Monotone Spline Interpolation of Images for PDE-Based Numerical Schemes

Bachelor's Thesis in Mathematics

submitted by
Moritz van Recum

Supervisor
Prof. Dr. Joachim Weickert

Examiners
Prof. Dr. Joachim Weickert
Prof. Dr. Sergej Rjasanow

Department of Mathematics
Faculty of Mathematics and Computer Science
Saarland University

August 25, 2023
Saarbruecken, Germany

Abstract

Image derivative approximation is a fundamental step in image processing. This can be challenging when directional derivatives are involved, as is the case with the mean curvature motion PDE.

In this thesis we apply monotone spline interpolation to images in order to approximate image derivatives in a given direction. A key requirement of the interpolant is to preserve monotonicity since this ensures a maximum-minimum principle. Together with further considerations such as smoothness and accuracy requirements this naturally leads to monotone cubic and monotone quintic spline interpolation methods. Several such interpolation methods from existing literature are implemented and tested for this purpose [5, 12, 9]. To interpolate the image, they are then applied separately in the axial directions of the image. The experiments show that the resulting methods can compete with the explicit delta scheme [13, 18] in terms of image sharpness and accuracy of the approximation while satisfying the maximum-minimum principle. The monotonic quintic spline method [8] produces a remarkably high level of image sharpness, surpassing all other methods in the experiments.

Declaration under Oath

I hereby declare in lieu of an oath that I have written this thesis independently and have not used any sources or aids other than those indicated.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass die vorliegende Arbeit eigenständig und ausschließlich unter Verwendung der genannten Quellen und Hilfsmittel erstellt wurde.

Moritz van Recum
Saarbrücken, September 16, 2023

Acknowledgments

I would like to thank Prof. Dr. Joachim Weickert for offering me a wide range of topics to choose from, for his guidance, and for giving me the opportunity to work on this fascinating subject.

In addition, I would like to express my gratitude to Dr. Thomas Lux for providing feedback on technical aspects of his work on monotone quintic spline interpolation.

Finally, I want to thank my parents for their unwavering support.

Contents

1 Introduction

The PDE-based approach in image processing assumes that the processed image is a sample of a smooth function that can be differentiated. Mean curvature motion is a prototype of a second-order anisotropic PDE for image analysis. It requires the discretisation of second-order directional derivatives. For this, we can use one-dimensional interpolants to generate corresponding grid points, as depicted in Figure 1.1. Directional Derivatives

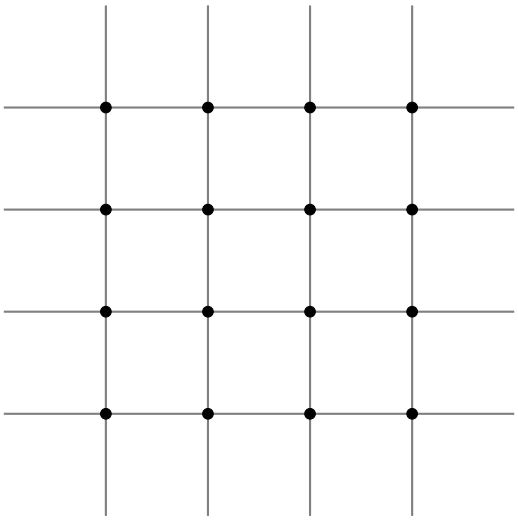


Fig. 1.1. Illustration that we interpolate along the grid lines in x and y direction of the image. Dots represent the pixels of the image. The generated grid points will be located somewhere on the grid lines and acquired by interpolation.

can then be approximated from these grid points by means of a standard one-dimensional finite difference scheme. In order to obtain a consistent numerical scheme, the interpolant should be at least third-order accurate. If the interpolant is monotone, we can also guarantee a maximum-minimum

principle. Other desirable properties are high edge sharpness, good rotation invariance and high accuracy.

All the original interpolation methods discussed here are one-dimensional. Chapter 2 introduces the basic underlying cubic and quintic splines. For the cubic case, we use the derivatives of the cubic C^2 spline, which is twice continuously differentiable, as initial estimates. The boundary conditions used are those known as the 'not-a-knot' boundary conditions. The interpolant is represented using a piecewise cubic or quintic Hermite spline function. In the cubic case, the dependence is solely on the estimated derivatives at the sample positions, besides the interpolation conditions. In the quintic case, the specification of second derivatives at sample positions is also required. Here, a quadratic facet model [6] will be used for initial estimates, as in [8].

In order to mitigate over-and under shoots of the spline functions, in Chapter 3 we discuss monotonicity preserving methods that modify the estimated derivatives to obtain a monotone interpolant on each sub-interval. For the cubic case, the Fritsch-Carlson method [5] is introduced, which also provides the framework for a further refinement of the method of Eisenstat et al. [12]. The resulting method is fourth-order accurate in space, which is what we are aiming for to achieve second-order approximations in Chapter 5. A monotone quintic spline interpolation [8] is implemented as well. We then test and apply the methods to images by interpolating separately in both axial directions of the image.

Chapter 5 presents the idea of using image interpolation to derive a numerical scheme for the mean curvature motion PDE. While the basic idea can be motivated by the delta stencil discretisation [13, 18], it skips the process of directional splitting into four finite difference approximations. For this purpose, the image is interpolated along the grid lines. In order to approximate the directional derivative, the grid is evaluated at the corresponding points of intersection in a given direction. A one-dimensional finite difference scheme can then be used for the derivative approximation, taking into account the central point at which the derivative is approximated and two interpolated grid values. Finally, an explicit time discretisation completes the numerical scheme for the PDE.

The results in Chapter 6 show that the interpolation-based approach can achieve comparable results to those of the delta stencil. The cubic interpolation methods produce fewer artefacts and are similar in quality regarding image sharpness and approximation accuracy, while satisfying a maximum-minimum principle. The delta stencil, while violating a maximum-minimum

principle, still performs slightly better in terms of rotation invariance.

In an attempt to improve rotation invariance, we implement a monotone quintic spline from Lux et al. [8], which is third-order accurate due to quadratic derivative estimates. Although rotation invariance was not improved, the images of the evolution are significantly sharpened. Possibly, the improvement is due to the use of quintic splines, which can take second derivatives as input, in combination with minimizing local curvature in the quadratic facet model [6].

2 Splines

The term spline is not strictly tied to a specific mathematical definition. It is commonly used to refer to piecewise polynomial interpolation. Historically speaking the term spline is derived from the shipbuilding industry, where wooden splines with a constant cross-section were used to model the curvature of the ship's planking. The spline was fixed in the transverse direction to simulate the natural curvature, as depicted in Figure 2.1. It

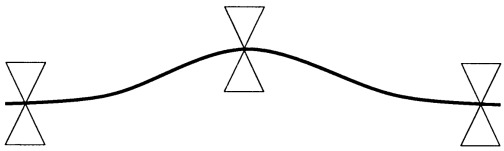


Fig. 2.1. Model of a wooden spline, fixed in traverse direction, taken from [14].

is important that no forces act in the longitudinal direction, so that the spline has minimal bending energy to assume its optimum shape. If the fixed points are taken to be knots, the modeling of the planking can be the spline as an interpolation problem. Here the bending line of the spline specifies the interpolating function of the fixed points. It can be shown that in a linearised case the interpolating function that minimizes the bending energy is exactly the cubic C^2 spline function [14].

