FACULTEIT WETENSCHAPPEN

Vakgroep Wiskundige Analyse

DIGITALE BEELDVERWERKING MET BEHULP VAN PARTIËLE DIFFERENTIAALVERGELIJKINGEN

Arnaud DEVOS

Promotor: Prof. Dr. Marian Slodička
Copromotor: Dr. Karel Van Bockstal

Masterproef ingediend tot het behalen van de academische graad van Master in de Wiskunde, afstudeerrichting Toegepaste Wiskunde.

Academiejaar: 2016 - 2017

UNIVERSITEIT GENT

FACULTEIT WETENSCHAPPEN

Vakgroep Wiskundige Analyse

DIGITALE BEELDVERWERKING MET BEHULP VAN PARTIËLE DIFFERENTIAALVERGELIJKINGEN

Arnaud DEVOS

Promotor: Prof. Dr. Marian Slodička
Copromotor: Dr. Karel Van Bockstal

Masterproef ingediend tot het behalen van de academische graad van Master in de Wiskunde, afstudeerrichting Toegepaste Wiskunde.

Academiejaar: 2016 - 2017

UNIVERSITEIT GENT

Voorwoord

Deze masterproef is de apotheose van mijn opleiding Wiskunde. De voorbije vijf jaren waren dikwijls hard knokken, vooral in het begin dan. Maar het plezier, de kennis en de vriendschap die ik ervoor in de plaats kreeg, zijn voor mij onbetaalbaar.
Tijdens mijn opleiding ontdekte ik al snel dat ik het meest gefascineerd werd door de toegepaste en analyse vakken. Toen professor Slodička mij een jaar geleden dit thesisonderwerp voorstelde, was ik dan ook meteen enthousiast.
Bij deze wil ik graag enkele mensen bedanken die mij geholpen hebben bij het realiseren van deze masterproef.

Eerst en vooral mijn promotor professor Marian Slodička. Zoals eerder gezegd voor het voorstellen van het onderwerp, maar ook voor het begeleiden en nalezen van dit werk.
Ook mijn copromotor, doctor Karel Van Bockstal wil ik bedanken. Karel heeft mijn werk meermaals nagelezen en hij heeft mij geholpen bij de stijl en opbouw van mijn werk, alsook met diverse LaTeX-problemen.
Daarnaast wil ik mijn ouders bedanken. Ze hebben mijn thesis, als niet-wiskundigen, meerdere keren doorgenomen op zoek naar taal-en spellingsfouten. Ook ben ik hen erkentelijk voor hun vertrouwen in alle beslissingen die ik neem.
Ten slotte wil ik mijn vriendin Hanne bedanken voor haar onvoorwaardelijke steun, luisterend oor, inhoudelijke opmerkingen en het nalezen van deze masterproef.

Arnaud Devos, mei 2017

Toelating tot bruikleen

"De auteur geeft de toelating deze scriptie voor consultatie beschikbaar te stellen en delen van de scriptie te kopiëren voor persoonlijk gebruik.
Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze scriptie."

Arnaud Devos, mei 2017

Inhoudsopgave

Inleiding

Wanneer digitale afbeeldingen worden geanalyseerd, treden er verscheidene problemen op. De afbeeldingen kunnen een zekere ruis vertonen of de kwaliteit van de afbeelding kan simpelweg zeer slecht zijn. Aan de hand van partiële differentiaalvergelijkingen kunnen dergelijke afbeeldingen onder andere gerestaureerd, verbeterd en verscherpt worden.

De nood aan digitale beeldverwerking dringt zich vooral op door de vele toepassingen. Typische 2D voorbeelden betreffen beschadigde foto's, oude archiefdocumenten, satellietbeelden, camerabeelden of andere digitale beelden die van een slechte kwaliteit zijn. Deze beelden zijn initieel zwart-wit afbeeldingen. Op deze beelden is vaak ruis aanwezig en bijgevolg dienen ze te worden gerestaureerd of verscherpt. De meest gebruikte toepassing is wellicht te vinden in de medische wereld. Voor MRI, Echografie en CT scan afbeeldingen, kan aanwezige ruis het verschil maken tussen een vroege vaststelling van een medisch probleem of een verkeerde diagnose.

De klassieke manier om een zwart-wit afbeelding gladder[1] te maken, is door ze te convoluteren met een Greense functie G_σ, i. e.

$$G_\sigma(\boldsymbol{x}) = \frac{1}{4\pi\sigma} e^{-|\boldsymbol{x}|^2/4\sigma}.$$

Het is bekend dat G_σ een fundamentele oplossing is van de lineaire warmtevergelijking (diffusievergelijking). Hierdoor is het mogelijk om, in plaats van deze klassieke bewerking, de lineaire warmtevergelijking op te lossen voor een correcte tijd $t = \sigma$. De beginconditie is de verstoorde afbeelding. Deze observatie werd voor het eerst gedaan begin de jaren tachtig [21, 52]. Deze methode reduceert niet enkel ruis, maar vervaagt ook belangrijke kenmerken zoals randen. Lineaire diffusie neemt geen a priori informatie in zich op van structuren die het waard zijn om te worden behouden (of zelfs verscherpen). Het is duidelijk dat zo'n eigenschap net belangrijk is voor de reeds opgesomde toepassingen. Een manier om deze tekortkoming op te lossen is door over te gaan naar niet-lineaire diffusiemodellen. Het doel van deze thesis is om dergelijk model te bestuderen.

Inhoud

In Hoofdstuk 2 introduceren we de beroemde niet-lineaire diffusievergelijking in digitale beeldverwerking, de Perona-Malik vergelijking. Dit hoofdstuk zorgt ervoor dat de lezer vertrouwd

[1] Engels: *to smooth an image.* De afbeelding wordt zachter/waziger zodat de ruis verdwijnt.

geraakt met het model en inzicht krijgt in zijn determinanten (parameters, functies). In hoofdstuk 3 wordt de Perona-Malik vergelijking aangepast, om ze daarna op een theoretische manier te bestuderen inzake uniciteit en existentie. Deze analyse is vrij ingewikkeld voor wie niet vertrouwd is met reële & complexe analyse, lineair genormeerde ruimten, Hilbert ruimten, de Lebesgue integraal en partiële differentiaalvergelijkingen. De voornaamste lemma's en stellingen die we in Hoofdstuk 3 gebruiken, zijn opgesomd in Appendix B. Lezers uit een opleiding bachelor Wiskunde hebben zeker voldoende basiskennis.
Een grote component van deze thesis zit bevat in het programmeren van de numerieke schema's in MATLAB. De uitwerkingen hiervan zijn terug te vinden in Appendix C. In Hoofdstuk 2 en 3 geven we bijkomende uitleg over deze implementaties. In het vierde en laatste hoofdstuk vergelijken we de Perona-Malik vergelijking met de aangepaste vergelijking uit Hoofdstuk 3. We beginnen deze scriptie met een inleiding over digitale afbeeldingen.

Bronnen en eigen inbreng

De inhoud van Hoofdstuk 1 is afkomstig uit Burger en Burge [5], Gonzalez et al. [15] en Gonzalez en Woods [14]. De inhoud van Hoofdstuk 2 is gebaseerd op tal van bronnen. De voornaamste bronnen zijn Perona en Malik [31] en Mikula [27]. Op suggestie van de promotoren is een impliciet numeriek schema opgesteld, in plaats van een expliciet zoals in [31]. Omdat de uitwerking hiervan telkens beknopt is in de literatuur, hebben we ervoor gekozen om de numerieke methode stap voor stap uit te leggen. Voor de implementaties in MATLAB van de schema's konden we ons baseren op de expliciete schema's uit Phillpot [32], Kovesi [22] en een impliciet schema uit Schlasche [35]. Uiteindelijk hebben we ervoor gekozen om een volledig eigen implementatie te schrijven. Deze implementatie maakt efficiënter gebruik van MATLAB. De parameters studie op het einde van Hoofdstuk 2 werd voorgesteld door de promotoren en is verricht met de zelfgeschreven MATLAB code. Uit eigen interesse is geprobeerd om aan te tonen dat het semi-impliciete schema in tijd goed gedefinieerd is, op basis van het artikel Kačur en Mikula [18]. Dit bleek helaas niet haalbaar en daarom hebben we een geregulariseerde versie van de Perona-Malik vergelijking theoretisch bestudeerd. Voor het bewijzen van de uniciteit hebben we ons gebaseerd op Catté et al. [8]. Voor het bewijzen van de existentie hebben we ons gebaseerd op [18]. Het compactheid-argument dat in [18] gebruikt wordt, hebben we vervangen door een argument waar we wel mee vertrouwd zijn. Het is namelijk gebaseerd op de cursus Slodička [37] en het doctoraat Van Bockstal [44]. Lemma 3.8 is bewezen met hulp van de promotoren. Hoofdstuk 4 is toegevoegd op voorstel van de promotoren. De aanpak en uitwerking hiervan is zelfstandig gebeurd. De enige bron in dit hoofdstuk is de definitie van PSNR uit Wikipedia [51], Mathworks [26]. De gebruikte afbeeldingen doorheen deze masterproef zijn eigen afbeeldingen tenzij de bron vermeld wordt.

1 Digitale afbeeldingen

"A picture is worth a thousand words."

Frederick R. Bernard

Digitale beeldverwerking wordt veelal omschreven als het gebruik van algoritmen om digitale afbeeldingen te creëren, te verwerken en weer te geven. Het proces dat een fysisch tafereel omzet naar een digitale afbeelding houdt op zich al heel wat wiskunde in. Het resultaat van dit proces is uiteindelijk niets meer dan een eenvoudige matrix. Het is deze matrix die, met de juiste methoden, gemanipuleerd kan worden op elke mogelijke manier.

Er is een hele waslijst aan toepassingen en voordelen van digitaliseren. Om er één uit te pikken: de toegankelijkheid van foto's en documenten. De Universiteitsbibliotheek Gent heeft in samenwerking met Google reeds honderdduizenden werken online gebracht door deze werken te digitaliseren.[1] Dit project, *Google Library Project*, ging van start eind 2007. Universiteit Gent was echter niet de enige universiteit, andere deelnemers aan dit project zijn o.a. Harvard University, Stanford University en Oxford University. Eenmaal de boeken gedigitaliseerd zijn, is het eenvoudig om ze voor iedereen online beschikbaar te stellen.

In dit hoofdstuk focussen we op het gebruik van MATLAB voor digitale beeldverwerking. Alvorens we hier toe komen, leggen we formeel uit wat een digitale afbeelding is (hiervoor hebben we ons gebaseerd op [5, 40]). De kennis van de pixel-structuur van de digitale afbeelding zal van belang zijn voor het volgende hoofdstuk.

1.1 Beeldverwerving

Beeldverwerving[2] is het proces dat een fotografisch beeld creëert. Normaliter is de afbeelding waarmee gewerkt wordt in beeldverwerking al in digitale vorm, daarom worden enkel de belangrijkste stadia van dit proces vermeld. De methoden komen in essentie neer op variaties van de klassieke optische camera. Voor een gedetailleerde uitleg wordt verwezen naar de literatuur, zie bijvoorbeeld [47][Hoofdstuk 1].

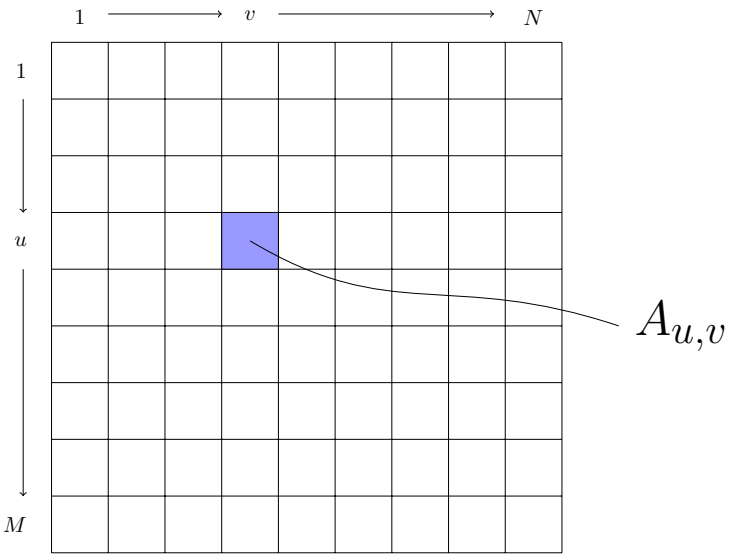
[1] Op deze werken is geen auteursrecht meer van toepassing [42].

[2] In het Engels *image acquisition* of *digital imaging* genoemd.

De projectie van het fysisch beeld op het twee-dimensionale beeldvlak van de camera resulteert in een *afbeelding*. Deze afbeelding kan gedefinieerd worden als een twee-dimensionale functie $a(x, y)$, waarbij x en y de ruimtelijke coördinaten voorstellen. De waarde van a geëvalueerd in een koppel (x, y) wordt de intensiteit van de afbeelding in dat punt genoemd. Om uiteindelijk tot een *digitale afbeelding* te komen, zijn er nog twee stappen nodig. Ten eerste, de ruimtelijke coördinaten x, y dienen te worden gediscretiseerd (*sampling*). Ten tweede, de intensiteit waarden moeten worden omgezet naar gehele waarden binnen een eindig interval zodat ze kunnen worden voorgesteld door digitale getallen (*quantization*).

1.2 Voorstelling van een digitaal beeld

Zij $a(x, y)$ een continue functie van twee continue variabelen x en y. We converteren deze functie naar een digitale afbeelding zoals uitgelegd in vorige sectie. Veronderstel dat we de continue afbeelding verdelen in een $2D$ matrix, A. We veronderstellen steeds dat we te maken hebben met rechthoekige afbeeldingen.[1] We noemen het aantal kolommen N de breedte en het aantal rijen M de hoogte van de afbeelding A. De grootte van de afbeelding wordt dan bepaald door het product MN. Om te weten welke positie op de afbeelding overeenkomt met welk element van de afbeelding, moet een coördinatensysteem ingevoerd worden. De meest gebruikte methode is om pixel indices te gebruiken. De afbeelding wordt beschouwd als een rooster van discrete elementen, geordend van boven naar onder en van links naar rechts, zoals geïllustreerd in Figuur 1.1. Dit komt in essentie overeen met matrix indices. Voor de pixel indices (u, v) geldt $u = 1, \ldots, M$ en $v = 1, \ldots, N$.



Figuur 1.1: Pixel indices. Elke pixel komt overeen met een rij- en kolomindex (u, v).

[1] Dit is een redelijk veilige assumptie al bestaan er uitzonderingen.

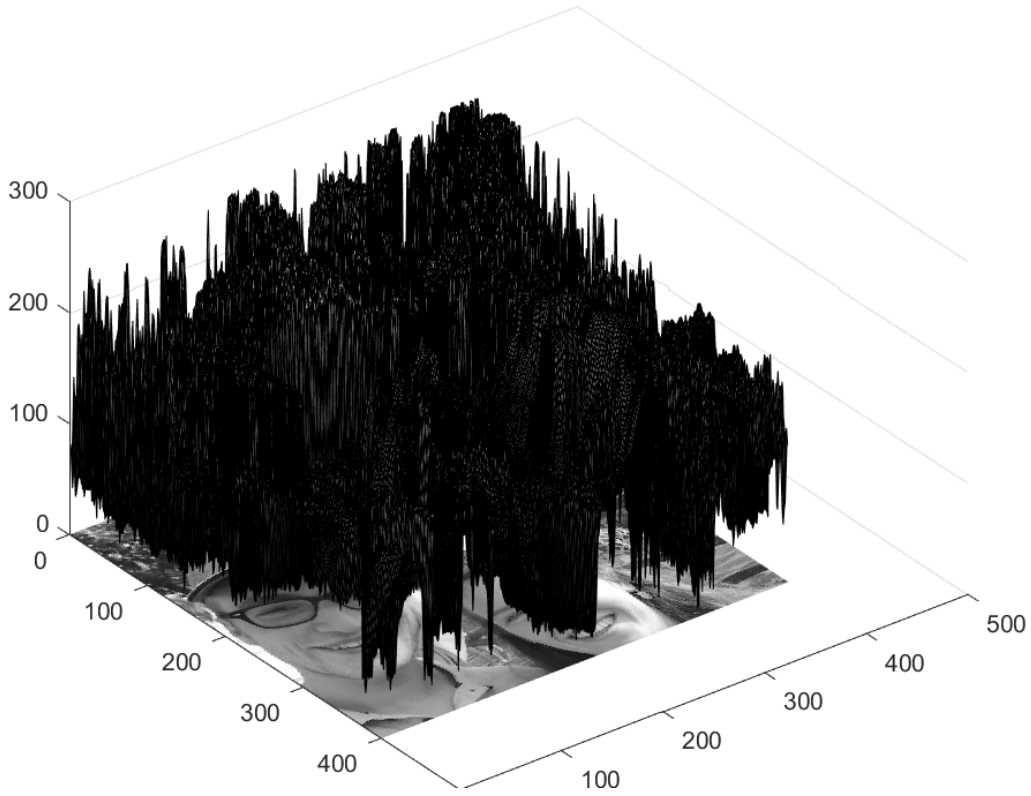
$$\begin{bmatrix} 81 & 62 & 71 & 97 & \dots & \dots & \dots & 247 \\ 64 & 42 & 55 & 98 & \dots & \dots & \dots & \vdots \\ 62 & 65 & 63 & 69 & \dots & \dots & \dots & \vdots \\ 82 & 110 & 109 & 91 & \ddots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \ddots & \ddots & \ddots & \dots & \vdots \\ \vdots & \dots & \dots & \dots & \ddots & \ddots & 84 & 97 \\ \vdots & \dots & \dots & \dots & \dots & \ddots & \ddots & 121 \\ 174 & \dots & \dots & \dots & \dots & \dots & 139 & 124 \end{bmatrix}$$

Figuur 1.2: Links: Afbeelding als een $2D$ array van gehele getallen. Rechts: Afbeelding weergegeven zoals op een monitor. Hierbij is $M = N = 400$.

Een mogelijke representatie van een digitale afbeelding zijn de waarden van $A_{u,v}$ in een matrix, zoals in Figuur 1.2, links. Rechts in deze figuur wordt getoond hoe deze representatie verschijnt op een scherm. Een monitor converteert de waarden $A_{u,v}$ naar een zekere kleur. Een derde mogelijke voorstelling van een digitale afbeelding wordt gegeven in Figuur 1.3. Hierbij wordt de afbeelding als discrete functie voorgesteld. Deze functie beschrijft een oppervlak. Door de afbeelding terug als een functie te zien, moet er wel gelet worden op de volgorde van de coördinaten. De x-coördinaat slaat dan namelijk op de kolom-index, en de y-coördinaat op de rij-index. Dit is dus net het omgekeerde als de volgorde van de pixelindices. Tevens is het coördinatensysteem gespiegeld ten opzichte van de $x - as$. Dit betekent dat de y-coördinaat loopt van boven naar onder en de oorsprong ligt in de linkerbovenhoek. Deze twee conventies zijn niet altijd even praktisch en kunnen soms verwarrend zijn in de context van meetkundige transformaties, toch is dit de meest gebruikte conventie omdat ze overeenkomt met de matrix structuur van een afbeelding. In het volgend hoofdstuk zullen we hier uitgebreid op terugkomen.

De presentatie links in Figuur 1.2 is de meest gebruikte om aan beeldverwerking te doen. Algemeen noteren we de voorstelling van een $M \times N$ afbeelding als

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,N} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M,1} & A_{M,2} & \cdots & A_{M,N} \end{bmatrix}. \tag{1.1}$$



Figuur 1.3: Afbeelding als de grafiek van een oppervlakte.

De elementen van de matrix (1.1) noemen we *pixels*.[1] Deze pixels zijn praktisch altijd binaire woorden van lengte k, zodat een pixels 2^k verschillende mogelijke waarden kan hebben. De waarde k wordt ook wel de bit-diepte van de afbeelding genoemd. Het exacte bit-niveau van een individuele intensiteit zal afhangen van de soort van afbeelding; binair, zwart-wit of RGB[2] (kleur). We leggen nu zorgvuldig de verschillen uit tussen deze soorten afbeeldingen.

Zwart-wit afbeeldingen

Zwart-wit afbeeldingen bestaan uit zwart, wit en tussenliggende grijstinten. Dergelijke afbeeldingen bestaan dus uit meer dan twee kleuren. Deze worden ook wel *grayscale* of *intensity* afbeeldingen genoemd. Een pixel uit een zwart-wit afbeelding is één getal dat de intensiteit weergeeft van de afbeelding in dat punt. Typisch worden getallen gebruikt in het bereik $0, \ldots, 2^k - 1$. Een zwart-wit afbeelding gebruikt vaak $k = 8$ bits per pixel zodat de intensiteit waarden aanneemt in het bereik $0, \ldots, 255$, waarbij de waarde 0 de minimum intensiteit voorstelt (zwart) en 255 de maximum intensiteit (wit).

[1] *Pixelintensiteiten* zou hier een betere benaming zijn. Soms bedoelen we met een *pixel* het vierkant dat het voorstelt in de afbeelding. De intensiteit van de pixel is dan de waarde of kleur in dit vierkant. Als er geen verwarring mogelijk is tussen beide, noemen we ook deze intensiteiten pixels.

[2] Rood, groen en blauw afkomstig van het Engels *red, green and blue*.

Binaire afbeeldingen

De pixels bij binaire afbeeldingen kunnen slechts twee waarden aannemen, 0 (zwart) of 1 (wit). Binaire afbeeldingen zijn bijgevolg gewoon een speciaal geval van zwart-wit afbeeldingen. Ze worden vaak gebruikt voor het voorstellen van lijnen, het archiveren van documenten en bij elektronisch printen.

Kleurafbeeldingen

De meeste kleurafbeeldingen zijn gebaseerd op de primaire kleuren rood, groen en blauw (RGB), waarbij meestal 8 bits voorzien zijn voor elk van deze drie kleuren. In dit geval vereist een pixel $3 \times 8 = 24$ bits. Het bereik van een individuele kleur component is de verzameling van gehele getallen in $[0, 255]$. De kleur van de pixel wordt bepaald door de combinatie van de rode, groene en blauwe intensiteit.

Om een RGB afbeelding om te zetten naar een zwart-wit afbeelding worden de drie kleurcomponenten van de RGB afbeeldingen gelijk gemaakt. Stel dat de waarden van de kleur afbeelding in een pixel (r, g, b) zijn, dan is een mogelijkheid om $(r + g + b)/3$ als waarde voor die pixel te kiezen.

1.3 Beeldcompressie

Het opslaan van een afbeelding op de computer vergt een zekere opslagcapaciteit. Zo zal een binaire afbeelding van grootte 640×480 pixels, 38400 bytes[1] vereisen om ze op te slaan. Een binaire afbeelding voorziet per pixel 1 bit, in dit voorbeeld zijn er 307200 pixels, die evenveel bits vereisen of dus 38400 bytes.

Een RGB afbeelding zal meer opslag vragen. Zo zal een RGB afbeelding van 640×480 pixels, gebruik makend van 8 bits per kleur kanaal, 921 600 ($= 640 \times 480 \times 24/8$) bytes opslagcapaciteit vereisen.

Beeldcompressie is een bewerking die ervoor zorgt deze opslagcapaciteit te beperken. De manier waarop een originele afbeelding gecomprimeerd wordt, bepaalt het bestandsformaat van de afbeelding. Het bekendste voorbeeld van een bestandsformaat is wellicht de JPEG[2]. Beeldcompressie houdt op zichzelf al heel wat complexe, mooie wiskunde in. Dit wordt hier niet verder uitgediept, het onderwerp is op zichzelf al een thesis waard.

Met enige Belgische trots verdient Ingrid Daubechies hier toch nog een vermelding. Deze Belgische natuurkundige en wiskundige, geboren in 1954, wordt omschreven als “Wereldautoriteit op het gebied van wavelets die gebruikt worden voor beeldcompressie” [51]. Daubechies haar studie rond wavelets is van groot belang geweest voor het JPEG-2000 bestandsformaat.

[1] Ter herinnering, 1 byte is 8 bits.

[2] Dit formaat werd ontwikkeld door de Joint Photographic Experts Group, `www.jpeg.org`.

1.4 Afbeeldingen in Matlab

MATLAB is als wiskundige softwareomgeving voorzien van een uitgebreide verzameling van functies voor het bewerken van *arrays*. Deze ingebouwde basisstructuur is gedefinieerd als een geordende verzameling van reële of complexe elementen. Deze structuur vormt een natuurlijke representatie voor afbeeldingen. De meeste afbeeldingen worden in MATLAB voorgesteld als twee-dimensionale arrays (lees: matrices), waarin elk element van de matrix overeenstemt met één enkele pixel van de afbeelding. Bijvoorbeeld, een afbeelding die bestaat uit 200 rijen en 300 kolommen van verschillende gekleurde punten wordt opgeslagen als een 200 bij 300 matrix. Deze conventies maken het werken met grafische bestanden nagenoeg zo eenvoudig als het werken met een matrix data type object.

Algemeen ziet een digitale afbeelding er in MATLAB als volgt uit:

$$\texttt{I} = \begin{bmatrix} \texttt{I(1,1)} & \texttt{I(1,2)} & \cdots & \texttt{I(1,N)} \\ \texttt{I(2,1)} & \texttt{I(2,2)} & \cdots & \texttt{I(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ \texttt{I(M,1)} & \texttt{I(M,2)} & \cdots & \texttt{I(M,N)} \end{bmatrix}.$$

Merk dus op dat deze representatie identiek is als (1.1). De notatie `I(p,q)` is het element dat zich bevindt op rij p en kolom q. Om bijvoorbeeld de pixel weer te geven op de tweede rij van de vijftiende kolom van een afbeelding I:

```
>> I(2,15)
```

MATLAB heeft bovendien een Image Processing Toolbox, die de mogelijkheden om aan beeldverwerking te doen nog vergroot. Voor het resterend gedeelte van dit hoofdstuk gaan we dieper in op de functionaliteiten van MATLAB om aan beeldverwerking te doen.

