University Mohamed Boudiaf-Msila
Faculty of Mathematics and computer sciences
Department of Mathematics

Master Thesis

Field : Mathematics and computer sciences
Branch : Mathematics
Option : PDE And Application

Theme

Finite Difference Schemes For Image Edge Enhancement Models

Presented by :
M^r Ferradi Ali

Publicly defended on : 25/06/2022.

Before the jury composed of :

BEN HAMIDOUCHE.N	Prof	University of Msila	**President**.
CHOUDER.R	M.C.B	University of Msila	**Supervisor.**
MERZOUGUI.A	Prof	University of Msila	**Examiner.**

Academic year 2021 / 2022

Acknowledgments

*First, I would like to thank my supervisor, Dr. CHOUDER Rafaa , who has entrusted me by supervising me in this work. I would like to thank him for being so gentle and for the advice he gave me during the fulfillment of this thesis.
I extend my thanks to the professors and members of the jury.
Again, I would like to extend my thanks and gratitude to the professors who taught me during the past two years and provided me with advice.
Finally, I thank all those who helped in this work.*

Contents

Introduction

Partial differential equations (PDEs) constitute a natural framework to model processes in numerous real-world applications, ranging from physics over life sciences to economy. Thus, it is not surprising that they have also contributed substantially to the mathematical foundations of signal and image analysis. For instance, they appear as Euler-Lagrange equations when solving continuous optimization problems that result from variation models [3][9] or regularizations of ill-posed problems [4]. It has also been shown that they are the natural setting for scale-spaces [1], they are successfully used for image enhancement [33], inpainting [29], and image compression [12]. PDE-based models benefit from many decades of research on their theoretical foundations and efficient numerical algorithms. Since they are continuous concepts, it is also very easy to incorporate useful invariance such as rotation invariance.
One of the most fascinating aspects of PDE-based image analysis is its capability to unify a number of existing methods in image analysis. This has led to deeper structural insights as well as to novel algorithms. For instance, PDE formulations and connections to PDE-based image analysis are known for Gaussian smoothing [15], dilation and erosion [1] [6] [32], morphological amoebas [35], mean curvature motion [18][1] and nonlinear diffusion filtering [23][8].
Since mean curvature motion and nonlinear diffusion filtering are classical methods in image processing for which feature directions in the image are important. Usually the two prominent directions for the local geometry in the image are the direction of the level or isophote (along an edge) and its orthogonal direction, the flow-line (across an edge). Choosing the amount of diffusion along these two directions appropriately gives a various range of different methods [2, 26, 20, 7].
A prominent example where we only have diffusion along edges is mean curvature flow. Its theoretical properties have first been investigated by Gage, Hamilton and Huisken in the 1980s [10, 11, 14]. It is known that a plane curve moving with normal speed equal to its curvature will shrink to a point, its shape becoming smoother and circular. More complicated phenomena are expected in higher dimensions, but no classification is available. In the context of image processing, following the work of Osher and Sethian [22, 30], a more rigorous view with the realization that the iso-intensity contours of an image can be moved under their curvature was achieved. This lead to a series of papers, where the images viewed as a set of level contours and moving then under their curvature. Image

smoothing by way of level set curvature motion [17, 18], thwarts the diffusion in the edge direction, thereby preserving the edge information. This work showed that in addition to this basic approach, a natural stopping criterion can also be chosen to prevent over smoothing a given image. A general mathematical framework for feature-preserving image smoothing that applies seamlessly to Gray-level, vector-value (color) images, volumetric images and movies is achieved by Sochen, Kimmel, and Malladi in [31, 28]. The main idea is to view the image as a two dimensional manifold embedded in a hybrid spatial-feature space. The authors in [28] showed that many classical geometric flows emerge as special cases in this view as well as a new flow, the so called Beltrami flow that moves a Gray level image under a scaled mean curvature. By following a different approach, Yezzi in [37] arrived at a similar equation.
Smoothing of noisy images presents usually a numerical integration of a parabolic PDE in scale or two dimensions in space. This often the most time consumptive component of image processing algorithms.
The explicit numerical integration scheme is conditionally stable. Thus, the unconditionally stable numerical scheme becomes an important matter.
The method based on Additive Operator Split (AOS), applied originally by Weickert for the nonlinear diffusion flow, may be applied for the Beltrami equation. This method makes it possible to develop the unconditionally stable semi-implicit finite difference schemes for image filtering.
In this work, we study some linear and non-linear filters and their effect on Images Processing. This work is divided into four chapters. The first chapter deals with the image from a mathematical point of view (definitions and applications of the image in reality), then we touched on the finite difference and some important theories (stability and consistency) The second chapter deals with the heat diffusion equation, where we have studied it in terms of finite difference, stability of types (Explicit-Implicit schemes) to its applications in image processing. In the next chapter, we dealt with the PM equation, where we studied it in 1D and 2D in terms of the stability of types (Explicit-Semi Explicit schemes) We also touched on a new technique, AOS(Additive Operator Splitting) and finally, the applications of the Perona-Malik equation in image processing. The last chapter we devoted to studying a new type of filter (Beltrami Flow) using AOS(Additive Operator Splitting) technique and its applications in image processing.

CHAPTER 1

PDE BASED IMAGE PROCESSING

Partial differential equations (PDEs) are one of the most important mathematical tools in image processing and computer vision. As they are appear in diffusion methods as well as in the Euler-Lagrange equations of variational models, they have been considered appear in many image processing tasks. Consequently, this chapter cannot give a complete overview about the field of PDE based image processing. Instead we pick out mainly those methods that will be utilized in the other chapters of this thesis.

1.1 The Image Society

Faced with a society in evolution ever faster, our society very well be considered as an image society. This is not only due to image is power and importance as a mean of communication but also for it is so easy, compact, and very famous existing tool used by all the people and everywhere to describe, express and represent the physical world. It captures a moment, an event that it freezes and offers us to analysis, to ameliorate their quality, enhance some characteristics to efficiently combine different pieces of information. Advertising, Photography, video games, ..etc., our daily lives of images.

We could also mention many different applications where image processing is concerned. For examples, medical imaging, satellite and aerial imaging, forecasting the weather, fingerprint analysis, robotics, quality control, Multimedia data management, video processing, restoration of old movies, etc. (See Figure 1.1).

Without necessarily knowing it, we are consumers of image processing on a daily basis.

1.1.1 Image processing

The goal of image processing and computer vision is to process images in such a way that they are easier to interpret by human beings or better to process by further algorithms. Computer Vision tries to do what a human brain does with the retinal input, it includes understanding and predicting the visual input. That could consist of segmentation, recognition, reconstruction (3D) and prediction (over video data). Classically, many Computer Vision algorithms employed image processing and machine learning or sometimes other methods (e.g Variational Methods, Combinatorial approaches,...) to do the mentioned tasks.

Image processing focuses on enhancing the quality of single images. Image processing algorithms are used for:

- Extraction of important image structures such as edges and corners (since the human visual system is very sensitive to this kind of discontinuities).
- Segmentation: dividing the image into regions of constant color where one has discontinuities at the region boundaries.
- De-blurring: Reconstruction of a visually better image from a blurred image (since in a blurred image edges are smeared and dislocated).
- De-noising: Removing noise from a noisy image.

Image processing has an enormous range of applications; almost every area of science and technology can make use of image processing methods. Here is a short list just to give some indication of the range of image processing applications.

- **1. Medicine**
  ◦ Inspection and interpretation of images obtained from X-rays, MRI or CAT scans.
  ◦ Analysis of cell images, of chromosome karyotypes.
- **2. Agriculture**
  ◦ Satellite/aerial views of land, for example to determine how much land is being used for different purposes, or to investigate the suitability of different regions for different crops.
  ◦ Inspection of fruit and vegetables—distinguishing good and fresh produce from old.
- **3. Industry**
  ◦ Automatic inspection of items on a production line, inspection of paper samples.
- **4. Law enforcement**
  ◦ Fingerprint analysis, sharpening or deblurring of speed-camera images.

(a) X-Ray, CT, and MRI Scans

(b) A fingerprint Image

(c) Surveillance camera photo at Home

(d) Road Surveillance Camera Photo

(e) Pregnancy Follow-Up Picture

(f) Satellite farmland image

Figure 1.1: Figures showing some applications of image processing

1.1.2 Mathematical Representation of an image

In image processing, a continuous Gray-scale image is considered that is a two-dimensional function

$$\begin{aligned} u \ : \ & \mathbb{R}^2 \to \mathbb{R} \\ & (x,y) \to u(x,y) \end{aligned}$$

The variable (x, y), for digital images, actually belongs to $\mathbb{N}^2$, it is a pixel of the image. $u(x, y)$ represents the luminous intensity, or the Gray level of the image at the pixel (x, y).

Digital images are most commonly presented as a matrix of scalars for gray-scale images or vectors for color images. Digital Gray scale images on the other hand are sampled and quantized. Sampling is the discretization of the image domain. Here an image consists of Gray values of a rectangular point grid

$$\{u_{i,j} \setminus \ i = 0, ..., N-1 \text{ and } j = 0, ..., M-1\}.$$

A grid point (i, j) is called pixel. Here N and M is the width and height of the image in pixels, respectively. $u_{i,j}$ denotes the Gray value of pixel (i, j).

For a color image, it is not enough to know its Gray level for each pixel: it is necessary to know the intensity of each of the three channels of the fundamental colors, the red R, the green G, and the blue B. An color image can then be defined as a vector function

$$\begin{aligned} u \ : \ & \mathbb{R}^2 \to \mathbb{R}^3 \\ & (x,y) \to u(x,y) = (R(x,y); G(x,y); B(x,y)) \end{aligned}$$

1.2 Diffusion in Image Processing

1.2.1 Linear diffusion filtering

The first step to use PDEs for smoothing images was done in the beginning of eighties, when the idea of scale-space filtering has introduced by Witkin [34] and further developed by Koenderink [21].

The essential idea of this approach is quite simple: embed the original image in a family of derived images $u(x,y,t)$ obtained by convolving the original image $u_0(x,y)$ with a Gaussian kernel $G_{\sqrt{2t}}(x,y)$ of variance t:

$$u(x,y,t) = \left(G_{\sqrt{2t}} * u_0\right)(x,y)\,,$$

where

$$G_\sigma(x,y) = \frac{1}{2\pi\delta^2}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right).$$

This family of derived images may equivalently be viewed as the solution of the following heat equation or the linear diffusion equation

$$\begin{cases} \frac{\partial u}{\partial t} = \triangle u \\ u(x,y,0) = u_0(x,y) \end{cases},$$

for a function $u(x,y,t)$ is the smoothed image at t, and $\triangle = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ denote Laplacian operator, with the initial condition $u(x,y,0) = u_0(x,y)$, the original image.

Koenderink [21] motivates the diffusion equation formulation by stating two criteria:

- **Causality**: Any feature at a coarse level of resolution is required to possess a (not necessarily unique) "cause" at a finer level of resolution although the reverse need not be true. In other words, no spurious detail should be generated when the resolution is diminished.

- **Homogeneity and Isotropy**: The blurring is required to be space invariant.

The linear diffusion filter has its limitations: whether we smooth uniformly by a rotational symmetric Gaussian kernel, or diffuse the data equally in all directions, the process not only removes undesirable local extrema (noise) but also deforms important features of the image, blurs and dislocates edges. To overcome these drawbacks, we have to move to nonlinear filters; nonlinear diffusion offers an excellent alternative.

1.2.2 Nonlinear diffusion filtering

Overcoming the undesirable effects of linear smoothing filtering, such as blurring or dislocating the semantically meaningful edges of the image, nonlinear diffusion equations can be used because the nonlinear diffusion technique not only preserves the edge sharpness, it may also enhance it. This technique was firstly proposed by Perona and Malik [23] by stating three criteria:

- **Causality:** no spurious detail should be generated passing from finer to coarser scales.
- **Immediate localization:** boundaries should be sharp and coincide with the semantically meaningful boundaries at that resolution.
- **Piecewise smoothing:** intra-region smoothing should be preferred to inter-region smoothing.

To satisfy the second and third criteria, Perona and Malik proposed to change the diffusion coefficient D, (D is constant in linear diffusion); by introducing a space-time-variant diffusion coefficient. Therefore the Perona-Malik model can be written as

$$\begin{cases} \frac{\partial u}{\partial t} = div(g\left(|\nabla u|\right)\nabla u) \\ u(x,y,0) = u_0(x,y) \end{cases},$$

where div denotes the divergence operator, $u(x,y,t)$ is the smoothed image at time step t, $|\nabla u|$ is the gradient magnitude of u, and $g(x)$ is the diffusivity function. $g(x)$ should be a nonnegative, monotonically decreasing function with $g(0) = 1$, so that the diffusion is maximal within uniform regions, and approaching $zero$ at infinity, so that the diffusion is stopped across edges.

Nonlinear diffusion filtering can successfully smooth noise while respecting the region boundaries and small structures within the image, as long as some of its crucial parameters are determined or estimated correctly. According to Perona and Malik the choice of functions g leads to the desirable result of edges enhancement.

1.2.3 The Beltrami Flow

The main objective in early computer vision is to smooth images without destroying the semantic content, i.e.,edges, features, corners, etc. In other words, the boundaries between objects in the image should survive as long as possible along the scale, while homogeneous regions should be simplified and flattened in a rapid way. This is particularly important since the denoising of an image is usually a precursor to segmentation and representation, which often rely on edge fidelity. We present a fast method to move (image) manifolds that has many real applications in image analysis and visualization.

Let us denote by (Σ, g) the image manifold and its metric and by (M, h) the space-feature manifold and its metric, then the map $X : \Sigma \to M$ has the following measure[19]:

$$S\left[X^i, g_{\mu\nu}, h_{ij}\right] = \int d^m\sigma\sqrt{g}g^{\mu\nu}\partial_\mu X^i \partial_\nu X^j h_{ij}(X), \tag{1.1}$$

where m is the dimension of Σ, g is the determinant of the image metric, $g^{\mu\nu}$ is the inverse of the image metric, the range of indices is $\mu, \nu = 1, \ldots, \dim\Sigma$, and $i, j = 1, \ldots, \dim M$, and h_{ij} is the metric of the embedding space. This is a natural generalization of the L^2 norm to manifolds.
As an example, a gray level image can be treated as a $2D$ manifold embedded in R^3, i.e. a mapping $X : (x, y) \to (X^1 = x, X^2 = y, X^3 = U(x, y))$

Many scale-space methods, linear and nonlinear can be shown to be a gradient descent flows of this functional with appropriately chosen metric of the image manifold. The gradient descent equation is $X^i_t = -\frac{1}{\sqrt{g}}\frac{\delta S}{bX^2}$. minimizing the area action in equation(1.1). With respect to the feature coordinate U, we obtain the following Beltrami flow equation,

$$U_t = \frac{U_{xx}\left(U_y^2+1\right) - 2U_xU_yU_{xy} + U_{yy}\left(U_x^2+1\right)}{\left(1+U_x^2+U_y^2\right)^2}. \tag{1.2}$$

1.3 Finite-Difference Approximation to Derivatives

Let a function U derivatives and single-valued, finite and continuous functions of x, then by Taylor's theorem[27].

$$U(x+h) = U(x) + hU'(x) + \frac{1}{2}h^2U''(x) + \frac{1}{6}h^3U^{(3)}(x) + \frac{1}{4!}h^4U^{(4)}(x) + \ldots \tag{1.3}$$

and

$$U(x-h) = U(x) - hU'(x) + \frac{1}{2}h^2u''(x) - \frac{1}{6}h^3U^{(3)}(x) + \frac{1}{4!}h^4U^{(4)}(x) + \ldots \tag{1.4}$$

Addition of these expansions gives

$$U(x+h) + U(x-h) = 2U(x) + h^2U''(x) + O\left(h^4\right)$$

where $O\left(h^4\right)$ denotes terms containing fourth and higher powers of h. Assuming these are negligible in comparison with lower powers of h it follows that,

$$U''(x) = \left(\frac{\mathrm{d}^2U}{\mathrm{d}x^2}\right)_{x=x} \approx \frac{1}{h^2}\{U(x+h) - 2U(x) + U(x-h)\} \tag{1.5}$$

with a leading error on the right-hand side of order h^2.
Subtraction of equation (1.4) from equation (1.3) and neglect of terms of order h^3 leads to

$$U'(x) = \left(\frac{\mathrm{d}U}{\mathrm{d}x}\right)_{x=x} \approx \frac{1}{2h}\{U(x+h) - U(x-h)\} \tag{1.6}$$

with an error of order h^2. equation (1.6) clearly approximates the slope of the tangent at the point P by the slope of the chord AB, and is called a central-difference approximation (Figure 1.2). We can also approximate the slope of the tangent at P by either the slope of the chord PB, giving the forward-difference formula,

$$U'(x) \approx \frac{1}{h}\{U(x+h) - U(x)\},$$

or the slope of the chord AP giving the backward-difference formula,

$$U'(x) \approx \frac{1}{h}\{U(x) - U(x-h)\},$$

with an error of order h.

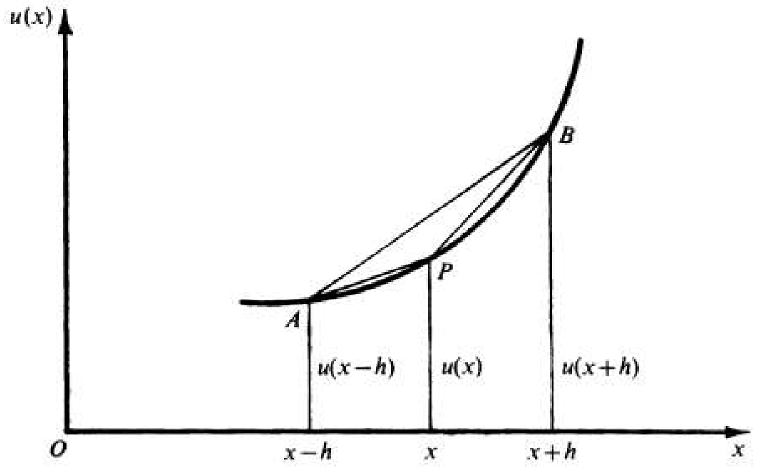


Figure 1.2: Approximations of the slope of the tangent at point P by the slope of the chord AB, and by the slope of the chord PB.