In the following, we assume that the fixed points we want to interpolate are samples of a smooth function

$$f : [a, b] \to \mathbb{R}$$

where $a = x_0 < \ldots < x_n = b$ is a partition of the interval $[a, b]$, and $(x_0, f_0), \ldots, (x_n, f_n)$ are the $n + 1$ sampled points to interpolate. Before defining the cubic spline, we define the spline as a piecewise polynomial

interpolant of a certain order. Then we can think of a spline of order p as a piecewise polynomial function $S : [a, b] \to \mathbb{R}$ with

$$S|_{[x_i, x_{i+1}]} = s_i \tag{2.1}$$

composed of polynomial functions

$$s_i(x) = \sum_{k=0}^{p-1} a_{i,j}\, x^k,$$

of degree $p - 1$ that satisfy

$$s_i(x_i) = f_i, \tag{2.2a}$$
$$s_i(x_{i+1}) = f_{i+1} \tag{2.2b}$$

for $i = 0, \ldots, n - 1$ [14].

2.1 Cubic Splines

To define the cubic spline, we follow the reference [14] and let S be the previously defined spline function (2.1) with $p = 4$ on the interval $[a, b]$. Besides the interpolation conditions (2.2), the cubic spline is usually assumed to have continuous second derivatives, making it a cubic C^2 spline. This smoothness requirement leads to the $2n - 2$ conditions

$$s'_{i-1}(x_i) = s'_i(x_i), \tag{2.3a}$$
$$s''_{i-1}(x_i) = s''_i(x_i). \tag{2.3b}$$

for $i \in 1, \ldots, n - 1$. The $2n$ interpolation conditions (2.2), along with the $2n - 2$ smoothness conditions (2.3) and 2 additional boundary conditions (2.8), can be written as a uniquely solvable linear system of equations in the $4n$ coefficients of the s_i. This demonstrates the uniqueness and existence of the interpolating function S.

The aim is to express this linear system of equations in terms of the derivatives

$$d_i \coloneqq s'_i(x_i), \qquad i = 1, \ldots, n - 1 \tag{2.4a}$$
$$d_n \coloneqq s'_{n-1}(x_n) \tag{2.4b}$$

The slopes d_i are considered as independent variables. From now on, let s_i be expressed in terms of a coordinate system that has been shifted by x_i. To be more specific, let the new coefficients be represented by $\{c_{i,k}\}$ such that

$$s_i(x) \coloneqq c_{i,0} + c_{i,1}\,(x - x_i) + c_{i,2}\,(x - x_i)^2 + c_{i,3}\,(x - x_i)^3.$$

for $i = 0, \ldots, n-1$. The coefficients of $c_{i,k}$ can be expressed in terms of the slopes d_i by applying the interpolation conditions and the smoothness conditions. We immediately obtain the coefficients

$$\begin{aligned} c_{0,i} &= f_i, \\ c_{1,i} &= d_i. \end{aligned}$$

After applying linear transformations we obtain the coefficients

$$c_{2,i} = \frac{3f_{i+1} - 3f_i - 2d_i\,\Delta x_i - d_{i+1}\,\Delta x_i}{{\Delta x_i}^2} \tag{2.5}$$

and

$$c_{3,i} = \frac{2f_i - 2f_{i+1} + d_i\,\Delta x_i + d_{i+1}\,\Delta x_i}{{\Delta x_i}^3}. \tag{2.6}$$

where $\Delta x_i \coloneqq x - x_i$. The linear system of equations in terms of the slopes can be written as

$$\begin{pmatrix} b_0 & c_0 & & & & \\ \Delta x_1 & 2(\Delta x_1 + \Delta x_0) & \Delta x_0 & & & \\ & \Delta x_2 & 2(\Delta x_2 + \Delta x_1) & \Delta x_1 & & \\ & \ddots & \ddots & \ddots & & \\ & & \Delta x_{n-1} & 2(\Delta x_{n-1} + \Delta x_{n-2}) & \Delta x_{n-2} & \\ & & & a_n & b_n & \end{pmatrix} \begin{pmatrix} d_0 \\ \\ \vdots \\ \\ d_n \end{pmatrix}$$

$$= \begin{pmatrix} e_0 \\ 3\left(\frac{(f_2 - f_1)\Delta x_0}{\Delta x_1} + \frac{(f_1 - f_0)\Delta x_1}{\Delta x_0}\right) \\ \vdots \\ 3\left(\frac{(f_n - f_{n-1}\Delta x_{n-2}}{\delta x_{n-1}} + \frac{(f_{n-1} - f_{n-2})\Delta x_{n-1}}{\Delta x_{n-2}}\right) \\ e_n \end{pmatrix} \tag{2.7}$$

The matrix of this system of equations is tridiagonal and consists of a main diagonal $b_0, \ldots, b_n$, lower sub-diagonal $a_1, \ldots, a_n$, upper sub-diagonal

$c_0, \ldots, c_{n-1}$ and a right side $e_0, \ldots, e_n$. The values b_0, c_0, a_n, b_n, e_0 and e_n are free parameters which depend on the choice of boundary conditions. We use the "not-a-knot" boundary conditions. These conditions require the second knot (x_1, f_1) and the second last knot (x_{n-1}, f_{n-1}) to be completely smooth [12], i.e., the spline S must be three times differentiable at x_1 and at x_{n-1}. The boundary conditions can be expressed as

$$s_0'''(x_1) = s_1'''(x_1), \tag{2.8}$$
$$s_{n-2}'''(x_{n-1}) = s_{n-1}'''(x_{n-1}) \tag{2.9}$$

which is equivalent to

$$c_{3,0} = c_{3,1},$$
$$c_{3,n-2} = c_{3,n-1}.$$

With (2.6) the boundary conditions can be expressed in terms of the values f_i and slopes d_i as

$$\frac{2f_0 - 2f_1 + d_0\,\Delta x_0 + d_1\,\Delta x_0}{{\Delta x_0}^3} = \frac{2f_1 - 2f_2 + d_1\,\Delta x_1 + d_2\,\Delta x_1}{{\Delta x_1}^3},$$
$$\frac{2f_{n-2} - 2f_{n-1} + d_{n-2}\,\Delta x_{n-2} + d_{n-1}\,\Delta x_{n-2}}{{\Delta x_{n-2}}^3} = \frac{2f_{n-1} - 2f_n + d_{n-1}\,\Delta x_{n-1} + d_n\,\Delta x_{n-1}}{{\Delta x_{n-1}}^3}.$$

