} = (I - A)x_k + b.$$

Stelling 4.5. ***Richardsoniteratie:*** *De vastepuntiteratie*

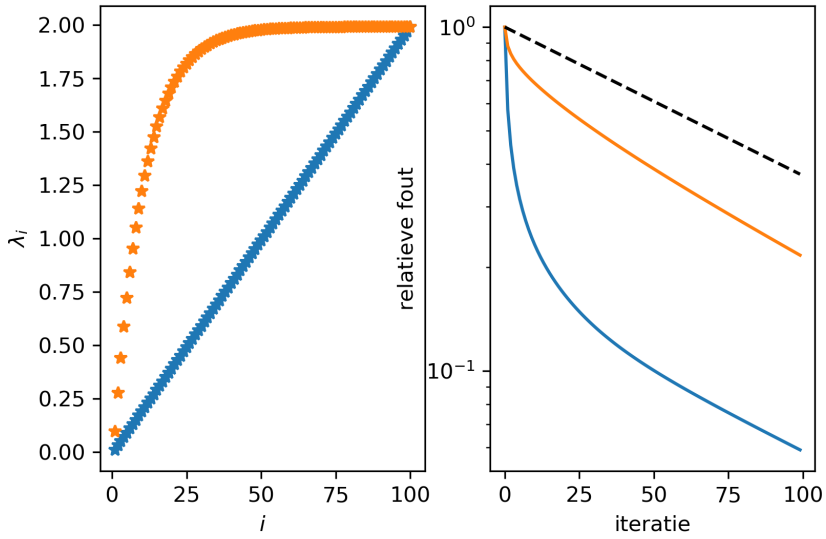
$$x_{k+1} = (I - A)x_k + b,$$

convergeert naar de oplossing van het stelsel $Ax = b$ als $\rho(I - A) < 1$. Merk op dat $\rho(G) \leq \|G\|$ voor een willekeurige matrix G zodat we in de praktijk ook de eis $\|I - A\| < 1$ kunnen gebruiken als voldoende voorwaarde.

Voorbeeld 4.5. ***Stapgrootte:*** *Als een gegeven matrix alleen positieve eigenwaarden heeft kunnen we de Richardsoniteratie altijd laten convergeren door een $\alpha > 0$ te kiezen in de geschaalde iteratie*

$$x_{k+1} = (I - \alpha A)x_k + \alpha b.$$

In figuur 4.1 zien we dat keus van α nogal een verschil kan maken.



Figuur 4.2: Convergentie van de Richardsoniteratie voor twee verschillende matrices. Links het spectrum van de matrices en rechts de relatieve fout.

Voorbeeld 4.6. ***Onregelmatige convergentie:*** *De stapgrootte is niet de enige factor die invloed heeft op de convergentie. In dit voorbeeld bekijken we de convergentie voor 2 verschillende matrices. De spectra en de relatieve fouten zijn te zien in figuur 4.2. De fout daalt in het begin veel sneller wanneer de eigenwaarden uniform verdeeld zijn. Na een aantal iteraties dalen beide fouten even snel.*

Om beter te begrijpen hoe deze eenvoudige iteratieve methode werkt, beginnen we met $x_0 = 0$ en schrijven we de iteratie als

$$x_k = \left(\sum_{i=0}^{k-1} (I - A)^i \right) b.$$

De uitdrukking bevat positieve machten van A, en kan dus worden gezien als een polynoom, $p_k(A)$. Wanneer de matrix diagonaliseerbaar is ($S^{-1}AS = \Lambda$) dan kunnen we dit polynoom uitdrukken als

$$p(A) = S^{-1} p_k(\Lambda) S,$$

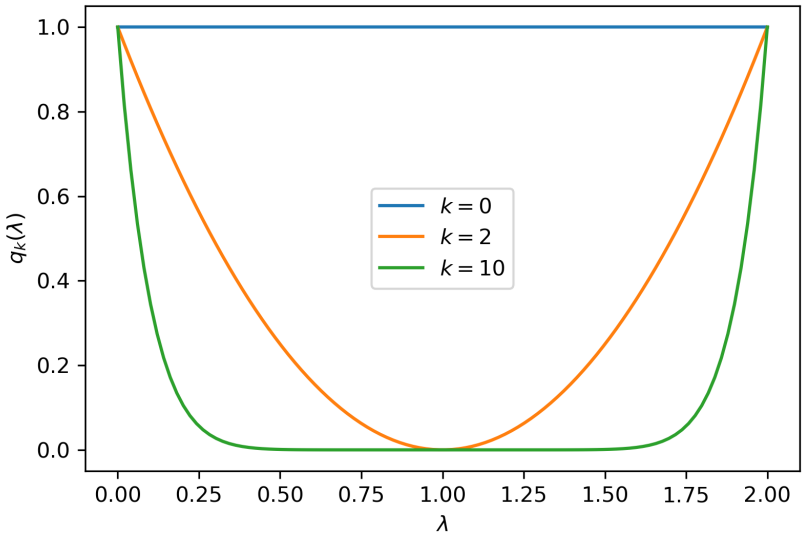
waarbij $p(\Lambda)$ een diagonaalmatrix is met elementen $p_k(\lambda_i)$. Aangezien de oplossing is gegeven door $x_* = A^{-1}b$ kunnen we ons afvragen in welk opzicht $p_k(\lambda) \approx \lambda^{-1}$. De connectie blijkt een Taylorbenadering te zijn! De Taylorserie van λ^{-1} rond 1 is namelijk gegeven door

$$\lambda^{-1} = \sum_{i=0}^{\infty} (1 - \lambda)^i.$$

Deze som convergeert voor $\lambda \in (0, 2)$.
Op een soortgelijke manier kunnen we naar de fout kijken:

$$e_{k+1} = (I - A)^{k+1} e_k.$$

Het polynoom $q_k(\lambda) = (1 - \lambda)^k$ convergeert naar nul voor $\lambda \in (0, 2)$ zoals te zien in figuur 4.3. Hiermee kunnen we ook het verschil in convergentie uit het voorbeeld verklaren: de foutcomponent in de richting van eigenvectoren met eigenwaarden rond 1 daalt veel sneller dan die met eigenwaarden dichtbij 2.



Figuur 4.3: Het polynoom $(1-\lambda)^k$ voor verschillende k.

4.2 Slecht gestelde stelsels lineaire vergelijkingen*

Voorbeeld 4.7. ***Niet-inverteerbare matrix:*** *De matrix*

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix},$$

is niet inverteerbaar. De matrix heeft wel een orthonormale basis van eigenvectoren dus kunnen we de oplossing schrijven als $x = \alpha v_1 + \beta v_2$. We kunnen het stelsel dan schrijven als

$$\alpha\lambda_1 v_1 = \langle v_1, b\rangle, \quad \beta\lambda_2 v_2 = \langle v_2, b\rangle.$$

Omdat $\lambda_1 = 0$ kunnen we α niet bepalen en dus net zo goed ook op nul zetten. Als $\langle v_1, b\rangle = 0$ dan is het stelsel vergelijkingen consistent en kunnen we alsnog een oplossing vinden. Wanneer $\langle v_1, b\rangle \neq 0$ dan kunnen we géén x vinden waarvoor $Ax = b$.

Voorbeeld 4.8. ***Onderbepaald stelsel:*** *Voor een onderbepaald stelsel kunnen we bovenstaande aanpak niet toepassen; de matrix is immers niet vierkant. We kunnen we op zoek gaan naar een oplossing in de rij-ruimte van de matrix $x = Az$, waarbij z een oplossing is van*

$$AA^T z = b.$$

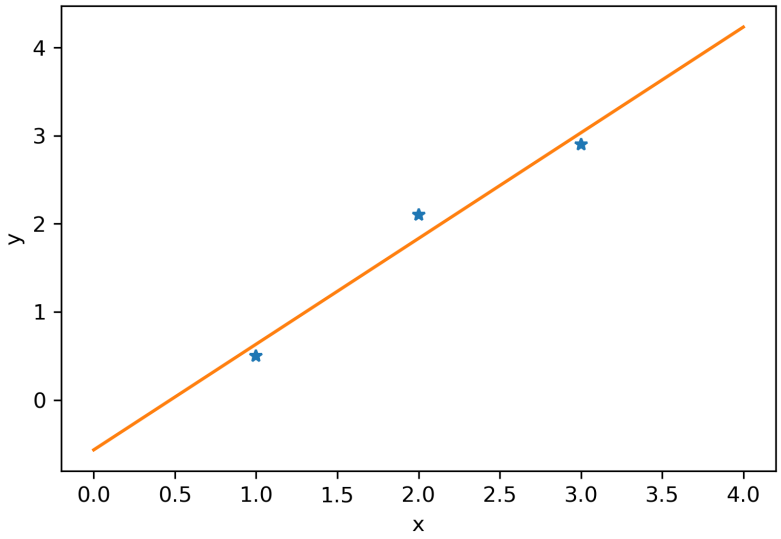
Indien AA^T inverteerbaar is, kunnen we zo een oplossing vinden in de rij-ruimte van A: $x = A^T \left(AA^T\right)^{-1} b$. Dit is natuurlijk niet de enige oplossing; we kunnen er altijd een element uit de nulruimte van A bij optellen. Merk op dat de matrix $A^T(AA^T)^{-1}$ een rechter-inverse *van A is: $AA^T(AA^T)^{-1} = I$.*

Voorbeeld 4.9. ***Overbepaald stelsel:*** *Dit is de omgekeerde situatie, we hebben te veel vergelijkingen. Als deze niet consistent zijn, dan is er niet eens een oplossing. We kunnen dan bijvoorbeeld b beperken tot de kolomruimte van A en het stelsel*

$$A^T Ax = A^T b,$$

oplossen. Als A^TA inverteerbaar is krijgen we zo een oplossing die b zo goed mogelijk benadert in de kolomruimte van A. Merk op dat $(A^TA)^{-1}A^T$ een linker-inverse *van A is: $(A^TA)^{-1}A^TA = I$.*

Wanneer AA^T of A^TA niet inverteerbaar zijn kunnen we die natuurlijk diagonaliseren, en het stelsel projecteren op de deelruimte waar ze wel inverteerbaar zijn. Het blijkt dat de eigenvectoren van AA^T en A^TA veel



Figuur 4.4: Kleinstekwadraten schatting van een lijn door 3 punten

nuttige informatie over de matrix A bevatten en kunnen worden gebruikt om iedere willekeurige matrix te diagonaliseren:

Stelling 4.6. ***Singuliere waardendecompositie:*** *We kunnen een $m \times n$ matrix A diagonaliseren:*

$$A = U\Sigma V^T,$$

waarbij U een $m \times m$ matrix is met de linker singuliere vectoren, V een $n \times n$ matrix is met de rechter singuliere vectoren en Σ een $m \times n$ matrix met de singuliere waarden $\sigma_1 > \sigma_2 > \ldots > \sigma_k > 0$ op de diagonaal. De linker singuliere vectoren zijn de eigenvectoren van AA^T en de rechter singuliere vectoren zijn de eigenvectoren van A^TA. De eigenwaarden van AA^T en A^TA zijn het kwadraat van de singuliere waarden van A.

Met behulp van deze decompositie kunnen we de *pseudo-inverse* van een matrix definiëren:

Stelling 4.7. ***Pseudo-inverse:*** *De pseudo-inverse van een matrix A van rang k is*

$$A^\dagger = V_k \Sigma_k^{-1} U_k^T,$$

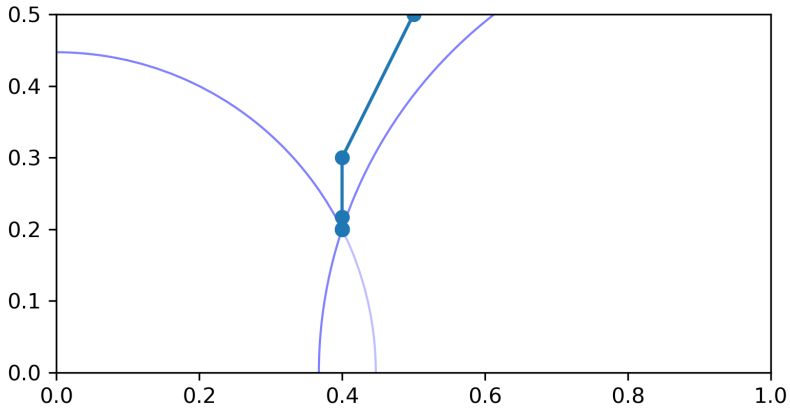
waar bij U_k en V_k de eerste k singuliere vectoren van A bevat en Σ_k een $k \times k$ diagonaalmatrix is met de singuliere waarden $\sigma_1 > \sigma_2 > \ldots > \sigma_k > 0$ op de diagonaal.

Lossen we een stelsel op met de pseudo-inverse dan krijgen we een oplossing met de kleinste norm $\|x\|_2$ en het kleinste residu $\|Ax - b\|_2$.

Voorbeeld 4.10. ***Kleinstekwadratenschatting:*** *Wanneer we een rechte lijn $y = ax + b$ door een set punten (x_i, y_i) willen trekken dan kunnen we een stelsel vergelijkingen opstellen voor de coëfficiënten (a, b). Wanneer de punten niet precies op één lijn liggen zal het stelsel inconsistent zijn en moeten een we een kleinstekwadratenoplossing bepalen. Een voorbeeld is te zien in figuur 4.4.*

4.3 Stelsels niet-lineaire vergelijkingen*

We hebben in het vorige hoofdstuk een aantal methoden gezien om niet-lineaire vergelijkingen in één variabele op te lossen. Een aantal van deze methoden laat zich generaliseren naar meer variabelen.



Figuur 4.5: Newtoniteraties voor het oplossen van een stelsel niet-lineaire vergelijkingen.

4.3.1 Vastepuntiteratie

Als het lukt een om een operator G te vinden waarvan het vaste punt overeenkomt met een oplossing van $F(x) = 0$, dan kunnen we ook hier een vastepuntiteratie gebruiken:

$$x_{k+1} = G(x_k),$$

waarbij G gekozen is zodat $G(x_*) = x_* \Leftrightarrow F(x_*) = 0$.

Stelling 4.8. ***Vastepuntstelling:*** *De vastepuntiteratie*

$$x_{k+1} = G(x_k),$$

convergeert als G een contractie is, dwz $\exists \rho \in [0,1)$ zodanig dat

$$\|G(x) - G(y)\| \leq \rho \|x - y\|.$$

4.3.2 Newtons methode

Een specifieke vastepuntiteratie voor stelsels niet-lineaire vergelijkingen is Newtons methode in meer dimensies:

$$x_{k+1} = x_k + J(x_k)^{-1} F(x_k),$$

waarbij J de Jacobimatrix van F is. Het bewijs dat Newtons methode kwadratisch convergeert is vergelijkbaar met het bewijs in één variabele.

Voorbeeld 4.11. ***GPS:*** *We kunnen onze locatie (in het vlak) bepalen door middel van een driehoekspeiling. Gegeven de afstanden d_0, d_1 tot 2 referentiepunten (x_i, y_i) stellen we 2 vergelijkingen op*

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2, \quad \text{voor} \quad i \in \{0,1\}.$$

Een voorbeeld is te zien in figuur 4.5.

4.4 Opdrachten

Opdracht 4.1. ***Pivoting:*** *Vind de optimale rij- en kolompermutatie voor de volgende matrices:*

1. $$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}.$$

2. $$A = \begin{pmatrix} 0 & 5 & 8 \\ 1 & 2 & 3 \\ 10 & 20 & 5 \end{pmatrix}.$$

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

Je kunt dit met hand doen maar net zo gemakkelijk in Python. Je kunt een $n \times n$ permutatiematrix maken met `P = sparse.coo_matrix((np.ones(n), (np.array(range(n)), I)), shape=(n,n))`*, waarbij* `I` *een array is van lengte n waarin de nieuwe volgorde van de rijen staat aangegeven.*

Opdracht 4.2. ***Voorwaartse substitutie:*** *Om een stelsel met een benedendriekhoeksmatrix op te lossen doen we een voorwaartse eliminatie*

$$x_k = \left(b_k - \sum_{i=1}^{k-1} l_{ki} x_i \right) / l_{kk}, \quad k = 1, \ldots, n.$$

Wanneer we dit in eindige precisie uitrekenen krijgen we een benadering $\widetilde{x}$.
Aangenomen dat l_{ki} en b_k zwevendekommagetallen zijn, laat zien dat

1. *$\widetilde{x}$ voldoet aan $l_{kk}\widetilde{x}_k(1+\theta_k) = b_k - \sum_{i=1}^{k-1} l_{ki}\widetilde{x}_i(1+\theta_i)$, met $|\theta_k| \leq \gamma_k$. (Hint: gebruik de alternatieve vorm van de represenatiefout $fl(x) = x(1+\delta)^{-1}$ met $|\delta| \leq \eta$.*

2. *$\widetilde{x}$ de oplossing is van een verstoord stelsel $(L+E)\widetilde{x} = b$ met $|E| \leq \gamma_n |L|$ (elementsgewijs).*

3. *de voorwaartse fout is begrens door: $\|x - \widehat{x}\| \leq \gamma_n \kappa(L) \|\widehat{x}\|$.*

Opdracht 4.3. ***Richardsoniteratie:*** *We willen een stelsel vergelijkingen $Ax = b$ oplossen met de gedempte Richardsoniteratie:*

$$x_{k+1} = x_k - \alpha(Ax_k - b).$$

Gegeven is dat de matrix A diagonaliseerbaar is en alleen strikt positieve eigenwaarden heeft.