1.3.1 Notation for functions of several variables

Assume U is a function of the independent variables x and t. Subdivide the $x-t$ plane into sets of equal rectangles of sides $\Delta x = h, \Delta t = k$, by equally spaced grid lines parallel to Oy, defined by $x_i = ih, i = 0, \pm 1, \pm 2, \ldots$, and equally spaced grid lines parallel to Ox, defined by $t_j = jk, j = 0, 1, 2, \ldots$, as shown in figure 1.3. Denote the value of U at the representative mesh point $P(ih, jk)$ by

$$U_{i,j} = U(ih, jk).$$

Then by equation(1.5),

$$\left(\frac{\partial^2 U}{\partial x^2}\right)_{i,j} \approx \frac{U\{(i+1)h, jk\} - 2U\{ih, jk\} + U\{(i-1)h, jk\}}{h^2}.$$

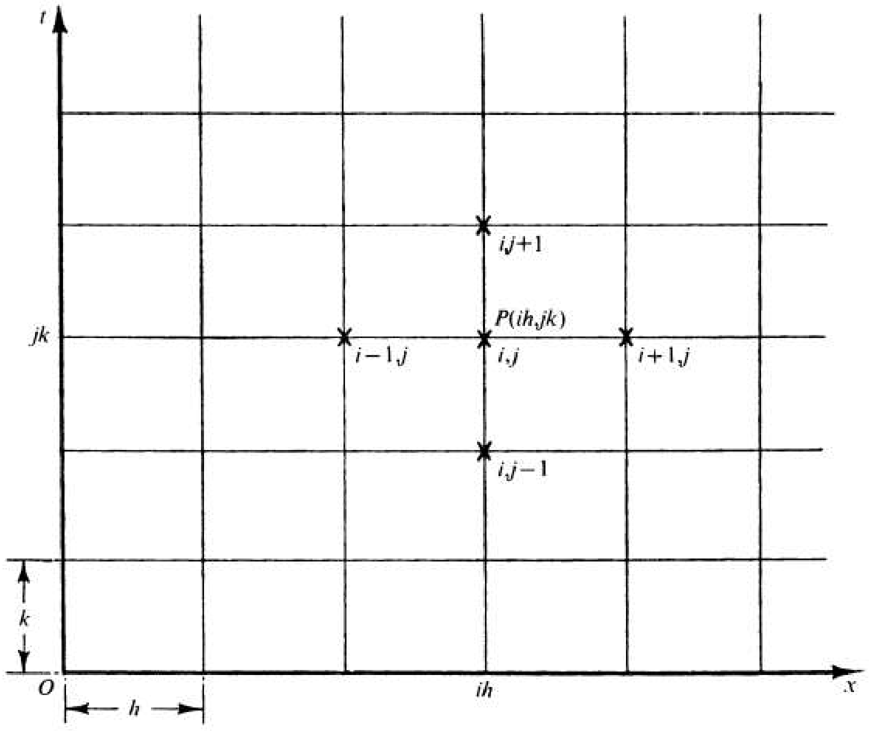


Figure 1.3: Approximated value of U at the representative mesh point P.

i.e.

$$\left(\frac{\partial^2 U}{\partial x^2}\right)_{i,j} \approx \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2},$$

with a leading error of order h^2.

With this notation the forward (resp. backward) difference approximation for $\partial U/\partial x$ at P is

$$\left(\frac{\partial U}{\partial x}\right)^{+}_{i,j} \approx \frac{U_{i+1,j} - U_{i,j}}{h}, \left(\frac{\partial U}{\partial x}\right)^{-}_{i,j} \approx \frac{U_{i,j} - U_{i-1,j}}{h},$$

with a leading error of $O(h)$.
The forward-difference approximation for $\partial U/\partial t$ at P is

$$\frac{\partial U}{\partial t} \approx \frac{U_{i,j+1} - U_{i,j}}{k},$$

with a leading error of $O(k)$.

1.3.2 Stability and consistency analysis

We now look at the concepts of consistency and stability which allow us to understand when a numerical solution to a PDE converges to the exact solution. We shall then apply them to the explicit scheme.

Definition 1.3.1. Consistency [27]

A finite difference scheme is consistent if the numerical solution computed after a fixed number of steps converges to the exact solution as h and k tend to zero. Consistency ensures that the finite difference equation converges to the original PDE.

Definition 1.3.2. Stability [27]

A finite difference scheme is stable if the numerical solution computed after a fixed time remains bounded as $h \to 0$.

Stability ensures that the numerical solution at a finite time does not blow up as the time-step is reduced to zero. Consistency and stability together ensure convergence of the numerical solution according to the following

Theorem 1.3.3. ***Lax-Richtmeyer [27]***

A finite difference approximation to a well posed linear initial value problem converges to the exact solution as k and h tend to zero if and only if it is consistent and stable.

1.3.3 Von Neumann (Fourier) stability analysis

Define the error in the numerical approximation as

$$\epsilon_j^n = U_j^n - u_j^n, \tag{1.7}$$

where u_j^n is the exact solution. Both the numerical solution U_j^n and the exact solution u_j^n satisfy PDE, therefore, the error ϵ_j^n also follows the discretized ODE. The spatial variation of error may be expanded in a finite Fourier series, in the interval L, as

$$\epsilon(x) = \sum_{m=1}^{M} A_m e^{ik_m x} \tag{1.8}$$

where $k_m = \frac{\pi m}{L}$, $m = 1, 2, \ldots, M$ and $M = L/h, i = \sqrt{-1}$. $e^{ik_m x}$ is the complex exponential. A_m is a function of time. Since the error tends to grow or decay exponentially with time, it is reasonable to assume that the amplitude varies exponentially with time; hence

$$\epsilon(x,t) = \epsilon_j^n = \sum_{m=1}^{M} e^t e^{ik_m x} \tag{1.9}$$

Since the difference equation for error is linear, it is enough to consider the growth of error of a typical term:

$$\epsilon_m(x,t) = \epsilon_j^n = e^t e^{ik_m x} \tag{1.10}$$

The goal is to show that the error incurred by a particular numerical scheme doesn't grow in the evolving steps of time. Define the amplification factor

$$G \equiv \frac{\epsilon_j^{n+1}}{\epsilon_j^n}$$

Then, the necessary condition for stability is:

$$|G| \leq 1 \text{ or } [-1 \leq G \leq 1] \tag{1.11}$$

CHAPTER 2

HEAT EQUATION

As we mentioned in the previous chapter, the first PDE that have been used in image processing is the Heat equation that realizes a spatial diffusion of the Gray values of a given image. It is a parabolic equation that reads:

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} &= \Delta u(t,x) \qquad \text{for } t \geq 0 \text{ and } x \in \Omega \\ u(0,x) &= f(x) \qquad \text{for } x \in \Omega \\ \frac{\partial u(t,x)}{\partial N} &= 0 \qquad \text{for } t > 0 \text{ and } x \in \partial\Omega \end{cases} \tag{2.1}$$

where Δ is the Laplacian operator, f is the initial temporal condition and $\frac{\partial u}{\partial N} = 0$ are Neumann boundary conditions. This model diffuses the initial condition f (that can be seen as an initial temperature) along time.

This chapter consist to numerical integration of the heat equation. Stability of finite difference schemes are established in one and two dimensional cases. We finished this chapter by simulation results for Gray level image.

2.1 Discretization of 1-D Heat Equation

In order to deeper understanding of numerical integration to the heat equation, we start by the one-dimensional case. We consider the following problem:

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} &= \frac{\partial^2 u(t,x)}{\partial x^2} \qquad \text{for } t \geq 0 \text{ and } x \in \Omega \\ u(0,x) &= f(x) \qquad \text{for } x \in \Omega \\ \frac{\partial u(t,x)}{\partial N} &= 0 \qquad \text{for } t > 0 \text{ and } x \in \partial\Omega \end{cases} \tag{2.2}$$

According to use right-shifted or left-decentered finite difference scheme in time, two different schemes are considering: The explicit scheme (FTCS), and the implicit scheme (BTCS).

2.1.1 The Explicit scheme (FTCS)

The explicit Euler scheme consists of using a finite difference scheme that is right-shifted in time. It is written:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} - \Delta u_i^n = 0, \text{ for } n = 0, 1, \dots, M, \text{ and } \Delta t = \frac{T}{M}$$

So the schema is written as follow:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}.$$

After simplification we get:

$$u_i^{n+1} = \frac{\Delta t}{\Delta x^2} u_{i-1}^n + (1 - 2\frac{\Delta t}{\Delta x^2}) u_i^n + \frac{\Delta t}{\Delta x^2} u_{i+1}^n.$$

We can write u in the form of a vector U, the matrix of this scheme is writing as:

$$U^{n+1} = AU^n, n = 0, 1, \dots, M,$$

where A is tridiagonal matrix and it only depends on Δt .

The matrix-format is writing in the following form:

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 1-\lambda & \lambda & 0 & 0 \\ \lambda & 1-2\lambda & \lambda & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \lambda & 1-2\lambda & \lambda \\ 0 & 0 & \lambda & 1-\lambda \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_N^n \end{pmatrix}.$$

This scheme only requires a matrix-vector product in each time step.

The explicit numerical integration scheme is conditionally stable. It imposes a constraint on the time step length, depending on the spatial resolution:

$$k \leq \frac{h^2}{2}.$$

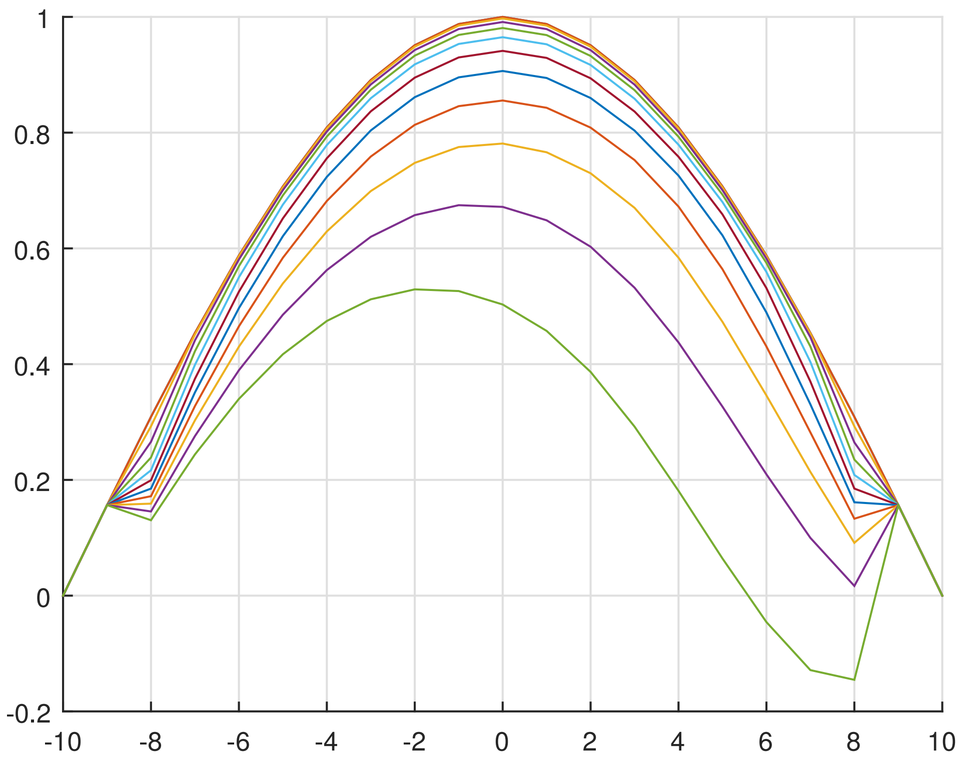


Figure 2.1: FTCS solution to the heat equation. The instability in the solution is now obvious.

2.1.2 The Implicit scheme (BTCS)

The implicit Euler scheme consists in using a left-decentered finite difference scheme in time

$$\frac{u_i^{n+1}-u_i^n}{\Delta t}=\frac{u_{i-1}^{n+1}-2u_i^{n+1}+u_{i+1}^{n+1}}{\Delta x^2} \tag{2.3}$$

or

$$u_i^n=-\frac{k}{\Delta x^2}u_{i-1}^{n+1}+(1+2\frac{\Delta t}{\Delta x^2})u_i^{n+1}-\frac{\Delta t}{\Delta x^2}u_{i+1}^{n+1}.$$

We will write u in the form of a vector U, the matrix writing of this scheme is:

$$U^n=AU^{n+1}, n=0,1,\dots,M,$$

The matrix writing is as follows:

$$\begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_N^n \end{pmatrix}=\begin{pmatrix} 1+\lambda & -\lambda & 0 & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & 0 & -\lambda & 1+\lambda \end{pmatrix}\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_N^{n+1} \end{pmatrix} \tag{2.4}$$

The advantage of this scheme is that it is unconditionally stable. But it requires the resolution of a linear system in each time step.

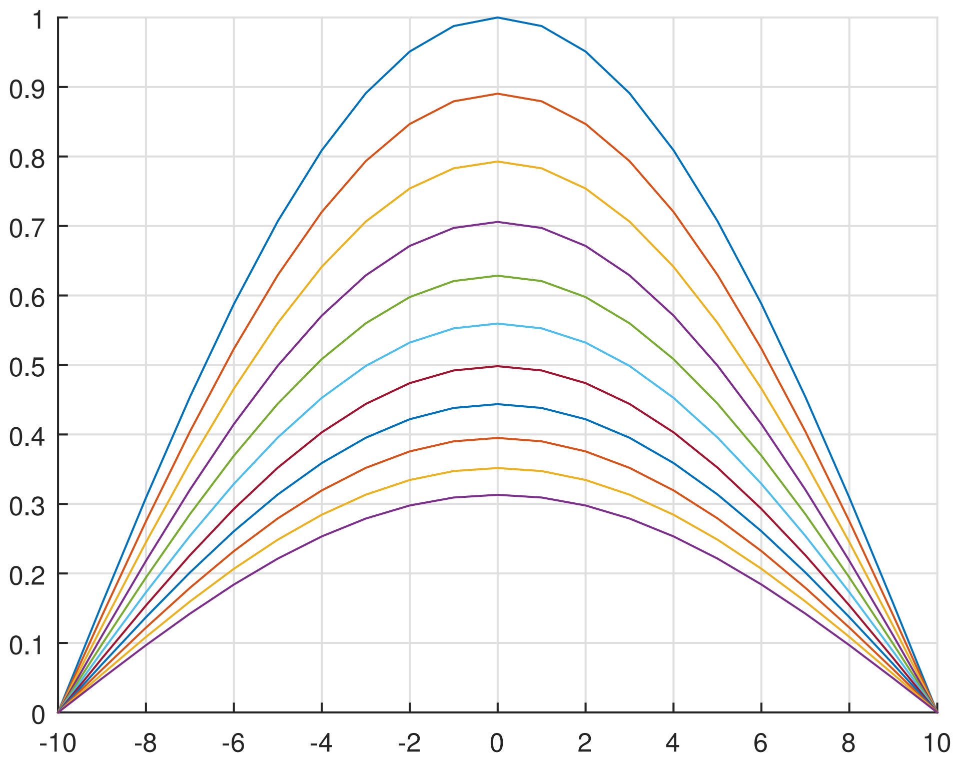


Figure 2.2: Stable BTCS solution to the heat equation.

2.2 Numerical Treatment in 2D

2.2.1 The Explicit scheme in 2D

In the case of two dimensions the explicit FTCS scheme reads

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2} \right), \tag{2.5}$$

or, with $\alpha = \Delta t / \Delta x^2$ and $\beta = \Delta t / \Delta y^2$

$$u_{i,j}^{n+1} = \alpha \left(u_{i+1,j}^{n} + u_{i-1,j}^{n} \right) + \beta \left(u_{i,j+1}^{n} + u_{i,j-1}^{n} \right) + (1 - 2\alpha - 2\beta) u_{i,j}^{n}$$

Assuming

$$\varepsilon_{i,j}^{n} = A^n e^{i(k_x x_j + k_y y_j)}$$

leads to the following relation for the amplification factor $A(k)$

$$A(k) = 1 - 4\alpha \sin^2\left(\frac{k_x \triangle x}{2}\right) - 4\beta \sin^2\left(\frac{k_y \triangle y}{2}\right)$$

In this case the stability condition reads

$$\alpha + \beta \leq \frac{1}{2}$$

This stability condition imposes a limit on the time step:

$$\Delta t \leq \frac{\Delta x^2 \Delta y^2}{2\left(\Delta x^2 + \Delta y^2\right)}$$

In particular, for the case $\Delta x = \Delta y$ we have

$$k \leq \frac{h^2}{4}$$

which is even more restrictive, than in the one-dimensional case.