To calculate the values of b_i, c_i, e_i, $(i = 0, n)$ required for the boundary conditions, we must first write these equations in terms of the slopes d_0, d_1, d_3, d_{n-2}, d_{n-1}, and d_n.. We obtain the equations

$$\frac{1}{{\Delta x_0}^2}\,d_0 + \left(\frac{1}{{\Delta x_0}^2} - \frac{1}{{\Delta x_1}^2}\right) d_1 - \frac{1}{{\Delta x_1}^2}\,d_2 = \frac{2f_1 - 2f_2}{{\Delta x_1}^3} + \frac{2f_1 - 2f_0}{{\Delta x_0}^3},$$
$$\frac{1}{{\Delta x_{n-2}}^2}\,d_{n-2} + \left(\frac{1}{{\Delta x_{n-2}}^2} - \frac{1}{{\Delta x_{n-1}}^2}\right) d_{n-1} - \frac{1}{{\Delta x_{n-1}}^2}\,d_n = \frac{2f_{n-1} - 2f_n}{{\Delta x_{n-1}}^3} + \frac{2f_{n-1} - 2f_{n-2}}{{\Delta x_{n-2}}^3}.$$

With this we obtain the linear system

$$\begin{pmatrix} \frac{1}{{\Delta x_0}^2} & \left(\frac{1}{{\Delta x_0}^2} - \frac{1}{{\Delta x_1}^2}\right) & -\frac{1}{{\Delta x_1}^2} & & \\ \Delta x_1 & 2(\Delta x_1 + \Delta x_0) & \Delta x_0 & & \\ & \Delta x_2 & 2(\Delta x_2 + \Delta x_1) & \Delta x_1 & \\ & \ddots & \ddots & \ddots & \\ & & \Delta x_{n-1} & 2(\Delta x_{n-1} + \Delta x_{n-2}) & \Delta x_{n-2} \\ & & \frac{1}{{\Delta x_{n-2}}^2} & \left(\frac{1}{{\Delta x_{n-2}}^2} - \frac{1}{{\Delta x_{n-1}}^2}\right) & -\frac{1}{{\Delta x_{n-1}}^2} \end{pmatrix} \begin{pmatrix} d_0 \\ \\ \vdots \\ \\ d_n \end{pmatrix}$$

$$= \begin{pmatrix} \frac{2f_1 - 2f_2}{{\Delta x_1}^3} + \frac{2f_1 - 2f_0}{{\Delta x_0}^3} \\ 3\left(\frac{(f_2 - f_1)\Delta x_0}{\Delta x_1} + \frac{(f_1 - f_0)\Delta x_1}{\Delta x_0}\right) \\ \vdots \\ 3\left(\frac{(f_n - f_{n-1}\Delta x_{n-2}}{\Delta x_{n-1}} + \frac{(f_{n-1} - f_{n-2})\Delta x_{n-1}}{\Delta x_{n-2}}\right) \\ \frac{2f_{n-1} - 2f_n}{{\Delta x_{n-1}}^3} + \frac{2f_{n-1} - 2f_{n-2}}{{\Delta x_{n-2}}^3} \end{pmatrix}.$$

The system now needs to be transformed into a tridiagonal form. To achieve this, the first (last) row can be subtracted by a multiple of the second (second last) row. For the first row, this multiple is

$$\frac{1}{{\Delta x_1}^2 \, \Delta x_0},$$

and for the last row it is

$$-\frac{1}{{\Delta x_{n-2}}^2 \, \Delta x_{n-1}}.$$

Applying this linear transformation results in a tridiagonal system that is

equivalent, and we can obtain the entries

$$b_0 = \frac{1}{{\Delta x_0}^2} + \frac{1}{\Delta x_1 \Delta x_0}, \tag{2.10a}$$

$$c_0 = \frac{1}{{\Delta x_0}^2} - \frac{1}{{\Delta x_1}^2} + \frac{2\Delta x_1 + \Delta x_0}{{\Delta x_1}^2 \Delta x_0}, \tag{2.10b}$$

$$e_0 = \frac{2f_1 - 2f_2}{{\Delta x_1}^3} + \frac{2f_1 - 2f_0}{{\Delta x_0}^3} + 3\left(\frac{f_2 - f_1}{{\Delta x_1}^3} + \frac{f_1 - f_0}{\Delta x_1 {\Delta x_0}^2}\right), \tag{2.10c}$$

$$a_n = \frac{1}{{\Delta x_{n-2}}^2} - \frac{1}{{\Delta x_{n-1}}^2} - \frac{2(\Delta x_{n-1} + \Delta x_{n-2})}{{\Delta x_{n-2}}^2 \Delta x_{n-1}}, \tag{2.10d}$$

$$b_n = -\frac{1}{{\Delta x_{n-1}}^2} - \frac{1}{\Delta x_{n-2} \Delta x_{n-1}}, \tag{2.10e}$$

$$e_n = \frac{2f_{n-1} - 2f_n}{{\Delta x_{n-1}}^3} + \frac{2f_{n-1} - 2f_{n-2}}{{\Delta x_{n-2}}^3} - 3\left(\frac{(f_n - f_{n-1}}{\Delta x_{n-2} {\Delta x_{n-1}}^2} + \frac{(f_{n-1} - f_{n-2})}{{\Delta x_{n-2}}^3}\right). \tag{2.10f}$$

needed for the "not-a-knot" boundary conditions. The following algorithms outline the computation of the cubic C^2 spline derivatives.

Algorithm 1. `cubic_spline_derivs` (x, f, n)
where $x_i, f_i \in \mathbb{R}$ for $i = 0, \ldots, n-1$ and $n \geq 2$. Returns the slopes d_i, $i = 1, \ldots, n-1$ such that the resulting piecewise cubic Hermite spline through (x_i, f_i, d_i), $i = 1, \ldots, n-1$ is twice differentiable.
$\Delta x_i \coloneqq x_{i+1} - x_i$ for $i = 0, \ldots, n-1$
set $b_0\, c_0$, e_0 and a_{n-1}, b_{n-1}, e_{n-1} as in (2.10).
set $a_i, \ldots, e_i$ as in (2.7):
`for` $i \coloneqq 1$ `step` 1 `to` $n-2$ `do`
    $a_i \coloneqq \Delta x_i$
    $b_i \coloneqq 2\,(\Delta x_i + \Delta x_{i-1})$
    $c_i \coloneqq \Delta x_{i-1}$
    $e_i \coloneqq 3\,((f_{i+1} - f_i)\Delta x_{i-1}/\Delta x_i + (f_i - f_{i-1})\Delta x_i/\Delta x_{i-1})$
`end for`
$d \coloneqq$ `tridiagonal_matrix_algorithm` (a, b, c, e, n)
`return` d

Algorithm 2. [4] `tridiagonal_matrix_algorithm` (a, b, c, e, n)
where $a_1, \ldots, a_{n-1} \in \mathbb{R}$ are the lower diagonal entries,
$b_0, \ldots, b_{n-1} \in \mathbb{R}$ are the diagonal entries,

$c_0, \ldots, c_{n-2} \in \mathbb{R}$ are the upper diagonal entries and
$e_0, \ldots, e_{n-1} \in \mathbb{R}$ is the right side of the tridiagonal system.
Returns the solution of the tridiagonal system$d_1, \ldots, d_n \in \mathbb{R}$.
forward sweep:
$c_0 = c_0 / b_0$
$e_0 = e_0 / b_0$
`for` $i \coloneqq 1$ `step` 1 `to` $n - 2$ `do`
    $c_i \coloneqq c_i / (b_i - c_{i-1}\, a_i)$
    $d_i \coloneqq (e_i - e_{i-1}\, a_i) / (b_i - c_{i-1}\, a_i)$
`end for`
back substitution:
$d_{n-1} = e_{n-1}$
`for` $i \coloneqq n - 2$ `step` -1 `to` 0 `do`
    $d_i \coloneqq e_i - c_i\, d_{i+1}$
`end for`
`return` d