1. *Onder welke voorwaarde op α convergeert de gedempte Richardsoniteratie naar de oplossing van het stelsel vergelijkingen?*

2. *Bepaal de waarde van α waarvoor de methode het snelst convergeert.*

3. *Waar convergeert de iteratie naar toe indien de matrix ook eigenwaarden gelijk aan nul heeft?*

Opdracht 4.4. ***Richardsoniteratie voor defecte matrices:*** *Gebruik een Richardson iteratie om een stelsel op te lossen met de matrix*

$$A = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}.$$

We weten dat de iteratie convergeert als $\rho(I - \alpha A) < 1$, maar in de praktijk is het makkelijk om de eis $\|I - \alpha A\|_1 < 1$ te gebruiken.

1. *Onderzoek door middel van een numeriek experiment of de twee verschillende eisen tot verschillend convergentiegedrag leiden voor* $b = (1, 1)$
2. *Doe hetzelfde voor* $b = (1, 0)$. *Kun je het verschil verklaren?*

Opdracht 4.5. ***Een snellere Richardson iteratie:*** *We beschouwen hier de volgende Richardson iteratie:*

$$x_{k+1} = (I - \alpha_k A)x_k + \alpha_k b,$$

met α_k *gegeven. We nemen aan dat* A *diagonaliseerbaar is met eigenwaarden* $\lambda_0, \ldots, \lambda_{n-1}$.

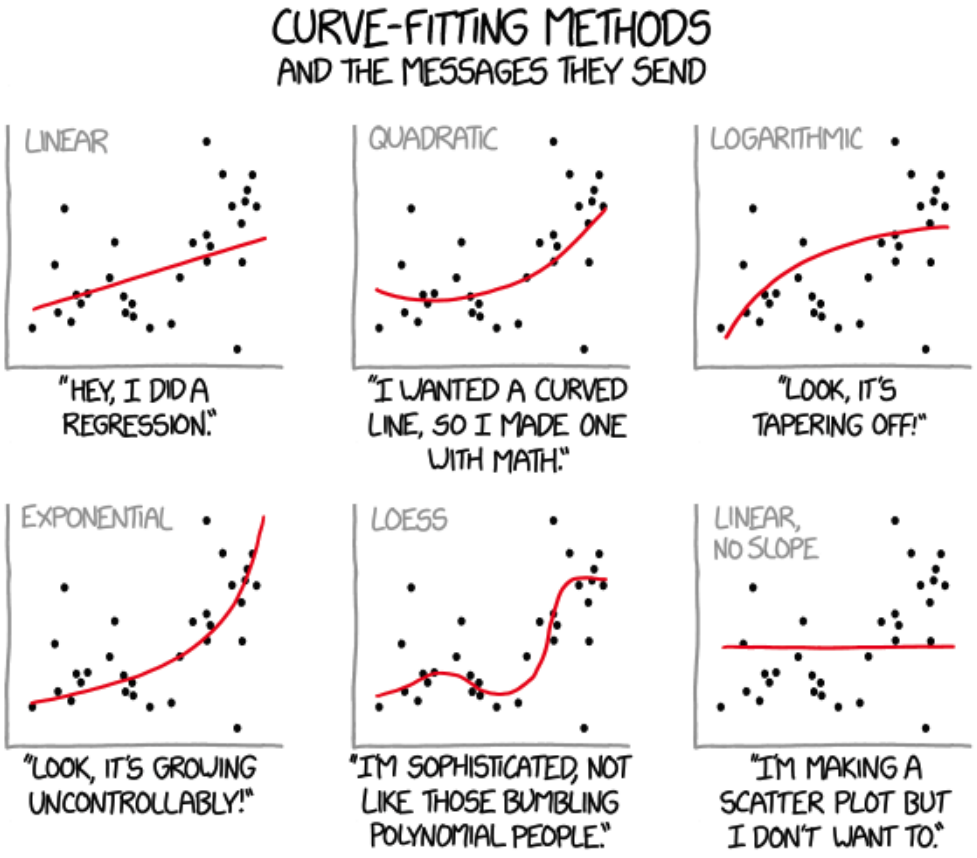
1. *Laat zien dat deze iteratie in exacte aritmetiek in precies* n *stappen het exacte antwoord geeft als we* $\alpha_k = 1/\lambda_k$ *kiezen.*
2. *Probeer de methode uit op een diagonaalmatrix met eigenwaarden* $1, 2, \ldots, n$. *Is dit praktisch gezien ook een aantrekkelijke methode?*

Opdracht 4.6. ***Pseudo-inverse:*** *We willen hier een stelsel vergelijkingen* $Ax = b$ *oplossen met* $A \in \mathbb{R}^{m \times n}$ *met* $m > n$ *of* $m < n$. *In beide gevallen heeft de matrix volledige rang rang*$(A) = \min(m, n)$. *We definieren de oplossing aan de hand van de pseudo-inverse:* $\widehat{x} = A^{\dagger} b$.

1. *Laat zien dat voor* $m > n$ *de verkregen oplossing de unieke oplossing is met het kleinste residue* $\|A\widehat{x} - b\|_2$.
2. *Laat zien dat voor* $m < n$ *de verkregen oplossing de unieke oplossing is met de kleinste norm* $\|\widehat{x}\|_2$.

Hoofdstuk 5

Het benaderen van functies



In dit hoofdstuk gaan we methoden bespreken om functies te benaderen. We gaan er van uit dat we een gegeven (mogelijk zeer ingewikkelde) functie f op een gegeven interval $[a,b]$ willen benaderen met een eenvoudigere functie p. Het kan zijn dat het uitrekenen van f veel rekenkracht vergt en we een eenvoudige benadering nodig hebben om snel te kunnen rekenen. Het kan ook zijn dat f slechts gegeven is op een eindig aantal punten x_i en we willen hieruit een continue functie construeren. Je kunt bijvoorbeeld denken aan metingen van de temperatuur in de Bilt op gegeven tijdstippen maar ook aan de functie $f(x) = x$!
Om dit te doen gaan we de benadering p uitdrukken als lineaire combinatie van n basisfuncties $\{\phi_i\}_{i=1}^n$

$$p(x) = \sum_{i=1}^{n} c_i \phi_i(x).$$

Vervolgens moeten we de coëfficiënten c_i bepalen zodat p een goede benadering is van f. Een manier om dit te doen is te eisen dat $p(x_i) = f(x_i)$ op een aantal gegeven steunpunten $\{x_i\}_{i=1}^m$. We krijgen dan een stelsel van m vergelijkingen in n onbekenden

$$Bc = y,$$

met $b_{ij} = \phi_j(x_i)$.
Als we p gebruiken om f te benaderen voor een punt $x \in [a,b]$ noemen we dat *interpolatie*. Als we p gebruiken om f buiten het interval $[a,b]$ te benaderen noemen we dit *extrapolatie*.
Een andere optie is om een p te zoeken waarvoor de fout $\|f - p\|$ over het gehele interval zo klein mogelijk is. Ook uit deze eis kunnen we een stelsel vergelijkingen opstellen om de coefficienten c_i te vinden.
We zullen ons voornamelijk bezighouden met het benaderen van f met polynomen, maar maken ook kleine uitstapjes naar andere mogelijkheden en multivariate functies.

5.1 Interpolatie met polynomen

We beginnen met het geval dat we precies genoeg punten hebben om een n-de graads polynoom te bepalen:

Gegeven $n+1$ punten $\{x_i\}_{i=0}^n$ en waarden $\{y_i\}_{i=0}^n$, vind een n-de graads polynoom, $p_n(x)$, zodat $p_n(x_i) = y_i$.

Het vinden van een polynoom die door de gegeven punten gaat begint bij het vinden van een geschikte basis voor polynomen. Een voordehandliggende is $\{1, x, x^2, \ldots, x^n\}$. We krijgen dan

$$p_n(x) = \sum_{k=0}^{n} c_k x^k.$$

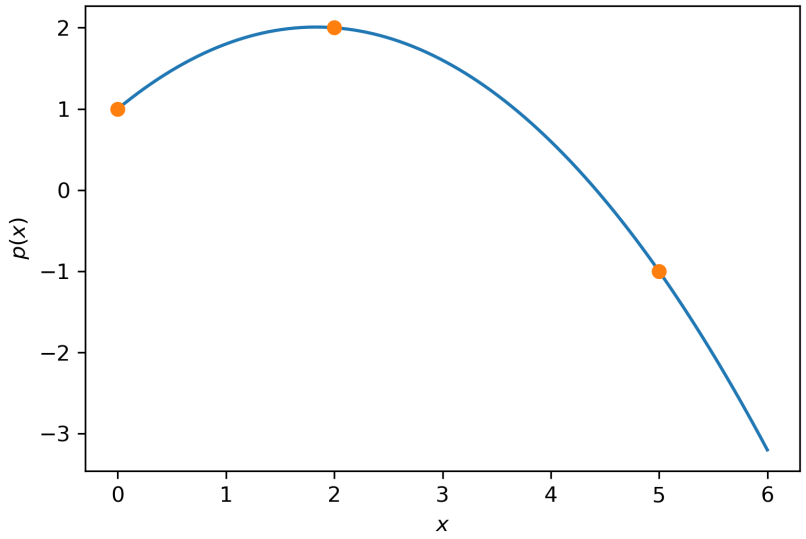
We kunnen nu de onbekende coefficienten c_k vinden door te eisen dat $p_n(x_i) = y_i$, hetgeen leidt tot een stelsel van $n+1$ vergelijkingen in $n+1$ onbekenden $Bc = y$ waarbij

$$B = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix}.$$

Dit is een *Vandermondematrix*, waarvan de determinant is gegeven door

$$\det(B) = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (x_i - x_j).$$

Het stelsel vergelijkingen heeft dus een unieke oplossing wanneer de punten x_i allemaal verschillend zijn. Het interpolerende polynoom is dan dus uniek. We zullen zien dat er verschillende manieren zijn om deze te construeren, maar uiteindelijke zullen ze allemaal tot hetzelfde polynoom leiden.



Figuur 5.1: Interpoleren polynoom door de punten $\{(0,1),(2,2),(5,-1)\}$

Voorbeeld 5.1. ***Verbind de punten:*** *Vind een polynoom door de punten* $\{(0,1),(2,2),(5,-1)\}$. *Het bijbehorende stelsel vergelijkingen is*

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 4 \\ 1 & 5 & 25 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}.$$

Het bijbehorende polynoom is

$$p_2(x) = -0.3x^2 + 1.1x + 1,$$

en is weergegeven in figuur 5.1.

We kunnen het interpolerende polynoom opstellen met behulp van een willekeurige basis voor n-de graads polynomen $\{\phi_0, \phi_1, \ldots, \phi_n\}$. Dit polynoom is uniek wanneer de polynomen ϕ_i lineair onafhankelijk zijn. Volgens de definitie van lineaire onafhankelijkheid zijn er dan immers geen coëfficiënten c_i te vinden zodat $\sum_{j=0}^{n} c_j \phi_j(x) = 0$ voor alle x. Aangezien de ϕ_i polynomen zijn van graad n of minder, hebben ze maximaal n nulpunten en is het voldoende om ze op $n+1$ punten te bekijken.

Voorbeeld 5.2. ***Lineair afhankelijke polynomen:***
Gegeven zijn de volgende 3 polynomen:

$$\phi_0(x) = x^2 - 1, \quad \phi_1(x) = 2x^2, \quad \phi_2(x) = 5.$$

We kiezen als steunpunten $\{0,1,2\}$ *en krijgen dan de volgende matrix*

$$B = \begin{pmatrix} -1 & 0 & 5 \\ 0 & 2 & 5 \\ 3 & 8 & 5 \end{pmatrix}.$$

We zien dat $Bc = 0$ *met* $c = (5, -5/2, 1)$*; de 3 polynomen zijn dus niet lineair onafhankelijk. We vinden inderdaad dat*

$$5\phi_0(x) - (5/2)\phi_1(x) + \phi_2(x) = 0.$$

5.1.1 Lagrange polynomen

De Lagrange polynomen zijn zo gedefinieerd dat de bijbehorende matrix B de identiteitsmatrix is, ofwel $\phi_i(x_j) = \delta_{ij}$. Hier is δ_{ij} de *Kronecker delta*

$$\delta_{ij} = \begin{cases} 1 & \text{als } i = j \\ 0 & \text{als } i \neq j \end{cases}.$$

Voor ieder steunpunt zoeken we een bijbehorend polynoom dat de waarde 1 aanneemt op dat steunpunt en 0 op alle andere steunpunten. Dit polynoom heeft dus n nulpunten en moet dus wel minstens een $n-$de graads polynoom zijn. Omdat de nulpunten gegeven zijn kunnen we zo'n polynoom gemakkelijk construeren. We noemen deze polynomen de *Lagrange polynomen*.

Definitie 5.1. ***Lagrange polynomen:*** *Voor gegeven steunpunten $\{x_i\}_{i=0}^n$ zijn de Lagrange polynomen $\{l_i\}_{i=0}^n$ gegeven door*

$$l_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}.$$

Deze polynomen hebben de eigenschap dat $l_i(x_j) = \delta_{ij}$.

We kunnen het interpolerende polynoom nu gemakkelijk uitdrukken in termen van de Lagrange polynomen als

$$p_n(x) = \sum_{i=0}^{n} y_i l_i(x).$$

Voorbeeld 5.3. ***Lagrange polynomen voor $n = 2$:*** *Voor de steunpunten $\{0, 2, 5\}$ zijn de Lagrange polynomen gegeven door*

$$l_0(x) = \frac{1}{10}(x-2)(x-5),$$

$$l_1(x) = \frac{-1}{6}x(x-5),$$

$$l_2(x) = \frac{1}{15}x(x-2).$$

Ze zijn te zien in figuur 5.2.

5.1.2 Newton polynomen

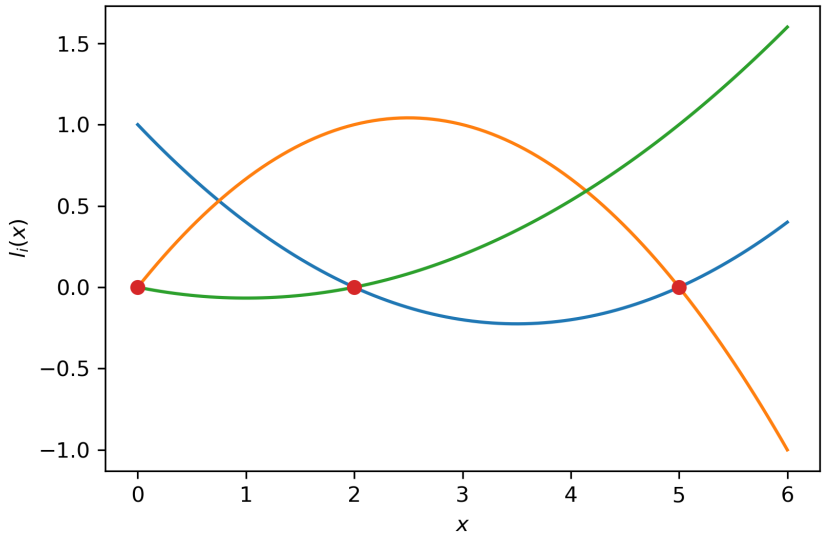
De Lagrange polynomen zijn erg handig omdat we geen stelsel vergelijkingen hoeven op te lossen om de coëfficiënten te bepalen. Een mogelijk nadeel is dat we de polynomen opnieuw moeten bepalen als we een steunpunt toevoegen. Om dit te voorkomen willen we dus polynomen ϕ_i vinden die alleen maar van de steunpunten $x_{j<i}$ afhangen. We zullen zien dat het bijbehorende stelsel benedendiagonaal is en dus gemakkelijk kan worden opgelost door middel van voorwaartse substitutie. Zulke polynomen noemen we de *Newton polynomen*:

Definitie 5.2. ***Newton polynomen:*** *Voor gegeven steunpunten x_i zijn de Newton polynomen $\{\phi_i\}_{i=0}^n$ gegeven door*

$$\phi_0(x) = 1,$$

$$\phi_i(x) = \prod_{j=0}^{i-1}(x - x_j), \quad i = 1, \ldots, n.$$

Ze hebben de eigenschap dat $\phi_i(x_j) = 0$ voor $i > j$.



Figuur 5.2: Lagrange polynomen voor de steunpunten $\{0, 2, 5\}$

Voorbeeld 5.4. ***Newton interpolatie:*** *We beginnen met het polynoom door de punten* $\{(0,1),(5,-1)\}$. *De Newton polynomen zijn*

$$\phi_0(x) = 1, \quad \phi_1(x) = x.$$

De coëfficiënten worden verkregen door het volgende stelsel op te lossen

$$\begin{pmatrix} 1 & 0 \\ 1 & 5 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

We vinden $c_0 = 1$, $c_1 = -2/5$.
Voegen we nu het steunpunt $x_2 = 2$ *met waarde* $y_2 = 2$ *toe, dan krijgen we*

$$\phi_0(x) = 1, \quad \phi_1(x) = x, \quad \phi_2(x) = x(x-5)$$

en stelsel

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 5 & 0 \\ 1 & 2 & -6 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix},$$

met oplossing $c_0 = 1$, $c_1 = -2/5$, $c_2 = -3/10$. *We hoeven hier dus alleen het laatste coëfficiënt uit te rekenen om het polynoom aan te passen. We zien inderdaad in dat*

$$p_2(x) = p_1(x) + c_2\phi_2(x).$$

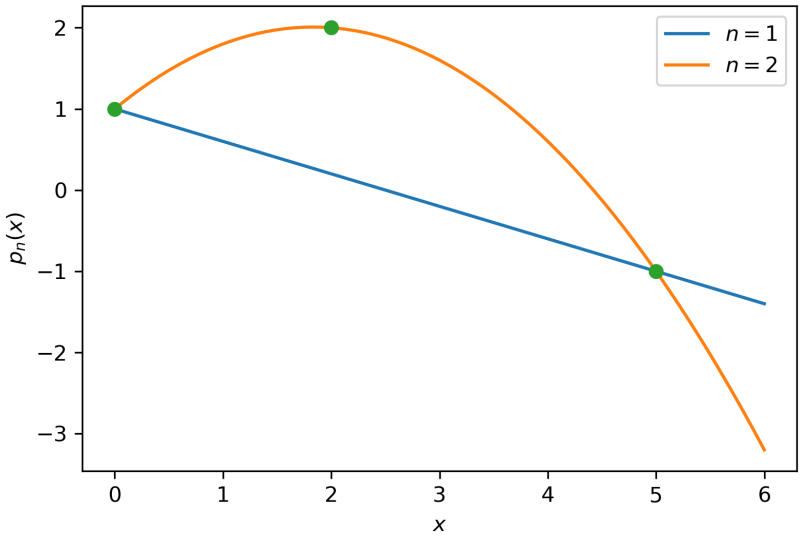
De bijbehorende interpolerende polynomen zijn te zien in figuur 5.3.