2.2.2 The implicit scheme in 2D

To overcome the stability restriction of the FTCS method 2.5, we can use an implicit BTCS schema in the two-dimensional case. The schema reads:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\triangle t} = \frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{h^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\triangle y^2},$$

or

$$-\alpha\left(u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1}\right) + (1 + 2\alpha + 2\beta)u_{i,j}^{n+1} - \beta\left(u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}\right) = u_{i,j}^{n} \qquad (2.6)$$

Assuming

$$\varepsilon_{i,j}^{n} = A^n e^{i(k_x x_j + k_y y_j)}$$

leads to the following relation for the amplification factor $A(k)$

$$A(k) = \frac{1}{1 + 4\alpha \sin^2\left(\frac{k_x \triangle x}{2}\right) + 4\beta \sin^2\left(\frac{k_y \triangle y}{2}\right)}$$

This Scheme is unconditionally stable.

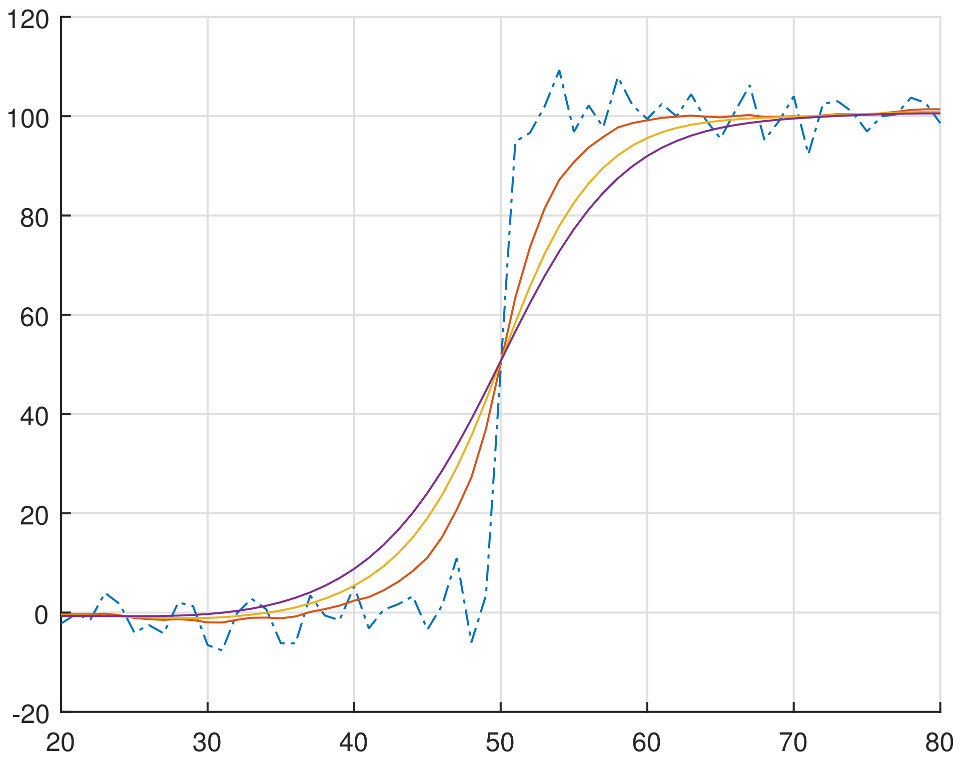


Figure 2.3: The 1-D edge profile (dot-dash) and its linear diffusion filtered versions (solid).

2.3 Simulation Results for 1-D Signal and Gray Level Images

In Figure 2.3 we can see that the approximate of the linear filter gives a good approximation except for the edge points.
In Figure 2.4 we can see that the linear filter has blurred noise and edges together.
We also note that linear filters work very quickly in implementation
Smoothing edges cause the image to lose its meaning(Because The edges is very useful in many applications such as pattern recognition and fingerprint and iris biometric identification, space science and robot vision).
This is one of the main drawbacks of linear filters, and this leads us to a new type of nonlinear filter(Perona-Malik) that preserves edges well compared to linear filters.

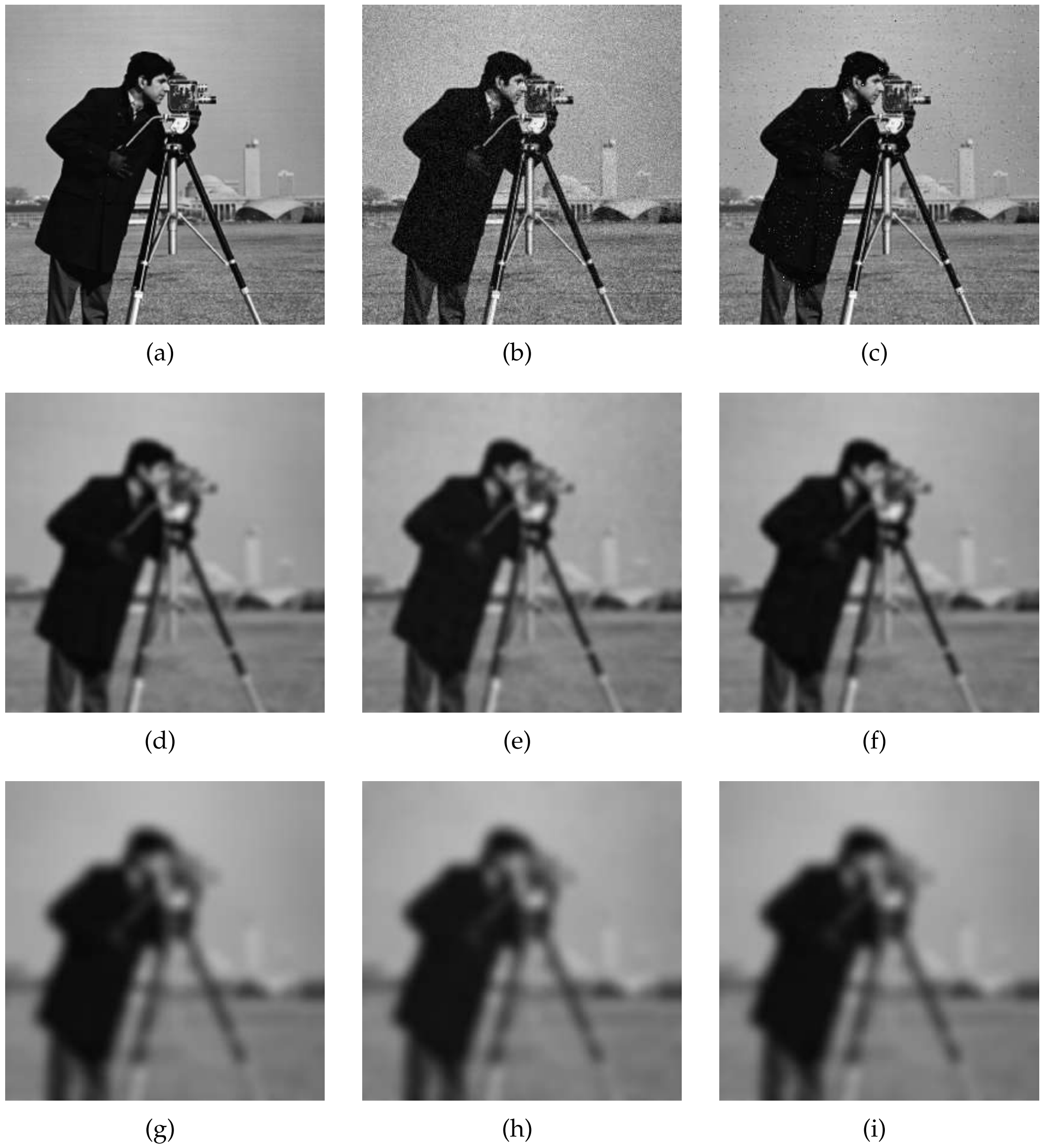


Figure 2.4: The original image and its linear diffusion filtered versions. a: Original Image, b: Gaussian Noise , c: salt and pepper. d,e and f: Image filtered by Linear-Filter after 5 Iteration. g,h and i: Image filtered by Linear-Filter after 20 Iteration

CHAPTER 3

PERONA-MALIK EQUATION

Smoothing of noisy images presents usually a numerical integration of a parabolic PDE in scale or two dimensions in space. This often the most time consumptive component of image processing algorithms.
The explicit numerical integration scheme is conditionally stable. It imposes a constraint on the time step length, depending on the spatial resolution $s = \Delta x = \Delta y$:

$$\Delta t \leq \lambda s^2.$$

Theoretical and numerical aspects of the nonlinear anisotropic diffusion were extensively studied by Weickert [33]. It proves to be difficult to define analytically the threshold value of λ for nonlinear smoothing. Thus, the unconditionally stable numerical scheme becomes an important matter.
The Additive Operator Splitting (AOS) Schemes were introduced by Weickert et al. as unconditionally stable for the nonlinear diffusion in image processing.

3.1 One-Dimensional Perona-Malik Model and Edges Enhancement

In order to establish the results concerning the enhancement of edges in Perona-Malik model [23],

$$\begin{cases} u_t = \dfrac{\partial}{\partial x}\left(g\left(|u_x|\right)u_x\right) & \text{in } \Omega \times (0,+\infty), \\ \dfrac{\partial u}{\partial n} = 0 & \text{in } \partial\Omega \times (0,+\infty), \\ u(x,0) = u_0(x) & \text{in } \Omega. \end{cases} \tag{3.1}$$

where $g(s)$ is a smooth non-increasing function with $g(0) = 1$, $g(s) > 0$ and $g(\infty) = 0$.
We analyze the way in which the gradients in the edge region evolve with time t. In other words, we need to find the function of $\frac{\partial(u_x)}{\partial t}$. In the case of a 1-D signal, the Perona-Malik diffusion equation can be written as

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial}{\partial x}\left(g(u_x)u_x\right) \\ &= \frac{\partial g(u_x)}{\partial x}u_x + g(u_x)u_{xx}. \end{aligned} \tag{3.2}$$

For ease of understanding, we define the flux function $\phi(x) = g(x).x$, where $g(x)$ is the diffusivity function. Therefore, we can write (3.2) as follow

$$\frac{\partial u}{\partial t} = \phi'(u_x)u_{xx}.$$

The change of the gradients, at any location of the edge, is given by:

$$\frac{\partial\left(u_x\right)}{\partial t} = \phi''(u_x)u_{xx}^2 + \phi'(u_x)u_{xxx}. \tag{3.3}$$

The sign of the right hand-side of equation (3.3) tells us whether a gradient u_x at location x is increasing $\left(\frac{\partial(u_x)}{\partial t} > 0\right)$ or decreasing $\left(\frac{\partial(u_x)}{\partial t} < 0\right)$, which means that the blurring/enhancing behaviour of the Perona-Malik filter depends on the sign of $\phi'(u_x)$. On the edge, the second derivative $u_{xx} = 0$ and $u_{xxx} \leq 0$ since at this point the gradient is maximum (See 3.1). The gradient at the edge decreases if $\phi'(u_x) > 0$, so the edge is blurred. If $\phi'(u_x) < 0$, the gradient at the edge increases with time while the neighbouring gradients decrease, this process leads to sharp edges. (See Figure 3.2).

3.1.1 Discretization of 1-D Perona-Malik equation

By simply using the finite differences introduced above, one way of discretising the right terms in 3.1 is using the central difference. First we apply the central

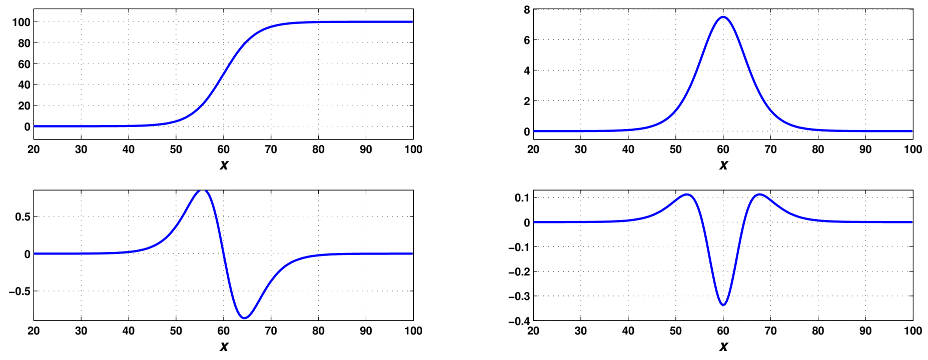


Figure 3.1: The edge function $u(x)$ and its first, second, and third derivatives, in the top left, top right, bottom left, and bottom right, respectively.

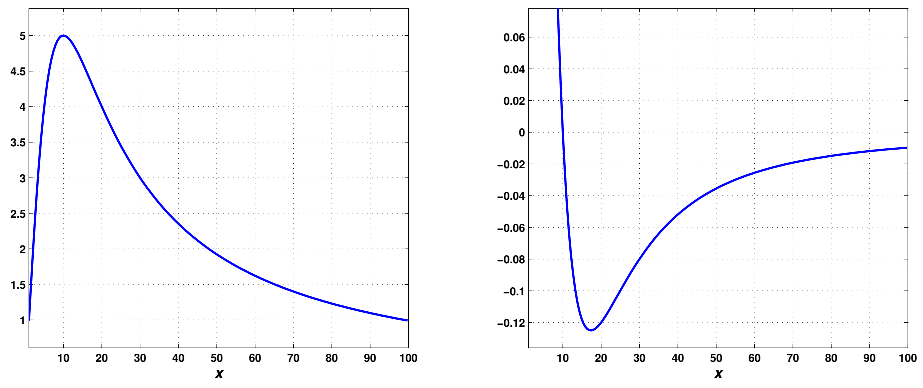


Figure 3.2: Flux function ϕ (left) and its derivative ϕ' (right).

difference and then the forward and the backward differences for approximating the corresponding derivatives.
The 'trick' here is to realise that $(g_x)(x+0.5,y)$ is actually the forward difference $D_x^+ g(x)$, while $(g_x)(x-0.5,y)$ is the backward difference $D_x^- g(x)$.
equation 3.4 shows the discretisations for $(g(|u_x|)u_x)_x$. This is the same discretisation as in the famous paper by Perona and Malik [23].

$$\begin{aligned}\frac{\partial}{\partial x}(g.u_x) &= \frac{\left(g\dfrac{\partial u}{\partial x}\right)(x+0.5)-\left(g\dfrac{\partial u}{\partial x}\right)(x-0.5)}{\Delta x}\\ &= \frac{g(x+0.5)\dfrac{\partial u}{\partial x}(x+0.5)-g(x-0.5)\times\dfrac{\partial u}{\partial x}(x-0.5)}{\Delta x}\\ &= g(x+0.5)\left[\frac{u(x+1)-u(x)}{\Delta x^2}\right]-g(x-0.5)\left[\frac{u(x)-u(x-1)}{\Delta x^2}\right]\end{aligned} \tag{3.4}$$

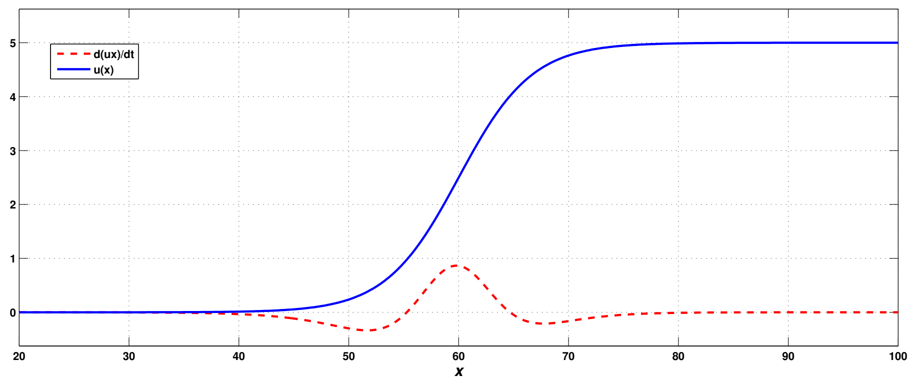


Figure 3.3: An edge function $u(x)$ and the rate of change of the edge gradient $\frac{\partial(u_x)}{\partial t}$ with time.

3.1.2 The Explicit scheme

The explicit scheme consists of using a finite difference scheme that is right-shifted in time. It is written:

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t} \tag{3.5}$$

Then,

$$\begin{aligned} u_i^{n+1} - u_i^n &= \frac{\Delta t}{\Delta x^2} u_{i-1}^n g_{i-\frac{1}{2}}^n - \frac{\Delta t}{\Delta x^2} u_i^n \left(g_{i+\frac{1}{2}}^n + g_{i-\frac{1}{2}}^n \right) + \frac{\Delta t}{\Delta x^2} u_i^n g_{i+\frac{1}{2}}^n \\ u_i^{n+1} &= \frac{\Delta t}{\Delta x^2} u_{i-1}^n g_{i-\frac{1}{2}}^n + u_i^n \left[1 - \frac{\Delta t}{\Delta x^2} \left(g_{i+\frac{1}{2}}^n + g_{i-\frac{1}{2}}^n \right) \right] + \frac{\Delta t}{\Delta x^2} g_{i+\frac{1}{2}}^n u_{i+1}^n \end{aligned}$$

we will write u in the form of a vector U, the matrix of this scheme is writing as:

$$U^{n+1} = AU^n, n = 0, 1, \ldots, M,$$

The matrix is writing as follows:

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 1-\beta_1 & \beta_1 & 0 & 0 \\ \alpha_2 & 1-(\alpha_2+\beta_2) & \beta_2 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \alpha_{N-1} & 1-(\alpha_{N-1}+\beta_{N-1}) & \beta_{N-1} \\ 0 & 0 & \alpha_N & 1-(\alpha_N+\beta_N) \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_N^n \end{pmatrix}$$

where: $\alpha_i = \frac{\Delta t}{\Delta x^2} g_{i+\frac{1}{2}}^n, \beta i = \frac{\Delta t}{\Delta x^2} g_{i-\frac{1}{2}}^n$ and A is tridiagonal matrix and it only depends on Δt .
This scheme only requires a matrix-vector product in each time step.