2.2 Cubic Hermite Splines

In the previous section, we obtained the derivatives $d_0, \ldots, d_n$ which lead to a cubic spline function that is twice continuously differentiable. Conversely, the cubic Hermite spline offers a way of expressing the resulting spline function based on these or even arbitrarily chosen derivatives. The cubic Hermite spline can be seen as a piecewise polynomial function S of order four as described in Chapter 2. It is defined on each interval $[x_i, x_{i+1}]$ by a polynomial function that interpolates the given function values (x_i, f_i), (x_{i+1}, f_{i+1}) and the derivatives (x_i, d_i), (x_{i+1}, d_{i+1}) for $0 \leq i < n$. For each interval $[x_i,\, x_{i+1}]$ these are four conditions for the four coefficients. More specifically, let

$$s_i = a_0 + a_1\, x + a_2 x^2 + a_3 x^3$$

be a the cubic polynomial that defines the cubic Hermite spline on the interval $[x_i, x_{i+1}]$. Then we have the conditions

$$s_i(x_i) = f_i, \tag{2.11a}$$
$$s_i(x_{i+1}) = f_{i+1}, \tag{2.11b}$$
$$s_i'(x_i) = d_i, \tag{2.11c}$$
$$s_i'(x_{i+1}) = d_{i+1}. \tag{2.11d}$$

The polynomial that satisfies these conditions is called the Hermite interpolant through (x_i, f_i, d_i) and $(x_{i+1}, f_{i+1}, d_{i+1})$. It is uniquely determined by these four conditions and is of order four [3]. As a consequence, the interpolation conditions (x_i, f_i, d_i) for $i = 0, \ldots, n$ lead to a spline function of order four in the sense of (2.1). It is called a cubic Hermite spline in this case. Note that by construction the cubic Hermite spline is always at least once continuous differentiable.

A representation of the polynomials s_i for the cubic Hermite spline is given by

$$s_i(x) = c_0 + c_1 \Delta x_i + c_2 {\Delta x_i}^2 + c_3 {\Delta x_i}^3 \tag{2.12}$$

with the coefficients

$$c_0 = f_i \tag{2.13a}$$
$$c_1 = d_i \tag{2.13b}$$
$$c_2 = -3\, f_i + 3\, f_{i+1} - 2 d_i - d_{i+1} \tag{2.13c}$$
$$c_3 = 2\, f_i + d_i - 2\, f_{i+1} + d_{i+1}. \tag{2.13d}$$

where $\Delta x_i \coloneqq x_{i+1} - x_i$ for $0 \leq i \leq n-1$ [11]. This representation (2.12), (2.13) will be used in the algorithms to calculate the cubic Hermite polynomials.

2.3 Quintic Hermite Splines

Similarly, the quintic Hermite spline can be seen as a piecewise polynomial function S of order six as described in Chapter 2. It is defined on each interval $[x_i, x_{i+1}]$ by a polynomial function that interpolates the given function values (x_i, f_i), (x_{i+1}, f_{i+1}), the derivatives (x_i, Df_i), (x_{i+1}, Df_{i+1}) and the second derivatives $(x_i, D^2 f_i)$, $(x_{i+1}, D^2 f_{i+1})$ for $0 \leq i < n$, following the notation in [8].

On each interval $[x_i, x_{i+1}]$, the interpolation conditions for the polyno-

mial s_i are

$$s_i(x_i) = f_i, \tag{2.14}$$
$$s_i(x_{i+1}) = f_{i+1}, \tag{2.15}$$
$$s_i'(x_i) = Df_i, \tag{2.16}$$
$$s_i'(x_{i+1}) = Df_{i+1}, \tag{2.17}$$
$$s_i''(x_i) = D^2 f_i, \tag{2.18}$$
$$s_{i+1}''(x_{i+1}) = D^2 f_{i+1}. \tag{2.19}$$

The polynomial that satisfies these conditions is called the Hermite interpolant through $(x_i, f_i, Df_i, D^2 f_i)$ and $(x_{i+1}, f_{i+1}, Df_{i+1}, D^2 f_{i+1})$. It is uniquely determined by these six conditions and is of order six. As a consequence, the interpolation conditions $(x_i, f_i, Df_i, D^2 f_i)$ for $i = 0, \ldots, n$ lead to a spline of order six in the sense of (2.1). It is called a quintic Hermite spline in this case [3]. Note that by construction the resulting quintic Hermite spline $S : [a, b] \to \mathbb{R}$ is at least twice differentiable since the first and second derivatives on the interval boundaries coincide. The quintic Hermite Spline can be defined on each interval $[x_i, x_{i+1}]$ by

$$s_i(x) = c_0 + c_1\,\Delta x_i + c_2\,\Delta {x_i}^2 + c_3\,\Delta {x_i}^3 + c_4\,\Delta {x_i}^4 + c_5\,\Delta {x_i}^5 \tag{2.20}$$

where $\Delta x_i \coloneqq x_{i+1} - x_i$ with the coefficients

$$c_0 = f_i \tag{2.21a}$$
$$c_1 = Df_i \tag{2.21b}$$
$$c_2 = \tfrac{1}{2} D^2 f_i \tag{2.21c}$$
$$c_3 = -10 f_i - 6\,Df_i - \tfrac{3}{2}\,D^2 f_i + \tfrac{1}{2}\,D^2 f_{i+1} - 4\,Df_{i+1} + 10 f_{i+1} \tag{2.21d}$$
$$c_4 = 15 f_i + 8\,Df_i + \tfrac{3}{2}\,D^2 f_i - D^2 f_{i+1} + 7\,Df_{i+1} - 15 f_{i+1} \tag{2.21e}$$
$$c_5 = -6 f_i - 3\,Df_i - \tfrac{1}{2}\,D^2 f_i + \tfrac{1}{2}\,D^2 f_{i+1} - 3\,Df_{i+1} + 6 f_{i+1}. \tag{2.21f}$$

for $0 \leq i \leq n - 1$ [11].

This representation (2.20), (2.21) will be used in the algorithms to calculate the quintic Hermite polynomials.

3 Monotone Splines

To introduce the monotone splines, it is necessary to define what we mean by monotone interpolation. In our applications, the interpolated data may not necessarily be monotone across the entire interpolation domain. However, it makes sense to first assume that the data is monotone increasing and then expand the method to handle local extrema. The monotone decreasing case can then be addressed similarly to the monotone increasing case. We say that a the spline function S as defined in Chapter 2 preserves monotonicity if it is monotone on each interval $[x_i, x_{i+1}]$. A method that produces spline functions that preserve monotonicity for a given data set is referred to as a monotone spline. This definition implies that the derivatives at local maxima must vanish in order for the interpolant to be differentiable while preserving monotonicity. If $(f_{i+1}-f_i)\,(fi-f_{i-1}) < 0$ for some i in $\{1, \ldots, n-1\}$, then d_i must be zero.