1.4.1 Data klassen

MATLAB ondersteunt enkele verschillende soorten klassen om de pixels van een afbeelding weer te geven:

Klasse	Definitie	Aantal bytes per element
`double`	Dubbele precisie	8
`uint8`	Tekenloze (“+”) 8-bit integer	1
`uint16`	Tekenloze 16-bit integer	2
`int8`	8-bit integer	1
`int16`	16-bit integer	2
`logical`	Waarden zijn 0 of 1	1

De meest gebruikte arrays zijn van de klasse `double`. De data in deze arrays zijn opgeslagen als een (64-bit) vlottende punt-voorstelling. Deze nemen (bij benadering) waarden aan in het bereik $\pm 10^{308}$. Om de geheugencapaciteit te beperken, worden afbeeldingen vaak opgeslagen als `uint8` of `uint16` klassen. De data in zo'n arrays worden opgeslagen als 8-bit of 16-bit positieve gehele getallen. Zo ligt een element uit de klasse `uint8` in het interval $[0, 255]$. De klassen `int8` en `int16` werken analoog, maar laten negatieve getallen toe. Zo zal een element uit de klasse `int8` in het interval $[-128, 127]$ liggen. Deze klassen worden nauwelijks gebruikt voor beeldverwerking. De klasse `logical` wordt dan weer vaak gebruikt voor het voorstellen van binaire afbeeldingen.

1.4.2 Afbeeldingen inlezen, weergeven en wegschrijven

Afbeeldingen worden in MATLAB als volgt ingelezen:

```
>> I= imread('Blastoise.jpg');
```

Het commando `imread` geeft in dit voorbeeld een $3456 \times 4608 \times 3$ array van de klasse `uint8` terug. Om de afbeelding vervolgens weer te geven volstaat het commando `imshow`:

```
>> imshow(I)
>> iptsetpref('ImshowBorder','loose');
```

Het tweede commando zegt dat figuur een rand moet weergeven. Standaard staat dit op *tight'* en wordt er geen rand weergegeven door `imshow`. Meestal zullen we dit commando achterwege laten. Hier wordt het echter gedaan om nog eens duidelijk het coördinatensysteem te zien dat we gebruiken.

De functie `size` geeft de rij- en kolomdimensie weer van de afbeelding:

```
>> [M,N]= size(I);
ans =

        3456        4608           3
```

De laatste output 3, slaat op de dimensie. Aangezien het hier om een RGB afbeelding gaat, zijn er drie twee-dimensionale arrays. Bij een zwart-wit afbeelding zal dit niet weergegeven worden als output omdat het dan om één enkele array gaat. De `whos` functie geeft bijkomstige informatie weer over de matrix. Bijvoorbeeld

```
>> whos I
```

geeft

```
      Name          Size                    Bytes  Class    Attributes

      C          3456x4608x3             47775744  uint8
```

Afbeeldingen worden naar de huidige directory geschreven met behulp van de functie `imwrite`:

```
>> imwrite(I,'Blastoise_copy.jpg');
```

Soms kan het handig zijn om de afbeelding op te slaan zonder dat er compressie zal plaats vinden (om details van de afbeelding te behouden). Dit kan gedaan worden door de afbeelding op te slaan als een bitmap (bmp). Hiervoor gebruiken we het commando

```
print('Blastoise_copy',-dtiffn',-r0');
```

Dit schrijft de huidige afbeelding, te zien in het venster via `imshow`, naar de huidige directory als een TIFF bestand. De resolutie van de afbeelding is die van het computerscherm. TIFF (*Tagged Image File Format*) is een bestandsformaat dat vaak wordt gebruikt voor het opslaan van pixelbestanden omdat dit geen kwaliteitsverlies geeft.

1.4.3 Types van afbeeldingen

Een zwart-wit afbeelding is een data matrix I, wiens elementen grijstinten voorstellen. Een zwart-wit afbeelding wordt voorgesteld door één enkele matrix. De matrix kan van de klasse `double`, `uint8` of `uint16` zijn.

Binaire afbeeldingen hebben een zeer specifieke betekenis in MATLAB. Een binaire afbeelding is met name een array van de klasse logical. Dus een array die bestaat uit nullen en enen maar die van de klasse uint8 is, wordt in MATLAB niet beschouwd als een binaire afbeelding.

Een RGB afbeelding wordt opgeslagen als een m-bij-n-bij 3 data array die de rode, groene en blauwe kleurcomponenten definieert voor elke individuele pixel. Zo'n array kan opnieuw van de klasse `double`, `uint8` of `uint16` zijn. In een RGB array van de klasse `double` is elke kleur component een waarde tussen 0 en 1. Een pixel met intensiteit (0,0,0) wordt als zwart weergegeven terwijl een pixel met intensiteit (1,1,1) als wit wordt weergegeven. De drie kleur componenten worden opgeslagen in de derde dimensie van de array. Bijvoorbeeld, de rode, groene en blauwe kleur component van de pixel (10,5) worden opgeslagen in respectievelijk RGB(10,5,1), RGB(10,5,2) en RGB(10,5,3).

1.4.4 Ruis toevoegen aan een afbeelding

De Image Processing Toolbox van MATLAB voorziet een functie die ruis kan toevoegen aan een afbeelding. `J = imnoise(I,type)` voegt ruis toe aan een gegeven afbeelding `I`. De parameter `type` geeft aan welke soort ruis er wordt toegevoegd. Er zijn vijf verschillende soorten types ruis:

Type	**beschrijving**
'gaussian'	Gaussische witte ruis met constant gemiddelde M en variantie V
'localvar'	Gaussische witte ruis met constant gemiddelde 0 en een lokale variantie V
'poisson'	Poisson ruis
'salt & pepper'	Slechts een beperkt aantal beschadigde pixels
'speckle'	Uniforme ruis met gemiddelde 0 en variantie V

Voor een gedetailleerde beschrijving wordt verwezen naar [26]. Als voorbeeld tonen we hoe Gaussische witte ruis met gemiddelde 0 en een variantie die overal 1/10 is, wordt toegevoegd aan een afbeelding I.

```
>> v=ones(size(I))/10;
>> J=imnoise(I,'localvar',v);
>> imshow(J)
```

Op deze manier kan kunstmatig ruis worden toegevoegd aan een afbeelding. Hierdoor ontstaat een nieuwe afbeelding J. Door een wiskundige methode toe te passen op de afbeelding J kan de ruis opnieuw, zij het gedeeltelijk, worden weggenomen. Het resultaat hiervan kan dan vergeleken worden met de oorspronkelijke afbeelding I.

2 De Perona-Malik vergelijking

"You sort of start think anything's possible if you've got enough nerve."
J.K. Rowling, Harry Potter and the Order of the Phoenix.

De Perona-Malik vergelijking is een niet-lineaire diffusievergelijking die in beeldverwerking wordt gebruikt om ruis weg te filteren en randen te verscherpen. Perona en Malik waren de eersten die een niet-lineaire diffusievergelijking toepasten op afbeeldingen. Hierdoor kunnen zij gerust de pioniers genoemd worden op het gebied van beeldverwerking via partiële differentiaalvergelijkingen.

De originele vergelijking werd in [31] als volgt geïntroduceerd

$$u_t = \nabla \cdot (g(|\nabla u|)\nabla u), \tag{2.1}$$

met

$$u : [0,T] \times \Omega \to \mathbb{R} : (t, \mathbf{x}) \mapsto u(t, \mathbf{x})$$

de afbeelding als functie van de tijd en de plaats,

$$g : \mathbb{R} \to \mathbb{R}^+ : x \mapsto g(x)$$

de diffusiecoëfficiënt als functie van de ruimte. De functie g wordt beperkt tot de klasse van monotoon dalende functies.[1] Er wordt verondersteld dat $\Omega \subset \mathbb{R}^2$ een begrensd, rechthoekig domein is.
Vergelijking (2.1) gaat gepaard met een beginconditie, i.e. de initiële afbeelding, en een Neuman randconditie

$$\nabla u(t, \boldsymbol{x}) \cdot \boldsymbol{\nu} = 0 \quad \text{op } (0,T) \times \partial\Omega, \tag{2.2}$$

$$u(0, \boldsymbol{x}) = u_0(\mathbf{x}) \quad \text{in } \Omega, \tag{2.3}$$

waarbij $\boldsymbol{\nu}$ de uitwaartse normaalvector is aan $\partial\Omega$. De randconditie (2.2) zorgt ervoor dat de totale intensiteit van de afbeelding bewaard blijft. De beginconditie $u_0(\mathbf{x})$ stelt de zwart-wit

[1] In de eerstvolgende sectie wordt uitgelegd waarom dit zo is.

afbeelding voor die we wensen te bewerken. De oplossing $u(t, \mathbf{x})$ van (2.1)-(2.3) vertegenwoordigt een familie van geschaalde, gefilterde, gladdere versies van $u_0(\mathbf{x})$. De parameter $t \in [0, T]$ krijgt de betekenis van de *schaal-parameter*. Het bewerken van u_0 door een evolutionaire partiële differentiaalvergelijking wordt een inbedding in de schaalruimte genoemd. De axioma's en fundamentele eigenschappen van zulke inbeddingen zijn bestudeerd in [52].

Figuur 2.1: Toepassing van de Perona-Malik vergelijking (rechts) op een afbeelding met ruis (links). De foto is genomen tijdens de beklimming van de Oude Kwaremont in de Ronde Van Vlaanderen 2015. Parameters: $\tau = 1/4, T = 2$, optie 2. Voor uitleg omtrent deze parameters, zie later.

Figuur 2.1 toont het resultaat van de Perona-Malik vergelijking, toegepast op een foto waar lichte ruis aanwezig is.[1] Dit resultaat werd bekomen door (2.1) te discretiseren in tijd en ruimte en toe te passen op de initiële afbeelding. De pixelwaarden in de matrixvoorstelling van de afbeelding worden op elk discreet tijdstip aangepast volgens een numeriek schema.

In dit hoofdstuk zullen we in detail uitleggen hoe een numeriek schema voor (2.1) wordt bekomen. Bovendien verduidelijken we de moeilijke punten en de implementatie ervan in MATLAB. Op het einde van het hoofdstuk voeren we met deze implementatie numerieke testen uit om het gedrag van de parameters te bestuderen. We starten het hoofdstuk met het beschrijven van het originele idee achter de vergelijking.

2.1 Het idee achter de vergelijking

De diffusievergelijking is een partiële differentiaalvergelijking die vaak optreedt bij fysische processen. De algemene vorm is

$$\partial_t u(\boldsymbol{x}, t) = \nabla \cdot (C(u, \boldsymbol{x}, t)) \nabla u(\boldsymbol{x}, t)), \tag{2.4}$$

waarbij de C de diffusiecoëfficiënt wordt genoemd. Als C constant is, spreken we over homogene diffusie. Als C afhankelijk is van de plaats spreken we over inhomogene diffusie. De vergelijking wordt niet-lineair van zodra C afhankelijk wordt van de oplossing u.

[1] De originele afbeelding is vrij te downloaden op `https://commons.wikimedia.org/wiki/File:Ronde_van_Vlaanderen_2015_-_Oude_Kwaremont_(16867172150).jpg`.


Figuur 2.2: Links ziet u een zwart-wit foto I van de Graslei in Gent. Rechts ziet u $|\nabla I|$ waarbij de pixelwaarden herschaald en geïnverteerd zijn, d.w.z. dat de minimale pixel waarde de kleur wit toekent, en de maximale waarde de kleur zwart.

Voor de constante $C \equiv 1$ krijgen we de lineaire warmtevergelijking. Het is welbekend dat lineaire parabolische differentiaalvergelijkingen een *smoothing* effect hebben, dit betekent dat de oplossingen oneindig glad worden, zelfs al is de initiële data *ruw* [37].

Het idee van Perona en Malik was om dit smoothing effect (sterke diffusie) te bekomen in het binnenste van elke *regio* en tevens de *randen*[1] te verscherpen. Dit zou kunnen verwezenlijkt worden door de diffusiecoëfficiënt C gelijk te stellen aan 1 in het binnenste van elke regio en ze gelijk te stellen aan 0 langs de randen. Op deze manier zullen de randen scherp blijven en zal het smoothing effect apart plaats vinden in elke regio, zonder interactie tussen de verschillende regio's.

De gradiënt van een afbeelding

Om een schatter te hebben voor de locatie van de randen wordt de gradiënt $\nabla u(\boldsymbol{x}, t)$ gebruikt. De gradiënt van een functie, geëvalueerd in een punt, zegt iets over hoe sterk de functie in dat punt verschilt van de functie in zijn omgeving. Met andere woorden, bij een afbeelding zal de gradiënt, in absolute waarde, het grootst zijn langs de randen en klein binnen een regio. Deze eigenschap wordt geïllustreerd in Figuur 2.2. Deze figuur werd in Matlab als volgt bekomen:

```
>> [J,~]=imgradient(I,'central');
>> J=max(J(:))-J;
>> imshow(J,[])
```

De functie `imgradient` berekent de omvang van de gradiënt. Door de methode `central` te kiezen, wordt dit berekend via centrale differenties: $\partial_x I = (I(x+1) - I(x-1))/2$. Het resultaat van de functie is de absolute waarde van de partiële afgeleiden $J = [(\partial_x I)^2 + (\partial_y I)^2]^{1/2}$. De partiële afgeleiden kunnen afzonderlijk worden berekend via

[1] Met randen op een afbeelding bedoelen we wat effectief waargenomen wordt als randen. In de buurt van een rand zullen de pixelwaarden significante verschillen vertonen. Een regio is een gebied omgeven door randen.

```
[Gx, Gy] = imgradientxy(I,'central');
```

Wanneer de gradiënt operator wordt toegepast aan de rand van de afbeelding, wordt er aangenomen dat de waarden buiten de randen van de afbeelding gelijk zijn aan de dichtste buren. Dit betekent dat de buitenste rand wordt overgenomen. Bij een Neumann randconditie wordt dit anders ingebouwd, namelijk door langs de rand te spiegelen. Zonder zelf de functie opnieuw te schrijven (wat ook niet zo moeilijk is), kan de randconditie eigenhandig ingebouwd worden als volgt

```
>> Gx(:,1)=0;  Gx(:,end)=0; Gy(1,:)=0; Gy(end,:)=0;
>> J=(Gx.^2+Gy.^2).^(1/2);
```

Voorwaartse en achterwaartse diffusie

Met ∇u als schatter voor de locatie van de randen, kan de diffusiecoëfficiënt gekozen worden als een functie van de grootte van de ∇u, i.e.

$$C = g(|\nabla u|).$$

Volgens bovenvermelde strategie moet g een niet-negatieve monotoon dalende functie zijn met $g(0) = 1$. In [31] werden volgende twee functies voorgesteld

$$g_1(x) = e^{-\frac{1}{2}(\frac{x}{\kappa})^2} \tag{2.5}$$

en

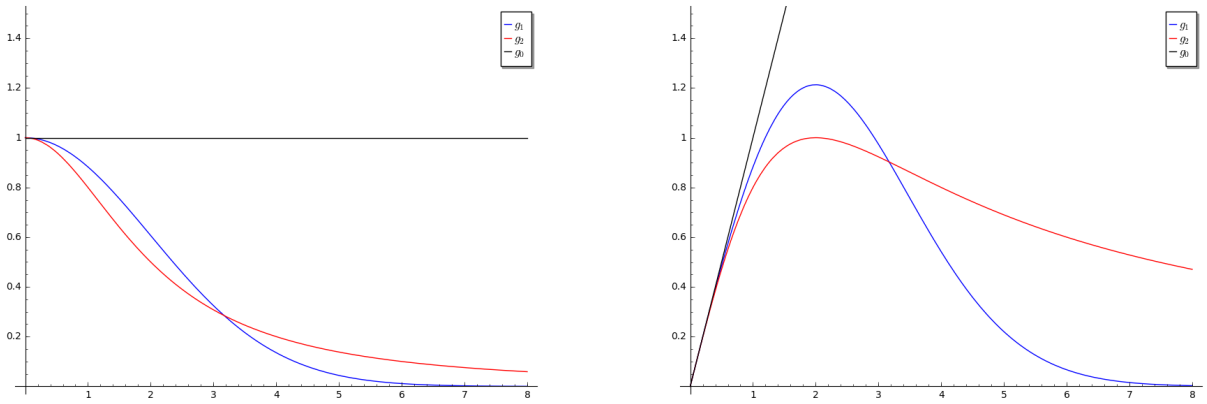
$$g_2(x) = \frac{1}{1 + (x/\kappa)^2}, \tag{2.6}$$

waarbij $\kappa \in \mathbb{R}^+$ de contrast parameter wordt genoemd. De invloed van deze parameter op het model mag niet onderschat worden. Om een goed idee te hebben voor de keuze van κ is het belangrijk om in te zien wat de rol van deze parameter is binnen het model. Dit wordt duidelijker door (2.1) te beperken tot het één-dimensionale geval en de flux $f(x) = g(|x|)x$ te beschouwen [48, 4].

Omdat het ééndimensionale geval van de Perona-Malik vergelijking herschreven kan worden als

$$\begin{aligned} \partial_t u &= \partial_x(g(|u_x|)u_x) \\ &= f'(u_x)u_{xx}, \end{aligned} \tag{2.7}$$

kan er onderscheid gemaakt worden wanneer $f'(u_x)$ positief (voorwaartse diffusie) of negatief (achterwaartse diffusie) is. Bij voorwaartse diffusie zal een pixel altijd meer massa verliezen aan zijn kleinere buren dan het zal krijgen van zijn grotere buren. Dit wil zeggen dat de afbeelding lokaal gladder, maar minder scherp wordt. Echter, bij achterwaartse diffusie kan een pixel meer massa krijgen van zijn grotere buren dan het verliest aan zijn kleinere. In dit geval wordt de afbeelding lokaal verscherpt. Dit is net wat we wensen voor de randen van de afbeelding.



Figuur 2.3: Links de diffusiefuncties $g(x)$, rechts de corresponderende flux functies $f(x) = g(|x|)x$. Zwart is de homogene diffusie $g_0(x) = 1$, blauw stelt $g_1(x) = e^{-x^2/2\kappa^2}$ voor en rood is de diffusiefunctie $g_2(x) = 1/(1 + x^2/\kappa^2)$, met telkens $\kappa = 2$.

In Figuur 2.3, de linker grafiek, zijn de twee varianten (2.5) en (2.6) uitgezet voor $\kappa = 2$, alsook de homogene diffusie $g_0(x) = 1$. Op de grafiek rechts zijn de corresponderende flux functies f uitgezet. De homogene diffusie heeft een constant stijgend verloop. Voor de niet-lineaire diffusie wordt er een maximum bereikt in $\kappa = 2$, waarna ze terug daalt en naar nul nadert in de limiet naar oneindig. De grafiek voor negatieve waarden van x is de puntspiegeling om de oorsprong.

De eerste afgeleide van f kan algemeen geschreven worden als

$$\begin{aligned} f'(x) &= g'(|x|)\frac{x}{|x|}x + g(|x|) \\ &= g'(|x|)|x| + g(|x|). \end{aligned} \tag{2.8}$$

Noemen we $f_i'(x) = g_i(|x|)x$, $i = 1, 2$, voor de praktische keuzes van g. Dan geldt er

$$\begin{aligned} g_1'(x) = -\frac{x}{\kappa^2}g_1(x) \Rightarrow f_1'(x) &= \left(-\frac{|x|^2}{\kappa^2} + 1\right) g_1(|x|), \\ g_2'(x) = -\frac{2x}{\kappa^2}g_2^2(x) \Rightarrow f_2'(x) &= \left(-\frac{|x|^2}{\kappa^2} + 1\right) g_2^2(|x|). \end{aligned}$$

Hieruit volgt dat f_i, $i = 1, 2$, een extremum bereikt bij $x = \pm\kappa$. Merk op dat $f_i'(0) = g(0) = 1$ en $\lim_{|x|\to\infty} f_i(x) = 0$, voor $i = 1, 2$. Hierdoor wordt bij $x = \kappa$ (respectievelijk $x = -\kappa$) het maximum (respectievelijk het minimum) van f bereikt. Bijgevolg is $f_i'(x) > 0$ voor $|x| < \kappa$ en $f_i'(x) < 0$ voor $|x| > \kappa$.

Bovenstaande berekeningen verklaren waarom κ de contrastparameter genoemd wordt. Als $|u_x| < \kappa$, is $f'(u_x)$ positief. Wanneer $|u_x| > \kappa$, is $f'(u_x)$ negatief. Bijgevolg bepaalt κ in (2.7) de gebieden waar er voorwaartse (laag contrast) en achterwaartse diffusie (hoog contrast) plaatsvindt. Concreet voor het twee-dimensionale geval: daar waar $|\nabla u| < \kappa$ is, vindt er voorwaartse diffusie plaats en krijgen we een gelijkaardig effect als de lineaire warmtevergelijking. Voor pixels waarvoor $|\nabla| > \kappa$, vindt er achterwaartse diffusie plaats, hier wordt de afbeelding

lokaal verscherpt. Merk op dat $|\nabla u|$ diende als schatter van de locatie van de randen. Grote waarden van $|\nabla u|$ gaven een indicatie dat de beschouwde pixel een rand is. Vanaf welke waarde $|\nabla u|$ pixels nu effectief als een rand beschouwd worden, wordt nu wegens voorgaande bespreking gegeven door de keuze van κ.

Voor slechte keuzes van κ kunnen de resultaten een vertekend beeld geven. In [7] wordt voorgesteld om κ gelijk te stellen aan het 90% kwantiel van $|\nabla u_0|$. De implementatie van Wolfram Alpha opteert dan weer voor het 50% kwantiel van $|\nabla u_0|$.[1] Rond de keuze van κ zijn er in de literatuur diverse artikels te vinden. Zie bijvoorbeeld [41] voor een mooi overzicht. In onze implementatie wordt geopteerd voor de methode beschreven door [3]. Daarin wordt κ gekozen als

$$\kappa = 1.4826\text{MAD}(|\nabla u_0|), \tag{2.9}$$

waarbij MAD staat voor de *median absolute variation* en gedefinieerd wordt als

$$\text{MAD}(|\nabla u_0|) = \text{median}(|\nabla u_0 - \text{median}(|\nabla u_0|)|).$$

De MAD is een consistente schatter voor de standaarddeviatie en heeft als voordeel ten opzichte van andere schatters dat ze robuust is (zie bijvoorbeeld [25]).

2.2 Numerieke oplossing

In dit deel wordt een numeriek schema opgesteld die de oplossing van (2.1) benadert in een discreet aantal tijdstippen. We kiezen ervoor om eerst de tijdsafgeleide te benaderen.

2.2.1 Semi-discretisatie in tijd

Er wordt gestart met de semi-discretisatie in tijd van (2.1). Neem $l \in \mathbb{N}$. Het interval $[0, T]$ wordt uniform verdeeld in N_t deel-intervallen. Dus $[0, T]$ is verdeeld in $[t_{l-1}, t_l]$ voor $l = 1, \ldots, N_t$, waar $t_l = l\tau$ en $\tau = \frac{T}{N_t}$. We noemen τ de tijdstap. De tijdsafgeleide in (2.1) wordt vervangen door de achterwaartse differentie. De niet-lineaire termen van de vergelijking worden geëvalueerd op het vorige tijdstip terwijl de lineaire termen worden beschouwd op het huidig tijdstip. Op deze manier ontstaat een semi-discreet lineair schema.

Semi-discreet lineair schema voor het oplossen van vergelijking (2.1)
Zij $N_t \in \mathbb{N}$ en $\tau = T/N_t$ vaste getallen, en zij u_0 gegeven door (2.3). Voor elke $l = 1, \ldots, N_t$, zoeken we een funtie u^l die een oplossing is van de vergelijking

$$\frac{u^l - u^{l-1}}{\tau} - \nabla \cdot (g(|\nabla u^{l-1}|)\nabla u^l) = 0. \tag{2.10}$$

De volgende stap is de ruimtelijke coördinaten discretiseren.

[1] Hierbij wordt wel gewerkt met de functie $e^{-(x/\kappa)^2}$ in plaats van g_1. Ook in [31] werd in feite deze vorm voorgesteld. Hier is de functie herschaald zodat $g_1(x)x$ en $g_2(x)x$ in hetzelfde punt hun maximum bereiken.

2.2.2 Discretisatie in de ruimte

Keuze van de methode

We moeten een keuze maken voor de discretisatiemethode voor de ruimtelijke coördinaten. Enerzijds kan voor de eindige elementen methode (EEM) geöpteerd worden. De EEM is gebaseerd op de variationele formulering van de partiële differentiaalvergelijking die moet worden opgelost. Voor een uitgebreide uitleg van de EEM verwijzen we naar [36, 44]. Belangrijk is dat de benadering van de oplossing hier stuksgewijze lineaire functies zijn.
Anderzijds kan gekozen worden voor de eindige volume methode (EVM). Hierbij wordt de partiële differentiaalvergelijking geïntegreerd over een eindig volume. Een uitgebreide uitleg van de methode is te vinden in [12, 29, 27]. De benadering van de oplossing bij de EVM is stuksgewijs constant verondersteld. Het rooster wordt bij de EVM gekozen zoals in Figuur 1.1. Doordat de intensiteit binnen een pixel constant is, lijkt deze methode de meest natuurlijke. In het resterend gedeelte van deze sectie leggen we uitgebreid uit hoe het numerieke schema wordt opgesteld voor probleem (2.1), gebruik makend van de EVM.