The explicit numerical integration scheme is conditionally stable. It imposes a constraint on the time step length, depending on the spatial resolution Δx:

$$\Delta t \leq \lambda \Delta x^2.$$

It proves to be difficult to define analytically the threshold value of λ for nonlinear diffusion, but various numerical numerical tests revealed that for nonlinear diffusion, the value is almost exactly the same as for the linear diffusion: $\lambda \approx \frac{1}{2}$. Thus, the unconditionally stable numerical scheme becomes an important matter.

3.1.3 The Semi-Implicit scheme

We have:

$$\begin{aligned} u_i^{n+1} - u_i^n &= \frac{\Delta t}{\Delta x^2} u_{i-1}^{n+1} g_{i-\frac{1}{2}}^n - \frac{\Delta t}{\Delta x^2} u_i^{n+1} \left(g_{i+\frac{1}{2}}^n + g_{i-\frac{1}{2}}^n \right) + \frac{\Delta t}{\Delta x^2} u_{i+1}^{n+1} g_{i+\frac{1}{2}}^n \\ & then \\ u_i^n &= \frac{\Delta t}{\Delta x^2} u_{i-1}^{n+1} g_{i-\frac{1}{2}}^n + u_i^{n+1} \left[1 + \frac{\Delta t}{\Delta x^2} \left(g_{i+\frac{1}{2}}^n + g_{i-\frac{1}{2}}^n \right) \right] - \frac{\Delta t}{\Delta x^2} g_{i+\frac{1}{2}}^n u_{i+1}^n \end{aligned} \tag{3.6}$$

We will write u in the form of a vector U, the matrix writing of this scheme is:

$$U^n = AU^{n+1}, n = 0, 1, \ldots, M,$$

The matrix is writing as follows:

$$\begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_N^n \end{pmatrix} = \begin{pmatrix} 1+\beta_1 & -\beta_1 & 0 & 0 \\ -\alpha_2 & 1+(\alpha_2+\beta_2) & -\beta_2 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & -\alpha_{N-1} & 1+(\alpha_{N-1}+\beta_{N-1}) & \beta_{N-1} \\ 0 & 0 & -\alpha_N & 1+\alpha_N \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_N^{n+1} \end{pmatrix}$$

where: $\alpha_i = \frac{\Delta t}{\Delta x^2} g_{i+\frac{1}{2}}^n$ and $\beta i = \frac{\Delta t}{\Delta x^2} g_{i-\frac{1}{2}}^n$.
The advantage of this scheme is that it is unconditionally stable for all values of the time step Δt.

3.2 2D Perona-Malik Equation

The two-dimensional Perona-Malik equation is written as follows:

$$\begin{cases} u_t = \operatorname{div}\left(g\left(|\nabla u| \right) \nabla u \right) & \text{in } \Omega \times (0, +\infty), \\ \dfrac{\partial u}{\partial n} = 0 & \text{in } \partial\Omega \times (0, +\infty), \\ u(x, 0) = u_0(x) & \text{in } \Omega. \end{cases} \tag{3.7}$$

3.2.1 Discretisation of div operator

Now that we know how to approximate first and second order derivatives, we can discretise the divergence, div, operator. Conceptually, we have two different cases:

$$\begin{aligned} &\mathrm{div}(\nabla f) \\ &\mathrm{div}(g(x,y,t)\nabla f) \end{aligned} \tag{3.8}$$

Here, physical interpretation of the divergence is, in a sense, that of diffusion . In the case of $div(\nabla f)$, diffusivity is the same in each direction, whereas in the case of $div(g(x,y,t)\nabla f)$, diffusivity is defined (or controlled) by the function g and is not necessarily the same in all the directions.
Mathematically, for a differentiable vector function $F = U\vec{i} + V\vec{j}$, divergence operator is defined as:

$$div(F) = \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} \tag{3.9}$$

In other words, divergence is a sum of partial derivatives of a differentiable vector function. Therefore, in our case, we have [25]:

$$\begin{aligned} div(\nabla f) &= \frac{\partial}{\partial x}(f_x) + \frac{\partial}{\partial y}(f_y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \Delta f \\ div(g(x,y,t)\nabla f) &= \frac{\partial}{\partial x}(g(x,y,t)f_x) + \frac{\partial}{\partial y}(g(x,y,t)f_y) = \nabla g \cdot \nabla f + g\Delta f \end{aligned} \tag{3.10}$$

Now, by simply using the finite differences introduced above, one way of discretising the divergence terms in 3.10 is using the central difference. First we apply the central difference and then the forward- and the backward differences for approximating the corresponding derivatives. The 'trick' here is to realise that $(f_x)(x+0.5,y)$ is actually the forward difference $D_x^+ f(x)$, while $(f_x)(x-0.5,y)$ is the backward difference $D_x^- f(x)$. Equations 3.11 , and 3.12 show the discretisations for $div(\nabla f)$, and $div(g(x,y,t)\nabla f)$, respectively. This is the same discretisation as in the famous paper by Perona and Malik [23].

$$\begin{aligned} \frac{\partial}{\partial x}(f_x)(x,y) + \frac{\partial}{\partial y}(f_y)(x,y) = &(f_x)(x+0.5,y) - (f_x)(x-0.5,y) \\ &+ (f_y)(x,y+0.5) - (f_y)(x,y-0.5) \\ =& f(x+1,y) - f(x,y) + f(x-1,y) - f(x,y) \\ &+ f(x,y+1) - f(x,y) + f(x,y-1) - f(x,y) \\ =& \nabla_E f + \nabla_W f + \nabla_S f + \nabla_N f \end{aligned} \tag{3.11}$$

Where $\nabla_{\{W,N,E,S\}} f$ denotes the difference in the directions given by W, N, E, S. As it was already mentioned, first we apply first order central difference on $f_x(x,y)$, and thus obtain $D_x^0 f_x(x,y) = (f_x)(x+0.5,y) - (f_y)(x-0.5,y)$.

$$\begin{aligned}\frac{\partial}{\partial x}(gf_x)(x,y) + \frac{\partial}{\partial y}(gf_y)(x,y) =& (gf_x)(x+0.5,y) - (gf_x)(x-0.5,y) \\ &+ (gf_y)(x,y+0.5) - (gf_y)(x,y-0.5) \\ =& g(x+0.5,y)(f(x+1,y) - f(x,y)) \\ &+ g(x-0.5,y)(f(x-1,y) - f(x,y)) \\ &+ g(x,y+0.5)(f(x,y+1) - f(x,y)) \\ &+ g(x,y-0.5)(f(x,y-1) - f(x,y)) \\ =& g_E \nabla_E f + g_W \nabla_W f + g_S \nabla_S f + g_N \nabla_N f \end{aligned} \tag{3.12}$$

Where $g_{\{W,N,E,S\}}$ denotes diffusivity in the directions given by W, N, E, S. As it can be observed from equation 3.12 , we need to approximate the diffusivity between the pixels. A simple '2-point' approximation would be the average between neighbouring pixels, for example $g(x+0.5,y) = [g(x+1,y) + g(x,y)]/2$. A more precise approximation, leading to better results, is a '6-point' approximation of Brox.
As it was already mentioned above, physical interpretation of the divergence is, in a sense, diffusion: the divergence operator introduces a 'connectivity' between the pixels. This simply means, as will be shown later on, that a solution at any position (i,j) will depend on the solution at neighbouring positions. Because of this kind of a dependency of the solution between the adjacent positions, variational correspondence methods are said to be 'global'. This kind of connectivity is problematic at image borders, where we do not have neighbour anymore. In order to deal with this problem, we use a scheme called eliminated boundary conditions [25], shown in Figure 3.4

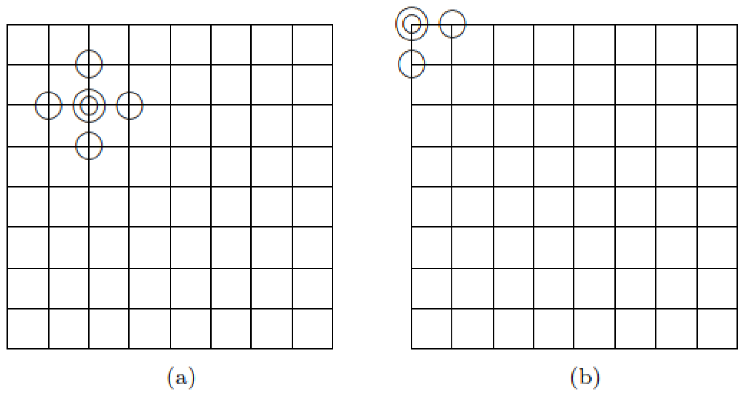


Figure 3.4: Double circle denotes the position of interest while single circles are the neighbouring positions W, N, E, S; (b) shows the eliminated boundary conditions.

3.2.2 Discretised diffusion equation

In order to solve Equation of Implicit Scheme weed to discretise the divergence operator. We start by marking the positions of the pixels of interest with (i, j) :

$$\frac{(u)_{i,j}^{n+1} - (u)_{i,j}^{n}}{\Delta t} = \operatorname{div}\left(g^n \nabla u^{n+1}\right)$$

After this we discretise the div operator, as given by equation 3.12 , and obtain the following:

$$\begin{aligned}(u)_{i,j}^{n+1} - (u)_{i,j}^{n} =& \Delta t g_N^n \left((u)_{i-1,j}^{n+1} - (u)_{i,j}^{n+1}\right)\\ &+ \Delta t g_S^n \left((u)_{i+1,j}^{n+1} - (u)_{i,j}^{n+1}\right)\\ &+ \Delta t g_W^n \left((u)_{i,j-1}^{n+1} - (u)_{i,j}^{n+1}\right)\\ &+ \Delta t g_E^n \left((u)_{i,j+1}^{n+1} - (u)_{i,j}^{n+1}\right)\end{aligned}$$

The only thing left to do, is arrange the terms:

$$\begin{aligned}(u)_{i,j}^{n+1}\left(1 + \Delta t\left(g_N^n + g_S^n + g_W^n + g_E^n\right)\right) = & (u)_{i,j}^{n}\\ &+ \Delta t g_N^n \left((u)_{i-1,j}^{n+1}\right)\\ &+ \Delta t h g_S^n \left((u)_{i+1,j}^{n+1}\right)\\ &+ \Delta t g_W^n \left((u)_{i,j-1}^{n+1}\right)\\ &+ \Delta t g_E^n \left((u)_{i,j+1}^{n+1}\right)\end{aligned} \tag{3.13}$$

3.2.3 Matrix format

While equation 3.13f shows equation to be solved for a pixel position (i, j), we can write the system equations in matrix/vector format, covering the whole image, as given in 3.14 . Now, the components of the vector are $I_{\mathcal{I}}$ where $\mathcal{I} \in \{1, \ldots, N\}$ and N is the number of pixels in image. We use $\mathcal{I}, \mathcal{J}$ also to mark the positions in the system matrix A given in 3.14. This is done in order to convey clearly the idea that the domains of the discretised images and the system matrices are different. If the domain of the discretised image is $\Omega_h : [1, m] \times [1, n]$ (discrete image with m columns and n rows), the system matrix A is defined on $[m \cdot n] \times [m \cdot n]$ (here - denotes multiplication). Now, we can write the Euler forward, semi-implicit formulation in a vector/matrix format as follows [36, 25]:

$$\frac{(\mathbf{u})^{n+1} - (\mathbf{u})^{n}}{\Delta t} = A\left((\mathbf{u})^n\right)\mathbf{u}^{n+1} \tag{3.14}$$

where $\mathbf{u} := (u)_{\mathcal{I}}$ with $\mathcal{I} = [1 \ldots N]$, and k refers to the channel in question (e.g. R, G or B). The system matrix $A\left((\mathbf{u})^n\right)$ is defined as follows: $A\left((\mathbf{u})^n\right) = \left[a_{\mathcal{I},\mathcal{J}}^n\right]$ where $g_{\mathcal{J}\sim\mathcal{I}}^n$ refers to the 'diffusion' weight between pixels $\mathcal{J}$ and $\mathcal{I}$ at time t.

In other words, these refer to the $g_{\{W,N,E,S\}}$ seen previously. equation 3.15 gives an example of how the system matrix A would look like for a 3×3 size image. Here C and N are block matrices that refer to the 'central' and the 'neighbouring' matrices, correspondingly.

$$A = \begin{bmatrix} C & N & 0 \\ N & C & N \\ 0 & N & C \end{bmatrix}$$

$$C = \begin{bmatrix} 1 + \sum_{\substack{\mathcal{J} \in N^-(\mathcal{I}) \\ \mathcal{J} \in N^+(\mathcal{I})}} \Delta t g^n_{\mathcal{J}\sim\mathcal{I}} & -\Delta t g^n_{\mathcal{J}\sim\mathcal{I}} & 0 \\ -\Delta t g^n_{\mathcal{J}\sim\mathcal{I}} & 1 + \sum_{\substack{\mathcal{J} \in N^-(\mathcal{I}) \\ \mathcal{J} \in N^+(\mathcal{I})}} \Delta t g^n_{\mathcal{J}\sim\mathcal{I}} & -\Delta t g^n_{\mathcal{J}\sim\mathcal{I}} \\ 0 & -\Delta t g^n_{\mathcal{J}\sim\mathcal{I}} & 1 + \sum_{\substack{\mathcal{J} \in N^-(\mathcal{I}) \\ \mathcal{J} \in N^+(\mathcal{I})}} \Delta t g^n_{\mathcal{J}\sim\mathcal{I}} \end{bmatrix}$$

$$N = \begin{bmatrix} -\Delta t g^n_{\mathcal{J}} \sim \mathcal{I} & 0 & 0 \\ 0 & -\Delta t g^n_{\mathcal{J}} \sim \mathcal{I} & 0 \\ 0 & 0 & -\Delta t g^n_{\mathcal{J}\sim I} \end{bmatrix} \tag{3.15}$$

From 3.14 we can see how the matrix A looks like for a 3×3 size image: it is a block tridiagonal square matrix, of size 9×9, that has non-zero components only on the main diagonal and on the diagonals adjacent to this. Therefore, unless the image is very small, it is infeasible to solve the system by inverting A directly. Instead, we search for a solution using iterative methods.

$$\left(\begin{array}{ccc|ccc|ccc} C & S & 0 & E & 0 & 0 & 0 & 0 & 0 \\ N & C & S & 0 & E & 0 & 0 & 0 & 0 \\ 0 & N & C & 0 & 0 & E & 0 & 0 & 0 \\ \hline W & 0 & 0 & C & S & 0 & E & 0 & 0 \\ 0 & W & 0 & N & C & S & 0 & E & 0 \\ 0 & 0 & W & 0 & X & C & 0 & 0 & E \\ \hline 0 & 0 & 0 & W & 0 & 0 & C & S & 0 \\ 0 & 0 & 0 & 0 & W & 0 & N & C & S \\ 0 & 0 & 0 & 0 & 0 & W & 0 & N & C \end{array}\right)$$

(a): Column wise traversing

$$\left(\begin{array}{ccc|ccc|ccc} C & E & 0 & S & 0 & 0 & 0 & 0 & 0 \\ W & C & E & 0 & S & 0 & 0 & 0 & 0 \\ 0 & W & C & 0 & 0 & S & 0 & 0 & 0 \\ \hline N & 0 & 0 & C & E & 0 & S & 0 & 0 \\ 0 & N & 0 & W & C & E & 0 & S & 0 \\ 0 & 0 & N & 0 & X & C & 0 & 0 & S \\ \hline 0 & 0 & 0 & N & 0 & 0 & C & E & 0 \\ 0 & 0 & 0 & 0 & N & 0 & W & C & E \\ 0 & 0 & 0 & 0 & 0 & N & 0 & W & C \end{array}\right)$$

(b): Row-wise traversing

3.2.4 AOS (Additive Operator Splitting)

With the vector/matrix format in place, we can now formulate the 'additive operator splitting' scheme by Weickert et al.[36]. In order to simplify the , we write A instead of $A\left((\mathbf{u})^n\right).Id$ refers to the identity matrix. Therefore, we have:

$$(\mathbf{u})^{n+1} = (\mathbf{u})^n + \Delta t A \mathbf{u}^{n+1}$$

From which $(\mathbf{u})^{n+1}$ can be solved as follows:

$$(\mathbf{u})^{n+1} = (Id - \Delta t A)^{-1} \mathbf{u}^n$$

Now, we 'decompose' A so that $A = \sum_{l=1}^{m} A_l$, which allows as to write the above equation as:

$$(\mathbf{u})^{n+1} = \left(\sum_{l=1}^{m} \frac{1}{m} Id - \Delta t \sum_{l=1}^{m} A_l \right)^{-1} \mathbf{u}^n$$

where m is the number of dimensions (in our case $m = 2$). Previous equation can be written, using only a single summation operator, as:

$$(\mathbf{u})^{n+1} = \left(\sum_{l=1}^{m} \frac{1}{m} \left(Id - \Delta t m A_l\right) \right)^{-1} \mathbf{u}^n \tag{3.16}$$

The idea of writing A as $A = \sum_{l=1}^{m} A_l$ is based on the traversing order of A as shown in Figure **??** . When constructing each A_l (here l can be though of referring to the traversing order based on the dimension), we only take into account the neighbours on the diagonals next to the main diagonal and discard the rest.
Physically this can be interpreted as diffusing separately along the vertical and horizontal directions.
When constructing the matrices A_l, one must be careful with the 'central' elements, equation 3.15, so that only the neighbours on the diagonals next to the main diagonal are taken into account. equation 3.16 has interesting 'form' in the sense that the 'system matrix' is decomposed.
The problem is that the decomposed system matrix is inside the $()^{-1}$ operator. Instead, we would like to construct the solution in parts as follows[25]:

$$(\mathbf{u})^{n+1} = \sum_{l=1}^{m} \left(\frac{1}{m} \left(Id - \Delta t m A_l\right) \right)^{-1} \mathbf{u}^n \tag{3.17}$$

The problem is that the right hand sides of Equations 3.16 and 3.17 are not equal, as can be easily verified. Therefore, we pose the question if there exists a simple variable x, when used to multiply the right hand side of 3.17, would make these equal:

$$(\sum_{l=1}^{m} \frac{1}{m} \underbrace{(Id - \Delta t m A_l)}_{B})^{-1} \mathbf{u}^n = x \sum_{l=1}^{m} (\frac{1}{m} \underbrace{(Id - \Delta t m A_l)}_{B})^{-1} \mathbf{u}^n \tag{3.18}$$

The above can be simplified into:

$$B^{-1} = xm^2 B^{-1}$$

And, thus we have:

$$x = \frac{1}{m^2}$$

Based on this, in order to use the 'additive operator splitting' scheme given by equation 3.17., we multiply the right hand side with $\frac{1}{m^2}$, and obtain the following:

$$(\mathbf{u})^{n+1} = \frac{1}{m^2} \sum_{l=1}^{m} \left(\frac{1}{m} (Id - \Delta t m A_l) \right)^{-1} \mathbf{u}^n$$

which is the same as:

$$(\mathbf{u})^{n+1} = \sum_{l=1}^{m} \left(mId - \Delta t m^2 A_l \right)^{-1} \mathbf{u}^n$$

As an example, if $l = 2(2\text{D})$, then we would have:

$$(\mathbf{u})^{n+1} = (2Id - 4\Delta t A_x)^{-1} \mathbf{u}^n + (2Id - 4\Delta t A_y)^{-1} \mathbf{u}^n$$

Introducing the notation: $V = (2Id - 4\Delta t A_x)^{-1} U^n$ and $W = (2Id - 4\Delta t A_y)^{-1} u^n$
The solution is simply:

$$U^{n+1} = V + W$$

We finally obtain the equation sets for V and W as follows:

$$(2Id - 4\Delta t A_x) V = U^n \text{ and } (2Id - 4\Delta t A_y) W = U^n.$$

Now, as it can be understood, the whole idea of this scheme is to bring the equations to a 'simpler' form, allowing us to use efficient block-wise solvers, such as TDMA (TriDiagonal Matrix Algorithm) see the Appendix.