In the following sections, we introduce the implemented monotone cubic spline methods and extend them to the case of nonmonotone data.

3.1 The Fritsch-Carlson Method

As in Section 2.2 let p_i be the unique cubic polynomial that interpolates f_i, f_{i+1} on the interval $[x_i, x_{i+1}]$ with derivatives d_i, d_{i+1}. In order to find a way to modify the initial derivatives in an appropriate way, it is necessary to have at least a sufficient condition for the monotonicity of p_i on $[x_i, x_{i+1}]$. In [5], Fritsch and Carlson present sufficient and necessary conditions on the derivatives such that the resulting interpolant is monotone. These conditions can be expressed in terms of a compact monotonicity region $\mathcal{M} \subseteq \mathbb{R}^2$ that constrains the derivative pair $(d_i,\, d_{i+1})$. This region has the geometric shape of an ellipse given by

$$\{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 + xy - 6x - 6y + 9 >= 0\}$$

conjoint with the triangle defined by the points $(0, 0)$, $(0, 3)$ and $(3, 0)$, as depicted in Figure 3.1.

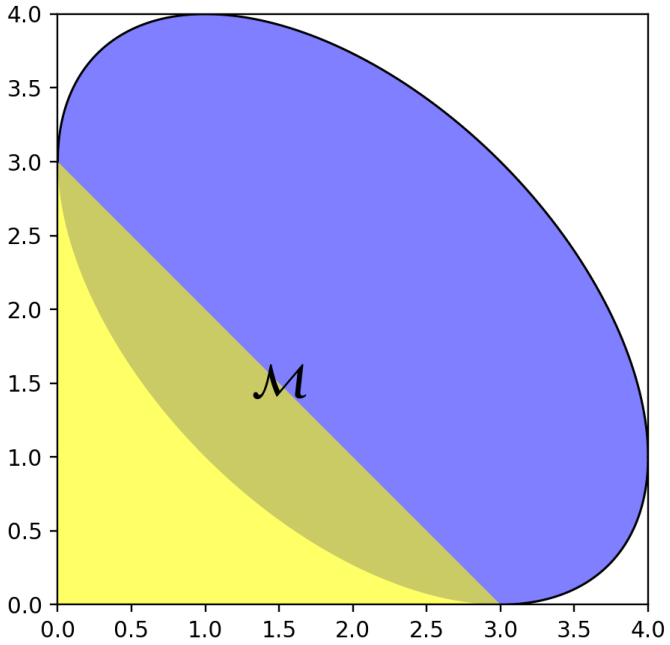


Fig. 3.1. The monotonicity region $\mathcal{M}$ [5]. The colored area identifies the compact region $\mathcal{M}$. For a given pair of derivatives (d_i, d_{i+1}), d_i corresponds to the x coordinate and d_{i+1} to the y coordinate, up to a scaling factor defined below.

Let p_i be the Hermite polynomial function on the interval $[x_i, x_{i+1}]$ that is defined by the function values f_i, f_{i+1} and derivatives d_i, d_{i+1} as in Section 2.2. Fritsch and Carlson show that the polynomial p_i is monotone on the interval $[x_i, x_{i+1}]$ if and only if the corresponding derivative pair

$$(d_i, d_{i+1}) \in \mathcal{M} \cdot \Delta_i$$

where the scaling factor $\Delta_i \coloneqq \Delta f_i \,/ \Delta x_i$ is the secant line between $(x_i,\ f_i)$ and $(x_{i+1},\ f_{i+1})$ and

$$\mathcal{M} \cdot \Delta_i \coloneqq \{(x\,\Delta_i,\ y\,\Delta_i) \in \mathbb{R}^2 \,|\, (x, y) \in \mathcal{M}\}$$

is the set $\mathcal{M}$ scaled by a factor of Δ_i [5].

The goal of the following methods is to modify the derivatives $d_0, \ldots, d_n$ such that $(d_i, d_{i+1}) \in \mathcal{M} \cdot \Delta_i$ for $i \in \{0, \ldots, n-1\}$. The main difficulty in finding such modifications is that neighbouring intervals share common derivative values. For example, if we change (d_i, d_{i+1}) for some $i < n-2$ such that p_i is monotone on $[x_i, x_{i+1}]$ then we implicitly change p_{i+1} by the derivative d_{i+1}. This can make previous modifications on the interval $[x_{i+1}, x_{i+2}]$ ineffective.

The approach by Fritsch and Carlson in [5] is to project each (d_i, d_{i+1}) onto a subset of $\mathcal{M} \cdot \Delta_i$. They show for certain subsets $\mathcal{S} \subseteq \mathcal{M}$ that the modified derivatives of adjacent intervals always remain within $\mathcal{M}$ when projected onto $\mathcal{S} \cdot \Delta_i$ for $i \in \{0, \ldots, n-1\}$. According to Eisenstat et al. [12] the most common of these subsets is

$$\mathcal{S}_2 = \{(x, y) \in \mathbb{R}^2_{\geq 0} \mid \|(x, y)\|_2 \leq 3\}$$

as depicted in Figure 3.2. In the following we will refer to the method that results from projection onto the subset $\mathcal{S}_2$ as the Fritsch-Carlson method. Algorithm 3 shows this basic modification of the slopes d_i.

Algorithm 3. `modify_slopes_basic` (d, Δ, n)
where $d = (d_0, \ldots, d_{n-1})$ are the initial derivatives and $\Delta = (\Delta_0, \ldots, \Delta_{n-2})$ are the secant lines. Modifies the slopes d_i using the Fritsch-Carlson method with the subset $\mathcal{S}_2$ [5].

```
for i ≔ 0 step 1 to n − 2 do
  if (r ≔ ‖(d_i, d_{i+1})‖_2 > 3 Δ_i)
    d_i ≔ d_i 3Δ_i/r
    d_{i+1} ≔ d_{i+1} 3Δ_i/r
  end if
end for
```

Algorithm 3 will be used to obtain derivatives that lead to a monotone spline based on the Fritsch-Carlson method. The framework for obtaining a monotone spline starting from arbitrary sample points (x_i, f_i) for $i = 0, \ldots, n$ will be developed in the following, together with other ways of modifying the initial derivatives for the sake of monotonicity.