Gebruikte notaties

We discretiseren $\Omega = (0, n) \times (0, m)$ in een rooster van mn elementen. Deze elementen zijn dan vierkanten (lees: pixels). We veronderstellen dat de oppervlakte van één zo'n pixel de basiseenheid definieert. De eindige volumes komen overeen met deze pixels. Zij (i, j) met $1 \leq i \leq m$ en $1 \leq j \leq n$, de pixelindex van zo'n pixel. Dan worden de coördinaten van het midden van deze pixel gegeven door $(x_j, y_i) := (j - 1/2, i - 1/2)$. We zullen (x_j, y_i) als representatief punt gebruiken van (i, j). De pixelindexen kunnen afgebeeld worden op de oppervlakte die ze beschrijven in $\mathbb{R}^2$,

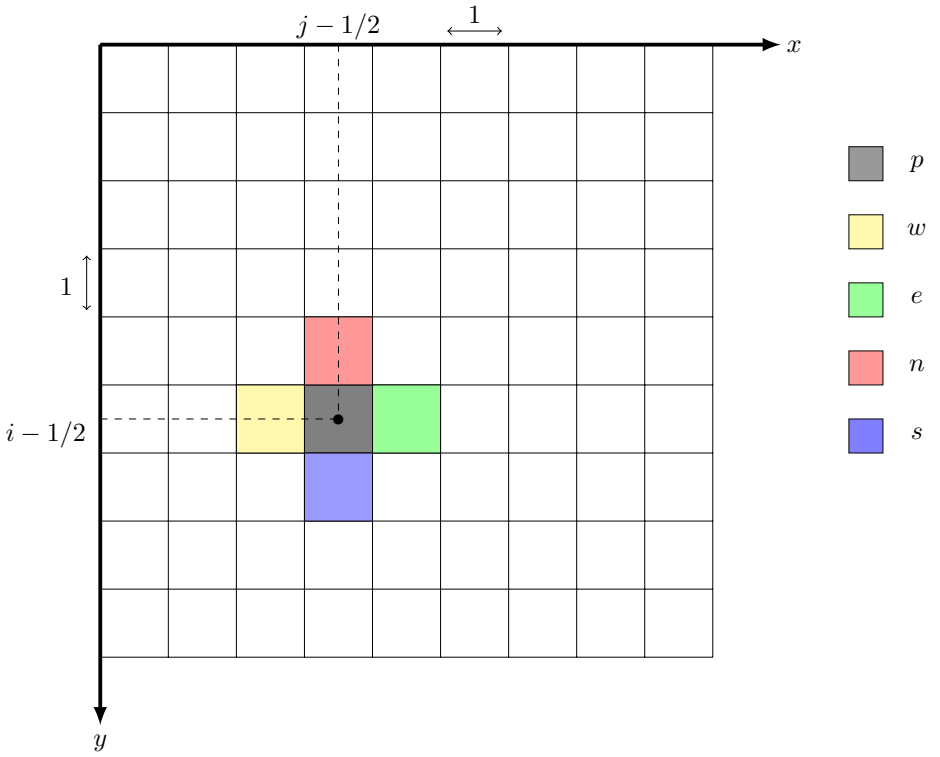
$$\alpha : \{1, \ldots, m\} \times \{1, \ldots, n\} \mapsto \Omega. \tag{2.11}$$

Definitie 2.1. *Beschouw een pixel $p = \alpha(i, j)$. Dan heeft p ten hoogste vier buren in het rooster, die we noteren als e, w, s en n. De respectievelijke representatieve punten van deze buren worden gegeven door*

$$(x_j + 1, y_i), (x_j - 1, y_i), (x_j, y_i + 1), (x_j, y_i - 1).$$

Deze buren komen overeen met de windrichtingen 'east','west', 'south' en 'north'. We noteren deze verzameling van pixels als $\mathcal{N}(p)$.

Tenslotte noteren we met u_p^l de benaderde waarde van de oplossing binnenin het eindige volume p.



Figuur 2.4: De relaties tussen de pixels. Hierbij is $p = \alpha(i, j)$.

Eindige volume methode

Om de de ruimtelijke coördinaten x en y te discretiseren, wordt vergelijking (2.10) geïntegreerd over een pixel $p = \alpha(i, j)$, i.e.

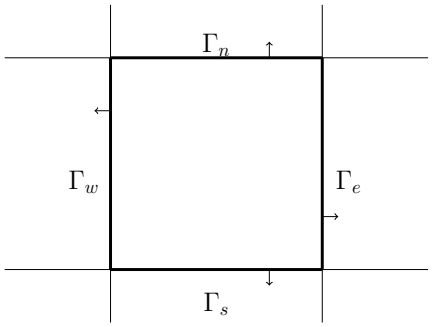
$$\int_p \frac{u^l - u^{l-1}}{\tau} dxdy = \int_p \nabla \cdot \left(g(|\nabla u^{l-1}|) \nabla u^l \right) dxdy. \tag{2.12}$$

Op het rechterlid kan nu de stelling van Green worden toegepast zodat

$$\int_p \nabla \cdot \left(g(|\nabla u^{l-1}|) \nabla u^l \right) dxdy = \int_{\partial p} g(|\nabla u^{l-1}|) \nabla u^l \cdot \boldsymbol{\nu} ds, \tag{2.13}$$

met $\boldsymbol{\nu}$ de uitwaartse normaalvector.

De doorsnede van p met $\{e, w, n, s\}$ noteren we als $\Gamma_{\{e,w,n,s\}}$. De kromme ∂p kan worden opgedeeld in deze vier disjuncte lijnstukken zodanig dat $\partial p = \overline{\Gamma_e} \cup \overline{\Gamma_w} \cup \overline{\Gamma_s} \cup \overline{\Gamma_n}$, zie Figuur 2.5.



Figuur 2.5: Opsplitsing rand van een pixel p. De pijlen stellen de richting van de normaalvector voor.

Rekening houdend met de normaalvector $\boldsymbol{\nu}$ en het assenstelsel waarbij de y-as naar beneden gericht is, krijgen we op elk van deze lijnstukken

$$\int_{\Gamma_e} g(|\nabla u^{l-1}|)\nabla u^l \cdot (1,\ 0)^t dy = \int_{\Gamma_e} g(|\nabla u^{l-1}|)\partial_x u^l dy, \tag{2.14}$$

$$\int_{\Gamma_w} g(|\nabla u^{l-1}|)\nabla u^l \cdot (\text{-}1,\ 0)^t dy = -\int_{\Gamma_w} g(|\nabla u^{l-1}|)\partial_x u^l dy, \tag{2.15}$$

$$\int_{\Gamma_n} g(|\nabla u^{l-1}|)\nabla u^l \cdot (0,\ \text{-}1)^t dx = -\int_{\Gamma_n} g(|\nabla u^{l-1}|)\partial_y u^l dx, \tag{2.16}$$

$$\int_{\Gamma_s} g(|\nabla u^{l-1}|)\nabla u^l \cdot (0,\ 1)^t dx = \int_{\Gamma_s} g(|\nabla u^{l-1}|)\partial_y u^l dx. \tag{2.17}$$

Merk alvast op dat wanneer de pixel zich langs de rand van de afbeelding bevindt, er één van de vier uitdrukkingen (2.14), (2.15), (2.16), (2.17) nul wordt, krachtens de Neumann randconditie (2.2). Als het gaat om een pixel uit één van de vier hoekpunten worden er twee van de vier uitdrukkingen (2.14), (2.15), (2.16), (2.17) nul. Veronderstel voorlopig dat p zich niet op één van de vier randen van de afbeelding bevindt. Elke van deze vier lijnintegralen wordt vervolgens benaderd door het product van de lengte van het lijnstuk (wat gelijk aan één is, wegens de veronderstellingen) en de waarde van het integrandum in het midden van het lijnstuk. We krijgen volgende kwadratuurformules:

$$\int_{\Gamma_e} g(|\nabla u^{l-1}|)\partial_x u^l dy \approx g(|\nabla u^{l-1}(x_j + 0.5, y_i)|)\partial_x u^l(x_j + 0.5, y_i), \tag{2.18}$$

$$-\int_{\Gamma_w} g(|\nabla u^{l-1}|)\partial_x u^l dy \approx g(|\nabla u^{l-1}(x_j - 0.5, y_i)|)\partial_x u^l(x_j - 0.5, y_i), \tag{2.19}$$

$$-\int_{\Gamma_n} g(|\nabla u^{l-1}|)\partial_y u^l dx \approx g(|\nabla u^{l-1}(x_j, y_i - 0.5)|)\partial_y u^l(x_j, y_i - 0.5), \tag{2.20}$$

$$\int_{\Gamma_s} g(|\nabla u^{l-1}|)\partial_y u^l dx \approx g(|\nabla u^{l-1}(x_j, y_i + 0.5)|)\partial_y u^l(x_j, y_i + 0.5). \tag{2.21}$$

De rechterleden in (2.18), (2.19), (2.20) en (2.21) bestaan telkens uit twee factoren. De tweede factor, de partiële afgeleide, wordt benaderd door een centrale differentie met halve staplengte.

De eerste factor zullen we kortweg noteren als $g_q^{l-1}(p)$, waarbij q hier de symbolische notatie is voor een pixel uit de verzameling $\mathcal{N}(p)$. Wegens Definitie 2.1 laat dit ons toe om bovenvermelde benaderingen nu als volgt te noteren,

$$(2.18) \approx g_e^{l-1}(p) \cdot (u_e^l - u_p^l), \tag{2.22}$$

$$(2.19) \approx g_w^{l-1}(p) \cdot (u_w^l - u_p^l), \tag{2.23}$$

$$(2.20) \approx g_n^{l-1}(p) \cdot (u_n^l - u_p^l), \tag{2.24}$$

$$(2.21) \approx g_s^{l-1}(p) \cdot (u_s^l - u_p^l). \tag{2.25}$$

Optellen van (2.22), (2.23), (2.24) en (2.25) geeft ons dan de uiteindelijke benadering van het rechterlid in (2.12), voor alle inwendige pixels. Beschouw nu een pixel die langs de rand gelegen is, die niet in een hoekpunt gelegen is. De term die dan nul wordt in (2.14), (2.15), (2.16) of (2.17) is net degene in de richting waar de pixel geen buur heeft. Bijvoorbeeld voor pixels gelegen langs de linkerrand van de afbeelding wordt (2.15) nul. Voor de vier pixels in de hoekpunten geldt dezelfde redenering, alleen worden er nu twee termen nul doordat ze aan twee randen van de afbeelding grenzen. Samengevat kunnen we de benadering van het rechterlid in (2.12) noteren als

$$\sum_{q \in \mathcal{N}(p)} g_q^{l-1}(p)(u_q^l - u_p^l), \tag{2.26}$$

en dit voor alle pixels p uit het rooster.

Het linkerlid in (2.12) is exact gelijk aan

$$\frac{u_p^l - u_p^{l-1}}{\tau}, \tag{2.27}$$

doordat u^l en u^{l-1} een constante waarde hebben binnen een pixel.

Tenslotte stellen we (2.26) en (2.27) aan elkaar gelijk en vermenigvuldigen we alles met τ zodat we het volgende numerieke schema krijgen.

Lineair volledig discreet eindige volume schema voor het oplossen van vergelijking (2.1). *Zij $N_t \in \mathbb{N}$ en $\tau = T/N_t$ vaste getallen, $t_l = l\tau, l = 0, \ldots N_t$. Voor elke $l = 1, \ldots, N_t$, bepalen we u_p^l, met $p = \alpha(i,j)$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, die voldoet aan*

$$u_p^l + \tau \sum_{q \in \mathcal{N}(p)} g_q^{l-1}(p)(u_p^l - u_q^l) = u_p^{l-1}, \tag{2.28}$$

startend met u^0 gegeven door (2.3).

Door de positieve coëfficiënten g_q^{l-1} verkrijgen we een uniforme L_∞-stabiliteit,

$$\min_{p \in \mathcal{P}} u_p^0 \leq \min_{p \in \mathcal{P}} u_p^l \leq \max_{p \in \mathcal{P}} u_p^l \leq \max_{p \in \mathcal{P}} u_0^l, \quad 1 \leq l \leq N_t, \tag{2.29}$$

waarbij $\mathcal{P}$=$\{\alpha(i,j) |\ i = 1, \ldots, m,\ j = 1, \ldots, n\}$ de verzameling is van alle eindige volumes.
Zij $u_p^l = \max_{r \in \mathcal{P}} u_r^l$. Dan zijn alle termen uit de som $\sum_{q \in \mathcal{N}(p)} g_q(p)(u_p^l - u_q^l)$ positief en geldt

er bijgevolg dat $u_p^l \leq u_p^{l-1} \leq \max_{r\in\mathcal{P}} u_r^{l-1}$, wat het resultaat oplevert voor het maximum. De relatie voor het minimum wordt op een analoge manier bekomen.

Eigenschap (2.29) is belangrijk omdat ze ons garandeert dat we steeds een afbeelding krijgen als output. De gewijzigde waarden blijven immers binnen het domein van de initële afbeelding u_0.

Het schema (2.28) is eigenhandig geïmplementeerd in MATLAB. De functie `PeronaMalik` is te vinden achteraan in Appendix C.1. Deze functie hangt af van de volgende parameters:

Naam	Klasse	Uitleg
`u0`	$m \times n$ matrix	de begin afbeelding
`timeStep`	integer	de tijdstap τ
`finalTime`	integer	de eindtijd T
`option`	1 of 2	keuze diffusiefunctie g: $1 \mapsto$ (2.5), $2 \mapsto$ (2.6)

Wanneer deze parameters correct zijn, kan de functie `PeronaMalik` als volgt gebruikt worden:

```
>> images = PeronaMalik(u0,scaleStep,finalTime,option)
```

Dit commando geeft een *cell array* images terug als output. Het commando `images{l}` geeft de evolutie van de afbeelding terug op het tijdstip t_l van de partitie $[0, T]$ (zie Sectie 2.2.1 voor de gebruikte notatie). Het resultaat op het eindtijdstip T wordt gegeven door `images{end}`.

Een nadeel van een impliciet schema is dat het moeilijker te implementeren valt. Het schema (2.28) is een stelsel $Au^l = u^{l-1}$ dat moet worden opgelost. Het opstellen van deze matrix is niet zo eenvoudig. Daarom geven we in volgende sectie hierover wat meer uitleg.

2.3 Implementatie van het numerieke schema

Lineaire indices

Vergelijking (2.28) moet worden omgezet naar een lineair stelsel van de vorm $Au^l = u^{l-1}$. Dat dit niet zo eenvoudig is, komt doordat de onbekende u^l een matrix is, en geen vector. Om dit probleem op te lossen, moet er worden overgegaan op lineaire indices zodat u^l een vector wordt. We noemen m het aantal rijen en n het aantal kolommen van u^0. We gaan over op lineaire indices door kolom per kolom te overlopen. Om te zien hoe deze indexering werkt, nummeren we de elementen van een algemene matrix $b \in \mathbb{R}^{m\times n}$:

$$\begin{pmatrix} b_{1,1}^{1} & b_{1,2}^{m+1} & \dots & b_{1,n}^{(n-1)m+1} \\ b_{2,1}^{2} & b_{2,2}^{m+2} & \dots & b_{2,n}^{(n-1)m+2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,1}^{m} & b_{m,2}^{2m} & \dots & b_{m,n}^{mn} \end{pmatrix}$$

Om over te gaan van matrix indices (i, j) naar lineaire indices k wordt volgende transformatie gebruikt

$$(i, j) \to k = (j - 1)m + i. \tag{2.30}$$

Omgekeerd om van lineaire indices naar matrix indices over te gaan, wordt volgende transformatie toegepast

$$k \to \begin{cases} i = k \ (mod\ m) \text{ als } k \nmid m, \text{ anders } i = k, \\ j = \lceil k/m \rceil. \end{cases} . \tag{2.31}$$

Hierbij is $\lceil x \rceil$ het kleinste geheel getal groter dan of gelijk aan x. In het Engels is dit bekend als de *ceil function*. Om in MATLAB een $m \times n$ matrix `b` om te zetten naar een vector van lengte mn, volgens transformatie (2.30), volstaat het commando `b(:)`.

Opstellen matrix

We stellen de matrix $A \in \mathbb{R}^{mn \times mn}$ op uit het stelsel $Au^l = u^{l-1}$. We proberen de structuur van de matrix zo duidelijk mogelijk uit te leggen. Hiertoe definiëren we volgende vier $m \times n$ matrices

$$g^s_{i,j} = \begin{cases} g(|\nabla u^{l-1}(j - 1/2, i)|), & \text{als } i = 1, \ldots, m - 1,\ j = 1, \ldots, n; \\ 0, & \text{als } i = m. \end{cases} \tag{2.32}$$

$$g^n_{i,j} = \begin{cases} g(|\nabla u^{l-1}(j - 1/2, i - 1)|), & \text{als } i = 2, \ldots, m,\ j = 1, \ldots, n \\ 0 & \text{als } i = 1. \end{cases} \tag{2.33}$$

$$g^e_{i,j} = \begin{cases} g(|\nabla u^{l-1}(j, i - 1/2)|), & \text{als } i = 1, \ldots,\ m, j = 1, \ldots, n - 1; \\ 0 & \text{als } j = n. \end{cases} \tag{2.34}$$

$$g^w_{i,j} = \begin{cases} g(|\nabla u^{l-1}(j - 1, i - 1/2)|), & \text{als } i = 1, \ldots, m,\ j = 2, \ldots, n; \\ 0 & \text{als } j = 1. \end{cases} \tag{2.35}$$

Het zal ook handig blijken te zijn om de som van deze matrices te beschouwen:

$$s_{i,j} = (g^s_{i,j} + g^n_{i,j} + g^e_{i,j} + g^w_{i,j}). \tag{2.36}$$

Merk nogmaals op dat voor een pixel met (i, j) als matrix index, de coördinaten van het middelpunt van deze pixel juist worden gegeven door $(j - 1/2, i - 1/2)$, zie Figuur 2.4. Enige voorzichtigheid is dus zeker op zijn plaats, aangezien de volgorde in de laatste notatie juist omgekeerd is.

Voor een pixel p met als matrix index (i, j) hebben we volgende vertaling van vergelijking (2.28) naar matrix indices

$$(1+\tau s_{i,j})u^l_{i,j} - \tau g^e_{i,j}u^l_{i,j+1} - \tau g^w_{i,j}u^l_{i,j-1} - \tau g^n_{i,j}u_{i-1,j} - \tau g^s_{i,j}u^l_{i+1,j} = u^{l-1}_{i,j}. \tag{2.37}$$

Wegens de definities van g_s, g_n, g_e en g_w is (2.37) geldig voor alle $i = 1, \ldots, m$, $j = 1, \ldots, n$.

Stel nu de lineaire index van (i, j) gelijk aan k, dan volgt uit (2.30) dat we (2.37) kunnen schrijven als

$$(1+\tau s_k)u^l_k - \tau g^e_k u^l_{k+m} - \tau g^w_k u^l_{k-m} - \tau g^n_k u_{k-1} - \tau g^s_k u^l_{k+1} = u^{l-1}_k. \tag{2.38}$$

Uit vergelijking (2.38) volgt hoe we de matrix A uit het stelsel $Au^l = u^{l-1}$ moeten definiëren. Deze matrix A zal een bandmatrix zijn, bestaande uit de vijf volgende diagonalen

$$\begin{aligned}
A_{k,k} &= 1+\tau s_k, \quad 1 \le k \le mn;\\
A_{k,k+1} &= -\tau g^s_k, \quad 1 \le k \le mn-1;\\
A_{k,k-1} &= -\tau g^n_k, \quad 2 \le k \le mn;\\
A_{k,k+m} &= -\tau g^e_k, \quad 1 \le k \le mn-m;\\
A_{k,k-m} &= -\tau g^w_k, \quad m+1 \le k \le mn.
\end{aligned}$$

De structuur van de matrix A ziet er dus als volgt uit,

$$A = \begin{bmatrix}
1+\tau s_1 & -\tau g^s_1 & -\tau g^e_1 & & & & & & \\
-\tau g^n_2 & \ddots & \ddots & \ddots & & & & & \\
 & \ddots & \ddots & \ddots & \ddots & & & & \\
-\tau g^w_{m+1} & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & -\tau g^e_{mn-m} & \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & -\tau g^s_{mn-1} \\
 & & & & -\tau g^w_{mn} & \ddots & -\tau g^n_{mn} & 1+\tau s_{mn} &
\end{bmatrix}.$$

Zo'n type van matrix kan in MATLAB elegant worden ingegeven met behulp van de functie `spdiags`. De functie

```
>> A=spdiags(B,d,m,n)
```

creëert een $m \times n$ *ijle*[1] matrix, door de kolommen van de matrix B op de diagonalen te plaatsen, gespecificeerd door de vector d. Bijgevolg is onze matrix B niets anders dan een $mn \times 5$ matrix wiens kolommen juist de vectoren g^n, g^s, g^e, g^w en s zijn. De vector d geeft aan op welke diagonalen de kolommen van B komen te liggen. De hoofddiagonaal wordt aangeduid met het getal 0. De eerste bovendiagonaal wordt aangeduid met 1, de tweede bovendiagonaal met 2, enzovoort.

[1]Engels: *sparse matrix*. Dit zijn grote matrices waarbij de meeste elementen gelijk aan nul zijn.

Analoog wordt de eerste onderdiagonaal aangeduid met -1, de tweede met -2, enzovoort. In het geval van boven-en onderdiagonalen zijn de overeenstemmende kolommen van B langer dan nodig. In het geval van bovendiagonalen plaatst MATLAB het onderste gedeelte van de kolom op de diagonaal. In het geval van onderdiagonalen plaatst MATLAB het bovenste gedeelte van de kolom op de diagonaal. Gezien onze definities van g^n, g^s, g^e, g^w zouden we juist de omgekeerde werkwijze willen. In volgende sectie zullen we aantonen dat A symmetrisch is. Dit laat ons toe om de plaatsing van de boven-en onderdiagonalen te verwisselen. Merk tenslotte op dat het opstellen van A via `spdiags` niet enkel elegant is, maar ook efficiënt. De volledige matrix A element per element bepalen zou immers m^2n^2 geheugenplaatsen vereisen, terwijl we nu de matrix B moeten opstellen die slechts $5mn$ geheugenplaatsen voorziet.

Eigenschappen van A

Door de definitie van de functie g zijn alle termen van g^s, g^n, g^e en g^w positief. Dit heeft als gevolg dat de matrix A strikt diagonaal-dominant is, want

$$|a_{k,k}| = |1 + \tau s_k| = 1 + \tau(g_k^s + g_k^n + g_k^e + g_k^w) > \tau(g_k^s + g_k^n + g_k^e + g_k^w) = \sum_{l \neq k} |a_{k,l}|,$$

en dit voor alle $k = 1, \ldots, mn$. Deze eigenschap impliceert dat het stelsel altijd een unieke oplossing zal hebben.

De matrix A is bovendien symmetrisch. Om dit na te gaan, hoeven we enkel te kijken of de twee bovendiagonalen symmetrisch zijn ten opzichte van de twee onderdiagonalen. Er zijn immers geen andere niet-nul elementen aanwezig in de matrix. Uit de definitie van g^s en g^n volgt dat $g_{i,j}^s = g_{i+1,j}^n$ voor $i = 1, \ldots, m-1$ en $j = 1, \ldots, n$. Stel nu de lineaire index van (i, j) gelijk aan k dan volgt er dat $g_k^s = g_{k+1}^n$ voor $k = 1, \ldots mn - 1$. Dit impliceert dat $A_{k,k+1} = A_{k+1,k}$ voor alle $k = 1, \ldots, mn - 1$. Kijken we nu naar de twee andere diagonalen, dan stellen we vast dat $g_{i,j}^e = g_{i,j+1}^w$ voor $i = 1, \ldots, m$ en $j = 1, \ldots, n-1$. Stellen we opnieuw de lineaire index van (i, j) gelijk aan k dan volgt er dat $g_k^e = g_{k+m}^w$. Of dus $A_{k,k+m} = A_{k+m,k}$ voor $k = 1, \ldots, mn - m$. Hierdoor kunnen we besluiten dat A symmetrisch is.

Doordat de diagonaalelementen van A strikt positief zijn, en omdat A strikt diagonaal-dominant is, liggen de eigenwaarden van de matrix A in het open rechterhalfvlak wegens het Gerschgorin-cirkeltheorema.[1] Doordat A een reële symmetrische matrix is, zijn de eigenwaarden van A reëel. Beide eigenschappen samen impliceren dat de eigenwaarden van A strikt positief zijn. Bijgevolg is de matrix A positief-definiet.

De eigenschappen van de matrix (symmetrisch, positief-definiet en zeer ijl) suggereren dat de *conjugate gradient method* toepasbaar is om het stelsel efficiënt op te lossen [13][Hoofdstuk 10.2]. Een goede *preconditioner* kan ervoor zorgen dat de conjugate gradient methode convergeert in een aanzienlijk kleiner aantal stappen. Een preconditioner is een matrix C zodat in plaats van

[1] Zie [13][Stelling 7.2.1] of [46][Stelling 4.1].

$Ax = b$, het stelsel $(C^{-1}A)x = C^{-1}b$ wordt opgelost. Hierbij is $K(C^{-1}A) < K(A)$.[1] De keuze van een preconditioner is een theorie op zich [13][Hoofdstuk 10.3] en daarom gaan we hier niet te diep op in. Een veelgebruikte precondioner is de incomplete Cholesky decompositie. Omdat dit een eenvoudige preconditioner is, die symmetrisch en positief-definiet is, kiezen we voor deze preconditioner. De resultaten die we hiermee verkrijgen zijn bevredigend.

De MATLAB-functie `pcg(A,b,tol,maxit,M1,M2)` gebruikt de preconditioner `M=M1*M2` en lost efficiënt het stelsel $M^{-1}Ax = M^{-1}b$ op. `MaxIt` specificeert het maximum aantal iteraties. `tol` specificeert de tolerantie van de methode. Een incomplete Cholesky factor van de matrix `A` kan berekend worden via `H=ichol(A)`.

Benadering coëfficiënten

Het enige dat ons nog rest te doen, is het berekenen van de matrices g^s, g^n, g^e en g^w. Deze moeten opnieuw benaderd worden aangezien het gaat om de intensiteit op een gemeenschappelijke zijde van twee pixels. Hiervoor zijn verschillende benaderingen mogelijk. Bijvoorbeeld een 2-punt-formule die het gemiddelde van de buren neemt:

$$g^e_{i,j} \approx \frac{g(|\nabla u^{l-1}_{i,j}|) + g(|\nabla u^{l-1}_{i,j+1}|)}{2}.$$

De gradiënt die hierin optreedt, kan dan op zijn beurt benaderd worden door centrale differenties. Het berekenen van de gradiënt gebeurt zoals uitgelegd in Sectie 2.1.

In [31] wordt dan weer volgende benadering gebruikt:

$$g^e_{i,j} \approx g(|u^{l-1}_{i,j+1} - u^{l-1}_{i,j}|).$$

Uiteraard kunnen er ook nog andere benaderingen gebruikt worden dan de vorige twee. In de implementatie uit Appendix C.1 hebben we geöpteerd voor de twee punt-formule.