Het stelsel vergelijkingen dat we krijgen bij Newton interpolatie heeft een speciale structuur, het is een *benedendriehoeksmatrix*:

$$\begin{pmatrix} 1 & 0 & 0 & \\ 1 & (x_1 - x_0) & 0 & \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & 0 \\ \vdots & & \ddots & \end{pmatrix}$$

We kunnen het stelsel gemakkelijk oplossen door middel van een Gauss-eliminatie:

$$c_0 = y_0, \quad c_1 = \frac{y_1 - y_0}{x_1 - x_0}, \quad c_2 = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}$$



Figuur 5.3: Newton interpolatie voor $n = 1$ en $n = 2$

De oplossing heeft dus een speciale recursieve structuur. Om deze te kunnen benutten voeren we wat nieuwe notatie in; de *gedeelde differenties*:

Definitie 5.3. ***Gedeelde differenties:*** *De gedeelde differenties van een functie f op steunpunten $\{x_k\}_{k=0}^n$ zijn gedefinieerd als*

$$f[x_i] = f(x_i), \quad f[x_i, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}.$$

De volgorde van de punten $\{x_k\}_{k=0}^n$ maakt niet uit. We kunnen de recursieve structuur duidelijk maken in tabelvorm:

x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_0, x_1, x_2]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

We krijgen dan de volgende compacte uitdrukking voor de Newton interpolant:

$$p_n(x) = \sum_{i=0}^{n} f[x_0, \ldots, x_i]\phi_i(x).$$

Een mooie eigenschap van deze vorm van de interpolant is dat we deze gemakkelijk kunnen aanpassen wanneer we een punt toevoegen:

$$p_{n+1}(x) = p_n(x) + f[x_0, \ldots, x_n]\phi_{n+1}(x).$$

De gedeelde differenties hebben de volgende handige eigenschappen die we later nodig zullen hebben:

Stelling 5.1. ***Symmetrie van gedeelde differenties:*** *We kunnen de gedeelde differentie van f uitdrukken als*

$$f[x_0, x_1, \ldots, x_n] = \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{j \neq i}(x_i - x_j)}.$$

We zien dus dat de volgorde van de steunpunten niet uitmaakt.

Stelling 5.2. ***Middelwaardestelling voor gedeelde differenties:*** *De gedeelde differentie van een functie $f \in C^n([a,b])$ op $n+1$ verschillende punten $\{x_i\}_{i=0}^n$ heeft de volgende relatie met de n-de afgeleide van f:*

$$f[x_0,\ldots,x_n] = \frac{f^{(n)}(\xi)}{n!}.$$

Stelling 5.3. ***Gedeelde differenties van polynomen:*** *Gegeven een m-de graads polynoom, f, dan is $f[x_0,\ldots,x_n]$ een polynoom van graad $\max\{0, m-n\}$ in ieder van de variabelen x_i. Voor $m = n$ is $f[x_0,\ldots,x_n]$ dus constant en voor $m < n$ hebben we $f[x_0,\ldots,x_n] = 0$.*

Stelling 5.4. ***Afgeleide van een gedeelde differentie:*** *De afgeleide van een gedeelde differentie is een hogere-orde gedeelde differentie:*

$$\frac{\mathrm{d}}{\mathrm{d}x} f[x_0,x_1,\ldots,x_n,x] = f[x_0,x_1,\ldots,x_n,x,x].$$

5.1.3 Stuksgewijs polynomen

Het is niet altijd aantrekkelijk om met polynomen te werken. Als we bijvoorbeeld 100 steunpunten hebben worden we gedwongen met een polynoom van graad 99 te werken! Liever zouden we dat geval de functie lokaal benaderen met een lage orde polynoom. We kunnen dit in hetzelfde raamwerk doen als we tot nu toe deden door met *stuksgewijs polynomiale functies* te werken.

Voorbeeld 5.5. ***Stuksgewijs lineaire interpolatie:*** *Een stuksgewijs lineaire functie met breekpunten x_i kan worden gedefinieerd als*

$$p(x) = \sum_{k=0}^{n-1} I_{[x_k,x_{k+1})}(x)(c_k + c_k' x),$$

waarbij

$$I_{[a,b)}(x) = \begin{cases} 1 & x \in [a,b) \\ 0 & \textit{anders} \end{cases}.$$

Voor gegeven waarden y_i bepalen we de coefficienten c_i, c_i' door te eisen dat $p(x_i) = y_i$ en dat p continue is op de breekpunten $x_1,\ldots,x_{n-1}$. We vinden

$$c_i + c_i' x_i = y_i,$$

$$c_i + c_i' x_{i+1} = y_{i+1},$$

met oplossing $c_i = ((y_i + y_{i+1}) - c_i'(x_i + x_{i+1}))/2$, $c_i' = (y_{i+1} - y_i)/(x_{i+1} - x_i)$

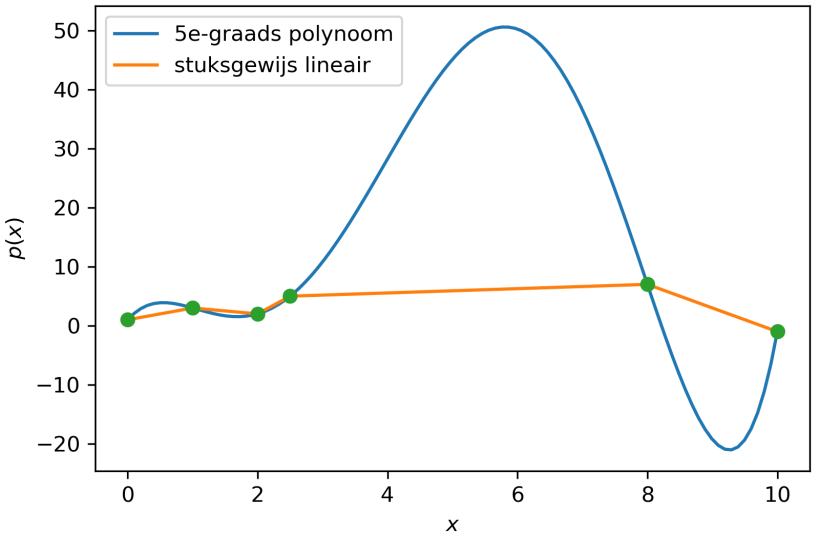
Net als bij globale interpolatie, kunnen we het oplossen van een stelsel vergelijkingen hier voorkomen door de basisfuncties slim te kiezen. In het geval van stuksgewijs lineaire interpolatie kiezen we

$$s_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & x \in [x_{i-1}, x_i) \\ \frac{x-x_{i+1}}{x_i-x_{i+1}} & x \in [x_i, x_{i+1}] \\ 0 & \text{anders} \end{cases}.$$

We noemen deze basisfuncties *lineaire splines*. Het interpolerende polynoom is nu gegeven door

$$p(x) = \sum_{i=0}^{n} y_i s_i(x).$$

Als we het polynoom ook buiten het interval $[x_0, x_n]$ willen evalueren dan moeten we nog twee extra steunpunten x_{-1} en x_{n+1} toevoegen. Op deze punten zal het interpolerende polynoom 0 zijn.
Zoals de naam doet vermoeden zijn er ook splines van hogere orde. We kunnen op die manier een gladder interpolerend polynoom maken.



Figuur 5.4: Interpolatie met stuksgewijs lineair polynoom en 5−de graads polynoom.

Voorbeeld 5.6. ***kubische splines:*** *Voor stuksgewijs kubische polynomen* $p_i(x) = c_i + c_i'(x - x_i) + c_i''(x - x_i)^2 + c_i'''(x - x_i)^3$ *voor* $i = 0, \ldots, n-1$ *hebben we* 4 *onbekenden per interval* $[x_i, x_{i+1}]$. *De vergelijkingen voor ieder interval zijn dan*

$$p_i(x_i) = y_i, \quad p_i(x_{i+1}) = y_{i+1}, \quad i = 0, \ldots, n-1,$$

$$p_i'(x_{i+1}) = p_{i+1}'(x_{i+1}), \quad p_i''(x_{i+1}) = p_{i+1}''(x_{i+1}) \quad i = 0, \ldots, n-2$$

Op deze manier kunnen we een stelsel van $4n - 2$ *vergelijkingen in* $4n$ *onbekenden opstellen. Op de randen missen we dus een vergelijking, en hier wordt vaak gekozen om de afgeleide op nul te zetten:* $p_0'(x_0) = 0$ *en* $p_{n-1}'(x_n) = 0$. *In de Python-implementatie is ervoor gekozen om de eerste en laatste 3 steunpunten met ieder met één derdegraads polynoom te beschrijven. Een voorbeeld is te zien in figuur 5.5.*

5.2 Foutschatting

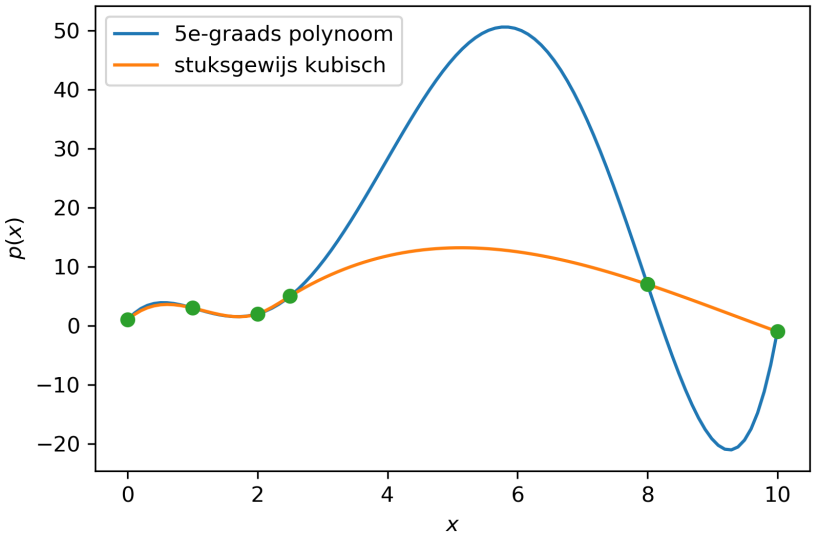
Indien de waarden y_i zijn verkregen door een functie f te evalueren op de steunpunten, hoe groot is dan de interpolatiefout $e_n(x) = f(x) - p_n(x)$?

Stelling 5.5. ***Interpolatiefout:*** *Zij* $f \in C^{n+1}([a,b])$ *en* $p_n(x)$ *het* $n-$*de graads interpolerende polynoom met steunpunten* $a \leq x_0 < x_1 < \ldots < x_{n+1} \leq b$. *De interpolatiefout voor* $x \in [a,b]$ *is gegeven door*

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i),$$

met $\xi_x \in [a,b]$. *Merk op dat* ξ_x *afhangt van* x.

We zien hieruit dat de maximale fout dus afhangt van twee factoren; de grootte van de $n+1$-ste afgeleide van f en de maximale waarde van het polynoom $\phi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$. We zullen later zien dat we de steunpunten zo kunnen kiezen dat de invloed hiervan kan worden geminimaliseerd.



Figuur 5.5: Interpolatie met stuksgewijs kubisch polynoom en 5−de graads polynoom.

Voorbeeld 5.7. ***De Sinus:*** *We kunnen bovenstaande stelling toepassen op een kwadratische benadering van* $\sin(\pi x)$*:* $p_2(x) = 4x(1-x)$ *met steunpunten* $\{0, \frac{1}{2}, 1\}$*. We krijgen dan*

$$e_2(x) = \frac{-\pi^3 \cos(\pi\xi)}{6} x(x-\tfrac{1}{2})(x-1),$$

wat we kunnen afschatten als

$$\|e_2\|_\infty \leq \frac{\pi^3}{72\sqrt{3}} \approx 0.248.$$

In figuur 5.6 zien we dat de fout in de praktijk veel kleiner is dan deze grove bovengrens.

Voorbeeld 5.8. ***Stuksgewijs lineaire interpolatie:*** *Als we een stuksgewijs lineaire interpolant,* p*, van een functie* f *opstellen op de steunpunten* $\{x_i\}_{i=0}^n$*, dan kunnen we bovenstaande stelling ook toepassen. Op ieder deel-interval* $[x_i, x_{i+1}]$ *hebben we immers een lineaire benadering van de functie met fout*

$$e(x) = \frac{f''(\xi_i)}{2}(x-x_i)(x-x_{i+1}),$$

met $x, \xi_i \in [x_i, x_{i+1}]$*. De maximale fout over het hele interval is dan gegeven door*

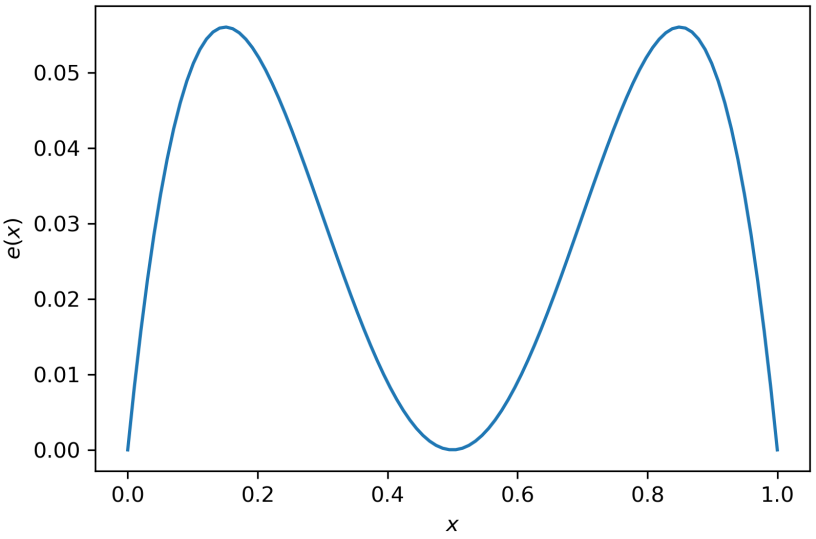
$$\|f-p\|_\infty \leq C\|f''\|_\infty h^2,$$

met $C = 1/4$ *and* $h = \max_i |x_{i+1} - x_i|$*.*

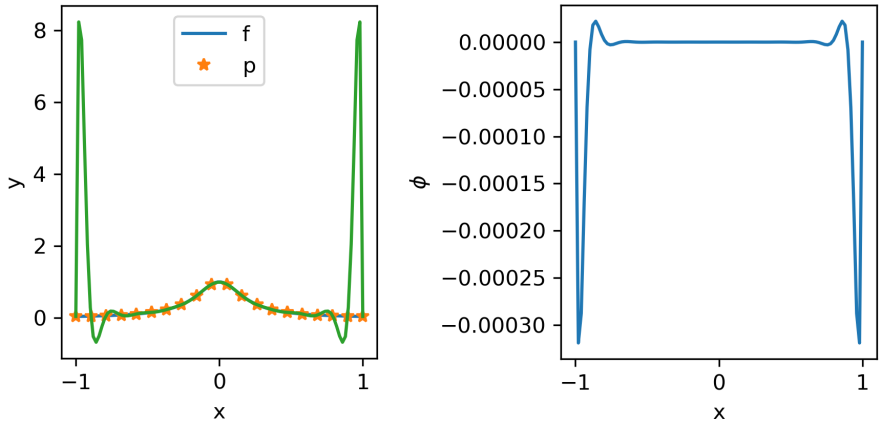
Voorbeeld 5.9. ***Een randgeval:*** *We willen de functie* $f(x) = (1+25x^2)^{-1}$ *interpoleren. In figuur 5.7 zien het resultaat voor* $n = 20$*. De grote fout aan de rand kan worden verklaard uit het feit dat* $f^{(n+1)}(x)$ *daar opblaast en bovendien* ϕ_{n+1} *niet uniform naar nul convergeert.*

5.3 Chebyshevinterpolatie*

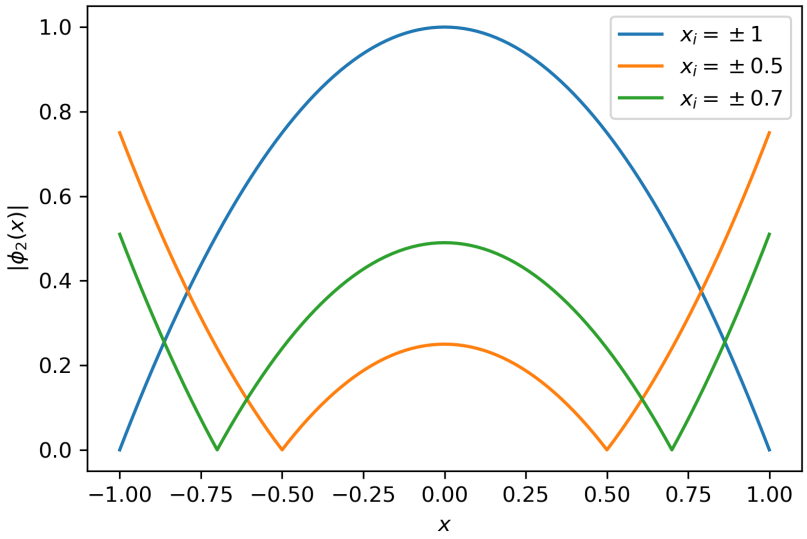
We zagen eerder al dat de fout wordt beïnvloed door het polynoom $\phi_{n+1}(x) = \prod_{i=0}^n (x-x_i)$. We kunnen ons nu het volgende afvragen:



Figuur 5.6: De functie $\sin(x)$ en een kwadratische benadering $\widetilde{f}(x) = 4x(1-x)$.