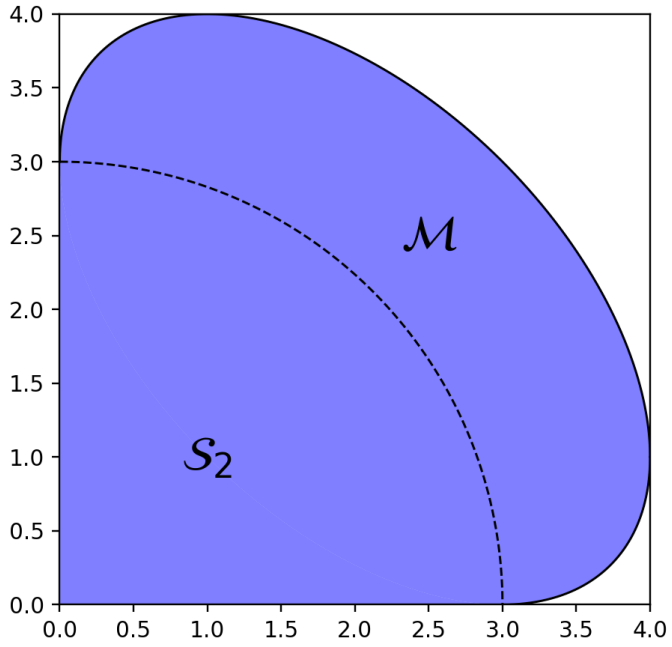


Fig. 3.2. The region $\mathcal{S}_2$ a subset of the region $\mathcal{M}$ in [5]. The dashed line indicates the boundary of the region $\mathcal{S}_2$. It is a circle with radius 3 centered in the origin in the first quadrant of the coordinate system.

3.2 The Extended Two-Sweep Method

To discuss the methods in this section, we first define what we mean by the order of accuracy of a given interpolation method. It should be noted that the following definitions are not intended to serve as evidence of the order of accuracy, but to offer a fundamental understanding of the term.

For interpolants as approximating functions, the order of accuracy is defined as the order of the error between the interpolating function S and the approximated smooth function f. More specifically, if for any smooth function f with sample points $(x_0, f_0), \ldots, (x_n, f_n)$, for the corresponding spline S there exists some C that is independent of h such that

$$\|S - f\|_\infty \leq C\, h^p \tag{3.1}$$

for sufficiently small h, then the method that results in the spline S is order

p accurate. Under these conditions we also say that p is the approximation order of the method. Here, h is the maximal distance between two neighbouring sample positions x_{i+1} and x_i for $0 \leq i < n$, i.e.

$$h = \max_{1 \leq i < n} |x_{i+1} - x_i|. \tag{3.2}$$

The definition in (3.1) follows from the order of approximation in the L_∞ space of functions. In the case of piecewise cubic Hermite splines with exact function values f_i and estimated derivatives d_i for $i = 0, \ldots, n$, the order of accuracy only depends on the estimated derivatives d_i. If the derivatives are second order accurate, then the resulting spline function is third-order accurate. If the derivatives are third order accurate, then the resulting spline function is fourth-order accurate, etc. Here, the order of accuracy of the derivatives is the error of the approximation depending on the grid size h, as in (3.2). If there exists a C independent of h such that

$$|d_i - f'(x_i)| < C\, h^{p-1} \tag{3.3}$$

for all $i = 0, \ldots, n$ and sufficiently small h, then the derivatives are order $p-1$ accurate, and the resulting spline interpolant is order p accurate [12].

Having defined the order of accuracy, we can now proceed with the discussion of the monotone spline methods. The previous section introduced approach by Fritsch and Carlson [5]. They derive necessary and sufficient conditions for cubic Hermite monotonicity, in form of a monotonicity region $\mathcal{M}$ for the derivative pairs (d_i, d_{i+1}). The derivative pairs of the spline are then projected onto a certain subset of $\mathcal{M}$ in order to achieve monotonicity for the spline function.

In [12], Eisenstat et al. propose a more sophisticated projection of the derivatives d_i directly onto the boundary on $\mathcal{M}$. They propose a two sweep method which will be discussed below. They show that this method is third-order accurate, as is the Fritsch-Carlson method. Then, they extend their two-sweep method to the extended two-sweep method. They show that their extended two-sweep method is fourth-order accurate, which is what we are aiming for.

The two-sweep method consists of a forward-sweep and a backward-sweep which are executed in succession. In the forward-sweep only derivatives d_{i+1} of the pair (d_i, d_{i+1}) are modified. This ensures that changes that have been made to interval i during the forward sweep are not being influenced by the modifications of the next interval $i+1$. Similarly during the backward sweep only d_i of the pair (d_i, d_{i+1}) is modified.

In the extended two-sweep there are two exceptions to this rule which make the projection distance shorter in some edge cases. This leads to the higher approximation order.

The algorithm itself is described in three steps. The first step is to approximate the derivatives which are assumed to be the cubic C^2 spline derivatives from Section 2.1.

The second step is to ensure that each d_i is nonnegative. In order to expand the method to nonmonotone samples we change this step to ensure that d_i have the same sign as the secant line Δ_i.

In the third step the derivatives are modified by the respective method. By using Algorithm 3 as second step we obtain the Fritsch-Carlson method. Replacing it by Algorithm 4 and Algorithm 5 results in the two-sweep methods.

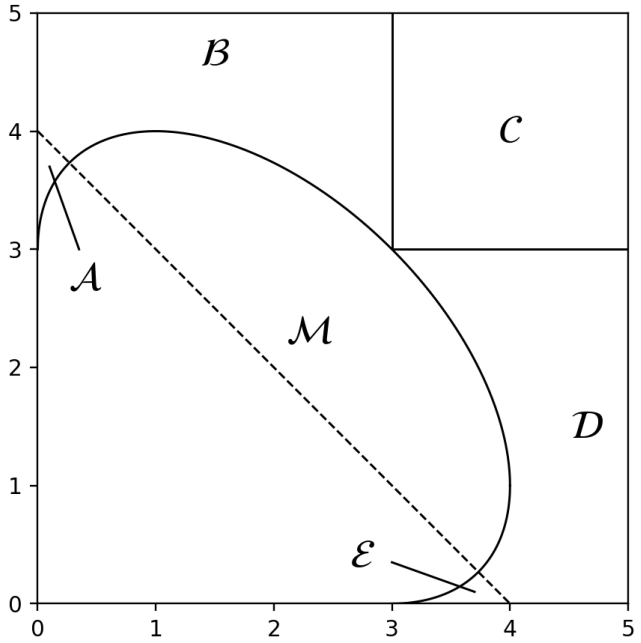


Fig. 3.3. The monotonicity region $\mathcal{M}$ and the regions $\mathcal{A}, \ldots, \mathcal{E}$, reproduced original plot in [12].

In the following the regions $\mathcal{A}, \ldots, \mathcal{E}$ are defined in order to describe the actual modifications of the two-sweep methods. Let $I \coloneqq \{(x, y) \in \mathbb{R}^2 \,|\, x \geq$

0 and $y \geq 0\}$ be the first quadrant of the coordinate system. The regions given by

$$\begin{aligned}
\mathcal{A} &= \{(x,y) \in I \,|\, x + y \leq 4 \text{ and } x \leq 3\} \cap (I \setminus \mathcal{M}) \\
\mathcal{B} &= \{(x,y) \in I \,|\, x \leq 3 \text{ and } x + y \geq 4\} \cap (I \setminus \mathcal{M}) \\
\mathcal{C} &= \{(x,y) \in I \,|\, x \geq 3 \text{ and } y \geq 3\} \\
\mathcal{D} &= \{(x,y) \in I \,|\, x \leq 3 \text{ and } x + y \geq 4\} \cap (I \setminus \mathcal{M}) \\
\mathcal{E} &= \{(x,y) \in I \,|\, x + y \leq 4 \text{ and } x \geq 3\} \cap (I \setminus \mathcal{M})
\end{aligned}$$

as depicted in Figure 3.3.