2.4 Parameter studie

De implementatie van (2.28), te vinden in Appendix C.1, kan toegepast worden op een zwart-wit afbeelding u_0 (die bijvoorbeeld ruis bevat):

```
>> images = PeronaMalik(u0,scaleStep,finalTime,option,a)
```

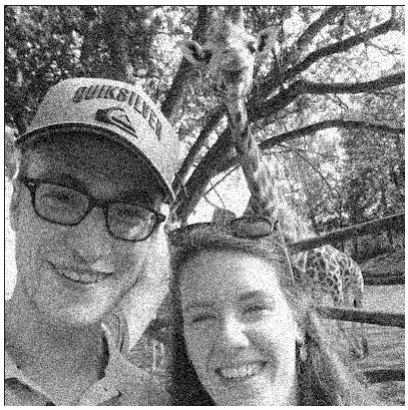
Daarbij moeten de parameters `finalTime` en `scaleStep` eigenhandig gekozen worden. De implementatie laat ook toe een keuze te maken voor de diffusiecoëfficiënt. Er zijn voor g twee opties voorzien, namelijk

$$g_1(x) = e^{-\frac{1}{2}(\frac{x}{\kappa})^\alpha} \quad \text{of} \quad g_2(x) = \frac{1}{1 + (x/\kappa)^\alpha},$$

[1] Het conditiegetal $K(A)$ is gedefinieerd als $K(A) = \|A\| \, \|A^{-1}\|$. Voor mogelijke matrix-normen, zie [46, 38]. Hoe kleiner het conditiegetal, hoe sneller de conjugate gradient methode convergeert.

waarbij $\alpha \in \mathbb{R}$ een extra parameter is. Dit is de extra inputvariable `a` van `PeronaMalik`. Merk op dat voor $\alpha = 2$ we de standaard Perona-Malik functies (2.5) en (2.6) terugvinden, terwijl de keuze $\alpha = 0$ ervoor zorgt dat de diffusie homogeen wordt.[1] De waarde van κ wordt gegeven door (2.9).

In deze sectie worden verschillende waarden van α getest om de invloed van deze parameter op de resultaten te onderzoeken. We nemen steeds dezelfde afbeelding als beginconditie u_0. We passen het numerieke schema (2.28) toe op u_0, voor verschillende waarden van α, ceteris paribus. De afbeelding u_0 die werd gekozen is te zien in Figuur 2.6. Deze afbeelding werd bekomen door aan de afbeelding uit Figuur 1.2 Gaussische witte ruis toe te voegen zoals uitgelegd in Sectie 1.4.4. Deze studie kan ook gemaakt worden voor andere types ruis. Merk wel op dat wij hier voor een relatief kleine variantie kiezen, namelijk 1/100, omdat anders de ruis moeilijk weg te filteren is en de resultaten moeilijker te vergelijken zijn.



Figuur 2.6: De afbeelding u_0 die ruis vertoont (Gaussische witte ruis met variantie $\sigma = 1/100$).

Er werd gewerkt met g_1 als de diffusiecoëfficiënt, de exponentiële variant. De resultaten bekomen met g_2 hebben weinig tot geen verschillen, in vergelijking met die bekomen onder g_1. Één resultaat onder g_2 is te zien in Figuur 2.7.

De parameters τ en T werden telkens dezelfde waarde gegeven. Er werd gekozen voor $T = 6$ en $\tau = 1/3$.[2] Met deze keuze van T en τ zijn er achttien tijdstippen waarop er een oplossing is berekend voor de differentiaalvergelijking, met name $t = 1/3, t = 2/3, \ldots, t = 18/3 = 6$.

Voor een keuze $\alpha = \bar{\alpha}$ zullen we de resultaten op een zo dynamisch mogelijke manier weergeven zoals beschreven in de volgende paragraaf. Voor negen van de achttien tijdstippen wordt het resultaat weergegeven in één enkele figuur die bestaat uit negen kleinere afbeeldingen. Deze negen afbeeldingen worden in een drie bij drie matrix weergegeven, waarbij de afbeelding in de linkerbovenhoek de afbeelding op het tijdstip $t = 2/3$ voorstelt en de afbeelding rechtsonder in de matrix het eindtijdstip $t = 6$ voorstelt. Hoe de afbeelding evolueert van begin tot einde

[1] Voor g_1 wordt de diffusiecoëfficiënt $e^{-1/2}$ terwijl voor g_2 deze 1/2 wordt.

[2] De kleine keuze van τ zorgt ervoor dat we goed zien wat er per stap gebeurt. De grote waarde van T is gekozen om te zien vanaf welk tijdstip we terug slechtere resultaten krijgen.

wordt afgelezen op de andere afbeeldingen. Voor de duidelijkheid wordt in onderstaande matrix getoond welke positie met welk tijdstip overeenkomt.

$$\begin{bmatrix} t = 2/3 & t = 4/3 & t = 2 \\ t = 8/3 & t = 10/3 & t = 4 \\ t = 14/3 & t = 16/3 & t = 6 \end{bmatrix}.$$

Voor de eenvoud is er dus telkens een tijdstip overgeslagen. Alle afbeeldingen weergeven in een achttien bij achttien matrix zou de afzonderlijke afbeeldingen te klein maken om hier te kunnen weergeven. Een keuze van T en τ zodat er juist negen tijdstippen zijn is ook mogelijk. Hier wordt bewust gekozen voor dubbel zoveel tijdstippen zodat de afbeelding niet te grof/snel evolueert.

Figuur 2.7: Evolutie van de Perona-Malik vergelijking op de afbeelding u_0 met parameters $\alpha = 2$, $T = 6$, $\tau = 1/3$ en (2.6) als keuze van g. Deze Figuur is louter ter aanvulling en wordt niet in detail besproken.

Figuur 2.8: Evolutie van de Perona-Malik vergelijking op de afbeelding u_0 met parameters $\alpha = 2$, $T = 6$, $\tau = 1/3$ en (2.5) als keuze van g.

Het resultaat van de hierboven beschreven methode voor $\alpha = 2$ is te zien in Figuur 2.8. Dit is dus een toepassing van de originele Perona-Malik vergelijking op de afbeelding te zien in Figuur 2.6. Het resultaat is consistent met onze verwachtingen, gebaseerd op het ontwerp en doelstelling van de Perona-Malik vergelijking. De ruis die aanwezig was, is vervaagd én de randen zijn verscherpt. Het beste resultaat toont zich op de tijdstippen $t = 8/3$ en $t = 10/3$. Dit zijn de eerste twee afbeeldingen op de tweede rij. De convergentie is duidelijk zichtbaar. Op de eerste rij is nog heel wat ruis aanwezig. Deze ruis is zo goed als volledig verdwenen op de tweede en laatste rij.

Om een eerste vergelijking te kunnen maken, passen we nu het schema toe met $\alpha = 0$. Door deze keuze van α wordt de diffusievergelijking homogeen. Het resultaat hiervan wordt gegeven in Figuur 2.9. Opnieuw is dit het resultaat zoals we zouden verwachten. Doordat het nu om een homogene diffusie ($\sim$ warmtevergelijking) gaat, vind de diffusie veel sneller plaats en wordt de afbeelding snel te wazig. Het verschil tussen de homogene diffusie (Figuur 2.9) en de Perona-Malik vergelijking (Figuur 2.8) is goed te zien als er op de letters *quiksilver* op het hoofddeksel gefocust wordt. Deze letters definiëren randen op de afbeelding en worden duidelijk behouden en verscherpt voor $\alpha = 2$ terwijl ze voor $\alpha = 0$ nog nauwelijks leesbaar zijn.

Figuur 2.9: Evolutie van de Perona-Malik vergelijking op de afbeelding u_0 met parameters $\alpha = 0$, $T = 6$, $\tau = 1/3$ en (2.5) als keuze van g.

Figuur 2.10: Evolutie van de Perona-Malik vergelijking op de afbeelding u_0 met parameters $\alpha = 1/2$, $T = 6$, $\tau = 1/3$ en (2.5) als keuze van g.

Vervolgens wordt hetzelfde gedaan voor $\alpha = 1/2$. Het resultaat hiervan is te zien in Figuur 2.10. Voor deze waarde van α was er vooraf niet echt een verwachting. Op de figuur is te zien dat het proces meer gelijkt op dat van $\alpha = 0$ dan op dat van $\alpha = 2$. Het resultaat is lichtjes beter dan de homogene diffusie, toch slechter als het vergeleken wordt met $\alpha = 2$. De randen worden hier niet zo goed bewaard als bij de Perona-Malik vergelijking. In volgende Figuur 2.11 is het resultaat te zien voor $\alpha = 1$. Om een goede vergelijking te kunnen maken, worden de figuren best telkens per twee vergeleken om eventuele verschillen te kunnen opmerken.

Wordt Figuur 2.11 ($\alpha = 1$) vergeleken met Figuur 2.9 ($\alpha = 0$), dan is er te zien dat de randen voor $\alpha = 1$ beter bewaard worden en dus als een beter resultaat kan beschouwd worden. Als de vergelijking wordt gemaakt tussen Figuur 2.11 en Figuur 2.8, is de Perona-Malik vergelijking duidelijk beter. Het verschil tussen Figuur 2.11 en 2.10 is minder duidelijk, toch wordt het resultaat onder $\alpha = 1$ net iets beter bevonden dan onder $\alpha = 1/2$.

Tenslotte wordt in Figuur 2.12 het resultaat gegeven voor $\alpha = 3/2$. Hier gelden dezelfde conclusies als voorgaande. Samenvattend kan er gepostuleerd worden dat: hoe groter α, hoe beter het resultaat.

Figuur 2.11: Evolutie van de Perona-Malik vergelijking op de afbeelding u_0 met parameters $\alpha = 1$, $T = 6$, $\tau = 1/3$ en (2.5) als keuze van g.

Figuur 2.12: Evolutie van de Perona-Malik vergelijking op de afbeelding u_0 met parameters $\alpha = 3/2$, $T = 6$, $\tau = 1/3$ en (2.5) als keuze van g.

Afgaand op de Figuren 2.8, 2.9, 2.10, 2.11 en 2.12, wordt het beste resultaat gevonden bij $\alpha = 2$ en wordt dit resultaat slechter naarmate α kleiner wordt. Een verklaring die hiervoor kan gegeven worden, is de betekenis van κ die telkens verandert als α wijzigt. We beschouwen opnieuw de flux functie $f(x) = g_1(|x|)x = e^{-\frac{1}{2}(\frac{|x|}{\kappa})^\alpha}x$. Met behulp van (2.8) berekenen we de eerste afgeleide

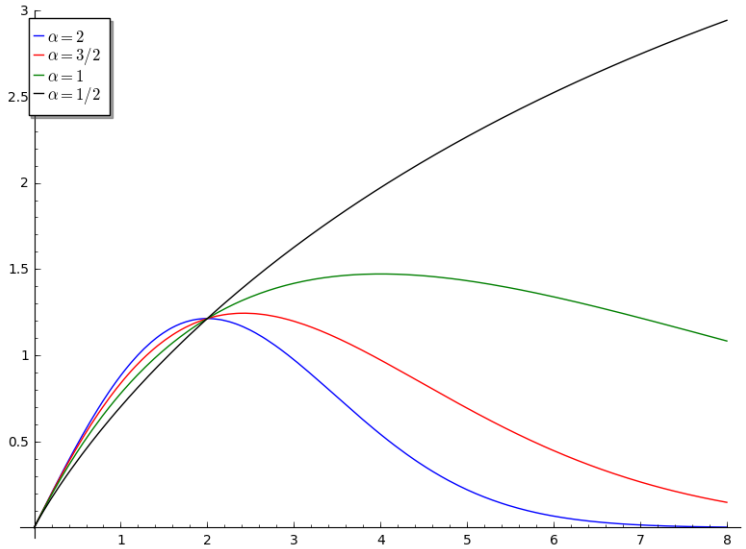
$$g_1'(x) = -\frac{\alpha x^{\alpha-1}}{2\kappa^\alpha} g_1(x) \;\Rightarrow\; f'(x) = \left(-\frac{\alpha|x|^\alpha}{2\kappa^\alpha} + 1\right) g_1(|x|).$$

Hieruit volgt dat f een extremum bereikt bij

$$x_{\pm} = \pm \left(\frac{2\kappa^{\alpha}}{\alpha} \right)^{\frac{1}{\alpha}}. \tag{2.39}$$

Doordat $f'(0) = 1$ en $\lim_{|x| \to \infty} f(x) = 0$ wordt bij x_+ een maximum en bij x_- een minimum bereikt.

Zoals te zien op Figuur 2.13, wordt het maximum behorend bij een grotere waarde van α eerder bereikt dan bij een kleinere waarde van α. Het maximum behorend bij $\alpha = 1/2$ wordt wegens (2.39) bereikt bij $x = 16\kappa$ en werd niet weergegeven op de figuur. Er valt tevens op te merken dat de functies f met $\alpha \neq 2$ door het maximum van f behorend bij $\alpha = 2$ gaan. Merk op dat voor $\alpha = 0$ er logischerwijze geen maximum kan bereikt worden en er dus steeds voorwaartse diffusie in het model plaatsvindt. De parameter κ bepaalt, als grenswaarde, waar er voorwaartse en achterwaartse diffusie plaatsvindt. Doordat α nu niet langer twee is, is deze grenswaarde verschoven. Bijvoorbeeld voor $\alpha = 1$ is dit 2κ. Dit betekent dat achterwaartse diffusie nu enkel plaatsvindt voor waarden van $|\nabla u|$ groter dan het dubbele van κ. Dit is een strengere eis dan bij de originele Perona-Malik vergelijking en zou als verklaring kunnen dienen waarom de resultaten minder worden voor afnemende α. In elk van de experimenten in deze sectie was κ immers constant en werd ze een waarde (2.9) toegekend, gebaseerd op de interpretatie voor het originele model met $\alpha = 2$.



Figuur 2.13: De functies $f(x) = x \exp\left(-\frac{|x|^{\alpha}}{2\kappa^{\alpha}}\right)$, voor vier verschillende waarden van α. Hierbij is κ gelijkgesteld aan twee.

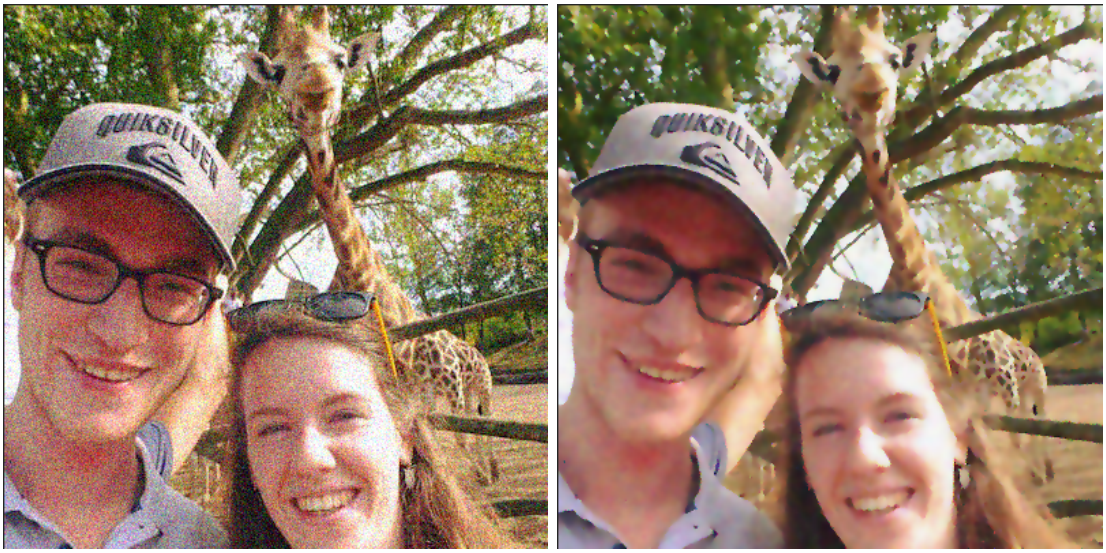
De aandachtige lezer zal zich afvragen of bovenstaande verklaring ook geldig is voor die andere diffusiefunctie (2.6). In dit geval bereikt de flux functie $f(x) = x(1 + (|x|/\kappa)^\alpha)^{-1}$ enkel een maximum voor waarden van α strikt groter dan één. Dit maximum wordt bereikt daar waar

$$|x|^\alpha = \frac{\kappa}{\alpha - 1}.$$

Dit betekent dat voor waarden van α waarvoor $0 \leq \alpha \leq 1$, enkel voorwaartse diffusie plaatsvindt. In de analyse met $g(x) = \exp(-\frac{1}{2}(\frac{|x|}{\kappa})^\alpha)$ was dit, op $\alpha = 0$ na, niet zo voor deze waarden van α. Toch vinden we dezelfde resultaten terug voor beide diffusiefuncties. Bij deze waarden van α vindt er nauwelijks achterwaartse diffusie plaats. Het maximum van de flux functie $f(x)$ wordt in dit geval bereikt bij een zeer grote waarde van x.

Kleur afbeeldingen

De Perona-Malik vergelijking hoeft niet enkel beperkt te blijven tot zwart-wit afbeeldingen. Een mogelijkheid om deze diffusievergelijking toe te passen op een kleur afbeelding, is door ze toe te passen op elk kanaal apart (RGB). Als voorbeeld tonen we dit voor de kleur versie van Figuur 2.6. Het resultaat is te zien in Figuur 2.14.



Figuur 2.14: Toepassing van de Perona-Malik vergelijking (rechts) op een kleur afbeelding met ruis (links). Parameters: $\tau = 1/3$, $T = 6$, $\alpha = 2$, optie 1 op elk kanaal. De afbeelding die rechts wordt getoond is deze op het eindtijdstip $t = T$.

Limiet voor $T \to \infty$

Tenslotte tonen we wat er gebeurt met een afbeelding als we de Perona-Malik vergelijking erop toepassen met een zeer grote eindtijd. De vergelijking is juist zo ontworpen dat de randen meer massa opnemen en alles wat niet als rand wordt aangeduid massa verliest. Uiteindelijk verwachten we dat dit proces convergeert naar een toestand die op een tekening lijkt. Als voorbeeld hebben we dit gedaan voor de afbeelding rechts uit Figuur 1.2. Het resultaat hiervan is te zien in Figuur 2.15. Dit soort afbeeldingen komen vaak terug in de literatuur bij het bestuderen van de Perona-Malik vergelijking [48, 27, 4, 50]. Het is tevens door dit soort gedrag dat de Perona-Malik vergelijking ook kan gebruikt worden om randen te detecteren.

Figuur 2.15: Toepassing van de Perona-Malik vergelijking op Figuur 1.2. Parameters: $\tau = 1/2, T = 100, \alpha = 2$, optie 1. Afbeelding getoond op eindtijdstip $t = T$.

De korte analyse in Sectie 2.1 leert ons dat de Perona-Malik vergelijking zich gedraagt als een voorwaartse en achterwaartse diffusievergelijking. Ondanks het feit dat de achterwaartse diffusie een slecht gesteld probleem (zie volgend Hoofdstuk) is, hebben we in het laatste deel van dit hoofdstuk mooie resultaten verkregen. De Perona-Malik vergelijking is handig in praktijk om (lichte) ruis weg te filteren en randen te detecteren.

3 Regularisatie

"If you think you are doing well, push harder because there is always something better."

Sven Nys

In Hoofdstuk 2 hebben we aangetoond dat de originele Perona-Malik vergelijking, voor praktische keuzes van g, zich lokaal kan gedragen als de achterwaartse warmtevergelijking (zie hiervoor Sectie 2.1). De achterwaartse warmtevergelijking is echter een klassiek voorbeeld van een slecht gesteld probleem[1] [37]. Deze observatie zorgt ervoor dat we de Perona-Malik vergelijking met een kritisch oog moeten bekijken.

Vooraleer we dit mogelijk probleem bespreken, formuleren we nog eens het volledige Perona-Malik model:

$$\begin{cases} \partial_t u(t,\mathbf{x}) - \nabla \cdot (g(|\nabla u(t,\mathbf{x})|)\nabla u(t,\mathbf{x})) = 0 & \text{in } (0,T]\times\Omega, \\ \nabla u(t,\mathbf{x})\cdot \boldsymbol{\nu} = 0 & \text{in } (0,T]\times\partial\Omega, \\ u(0,\mathbf{x}) = u_0(\mathbf{x}) & \text{in } \Omega. \end{cases} \tag{3.1}$$

Voor de functie g hebben we de volgende eigenschappen

$$\begin{gathered} g:\mathbb{R}_0^+ \to \mathbb{R}^+ \text{ is een niet-stijgende functie,} \quad g\in C^\infty(\mathbb{R}), \\ g(0)=1, \quad \text{en } g(x)\to 0 \text{ voor } x\to\infty. \end{gathered} \tag{3.2}$$

Een voorbeeld van zo'n functie is

$$g(x) = \frac{1}{1+(x/\kappa)^2}, \tag{3.3}$$

met een constante $\kappa > 0$.

Om uniciteit en existentie van de oplossing van (3.1) te garanderen, moet de flux functie

$$f(x) = g(|x|)x \tag{3.4}$$

[1] Een wiskundig probleem is goed gedefinieerd in de betekenis van Hadamard als er een unieke oplossing bestaat die continu afhangt van de data. Problemen die niet goed gedefinieerd zijn in de betekenis van Hadamard worden slecht gesteld genoemd.

monotoon stijgend zijn in x. Met behulp van de theorie van monotone operatoren [38, 43, 30, 54] en een aantal a priori afschattingen kan er aangetoond worden dat het probleem goed gedefinieerd is. Praktische keuzes van g zoals (2.5) en (2.6) leiden echter niet tot flux functies f die monotoon zijn. Er is (helaas) geen algemene theorie beschikbaar voor dergelijke gladde niet-monotone flux functies f. Een tegenvoorbeeld is gegeven in [16] waar een diffusieproces met niet-monotone flux functies werd geconstrueerd, dat een oneindig aantal oplossingen heeft. Ondanks dat het proces verschillend was van (3.1), was dit een stevige waarschuwing en werd de houding pessimistischer tegenover de Perona-Malik vergelijking vanuit de wiskundige wereld.

Vreemd genoeg werken praktische implementaties van het Perona-Malik proces dan weer veel beter dan verwacht wordt uit de theorie. De enige instabiliteit die geobserveerd wordt, is het zogenaamde *trapvorming effect*[1], waarbij een gladde rand evolueert in stuksgewijze segmenten die gescheiden worden door sprongen. Over dit effect zijn verscheidene studies uitgevoerd en deze resulteren allemaal in dezelfde conclusie: de oplossing hangt sterk af van het regularisatie effect van de discretisatie. In [49] wordt aangetoond dat het regulariserend effect van de standaard eindige differenties voldoende is om het probleem om te zetten in een goed gedefinieerd beginwaarde probleem. Dit is niets anders dan een stelsel van gewone differentiaalvergelijkingen, met als parameter de tijd t. Deze kunnen vervolgens opgelost worden met gepaste numerieke methoden [45]. De globale oplossing hiervan voldoet aan het maximum-principe en convergeert naar een evenwichtstoestand.

Ondanks het feit dat numerieke schema's impliciete regularisaties kunnen voorzien die het proces stabiel maken, lijkt het natuurlijker om de regularisatie rechtstreeks in de continue Perona-Malik vergelijking te introduceren. Op deze manier wordt ze onafhankelijker van de numerieke discretisatie.

Wegens bovenstaande bespreking zou er simpelweg gekozen kunnen worden voor hetzelfde model (3.1) maar dan met de eis dat de de flux van g monotoon moet zijn. Een voorbeeld van zo'n functie is

$$g(x) = \frac{1}{\sqrt{1 + (x/\kappa)^2}}, \tag{3.5}$$

met κ een positieve constante. Het moge duidelijk zijn dat diffusieprocessen met een monotone stijgende flux functie nooit de randen kunnen verscherpen. Dit is juist een eigenschap die voorop werd gesteld bij het ontwerpen van de Perona-Malik vergelijking. Bijgevolg is dit niet de regularisatie die we voor ogen hebben.

In de bewijzen omtrent uniciteit en existentie is het noodzakelijk dat $g(|\nabla u|)$ naar onder kan worden afgeschat door een positieve constante, i.e. er bestaat een $\mu > 0$ zodat

$$g(|\nabla u|) \geq \mu. \tag{3.6}$$

Wegens (3.2) is de waarde van $g(|\nabla u|)$ gelegen tussen 0 en 1, en is ze enkel gelijk aan 0 als $|\nabla u|$ onbegrensd is. Er kan onmogelijk gegarandeerd worden dat $|\nabla u| < \infty$. Er bestaat niet zoiets

[1] Engels: *staircasing effect*.

als een maximum-principe voor de gradiënt van de oplossing van een parabolische differentiaalvergelijking.

Om de gewenste eigenschap (3.6) toch te forceren, zouden we evengoed de functie g met een afstand ε verticaal kunnen opschuiven, i.e. beschouw $\tilde{g}(x) = g(x) + \varepsilon$ als diffusiefunctie. In de differentiaalvergelijking komt dit als volgt tot uiting

$$\partial_t u(t, \mathbf{x}) - \nabla \cdot ((g(|\nabla u(t, \mathbf{x})|) + \varepsilon)\nabla u(t, \mathbf{x})) = 0,$$

wat ook gelijk is aan

$$\partial_t u(t, \mathbf{x}) - \nabla \cdot (g(|\nabla u(t, \mathbf{x})|)\nabla u(t, \mathbf{x})) - \varepsilon \Delta u(t, \boldsymbol{x}) = 0.$$

We zien dat deze regularisatie niets anders is dan een heel klein beetje lineaire diffusie toevoegen met coëfficiënt ε.

Een wel heel vindingrijke manier is om $g(|\nabla u|)$ te vervangen door $g(|\nabla G_\sigma * u|)$, waarbij G_σ een Greense functie is,

$$G_\sigma(\boldsymbol{x}) = \frac{1}{4\pi\sigma} \exp\left(-\frac{|x|^2}{4\sigma}\right). \tag{3.7}$$

Dit idee werd voorgesteld in [8], waar de auteurs de existentie en uniciteit van de oplossing bewijzen volgens deze regularisatie. Merk op dat (3.7) een fundamentele oplossing is van de warmte vergelijking [37]. Doordat $\nabla G_\sigma * u \to \nabla u$ als $\sigma \to 0$,[1] leunt dit model zeer dicht aan bij de originele formulering voor kleine waarden van σ. Hierdoor behoudt het ook alle praktische eigenschappen van (3.1). Het begrenst tevens de gradiënt van de oplossing in de input van g, i.e. het probleem (3.6). Dit zorgt ervoor dat in de numerieke implementaties de gradiënten, geëvalueerd op een discreet rooster, eindig zijn.

Tot op vandaag wordt laatst genoemde regularisatie het meeste gebruikt in de praktijk. We zullen deze dan ook uitvoerig bestuderen.