3.3 Linear Versus Nonlinear Diffusion

For a useful comparison between the linear and nonlinear diffusion filtering methods and their impact on edge sharpness, Figure 3.5 shows the plots of the noisy 1-D edge profile (1-D signal) function and its filtered versions.
In Figure 3.5 the difference between a linear filter and a nonlinear filter can be observed.
Another example for a gray-scale image filtred by linear and nonlinear diffusion is given in Figure 3.6.
Figure 4.4 shows that The Gaussian noise is removed more effectively with the nonlinear filter compared to salt and pepper noise.
The salt and pepper noise becomes blurred but is not removed, this shows that

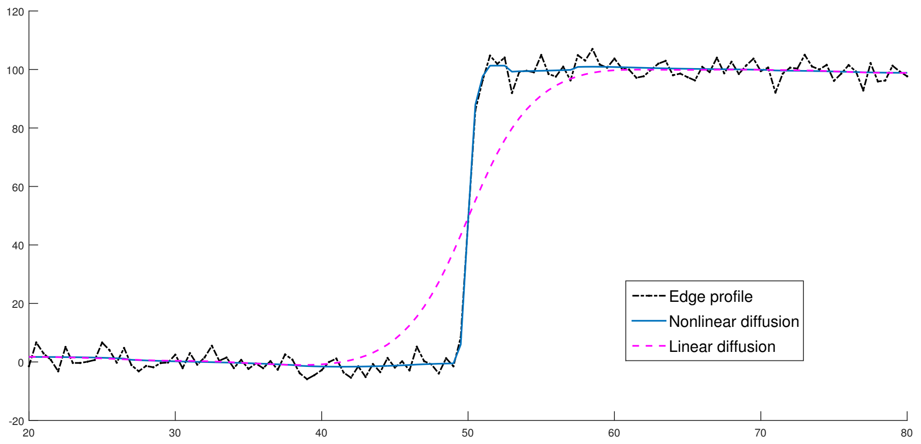


Figure 3.5: The 1-D edge profile $u(x)$ (dot-dash) and its filtered versions. Nonlinear diffusion filtering (solid) and linear diffusion filtering (dashed).

the non-linear filter is not suitable in the Salt and Pepper Images .

As seen in the figures 3.6 and 4.4 the linear diffusion filtering blurs the edge, whereas the nonlinear diffusion filtering preserves edge sharpness. The model of Malik and Perona had several serious, practical and theoretical difficulties.

- Assume that the signal is noisy, with the white noise for instance. Then large gradients $|\nabla u|$ are introduced by the noise. Moreover, ∇u is in theory unbounded. Thus, the conditional smoothing introduced by the model will not give good results, since all these noise edges will be kept.
- The second difficulty arose from the equation itself. The function g needs to be considered carefully to obtain the available theory. Indeed, in order to obtain both existence and uniqueness of the solutions, g must verify that $sg(s)$ is non-decreasing. In practice we will find out that if for some function g with $sg(s)$ non-increasing, very close pictures could produce divergent solutions and therefore different edges.

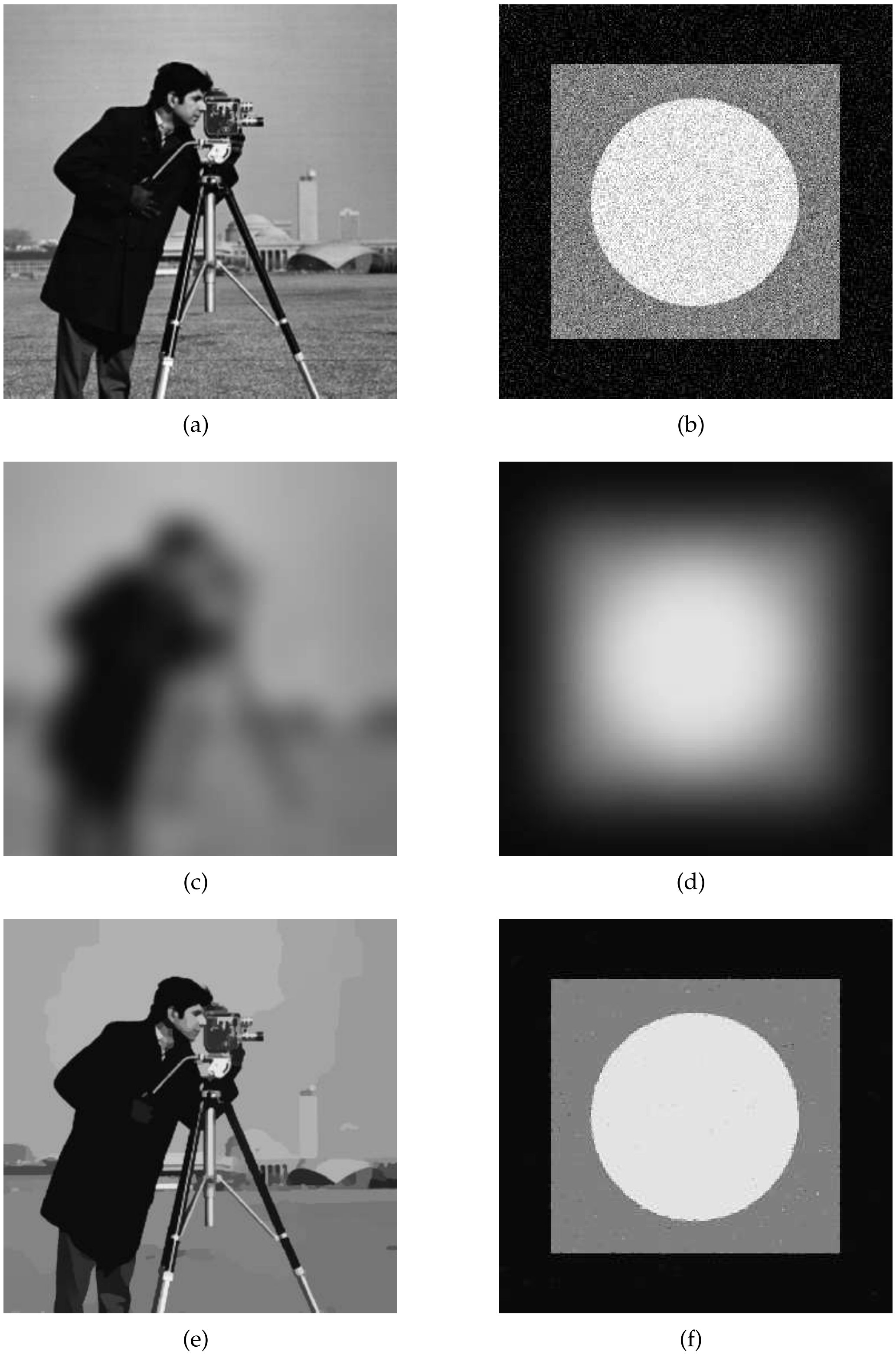
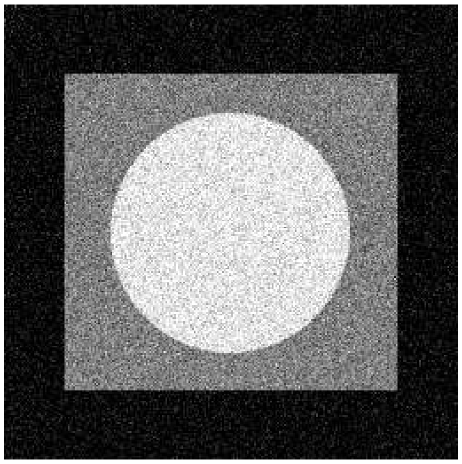
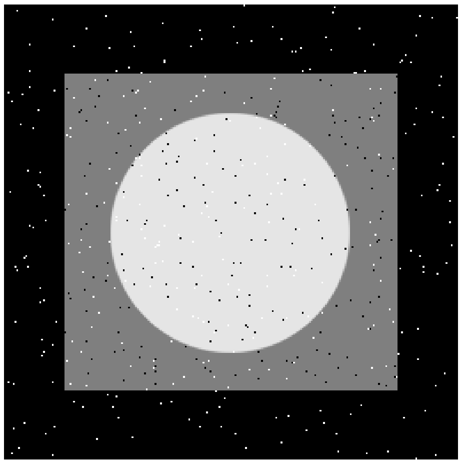
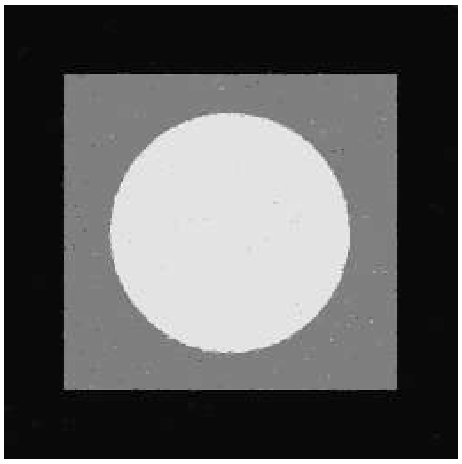
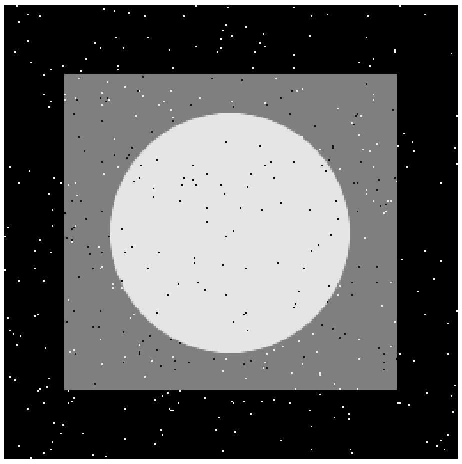


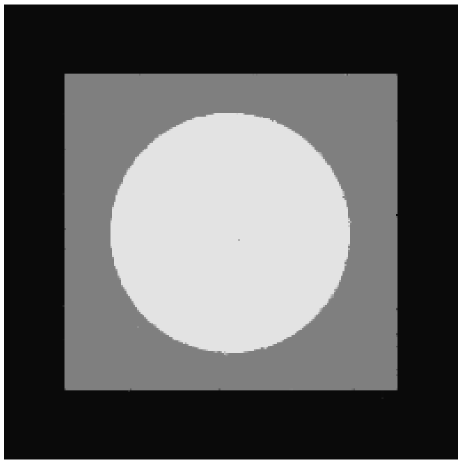
Figure 3.6: The original image and its linear and nonlinear diffusion filtered versions. a: Original Image b:Gaussian Noise Image c,d:Images Filtered by linear filter after 20 Iteration e,f: Filtered by nonlinear filter after 20 Iteration


(a)


(b)


(c)


(d)


(e)

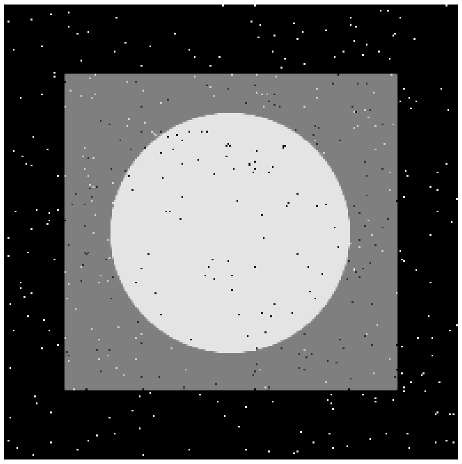

(f)

Figure 3.7: The original image and its nonlinear diffusion (PM) filtered versions.a: Gaussian Noise Image b: Salt and Pepper Image c,d:Images Filtered by nonlinear filter after 100 Iteration e,f: Images Filtered by nonlinear filter after 200 Iteration

CHAPTER 4

FAST DIFFERENCE SCHEMES FOR ANISOTROPIC BELTRAMI MODEL

The Beltrami flow [28, 31] is one of the most effective denoising algorithms in image processing. For gray-level images, we show that the Beltrami flow equation 1.1 can be arranged in a reaction-diffusion form. This reveals the edge-enhancing properties of the equation and suggests the application of additive operator splitting (AOS) methods [12, 13] for faster convergence. As we show with numerical simulations, the AOS method results in an unconditionally stable semi-implicit linearized difference scheme in 2D.

4.1 Splitting the Beltrami Operator

Compare the nonlinear diffusion of a gray-level image with Beltrami flow. Let U be the pixel value. The nonlinear diffusion is described by the following PDE[36, 13]

$$u_t = \nabla \cdot (g \nabla u) = \frac{\partial}{\partial x}(g u_x) + \frac{\partial}{\partial y}(g u_y) \tag{4.1}$$

where $g = \frac{1}{u_x^2 + u_y^2 + 1}$ is the Edge Indicator Function. equation 4.1 can be written in simplified form,

$$u_t = \frac{u_{xx}\left(-u_x^2 + u_y^2 + 1\right) - 4u_{xy}u_x u_y + u_{yy}\left(u_x^2 - u_y^2 + 1\right)}{\left(u_x^2 + u_y^2 + 1\right)^2}$$

Now consider the Beltrami flow:

$$\begin{aligned} u_t &= \frac{u_{xx}\left(u_y^2+1\right) - 2u_{xy}u_xu_y + u_{yy}\left(u_x^2+1\right)}{\left(u_x^2+u_y^2+1\right)^2} \\ &= \frac{\partial}{\partial x}(gu_x) + \frac{\partial}{\partial y}(gu_y) + \frac{u_xu_{xx}+u_yu_{xy}}{\left(u_x^2+u_y^2+1\right)^2}\cdot u_x + \frac{u_xu_{xy}+u_yu_{yy}}{\left(u_x^2+u_y^2+1\right)^2}\cdot u_y \\ &= \frac{\partial}{\partial x}(gu_x) + \frac{\partial}{\partial y}(gu_y) - \frac{u_x}{2g_x} - \frac{u_y}{2g_y} \\ &= \frac{1}{2}\frac{\partial}{\partial x}(gu_x) + \frac{1}{2}\frac{\partial}{\partial y}(gu_y) + \frac{g}{2}\left(u_{xx}+u_{yy}\right) \end{aligned} \tag{4.2}$$

The Beltrami equation may be reduced to a similar reaction-diffusion form, namely

$$U_t = \frac{1}{2}g\nabla^2U + \frac{1}{2}\nabla\left(g\cdot\nabla U\right) \tag{4.3}$$

In this form, the Beltrami flow equation is not a ”pure” diffusion equation. It has both an (parabolic) edge preserving and an (hyperbolic) edge-sharpening terms. In addition, the reaction-diffusion form of 4.3 hides the mixed derivative U_{xy}. This makes it possible to apply Additive Operator Split, to be used in the implicit numerical scheme for individual rows and columns of pixels:

$$u_t = \left(A_x + A_y\right)u \tag{4.4}$$

where $\cdot A_x$ and A_y are he following differential operators:

$$\begin{cases} A_x = \frac{\partial}{\partial x}\left(\frac{g}{2}\frac{\partial}{\partial x}\right) + \frac{g}{2}\frac{\partial^2}{\partial x^2} \\ A_y = \frac{\partial}{\partial y}\left(\frac{g}{2}\frac{\partial}{\partial y}\right) + \frac{g}{2}\frac{\partial^2}{\partial y^2} \end{cases} \tag{4.5}$$

4.2 Implicit Scheme for Gray level Image (2D)

BY Applying the backward difference formula to the above form we get [19, 16],

$$\frac{u^{n+1}-u^n}{\Delta t} = \left(A_x + A_y\right)u^{n+1}$$

The superscript n is related to the present and $n+1$ to the next time step. The subscripts i, j index the discrete pixel location; $u_{i,j}^n$ are known values, and $u_{i,j}^{n+1}$ are to be found. Using u^{n+1} on the right side of equation 4.3 makes the integration scheme implicit and unconditionally stable, namely

$$\left[Id - \Delta t\left(A_x + A_y\right)\right]u^{n+1} = u^n \tag{4.6}$$

Where Id is the identity matrix. Before proceeding in time, we calculate the values of the edge indicator function g, using the known values of u^n. Thus, the scheme is only semi-implicit. Although g depends on the gradient of U, we treat it like a given function of (x, y), making the governing PDE ”quasi-linear”.