Now we can describe the original two-sweep and the extended two-sweep method by Algorithm 4 and Algorithm 5. Here, the notation $\mathcal{M}_i \coloneqq \mathcal{M} \cdot \Delta_i$ is used, similarly for the sets $\mathcal{A}, \ldots, \mathcal{E}$, as in [12].

Algorithm 4. *The pseudocode is taken directly from [12] with some minor changes in notation.*
`two_sweep` (d, Δ, n)
where $d = (d_0, \ldots, d_n)$ are the approximated derivatives and $(\Delta_0, \ldots, \Delta_{n-1})$ are the secant lines. Modifies the derivatives d_i for monotonicity.
Forward Sweep
`for` $i \coloneqq 0$ `step` 1 `to` $n - 1$ `do`
  `if` $(d_i, d_{i+1}) \in \mathcal{C}_i$ `then`
    $d_{i+1} \coloneqq 3\,\Delta_i$
  `else if` $(d_i, d_{i+1}) \in \mathcal{A}_i \cup \mathcal{B}_i$ `then`
    Decrease d_{i+1} to project (d_i, d_{i+1}) onto the boundary of $\mathcal{M}_i$
  `end if`
`end for`
Backward Sweep
`for` $i \coloneqq n - 1$ `step` -1 `to` 0 `do`
  `if` $(d_i, d_{i+1}) \in (\mathcal{D}_i \cup \mathcal{E}_i)$ `then`
    Decrease d_i to project (d_i, d_{i+1}) onto the boundary of $\mathcal{M}_i$
  `end if`
`end for`

Note that the extended two-sweep method must be adapted for the general nonmonotone case. The derivatives at knots that are local extrema should always vanish. An additional if-statement is added to each sweep to ensure that derivatives at local extrema are not modified (see Algorithm 4 and

Algorithm 5). In this case, the algorithm reverts to the normal two-sweep method, which does not have a case where zero derivatives are modified.

Algorithm 5. *The pseudocode is taken directly from [12] with some minor changes in notation and with the documented adaptations.*
`extended_two_sweep` (d, Δ, n)
where $d = (d_0, \ldots, d_n)$ are the approximated derivatives and $(\Delta_0, \ldots, \Delta_{n-1})$ are the secant lines. Modifies the derivatives d_i for monotonicity.
Forward Sweep
`for` $i \coloneqq 0$ `step` 1 `to` $n-1$ `do`
  `case` $(d_i, d_{i+1}) \in \mathcal{C}_i$
    $d_i \coloneqq 3\,\Delta_i$
  `case` $(d_i, d_{i+1}) \in \mathcal{B}_i$
    Decrease d_{i+1} to project (d_i, d_{i+1}) onto the boundary of $\mathcal{M}_i$
  `case` $(d_i, d_{i+1}) \in \mathcal{A}_i$
    ensure vanishing derivatives at local extrema are not modified:
    `if` $(i > 0)$ and $(\Delta_i = 0$ or $\Delta_{i-1}\Delta_i < 0)$ `then`
      Decrease d_{i+1} to project (d_i, d_{i+1}) onto the boundary of $\mathcal{M}_i$
      `continue`
    `end if`
    Increase d_i until either
      (d_i, d_{i+1}) reaches the boundary of $\mathcal{A}_i$ or
      (d_{i-1}, d_i) reaches the boundary of $\mathcal{M}_{i-1} \cup \mathcal{D}_{i-1} \cup \mathcal{E}_{i-1}$ (if $i > 1$)
    `if` $(d_i, d_{i+1}) \notin \mathcal{M}_i$ `then`
      Decrease d_{i+1} to project (d_i, d_{i+1}) onto the boundary of $\mathcal{M}_i$
    `end if`
`end for`
Backward Sweep
`for` $i \coloneqq n-1$ `step` -1 `to` 0 `do`
  `case` $(d_i, d_{i+1}) \in \mathcal{D}_i$
    Decrease d_i to project (d_i, d_{i+1}) onto the boundary of $\mathcal{M}_i$
  `case` $(d_i, d_{i+1}) \in \mathcal{E}_i$
    ensure vanishing derivatives at local extrema are not modified:
    `if` $(i < n-1)$ and $(\Delta_{i+1} = 0$ or $\Delta_i\Delta_{i+1} < 0)$ `then`
      Decrease d_i to project (d_i, d_{i+1}) onto the boundary of $\mathcal{M}_i$
      `continue`
    `end if`
    Increase d_{i+1} until either
      (d_i, d_{i+1}) reaches the boundary of $\mathcal{E}_i$ or

```
      (d_{i+1}, d_{i+2}) reaches the boundary of M_{i+1} (i < n − 1)
    if (d_i, d_{i+1}) ∉ M_i then
      Decrease d_i to project (d_i, d_{i+1}) onto the boundary of M_i
    end if
end for
```

Note that all algorithms and code used to generate results relating to the two-sweep method and the extended two-sweep method in this thesis are based on the original paper by Eisenstat et al. [12]. In particular, the pseudocode instructions in [12] were taken verbatim to create Algorithm 4 and Algorithm 5, taking into account the modifications mentioned above. The experiments carried out in this thesis, which use these methods, are based on a direct adaptation of the code outlined in [12]. Reference is made to the original article [12] when describing the experiments that use the adaptation described here.

3.3 Monotone Quintic Spline

In [8], Lux et al. construct a monotone quintic spline interpolant. We represent the actual spline function here by the quintic Hermite spline from Section 2.3, rather than using a B-spline basis as in the original method. This does not change the actual spline function, it just fits into our framework of using piecewise Hermite splines.

The difference to the cubic case is that for the quintic Hermite spline, in addition to the first derivatives, the second derivatives at the knots must also be specified. Then they are modified to preserve monotonicity, as we know from the cubic case. However, now we have to modify the second derivatives as well. This includes local extrema where the first derivative must still vanish, but the second derivative must be estimated.

For the first step of estimating initial derivatives, Lux et al. implemented a quadratic facet model from image processing [6]. For the second step of modifying the derivatives to preserve monotonicity, they use the tight constraints established by Ulrich and Watson [15] and a simplified case by Heß and Schmidt [7].

Let $(x_0, f_0), \ldots, (x_n, f_n)$ be the data points that should be interpolated where $n \geq 3$ and $x_0 \leq \ldots \leq x_n$. Then the quadratic facet model generally includes five points for the estimation of each slope Df_i and curvature $D^2 f_i$

at x_i. The general idea is to take the slope and curvature of the left, middle or right quadratic interpolant of (x_i, f_i), i.e. a parabola, depending on which has the least curvature at x_i. Of course, some special cases must be considered. On the one hand, there are boundaries $i = 0$ and $i = n$ where only at most two quadratic interpolants can be checked. On the other hand, the fact that local extrema should have zero slopes should be considered in terms of monotonicity. This is achieved by a Hermite interpolation through the extremum point say (x_i, f_i), its zero derivative value $(x_i, 0)$ and either the left point (x_{i-1}, f_{i-1}) or the right point (x_{i+1}, f_{i+1}). Determining the least curvature magnitude of these two interpolants at x_i determines the approximated slopes Df_i and curvature $D^2 f_i$ at x_i. In this case, of course $Df_i := 0$.