3.1 Geregulariseerd model

De voorgestelde regularisatie in [8] levert het volgende model op

$$\begin{cases} \partial_t u(t, \mathbf{x}) - \nabla \cdot (g(|\nabla G_\sigma * u(t, \mathbf{x})|)\nabla u(t, \mathbf{x})) = 0 & \text{in } (0, T] \times \Omega, \\ \nabla u(t, \mathbf{x}) \cdot \boldsymbol{\nu} = 0 & \text{in } (0, T] \times \partial\Omega, \\ u(0, \mathbf{x}) = u_0(\mathbf{x}) & \text{in } \Omega, \end{cases} \tag{3.8}$$

waarbij $\Omega \subset \mathbb{R}^2$ nog steeds een eindig begrensd rechthoekig domein is en $\boldsymbol{\nu}$ de uitwaartse normaalvector op $\partial\Omega$. De functie g voldoet aan de eigenschappen opgesomd in (3.2).

[1] Omdat $G_\sigma \to \delta$, met δ de Dirac distributie.

De beschouwde functie G_σ hoeft niet noodzakelijk (3.7) te zijn. Ze mag elke gladde kern[1] zijn, al is gebruikelijk dat ze afhankelijk is van een parameter σ. De functie G_σ die wij beschouwen voldoet aan

$$\begin{gathered} G_\sigma \in C^\infty(\mathbb{R}^2), \quad \int_{\mathbb{R}^2} G_\sigma d\mathbf{x} = 1, \quad \int_{\mathbb{R}^2} |\nabla G_\sigma|^2 dx \leq C_\sigma, \\ G_\sigma(\mathbf{x}) \to \delta_x, \text{ voor } \sigma \to 0, \quad \delta_x \text{ is de Dirac distributie in het punt } \mathbf{x}. \end{gathered} \tag{3.9}$$

Met de term $\nabla G_\sigma * u$ in (3.8) bedoelen we

$$(\nabla G_\sigma * u)(\boldsymbol{x}, t) = \int_{\mathbb{R}^2} \nabla G_\sigma(\mathbf{x} - \boldsymbol{\varepsilon}) \tilde{u}(\boldsymbol{\varepsilon}, t) d\boldsymbol{\varepsilon},$$

met $\tilde{u}$ een continue uitbreiding van u tot $\mathbb{R}^2$. Zo'n uitbreiding bestaat wegens [6][Stelling 2.1]. Deze stelling geeft ons bovendien dat er positieve constanten C_1, C_2 bestaan zodat

$$\|\tilde{u}\|_{L^2(\mathbb{R}^2)} \leq C_1 \|u\|_{L^2(\Omega)}, \tag{3.10}$$

$$\|\nabla \tilde{u}\|_{L^2(\mathbb{R}^2)} \leq C_2 \|\nabla u\|_{L^2(\Omega)}. \tag{3.11}$$

In [8] wordt een bewijs geleverd voor het bestaan en uniek zijn van de zwakke oplossing u van (3.8) zodat $u \in C([0,T], L^2(\Omega)) \cap L^2((0,T), H^1(\Omega))$. Om de existentie te bewijzen, wordt er gebruik gemaakt van de fixpuntstelling van Schauder [38]. Wij zullen echter gebruik maken van de zogenaamde Rothemethode, zoals in [18]. Het doel van de Rothemethode is om de existentie van een zwakke oplossing van het probleem (3.8) aan te tonen. Bijgevolg moet de variationele formulering van dit probleem goed gedefinieerd zijn (elke term moet eindig zijn). Bij het opstellen van de variationele formulering is het gebruikelijk om enige informatie over de oplossing te bekomen door zijn natuurlijke stabiliteit te bestuderen. We doen dit door voorlopig te veronderstellen dat er een oplossing u van het probleem (3.8) bestaat.

Vermenigvuldigen we de partiële differentiaalvergelijking in (3.8) met u en integreren deze over het gebied Ω, dan krijgen we

$$\int_\Omega u \partial_t u d\boldsymbol{x} - \int_\Omega u \nabla \cdot (g(|\nabla G_\sigma * u|) \nabla u) d\boldsymbol{x} = 0.$$

Op de tweede term passen we de Stelling van Green toe. Rekening houdend met de Neumann randconditie in (3.8) levert dit ons

$$\int_\Omega u \partial_t u d\boldsymbol{x} + \int_\Omega g(|\nabla G_\sigma * u|) |\nabla u|^2 d\boldsymbol{x} = 0. \tag{3.12}$$

Merk op dat $u \partial_t u = \frac{1}{2} \partial_t (u^2)$. We integreren vergelijking (3.12) in de tijd t over $(0, \eta) \subset [0, T]$ en bekomen

$$\frac{1}{2} \int_0^\eta \int_\Omega \partial_t (u^2) d\boldsymbol{x} dt + \int_0^\eta \int_\Omega (g(|\nabla G_\sigma * u|) |\nabla u|^2 d\boldsymbol{x}\, dt = 0. \tag{3.13}$$

[1] Engels: *smoothing kernel*.

Vanaf nu noteren we het inproduct op $L^2(\Omega)$ met $(\cdot,\cdot)$ en zijn geïnduceerde norm met $\|\cdot\|$.

Door aan te nemen dat alle termen in (3.13) eindig zijn, is de eerste term gelijk aan $\frac{1}{2}\|u(\eta)\|^2 - \|u(0)\|^2$. De tweede term in (3.13) is steeds positief zodat

$$\begin{aligned}&\Leftrightarrow \|u(\eta)\|^2 \leq \|u_0\|^2\\ &\Leftrightarrow \|u(\eta)\| \leq \|u_0\|.\end{aligned}$$

De laatste pijl volgt uit het feit dat een norm altijd positief is. De norm $\|u_0\|$ is dus een bovengrens voor $\|u(\eta)\|$, voor bijna elke η in $[0,T]$. Doordat η willekeurig was, kunnen we het maximum nemen over alle $t \in [0,T]$

$$\max_{t\in[0,T]} \|u(t)\| \leq \|u_0\|. \tag{3.14}$$

Merk nu op dat als u een oplossing is van (3.8), dan geldt wegens de Cauchy-ongelijkheid, (3.9), (3.10) en (3.14) dat

$$\begin{aligned}|(\nabla G_\sigma * u)(\boldsymbol{x},t)| &= \left|\int_{\mathbb{R}^2} \nabla G_\sigma(\mathbf{x}-\boldsymbol{\varepsilon})\tilde{u}(\boldsymbol{\varepsilon},t)d\boldsymbol{\varepsilon}\right|,\\ &\leq \sqrt{\int_{\mathbb{R}^2} |\nabla G_\sigma(\mathbf{x}-\boldsymbol{\varepsilon})|^2\, d\boldsymbol{\varepsilon}}\sqrt{\int_{\mathbb{R}^2} |\tilde{u}(\boldsymbol{\varepsilon},t)|^2 d\boldsymbol{\varepsilon}},\\ &\leq C\|u(t)\|,\\ &\leq C\|u_0\|,\end{aligned} \tag{3.15}$$

wat eindig is als $u_0 \in L^2(\Omega)$. Dus bestaat er $\mu > 0$ zodat

$$g(|\nabla G_\sigma * u|) > \mu, \quad \forall t \in [0,T], \tag{3.16}$$

op voorwaarde dat $u_0 \in L^2(\Omega)$.

Gebruiken we deze afschatting (3.16) in vergelijking (3.13), dan bekomen we

$$\frac{1}{2}\|u(\eta)\|^2 + \mu\int_0^\eta \|\nabla u\|^2 dt \leq \frac{1}{2}\|u_0\|^2. \tag{3.17}$$

We vinden uiteindelijk de volgende a priori afschatting

$$\max_{t\in[0,T]} \|u(t)\|^2 + \int_0^T \|\nabla u\|^2 dt \leq C(\|u_0\|^2). \tag{3.18}$$

De afschatting (3.18) geeft ons geen informatie over $\partial_t u$. Er geldt dat

$$(\partial_t u(t), \varphi) = -(g(|\nabla G_\sigma * u(t)|)\nabla u(t), \nabla\varphi)$$

voor alle $\varphi \in H^1(\Omega)$ en voor bijna elke $t \in [0,T]$. Hierdoor kunnen we $\partial_t u(t)$ zien als een operator van $H^1(\Omega)$ naar $\mathbb{R}$. De duale norm in $H^1(\Omega)^*$ is gedefinieerd als

$$\|\partial_t u(t)\|_{H^1(\Omega)^*} = \sup_{\varphi\in H^1(\Omega)} \frac{(\partial_t u(t),\varphi)}{\|\varphi\|_{H^1(\Omega)}}.$$

Er geldt

$$\|\partial_t u(t)\|_{H^1(\Omega)^*} \leq \|\nabla u(t)\|.$$

Daardoor, gebruik makend van (3.18), geldt er dat

$$\int_0^T \|\partial_t u(s)\|^2_{H^1(\Omega)^*} ds \leq C(\|u_0\|^2). \tag{3.19}$$

Dit betekent dat $\partial_t u \in L^2((0,T), H^1(\Omega)^*)$ als $u_0 \in L^2(\Omega)$.

De afschattingen (3.18) en (3.19) tonen aan dat de variationele formulering afhankelijk is van de assumpties op de beginconditie u_0.

Definitie 3.1. *De variationele formulering van probleem (3.8) wordt gegeven als:*

$$\textit{Gegeven } u_0 \in L^2(\Omega), \textit{ vind } u(t) \in H^1(\Omega) \textit{ met } \partial_t u(t) \in H^1(\Omega)^* \textit{ zodat}$$

$$u \in L^2((0,T), H^1(\Omega)), \ \partial_t u \in L^2((0,T), H^1(\Omega)^*) \textit{ en}$$

$$(\partial_t u(t), \varphi) + (g(|\nabla G_\sigma * u(t)|)\nabla u(t), \nabla \varphi) = 0, \tag{3.20}$$

$$\textit{voor alle } \varphi \in H^1(\Omega) \textit{ en } t \in [0,T] \textit{ b.o.}$$

In de rest van dit hoofdstuk gaan we op zoek naar een unieke oplossing van de variationele formulering in Definitie 3.1. De tijdsafgeleide in vergelijking (3.20) wordt geïnterpreteerd als de veralgemeende afgeleide op $[0,T]$, hierdoor moet de vergelijking enkel voldaan zijn voor bijna elke $t \in [0,T]$. De oplossingsmethode is onderverdeeld in volgende stappen:

- uniciteit van de oplossing;
- oplossen van elliptische vergelijkingen;
- a priori afschattingen;
- convergentie van de Rothe functies (compactheid argument).

Hiervoor zullen we beroep doen op een aantal lemma's die we opgelijst hebben in Appendix B.

3.2 Uniciteit van de oplossing

In deze sectie bewijzen we de uniciteit van de oplossing. Merk daartoe op dat als u een oplossing is van (3.20), dan geldt ongelijkheid (3.16).

Stelling 3.2. *(Uniciteit). De oplossing $u \in L^2((0,T), H^1(\Omega))$ van (3.20) is uniek.*

Bewijs. Stel dat we twee oplossingen hebben u_1 en $u_2 \in L^2((0,T), H^1(\Omega))$. Dan voldoen u_1 en u_2 aan (3.20). Door φ gelijk te stellen aan $u_1(t) - u_2(t)$ is ook voldaan aan

$$\begin{aligned}(\partial_t(u_1(t) - u_2(t)), u_1(t) - u_2(t)) + (g_1 \nabla[u_1(t) - u_2(t)], \nabla[u_1(t) - u_2(t)]) \\ = ([g_2(t) - g_1(t)]\nabla u_2(t), \nabla[u_1(t) - u_2(t)]).\end{aligned}$$

Hierbij is $g_j = g(|\nabla G_\sigma * u_j|)$ voor $j = 1, 2$.

Door deze identiteit te integreren over de tijd $t \in (0, \eta) \subset (0, T)$ en het toepassen van Lemma B.21 en Lemma B.3 vinden we

$$\begin{aligned}\frac{1}{2}\|u_1(\eta) - u_2(\eta)\|^2 + \mu \int_0^\eta \|\nabla[u_1(t) - u_2(t)]\|^2 dt \leq C_\varepsilon \int_0^\eta \|g_1(t) - g_2(t)\|^2_{L^\infty(\Omega)} \|\nabla u_2(t)\|^2 dt \\ + \varepsilon \int_0^\eta \|\nabla[u_1(t) - u_2(t)]\|^2 dt.\end{aligned}$$

Doordat $g, G \in C^\infty$ hebben we dat $\|g_1(t) - g_2(t)\|^2_{L^\infty(\Omega)} \leq C\|u_1(t) - u_2(t)\|^2_{L^2(\Omega)}$. Inderdaad want wegens de Cauchy-ongelijkheid is

$$\begin{aligned}|g_1(t, \boldsymbol{x}) - g_2(t, \boldsymbol{x})| &= |\nabla G_\sigma * u_1(t, \boldsymbol{x}) - \nabla G_\sigma * u_2(t, \boldsymbol{x})| \\ &= |\nabla G_\sigma * (u_1 - u_2)(t, \boldsymbol{x})| \\ &\leq C_\sigma \|u_1(t) - u_2(t)\|. \end{aligned} \tag{3.21}$$

Door het maximum te nemen over alle $\boldsymbol{x} \in \Omega$ volgt $\|g_1(t) - g_2(t)\|^2_{L^\infty(\Omega)} \leq C\|u_1(t) - u_2(t)\|^2_{L^2(\Omega)}$.

We krijgen aldus (voor voldoende kleine $\varepsilon < \mu$)

$$\|u_1(\eta) - u_2(\eta)\|^2 + \int_0^\eta \|\nabla[u_1(t) - u_2(t)]\|^2 dt \leq C \int_0^\eta \|u_1(t) - u_2(t)\|^2 \|\nabla u_2(t)\|^2 dt, \quad \forall \eta \in [0, T]. \tag{3.22}$$

Op dit punt moeten we gebruik maken van Grönwall (Lemma B.5 (i)). Definieer

$$\begin{aligned}y(\eta) &= \|u_1(\eta) - u_2(\eta)\|^2, \\ r(\eta) &= C\|\nabla u_2(\eta)\|^2, \\ h(\eta) &= 0.\end{aligned}$$

De tweede term in het linkerlid van (3.22) is steeds positief, zodat

$$y(\eta) \leq \int_0^\eta r(t) y(t) dt, \quad \forall \eta \in [0, T].$$

Nu is $\int_t^\eta r(s)ds = C\int_t^\eta \|\nabla u_2(s)\|^2 ds < \infty$ omdat $u_2 \in L^2((0,T),H^1(\Omega))$. Toepassen van Grönwall (Lemma B.5 (i)) geeft

$$\|u_1(\eta) - u_2(\eta)\|^2 \leq 0, \quad \forall \eta \in [0,T].$$

Dit impliceert dat $\|u_1(\eta) - u_2(\eta)\|^2 = 0$ voor bijna elke $\eta \in [0,T]$. Hierdoor is de oplossing uniek, i.e. $u_1 = u_2$ b.o. in $\Omega \times [0,T]$. □

3.3 Existentie van de oplossing

Het tijdsinterval $[0,T]$ is verdeeld in $n \in \mathbb{N}$ equidistante deelintervallen $(t_{i-1}, t_i]$ met tijdstap $\tau = \frac{T}{n}$, d.w.z. $t_i = i\tau$, $i = 1,\ldots,n$. De achterwaartse Euler methode wordt gebruikt om de tijdsafgeleide te benaderen. De volgende standaard notaties voor de discrete velden worden gebruikt

$$u_i = u_i(\mathbf{x}) \approx u(\mathbf{x}, t_i) \quad \text{en} \quad \partial_t u(t_i) \approx \delta u_i = \frac{u_i - u_{i-1}}{\tau}.$$

Voor $i = 1,\ldots,n$ worden de volgende elliptische vergelijkingen opgelost

$$\delta u_i - \nabla \cdot (g(|\nabla G_\sigma * u_{i-1}|)\nabla u_i) = 0 \text{ in } \Omega, \tag{3.23}$$

op voorwaarde dat de oplossing u_{i-1} op het vorige tijdstip gekend is.

De variationele formulering van (3.23) luidt dan (zet $t = t_i$ in (3.20)):

$$\text{Gegeven } u_0 \in L^2(\Omega), \text{ vind } u_i \in H^1(\Omega) \text{ zodat}$$

$$(\delta u_i, \varphi) + (g(|\nabla G_\sigma * u_{i-1}|)\nabla u_i, \nabla\varphi) = 0, \quad \forall \varphi \in H^1(\Omega). \tag{3.24}$$

Dit probleem is equivalent met het oplossen van de vergelijking

$$a_i(u_i, \varphi) = f_{i-1}(\varphi), \quad \forall \varphi \in H^1(\Omega),$$

voor elke $i = 1,\ldots,n$, waarbij

$$a_i : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R} : (u,v) \mapsto \left(\frac{u}{\tau}, v\right) + (g(|\nabla G_\sigma * u_{i-1}|)\nabla u, \nabla v),$$
$$f_{i-1} : H^1(\Omega) \to \mathbb{R} : \varphi \mapsto \left(\frac{u_{i-1}}{\tau}, \varphi\right).$$

Stelling 3.3. *Stel dat $u_0 \in L^2(\Omega)$. Dan bestaat er voor het variationeel probleem (3.24) een unieke oplossing $u_i \in H^1(\Omega)$ voor elke $i = 1,\ldots,n$.*

Bewijs. De bilineaire vorm a_i is continu in $H^1(\Omega)$ want

$$\begin{aligned} |a_i(u,v)| &\leq \frac{1}{\tau}\|u\|\|v\| + \|\nabla u\|\|\nabla v\| \\ &\leq \frac{1}{\tau}\|u\|_{H^1(\Omega)}\|v\|_{H^1(\Omega)} + \|u\|_{H^1(\Omega)}\|v\|_{H^1(\Omega)} \\ &= \left\{\frac{1}{\tau} + 1\right\}\|u\|_{H^1(\Omega)}\|v\|_{H^1(\Omega)}. \end{aligned}$$

Merk op dat als $u_{i-1} \in H^1(\Omega) \subset L^2(\Omega)$, dan geldt er dat $|\nabla G_\sigma * u_{i-1}| \leq \|\nabla G_\sigma\| \|u_{i-1}\| < \infty$ zodat $g(|\nabla G_\sigma * u_{i-1}|) > \mu$, voor een zekere $\mu > 0$.

De bilineaire vorm a_i is bijgevolg $H^1(\Omega)$-elliptisch als $u_{i-1} \in L^2(\Omega)$:

$$\begin{aligned} a_i(u,u) &\geq \frac{1}{\tau}\|u\|^2 + \mu\|\nabla u\|^2 \\ &\geq \min\left\{\frac{1}{\tau}, \mu\right\}\left(\|u\|^2 + \|\nabla u\|^2\right) \\ &= \min\left\{\frac{1}{\tau}, \mu\right\}\|u\|^2_{H^1(\Omega)}. \end{aligned}$$

De functionaal f_{i-1} is lineair en begrensd in $H^1(\Omega)$ als $u_0 \in L^2(\Omega)$.

De existentie en uniciteit van $u_i \in H^1(\Omega)$, $i = 1, \ldots, n$ volgt uit de Stelling van Lax-Milgram (zie Lemma B.29) als $u_0 \in L^2(\Omega)$. □

3.3.1 A priori afschattingen

In dit deel bewijzen we enkele resultaten omtrent de stabiliteit van u_i. Deze hangen af van de regulariteit van de beginconditie u_0.

Lemma 3.4. *(A priori afschatting). Stel dat $u_0 \in L^2(\Omega)$. Dan bestaat er een positieve constante C zodat*

$$\max_{1\leq j\leq n}\|u_j\|^2 + \sum_{i=1}^{n}\|u_i - u_{i-1}\|^2 + \sum_{i=1}^{n}\|u_i\|^2_{H^1(\Omega)}\tau \leq C.$$

Bewijs. Zet $\varphi = u_i\tau$ in (3.24) en sommeer over $i = 1, \ldots, j$, met $1 \leq j \leq n$. We verkrijgen

$$\sum_{i=1}^{j}(\delta u_i, u_i)\tau + \sum_{i=1}^{j}(g(|\nabla G_\sigma * u_{i-1}|)\nabla u_i, \nabla u_i)\tau = 0.$$

Voor de eerste term maken we gebruik van Abel's sommatie-regel (zie Lemma B.7)

$$\sum_{i=1}^{j}(\delta u_i, u_i)\tau = \frac{1}{2}\|u_j\|^2 - \frac{1}{2}\|u_0\|^2 + \frac{1}{2}\sum_{i=1}^{j}\|u_i - u_{i-1}\|^2.$$

Uit Stelling 3.3 volgt dat $u_i \in H^1(\Omega)$ voor $i = 1, \ldots, n$. Dus geldt voor alle $i = 0, \ldots, n$ dat er een $\mu > 0$ bestaat zodat $g(|\nabla G_\sigma * u_i|) > \mu$. We krijgen hierdoor volgende ongelijkheid

$$\|u_j\|^2 + \sum_{i=1}^{j}\|u_i - u_{i-1}\|^2 + \sum_{i=1}^{j}\|\nabla u_i\|^2\tau \leq C(\|u_0\|),$$

dit geeft het te bewijzen. □

Vergelijking (3.24) definieert

$$(\delta u_i, \varphi) = -(g(|\nabla G_\sigma * u_{i-1}|)\nabla u_i, \nabla\varphi), \quad \forall\varphi \in H^1(\Omega).$$

Hierdoor kan δu_i gezien worden als een element van $H^1(\Omega)^*$. Gebruik makend van

$$\|\delta u_i\|_{H^1(\Omega)^*} = \sup_{\varphi\in H^1(\Omega)} \frac{(\delta u_i, \varphi)}{\|\varphi\|_{H^1(\Omega)}}$$

en Lemma 3.4, is

$$\sum_{i=1}^{n} \|\delta u_i\|^2_{H^1(\Omega)^*}\tau \leq C. \tag{3.25}$$

3.3.2 Convergentie

Tot nu toe hebben we de existentie van de u_i voor $i = 1, \ldots, n$ bewezen. Lemma 3.4 geeft ons nuttige informatie over de stabiliteit van u_i, welke uniform is met betrekking tot de lopende index i. Nu verlengen we de functies u_i naar het hele interval $[0, T]$, dit worden de zogenaamde Rothe functies genoemd. We definiëren de stuksgewijze, lineair in de tijd, functies u_n

$$u_n(t) = \begin{cases} u_0 & \text{voor } t = 0 \\ u_{i-1} + (t - t_{i-1})\delta u_i & \text{voor } t \in (t_{i-1}, t_i] \end{cases} \tag{3.26}$$

en de stuksgewijs constante functies $\bar{u}_n$

$$\bar{u}_n(t) = \begin{cases} u_0 & \text{voor } t = 0 \\ u_i & \text{voor } t \in (t_{i-1}, t_i]. \end{cases} \tag{3.27}$$

De variationele formulering (3.24) kan geschreven worden in termen van u_n en $\bar{u}_n$ als volgt

$$(\partial_t u_n(t), \varphi) + (g(|\nabla G_\sigma * \bar{u}_n^\tau(t)|)\nabla\bar{u}_n(t), \nabla\varphi) = 0, \tag{3.28}$$

waarbij $\bar{u}_n^\tau(t) := \bar{u}_n(t-\tau)$. Nu bewijzen we de convergentie van de rijen u_n en $\bar{u}_n$ naar de unieke zwakke oplossing van (3.8) als $\tau \to 0$ of $n \to \infty$.

De a priori afschattingen uit Lemma 3.4 en (3.25) kunnen herschreven worden voor elke $n \in \mathbb{N}$ als

$$\begin{aligned} &\max_{t\in[0,T]} \|u_n(t)\|^2 + \max_{t\in[0,T]} \|\bar{u}_n(t)\|^2 + \int_0^T \|\bar{u}_n(s)\|^2_{H^1(\Omega)} ds \\ &+ \int_0^T \|\partial_t u_n(s)\|^2_{H^1(\Omega)^*} ds + \sum_{i=1}^{n} \left\| \int_{t_{i-1}}^{t_i} \partial_t u_n(s) ds \right\|^2 \leq C. \end{aligned} \tag{3.29}$$

De a priori afschattingen impliceren tevens een zekere relatie tussen beide Rothe functies. Merk op dat $u_n(0) - \bar{u}_n(0) = 0$. Voor alle $t \in (t_{i-1}, t_i]$ met $1 \leq i \leq n$, geldt er dat

$$\begin{aligned} |u_n(t) - \bar{u}_n(t)| &= |u_{i-1} + (t - t_{i-1})\delta u_i - u_i| \\ &= |(t - t_{i-1})\delta u_i - \delta u_i \tau| \\ &= |(t - t_i)\delta u_i| \\ &\leq \tau |\delta u_i| \qquad (3.30) \\ &= |u_i - u_{i-1}|. \qquad (3.31) \end{aligned}$$

Gebruik makend van Lemma 3.4 (als $u_0 \in L^2(\Omega)$) hebben we dat:

Lemma 3.5. *De Rothe functies $\bar{u}_n$ en u_n hebben dezelfde limiet in $L^2((0,T), L^2(\Omega))$.*

Bewijs. Wegens (3.30) en Lemma 3.4 is

$$\begin{aligned} \int_0^T \|u_n - \bar{u}_n\|^2 dt &= \sum_{i=1}^n \int_{t_{i-1}}^{t_i} \|u_n - \bar{u}_n\|^2 dt \\ &= \sum_{i=1}^n \int_{t_{i-1}}^{t_i} \|u_i - u_{i-1}\|^2 dt \\ &\leq \tau \sum_{i=1}^n \|u_i - u_{i-1}\|^2 \\ &\leq \frac{C}{n}, \end{aligned}$$

zodat $\|u_n - \bar{u}_n\|^2_{L^2((0,T),L^2)} \to 0$, als $n \to \infty$. □

De volgende stap is om de limiet $n \to \infty$ te nemen in (3.28). Om dit te doen zullen we een compactheid argument en verscheidene convergentie regels nodig hebben.

Lemma 3.6. *Zij $u_0 \in L^2(\Omega)$. Dan bestaat er een $u \in C([0,T], H^1(\Omega)^*)$ zodat*

$$u_{n_k} \to u \quad \textit{als } k \to \infty,$$

met $\{u_{n_k}\}$ een deelrij van $\{u_n\}$.

Bewijs. Het compactheid argument is

$$H^1(\Omega) \hookrightarrow\hookrightarrow L^2(\Omega) \cong L^2(\Omega)^* \hookrightarrow\hookrightarrow H^1(\Omega)^*,$$

zie hiervoor de Rellich-Kondrachov Stelling B.22.