Note that 4.6 includes a large bandwidth matrix, because all equations, related to new pixel values u^{n+1} are coupled. Our aim is to decouple the set 4.6 so that each row and each column of pixels can be handled separately. For this, we rearrange the equations into the following form:

$$u^{n+1} = \left[Id - \Delta t\left(A_x + A_y\right)\right]^{-1} u^n \tag{4.7}$$

Of course, we do not intend to invert the matrix to solve the linear set. This is only a symbolic form used for further derivation. For a small value of Δt, the matrix in the brackets on the right side of 4.7 is close to the identity Id. Thus, its inverse can be expanded into the Taylor series in the proximity of Id: $\left[Id - \Delta t\left(A_x + A_y\right)\right]^{-1} \approx Id + \Delta t\left(A_x + A_y\right)$, where the linear term is retained and the high order terms are neglected. Introducing this form into 4.7, we get,

$$2u^{n+1} = \left(Id + 2\Delta t A_x\right) u_n + \left(Id + 2\Delta t A_y\right) u^n$$

Introducing the notations $V = \left(Id + 2\Delta t A_x\right) u^n$ and $W = \left(Id + 2\Delta t A_y\right) u^n$ the solution is simply

$$u^{n+1} = \frac{V+W}{2}$$

In order to get an implicit scheme, we apply the differential matrix operators A_x and A_y to u^{n+1} (and not to u^n), namely

$$\left(Id + 2\Delta t A_x\right)^{-1} V = u^n \quad \left(Id + 2\Delta t A_y\right)^{-1} W = u^n$$

Following the procedure of expanding the matrix inverses into Taylor series and applying the linearization for small Δt, we finally obtain the equation sets for V and W as follows:

$$\left(Id - 2\Delta t A_x\right) V = u^n \quad \left(Id - 2\Delta t A_y\right) W = u^n \tag{4.8}$$

These equations can be solved with either the Dirichlet or Neumann boundary conditions.

4.3 Finite Difference Equation

The differential operators A_x and A_y in. equation 4.5 are similar, and therefore we derive here the difference equation for a single row of pixels. Equation for the column of pixels is identical. Consider a row with $N+1$ pixels enumerated from 0 to N, Fig. 1 . Figure 1: Finite Difference Scheme

$$2A_x V = \frac{\partial}{\partial x}\left(\frac{1}{g}\frac{\partial V}{\partial x}\right) + \frac{1}{g}\frac{\partial^2 V}{\partial x^2} \tag{4.9}$$

In the difference equations 4.10,4.9 the omitted error terms are of order $O\left(\Delta x^2\right)$

$$\left(\frac{\partial V}{\partial x}\right)_{i+1/2} = \frac{V_{i+1} - V_i}{\Delta x}; \quad \left(\frac{\partial V}{\partial x}\right)_{i-1/2} = \frac{V_i - V_{i-1}}{\Delta x} \tag{4.10}$$

$$\begin{aligned} 2\left(A_x V\right)_i &= \frac{g_{i+1/2}\left(V_{i+1}-V_i\right)-g_{i-1/2}\left(V_i-V_{i-1}\right)}{\Delta_x^2}+g_i\frac{V_{i+1}-2V_i+V_{i-1}}{\Delta x^2} \\ 2\left(A_x V\right)_i &= \frac{g_{i-1/2}+g_i}{\Delta x^2}V_{i-1}-\frac{g_{i-1/2}+2g_i+g_{i+1/2}}{\Delta x^2}V_i+\frac{g_i+g_{i+1/2}}{\Delta x^2}V_{i+1} \end{aligned} \tag{4.11}$$

To avoid establishing the values of the edge indicator function g at the non-nodal points $i-1/2$ and $i+1/2$, we average the values of two neighbor nodes:

$$g_{i+1/2}=\frac{g_i+g_{i+1}}{2}+O\left(\Delta x^2\right) \quad g_{i-1/2}=\frac{g_{i-1}+g_i}{2}+O\left(\Delta x^2\right) \tag{4.12}$$

Introduce equation 4.11 into 4.9:

$$2\left(A_x V\right)_i=\frac{g_{i-1}+3g_i}{2\Delta x^2}V_{i-1}-\frac{g_{i-1}+6g_i+g_{i+1}}{2\Delta x^2}V_i+\frac{3g_i+g_{i+1}}{2\Delta x^2}V_{i+1} \tag{4.13}$$

It may seem that after introduction of equation 4.12 into 4.9, the overall error becomes huge, of order $O(1)$, due to Δx^2 in the denominator. However, due to a special symmetry of equation 4.11, the total error is still of order $O\left(\Delta x^2\right)$.
As we see, the erors are of the same order, and this justifies the validity of substitution (26). It follows from Equations. (4.8 and 4.12)

$$-\Delta t\frac{g_{i-1}+3g_i}{2\Delta x^2}V_{i-1}+\left(1+\Delta t\frac{g_{i-1}+6g_i+g_{i+1}}{2\Delta x^2}\right)V_i-\Delta t\frac{3g_i+g_{i+1}}{2\Delta x^2}V_{i+1}=U_i^n$$

Introduce the following notations:

$$\alpha_x=\frac{\Delta t}{2\Delta x^2} \quad \alpha_y=\frac{\Delta t}{2\Delta_y^2}$$

The finite difference equation comes to:

$$\begin{aligned} &-\alpha_x\left(g_{i-1}+3g_i\right)V_{i-1}+\left[1+\alpha_x\left(g_{i-1}+6g_i+g_{i+1}\right)\right]V_i \\ &-\alpha_x\left(3g_i+g_{i+1}\right)V_{i+1}=U_i^n \end{aligned} \tag{4.14}$$

equation 4.14 is a linear set with the three-diagonal matrix. The similar equation holds for W in y dimension, where $j=0,1,\ldots M$.

$$\begin{aligned} &-\alpha_y\left(g_{j-1}+3g_j\right)\quad W_{j-1}+\left[1+\alpha_y\left(g_{j-1}+6g_j+g_{j+1}\right)\right]W_j \\ &-\alpha_y\left(3g_j+g_{j+1}\right)W_{j+1}=U_i^n \end{aligned}$$

For the first and the last nodes, either the Dirichlet, or the Neumann boundary conditions hold.
- In case of Neumann BC we assume that the normal derivatives V_n and W_n vanish along the rectangular contour of the computational box, i.e. we accept the mirror boundary conditions:

$$V_{-1}=V_1 \quad \text{and} \quad V_{N+1}=V_N$$

Establish the normal derivatives of the edge indicator function along the vertical and horizontal boundary lines:

$$\begin{cases} \frac{\partial g}{\partial x} = -2\frac{U_xU_{xx}+U_yU_{xy}}{\left(1+U_x^2+U_y^2\right)^2} \\ \frac{\partial g}{\partial y} = -2\frac{U_xU_{xy}+U_yU_{yy}}{\left(1+U_x^2+U_y^2\right)^2} \end{cases}$$

Along the vertical boundary $x =$ constant, $U_x = 0$ and $U_{xy} = 0$. Along the horizontal boundary $y =$ constant, $U_y = 0$ and $U_{xy} = 0$. Thus, in both cases $h_n = 0$, and the mirror boundary condition hold not only for the pixel value, but also for the edge indicator function. The ghost values become:

$$g_{-1} = g_1 \quad \text{and} \quad g_{N+1} = g_{N-1}$$

This affects the first and the last equations of Set 4.14:

$$[1 + \alpha_x (6g_0 + 2g_1)] V_0 - \alpha_x (6g_0 + 2g_1) V_1 \quad = U_1^n$$

$$-\alpha_x (2g_{N-1} + 6g_N) V_{N-1} + [1 + \alpha_x (2g_{N-1} + 6g_N)] V_N = U_N^n$$

4.4 Simulation Results for Gray Level Images

In Figure 4.1, we provide the results of our algorithm applied with different numbers of iterations on a grayscale image taken from .
We can see that the Beltrame filter works well with noise and edges. Where he worked to blur the noise and preserve the edges, and this is after 50 iterations or after 100 iterations. Function g preserves the edges and shape of the image by stopping the non-linear filter at the edges.
In Figure 4.2 we have an image with Salt an Pepper noise and its filtered by non-linear filter, we can remark that Perona-Malik filter effective more than Beltrami with Salt an Pepper noise .

4.5 Beltrami Smoothing for Color Images

The Beltrami flow for color images is governed by the following set of PDE [16]:

$$\frac{\partial I_i}{\partial t} = \frac{\frac{\partial P_i}{\partial x} + \frac{\partial Q_i}{\partial y}}{g} - \frac{\frac{\partial g}{\partial x}P_i + \frac{\partial g}{\partial y}Q_i}{2g^2}$$

where $i = 1, 2, 3$ is the number of color (red, green, blue), I_i is the corresponding pixel value and P, Q are defined by:

$$P_i = g_{22}\frac{\partial I_i}{\partial x} - g_{12}\frac{\partial I_i}{\partial y} \quad Q_i = -g_{12}\frac{\partial I_i}{\partial x} + g_{11}\frac{\partial I_i}{\partial y}$$

g_{11}, g_{12} and g_{22} are components of a symmetric matrix (tensor) G of dimension 2×2, and g is: its discriminant:

$$G = \begin{bmatrix} g_{11} & g_{12} \\ g_{12} & g_{22} \end{bmatrix}$$

$g_{11} = 1 + \sum_{j=1}^{3} \left(\frac{\partial I_j}{\partial x}\right)^2 \quad g_{22} = 1 + \sum_{j=1}^{3} \left(\frac{\partial I_j}{\partial y}\right)^2 \; g_{12} = \sum_{j=1}^{3} \frac{\partial I_j}{\partial x}\frac{\partial I_j}{\partial y}$ and

$$g = \det G = g_{11}g_{22} - g_{12}^2$$

4.6 Implicit Scheme for color Image (3D)

The classical nonlinear diffusion equation can be rewritten as the following partial differential equation [16, 24]:

$$u_t = \nabla \cdot (h \nabla u) = \frac{\partial h u_x}{\partial x} + \frac{\partial h u_y}{\partial y} + \frac{\partial h u_z}{\partial z},$$

with $h(x,y,z) = (1 + {u_x}^2 + {u_y}^2 + {u_z}^2)^{-1}$. The Beltrami Equation may be reduced to a similar advection-diffusion form, namely,

$$u_t = \nabla \cdot \left(h \frac{\nabla u}{2}\right) + h \frac{\nabla^2 u}{2} = \frac{1}{2} \nabla h \cdot \nabla u + h \nabla^2 u. \tag{4.15}$$

In this form, the Beltrami flow equation is not a ”pure” diffusion equation. It has both a parabolic edge-preserving and a hyperbolic edge-sharpening term.

Another modification is to replace $\frac{\partial h}{\partial x}\frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\left(h\frac{\partial u}{\partial x}\right) - h\frac{\partial^2 u}{\partial x^2}$ in 4.15 . This enables replacement of the discretization of $\frac{\partial h}{\partial x}\frac{\partial u}{\partial x}$ by a second order derivative term. Therefore, the resulting linear system will be a tridiagonal matrix, thus more stable than a system with no value on the diagonal. In addition, the advection-diffusion form of 4.15 hides the mixed derivatives, thereby making it conducive to the AOS approach . In other words, the equation can be rearranged into the form $u_t = (\mathbf{A}_x + \mathbf{A}_y + \mathbf{A}_z)\, u$, where $\mathbf{A}_x$, $\mathbf{A}_y$, and $\mathbf{A}_z$ are the following differential operators:

$$\begin{cases} \mathbf{A}_x = \frac{\partial}{\partial x}\left(\frac{h}{2}\frac{\partial}{\partial x}\right) + \frac{h}{2}\frac{\partial^2}{\partial x^2} \\ \mathbf{A}_y = \frac{\partial}{\partial y}\left(\frac{h}{2}\frac{\partial}{\partial y}\right) + \frac{h}{2}\frac{\partial^2}{\partial y^2} \\ \mathbf{A}_z = \frac{\partial}{\partial z}\left(\frac{h}{2}\frac{\partial}{\partial z}\right) + \frac{h}{2}\frac{\partial^2}{\partial z^2} \end{cases}$$

Note that, although g depends on the gradient of u, we treat it like a given function of (x, y), making the governing PDE ”quasi-linear.”

Applying the backward difference formula to last equations, we get

$$\frac{u^{n+1} - u^n}{\Delta t} = (\mathbf{A}_x + \mathbf{A}_y + \mathbf{A}_z)\, u^{n+1} \tag{4.16}$$

Where the superscript n is related to the present and $n+1$ to the next time step. Using u^{n+1} on the right side of 4.16 makes the integration scheme implicit and unconditionally stable, namely,

$$[\mathbf{u} - \Delta t\,(\mathbf{A}_x + \mathbf{A}_y + \mathbf{A}_z)]\,u^{n+1} = u^n, \tag{4.17}$$

where I is the identity matrix. Before proceeding in time, we calculate the values of the edge indicator function g, using the known values of u^n. Thus, the scheme is only semi-implicit. It should be noted that this technique is often referred to as the ”lagged diffusivity fixed point scheme,” and further analysis of convergence to the solution of original equation is beyond the scope of the present study. Note that 4.17 includes a large bandwidth matrix because all equations related to new pixel values u^{n+1} are coupled. Our aim is to decouple the set in 4.17 so that each row and each column of pixels can be handled separately. For this, we rearrange the equations into the following form:

$$u^{n+1} = [\mathbf{u} - \Delta t\,(\mathbf{A}_x + \mathbf{A}_y + \mathbf{A}_z)]^{-1}\,u^n, \tag{4.18}$$

Of course, we do not intend to invert the matrix to solve the linear set. This is only a symbolic form used for further derivation. For a small value of Δt, the matrix in the brackets on the right side of 4.18 is close to the identity I. Thus, its inverse can be expanded into the Taylor series in the proximity of $\mathbf{u}$:

$$[\mathbf{u} - \Delta t\,(\mathbf{A}_x + \mathbf{A}_y + \mathbf{A}_z)]^{-1} \simeq \mathbf{u} + k\,(\mathbf{A}_x + \mathbf{A}_y + \mathbf{A}_z)$$

Where the linear term is retained and the high order terms are neglected. Introducing this form into 4.18, we get

$$3u^{n+1} = (\mathbf{u} + 3\Delta t\mathbf{A}_x) + (\mathbf{u} + 3\Delta t\mathbf{A}_y) + (\mathbf{u} + 3\Delta t\mathbf{A}_z)$$

the solution is simply:

$$u^{n+1} = \frac{\mathbf{B}_x + \mathbf{B}_y + \mathbf{B}_z}{3}$$

where:

$$\mathbf{B}_x = (\mathbf{u} + 3\Delta t\mathbf{A}_x)u^n$$
$$\mathbf{B}_y = (\mathbf{u} + 3\Delta t\mathbf{A}_y)u^n$$

and

$$\mathbf{B}_z = (\mathbf{u} + 3\Delta t\mathbf{A}_z)u^n$$

In order to get an implicit scheme, we apply the differential matrix operators $\mathbf{A}_x$, $\mathbf{A}_y$, and $\mathbf{A}_z$ to u^{n+1} (and not to u^n), namely,

$$u^n = (\mathbf{u} + 3\Delta t\mathbf{A}_x)^{-1}\mathbf{B}_x$$

$$u^n = (\mathbf{u} + 3\Delta t\mathbf{A}_y)^{-1}\mathbf{B}_y$$

$$u^n = (\mathbf{u} + 3\Delta t\mathbf{A}_z)^{-1}\mathbf{B}_z$$

Following the procedure of expanding the matrix inverses into Taylor series and applying the linearization for small Δt, we finally obtain the equation sets for $\mathbf{B}_x$, $\mathbf{B}_y$, and $\mathbf{B}_z$ as follows:

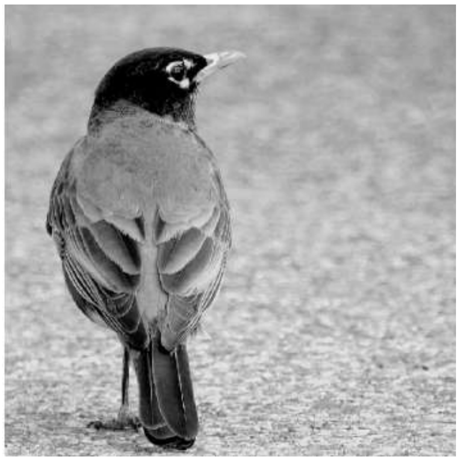
$$u^n = (\mathbf{u} - 3\Delta t\mathbf{A}_x)\mathbf{B}_x$$
$$u^n = (\mathbf{u} - 3\Delta t\mathbf{A}_y)\mathbf{B}_y$$
$$u^n = (\mathbf{u} - 3\Delta t\mathbf{A}_z)\mathbf{B}_z$$

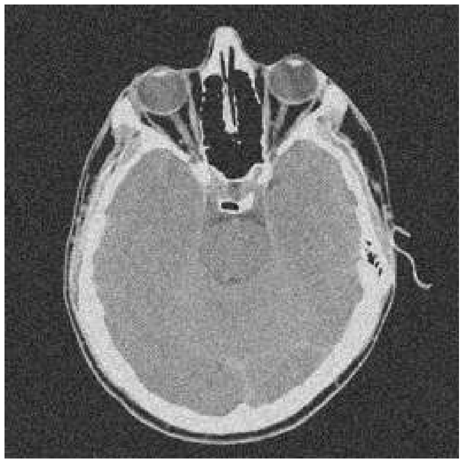
The equations can be solved with either the Dirichlet or Neumann boundary conditions.