Figuur 5.7: Interpolatie van $f(x) = (1+25x^2)^{-1}$ voor $n = 20$. We zien dat de fout op de rand bijzonder groot is.



Figuur 5.8: Grafiek van $|\phi_2|$ voor verschillende steunpunten.

Wanneer we de vrijheid hebben om de steunpunten zelf te kiezen, hoe kiezen we ze dan zo dat $\|\phi_{n+1}(x)\|_\infty$ zo klein mogelijk is?

Voorbeeld 5.10. ***Optimale steunpunten voor*** $n = 1$***:*** *Voor $n = 1$ hebben we $\phi_2(x) = (x - x_0)(x - x_1)$. Als we de grafiek tekenen op het interval $[-1, 1]$ en de steunpunten varieren zien we in figuur 5.8 dat het maximum van ϕ_2 sterk varieert. We vinden dat $x_0 = -x_1 \approx 0.7$ een maximum van ongeveer $1/2$ oplevert. Het blijkt dat de optimale waarde is gegeven door $x_0 = -x_1 = 1/\sqrt{2}$.*

Definitie 5.4. ***Chebyshev polynomen:*** *De polynomen die in deze zin optimaal zijn heten Chebyshev polynomen. Ze hebben de eigenschap dat alle extrema precies de waarde ± 1 hebben. Op het interval $[-1, 1]$ zijn ze gedefinieerd als*

$$T_n(x) = \cos(n \arccos(x)).$$

We zien direct dat $T_0(x) = 1$ en $T_1(x) = x$. Met hehulp van de trigonometrische identiteit $\cos(n\theta + \theta) - \cos(n\theta - \theta) = 2$ vinden we de volgende relatie

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

Verder lezen we van de definitie af dat de nulpunten van $T_n(x)$ zijn gegeven door

$$x_k = \cos\left(\frac{(2k-1)\pi}{2n}\right), \quad k = 1, \dots n$$

Tot slot zien we dat voor $n > 0$ de leidende term van $T_n(x)$ is gegeven door $2^{n-1}x^n$.

Als we de nulpunten van T_{n+1} kiezen als de steunpunten voor interpolatie dan hebben we $\phi_{n+1}(x) = 2^{-n}T_{n+1}(x)$ en dus dat $\|\phi_{n+1}\|_\infty = 2^{-n}$. In dat geval kunnen we de interpolatie als volgt uitdrukken:

Stelling 5.6. ***Chebyshevinterpolatie:*** *De fout voor Chebyshevinterpolatie is*

$$\|f - p_n\|_\infty \leq 2^{-n} \frac{\|f^{(n+1)}\|_\infty}{(n+1)!}.$$

5.4 Beste benadering met polynomen*

Het doel nu is om een gegeven functie f zo goed mogelijk te benaderen op het interval $[a,b]$ met een polynoom van graad n. We zoeken dus een polynomoom p_n van graad n waarvoor $\|f-p_n\|^2$ zo klein mogelijk is. Gegeven een geschikte basis $\{\phi_k\}_{k=0}^n$ voor de n-de graads polynomen kunnen we deze eis uitdrukken als

$$\langle \phi_k, f-p_n\rangle = 0, \quad \text{voor} \quad k=0,\ldots,n \quad .$$

Wanneer we p_n uitdrukken als

$$p_n(x) = \sum_{k=0}^{n} c_k\phi_k(x),$$

dan krijgen we een stelsel vergelijkingen

$$Ac = b,$$

met $b_k = \langle f,\phi_i\rangle$ en $a_{ij} = \langle \phi_i,\phi_j\rangle$.

5.4.1 Orthogonale polynomen

Het zou aardig zijn als we een *orthogonale* basis voor de vectorruimte van $n-$de graads polynomen zouden kunnen vinden; de matrix A is in dat geval een diagonaalmatrix. We kunnen het polynoom dat een gegeven f het best benadert dan uitdrukken als

$$p_n = \sum_{k=0}^{n} \frac{\langle f,\phi_k\rangle}{\|\phi_k\|^2}\phi_k.$$

Omdat we een inproduct kunnen uitrekenen kunnen we de Gram-Schmidt procedure gebruiken om zo'n orthogonale basis te maken. We beginnen simpelweg met de canonieke basis $\{1,x,x^2,x^2,\ldots\}$ en passen dan GS toe. Het resultaat zal uiteraard afhangen van het inproduct dat we kiezen.

Voorbeeld 5.11. ***Legendre polynomen:*** *Het standaard inproduct voor het interval* $[-1,1]$

$$\langle p,q\rangle = \int_{-1}^{1} p(x)q(x)\mathrm{d}x,$$

levert de zogenaamde Legendre polynomen *op. Wanneer we de GS-procedure toepassen op de kanonieke basis* $\{1,x,x^2,\ldots\}$ *krijgen we*

$$\phi_0(x)=1, \quad \phi_1(x)=x, \quad \phi_2(x)=\tfrac{1}{2}(3x^2-1).$$

Voorbeeld 5.12. ***Chebyshev polynomen:*** *De Chebyshev polynomen die we eerder zagen zijn orthogonaal ten opzichte van een gewogen inproduct:*

$$\langle p,q\rangle = \int_{-1}^{1} p(x)q(x)w(x)\mathrm{d}x,$$

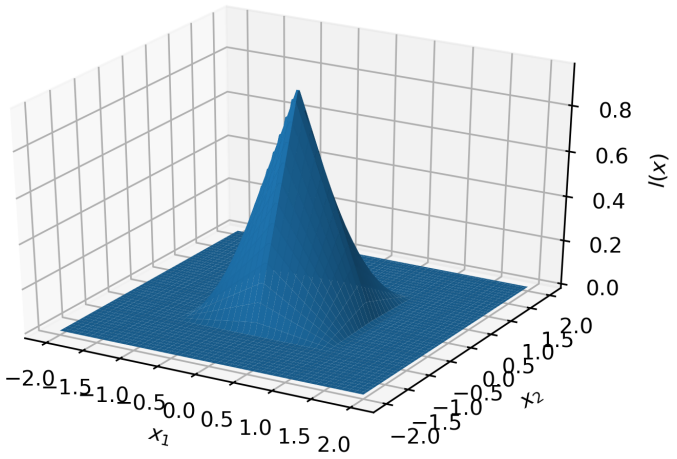
met $w(x) = 1/\sqrt{1-x^2}$.

5.5 Interpolatie in meer dimensies*

De vraagstelling in meer dimensies is in feite hetzelfde; voor gegeven steunpunten $x^{(i)} \in \mathbb{R}^d$ en data $y^{(i)} \in \mathbb{R}$, vind een functie $p:\mathbb{R}^d \to \mathbb{R}$ zodat $p(x^{(i)}) = y^{(i)}$. We kunnen het idee achter Lagrange interpolatie in één dimensie proberen uit te breiden en op zoek gaan naar functies $l_i(x)$ die voldoen aan $l_i(x^{(j)}) = \delta_{ij}$. Hieronder twee voorbeelden.

Voorbeeld 5.13. ***Bi-lineaire interpolatie:*** *Als we een functie* $f:[a,b]^2 \to \mathbb{R}$ *willen benaderen gegeven de waarden op die 4 hoekpunten kunnen we als basis de volgende Lagrange polynomen nemen:*

$$l_0(x) = \frac{(b-x_1)}{(b-a)}\cdot\frac{(b-x_2)}{(b-a)}, \quad l_1(x) = \frac{(x_1-a)}{(b-a)}\cdot\frac{(b-x_2)}{(b-a)},$$



Figuur 5.9: Basisfunctie voor bi-lineaire Lagrange-interpolatie.

$$l_2(x) = \frac{(b - x_1)}{(b - a)} \cdot \frac{(x_2 - a)}{(b - a)}, \quad l_3(x) = \frac{(x_1 - a)}{(b - a)} \cdot \frac{(x_2 - a)}{(b - a)}.$$

De basisfuncties zijn in dit geval simpelweg het directe product van de ééndimensionale Lagrange polynomen. Dit komt omdat het tweedimensionale rooster het cartesische product is van twee ééndimensionale roosters.
Op deze manier kunnen we een functie in twee variabelen stuksgewijs lineair interpoleren met behulp van piramidevormige basisfuncties (figuur 5.9).

Wanneer de steunpunten geen regelmatig rooster vormen zullen we iets anders moeten verzinnen. We kunnen uiteraard nog steeds met een basis voor multivariate functies werken en de interpolatiecondities gebruiken om een stelsel vergelijkingen op te lossen. Het is echter in het algemeen niet mogelijk om te garanderen dat het stelsel vergelijkingen ook een oplossing heeft. In specifieke gevallen, zoals stuksgewijs lineaire interpolatie kan dat wel.

Voorbeeld 5.14. ***Stuksgewijs lineaire interpolatie in*** $\mathbb{R}^2$***:*** *Om de interpolant te evalueren op een punt x, zoeken we de drie dichstbijzijnde steunpunten, $\{x^{(a)}, x^{(b)}, x^{(c)}\}$, en benaderen de functie als een lineaire combinatie van de functiewaarden $\{y^{(a)}, y^{(b)}, y^{(c)}\}$ op die drie punten:*

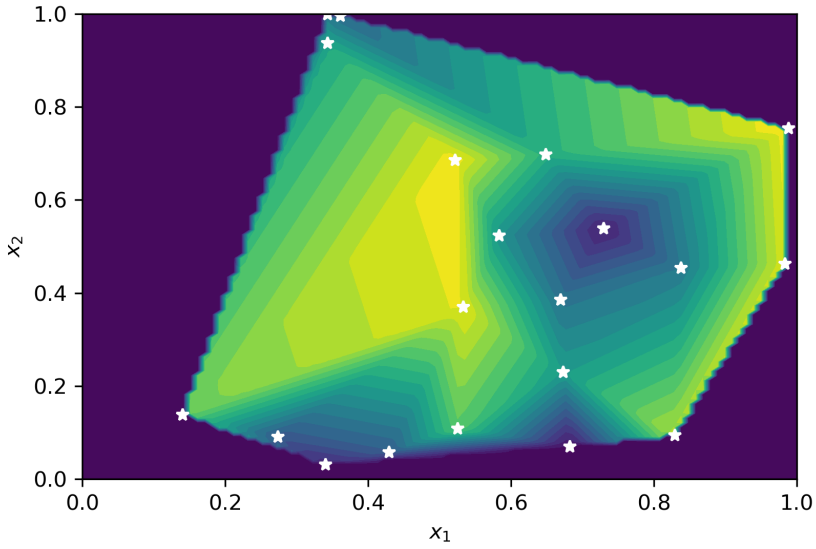
$$p(x) = w_a y^{(a)} + w_b y^{(b)} + w_c y^{(c)},$$

waarbij de w_a, w_b, w_c de baricentrische coordinaten *van de driehoek zijn. We krijgen deze door een stelsel vergelijkingen op te lossen:*

$$\begin{pmatrix} 1 & 1 & 1 \\ x_0^{(a)} & x_0^{(b)} & x_0^{(c)} \\ x_1^{(a)} & x_1^{(b)} & x_1^{(c)} \end{pmatrix} \begin{pmatrix} w_a \\ w_b \\ w_c \end{pmatrix} = \begin{pmatrix} 1 \\ x_0 \\ x_1 \end{pmatrix}.$$

Een voorbeeld is te zien in figuur 5.10

Voorbeeld 5.15. ***machine learning:*** *Veel problemen in de* machine learning *zijn ook te zien als interpolatieproblemen. Neem als voorbeeld het classificeren van punten in het vlak. Gegeven zijn m punten $x^{(i)}$, ieder met een bijbehorend* label *y_i. We zoeken nu een functie $p : \mathbb{R}^2 \to \mathbb{R}$ waarvoor $p(x^{(i)}) \approx y^{(i)}$. Uiteraard zijn er oneindig veel*



Figuur 5.10: Voorbeeld van stuksgewijs lineaire interpolatie.

functies die door de gegeven punten gaan, maar we willen graag een gladde functie. Een populaire manier om zo'n functie weer te geven is als een Neuraal Netwerk*, zoals bijvoorbeeld*

$$p(x) = c_0 + \sum_{i=1}^{2} c_i \sigma\left(\langle w_i, x\rangle + b_i\right),$$

waar $w_i \in \mathbb{R}^2$, $b_i \in \mathbb{R}$ en $\sigma(s) = (1+e^{-s})^{-1}$. De gewichten $\{(w, c, b)\}$ worden zo gekozen dat $p(x^{(i)}) \approx y^{(i)}$ door een stelsel niet-lineaire vergelijkingen op te lossen.

5.6 Opgaven

Opdracht 5.1. ***Interpolatie:*** *Stel het interpolerende polynoom op met steunpunten $\{0, 1, 2, 3\}$ en waarden $\{-4, -2, 2, 14\}$. Dit dit op 3 verschillende manieren; door een stelsel op te lossen met de Vandermonde matrix, met Lagrange polynomen en met Newton polynomen.*

Opdracht 5.2. ***Interpolatie met afgeleiden:*** *Gegeven zijn 2 steunpunten $x_0 = 0$ en $x_1 = 1$, de functiewaarden $f(0) = 1$ en $f(1) = 2$ en de afgeleiden $f'(0) = 0$ en $f'(1) = -1$.*
Stel een interpolerend polynoom, p, van graad 3 op zodat $p(x_i) = f(x_i)$ en $p'(x_i) = f'(x_i)$ voor $i = 0, 1$.

Opdracht 5.3. ***Sinus:*** *Stel een kwadratische benadering voor $\sin(x)$ op voor het interval $[0, \pi]$ met steunpunten $\{0, \pi/2, \pi\}$ en geef een bovengrens voor de benaderingsfout $e(x) = |\sin(x) - p_2(x)|$. Verifieer je bovengrens door middel van een numeriek experiment.*

Opdracht 5.4. ***Gedeelde differenties:*** *Gegeven en set steunpunten $\{x_i\}_{i=0}^{n}$, definieer de functie $g_n(x) = f[x_0, x_1, \ldots, x_n, x]$. Gegeven is dat f een polynoom van graad m is. Laat zien dat:*

1. *$g_n(x) = 0$ als $m \leq n$*
2. *$g_n(x)$ een polynoom is van graad $m - n - 1$ is als $m > n$.*

Opdracht 5.5. ***Extrapolatie:*** *Benader de functie $f(x) = 2^x$ met steunpunten $\{0, 1\}$.*

1. *Stel het interpolerende polynoom p op*

2. *Geef een bovengrens voor de fout wanneer we $\sqrt{2}$ benaderen met $p(1/2)$*
3. *Geef een bovengrens voor de fout wanneer we 2^3 benaderen met $p(3)$*
4. *Verklaar waarom de fout op $x = 3$ veel groter is dan de fout op $x = 1/2$.*

Opdracht 5.6. ***Stuksgewijs kwadratische interpolatie:*** *Schrijf een algoritme om een stuksgewijs kwadratische interpolant op te stellen voor gegeven invoer $\{(x_i, y_i)\}_{i=0}^{n}$. Je mag aannemen dat $n \geq 3$.*

Opdracht 5.7. ***Kleinste kwadraten:*** *Gegeven zijn 4 steunpunten $\{1, 2, 3, 4\}$ en functiewaarden $\{1, 4, 10, 15\}$. We willen een 2^e-graads polynoom vinden dat zo goed mogelijk door deze punten gaat.*

1. *Stel een stelsel vergelijkingen op om dit polynoom te vinden*
2. *Laat in een grafiek zien hoe goed het polynoom de punten benadert*

Opdracht 5.8. ***Orthogonale polynomen:*** *Geef de polynomen p_i die orthogonaal zijn in het gewogen inproduct*

$$\langle p, q \rangle = \int_0^1 p(x)q(x)x \, \mathrm{d}x.$$

Opdracht 5.9. ***Beste benadering met Chebyshevpolynomen:*** *Definieer het volgende inproduct voor functies op het interval $[-1, 1]$:*

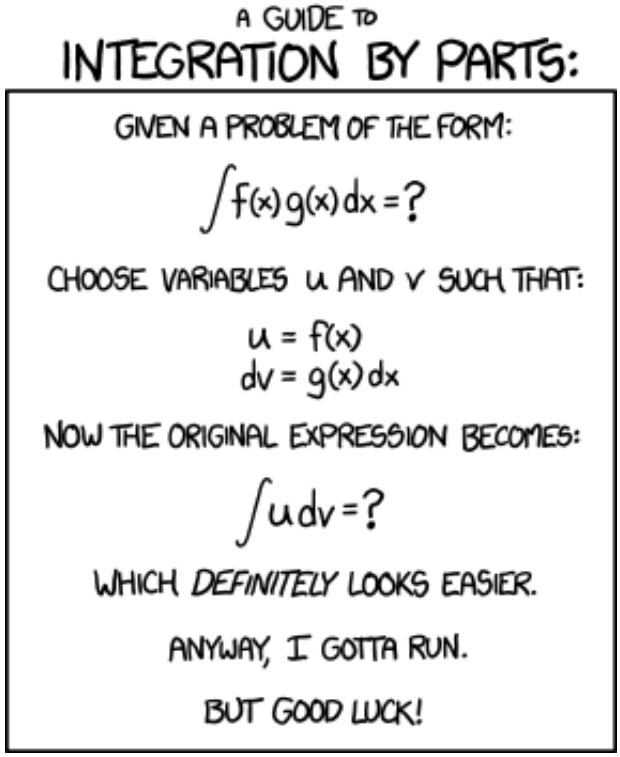
$$\langle p, q \rangle = \int_{-1}^1 p(x)q(x)w(x)\mathrm{d}x,$$

met $w(x) = 1/\sqrt{1 - x^2}$. Vind het 2-de graads polynoom die de functie $f(x) = x^6 - 2x^2$
Hoe verhoudt deze benadering zich tot het interpolerende Chebyshevpolynoom?