De uniforme begrensdheid van $u_n(t)$ in $L^2(\Omega)$ (3.29) en de compacte inbedding $L^2(\Omega) \hookrightarrow\hookrightarrow H^1(\Omega)^*$ impliceren dat er een deelrij $\{u_{n_k}(t)\}$ van $u_n(t)$ bestaat zodat

$$u_{n_k}(t) \to u(t) \quad \text{in } H^1(\Omega)^*, \forall t \in [0,T], \quad k \to \infty.$$

Dus is bijgevolg $\{u_n(t)\}$ uniform begrensd in $H^1(\Omega)^*$. Bovendien is u_n uniform equicontinu, want voor alle $t, s \in [0,T]$ geldt er

$$\begin{aligned}
\|u_n(t) - u_n(s)\|_{H^1(\Omega)^*} &= \left\| \int_s^t \partial_t u(\eta) d\eta \right\|_{H^1(\Omega)^*} \\
&\leq \int_s^t \|\partial_t u(\eta)\|_{H^1(\Omega)^*} d\eta \\
&\leq \sqrt{\int_s^t 1^2 d\eta} \sqrt{\int_s^t \|\partial_t u(\eta)\|^2_{H^1(\Omega)^*} d\eta} \\
&\leq \sqrt{\int_s^t 1^2 d\eta} \sqrt{\int_0^T \|\partial_t u(\eta)\|^2_{H^1(\Omega)^*} d\eta} \\
&\leq C|t-s|^{\frac{1}{2}}.
\end{aligned}$$

De veralgemeende Arzelà-Ascoli Stelling B.26 impliceert dan het bestaan van een deelrij $\{u_{n_k}\}$ van $\{u_n\}$ zodat

$$u_{n_k}(t) \to u(t) \quad \text{in } C([0,T], H^1(\Omega)^*).$$

□

Lemma 3.7. *Zij u zoals in Lemma 3.6. Dan gelden volgende zwakke convergenties*

$$\begin{cases}
u_{n_k}(t) \rightharpoonup u(t), \quad \textit{in} \quad L^2(\Omega) \textit{ voor alle } t \in [0,T], & \text{(3.32a)} \\
\bar{u}_{n_k}(t) \rightharpoonup u(t), \quad \textit{in} \quad L^2(\Omega) \textit{ voor alle } t \in [0,T], & \text{(3.32b)} \\
\bar{u}_{n_k} \rightharpoonup u, \quad \textit{in} \quad L^2((0,T), H^1(\Omega)), & \text{(3.32c)} \\
\partial_t u_{n_k} \rightharpoonup \partial_t u, \quad \textit{in} \quad L^2((0,T), H^1(\Omega)^*). & \text{(3.32d)}
\end{cases}$$

Hierbij is $\{u_{n_k}\}$ dezelfde deelrij van $\{u_n\}$ als in Lemma 3.6.

Bewijs. Uit (3.29) weten we dat $\|u_n(t)\| \leq C$. De reflexiviteit van $L^2(\Omega)$ impliceert dan wegens Lemma B.28

$$u_{n_k}(t) \rightharpoonup u(t), \quad \text{in} \quad L^2(\Omega) \text{ voor alle } t \in [0,T].$$

Wegens Lemma 3.5 volgt hieruit ook dat

$$\bar{u}_{n_k}(t) \rightharpoonup u(t), \quad \text{in} \quad L^2(\Omega) \text{ voor alle } t \in [0,T].$$

Dankzij $\int_0^T \|\bar{u}_n(s)\|^2_{H^1(\Omega)} ds \leq C$ en $\int_0^T \|\partial_t u_n(s)\|^2_{H^1(\Omega)^*} ds \leq C$ uit (3.29) en het feit dat deze ruimten separabele Hilbert ruimten zijn, volgt er dat (opnieuw wegens Lemma B.28)

$$\bar{u}_{n_k} \rightharpoonup u, \quad \text{in} \quad L^2((0,T), H^1(\Omega)),$$

en

$$\partial_t u_{n_k} \rightharpoonup \partial_t u, \quad \text{in} \quad L^2((0,T), H^1(\Omega)^*).$$

□

Lemma 3.8. *Zij $u_0 \in L^2(\Omega)$, u zoals in Lemma 3.6, dan geldt er dat*

$$\bar{u}_{n_k} \to u \quad \text{in } L^2((0,T),L^2(\Omega)) \text{ als } k \to \infty.$$

Bewijs. Er geldt dat

$$\begin{aligned}\|\bar{u}_{n_k}(t)-u(t)\|^2 &= \langle \bar{u}_{n_k}(t)-u(t), \bar{u}_{n_k}(t)-u(t)\rangle_{H^1(\Omega)^*\times H^1(\Omega)} \\ &\le \|\bar{u}_{n_k}(t)-u(t)\|_{H^1(\Omega)^*}\|\bar{u}_{n_k}(t)-u(t)\|_{H^1(\Omega)}.\end{aligned}$$

Zodat

$$\begin{aligned}\int_0^T \|\bar{u}_{n_k}(t)-u(t)\|^2 dt &\le \int_0^T \|\bar{u}_{n_k}(t)-u(t)\|_{H^1(\Omega)^*}\|\bar{u}_{n_k}(t)-u(t)\|_{H^1(\Omega)}dt \\ &\le \sqrt{\int_0^T \|\bar{u}_{n_k}(t)-u(t)\|^2_{H^1(\Omega)^*}dt}\sqrt{\int_0^T \|\bar{u}_{n_k}(t)-u(t)\|^2_{H^1(\Omega)}dt} \\ &\le C\sqrt{\int_0^T \|\bar{u}_{n_k}(t)-u(t)\|^2_{H^1(\Omega)^*}dt},\end{aligned}$$

omdat (3.32c) (zie Lemma B.24 (iv)). Door nu gebruik te maken van $(a+b)^2 \le 2(a^2+b^2)$ vinden we dat

$$\int_0^T \|\bar{u}_{n_k}(t)-u(t)\|^2 dt \le C\sqrt{\int_0^T \|\bar{u}_{n_k}(t)-u_n(t)\|^2_{H^1(\Omega)^*} + \int_0^T \|u_{n_k}(t)-u(t)\|^2_{H^1(\Omega)^*}}\,.$$

Wegens $L^2(\Omega) \hookrightarrow H^1(\Omega)^*$ en Lemma 3.5 nadert de eerste term naar 0 voor $k \to \infty$. De tweede term nadert eveneens naar 0 als $k \to \infty$, omdat $u_{n_k} \to u$ in $C([0,T],H^1(\Omega)^*)$. We kunnen besluiten dat $\bar{u}_{n_k} \to u$ in $L^2((0,T),L^2(\Omega))$. □

Stelling 3.9. *Zij $u_0 \in L^2(\Omega)$. Dan bestaat er een variationele oplossing $u \in L^2((0,T),H^1(\Omega))$ van (3.8) met $\partial_t u \in L^2((0,T),H^1(\Omega)^*)$.*

Bewijs. De discrete variationele formulering (3.24) kan herschreven worden voor bijna elke $t \in [0,T]$ in termen van u_{n_k} en $\bar{u}_{n_k}$ als volgt

$$(\partial_t u_{n_k}(t),\varphi) + (g(|\nabla G_\sigma * \bar{u}^\tau_{n_k}(t)|)\nabla \bar{u}_{n_k}(t),\nabla\varphi) = 0, \quad \forall \varphi \in H^1(\Omega). \tag{3.33}$$

Hierbij is opnieuw $\bar{u}^\tau_{n_k}(t) := \bar{u}_{n_k}(t-\tau)$. We moeten de limiet nemen voor $k \to \infty$ in (3.33). Hiertoe integreren we (3.28) eerst in de tijd over $(0,\eta) \subset [0,T]$

$$\int_0^\eta (\partial_t u_{n_k}(t),\varphi)dt + \int_0^\eta (g(|\nabla G_\sigma * \bar{u}^\tau_{n_k}(t)|)\nabla \bar{u}_{n_k}(t),\nabla\varphi)dt = 0, \quad \forall \varphi \in H^1(\Omega). \tag{3.34}$$

We beweren dat $g(|\nabla G_\sigma * \bar{u}^\tau_{n_k}|) \to g(|\nabla G_\sigma * u|)$ b.o. in $\Omega \times [0,T]$. Inderdaad want door achtereenvolgens de Lipschitz-continuïteit van g, de omgekeerde driehoeksongelijkheid en (3.21) te gebruiken:

$$\begin{aligned}
\int_0^T |g(|\nabla G_\sigma * \bar{u}^\tau_{n_k}(t)|) - g(|\nabla G_\sigma * u(t)|)|^2 dt &\leq C \int_0^T \left| |\nabla G_\sigma * \bar{u}^\tau_{n_k}(t)| - |\nabla G_\sigma * u(t)| \right|^2 dt \\
&\leq C \int_0^T |\nabla G_\sigma * \bar{u}^\tau_{n_k}(t) - \nabla G_\sigma * u(t)|^2 dt \\
&\leq C \int_0^T \|\bar{u}^\tau_{n_k}(t) - u(t)\|^2 dt \\
&\leq C \int_0^T \left(\|\bar{u}^\tau_{n_k}(t) - \bar{u}_{n_k}(t)\|^2 + \|\bar{u}_{n_k}(t) - u(t)\|^2 \right) dt \\
&\to 0, \text{ als } k \to \infty.
\end{aligned}$$

De tweede term nadert naar nul wegens Lemma 3.6. De eerste term nadert naar nul wegens de a priori-afschatting (analoog zoals in Lemma 3.5).

Om de limiet te nemen voor $k \to \infty$ in (3.34) maken we gebruik van (3.32d) en (3.32c) uit Lemma 3.7, plus het voorgaande feit dat $g(|\nabla G_\sigma * \bar{u}^\tau_n|) \to g(|\nabla G_\sigma * u|)$ b.o. in $\Omega \times [0,T]$. Voor $k \to \infty$ geldt er

$$\int_0^\eta (\partial_t u_{n_k}(t), \varphi) dt \to \int_0^\eta \langle \partial_t u(t), \varphi \rangle dt$$

en

$$\int_0^\eta (g(|\nabla G_\sigma * \bar{u}^\tau_{n_k}(t)|) \nabla \bar{u}_{n_k}(t), \nabla \varphi) dt \to \int_0^\eta (g(|\nabla G_\sigma * u(t)|) \nabla u(t), \nabla \varphi) dt.$$

Hierdoor geldt voor alle $\varphi \in H^1(\Omega)$ dat

$$\int_0^\eta \langle \partial_t u(t), \varphi \rangle dt + \int_0^\eta (g(|\nabla G_\sigma * u(t)|) \nabla u, \nabla \varphi) dt = 0,$$

wat geldig is voor elke $\eta \in [0,T]$. Afleiden met betrekking tot de tijdsvariabele geeft ons (3.20) voor bijna elke $t \in [0,T]$. Hieruit concluderen we dat u een variationele oplossing is van (3.8). □

De existentie van een zwakke oplossing is aangetoond. De convergentie van de Rothe functies naar de zwakke oplossing (3.20) is aangetoond voor een deelrij $\{u_{n_k}\}$. Rekening houdend met de uniciteit van de oplossing is het duidelijk dat de volledige rij $\{u_n\}$ convergeert in $C([0,T], H^1(\Omega)^*) \cap L^2((0,T), L^2(\Omega))$ naar de oplossing u.

Zoals eerder vermeld, wordt in het originele artikel [8] bewezen dat
$u \in C([0,T], L^2(\Omega)) \cap L^2((0,T), H^1(\Omega))$. We bewezen tot nu toe dat $u \in L^2((0,T), H^1(\Omega))$ en $u_t \in L^2((0,T), H^1(\Omega)^*)$. Toepassen van Lemma 7.3 uit [34] impliceert dat $u \in C([0,T], L^2(\Omega))$. In [18] wordt de $C([0,T], L^2(\Omega)) \cap L^2((0,T), H^1(\Omega))$ convergentie van u_n naar u bewezen. Zij bekomen daartoe eerst dezelfde resultaten als wij hebben, zij het dan via een ander compactheid argument.

3.4 Implementatie van de regularisatie

Volledig analoog zoals in Hoofdstuk 2, kan de eindige volume methode gebruikt worden om een numeriek schema te bekomen voor (3.8). De convergentie hiervan werd bewezen in [28, 18]. We kunnen simpelweg de implementatie uit Appendix C.1 overnemen. Het enige dat we moeten aanpassen, zijn de waarden voor de coëfficiënten g^e, g^w, g^n en g^s. Merk op dat deze coëfficiënten ditmaal afhankelijk zijn van een extra parameter σ.

In het schema moeten we $G_\sigma * u$ en/of $|\nabla G_\sigma * u|$ kunnen berekenen. Daarom kiezen we in onze implementatie een Greense functie als G_σ, i.e.

$$G_\sigma(\boldsymbol{x}) = \frac{1}{4\pi\sigma} \exp\left(-\frac{|\boldsymbol{x}|^2}{4\sigma}\right).$$

Door deze keuze van G_σ, kunnen we $G_\sigma * u^{l-1}$ berekenen door de warmtevergelijking op te lossen op tijdstip σ met beginconditie u^{l-1}:

$$\begin{cases} \partial_t u(t,\mathbf{x}) - \Delta u(t,\mathbf{x}) = 0 & \text{in } (0,\sigma] \times \Omega, \\ \nabla u(t,\mathbf{x}) \cdot \boldsymbol{\nu} = 0 & \text{in } (0,\sigma] \times \partial\Omega, \\ u(0,\mathbf{x}) = u^{l-1}(\mathbf{x}) & \text{in } \Omega. \end{cases} \tag{3.35}$$

Nadat we $G_\sigma * u^{l-1}$ berekend hebben, moeten we nog $|\nabla G_\sigma * u^{l-1}|$ bepalen. Dit kan opnieuw verwezenlijkt worden via centrale differenties zoals uitgelegd in Sectie 2.1.

De lineaire warmtevergelijking kan numeriek worden opgelost zoals het schema (2.28). Omdat σ voor praktische keuzes klein is, doen we dit in één tijdstap ($\tau = \sigma$). Hierdoor convergeert het schema na één iteratie. Alle g^e, g^w, g^n en g^s worden gelijkgesteld aan één (en gelijk aan nul indien er geen buur is in deze richting). Formeel: we zoeken naar een functie u^c die een oplossing is van de warmtevergelijking, gediscretiseerd in de tijd door de achterwaartse Euler met tijdstap σ, i. e.

$$\frac{u^c - u^{l-1}}{\sigma} = \Delta u^c.$$

Numeriek betekent dit dat we een vector u^c zoeken die de oplossing is van het stelsel $Au^c = u^{l-1}$, waarbij A de volgende structuur heeft,

$$\begin{bmatrix} 1+s_1\sigma & -\sigma & & -\sigma & & & \\ -\sigma & \ddots & \ddots & & \ddots & & \\ & \ddots & & \ddots & & \ddots & \\ -\sigma & & \ddots & & \ddots & & -\sigma \\ & \ddots & & \ddots & & \ddots & \\ & & \ddots & & \ddots & & -\sigma \\ & & & -\sigma & & -\sigma & 1+s_{mn}\sigma \end{bmatrix}.$$

Hierbij is de waarde van s_k met $k = 1, \ldots, mn$, de som van het aantal buren van de pixel met lineaire index k.
Nadat u^c berekend is, kan deze afbeelding gebruikt worden om de coëfficiënten g^n, g^s, g^e en g^s te bepalen, analoog zoals vermeld in Hoofdstuk 2.

De implementatie van de geregulariseerde Perona-Malik vergelijking is te vinden in Appendix C.2. De implementatie van de lineaire warmtevergelijking is te vinden in Appendix C.3

4 Vergelijkende studie

"A very large part of space-time must be investigated if reliable results are to be obtained."
Alan Turing

Op het einde van Hoofdstuk 2 zagen we reeds hoe de Perona-Malik vergelijking kan gebruikt worden om beschadigde afbeeldingen te restaureren. Deze resultaten oogden vrij indrukwekkend: de randen blijven stabiel over een lange tijd terwijl de andere gebieden gladder gemaakt worden. In Hoofdstuk 3 stelden we vast dat de originele Perona-Malik vergelijking kampt met een wiskundig probleem. Het is met name een slecht gesteld probleem (in zake existentie en uniciteit). Omwille hiervan hebben we een geregulariseerd model (3.8) bestudeerd, zij het louter vanuit theoretisch standpunt.

In dit hoofdstuk vergelijken we de performantie van het originele Perona-Malik model (3.1) met het geregulariseerde model (3.8). We gaan na of de theoretische ingreep een praktische impact geeft.

4.1 Methode

Parameters

Voor beide modellen hebben we een numeriek schema afgeleid op basis van de eindige volume methode. Om een eerlijke vergelijking te maken, gebruiken we telkens dezelfde diffusiefunctie g, met name

$$g(x) = \frac{1}{1 + \frac{x^2}{\kappa^2}}. \tag{4.1}$$

In beide implementaties wordt dezelfde waarde van κ gebruikt:

$$\kappa = 1.4826\text{MAD}(|\nabla u_0|).$$

We nemen voor beide modellen telkens dezelfde tijdstap $\tau = 0.2$ en eindtijd $T = 4$. Dit levert een totaal van 20 iteraties.

Voor het geregulariseerde model is er een extra parameter σ. We beschouwen drie waarden van σ, met name $\sigma = 0.1$, $\sigma = 0.5$ en $\sigma = 1$. Samen met de standaard Perona-Malik vergelijking geeft dit ons in totaal vier modellen om te vergelijken.

Testafbeelding

Om de modellen te vergelijken, passen we ze telkens toe op eenzelfde afbeelding met ruis. De testafbeelding die we zullen gebruiken is te zien in Figuur 4.1.[1] Deze afbeelding is een luchtfoto van het gebouw S8 op campus de Sterre, waar onder andere de onderzoeksgroep *Numerical Analysis and Mathematical Modelling* NaM^2 zich bevindt. We onderwerpen deze foto aan multiplicatieve ruis met variantie $v = 0.1$. Dit betekent dat de ruis uniform verdeeld is met gemiddelde 0 en variantie v.

Figuur 4.1: Testafbeelding.

Output objectief beoordelen

De laatste stap bestaat erin om de kwaliteit van de output van deze herstelde afbeeldingen te beoordelen. De kwaliteit van een afbeelding wordt in laatste instantie beoordeeld door het menselijk oog. In vele toepassingen kan het nuttig zijn om dit op een objectievere manier te beoordelen. Om de vier modellen te vergelijken hebben we een objectieve maat nodig die

[1] Deze afbeelding is beschikbaar op `http://www.ugent.be/we/img/we13/s8-luchtfoto/view` als een 735×426 jpg afbeelding in kleur.

kwaliteit meet van een afbeelding ten opzichte van een referentie-afbeelding. De maat die we daartoe gebruiken is de PSNR.

De PSNR is de Engelstalige afkorting voor *Peak signal-to-noise-ratio*. De PSNR is gedefinieerd als

$$PSNR = 10 \log_{10} \left(\frac{M^2}{MSE} \right), \tag{4.2}$$

waarbij MSE de *means-squared error* is tussen de twee afbeeldingen [25]. De factor M staat voor de grootst mogelijke intensiteit waarde van de afbeelding. Wanneer we 8 bits per pixel gebruiken is dit 255.

De PSNR wordt uitgedrukt in decibel. Hogere PSNR waarden geven in het algemeen een indicatie dat de gerestaureerde afbeelding van een hogere kwaliteit is. In MATLAB berekent

```
>> psnr(A,ref)
```

de PSNR voor de afbeelding A, met *ref* als referentie-afbeelding. Deze moeten noodzakelijkerwijze van dezelfde grootte en klasse zijn. Voor ons is de referentie-afbeelding de originele afbeelding uit Figuur 4.1.

4.2 Resultaten

In Figuur 4.2 zijn de PSNR waarden uitgezet ten opzichte van het aantal iteraties. Op deze figuur is te zien dat het originele model de meeste iteraties nodig heeft om zijn maximale PSNR waarde te bereiken. Wat ook opvalt, is dat hoe kleiner de waarde van σ, hoe dichter de corresponderende curve bij die van het originele model ligt. Dit is juist wat we verwachten, aangezien in de limiet voor $\sigma \to 0$, we het originele model terugvinden, zie begin Hoofdstuk 3.

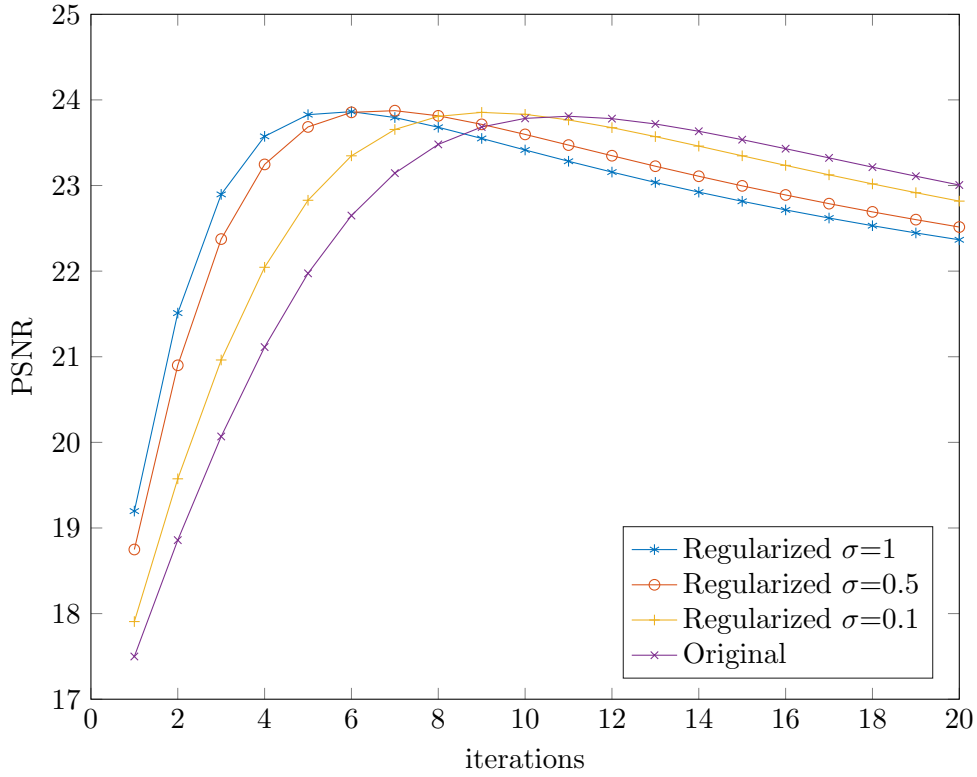
De prestatiematen voor de vier modellen zijn samengevat in Tabel 4.1. Voor de drie geregulariseerde modellen vinden we een (lichtjes) hogere PSNR dan bij het originele model. In Tabel 4.1 vermelden we ook de CPU-tijd[1] die nodig is om de maximale PSNR te bereiken. Op de onderste rij van de tabel hebben we de CPU-tijd van het volledige proces weergegeven. Logischerwijze is de benodigde CPU-tijd voor het originele model kleiner. Dit komt omdat in elke iteratie van het geregulariseerde model als extra de lineaire warmtevergelijking moet worden opgelost.

We kunnen concluderen dat de geleverde kwaliteit van de geregulariseerde modellen minstens zo goed is als die van het originele model. Om de uniciteit en existentie te garanderen, moeten we het geregulariseerd model gebruiken voor een σ die niet te groot is. De prijs die we daarvoor betalen, is de grotere rekentijd die het programma nodig heeft.

[1] De tijd die de computer nodig heeft voor het uitvoeren van de programmacode.

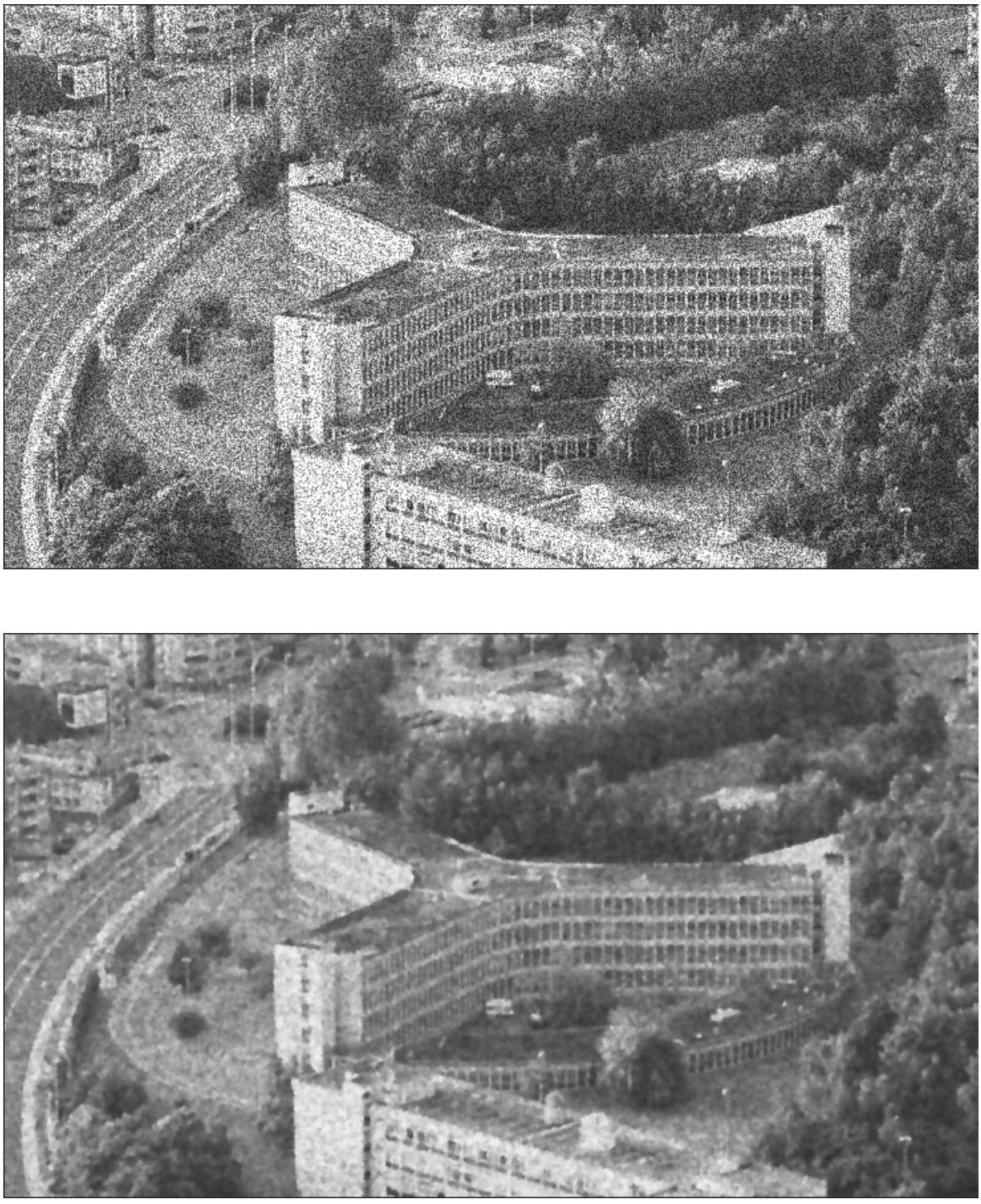
	Origineel	Geregulariseerd		
		$\sigma = 0.1$	$\sigma = 0.5$	$\sigma = 1$
maximale PSNR	23.81	23.85	23.87	23.86
benodigde iteraties	11	9	7	6
bijhorende CPU-tijd	37.05	51.12	44.71	42.14
CPU-tijd voor 20 iteraties	69.39	120.39	125.50	131.79

Tabel 4.1: Vergelijking tussen de verschillende modellen.