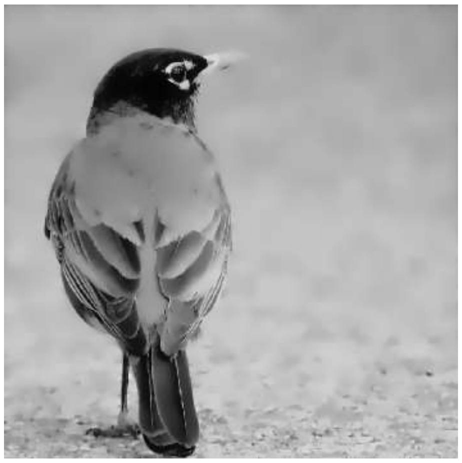
4.7 Simulation Results for Color Images

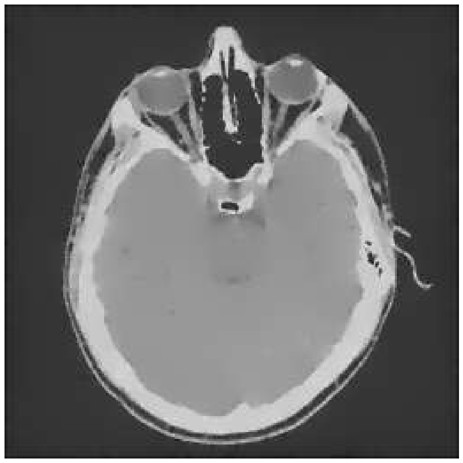
The goal of this numerical experiment is to show that the weakly coupled Beltrami smoothing operator may be replaced by its decoupled approximation, without essential loss of accuracy and with a great· saving of the computational time.

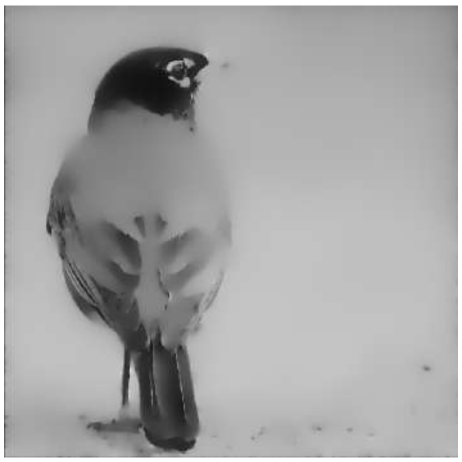
Figure. 4.3 presents the simulation results for implicit scheme with different values of iterations 20,50,200.
The first row includes the original image (left picture) and the smoothing simulation results for the Beltrami filtering.
As we see, for iteration of up to 200, the flow is stimulated with a reasonable accuracy [16, 24].


(a)


(b)


(c)


(d)


(e)

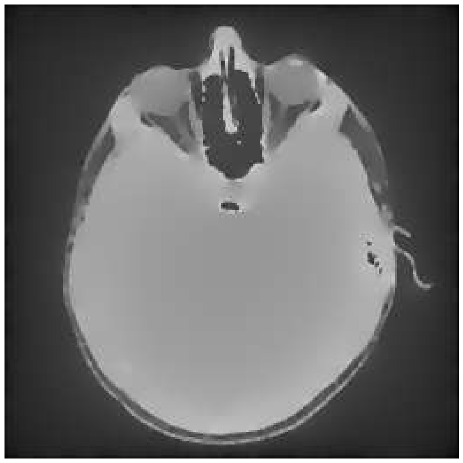
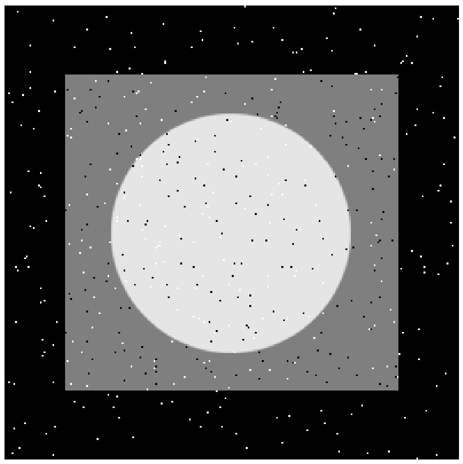
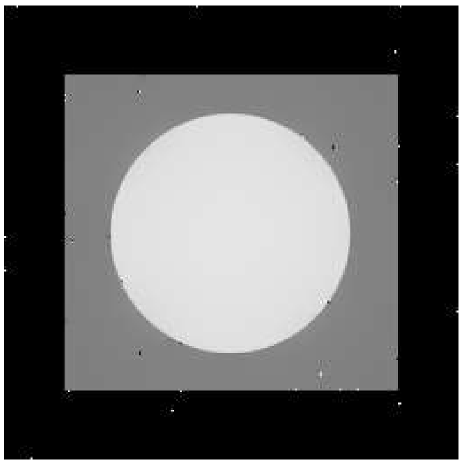

(f)

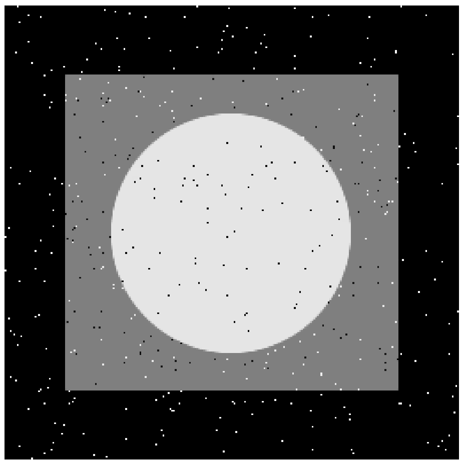
Figure 4.1: Edge Enhancement of Bird and Medical Image.a,b: Original Image c,d:Images Filtered by nonlinear filter (Beltrami) after 50 Iteration e,f: Images Filtered by nonlinear filter (Beltrami) after 200 Iteration



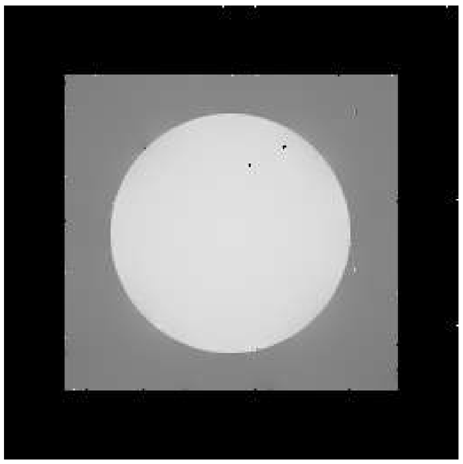
(a)



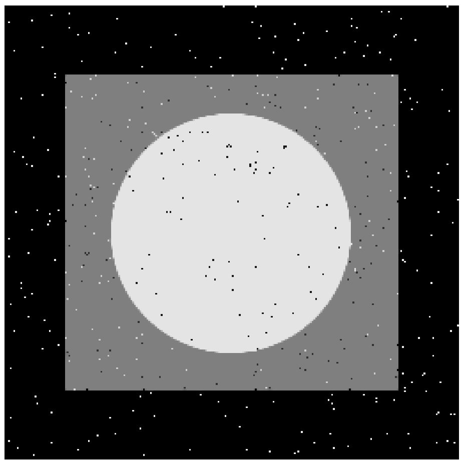
(b)



(c)

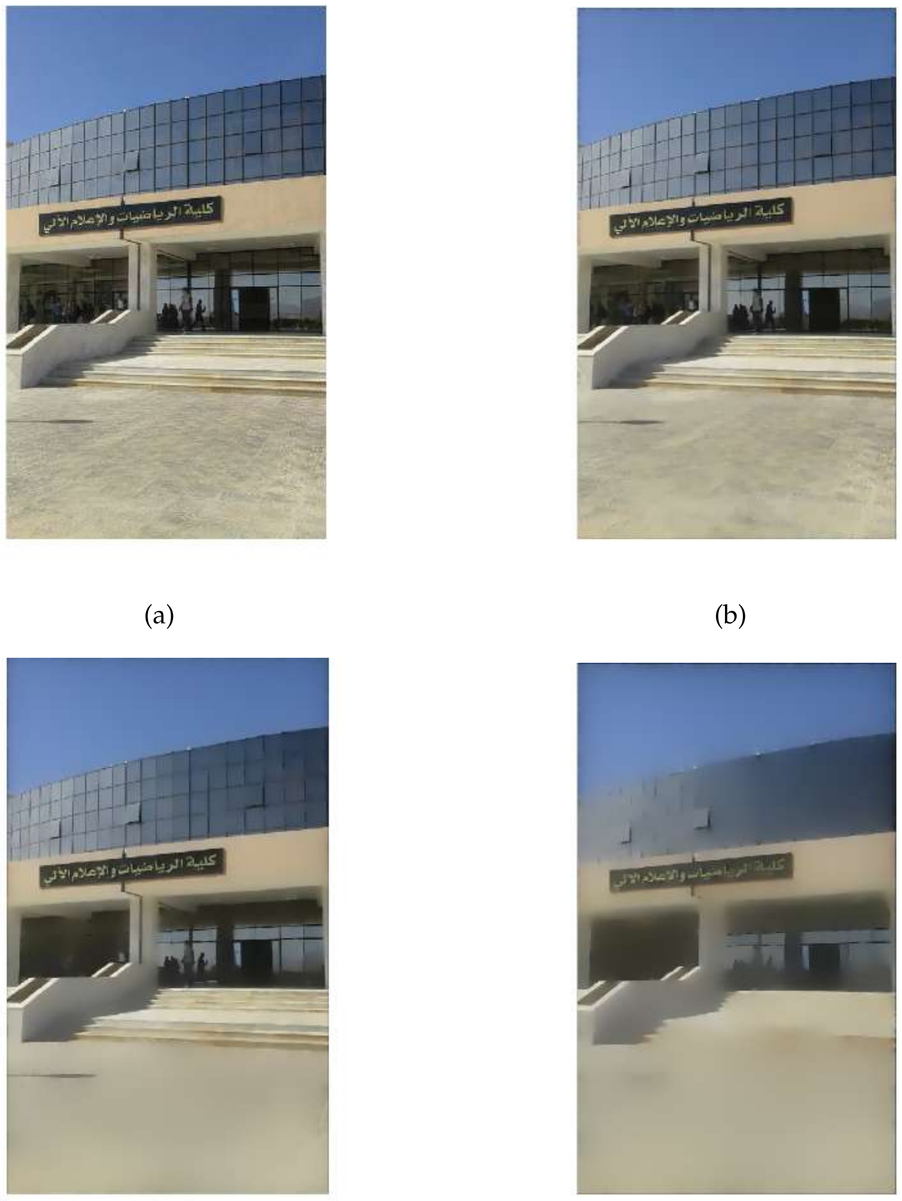


(d)



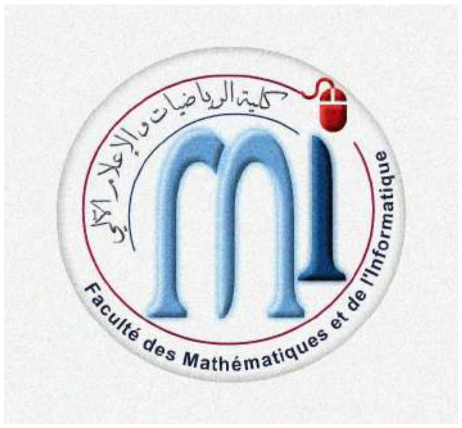
(e)

Figure 4.2: The original image and its nonlinear diffusion Perona-Malik and Beltrami filtered versions. **a:** salt and pepper Image **b:** Image filtered by Beltrami Flow after 100 iteration **c:** Images Filtered by nonlinear filter after 200 Iteration **d:** Image filtered by Beltrami Flow after 100 iteration **e:** Images Filtered by nonlinear filter after 200 Iteration

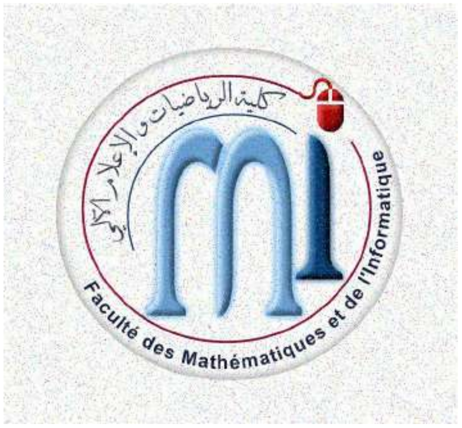


(a) (b) (c) (d)

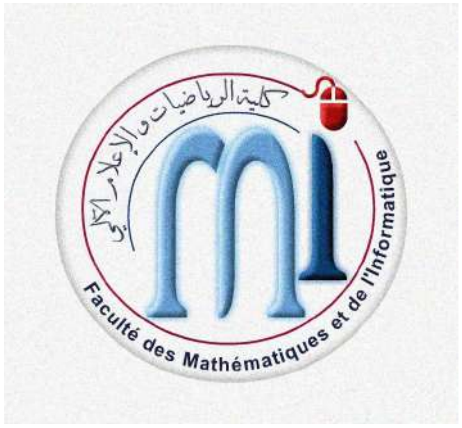
Figure 4.3: Beltrami Smoothing of Color Images. a: Original Image b,c and d: Images filtred by Beltrami filter after (20,50,200) iteration


كلية الرياضيات و الإعلام الآلي
Faculté des Mathématiques et de l'Informatique

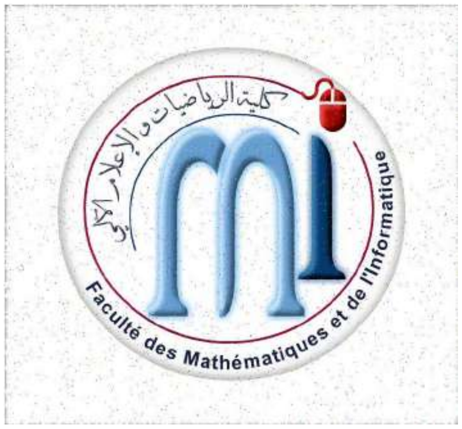
(a)


كلية الرياضيات و الإعلام الآلي
Faculté des Mathématiques et de l'Informatique

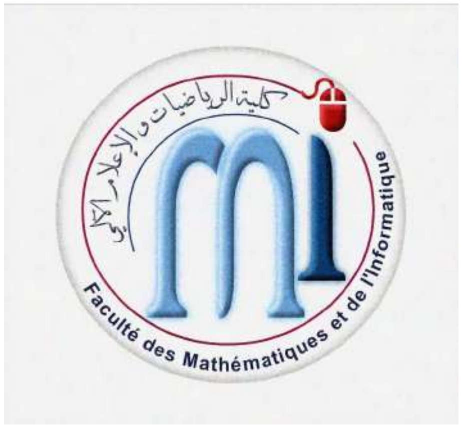
(b)


كلية الرياضيات و الإعلام الآلي
Faculté des Mathématiques et de l'Informatique

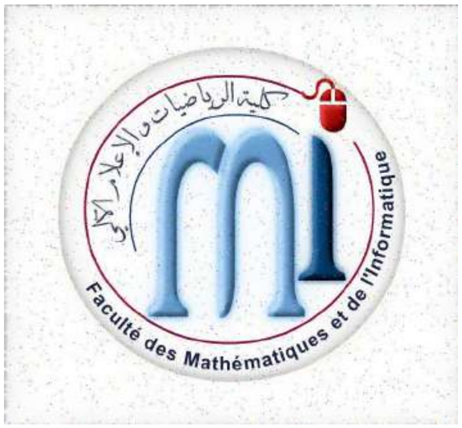
(c)


كلية الرياضيات و الإعلام الآلي
Faculté des Mathématiques et de l'Informatique

(d)


كلية الرياضيات و الإعلام الآلي
Faculté des Mathématiques et de l'Informatique

(e)


كلية الرياضيات و الإعلام الآلي
Faculté des Mathématiques et de l'Informatique

(f)

Figure 4.4: The original image and its Beltrami filtered versions.a: Gaussian Noise Image b: Salt and Pepper Image c,d:Images Filtered after 20 Iteration e,f: Images Filtered after 100 Iteration

CHAPTER 5

APPENDIX

5.1 Stability Analysis of Finite Difference schemes for 1-D Heat equation

5.1.1 Stability analysis of explicit scheme

Approximate the numerical error. Using the explicit method the discretized form of equation 2.1 is:

$$U_j^{n+1} = U_j^n + \lambda \left(U_{j+1}^n - 2U_j^n + U_{j-1}^n \right) \tag{5.1}$$

Define the error in the numerical approximation as

$$\epsilon_j^n = U_j^n - u_j^n \tag{5.2}$$

where u_j^n is the exact solution. Both the numerical solution U_j^n and the exact solution u_j^n satisfy equation 5.1, therefore, the error ϵ_j^n also follows the discretized ODE equation 5.1

$$\epsilon_j^{n+1} = \epsilon_j^n + \lambda \left(\epsilon_{j+1}^n - 2\epsilon_j^n + \epsilon_{j-1}^n \right) \tag{5.3}$$