Opdracht 5.10. ***Barycentrische interpolatie:*** *Laat zien dat de interpolant in het voorbeeld* Stuksgewijs lineaire interpolatie *exact is voor lineaire functies ($f(x) = ax_0 + bx_1 + c$). Wat gebeurt er als de openingshoek van de driehoek zeer klein wordt?*

Hoofdstuk 6

Integreren en differentieren



In dit hoofdstuk gaan we functiebenadering gebruiken om de afgeleide of integraal van een functie te schatten. Het ligt hier voor de hand om hiervoor direct de interpolant te gebruiken. Als f wordt benaderd door een interpolant p, dan kunnen we de afgeleide benaderen als

$$f'(x) \approx p'(x),$$

en de integraal als

$$\int_a^b f(x)\mathrm{d}x \approx \int_a^b p(x)\mathrm{d}x.$$

We weten al hoe groot de interpolatiefout is, dus we kunnen daarmee ook de fout in de benadering van de integraal of afgeleide uitdrukken. Centraal in deze discussie staat de *graad van nauwkeurigheid* van de methode. Deze is gedefinieerd als de maximale graad n waarvoor de benadering exact is voor polynomen van graad n. We moeten hierbij in ons achterhoofd houden dat differentieren en integreren niet allebij even makkelijk zijn; bij differentieren verliezen we een graad en bij integreren winnen we er een. Hieronder zien we daar een voorbeeld van:

Voorbeeld 6.1. ***Benadering met 1 steunpunt:*** *Stel, we benaderen een functie f op het interval $[-1,1]$ met een constante $p_0 = f(0)$. De bijbehorende benadering van de afgeleide is $p_0'(x) = 0$. Deze benadering is over het algemeen alleen exact wanneer f een polynoom van graad 0 is. De graad van nauwkeurigheid van deze benadering is dus 0. Benaderen we echter de integraal met p_0 dan krijgen we*

$$\int_{-1}^{1} p_0(x)\mathrm{d}x = 2f(0).$$

Deze benadering heeft een graad van nauwkeurigheid van 1.

6.1 Numerieke Differentiatie

Zoals gezegd kunnen we de afgeleide van een gegeven functie f benaderen door de interpolant te differentieren. Wanneer we de Lagrangevorm van de interpolant nemen krijgen we

$$f^{(k)}(x) \approx p_n^{(k)}(x) = \sum_{i=0}^{n} f(x_i) l_i^{(k)}(x),$$

waarbij $l_i(x)$ het i-de Lagrange polynoom is. Wanneer we de afgeleide vervolgens weer evalueren op de steunpunten dan krijgen we

$$p_n^{(k)}(x_j) = \sum_{i=1}^{n} d_{ij} p_n(x_i),$$

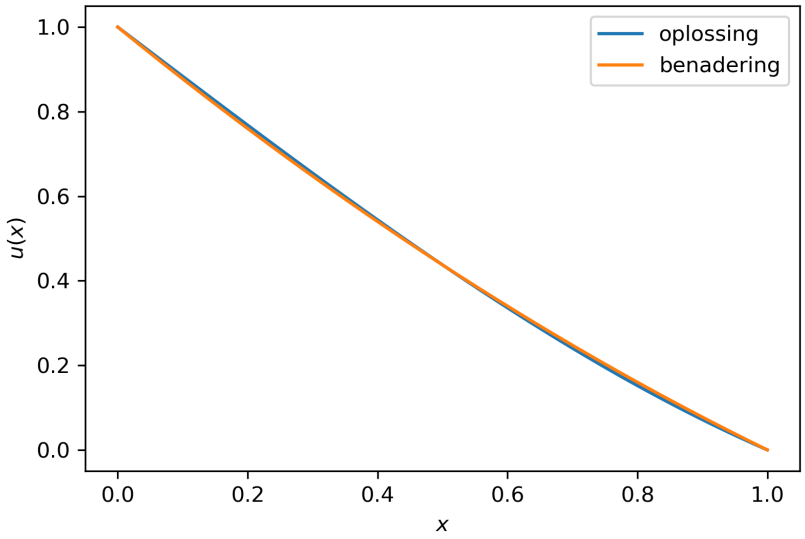
waarbij $d_{ij} = l_i^{(k)}(x_j)$ de elementen van de *differentiatiematrix zijn*. We kunnen dit concept meteen gebruiken om een randwaardeprobleem op te lossen:

Voorbeeld 6.2. ***Een randwaardeprobleem:*** *We willen de volgende differentiaalvergelijking oplossen*

$$u''(x) = x, \quad \textit{voor} \quad x \in (0,1),$$

met $u(0) = 1$ en $u(1) = 0$. De oplossing is $u(x) = (1/6)x^3 - (7/6)x + 1$.
We kunnen de oplossing benaderen door deze uit te drukken als tweedegraads polynoom op de steunpunten $0, \frac{1}{2}, 1$. De waarden van u op deze steunpunten zijn onbekend, dus daar moeten we een stelsel vergelijkingen voor opstellen. We doen dit door te eisen dat de interpolant $\widetilde{u}$ voldoet aan de randvoorwaarden en de differentiaalvergelijking. De interpolant $\widetilde{u}$ is in dit geval gegeven door

$$\widetilde{u}(x) = \widetilde{u}_0 l_0(x) + \widetilde{u}_1 l_1(x) + \widetilde{u}_2 l_2(x).$$



Figuur 6.1: Oplossing van een randwaardeprobleem.

Om de randvoorwaarden te voldoen hebben we $\widetilde{u}(0) = \widetilde{u}_0 = 1$ *en* $\widetilde{u}(1) = \widetilde{u}_2 = 0$. *De derde vergelijking krijgen we door de tweede afgeleide van* $\widetilde{u}$ *op* x_1 *gelijk te stellen aan* x_1: $\widetilde{u}''(x_1) = \widetilde{u}_0 l_0''(x_1) + \widetilde{u}_1 l_1''(x_1) + \widetilde{u}_2 l_2''(x_1) = x_1$. *Het stelsel vergelijkingen voor* $(\widetilde{u}_0, \widetilde{u}_1, \widetilde{u}_2)$ *is nu gegeven door:*

$$\begin{pmatrix} 1 & 0 & 0 \\ 4 & -8 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \widetilde{u}_0 \\ \widetilde{u}_1 \\ \widetilde{u}_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{2} \\ 0 \end{pmatrix}.$$

De numerieke oplossing is $\widetilde{u}_0 = 1$, $\widetilde{u}_1 = \frac{7}{16}$, $\widetilde{u}_2 = 0$. *De benadering is te zien in figuur 6.1.*

We kunnen de fout in de benadering van de afgeleide eenvoudig berekenen aan de hand van de interpolatiefout:

Stelling 6.1. ***Benaderingsfout:*** *Gegeven het interpolerende polynoom* $p_n(x)$ *van* $f(x)$ *op steunpunten* $\{x_i\}_{i=0}^n$, *dan is de benaderingsfout van de afgeleide,* $e_n^{(k)} = f^{(k)} - p_n^{(k)}$, *gegeven door*

$$e_n^{(k)}(x) = \frac{\mathrm{d}^k}{\mathrm{d}x^k}\left(f[x_0, x_1, \ldots, x_n, x]\phi_{n+1}(x)\right) = \sum_{i=0}^{k} \binom{k}{i} \left(\frac{\mathrm{d}^{k-i}}{\mathrm{d}x^{k-i}} f[x_0, x_1, \ldots, x_n, x]\right) \phi_{n+1}^{(i)}(x),$$

met

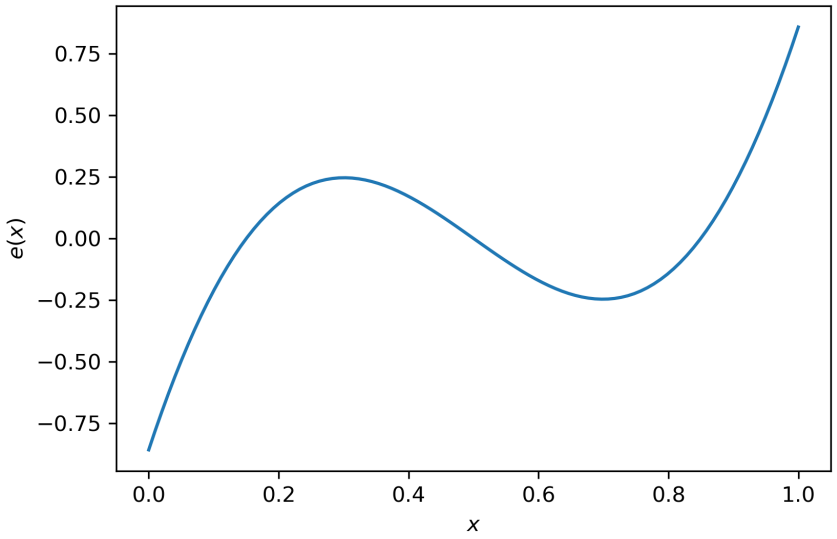
$$\phi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i).$$

Met behulp van de eerdere stellingen over gedeelde differenties kunnen we dit schrijven als

$$e_n^{(k)}(x) = \sum_{i=0}^{k} \binom{k}{i} \frac{f^{(n+1+k-i)}(\xi_i)}{(n+1+k-i)} \phi_{n+1}^{(i)}(x).$$

Voorbeeld 6.3. ***De sinus:*** *We hebben eerder de benadering* $p_2(x) = 4x(1-x)$ *gebruikt om de functie* $\sin(\pi x)$ *te benaderen op het interval* $[0, 1]$. *Op deze manier krijgen we de volgende benadering voor de afgeleide:*

$$\pi \cos(\pi x) \approx 4 - 8x.$$



Figuur 6.2: Benaderingsfout wanneer we de afgeleide van de functie $\sin(\pi x)$ benaderen met de afgeleide van $\widetilde{f}(x) = 4x(1-x)$.

De fout in deze benadering is volgens bovenstaande stelling gegeven door

$$e_2'(x) = \frac{\pi^4 \sin(\pi\xi)}{24} x(x-\tfrac{1}{2})(x-1) - \frac{\pi^3 \cos(\pi\xi')}{6}\left(3x^2 - 3x + \tfrac{1}{2}\right).$$

Als we dit afschatten, dan krijgen we

$$\|e_2'\|_\infty \leq \frac{\pi^4}{24 \cdot 12\sqrt{3}} + \frac{\pi^3}{24} \approx 1.48.$$

Zoals te zien in figuur 6.2 is de benadering in werkelijkheid een stuk beter.

6.1.1 Benaderingen op een regelmatig rooster

Wanneer we op een regelmatig rooster $x_i = i \cdot h$ werken krijgen we mooie compacte uitdrukkingen. Voor de eerste afgeleide krijgen we bijvoorbeeld de volgende bekende *eindige differentie* benaderingen:

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0)}{h} \quad \text{(voorwaartse differentie)},$$

$$f'(x_0) \approx \frac{f(x_0) - f(x_0-h)}{h} \quad \text{(terugwaartse differentie)},$$

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0-h)}{2h} \quad \text{(centrale differentie)}.$$

We kunnen de benaderingsfout in dit geval ook gemakkelijk analyseren met behulp van een Taylorreeks, zoals in het onderstaande voorbeeld.

Voorbeeld 6.4. ***Voorwaartse differentie:*** *De Taylorreeks voor f is gegeven door:*

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2 f''(\xi)}{2}, \quad \xi \in [x_0, x_0+h]$$

waaruit direct volgt dat

$$f'(x_0) = \frac{f(x_0+h)-f(x_0)}{h} - \frac{hf''(\xi)}{2}.$$

De benaderingsfout van de voorwaartse eindige differentie is in dit geval dus $\mathcal{O}(h)$. We kunnen ook direct de stelling gebruiken:

$$e_2'(x) = \frac{\mathrm{d}}{\mathrm{d}x} f[x_0, x_0+h, x](x-x_0)(x-x_0-h) + f[x_0, x_0+h, x]((x-x_0)+(x-x_0-h)),$$

waar uit volgt dat

$$e_2'(x_0) = -hf[x_0, x_0+h, x_0].$$

Hier moeten we oppassen met de gedeelde differentie met 2 dezelfde steunpunten. Om dit formeel uit te werken kijken we naar $f[x_0, x_0+h, x]$ en nemen de limiet $x \to x_0$. Uit de definitie van de gedeelde differentie krijgen we

$$f[x_0, x_0+h, x] = \frac{\frac{f(x_0+h)-f(x_0)}{h} - \frac{f(x)-f(x_0)}{(x-x_0)}}{h}.$$

Met behulp van een Taylorexpansie rond x_0 krijgen we

$$f[x_0, x_0+h, x] = \frac{(f'(x_0)+(h/2)f''(\xi)) - (f'(x_0)+((x-x_0)/2)f''(\xi'))}{h} = \frac{1}{2}f''(\xi) + \frac{(x-x_0)}{h}f''(\xi').$$

In de limiet $x \to x_0$ vinden we dus (zoals verwacht)

$$e_2'(x_0) = -\frac{hf''(\xi)}{2}.$$

Hogere orde centrale eindige-differentie benaderingen hebben de algemene vorm

$$f^{(k)}(x_0) = \sum_{i=-m}^{m} w_i f(x_0 + i \cdot h),$$

waarbij $w_i = l_i^{(k)}(x_0)$ met

$$l_i(x) = \frac{\prod_{j\neq i}(x - x_0 - j\cdot h)}{\prod_{j\neq i}((i-j)\cdot h)}.$$

In het algemeen is de fout van zo'n benadering van de vorm

$$e_n^{(k)}(x_0) = C \cdot h^q f^{(r)}(\xi_0), \quad \xi_0 \in [x_0 - mh, x_0 + mh],$$

waarbij q de *orde* van de benadering is en r de *graad van nauwkeurigheid*. Aan gezien de interpolant waar de benadering uit voortkomt exact is voor polynomen van graad $n = 2m$, verwachten we $r \geq 2m+1$.

Voorbeeld 6.5. ***Een benadering voor de tweede afgeleide:*** *Met $m = 1$ en $k = 2$ krijgen we de volgende benadering voor de tweede afgeleide:*

$$f''(x_0) \approx \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2}.$$

De benaderingsfout is gegeven door

$$e_2''(x_0) = \left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} f[x_0-h, x_0, x_0+h, x] \cdot \phi_3(x) + 2\frac{\mathrm{d}}{\mathrm{d}x} f[x_0-h, x_0, x_0+h, x] \cdot \phi_3'(x) + f[x_0-h, x_0, x_0+h, x] \cdot \phi_3''(x) \right|_{x=x_0},$$

met $\phi_3(x) = (x-x_0+h)(x-x_0)(x-x_0-h)$, $\phi_3'(x) = (x-x_0+h)(x-x_0) + (x-x_0)(x-x_0-h) + (x-x_0+h)(x-x_0-h)$, $\phi_3''(x) = 2(x-x_0+h) + 2(x-x_0) + 2(x-x_0-h)$.
We zien dat $\phi_3(x_0) = 0$ en $\phi_3''(x_0) = 0$ en we dit kunnen schrijven als

$$e_2''(x_0) = -2h^2 f[x_0-h, x_0, x_0+h, x_0, x_0] = -\frac{f^{(4)}(\xi)}{12}h^2.$$

Voorbeeld 6.6. ***Een 4-de orde benadering voor de eerste afgeleide:*** *We kiezen $m = 2$ en $x_0 = 0$ en vinden*

$$f'(0) = \frac{f(-2h) - 8f(-h) + 8f(h) - f(2h)}{12h},$$

met benaderingsfout

$$e_4'(0) = \frac{h^4 f^{(5)}(\xi)}{30}.$$

6.2 Numerieke Integratie

Nu willen we een integraal

$$I = \int_a^b f(x)\mathrm{d}x,$$

benaderen door in plaats van f het interpolerende polynoom p te integreren. We krijgen dan

$$I \approx \int_a^b p(x)\mathrm{d}x = \sum_{i=0}^{n} w_i f(x_i),$$

waarbij $\{x_i\}_{i=0}^n$ de steunpunten zijn en

$$w_i = \int_a^b l_i(x)\mathrm{d}x,$$

de gewichten. We noemen zo'n benadering van de integraal een *kwadratuurregel*.