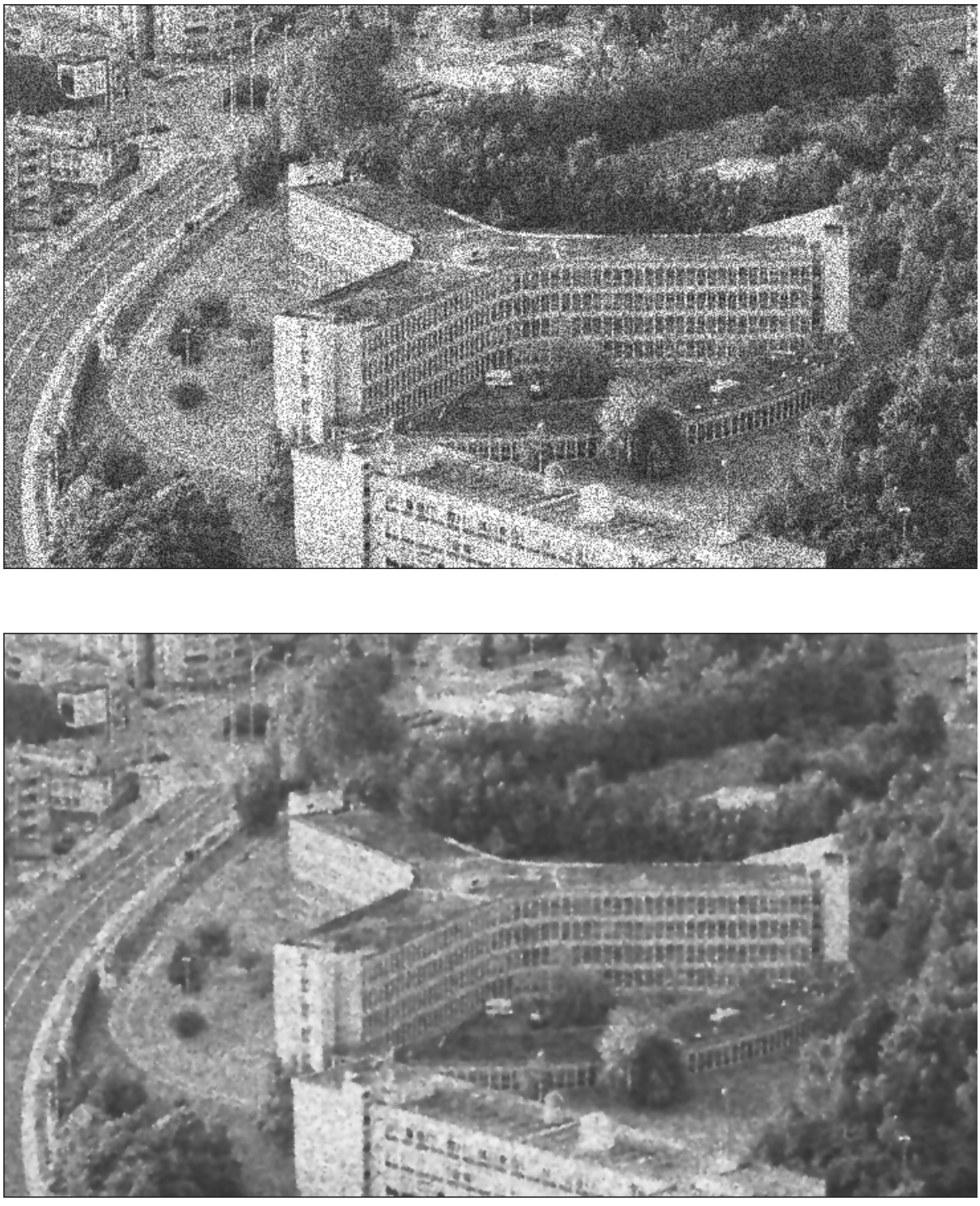


Figuur 4.2: PSNR waarden in functie van het aantal iteraties.

Ter illustratie tonen we in Figuur 4.3 het resultaat onder het geregulariseerde model met $\sigma = 0.5$, op het tijdstip waar de maximale PSNR bereikt wordt. In Figuur 4.4 tonen we het resultaat onder het originele model, opnieuw op het tijdstip waar de maximale PSNR bereikt wordt. We presenteren de afbeeldingen hier vrij groot, ook al zijn ze maar op een 4/5de schaal van hun originele grootte weergegeven. Indien we de afbeeldingen nog meer verkleinen zijn details minder zichtbaar. Het zijn net op deze details (ruis) die we willen letten.



Figuur 4.3: Boven: Testafbeelding beschadigd door multiplicatieve ruis met parameter $v = 0.1$. Onder: Herstelde testafbeelding door het geregulariseerde Perona-Malik model na 7 iteraties. Parameters: $\tau = 0.2$, $T = 4$, $g(x) = (1 + (x/\kappa)^2)^{-1}$, $\sigma = 0.5$.



Figuur 4.4: Boven: Testafbeelding beschadigd door multiplicatieve ruis met parameter $v = 0.1$. Onder: Herstelde testafbeelding door het originele Perona-Malik model na 11 iteraties. Parameters: $\tau = 0.2$, $T = 4$, $g(x) = (1 + (x/\kappa)^2)^{-1}$.

Algemeen besluit

In Hoofdstuk 2 introduceerden we de beroemde Perona-Malik vergelijking. Voor deze niet-lineaire diffusievergelijking hebben we een numeriek schema bepaald aan de hand van de eindige volume methode. Omdat we deze vergelijking willen toepassen op digitale afbeeldingen, hebben we de pixels als eindige volumes gebruikt. Dit schema hebben we - zo efficiënt en elegant mogelijk - vertaald naar MATLAB. De code hiervan is te vinden in Appendix C. In Hoofdstuk 3 hebben we een aangepaste versie van de Perona-Malik vergelijking in een theoretisch kader bestudeerd. De existentie van de oplossing hebben we bewezen aan de hand van de Rothemethode. Dit is een methode die gebruikt en onderzocht wordt door de onderzoeksgroep NaM^2 aan de UGent, in samenwerking met de vakgroep Numerieke Analyse, Comenius Universiteit Bratislava en met de vakgroep Modellering van Processen aan de Technische Universiteit Liberec. Tot slot hebben we in Hoofdstuk 4 het originele model vergeleken met het aangepaste model, voor verschillende waarden van de regularisatieparameter σ. We hebben dit gedaan op basis van de maximale PSNR waarde en de CPU-tijd. We konden concluderen dat de regularisatie hetzelfde resultaat oplevert als het originele model. Het nadeel was dat de numerieke implementatie een grotere CPU-tijd vraagt.

In deze masterproef hebben we geen nieuwe diffusievergelijking gemodelleerd en/of bestudeerd. We hebben de bekendste partiële differentiaalvergelijking in de digitale beeldverwerking onderzocht, zowel praktisch als theoretisch. Een relatief nieuw onderzoeksgebied zou zijn om fractionele afgeleiden te gebruiken. Een uitgebreid overzicht van methoden gebaseerd op fractionele afgeleiden in digitale beeldverwerking is te vinden in [53]. Een vervolg op deze masterproef zou eruit kunnen bestaan om deze modellen (theoretisch) te bestuderen, te implementeren en ze tenslotte te vergelijken met de klassieke modellen.

Appendices

A English Summary

In many applications, digital images may contain noise. Relevant examples are medical images. Correct denoising of these type of images is extremely important, because it can make the difference between an early detection of a medical problem or a wrong diagnosis.

A classical operation to smooth images is to convolve them with a Gaussian kernel G_σ. Because G_σ is a fundamental solution of the linear heat equation, one can replace this classical operation by solving numerically the linear heat equation. As this method don't preserve edges of an image, in this project we study the so-called Perona-Malik equation,

$$\partial_t u(t, \boldsymbol{x}) - \nabla \cdot (g(|\nabla u(t, \boldsymbol{x})|)\nabla u(t, \boldsymbol{x}) = 0 \quad \text{in } (0, T] \times \Omega. \tag{A.1}$$

Equation (A.1) comes along with a Neumann boundary condition. The initial state is given by the original noisy image u_0. We operate with gray-scale images, so $\Omega \subset \mathbb{R}^2$. The solution $u(t, \boldsymbol{x})$ represents a family of scaled, filtered, smoother versions of u_0.

Because of the intended properties in the design of the Perona-Malik equation, the choice of the function g in (A.1) has the following restrictions,

$$\begin{gathered} g : \mathbb{R}_0^+ \to \mathbb{R}^+ \text{ is a non-decreasing functie}, \quad g \in C^\infty(\mathbb{R}), \\ g(0) = 1, \quad \text{and } g(x) \to 0 \text{ for } x \to \infty. \end{gathered} \tag{A.2}$$

We use the following two choices of g,

$$g_1(x) = e^{-\frac{1}{2}(\frac{x}{\kappa})^2} \tag{A.3}$$

and

$$g_2(x) = \frac{1}{1 + (x/\kappa)^2}. \tag{A.4}$$

The parameter $\kappa > 0$ is called the scale parameter and is crucial in the model. One can easily check that the flux function $f(x) = g(|x|)x$ for the practical choices of g (A.3) and (A.4) attains it maximum at $x = \kappa$. The one-dimensional case of (A.1),

$$u_t - f'(u_x)u_{xx} = 0,$$

reveals that when $|u_x| < \kappa$ (this implies f' is positive) forward diffusion thakes place. This process is the same as the lineair heat equation and smooths the image locally. When $|u_x| > \kappa$ backwards diffusion takes place. This process does the opposite, it makes the image locally more

sharply. At first, an intuitive choice for κ would be the 90% quantile of $|\nabla u|$. However we choose for κ,

$$\kappa = 1.4826 * \text{median}(|\nabla u_0 - \text{median}(|\nabla u_0|)|),$$

based on statistical reasons, see [3].

Next, to solve the Perona-Malik equation (A.1), we have to find a numerical scheme. Here after, we give a brief summary of the the method we used.
First we approximate the time derivative with a backward Euler. We also evaluate ∇u inside the function g at the previous discrete scale step. Therefore, we get a linear parabolic equation at every discrete scale step l,

$$\frac{u^l - u^{l-1}}{\tau} - \nabla \cdot (g(|\nabla u^{l-1}|)\nabla u^l) = 0. \tag{A.5}$$

Secondly, we discretize these lineair equations (A.5) in space with the finite volume method[12, 29, 27]. This methods suits well with the pixel structure of an image. The finite volumes correspond to the pixels of the image. We integrate (A.5) over such a volume and derive a quadrature formule. We define $\mathcal{N}(p)$ as the set of neighbours for a pixel p. Let u_p^l be the approximated value of the solution inside the pixel p, at the discrete scale step l. Finally, we get the following.
Linear fully discrete finite volume scheme for solving equation (A.1). *Let $N_t \in \mathbb{N}$ and $\tau = T/N_t$ be fixed numbers, $t_l = l\tau, l = 0, \ldots N_t$. For every $l = 1, \ldots, N_t$, we look for u_p^l, with p a pixel of the image, $i = 1, \ldots, n$, $j = 1, \ldots, m$, satisfying*

$$u_p^l + \tau \sum_{q \in \mathcal{N}(p)} g_q(p)(u_p^l - u_q^l) = u_p^{l-1}, \tag{A.6}$$

starting with u^0 and where $g_q(p)$ is $g(|\nabla u|)$ at the center of the common edge of pixel p en his neighbour q.
Equation A.6 gives a lineair system $Au^l = u^{l-1}$, where the matrix A is symmetric and strictly diagonally dominant, so there exists a unique solution at every discrete scale step. Moreover one can prove the L_∞-stability of the scheme,

$$\min_p u_p^0 \leq \min_p u_p^l \leq \max_p u_p^l \leq \max_p u_0^l, \quad 1 \leq l \leq N_t. \tag{A.7}$$

We have written our own implementation of this scheme in MATLAB, which can be found in Appendix C. In this implementation we allow an extra parameter α to adapt the functions g,

$$g_1(x) = e^{-\frac{1}{2}(\frac{x}{\kappa})^\alpha} \quad \text{or} \quad g_2(x) = \frac{1}{1 + (x/\kappa)^\alpha}.$$

We allow α to be every real number. We've compared the values for $\alpha \in \{0, 1/2, 1, 3/2, 2\}$. We came to the conclusion that $\alpha = 2$ gives te best results. The quality decreases as $\alpha \to 0$. A reason for this could be the fixed value of κ in these experiments. As α decreases, the flux function f reaches no longer his maximum at $x = \kappa$, but for greater values of x. This means

that it's more likely to have forward diffusion, and so the images for smaller values of α looks like an application of the lineair heat equation.

As pointed out, the Perona-Malik equation can behave locally like a backward diffusion. This implies, it is in general, an ill-posed problem (in the sense of Hadamard). To deal with this mathematical disadvantage, the authors of [8] proposed to introduce a (spatial) convolution with a Gaussian kernel G_σ inside the argument of the diffusion coefficient,

$$\partial_t u(t,\mathbf{x}) - \nabla \cdot (g(|\nabla G_\sigma * u(t,\mathbf{x})|)\nabla u(t,\mathbf{x})) = 0 \quad \text{in } (0,T]\times\Omega. \tag{A.8}$$

With this regularisation, it has been proven that (A.8) is a well-posed problem. We've used Rothe's method [17, 44], a priori estimates and several compactness theorems to proove te existence of a solution. In summary, based on [18] we have proven the following theorem.

Theorem A.1. *Let $u_0 \in L^2(\Omega)$. Then there exists a unique variational solution of (A.8).*

Numerically, we are able to compute $G_\sigma * u$ by solving the lineair heat equation. This is the only extra computation needed, compared with our previous numerical implementation. This MATLAB function can also be found in Appendix C.

In the last chapter, we compare the regularized equation (A.8) with the original, for the values of $\sigma \in \{1/10, 1/2, 1\}$. We've done this by adding speckle noise to a test image. Then we apply the four models to this image for the same parameters. We compare the quality of the results based on the peak-signal-to-noise-ratio (PSNR) and the CPU-time needed. We concluded that all the models achieve the same quality, but the original model does this in significant smaller CPU-time. The price we pay for regularising the original model is the greather CPU-time needed, which lies in the extra compution $G_\sigma * u$.

Further work that could be done, is to investigate the use of fractional calculus in image processing [53]. One can study thes models theoretically (well-posedness), implement them in MATLAB and finally, compare them with (A.1) and (A.8).

B Hulpmiddelen

We veronderstellen dat de lezer reeds vertrouwd is met reële & complexe analyse, lineair genormeerde ruimten, Banachruimten, Hilbert ruimten, de Lebesgue integraal en partiële differentiaalvergelijkingen. De Sobolev ruimten die in deze masterproef gebruikt worden, zijn verderop in deze Appendix nog eens expliciet gedefinieerd. Voor een uitgebreide uitleg van Sobolev ruimten en begrippen uit de functionaalanalyse verwijzen we de lezer naar [44][Hoofdstuk 2].

Handige (on)gelijkheden

Lemma B.1. *(Handige ongelijkheden). Voor alle $a, b \in \mathbb{R}$ geldt de volgende ongelijkheid*

$$(a+b)^2 \leq 2(a^2+b^2).$$

Bovendien geldt voor alle $a, b \in \mathbb{R}$ en $p \geq 1$ dat

$$|a+b|^p \leq 2^{p-1}(|a|^p + |b|^p).$$

Lemma B.2. *(Young ongelijkheid). Stel dat $a, b \in [0, \infty), p \in (1, \infty)$ en $\frac{1}{p} + \frac{1}{q} = 1$. Dan*

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q}.$$

Lemma B.3. *(ε-Young ongelijkheid). Stel dat $a, b, \varepsilon \in [0, \infty), p \in (1, \infty)$ en $\frac{1}{p} + \frac{1}{q} = 1$. Dan*

$$ab \leq \varepsilon a^p + (\varepsilon p)^{\frac{-q}{p}} q^{-1} b^q = \varepsilon a^p + C_\varepsilon b^q.$$

Lemma B.4. *(Cauchy-Schwarz ongelijkheid voor integralen). Zij $f(\boldsymbol{x})$ en $g(\boldsymbol{x})$ willekeurige reëel integreerbare functies in Ω. Dan*

$$\left(\int_\Omega f(\boldsymbol{x}) g(\boldsymbol{x}) d\boldsymbol{x} \right)^2 \leq \left(\int_\Omega f^2(\boldsymbol{x}) d\boldsymbol{x} \right) \left(\int_\Omega g^2(\boldsymbol{x}) d\boldsymbol{x} \right).$$

Lemma B.5. *(Grönwall-continu). Zij $r(t), h(t), y(t)$ continue reële functies gedefinieerd op $[a, b]$ die voldoen aan $r(t), h(t) \geq 0$.*

(i) Als

$$y(t) \leq h(t) + \int_a^t r(s) y(s) ds \quad \text{voor } a \leq t \leq b,$$

Dan is

$$y(t) \leq h(t) + \int_a^t h(s)r(s)\exp\left(\int_s^t r(\tau)d\tau\right)ds \quad \textit{voor } a \leq t \leq b,$$

geldig voor alle $t \in [a, b]$.

(ii) Veronderstel (i). Als $r(s) = C$ *en de functie* h *is niet-dalend, dan*

$$y(t) \leq h(t)e^{C(t-a)} \quad \textit{voor } a \leq t \leq b.$$

Lemma B.6. *(Grönwall-discreet). Zij* $\{A_i\}$, $\{a_i\}$ *rijen van niet-negatieve getallen en zij* $q \geq 0$. *Veronderstel*

$$a_i \leq A_i + \sum_{j=1}^{i-1} a_j q,$$

voor $i \in \mathbb{N}$. *Dan*

$$a_i \leq A_i + e^{qi}\sum_{j=1}^{i-1} A_j q, \quad i \in \mathbb{N}.$$

Lemma B.7. *(Abel's sommatie regel). Zij* $\{a_i\}_{i=1}^n$ *een rij van reële getallen met* $n \geq 1$. *Dan*

$$2\sum_{i=1}^{n}(a_i - a_{i-1})a_i = a_n^2 - a_0^2 + \sum_{i=1}^{n}(a_i - a_{i-1})^2.$$

Domeinen

Definitie B.8. *Een begrensd domein* $\Omega \subset \mathbb{R}^d$ *heeft een stuksgewijs gladde rand* $\partial\Omega$ *als* $\partial\Omega$ *bestaat uit een eindig aantal gladde delen en de uitwendige normaalvector* $\boldsymbol{\nu}$ *bestaat in bijna alle punten van* $\partial\Omega$.

Definitie B.9. *(Lipschitz rand). Zij* $\Omega \subset \mathbb{R}^d$ *een begrensd domein. We zeggen dat* Ω *een Lipschitz continue rand* $\partial\Omega$ *heeft, indien er constanten* $\alpha > 0$, $\beta > 0$ *bestaan en er* $m \in \mathbb{N}$ *continue functies* $a_r(x_1^{(r)}, \ldots, x_{d-1}^{(r)})$ *met* $r = 1, \ldots, m$ *bestaan, die gedefinieerd zijn in* $(d-1)$ *-dimensionale open kubussen*

$$K^{(r)} = \{(x_1^{(r)}, \ldots, x_{d-1}^{(r)}) : |x_i^{(r)}| < \alpha,\ i = 1, \ldots, d-1\}$$

zodanig dat

1. *elk punt van* $\partial\Omega$ *kan in tenminste één van de coördinatensystemen uitgedrukt worden als*

$$\boldsymbol{x} = (x_1^{(r)}, \ldots, x_{d-1}^{(r)}, a_r(x_1^{(r)}, \ldots, x_{d-1}^{(r)})),$$

2. *de punten* $\boldsymbol{x} = (x_1^{(r)}, \ldots, x_d^{(r)})$, *waarvoor* $|x_i^{(r)}| < \alpha$ *voor* $i = 1, \ldots, d-1$ *en*

$$a_r(x_1^{(r)}, \ldots, x_{d-1}^{(r)}) < x_d^{(r)} < a_r(x_1^{(r)}, \ldots, x_{d-1}^{(r)}) + \beta,$$

respectievelijk,

$$a_r(x_1^{(r)}, \ldots, x_{d-1}^{(r)}) - \beta < x_d^{(r)} < a_r(x_1^{(r)}, \ldots, x_{d-1}^{(r)})$$

liggen in Ω, *respectievelijk buiten* $\overline{\Omega}$.

3. *alle functies* $a_r(x_1^{(r)},\ldots,x_{d-1}^{(r)})$ *voor* $r=1,\ldots,m$ *zijn Lipschitz continu in de kubus* $K^{(r)}$, *i.e.*

$$|a_r(x_1^{(r)},\ldots,x_{d-1}^{(r)})-a_r(y_1^{(r)},\ldots,y_{d-1}^{(r)})|^2\le C\sum_{i=1}^{d-1}|x_i^{(r)}-y_i^{(r)}|^2.$$

Ruimten van continue functies

Definitie B.10. *Zij* $\Omega\subset\mathbb{R}^d$ *een begrensd domein met een stuksgewijs continue rand. We noteren met* $C^\infty(\overline{\Omega})$ *de verzameling van alle functies wiens afgeleiden continu zijn in* $\overline{\Omega}$. *De verzameling van alle punten van* Ω *waarvoor* $u(\boldsymbol{x})\neq 0$ *wordt de drager genoemd van* u. *Het symbool* $C_0^\infty(\overline{\Omega})$ *staat voor de verzameling van alle* $u\in C^\infty(\overline{\Omega})$ *met compacte drager.*
Analoog definiëren we de ruimten $C^k(\overline{\Omega})$ *en* $C_0^k(\overline{\Omega})$ *voor* $k\in\mathbb{N}$. *Functies uit deze ruimten hebben continue afgeleiden tot en met orde* k.

Lebesgue ruimten

Definitie B.11. *(Lebesque-ruimte). Neem* $p\ge 1$. *De Lebesgue-ruimte* $L^p(\Omega)$ *is de verzameling van alle meetbare functies* f *in het domein* Ω, *waarvoor*

$$\|f\|_{L^p(\Omega)}^p=\int_\Omega|f(\boldsymbol{x})|^p d\boldsymbol{x}<\infty.$$

De ruimte $L_\infty(\Omega)$ *is een niet-separabele Banachruimte met norm*

$$\|f\|_{L^\infty(\Omega)}=\inf\{a\in\mathbb{R}:\ \mu(\{\boldsymbol{x}\in\Omega:\ f(\boldsymbol{x})>a\})=0\}=\lim_{p\to\infty}\|f\|_{L^p(\Omega)}.$$

Deze norm wordt het essentiële supremum genoemd van f. *De notatie* μ *staat voor de Lebesgue-maat.*

De ruimte $L^p(\Omega)$ is een reflexieve Banachruimte voor $1<p<\infty$. De duale ruimte is $L^q(\Omega)$ met $\frac{1}{p}+\frac{1}{q}=1$.
Als $p=2$ is de ruimte $L^p(\Omega)=L^2(\Omega)$ een Hilbert ruimte met inproduct

$$(f,g)_\Omega=\int_\Omega f(\boldsymbol{x})g(\boldsymbol{x})\,d\boldsymbol{x}.$$

Sobolev ruimten

Definitie B.12. *(multi-index). Zij* $\Omega\subset\mathbb{R}^d$. *De multi-index* $\boldsymbol{\alpha}$ *is een* d*-dimensionale vector met*

$$\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_d),\quad \alpha_i\ge 0,\ \alpha_i\in\mathbb{Z}.$$

De lengte van $\boldsymbol{\alpha}$ wordt gegeven door

$$|\boldsymbol{\alpha}| = \sum_{i=1}^{d} \alpha_i.$$

Verder introduceren we de volgende notatie

$$D^{\boldsymbol{\alpha}} u = \frac{\partial^{|\boldsymbol{\alpha}|} u}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}.$$

Definitie B.13. *(Ruimten $W^{k,p}(\Omega)$ met $k \in \mathbb{N}$). Neem $i \in \mathbb{N}$, $p \geq 1$. De semi-norm van de i-de afgeleide van $u \in C^\infty(\overline{\Omega})$ is*

$$|u|_{i,p,\Omega} = \left(\sum_{|\boldsymbol{\alpha}|=i} \int_\Omega |D^{\boldsymbol{\alpha}} u(\boldsymbol{x})|^p \, d\boldsymbol{x} \right)^{\frac{1}{p}}. \tag{B.1}$$

De norm van $u \in C^k(\overline{\Omega})$ met $k \in \mathbb{N}$ is gedefinieerd in termen van de semi-normen als volgt

$$\|u\|_{k,p,\Omega} = \left(\sum_{i=0}^{k} |u|_{i,p,\Omega}^p \right)^{\frac{1}{p}}. \tag{B.2}$$

De ruimte $W^{k,p}(\Omega)$ is de sluiting van $C^\infty(\overline{\Omega})$ met betrekking tot de norm (B.2).

De ruimten $W^{k,p}(\Omega)$ zijn Banachruimten. Voor $p \in (1, \infty)$ is $W^{k,p}(\Omega)$ reflexief. Voor $p \in [1, \infty)$ is $W^{k,p}(\Omega)$ separabel.