The spatial variation of error may be expanded in a finite Fourier series, in the interval L, as

$$\epsilon(x) = \sum_{m=1}^{M} A_m e^{ik\Delta x} \tag{5.4}$$

where $k\Delta t = \frac{\pi m}{L}$, $m = 1, 2, \ldots, M$ and $M = L/\Delta x, i = \sqrt{-1}$. $e^{ik\Delta x}$ is the complex exponential. A_m is a function of time. Since the error tends to grow or decay exponentially with time, it is reasonable to assume that the amplitude varies exponentially with time; hence

$$\epsilon(x,t) = \epsilon_j^n = \sum_{m=1}^{M} e^t e^{ik\Delta x} \tag{5.5}$$

Since the difference equation for error is linear, it is enough to consider the growth of error of a typical term:

$$\epsilon_m(x,t) = \epsilon_j^n = e^t e^{ik\Delta x} \tag{5.6}$$

The goal is to show that the error incurred by a particular numerical scheme doesn't grow in the evolving steps of time. Define the amplification factor

$$A \equiv \frac{\epsilon_j^{n+1}}{\epsilon_j^n}$$

Then, the necessary condition for stability is:

$$|A| \leq 1 \text{ or } [-1 \leq A \leq 1] \tag{5.7}$$

To find out how the error varies in steps of time, substitute equation 5.6 into each term of equation 5.3, as shown below

$$\begin{array}{rcl} \epsilon_j^n & = & e^t e^{ik\Delta x} \\ \epsilon_j^{n+1} & = & e^{(t+k)} e^{ik\Delta x} \\ \epsilon_{i+1}^n & = & e^t e^{ik(x+\Delta x)} \\ \epsilon_{j-1}^n & = & e^t e^{ik(x-\Delta x)} \end{array} \tag{5.8}$$

By eqs. 5.8 in equation 5.3, and Divide by $e^{at}e^{ik\Delta x}$ to yield

$$e^k = 1 + \lambda\left(e^{ik\Delta x} + e^{-ik\Delta x} - 2\right) \tag{5.9}$$

Using the identities $\sin\left(\frac{k\Delta x}{2}\right) = \frac{e^{\frac{ik}{2}} - e^{-\frac{ik}{\Delta}x}2}{2i}$, and $\sin^2\left(\frac{k\Delta x}{2}\right) = -\frac{\left[e^{ik\Delta x} + e^{-ik\Delta x} - 2\right]}{4}$ in equation 5.9

$$e^{a\Delta t} = 1 - 4\lambda \sin^2\left(\frac{k\Delta x}{2}\right) \tag{5.10}$$

then we have:

$$A = \frac{\epsilon_j^{n+1}}{\epsilon_j^n} = \frac{e^{(t+\Delta t)}e^{ik\Delta x}}{e^t e^{ik\Delta x}} = e^{\Delta t} \tag{5.11}$$

then:

$$|A| = \left|1 - 4\lambda \sin^2(k\Delta x/2)\right| \leq 1 \left\{ \begin{array}{c} \left(1 - 4\lambda \sin^2\left(\frac{k\Delta x}{2}\right)\right) \geq -1 \\ \left(1 - 4\lambda \sin^2\left(\frac{k\Delta x}{2}\right)\right) \leq 1 \end{array} \right. \tag{5.12}$$

last equation yields:

$$4\lambda \sin^2(k\Delta x/2) \leq 2 \tag{5.13}$$

that means:

$$\lambda = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{5.14}$$

equation 5.12 yields

$$4\lambda \sin^2 (k\Delta x/2) \geq 0$$

Since $\sin^2 (k\Delta x/2)$ range is $[0, 1]$ the last equation yields

$$\lambda \geq 0$$

which always holds. Hence, combining the last two equations: $0 \leq \lambda \leq \frac{1}{2}$ equation 5.14 gives the stability requirement for the explicit scheme as applied to the one-dimensional heat equation. We say that the method is conditionally stable; for a given Δx, the allowed value of Δt must be small enough to satisfy equation5.14 or

$$\Delta t \leq \frac{\Delta x^2}{2}$$

5.1.2 Stability analysis of implicit scheme

We apply the Von Neumann (Fourier) method of stability analysis to demonstrate that the simple implicit scheme is unconditionally stable. we have:

$$U_n - U_E = \epsilon$$

where U_n is the numerical solution , U_E the Exact solution and ϵ the error term we introduce the last equation in 5.15, we obtain:

$$\frac{\epsilon_j^{n+1} - \epsilon_j^n}{\Delta t} = \frac{\epsilon_{j-1}^{n+1} - 2\epsilon_j^{n+1} + \epsilon_{j+1}^{n+1}}{(\Delta x)^2} \tag{5.15}$$

where we replaced the space variable index i by j. The error terms $\varepsilon_j^{\mathrm{n}}$ represented as given by equations 5.6. Introducing ε_j from equations 5.6 into equation 5.15 and after cancellations and some rearrangement, we obtain:

$$A - 1 = \frac{2\alpha\Delta t}{(\Delta x)^2} A \left(\frac{e^{ik\Delta x} + e^{-ik\Delta x}}{2} - 1 \right) \tag{5.16}$$

where $i = \sqrt{-1}$. Noting that

$$\cos (k\Delta x) = \frac{e^{ik\Delta xh} + e^{-ik\Delta x}}{2}$$

equation 5.16 is written as

$$A - 1 = -4\lambda A \sin^2 \left(\frac{k\Delta \mathrm{x}}{2} \right)$$

where

$$\lambda = \frac{\Delta t}{\Delta x^2}$$

last equation is solved for A

$$A = \left[1 + 4\lambda \sin^2\left(\frac{k\Delta x}{2}\right)\right]^{-1}$$

For stability, we need $|A| \leq 1$, and this condition is satisfied for all positive values of λ. Therefore, the simple implicit finite difference approximation is stable for all values of the time step Δt.

5.2 Stability analysis of Finite Difference Semi-Implicit scheme for 1-D Perona-Malik equation

From equation 3.6 we have:

$$u_i^n = -\alpha_i u_{i-1}^{n+1} + (1 + \alpha_i + \beta_i) u_i^{n+1} - \beta_i u_{i+1}^{n+1}$$

and like the other schemes we have:

$$u_i^n = A^n e^{jk\Delta x}$$

and replace in last equation:

$$A^n e^{jk\Delta x} = -\alpha_i A^{n+1} e^{jk\Delta x(i-1)} + (1 + \alpha_i + \beta_i) A^{n+1} e^{jkih} - \beta_i A^{n+1} e^{jk(i+1)\Delta x}$$

that means:

$$A = \frac{1}{1 + \alpha_i + \beta_i - \alpha_i e^{-jk\Delta x} - \beta_i e^{jk\Delta x}}$$

In other word, we get:

$$A = \frac{1}{1 + (\alpha_i + \beta_i)(1 - cos(k\Delta x)) + i(\alpha_i - \beta_i) sin(k\Delta x)}.$$

Thus

$$\sqrt{[1 + (\alpha_i + \beta_i)(1 - cos(k\Delta x))]^2 + (\alpha_i - \beta_i)^2 sin^2(k\Delta x)} \geqslant 1.$$

Then, we have $\mid A \mid \leqslant 1$, that means our scheme is unconditionally stable for all values of the time step Δt.

5.3 Tridiagonal Matrix Algorithm (TDMA)

Tridiagonal matrix algorithm (TDMA) [36, 25] is a simplified form of Gaussian elimination, that can be used for solving tridiagonal systems of equations. In matrix/vector format this kind of a system can be written as in 5.17

$$\underbrace{\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & a_3 & b_3 & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & c_{N-1} \\ 0 & 0 & 0 & a_N & b_N \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_N \end{bmatrix}}_{d} \tag{5.17}$$

The algorithm consists of two steps: the first (forward) sweep eliminates the a_i, while the second (backward) sweep calculates the solution. Equation 34 introduces the forward sweep, while Equation 35 shows the backward sweep.

$$c'_i = \begin{cases} \frac{c_1}{b_1} & , i = 1 \\ \frac{c_i}{b_i - c'_{i-1} a_i} & , i = 2, 3, \dots, N-1 \end{cases}$$

$$d'_i = \begin{cases} \frac{d_1}{b_1} \\ \frac{d_i - d'_{i-1} a_i}{b_i - c'_{i-1} a_i} & , i = 2, 3, \dots, N \end{cases}$$

$$x_N = d'_N, i = N-1, N-2, \dots, 1$$

Physical interpretation of the terms a_i and b_i is that they are diffusion weights, i.e. how much the neighbouring solutions are taken into account.

Conclusion

In this paper, a group of partial differential equations with important application in the field of image processing is addressed, namely, the heat equation Perona-Malik equation and Beltrami flow.
Smoothing of noisy images presents usually a numerical integration of a parabolic PDE in scale or two dimensions in space. This often the most time consumptive component of image processing algorithms.
The explicit numerical integration scheme is conditionally stable. Thus, the unconditionally stable numerical scheme becomes an important matter.
The method based on Additive Operator Split (AOS), applied originally by Weickert for the nonlinear diffusion flow, may be applied for the Beltrami equation. This method makes it possible to develop the unconditionally stable semi-implicit finite difference schemes for image filtering.
A series of numerical simulations was presented in order to compare the three models and in order to clarify the difference, the focus was on study the edge enhancement effect on the one hand, and removing noise on the other hand.
We also concluded that, the linear diffusion equations lead to blurred edges, while the nonlinear diffusion equations give better results in terms of edges enhancement as they lead to sharp edges.
Finally, the difference between the Perona-Melek model and the Beltrami model can be observed, through a series of experiments, where both models remove noise and enhance the edges, but Beltrami model is more effective with 'salt and pepper' noise.

Bibliography

[1] L. Alvarez,F. Guichard, P.L. Lions,and J.M. Morel, *Axioms and fundamental equations of image processing*, Archive for Rational Mechanics and Analysis **123** , 199–257 (1993).

[2] L. Alvarez, P.L. Lions and J.M. Morel, *Image selective smoothing and edge detection by nonlinear diffusion II*, SIAM Journal on Numerical Analysis. **29**, 845–866 (1992).

[3] G. Aubert, P. Kornprobst *Mathematical Problems in Image Processing*, Partial Differential Equations and the Calculus of Variations, Applied Mathematical Sciences**147**,(2006).

[4] M. Bertero,T.A Poggio and V. Torre, *Ill-posed problems in early vision*. Proceedings of the IEEE. **76**, 869–889 (1988).

[5] N.BEN HAMIDOUCHE Image processing course Master EDP and Application, University of Msila ,2021.

[6] R.W. Brockett and P. Maragos, *Evolution equations for continuous-scale morphological filtering*. IEEE Transactions on Signal Processing **42**, 3377–3386 (1994).

[7] F. Cao, *Geometric Curve Evolution and Image Processing*, Lecture Notes in Mathematics **1805**, Springer (2003).

[8] F. Catte,P.L. Lions,J.M. Morel, and T. Coll,*Image Selective Smoothing and Edge Detection by Nonlinear Diffusion*, SIAM Journal on Numerical analysis. **29** ,182–193 (1992).

[9] T.F. Chan and J. Shen. *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*. Society for Industrial and Applied Mathematics, (2005).

[10] M.E. Gage, *Curve shortening makes convex curves circular*, Inventions Mathematica. **76**, 357–364 (1984).

[11] M.E. Gage and R.S. Hamilton, *The heat equation shrinking convex plane curves*, Journal of Differential Geometry. **23**, 69–96 (1986).

[12] I. Galic, J. Weickert, M. Welk, A. Bruhn, A. Belyaev and H.P. Seidel, *Towards PDE-based image compression*. Variational Geometric and Level-Set Methods in Computer Vision, **3752**, 37–48 (2005).

[13] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. ”Fast Geodesic Active Contours”. M. Nielsen, P. Johansen, O.F. Olsen, J. Weickert (Editors), Scale-space theories in computer vision, Lecture Notes in Computer Science, Vol. 1682, Springer, Berlin, 1999.

[14] G, Huisken, *Flow by mean curvature of convex surfaces into spheres*, Journal of Differential Geometry, **20**, 237–266 (1984).

[15] T. Iijima, *Basic equation of figure and observational transformation*. Systems Computers, **4**, 70–77 (1971).

[16] R.Igor , Emerson, *'Fast Difference Schemes for Edge Enhancing Beltrami Flow'*, Biol. Cybern, **50**,(2002).

[17] R. Malladi and J.A. Sethian, *Image processing via level set curvature flow*,proceedings of the National Academy of sciences, **92**, 7046–7050 (1995).

[18] R. Malladi and J.A. Sethian, *Image Processing: Flows under Min/ Max curvature and Mean Curvature*, Graphical Models and Image Processing. **58**, 127–141 (1996).

[19] R. Malladi and I. Ravve,Fast Difference Scheme for Anisotropic Beltrami Smoothing and Edge Contrast Enhancement of Gray Level and Color Images,University of California USA , 2001.

[20] R. Kimmel, *Numerical Geometry of Images: Theory, Algorithms, and Applications*, Springer (2003).

[21] J. Koendernik, *'The structure of images'*, Biol. Cybern, **50**, pp. 363-370 (1984).

[22] S.J. Osher and J.A. Sethian, *Fronts propagation with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics. **79**, 12–49 (1988).

[23] P. Perona, J. and Malik,*Scale-space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence **12**, 629–639 (1990).

[24] G.Rosman · L. Dascal · Xue-Cheng Tai . Ron Kimmel,On Semi-implicit Splitting Schemes for the Beltrami Color Image Filtering, Journal of Mathematical Imaging and Vision , 2011.

[25] J. Ralli,PDE Based Image Diffusion and AOS,University of Granada, Spain,2014

[26] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, (2006).

[27] K. Seongjai,Numerical Methods for Partial Differential Equations, University of Mississippi State USA ,2021.

[28] N. Sochen, R. Kimmel and R. Malladi, *A general framework for low level vision*, IEEE Transactions on Image Processing, **7**, 310–318 (1998).

[29] C.B. Schonlieb, *Partial Differential Equation Methods for Image Inpainting*. Cambri dgeUniversity Press **29**, (2015).

[30] J.A. Sethian, *An Analysis of Flame Propagation*, Ph. D. Dissertation, University of California, (1982).

[31] N. Sochen, R. Kimmel and R. Malladi, *From high energy physics to low level vision* International Conference on Scale-Space Theories in Computer Vision, 236–247, (1996).

[32] R. Van den Boomgaard, A. Smeulders, *The morphological structure of images : The differential equations of morphological scale-space*. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**, 1101–1113 (1994).

[33] J. Weickert, *Anisotropic Diffusion in Image Processing*, Stuttgart Teubner **1**, 59–60 (1998).

[34] A.P. Witken, *'Scale-space filtering'*, Presented at 8th int. Joint conf. Art. intell., Karlsruhe, Germany, (1983).

[35] M. Welk, M. Breu, O. Vogel, *Morphological amoebas are self-snakes* Journal of Mathematical Imagin and Vision **39**, 87–99 (2011).

[36] J. Weickert, Bart M. ter Haar Romeny, Max A. Viergever, Efficient and Reliable Schemes for Nonlinear Diffusion Filtering,1998.

[37] A. Yezzi, *Modified curvature motion for image smoothing and enhancement*, IEEE Transactions on Image Processing **7**, 345–352 (1998).

Abstract:

Smoothing of noisy images presents usually a numerical integration of a parabolic PDE in scale or two dimensions in space. This often the most time consumptive component of image processing algorithms.
The explicit numerical integration scheme is conditionally stable. Thus, the unconditionally stable numerical scheme becomes an important matter.
The method based on Additive Operator Split (AOS), applied originally by Weickert for the nonlinear diffusion flow, may be applied for the Beltrami equation. This method makes it possible to develop the unconditionally stable semi-implicit finite difference schemes for image filtering.
keywords: Image Processing , Edge Enhancement , Nonlinear Diffusion, Finite Difference , Perona-Malik Diffusion , Beltrami Flow.

Resumé:

Le lissage d'images bruitées présente généralement une intégration numérique d'une EDP parabolique à l'échelle ou à deux dimensions dans l'espace. C'est souvent le composant le plus chronophage des algorithmes de traitement d'image.
Le schéma d'intégration numérique explicite est conditionnellement stable. Ainsi, le schéma numérique inconditionnellement stable devient une question importante.
La méthode basée sur Additive Operator Split (AOS), appliquée à l'origine par Weickert pour le flux de diffusion non linéaire, peut être appliquée pour l'équation de Beltrami.
Cette méthode permet de développer les schémas aux différences finies semi-implicites inconditionnellement stables pour le filtrage d'images.
Mots clé: Traitement d'images , Amélioration des contours , Diffusion non linéaire , Différence finie , Diffusion Perona-Malik , Flux de Beltrami.

الملخص:

تنعيم الصور المشوشة يضهر عدديا من خلال المعادلات التفاضلية الجزئية ذات البعد الواحد او البعدين في الفضاء.
في الغالب يستهلك معالجة الصور وقتا كبيرا وهذا راجع الى الخوارزمية المستعملة.
النموذج الصريح مستقر بشروط وبالتالي فإن استقرار نموذج ما بدون شرط يعتبر ذو اهمية في معالجة الصور.
يمكن تطبيق طريقة فصل المتغيرات والتي تم استعمالها من قبل ويكارت على المعادلات غير الخطية للانتشار كما يمكن تطبيقها على معادلة بيلترامي .
الكلمات المفتاحية:معالجة الصور ، تحسين الحواف ، الانتشار غير الخطي ، الفروق المحدودة ، انتشار بيرونا مالك ، تدفق بلترامي.