Voorbeeld 6.7. ***Oppervlakte van een cirkel:*** *We kunnen de oppervlakte van een cirkel met straal 1 uitrekenen door de functie $f(x) = 2\sqrt{1-x^2}$ in integreren over $[-1,1]$. Door de integraal te benaderen, krijgen we de volgende benaderingen voor π:*

$$\pi = 21^{-2}\int_{-1}^{1}\sqrt{1^2 - x^2}\mathrm{d}x \approx \begin{cases} 4 & voor \quad n = 0 \\ \frac{8}{3} & voor \quad n = 2 \end{cases}.$$

De bijbehorende interpolanten zijn te zien in figuur 6.3

Stelling 6.2. ***Benaderingsfout:*** *De benaderingsfout*

$$e = \int_a^b f(x)\mathrm{d}x - \int_a^b p_n(x)\mathrm{d}x,$$

is gegeven door

$$e = \int_a^b f[x_0, x_1, \ldots, x_n, x]\phi_{n+1}(x)\mathrm{d}x.$$

Wanneer $\phi_{n+1}(x)$ niet van teken wisselt op het interval $[a,b]$, dan kunnen we dit uitdrukken als

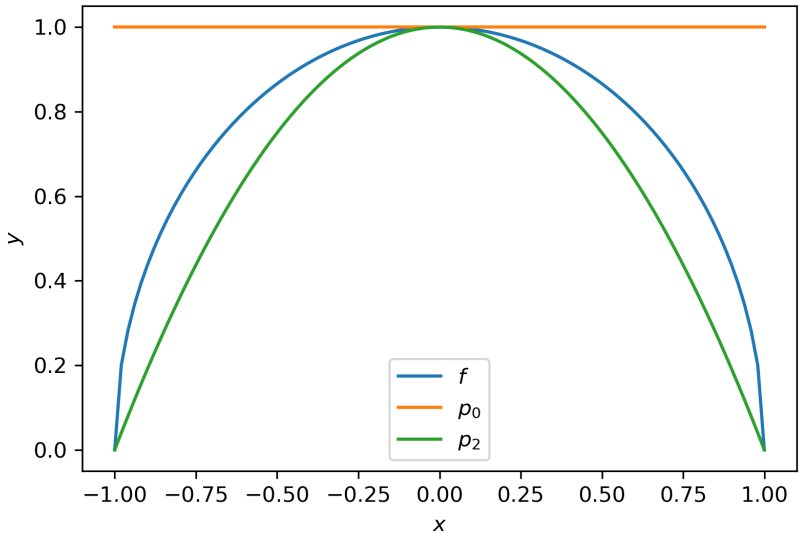
$$e = f[x_0, x_1, \ldots, x_n, \xi]\int_a^b \phi_{n+1}(x)\mathrm{d}x, \quad met \quad \xi \in [a,b].$$

Wanneer ϕ_{n+1} wel van teken wisselt op het interval $[a,b]$ dan kunnen we eerst partieel integreren.

Een aantal veelvoorkomende kwadratuurregels zijn

$$\int_a^b f(x)\mathrm{d}x \approx (b-a)f\left(\frac{a+b}{2}\right) \quad \text{(midpuntregel)},$$

$$\int_a^b f(x)\mathrm{d}x \approx \frac{(b-a)}{2}\left(f(a) + f(b)\right) \quad \text{(trapeziumregel)},$$



Figuur 6.3: Interpolanten gebruik om π te benaderen.

$$\int_a^b f(x)\mathrm{d}x \approx \frac{(b-a)}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right) \quad \text{(Simpson's regel)},$$

Dit zijn allen voorbeelden van *Newton-Cotes* kwadraturen die gebaseerd zijn op een interpolerend polynoom van graad n op een regelmatig rooster van $n+1$ steunpunten. De benaderingsfout kan ook weer worden geschreven als

$$e = C \cdot h^q f^{(r)}(\xi).$$

Omdat de gebruikte interpolant exact is voor polynomen van graad n verwachten we $r \geq n+1$. De factor h^q komt van het integreren van ϕ_{n+1}, wat een polynoom van graad $n+1$ is. We verwachten dus dat $q \geq n+2$.

Voorbeeld 6.8. ***Newton-Cotes op het interval*** $[0,1]$***:*** *De steunpunten zijn in dit geval gegeven door* $x_i = i \cdot h$ *voor* $i = 0, \ldots, n$ *met* $h = 1/n$*. De Lagrange polynomen zijn in dat geval gegeven door*

$$l_i(x) = \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)} = h^{-1} \prod_{j \neq i} \frac{(x - j \cdot h)}{(i - j)}.$$

We kwadratuurgewichten zijn dan gegeven door

$$w_i = \int_0^1 l_i(x)\mathrm{d}x.$$

De gewichten voor een aantal gevallen staan hieronder

n	w_0	w_1	w_2	w_3	w_4
1	$\frac{1}{2}$	$\frac{1}{2}$			
2	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$		
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$	
4	$\frac{7}{90}$	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$

Voorbeeld 6.9. ***Benaderingsfout van de midpuntregel:*** *Met behulp van de stelling vinden we de volgende benaderingsfout van de midpunt regel:*

$$\int_a^b f(x)\mathrm{d}x = \frac{f''(\xi)}{24}(b-a)^3.$$

We gaan als volgt te werk. De algemene uitdrukking voor de fout geeft in dit geval

$$e = \int_a^b f[m,x](x-m)\mathrm{d}x,$$

met $m = (a+b)/2$. Aangezien $(x-m)$ van teken wisselt schrijven we de integraal om door partieel te integreren:

$$e = \frac{1}{2}g(x)(x-a)(x-b)\Big|_a^b - \frac{1}{2}\int_a^b g'(x)(x-a)(x-b)\mathrm{d}x = \frac{(b-a)^3}{12}g'(\xi),$$

met $g(x) = f[m,x]$. Met behulp van de stelling voor het differentiëren van gedeelde differenties vinden we $g'(x) = f[m,x,x]$. We kunnen dit verder vereenvoudigen door de gedeelde differentie van $f[m,x,x']$ uit te schrijven:

$$f[m,x,x'] = \frac{\frac{f(x)-f(m)}{(x-m)} - \frac{f(x)-f(x')}{(x-x')}}{x'-m},$$

en $f(m)$ en $f(x')$ beide te Tayloren rond x. We vinden dan

$$f[m,x,x'] = \frac{1}{2}\frac{f''(\eta)(x-m) - f''(\eta')(x-x')}{x'-m}.$$

In de limiet dat $x' \to x$ vinden we dus dat $f[m,x,x] = f''(\eta)/2$, wat het gewenste resultaat geeft.

6.2.1 Herhaalde kwadratuur

Net als bij interpolatie en differentiatie is het soms aantrekkelijker om met stuksgewijs polynomiale interpolatie te werken. We splitsen in zo'n geval het interval $[a,b]$ op in $n+1$ deelintervallen $[a+i\cdot h, a+(i+1)\cdot h]$ voor $i = 0,\ldots,n$ en $h = (b-a)/(n+1)$ en passen onze favoriete kwadratuurregel op toe op de deelintervallen.

Voorbeeld 6.10. ***Herhaalde midpuntregel:*** *De herhaalde midpuntregel geeft de volgende kwadratuur*

$$\int_a^b f(x)\mathrm{d}x \approx h\sum_{i=0}^{n} f(a+(i+1/2)\cdot h).$$

De fout is gegeven door

$$e = \sum_{i=0}^{n}\frac{f''(\xi_i)}{24}h^3, \quad \xi_i \in [a+i\cdot h, a+(i+1)\cdot h],$$

wat we kunnen afschatten als

$$|e| \le \frac{(b-a)}{24}h^2\|f''\|_\infty.$$

Merk hier op dat we een orde h verliezen omdat we $n+1 = (b-a)/h$ fouttermen optellen.

6.2.2 Gauss-kwadratuur*

De Newton-Cotes kwadraturen gebruiken steunpunten op een regelmatig rooster. We zagen eerder al bij interpolatie dat het gunstig kan zijn om de steunpunten anders te kiezen; namelijk de nulpunten van een Chebyshevpolynoom. In het geval van integratie willen we de steunpunten zo kiezen dat de nauwkeurigheid zo groot mogelijk is. Met andere woorden: vind steunpunten zodat de kwadratuur exact is voor polynomen van een zo hoog mogelijk graad, m.

Wanneer f een polynoom van graad m is dan kunnen we fout uitdrukken als

$$e = \langle g, \phi_{n+1} \rangle,$$

waar $g(x) = f[x_0, x_1, \ldots, x_n, x]$ een polynoom van graad $m - n - 1$. De vraag is dus of we een polynoom ϕ_{n+1} van graad $n + 1$ kunnen vinden die loodrecht staat op alle polynomen van graad $m - n - 1$ voor zo hoog mogelijke graad m.

Definitie 6.1. ***Legendrepolynomen:*** *De Legendrepolynomen vormen een orthogonale basis voor de polynomen. We kunnen ze construeren door het Gramm-Schmidt procedure toe te passen op $\{1, x, x^2, \ldots\}$. We nemen het interval $[-1, 1]$ met het inproduct*

$$\langle f, g \rangle = \int_{-1}^{1} f(x)g(x)\mathrm{d}x.$$

We krijgen dan

$$P_0(x) = 1, \quad P_1(x) = x, \quad P_2(x) = (3x^2 - 1)/2,$$

en in het algemeen

$$(n + 1)P_{n+1}(x) = (2n + 1)P_n(x) - nP_{n-1}(x).$$

Wanneer we ϕ_{n+1} kiezen als het Legendrepolynoom van graad $n + 1$ dan hebben we $\langle g, \phi_{n+1} \rangle = 0$ wanneer g een polynoom is van graad $\leq n$. We vinden dan dat de kwadratuur exact is voor polynomen van graad $m = 2n + 1$.
Voor $n = 1$ krijgen we bijvoorbeeld

$$\int_a^b f(x)\mathrm{d}x \approx (b - a)f\left(\frac{a + b}{2}\right)$$

Voorbeeld 6.11. ***Benaderen van π:*** *Passen we de Gauss-kwadratuur voor $n = 1$ toe om π te benaderen dan krijgen we*

$$\pi \approx 4\sqrt{\frac{2}{3}} = 3.26\ldots$$

De bijbehorende interpolant is te zien in figuur 6.4.

6.3 Fout-schatting en -extrapolatie

We hebben een aantal methoden gezien om afgeleiden en integralen te benaderen uit gegeven functiewaarden $f(x_i)$ op een regelmatig rooster met stapgrootte h. In het algemeen willen we een grootheid, u, benaderen met $\widetilde{u}_h$. De benaderingsfout heeft de vorm

$$e(h) = \widetilde{u}_h - u = C_h \cdot h^q,$$

waarbij de orde, q, van de methode bekend is. De constante is meestal niet bekend, deze hangt immers af van de functie. Hoewel we soms een bovengrens voor de constante hebben, kan deze erg pessimistisch zijn. We willen dus graag de grootte van de fout schatten.
Het idee is om de benadering met twee verschillende stapgrootten, zeg h en $h/2$ te gebruiken. We drukken eerst $\widetilde{u}_h$ in termen van u en de fout:

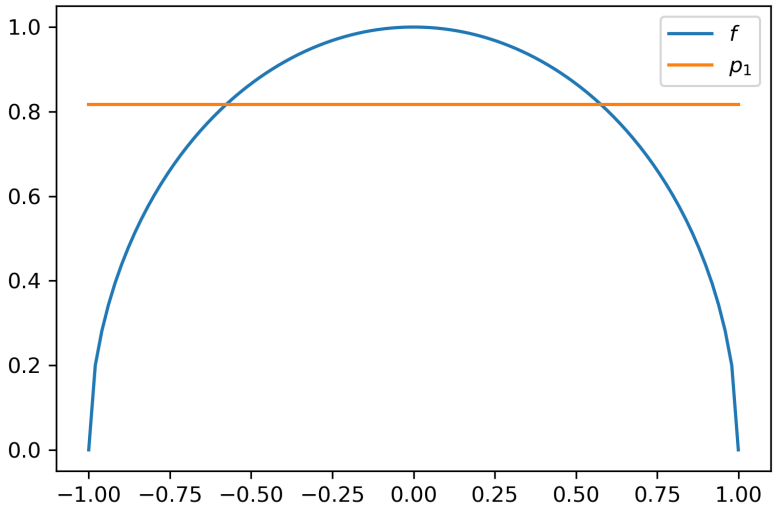
$$\widetilde{u}_h = u + C_h \cdot h^q,$$

en vervolgens, met behulp van de Taylorexpansie $C_{h'} = C_h + \mathcal{O}(h - h')$, krijgen we

$$\widetilde{u}_{h/2} = u + C_{h/2} \cdot (h/2)^q = u + 2^{-q}C_h \cdot h^q + \mathcal{O}(h^{q+1}).$$

Hieruit kunnen we nu de constante C_h schatten als:

$$C_h \approx \frac{\widetilde{u}_h - \widetilde{u}_{h/2}}{h^q(1 - 2^{-q})}.$$



Figuur 6.4: Interpolant gebruik om π te benaderen.

Benaderen van π: Passen we de herhaalde midpuntregel ($q = 2$) toe om π te schatten dan krijgen we

$$\pi \approx \widetilde{u}_{1/4} = 3.18\ldots \quad \text{voor} \quad n = 7,$$

$$\pi \approx \widetilde{u}_{1/8} = 3.15\ldots \quad \text{voor} \quad n = 15,$$

waaruit we de fout voor de eerste benadering schatten op

$$|\widetilde{u}_{1/4} - \pi| \approx (3/4)|\widetilde{u}_{1/4} - \widetilde{u}_{1/8}| = 0.036\ldots,$$

wat redelijk dicht in de buurt van de echte fout $0.0423\ldots$ ligt.

6.3.1 Richardsonextrapolatie

Als we de fout kunnen schatten, kunnen we deze dan ook gebruiken om de benadering te verbeteren? Het idee van *Richardsonextrapolatie* is om een betere benadering te construeren uit opeenvolgende benaderingen. Het simpelste geval is het gebruik van $\widetilde{u}_h$ en $\widetilde{u}_{h/2}$ om een nieuwe benadering $\widehat{u}$ te krijgen:

$$\widehat{u} = \alpha \cdot \widetilde{u}_h + \beta \cdot \widetilde{u}_{h/2}.$$

We schrijven weer $\widetilde{u}_h = u + C_h \cdot h^q$ en $\widetilde{u}_{h/2} = u + 2^{-q} C_h \cdot h^q + \mathcal{O}(h^{q+1})$, en krijgen dan

$$\widehat{u} = (\alpha + \beta) \cdot u + (\alpha + 2^{-q}\beta) \cdot C_h h^q + \mathcal{O}(h^{q+1}).$$

Neem nu $\alpha = 1/(1 - 2^q)$ en $\beta = 1 - \alpha$ en we krijgen

$$\widehat{u} = u + \mathcal{O}(h^{q+1}).$$

We hebben dus een orde gewonnen!

Voorbeeld 6.12. ***Alweer π:*** *We kunnen de benaderingen die we eerder hadden nu combineren om een betere benadering voor π te krijgen:*

$$\pi \approx -(1/3)\widetilde{u}_{1/4} + (4/3)\widetilde{u}_{1/8} = 3.14\ldots,$$

wat inderdaad een betere benadering is van π.

Een 4-punts benadering voor de eerste afgeleide: We kennen de volgende benadering voor de eerste afgeleide

$$f'(x_0) \approx \widetilde{u}_h = \frac{f(x_0+h) - f(x_0-h)}{2h}.$$

De benadering heeft orde $q = 2$ en we vinden dus een betere benadering door $\widetilde{u}_{2h}$ en $\widetilde{u}_h$ als volgt te combineren

$$\widehat{u}_h = -\frac{1}{3}\widetilde{u}_{2h} + \frac{4}{3}\widetilde{u}_h = \frac{-(1/4)f(x_0+2h) + 2f(x_0+h) - 2f(x_0-h) + (1/4)f(x_0-2h)}{3h}.$$

Opdracht 6.1. 6.4 Opdrachten

Opdracht 6.2. ***Eerste afgeleide:*** *Geef de benaderingsfout voor de terugwaartse en centrale eindige-differentiebenadering van de eerste afgeleide.*

Opdracht 6.3. ***Afrondfouten en numeriek differentiëren:*** *Onderzoek hoe gevoelig de centrale eindige-differentie benadering van de eerste afgeleide :*

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0-h)}{2h},$$

is voor afrondfouten. Je hoeft alleen rekening te houden met afrondfouten die gemaakt worden bij het evalueren van f.

- *Laat eerst zien dat de totale fout (benadering en afronding) is begrensd door*

  $$\frac{h^2 M}{6} + \frac{\epsilon}{h},$$

  waarbij $M = \max_{\xi\in[x_0-h,x_0+h]} |f'''(\xi)|$ en ϵ een bovengrens is op de afrondfout die wordt gemaakt bij het evalueren van f: $|fl(f(x)) - f(x)| \leq \epsilon$.

- *Bepaal de optimale h (in termen van M en ϵ) om een zo klein mogelijke totale fout te krijgen.*

- *Test de benadering uit op het benaderen van de afgeleide van $f(x) = \sin(x)$ op $x_0 = 0$. Is de optimale h uit de vorige stap inderdaad optimaal? Let op! Je moet dus eerst geschikte bovengrenzen M en ϵ bepalen.*

Opdracht 6.4. ***Tweede afgeleide:*** *Leid twee centrale benaderingen af voor de tweede afgeleide voor $m = 1$ en $m = 2$ (dus met 3 resp. 5 punten) en geef ook een bovengrens voor de fout.*

Opdracht 6.5. ***Herhaalde trapeziumregel:*** *Laat zien dat de benaderingsfout E in de herhaalde Trapeziumregel is begrensd door:*

$$|E| \leq \frac{\max_{x\in[a,b]} |f''(x)|}{12}(b-a)h^2.$$

Je hoeft hier geen rekening te houden met afrondfouten.