Neem $p = 2$. De norm (B.2) induceert het volgende inproduct op $W^{k,2}(\Omega)$

$$(u, v)_{k,\Omega} = \sum_{|\boldsymbol{\alpha}| \leq k} \int_\Omega D^{\boldsymbol{\alpha}} u D^{\boldsymbol{\alpha}} v \, d\boldsymbol{x}.$$

Er geldt dat

$$(u, u)_{k,\Omega} = \|u\|_{k,2,\Omega}^2.$$

De ruimte $W^{k,2}(\Omega)$ is een Hilbert ruimte en wordt genoteerd als $H^k(\Omega)$.

Definitie B.14. *Zij u, v lokaal integreerbare functies in Ω. De functie $v = u^{(\boldsymbol{\alpha})}$ wordt de veralgemeende afgeleide van orde $\boldsymbol{\alpha}$ genoemd van u als*

$$\int_\Omega u D^{\boldsymbol{\alpha}} \varphi d\boldsymbol{x} = (-1)^{|\boldsymbol{\alpha}|} \int_\Omega v \varphi d\boldsymbol{x}$$

geldig is voor alle $\varphi \in C_0^\infty(\overline{\Omega})$.

Nu introduceren we een vectorruimte $H^{k,p}(\Omega)$ voor $k \in \mathbb{N}$ en $p \geq 1$.

Definitie B.15. *(Ruimten $H^{k,p}(\Omega)$ met $k \in \mathbb{N}$). Veronderstel dat Ω een niet-lege open deelverzameling is van $\mathbb{R}^d$. De functie $u \in L^p(\Omega)$ is een element van $H^{k,p}(\Omega)$ als alle veralgemeende afgeleiden tot en met orde k bestaan van u en deze allemaal behoren tot $L^p(\Omega)$. De norm in deze ruimte is*

$$\|u\|^p_{H^{k,p}(\Omega)} = \sum_{|\boldsymbol{\alpha}|\leq k} \left\|u^{(\boldsymbol{\alpha})}\right\|_{L^p(\Omega)}.$$

Er bestaat een relatie tussen de ruimten $H^{k,p}(\Omega)$ en $W^{k,p}(\Omega)$. Als het domein Ω een Lipschitz continue rand heeft dan geldt $H^{k,p}(\Omega) = W^{k,p}(\Omega)$ voor $k \in \mathbb{N}$ en $p \in [1, \infty)$.

Er zijn speciale Sobolevruimten voor tijdsafhankelijke problemen. Deze zijn essentieel in de constructie van zwakke oplossingen van parabolische en hyperbolische partiële differentiaalvergelijkingen.

Definitie B.16. *Beschouw een Banachruimte X met de norm $\|\cdot\|_X$.*

- *De ruimte $L^p((0,T),X)$ bestaat uit abstracte functies $u : (0,T) \mapsto X$ zodat*

$$\|u\|_{L^p((0,T),X)} = \left(\int_0^T \|u(t)\|^p_X dt\right)^{1/p} < \infty.$$

- *De ruimte $C([0,T],X)$ bestaat uit continue functies $u : [0,T] \mapsto X$ die voldoen aan*

$$\|u\|_{C([0,T],X)} = \max_{[0,T]} \|u(t)\|_X < \infty.$$

- *De ruimte $L^\infty((0,T),X)$ bestaat uit alle meetbare functies $u : (0,T) \mapsto X$ die essentieel begrensd zijn, i.e.*

$$\|u\|_{L^\infty((0,T),X)} = \inf\{B : \ \|u(t)\|_X \leq B \text{ voor bijna elke } t \in (0,T)\} < \infty.$$

Duale ruimten

Zij B een Banachruimte. We noteren met B' de duale ruimte van B, i.e. de verzameling van alle lineaire functionalen in B. Het symbool B^* staat voor de verzameling van alle lineair begrensde functionalen in B. We herinneren eraan dat een lineaire functionaal L begrensd is als er een positieve constante C bestaat zodat

$$|L(u)| \leq C\|u\|_B \quad \forall u \in B.$$

De norm van een continue lineaire functionaal L is gedefinieerd als

$$\|L\|_{B^*} := \sup_{0 \neq v \in B} \frac{L(v)}{\|v\|_B}. \tag{B.3}$$

Inbeddingen

Definitie B.17. *Zij V en W twee Banach ruimten met $V \subset W$. We zeggen dat de ruimte V continu is ingebed in W en noteren $V \hookrightarrow W$, als*

$$\|v\|_W \leq C\|v\|_V, \quad \forall v \in V, \tag{B.4}$$

voor een positieve constante C.

We zeggen dat de ruimte V compact is ingebed in W en noteren $V \hookrightarrow\hookrightarrow W$, als (B.4) geldt en elke begrensde rij in V een convergente deelrij heeft in W.

Lemma B.18. *Beschouw twee genormeerde ruimten X en Y. Dan geldt er dat*

- *$X \hookrightarrow Y$ impliceert $Y^* \hookrightarrow X^*$;*
- *$X \hookrightarrow\hookrightarrow Y$ impliceert $Y^* \hookrightarrow\hookrightarrow X^*$.*

Definitie B.19. *Een evolutie triplet*

$$V \hookrightarrow H \cong H \hookrightarrow V^*$$

voldoet aan volgende eigenschappen

- *V is een reële, separabele en een reflexieve Banach ruimte,*
- *H is een reële, separabele Hilbert ruimte met inproduct $(\cdot,\cdot)_H$,*
- *De inbedding $V \subseteq H$ is continu en V is dicht in H.*

Stelling B.20. *Zij $V \hookrightarrow H \cong H \hookrightarrow V^*$ een evolutie triplet. Dan kan het dualiteitspaar $\langle\cdot,\cdot\rangle_{V^*\times V}$ als een continue extensie gezien worden van het inproduct op H, i.e. Voor elke $h \in H$ en $v \in V$ geldt het volgende*

$$\langle h, v\rangle_{V^*\times V} = (h, v)_H.$$

Lemma B.21. *(Partiële integratie). Zij $V \hookrightarrow H \cong H \hookrightarrow V^*$ een evolutie triplet. Voor alle $u \in L^2((0,T),V)$ met $\frac{du}{dt} \in L^2((0,T),V^*)$ en voor elke $0 \leq t_1 \leq t_2 \leq T$ geldt de volgende formule:*

$$\int_{t_1}^{t_2} \left\langle \frac{du(t)}{dt}, u(t) \right\rangle_{V^*\times V} dt = \frac{1}{2}\|u(t_2)\|_H^2 - \frac{1}{2}\|u(t_1)\|_H^2.$$

Stelling B.22. *(Rellich-Kondrachov Compactheid Stelling). Zij $\Omega \subset \mathbb{R}^d$ een begrensd Lipschitz domein. De ruimte $H^1(\Omega)$ is altijd compact ingebed in $L^2(\Omega)$:*

$$H^1(\Omega) \hookrightarrow\hookrightarrow L^2(\Omega).$$

Zwakke convergentie

Zij V een Hilbert ruimte en V^* zijn duale.

Definitie B.23. *De rij $u_n \in V$ wordt zwak convergent genoemd als er een $u \in V$ bestaat zodat*

$$\lim_{n\to\infty} \langle f, u_n \rangle = \langle f, u \rangle, \quad \forall f \in V^*.$$

Het element u wordt de zwakke limiet van u_n genoemd. We noteren het met $u_n \rightharpoonup u$ in V.

Lemma B.24. *(Eigenschappen van zwakke convergentie). Zij X een Banach ruimte en Y een genormeerde ruimte.*

(i) Een zwakke limiet is, indien hij bestaat, uniek;

(ii) De sterke convergentie in X impliceert de zwakke convergentie;

(iii) Zij X een Hilbert ruimte met norm $\|\cdot\|_X$. Stel dat $u_n \rightharpoonup u$ en $\|u_n\| \to \|u\|$ in V. Dan $u_n \to u$ in V;

(iv) Elke zwak convergentie rij in X is begrensd, i. e. $\|u_n\| \leq C$.

Compactheid

Een relatief compacte deelruimte Y van een topologische ruimte X is een deelruimte wiens sluiting compact is. In het geval de topologie voorzien is van een metriek, of algemener, wanneer rijen kunnen gebruikt worden om compactheid te testen, dan wordt het criterium voor relatieve compactheid dat elke rij in Y een convergente deelrij heeft in X.

Definitie B.25. *Zij $K \subset C(\overline{\Omega})$ voor $\Omega \subset \mathbb{R}^d$. De verzameling K wordt uniform equicontinu genoemd indien*

$$\forall \epsilon > 0 \exists \delta > 0 : \forall u \in K \; \forall \boldsymbol{x}, \boldsymbol{y} \in \overline{\Omega} \quad |\boldsymbol{x} - \boldsymbol{y}| < \delta \Rightarrow |u(\boldsymbol{x}) - u(\boldsymbol{y})| < \varepsilon.$$

De verzameling K is equibegrensd als

$$|u(\boldsymbol{x})| \leq C, \quad \forall \boldsymbol{x} \in \overline{\Omega}, \forall u \in K.$$

De fundamentele Arzelà-Ascoli Stelling geeft een karakterisatie van compacte deelverzamelingen in $C(\overline{\Omega})$ gebruik makend van de equibegrensdheid en uniforme equicontinuïteit eigenschappen.

Stelling B.26. *(Arzelà-Ascoli). Zij Ω een begrensd domein in $\mathbb{R}^d$. Dan is de deelverzameling K van $C(\overline{\Omega})$ relatief compact als en slechts als ze equibegrensd is en uniform equicontinu.*

Zwakke compactheid

Definitie B.27. *Een verzameling M in een Banach ruimte X wordt zwak compact genoemd als voor elke begrensde rij $u_n \in M$ een zwakke convergente deelrij naar $u \in M$ kan gevonden worden, i. e. ,*

$$\langle f, u_{n_k} \rangle \to \langle f, u \rangle, \quad \forall f \in X^*.$$

Stelling B.28. *(Zwakke compactheid). Zij V een reflexieve Banachruimte of zij V een separabele Hilbert ruimte. Dan is elke begrensde verzameling in V zwak compact.*

Lax-Milgram

Beschouw de elliptische partiële differentiaalvergelijking

$$Au = f,$$

met A een lineaire differentiaaloperator van tweede orde en f een continue functie. Het opstellen van de variationele formulering bestaat uit drie stappen:

- Vermenigvuldig de gegeven partiële differentiaalvergelijking met een testfunctie φ en integreer daarna over het domein Ω,
- Pas de stelling van Green toe (partiële integratie),
- Kies een gepaste testruimte V.

Bij het bestuderen van partiële differentiaalvergelijkingen is het gebruikelijk om de variationele formulering te schrijven in de vorm $a(u, \varphi) = (f, \varphi)$, $\forall \varphi \in V$. Hierbij stelt $a(\cdot, \cdot) : V \times V \to \mathbb{R}$ een bilineaire operator voor. Een oplossing u van deze vergelijking wordt een zwakke (variationele) oplossing genoemd.

Lemma B.29. *(Lax-Milgram). De variationele formulering gedefinieerd door*

$$a(u, \varphi) = (f, \varphi) \quad \forall \varphi \in V,$$

heeft een unieke zwakke (variationele) oplossing als

1. *De vectorruimte V is een Hilbert ruimte.*
2. *De bilineaire vorm a is continu, i.e. er bestaat een $C_M > 0$ zodat*

$$|a(u, v)| \leq C_M \|u\|_V \|v\|_V \quad \forall u, v \in V.$$

3. *De bilineaire vorm a is V-elliptisch, i.e. er bestaat een $C_m > 0$ zodat*

$$C_m \|u\|_V^2 \leq a(u, u) \quad \forall u \in V.$$

4. *f is een lineair begrensde functionaal op V, i.e. $f \in V^*$.*

C Matlab Code

C.1 Semi-Impleciete Perona-Malik vergelijking

```
%% Arnaud Devos - master student Ghent University
% December 2016
% Implementation of an implicit semi-linear sheme of the PM-equation
%-------------------------------------------------------------------
%% Input paramaters:
%
% u0 (m x n uint8 matrix) - the initial image
% timeStep (integer) - the time step on wich the PDE moves forward in time
% finalTime (integer) - the final time T on with the PDE is defined
% option (1 or 2) - the diffusion function g:
%                   option == 1 -> g(x)=e^(-1/2*|(x/K)|^a)
%                   option == 2 -> g(x)=1/(1+(x/K)^a)
% a (integer in [0,2]) - the power parameter in the function g
%-------------------------------------------------------------------
%% Returns:
% images (cell array) - the process of the image is stored in every
% timestep i in images{i}, the final result is given by images{end}
%-------------------------------------------------------------------
%% References:
%
% Peter Kovesi
% School of Computer Science & Software Engineering
% The University of Western Australia
%
%%
function images = PeronaMalik(u0, scaleStep,finalTime, option,a)
if nargin<5
    a=2;
end
images={};
```

```
u0 = double(u0);
diff = u0; %dummy
[m,n]=size(u0);

%partition of time interval
N=finalTime/scaleStep;

%estimate conduction parameter K
[Gx, Gy] = imgradientxy(diff,'central');
Gx(:,1)=0;
Gx(:,end)=0;
Gy(1,:)=0;
Gy(end,:)=0;
grad=(Gx.^2+Gy.^2).^(1/2);
K=1.4826*median(abs(grad(:)-median(grad(:))))

if option==1
    g=@(x) exp(-1/2*(x/K).^a);
elseif option==2
    g=@(x) 1./(1+(x/K).^a);
end

for l = 1:N
    diffl=padarray(diff,[1 1],'symmetric');

    [Gx, Gy] = imgradientxy(diffl,'central');
    Gx(:,1)=0;
    Gx(:,end)=0;
    Gy(1,:)=0;
    Gy(end,:)=0;
    grad=(Gx.^2+Gy.^2).^(1/2);

    gN=(g(grad(2:end-1,2:end-1))+g(grad(1:end-2,2:end-1)))/2;
    gS=(g(grad(2:end-1,2:end-1))+g(grad(3:end,2:end-1)))/2;
    gE=(g(grad(2:end-1,2:end-1))+g(grad(2:end-1,3:end)))/2;
    gW=(g(grad(2:end-1,2:end-1))+g(grad(2:end-1,1:end-2)))/2;

    gN(1,:)=0;
    gS(end,:)=0;
```

```
    gW(:,1)=0;
    gE(:,end)=0;

    D0=ones(m*n,1)+scaleStep*(gN(:)+gW(:)+gE(:)+gS(:));
    D1=-scaleStep*gS(:);
    D2=-scaleStep*gE(:);
    D3=-scaleStep*gN(:);
    D4=-scaleStep*gW(:);
    B=[D0 D1  D2 D3 D4];
    d=[0 -1 -m 1 m];
    M=spdiags(B,d,m*n,m*n);
    u=diff(:); %make vector of image
    L = ichol(M);
    z = pcg(M,u,1e-8,100,L,L');
    diff=reshape(full(z),[m,n]); %get matrix back
    %convert to unsigned 8-bit integer
    I=uint8(diff);
    images{l}=I; %store image



end
end
```

C.2 Semi-impliciete geregulariseerde Perona-Malik vergelijking

```
%% Arnaud Devos - master student Ghent University
% April 2017
% Implementation of an implicit semi-linear sheme of the
regularized PM-equation
%-------------------------------------------------------------------
%% Input paramaters:
%
% u0 (m x n uint8 matrix) - the initial image
% timeStep (integer) - the time step on wich the PDE moves forward in time
% finalTime (integer) - the final time T on with the PDE is defined
% option (1 or 2) - the diffusion function g:
%                   option == 1 -> g(x)=e^(-|(x/K)|^2)
%                   option == 2 -> g(x)=1/(1+(x/K)^2)
% s (integer) - parameter of the heat equation
%-------------------------------------------------------------------
%% Returns:
% images (cell array) - the process of the image is stored in every
% timestep i in images{i}, the final result is given by images{end}
%-------------------------------------------------------------------
%% Makes use of other functions:
% LinearHeat
%-------------------------------------------------------------------
%% References:
%
% Peter Kovesi
% School of Computer Science & Software Engineering
% The University of Western Australia
%
%%
function images = PeronaMalik_regular(u0, scaleStep,finalTime,option,s)

images={};

u0 = double(u0);
diff = u0; %dummy
[m,n]=size(u0);

%partition of time interval
```

```
N=finalTime/scaleStep;

%estimate conduction parameter K
[Gx, Gy] = imgradientxy(diff,'central');
Gx(:,1)=0;
Gx(:,end)=0;
Gy(1,:)=0;
Gy(end,:)=0;
grad=(Gx.^2+Gy.^2).^(1/2);
K=1.4826*median(abs(grad(:)-median(grad(:))));


if option==1
    g=@(x) exp(-1/2*(x/K).^2);
elseif option==2
    g=@(x) 1./(1+(x/K).^2);
end


for l = 1:N
    %compute convolution
    diffl=padarray(diff,[1 1],'symmetric');

    conv=LinearHeat(diffl,s);
    [Gx, Gy] = imgradientxy(conv,'central');
    Gx(:,1)=0;
    Gx(:,end)=0;
    Gy(1,:)=0;
    Gy(end,:)=0;
    grad=(Gx.^2+Gy.^2).^(1/2);

    gN=(g(grad(2:end-1,2:end-1))+g(grad(1:end-2,2:end-1)))/2;
    gS=(g(grad(2:end-1,2:end-1))+g(grad(3:end,2:end-1)))/2;
    gE=(g(grad(2:end-1,2:end-1))+g(grad(2:end-1,3:end)))/2;
    gW=(g(grad(2:end-1,2:end-1))+g(grad(2:end-1,1:end-2)))/2;

    gN(1,:)=0;
    gS(end,:)=0;
    gW(:,1)=0;
    gE(:,end)=0;
```

```
    D0=ones(m*n,1)+scaleStep*(gN(:)+gW(:)+gE(:)+gS(:));
    D1=-scaleStep*gS(:);
    D2=-scaleStep*gE(:);
    D3=-scaleStep*gN(:);
    D4=-scaleStep*gW(:);
    B=[D0 D1  D2 D3 D4];
    d=[0 -1 -m 1 m];
    M=spdiags(B,d,m*n,m*n);
    u=diff(:); %make vector of image
    L = ichol(M);
    z = pcg(M,u,1e-8,100,L,L');
    diff=reshape(full(z),[m,n]); %get matrix back
    %convert to unsigned 8-bit integer
    I=uint8(diff);
    images{l}=I; %store image

end
end
```

C.3 Semi-impliciete lineaire warmtevergelijking (1-staps)

```
%% Arnaud Devos - master student Ghent University
% April 2017
% Implementation of an one step implicit semi-linear sheme of the heat-equation
%-------------------------------------------------------------------
%% Input paramaters:
%
% u0 (m x n uint8 matrix) - the initial image
% sigma - parameter of the Green function of the fundamental solution
%-------------------------------------------------------------------
%% Returns:
% image (m x n double matrix) - image at final time sigma
%-------------------------------------------------------------------
%% References:
%
% Peter Kovesi
% School of Computer Science & Software Engineering
% The University of Western Australia
```

```
%
%%

function image = LinearHeat(u0, sigma)


u0 = double(u0);
diff = u0; %dummy
[m,n]=size(u0);
scaleStep=sigma;

gN=ones(m,n);
gN(1,:)=0;
gS=ones(m,n);
gS(m,:)=0;
gE=ones(m,n);
gE(:,n)=0;
gW=ones(m,n);
gW(:,1)=0;

D0=ones(m*n,1)+scaleStep*(gN(:)+gW(:)+gE(:)+gS(:));
D1=-scaleStep*gS(:);
D2=-scaleStep*gE(:);
D3=-scaleStep*gN(:);
D4=-scaleStep*gW(:);
B=[D0 D1  D2 D3 D4];
d=[0 -1 -m 1 m];
M=spdiags(B,d,m*n,m*n);
u=diff(:); %make vector of image
L = ichol(M);
u = pcg(M,u,1e-8,100,L,L');
diff=reshape(full(u),[m,n]); %get matrix back
image=diff;


end
```

Bibliografie

[1] E. Acerbi, V. Chiadò Piat, G. Dal Maso, en D. Percivale. An extension theorem from connected sets, and homogenization in general periodic domains. *Nonlinear Analysis: Theory, Methods & Applications*, 18(5):481–496, 1992.

[2] E. Bänsch en K. Mikula. Adaptivity in 3d image processing. *Computing and Visualization in Science*, 4(1):21–30, 2001.

[3] M. Black, G. Sapiro, D. Marimont, en D. Heeger. Robust anisotropic diffusion. *IEEE Transactions on image processing*, pages 421–432, 1998.

[4] T. Brox. *From Pixels to Regions: Partial Differential Equations in Image Analysis.* PhD thesis, Saarland University, Saarbrücken, Germany, 2005.

[5] W. Burger en M. J. Burge. *Digital image processing: an algorithmic introduction using Java*. Springer, 2016.

[6] F. Cagnetti en L. Scardia. An extension theorem for sbv functions and an application to homogenization of mumford-shah type energies. *Preprint*, 2008.

[7] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[8] F. Catté, P. Lions, J. Morel, en T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical analysis*, 29(1):182–193, 1992.

[9] L. Dascal. *Well-posedness and maximum principle for PDE based models in image processing.* PhD thesis, Tel-Aviv University, 2006.

[10] H. De Bie. Function Spaces. Course notes Ghent University, 2014-2015.

[11] H. De Meyer en V. Fack. Optimisation. Course notes Ghent University, 2014-2015.

[12] R. Eymard, T. Gallouët, en R. Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

[13] G. H. Golub en C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[14] R. C. Gonzalez en R. E. Woods. *Digital Image Processing.* Pearson, 2007.

[15] R. C. Gonzalez, R. E. Woods, en S. L. Eddins. *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., 2003.

[16] K. Höllig en J. Nohel. A diffusion equation with a nonmonotone constitutive function. In *Systems of Nonlinear Partial Differential Equations*, pages 409–422. Springer, 1983.

[17] J. Kačur. *Method of Rothe in evolution equations*. Springer, 1986.

[18] J. Kačur en K. Mikula. Solution of nonlinear diffusion appearing in image smoothing and edge detection. *Applied Numerical Mathematics*, 17(1):47–59, 1995.

[19] B. Kawohl. Variational versus pde-based approaches in mathematical image processing. In *CRM Proceedings and Lecture Notes*, volume 44, pages 113–126, 2008.

[20] S. Kichenassamy. The perona–malik paradox. *SIAM Journal on Applied Mathematics*, 57 (5):1328–1342, 1997.

[21] J. Koenderink. The structure of images. *Biological cybernetics*, 1984.

[22] P. D. Kovesi. Matlab and octave functions for computer vision and image processing. *Online: http://www. csse. uwa. edu. au/˜ pk/Research/MatlabFns/# match*, 2000.

[23] Z. Krivă. Explicit FV scheme for the Perona-Malik equation. *Computational Methods in Applied Mathematics Comput. Methods Appl. Math.*, 5(2):170–200, 2005.

[24] Z. Krivá en K. Mikula. An adaptive finite volume scheme for solving nonlinear diffusion equations in image processing. *Journal of Visual Communication and Image Representation*, 13(1-2):22–35, 2002.

[25] C. Ley. Statistical Inference. Course notes Ghent University, 2015-2016.

[26] Mathworks. Makers of Matlab and Simulink. `https://nl.mathworks.com`.

[27] K. Mikula. Image processing with partial differential equations. In *Modern methods in scientific computing and Applications*, pages 283–321. Springer, 2002.

[28] K. Mikula en N. Ramarosy. Semi-implicit finite volume scheme for solving nonlinear diffusion equations in image processing. *Numerische Mathematik*, 89(3):561–590, 2001.

[29] K. W. Morton en D. F. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press, New York, NY, USA, 2005. ISBN 0521607930.

[30] J. Nečas. *Introduction to the theory of nonlinear elliptic equations*. Chichester: John Wiley & Sons Ltd, 1986.

[31] P. Perona en J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 1990.

[32] J. Phillpot. Noise Reduction with the heat equation and the perona-malik equation. 2014.

[33] J. Ralli. PDE Based Image Diffusion and AOS. 2014.

[34] T. Roubíček. *Nonlinear partial differential equations with applications*, volume 153 of *ISNM*. Birkhäuser Verlag, Basel, 2005.

[35] J. W. Schlasche. Mathematical Methods of Image Processing to Characterize Micro Cups. Master's thesis, Universität Bremen, 2011.

[36] M. Slodička. Apprroximation Methods For Boundary Value Problems. Course notes Ghent University, 2015-2016.

[37] M. Slodička. Partial Differential Equations. Course notes Ghent University, 2015-2016.

[38] M. Slodička. Applied Functional Analysis. Course notes Ghent University, 2015-2016.

[39] A. Sparavigna. Fractional differentiation based image processing. *arXiv preprint arXiv:0910.2381*, 2009.

[40] H. J. Trussell en M. J. Vrhel. *Fundamentals of digital imaging*. Cambridge University Press, 2008.

[41] C. Tsiotsios en M. Petrou. On the choice of the parameters for anisotropic diffusion in image processing. *Pattern recognition*, pages 1369–1381, 2013.

[42] G. University. Google Books Project. `http://www.ugent.be/nl/univgent/bibliotheek/ugent-google.htm`.

[43] M. Vajnberg. *Variational method and method of monotone operators in the theory of nonlinear equations.* John Wiley & Sons, 1973.

[44] K. Van Bockstal. *Numerical techniques for partial differential equations in superconductivity and thermoelasticity*. PhD thesis, Ghent University, 2015.

[45] M. Van Daele. Numerical Methods for Differential Equations. Course notes Ghent University, 2015-2016.

[46] J. Van der Jeugt. Numerical Analysis. Course notes Ghent University, 2013-2014.

[47] D. Ventzas. *Advanced image Acquisition, Processing Techniques and Applications i.* InTech, 2012.

[48] J. Weickert. A review of nonlinear diffusion filtering. In *International Conference on Scale-Space Theories in Computer Vision*, pages 1–28. Springer, 1997.

[49] J. Weickert. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

[50] M. Wielgus. Perona-malik equation and its numerical properties. *arXiv preprint arXiv:1412.6291*, 2014.

[51] Wikipedia. The Free Encyclopedia. `http://en.wikipedia.org/`.

[52] A. Witkin. Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 150–153. IEEE, 1984.

[53] Q. Yang, D. Chen, T. Zhao, en Y. Chen. Fractional calculus in image processing: a review. *Fractional Calculus and Applied Analysis*, 19(5):1222–1249, 2016.

[54] E. Zeidler. *Nonlinear Functional Analysis and its Applications II/B: Nonlinear Monotone operators.* Springer-Verlag, 1990.

Lijst van figuren