Opdracht 6.6. ***Afrondfouten en numeriek integreren:*** *De kwadratuurregels die we tot nu toe hebben bekeken zijn van vorm*

$$\int_0^1 f(x)\mathrm{d}x \approx I = \sum_{i=0}^{n} w_i f(x_i).$$

Wanneer we alleen rekening houden met afrondfouten in het evalueren van f ($fl(f(x_i)) = f(x_i) + \epsilon_i$) dan krijgen we

$$fl(I) = I + \sum_{i=0}^{n} w_i \epsilon_i.$$

1. *Geef een bovengrens voor de afrondfout, aangenomen dat $|\epsilon_i| \leq \mathcal{E}$ en $|w_i| \leq W$.*
2. *Geef een bovengrens voor de gewichten voor de herhaalde midpuntregel; hoe gedraagt de afrondfout zich als n groter wordt?*

Opdracht 6.7. ***Benaderen van integralen:*** *Benader de volgende integralen tot (minstens) 3 cijfers achter de komma nauwkeurig met behulp van de herhaalde midpuntregel:*

- $\int_0^1 \cos(x)\mathrm{d}x$
- $\int_{-1}^1 e^{-x^2}\mathrm{d}x$ *Kies h a-priori aan de hand van de bovengrens voor de fout en schat de fout ook a-posteriori met behulp van de foutschattingstechnieken de we hebben behandeld.*

Opdracht 6.8. ***Simpsonregel:*** *Laat zien dat*

$$\int_a^b f(x)\mathrm{d}x = \frac{(b-a)}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) - \frac{f^{(4)}(\xi)}{2880}(b-a)^5, \quad \xi \in [a,b].$$

Je mag hier uiteraard aannemen dat f 4 keer continue differentieerbaar is.

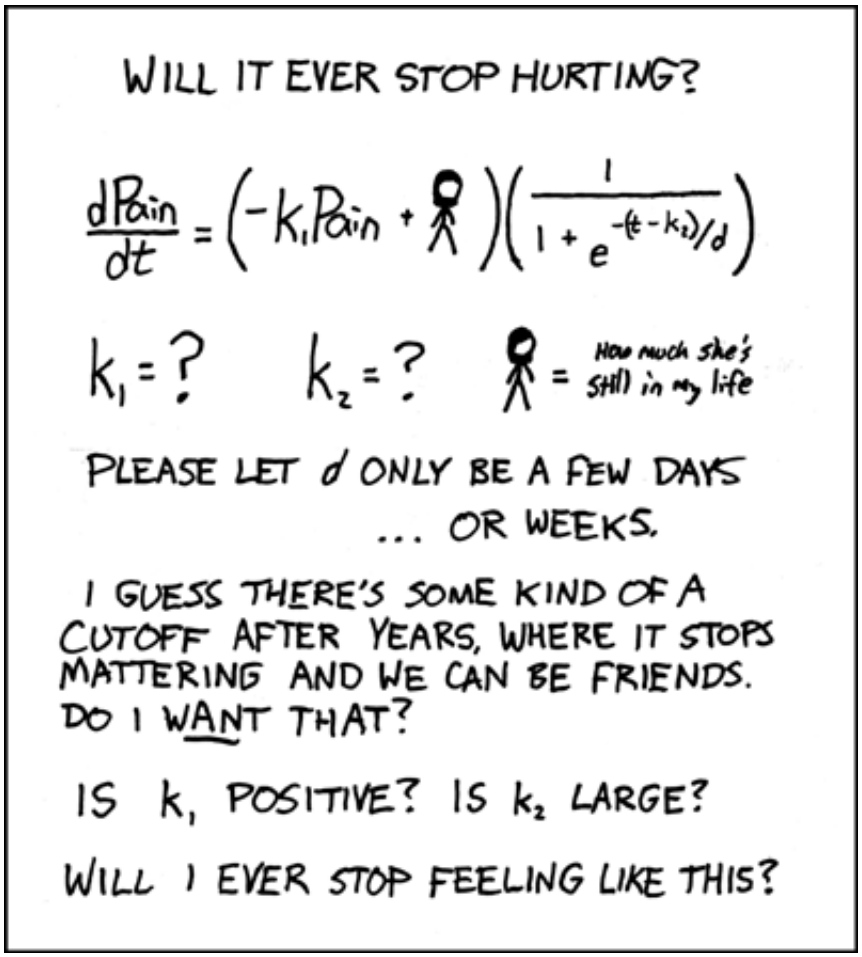
Opdracht 6.9. ***Richardsonextrapolatie:*** *Laat zien dat de benadering voor de eerste afgeleide uit het voorbeeld:*

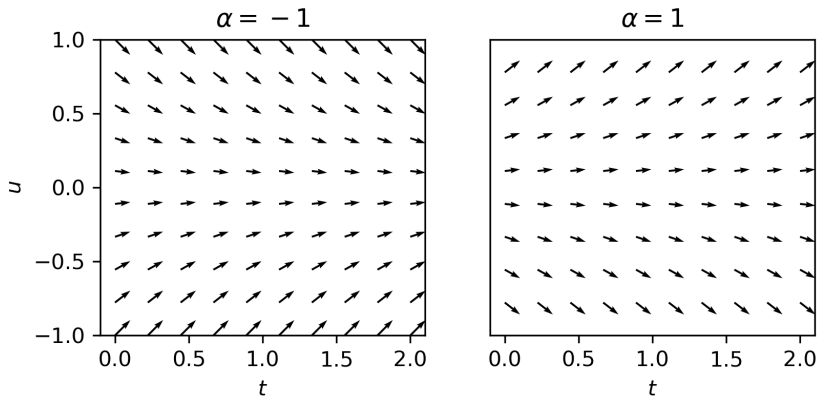
$$\frac{-(1/4)f(x_0+2h) + 2f(x_0+h) - 2f(x_0-h) + (1/4)f(x_0-2h)}{3h},$$

inderdaad van orde h^3 is.

Hoofdstuk 7

Beginwaardeproblemen





Figuur 7.1: Vectorveld van de differentiaalvergelijking $u' = \alpha u$.

In dit hoofdstuk gaan we differentiaalvergelijkingen van de vorm

$$u'(t) = f(u(t)), \quad \text{met} \quad u(0) = u_0,$$

oplossen. Hierbij zijn de functie $f : \mathbb{R}^n \to \mathbb{R}^n$ en de beginwaarde $u_0 \in \mathbb{R}^n$ gegeven en zoeken we een (vectorwaardige) functie $u : \mathbb{R} \to \mathbb{R}^n$ die voldoet aan de vergelijking en de beginwaarde. We maken ons in dit hoofdstuk geen zorgen over de existentie en uniciteit van oplossingen; er gaan er van uit dat de vergelijking goed gesteld is. In de praktijk is het vaak niet mogelijk om een expliciete uitdrukking voor de oplossing te vinden, en zullen we dus een benadering moeten gebruiken.

7.1 Analyse van differentiaalvergelijkingen

We kunnen al een idee krijgen van het gedrag van de differentiaalvergelijking door naar de functie f te kijken; deze bepaalt namelijk op ieder punt de afgeleide van u. Zo kunnen we de oplossing construeren door vanaf de beginwaarde de raaklijnen te volgen. Hieronder twee voorbeelden:

Voorbeeld 7.1. ***Exponentieel:*** *Voor $n = 1$ met $f(u) = \alpha u$ krijgen we een eenvoudige differentiaalvergelijking met oplossing $u(t) = u_0 \exp(\alpha t)$. De functie f is in dit geval de raaklijn aan de oplossing op ieder punt, en we kunnen dit visualiseren als een vectorveld in het (t, u)-vlak met richtingen $(1, f(t))$. Een voorbeeld is te zien in figuur 7.1*

Voorbeeld 7.2. ***Chemische reactie:*** *De temperatuur van een chemische reactie wordt beschreven door:*

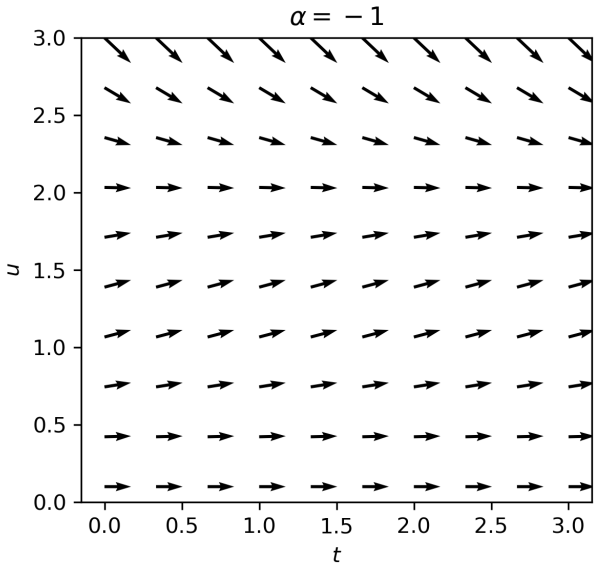
$$u'(t) = \alpha(2 - u(t)) \exp\left(\beta - \beta/u(t)\right).$$

Het vectorveld voor $\alpha = \frac{1}{4}$ en $\beta = 2$ is te zien in figuur 7.2. Het evenwichtspunt rond $u = 2$ is duidelijk te zien.

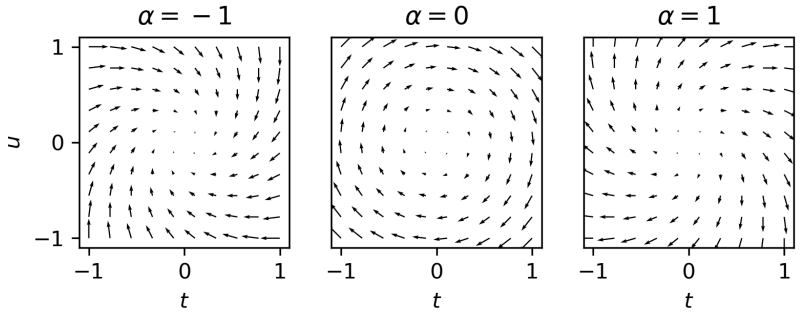
Voorbeeld 7.3. ***Een slinger:*** *We kunnen de beweging van een slinger beschrijven met een stelsel differentiaalvergelijkingen:*

$$\begin{pmatrix} u_1'(t) \\ u_2'(t) \end{pmatrix} = \begin{pmatrix} \alpha & 1 \\ -1 & \alpha \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$$

waarbij u_1 en u_2 de amplitude en snelheid van de slinger beschrijven en α de demping van de slinger is. In dit geval kunnen we het vectorveld visualiseren als functie van (u_1, u_2). Een voorbeeld is te zien in figuur 7.3.



Figuur 7.2: Vectorveld van de differentiaalvergelijking $u' = \alpha(2-u)\exp(\beta - \beta/u)$.



Figuur 7.3: Vectorveld van de differentiaalvergelijking $u' = Au$.

We kunnen het gedrag van een stelsel *lineaire* differentiaalvergelijkingen $u' = Au$, zoals die in het vorige voorbeeld, ook afleiden uit de eigenwaarden van de matrix. Als de matrix diagonaliseerbaar is, kunnen het stelsel ontkoppelen en schrijven als

$$v_i'(t) = \lambda_i v_i(t), \quad \text{voor} \quad i = 1, \ldots, n,$$

waarbij λ_i een eigenwaarde van A is en v_i de component van de oplossing in de richting van de bijbehorende eigenvector. Voor complexe eigenwaarden $\lambda_i = \alpha_i + \imath\beta_i$ hebben we een oplossing van de vorm

$$v_i(t) = v_i(0) \exp(\alpha_i t) \exp(\imath\beta_i t).$$

Als $\alpha_i > 0$, dan groeit deze component exponentieel; als $\alpha_i < 0$, dan daalt deze component exponentieel. Als $\beta_i \neq 0$ dan verwachten we oscilliaties.
Voor *niet-lineaire* stelsels $u' = f(u)$ kunnen we deze analyse nog steeds gebruiken om inzicht te krijgen in het gedrag van de oplossing ten opzichte van een referentieoplossing. Als we bijvoorbeeld een referentieoplossing u_0 hebben waarvoor $u_0' = f(u_0)$ en we de oplossing schrijven als $u = u_0 + u_1$ dan vinden we

$$u_1' = f(u) - f(u_0) \approx \frac{\partial f}{\partial u}(u_0) \cdot u_1,$$

waarbij $\frac{\partial f}{\partial u}$ de *Jacobimatrix* met partiële afgeleiden van f naar u is.

7.2 Voorwaartse en terugwaartse Euler

We kunnen de belangrijkste concepten en eigenschappen van numerieke methoden om differentiaalvergelijkingen op te lossen behandelen aan de hand van twee methoden: de voorwaartse en terugwaartse Eulermethode. Deze methoden komen tot stand door een Taylorreeks van de oplossing te beschouwen:

$$u(t+h) = u(t) + hu'(t) + \frac{h^2}{2}u''(\tau), \quad \text{(voorwaarts)}$$

$$u(t) = u(t+h) - hu'(t+h) + \frac{h^2}{2}u''(\tau'), \quad \text{(terugwaarts)}$$

Omdat de oplossing die we zoeken voldoet aan $u'(t) = f(u(t))$ kunnen we dit uitdrukken als een relatie tussen de oplossing op tijdstip t en $t+h$:

$$u(t+h) = u(t) + hf(u(t)) + \frac{h^2}{2}u''(\tau), \quad \text{(voorwaarts)}$$

$$u(t+h) = u(t) + hf(u(t+h)) - \frac{h^2}{2}u''(\tau'), \quad \text{(terugwaarts)}$$

Wanneer we de $\mathcal{O}(h^2)$ termen negeren, krijgen we de volgende twee methoden om de oplossing te benaderen op tijdstippen $t_k = k \cdot h$:

Definitie 7.1. ***Voorwaartse Euler methode:*** *Voor een gegeven stapgrootte h en beginwaarde u_0, benaderen we de oplossing van $u' = f(u)$ op tijdstippen $t_k = k \cdot h$ als volgt:*

$$\widetilde{u}_{k+1} = \widetilde{u}_k + h \cdot f(\widetilde{u}_k),$$

met $\widetilde{u}_0 = u_0$. Omdat we een expliciete *uitdrukken hebben voor $\widetilde{u}_{k+1}$ in termen van $\widetilde{u}_k$, noemen we dit een* expliciete *methode.*

Definitie 7.2. ***Terugwaartse Euler methode:*** *Voor een gegeven stapgrootte h en beginconditie u_0, benaderen we de oplossing van $u' = f(u)$ op tijdstippen $t_k = k \cdot h$ als volgt:*

$$\widetilde{u}_{k+1} = \widetilde{u}_k + h \cdot f(\widetilde{u}_{k+1}),$$

met $\widetilde{u}_0 = u_0$. We hebben hier geen expliciete uitdrukking voor $\widetilde{u}_{k+1}$ in termen van $\widetilde{u}_k$ en we noemen dit daarom een impliciete *methode. In de praktijk moeten we dus op iedere stap een vergelijking oplossen om $\widetilde{u}_{k+1}$ uit te rekenen.*

Voordat we de nauwkeurigheid van deze twee methoden gaan bestuderen, kijken we eerst naar een paar praktische voorbeelden.

Voorbeeld 7.4. ***Een kopje koffie:*** *We kunnen modelleren hoe een kopje koffie afkoelt met het volgende beginwaardeprobleem*

$$u'(t) = -\alpha u(t), \quad u(0) = u_0,$$

waarbij $\alpha > 0$ bepaalt hoe snel de koffie afkoelt. We weten natuurlijk dat de oplossing $u(t) = u_0 e^{-\alpha t}$ is; hoe goed doen de twee Eulermethoden het op deze vergelijking? In figuur 7.4 zien we de oplossing voor verschillende h. Wat opvalt is dat de fout voor $h = 2$ zich wezenlijk anders gedraagt; bij voorwaartse Euler daalt de fout niet.

Voorbeeld 7.5. ***Nogmaals de slinger:*** *We passen voorwaartse en terugwaartse Euler nu toe op de slinger die we eerder in dit hoofdstuk zagen. Voor $\alpha = 0$ zijn de eigenwaarden imaginair en verwachten we dus behoud van energie. In figuur 7.5 zien we dat de numerieke benadering opblaast met voorwaartse Euler en uitdempt met terugwaartse Euler.*

In de voorbeelden zagen we dat voorwaartse en terugwaartse Euler een vergelijkbare fout hebben, maar dat voor te grote h de fout van de voorwaartse methode kan opblazen. Hoe dit precies zit gaan we nu analyseren.

Voorbeeld 7.6. ***Foutvoortplanting voorwaartse Euler:*** *Voor de voorwaartse Euler methode toegepast op $u' = -\alpha u$ hebben we gezien dat de echte oplossing voldoet aan*

$$u((k+1)\cdot h) = u(k\cdot h) - h\alpha u(k\cdot h) + \frac{h^2}{2} f''(\tau_k),$$

terwijl de benadering $\widetilde{u}_k$ van $u(k\cdot h)$ voldoet aan

$$\widetilde{u}_{k+1} = \widetilde{u}_k - h\alpha \widetilde{u}_k.$$

We krijgen nu de volgende relatie voor de fout $e_k = u(k\cdot h) - \widetilde{u}_k$:

$$e_{k+1} = (1 - h\alpha)e_k + \frac{h^2}{2} f''(\tau_k).$$

We zien hier dus twee factoren, de truncatiefout $\frac{h^2}{2} f''(\tau_k)$ *en de* groeifactor $(1 - h\alpha)$*. Na k stappen van de methode krijgen we nu de volgende uitdrukking voor de fout (aangenomen dat $e_0 = 0$)*

$$e_k = h^2 \sum_{i=0}^{k-1} (1 - h\alpha)^{k-1-i} \epsilon_i,$$

met $\epsilon_i = \frac{f''(\tau_i)}{2}$. We kunnen hier het volgende uit afleiden

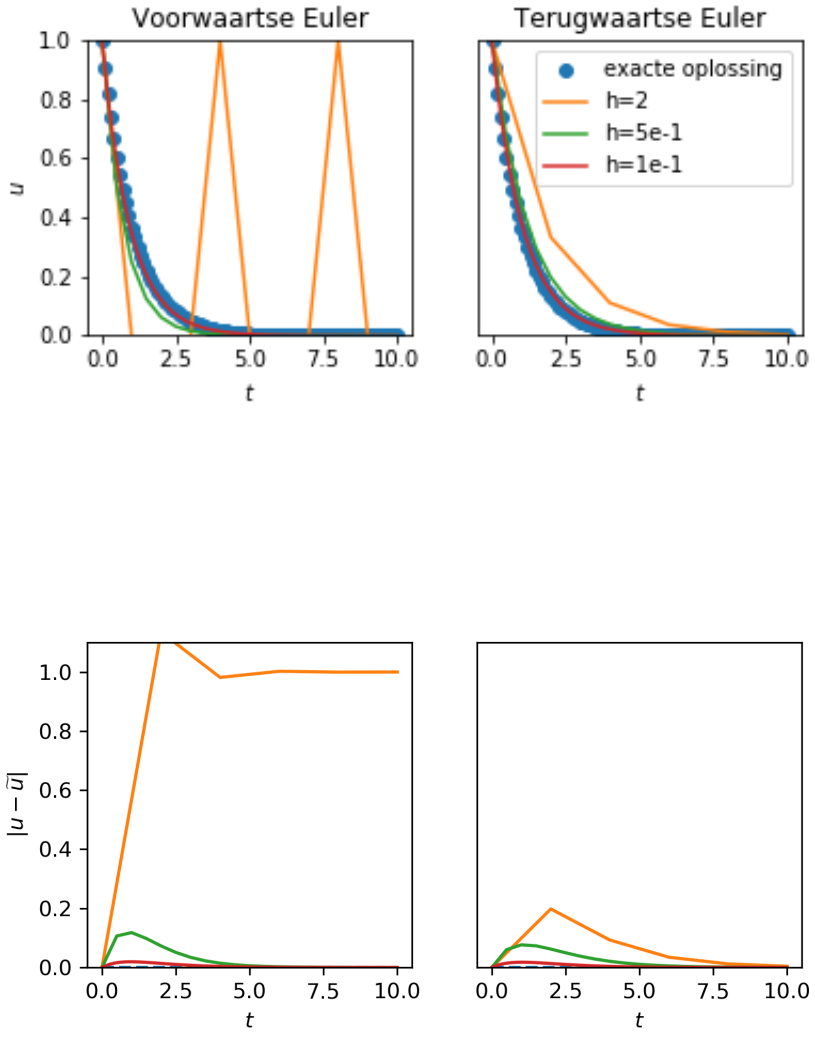
- *De fout gaat naar nul als $h \downarrow 0$*
- *Als $|1 - \alpha h| > 1$, dan kunnen we de fout afschatten op $|e_k| \leq k \cdot h^2 M |1 - \alpha h|^{k-1}$; voor vaste h stijgt de fout in het ergste geval dus exponentieel met k.*
- *Als $|1 - \alpha h| < 1$, dan kunnen we de fout afschatten op $|e_k| \leq k \cdot h^2 M$; voor vaste h stijgt de fout in het ergste geval dus lineair met k.*
- *Wanneer we de fout op een vaste eindtijd T bekijken dan hebben we $K = T/h$ stappen nodig en is de bovengrens van de fout in beide gevallen van orde h.*

Stelling 7.1. ***Nauwkeurigheid van voorwaartse Euler:*** *De voorwaartse Euler methode met stapgrootte h geeft een benadering $\widetilde{u}$ van de oplossing u van het beginwaardeprobleem $u' = f(u)$ op het interval $[0, T]$. Als $|f'(u)| \leq L$, $|u''(t)| \leq M$, dan is de fout op tijdstip $t = T$ begrensd door:*

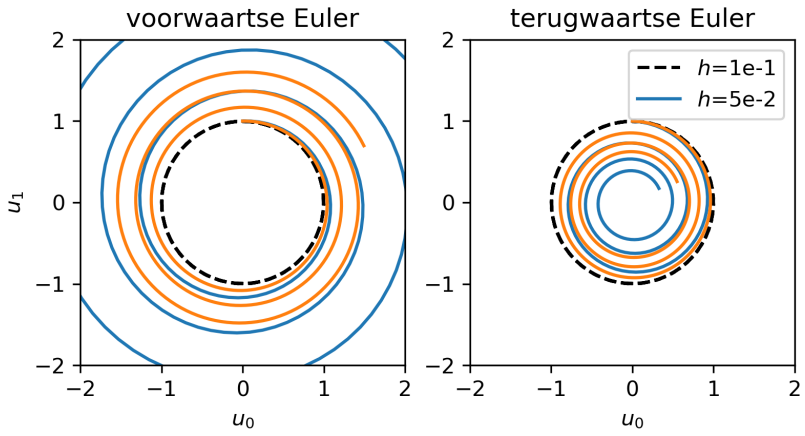
$$|u(T) - \widetilde{u}_K| \leq \frac{TMe^{TL}}{2} h,$$

met $K = T/h$.
Als $f'(u) < 0$ en $0 < h < L/2$ dan hebben we een kleinere bovengrens

$$|u(T) - \widetilde{u}_K| \leq \frac{TM}{2} h.$$



Figuur 7.4: Benadering en fout van de vergelijking $u' = -\alpha u$ met voorwaartse en terugwaartse Euler met verschillende stapgrootten.



Figuur 7.5: Benadering van een slinger met voorwaartse en terugwaartse Euler.

Stelling 7.2. ***Nauwkeurigheid van terugwaartse Euler:*** *De voorwaartse Euler methode met stapgrootte h geeft een benadering $\widetilde{u}$ van de oplossing u van het beginwaardeprobleem $u' = f(u)$ op het interval $[0, T]$. Als $|f'(u)| \leq L$, $|u''(t)| \leq M$ en $h < 1/L$, dan is de fout op tijdstip $t = T$ begrensd door:*

$$|u(T) - \widetilde{u}_K| \leq \frac{TM(1 - hL)^{-K}}{2} h,$$

met $K = T/h$.
Als $f'(u) < 0$ en $h > 0$ dan hebben we een kleinere bovengrens

$$|u(T) - \widetilde{u}_K| \leq \frac{TM}{2} h.$$

7.3 Methoden van hogere orde

We hebben nu twee methoden gezien die beide een globale nauwkeurigheid van $\mathcal{O}(h)$ hebben, maar soms is dat niet voldoende. Om nauwkeurigere methoden te ontwikkelen schrijven we de differentiaalvergelijking als een integraalvergelijking:

$$u(t + h) = u(t) + \int_t^{t+h} f(u(s))\mathrm{d}s.$$

We kunnen nu de methoden uit het vorige hoofdstuk gebruiken om de integraal te benaderen. Gebruiken we bijvoorbeeld de benadering $\int_a^b g(s)\mathrm{d}s \approx g(a)(b - a)$, dan krijgen we de voorwaartse Euler methode:

$$\widetilde{u}(t + h) = \widetilde{u}(t) + hf(\widetilde{u}(t)).$$

Wanneer we een kwadratuurregel van hogere orde gebruiken krijgen we ook een betere benadering van de oplossing van het beginwaardeprobleem.

Voorbeeld 7.7. ***De impliciete trapeziumregel:*** *Wanneer we de trapeziumregel toepassen, krijgen we de volgende methode:*

$$\widetilde{u}_{k+1} = \widetilde{u}_k + \frac{h}{2} \left(f(\widetilde{u}_k) + f(\widetilde{u}_{k+1}) \right).$$

Omdat $\widetilde{u}_{k+1}$ aan beide kanten voorkomt is dit een impliciete methode. *De truncatiefout is $\mathcal{O}(h^3)$, zodat de globale fout $\mathcal{O}(h^2)$ is.*

Voorbeeld 7.8. ***De expliciete trapeziumregel:*** *Een nadeel van de impliciete trapeziumregel is dat we op iedere stap een (niet-lineaire) vergelijking moeten oplossen. We kunnen een expliciete versie maken door $\widetilde{u}_{k+1}$ aan de rechterkant te benaderen met een voorwaartse Eulerstap:*

$$\widetilde{u}_{k+1} = \widetilde{u}_k + \frac{h}{2}\left(f(\widetilde{u}_k) + f(\widetilde{u}_k + hf(\widetilde{u}_k)\right).$$

Om te laten zien dat deze methode daadwerkelijk van orde 2 is schrijven we dit als

$$\widetilde{u}_{k+1} = \widetilde{u}_k + \frac{h}{2}\left(f(\widetilde{u}_k) + f(\widetilde{u}_{k+1} + \mathcal{O}(h^2))\right) = \widetilde{u}_k + \frac{h}{2}\left(f(\widetilde{u}_k) + f(\widetilde{u}_{k+1})\right) + \mathcal{O}(h^3).$$

De expliciete methode is een benadering van de impliciete methode met een truncatiefout van $\mathcal{O}(h^3)$. Omdat de truncatiefout van de impliciete ten opzichte van de echte oplossing ook $\mathcal{O}(h^3)$ is, is de totale trunctiefout van de expliciete methode ten opzichte van de echte oplossing ook $\mathcal{O}(h^3)$. De globale fout is dan inderdaad $\mathcal{O}(h^2)$.

7.4 Stabiliteitsanalyse

We zagen eerder al dat factoren van de vorm $(1 + h \cdot \lambda_i)$ en $(1 - h \cdot \lambda_i)^{-1}$ een rol speelde in de groei van de fout bij voorwaartse en terugwaartse Euler. Hier was λ_i een eigenwaarde van de Jacobiaan $\frac{\partial f}{\partial u}$. Kiezen we h klein genoeg zodat deze factor kleiner is dan één, dan daalt de fout exponentieel. In het algemeen kunnen we een goed idee krijgen van het gedrag van een numerieke methode door deze toe te passen op de *test vergelijking*

$$u' = \lambda u.$$

We weten in dit geval dat $\lim_{t\to\infty} |u(t)| = 0$ als $\Re(\lambda) < 0$ en $\lim_{t\to\infty} |u(t)| = \infty$ als $\Re(\lambda) > 0$. Als $\Re(\lambda) = 0$ puur imaginair is verwachten we dat $|u(t)|$ begrensd blijft.
We willen graag dat onze numerieke benadering $\widetilde{u}$ hetzelfde gedrag vertoont, en dit kunnen we analyseren door de methode toe te passen op de testvergelijking. Merk op dat we met deze analyse alleen kijken naar het kwalitatieve gedrag van de oplossing, en dus niks kunnen zeggen over de nauwkeurigheid van de methode. Hieronder twee voorbeelden.

Voorbeeld 7.9. ***Stabiliteit van voorwaartse Euler:*** *Passen we de methode toe op de testvergelijking, dan voldoet de benadering aan:*

$$\widetilde{u}_{k+1} = (1 + h \cdot \lambda)\,\widetilde{u}_k.$$

We noemen voorwaartse Euler stabiel wanneer $|1 + h \cdot \lambda| < 1$. Het stabiliteitsgebied *van de voorwaartse Euler methode is nu als volgt gedefinieerd:*

$$\mathcal{S} = \{z \in \mathbb{C} \quad | \quad |1 + z| < 1\}.$$

Figuur 7.6 laat zien hoe dit gebied in het complexe vlak eruit ziet. Voor een gegeven differentiaalvergelijking willen we h zo kiezen dat $h \cdot \lambda_i \in \mathcal{S}$ voor alle eigenwaarden λ_i van de Jacobiaan. Uiteraard krijgen we dit niet voor elkaar als de eigenwaarden in het rectherdeel van het complexe vlak liggen. Dit is niet erg omdat in dat geval de vergelijking zelf ook exponentieel groeiende oplossing heeft.

Voorbeeld 7.10. ***Stabiliteit van terugwaartse Euler:*** *Voor de terugwaartse Euler methode vinden we*

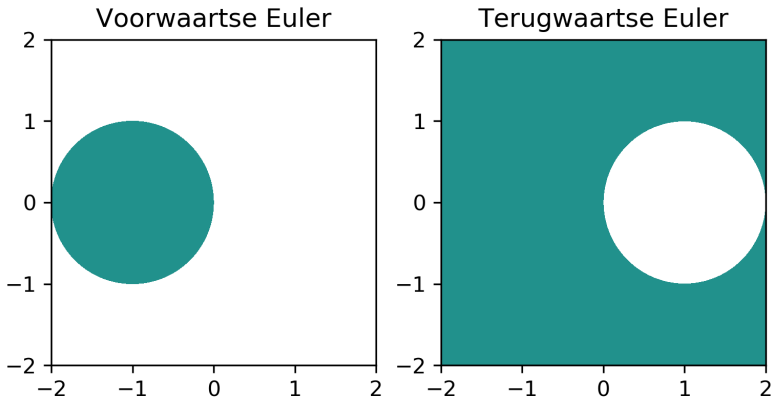
$$\widetilde{u}_{k+1} = (1 - h \cdot \lambda)^{-1}\widetilde{u}_k.$$

Het stabiliteitsgebied is te zien in figuur 7.6.

Samenvattend kunnen zeggen dat iedere methode zijn eigen stabiliteitsgebied, $\mathcal{S}$, heeft en dat we de stabiliteit van deze methode toegepast op $u' = f(u)$ kunnen bestuderen door te kijken naar de eigenwaarden, $\{\lambda_i\}$, van de Jacobiaan $\frac{\partial f}{\partial u}$:

- Als $\mathrm{Re}(\lambda_i) < 0$ dan willen we h zo kiezen dat $h \cdot \lambda_i \in \mathcal{S}$ voor alle i,
- Als $\mathrm{Re}(\lambda_i) > 0$ dan is de vergelijking zelf niet stabiel en kiezen we h klein genoeg om de fout binnen te perken te houden.

Als $f(u)$ niet lineair is, dan bekijken we de eigenwaarden $\lambda_i(t)$ van de Jacobiaan $\frac{\partial f}{\partial u}|_{u=u(t)}$ op de oplossing.



Figuur 7.6: Stabiliteitsgebieden van voorwaartse en terugwaartse Euler.

7.5 Opgaven

Opdracht 7.1. ***Vooruit of achteruit:*** *Los de differentiaalvergelijking*

$$u'(t) = \alpha(u(t) - 1), \quad u(0) = 2$$

op met de voorwaartse en terugwaartse Eulermethode voor $t \in [0, 2]$. De echte oplossing is in dit geval $1 + e^{\alpha t}$.

- *Voor $\alpha = 2$; bepaal voor beide methoden de Δt zodat de* relatieve fout *maximaal 10^{-2} is. Doe dit eerst door de bovengrens te gebruiken, en daarna door te vergelijken met de echte oplossing. Welke methode (voorwaarts of terugwaarts) is in de praktijk efficiënter in dit geval?*
- *Doe hetzelfde voor $\alpha = -10$; welke methode is nu het meests efficiënt?*

Opdracht 7.2. ***Trapeziumregel:*** *We vergelijken in deze vraag de impliciete en expliciete trapeziumregels.*

1. *Laat zien dat de truncatiefout van de impliciete trapeziumregel is gegeven door $\frac{h^3}{12}u'''(\tau)$*
2. *Laat zien dat de truncatiefout van de expliciete trapeziumregel van orde h^3 is.*
3. *Bepaal en vergelijk de stabiliteitsgebieden van beide methoden.*

Opdracht 7.3. ***Centrale Eulermethode:*** *We bekijken hier de centrale Eulermethode*

$$u_{k+1} = u_{k-1} + 2h \cdot f(u_k),$$

met u_0 en u_1 gegeven.

- *Laat zien dat de* truncatiefout *van orde h^3 is.*
- *Bepaal het stabiliteitsgebied van deze methode. (Hint: schrijf de iteratie $u_{k+1} = u_{k-1} + 2hf(u_k)$ eerst om als een stelsel $w_{k+1} = \begin{pmatrix} 2h\lambda & 1 \\ 1 & 0 \end{pmatrix} w_k$ met $w_k = (u_k, u_{k-1})$)*

Opdracht 7.4. ***Een chemische reactie:***
We gaan de volgende differentiaalvergelijking oplossen:

$$u'(t) = \alpha(2 - u(t)) \exp\left(\beta - \beta/u(t)\right),$$

met $\alpha = \frac{1}{4}$, $\beta = 2$ en $u(0) = 1$.

1. *Bepaal de functie f zodat deze vergelijking te schrijven is als $u'(t) = f(u(t))$ en bepaal de afgeleide van f.*
2. *Verwachten we groeiend of juist afnemend gedrag van de oplossing?*
3. *Kunnen we het beste een expliciete of impliciete methode gebruiken?*
4. *Pas voorwaartse Euler toe en schat de daadwerkelijke fout op $T = 1$ doormiddel van Richardsonextrapolatie.*
5. *Pas terugwaarste Euler toe (hoe zou je de niet-lineaire vergelijking oplossen?)*