Cases where a neighbouring point (x_{i-1}, f_{i-1}) or (x_{i+1}, f_{i+1}) is a local extremum in the data are treated similarly. At this point it should be noted that if $f_i = f_{i+1}$ for some $i < n$ then both the slope and the curvature at x_i and x_{i+1} must be zero in order for the interpolant to be monotone on $[x_i, x_{i+1}]$, i.e. zero. If this is the case, or if $f_i = f_{i-1}$, then the point (x_i, f_i) is called locally flat on $[x_i, x_{i+1}]$ or on $[x_i, x_{-1}]$ respectively [8].

In the second step a quasi-bisection search is performed to find a solution that is close to the initial estimation and is guaranteed to preserve monotonicity of the data. This procedure uses the fact that the quintic Hermite spline is monotone on the interval $[x_i, x_{i+1}]$ if both the slopes Df_i, Df_{i+1} and curvatures $D^2 f_i$, $D^2 f_{i+1}$ are zero [15]. Decreasing the derivatives eventually leads to a monotone interpolant. Iterative shrinking and growing by a decreasing step size approximates the derivatives until a certain accuracy is achieved. A procedure checks if a certain interval is monotone according to the tight conditions by Ulrich and Watson [15] and the simplified case by Hess and Schmidt [7] (see Algorithm 6). As the implementation of the second step is virtually identical to the original implementation in [10], it will not be described in detail here. A key factor of the method is the estimation of the initial derivatives (see Algorithm 7). In many test cases where the cubic C^2 spline derivatives produce oscillatory behaviour, the quadratic facet model derivatives give a monotone interpolant (e.g. Figure 4.1, Chapter 4).

The implementation of the quadratic facet model and the monotonicity check are described in the following algorithms. The methods have been adapted to the case of equidistant data points with distance $h := x_1 - x_0$. Minor modifications have been made to maintain symmetry.

Algorithm 6. *The pseudocode is a description of the original source code*

from [10] with the documented adaptations.
`is_monotone` $(x_0, x_1, f_0, f_1, Df_0, Df_1, D^2f_0, D^2f_1)$
Returns true if the quintic Hermite spline through
$(x_0, x_1, f_0, f_1, Df_0, Df_1, D^2f_0, D^2f_1)$ from Section 2.3 is monotone on the interval $[x_0, x_1]$.
`if` $f_0 = f_1$ `then`
    `return` $0 = f_0 = f_1 = Df_0 = Df_1 = D^2f_0 = D^2f_1$.
`end if`
mirror values if interval is decreasing:
`if` $f_0 > f_1$ then
    swap (f_0, f_1)
    swap (Df_0, Df_1)
    swap (D^2f_0, D^2f_1)
    multiply Df_0 and Df_1 by -1
`end if`
check if slopes have the right sign:
`if` $\operatorname{sgn}(f_1 - f_0)\, Df_0 < 0$ `or` $\operatorname{sgn}(f_1 - f_0)\, Df_1 < 0$ `then`
    `return false`
`end if`
check monotonicity conditions by Hess and Schmidt in [7] *studied in* [8]:
compute $\alpha, \beta, \delta, \gamma$ from [7]
`if` $\delta < 0$ `return false`
`if` $\alpha < 0$ `return false`
`if` $\gamma < \delta - 2\sqrt{\alpha\delta}$ `return false`
`if` $\beta < \alpha - 2\sqrt{\alpha\delta}$ `return false`
check full monotonicity conditions by Ulrich and Watson in [15] *studied in* [8]:
`if` $\tau_1 \leq 0$ `return false`
compute α, β, γ from the theorems in [15]
`if` $\beta \leq 6$ `then`
    `return` $\min(\alpha, \gamma) > -(\beta + 2)/2$
`end if`
`if` $\min(\alpha, \gamma) \leq -2\sqrt{\beta - 2}$ `then`
    `return` $\min(\alpha, \gamma) > -2\sqrt{\beta - 2}$
`end if`
`return false`

Algorithm 7. *The pseudocode is a description of the original source code from [10] with the documented adaptations.*
`estimate_derivs_equdist` (h, f, n)
where $(0, f_0), (h, f_1), \ldots, (nh, f_n)$ are the data points. Returns the estimated slopes Df_i and curvature $D^2 f_i$ for $0 \leq i \leq n$ by the quadratic facet model with minimal curvature.

```
for i := 0 step 1 to n do
    initialize curvature Df_i := M to be maximally large (e.g. M := 10^54)
    if f_i is locally flat then
        Df_i := 0
        D^2 f_i := 0
    else if f_i is a local extremum then
        p_L := parabola through (ih, f_i), (ih, 0), ((i − 1) h, f_{i−1})
        p_R := parabola through (ih, f_i), (ih, 0), ((i + 1) h, f_{i+1})
        if |p''_L| < |p''_R| then
            Df_i := p'_L(ih)
            D^2 f_i := p''_L
        else
            Df_i := p'_R(ih)
            D^2 f_i := p''_R
        end if
    else
        d := sign of the secant line slope at i
        interpolate left points (if i > 1):
        if f_{i−1} is a local extremum then
            p_L := parabola through ((i − 1) h, f_{i−1}), ((i − 1) h, 0), (ih, f_i)
        else
            p_L := parabola through ((i−2) h, f_{i−2}), ((i−1) h, f_{i−1}), (ih, f_i)
        end if
        check if p_L has the correct slope sign at ih:
        if p'_L(ih) d ≥ 0 then
            Df_i := p'_L(ih)
            D^2 f_i := p''_L(ih)
        end if
        interpolate central points
        (if i > 0  and  i < n − 1
                and  at least one neighbour is not an extremum):
        p := parabola through ((i − 1) h, f_{i−1}), (ih, f_i), ((i + 1) h, f_{i+1})
        The following condition was added to the original code.
```

If i is on the left of the center of f then check for "<",
if i is on the right of the center of f then check for "≤".
This eliminates asymmetric derivative choices in the "=" case.
Check if p has lower curvature than p_L and the correct slope sign:
`if` $i \leq \lfloor n/2 \rfloor - 1$ `then`
`if` $p'(ih)\, d \geq 0$ and $|p''(ih)| < |D^2 f_i|$ `then`
$Df_i := p'(ih)$
$D^2 f_i := p''(ih)$
`end if`
`else`
`if` $p'(ih)\, d \geq 0$ and $|p''(ih)| \leq |D^2 f_i|$ `then`
$Df_i := p'(ih)$
$D^2 f_i := p''(ih)$
`end if`
`end if`
interpolate right points (if $i < n - 2$):
`if` f_{i+1} is a local extremum `then`
$p_R :=$ parabola through (ih, f_i), $((i+1)\, h, f_{i+1})$, $((i+1)\, h, 0)$
`else`
$p_R :=$ parabola through (ih, f_i), $((i+1)\, h, f_{i+1})$, $((i+2)\, h, f_{i+2})$
`end if`
Conditional for symmetry purposes as above.
Check if p_R has lower curvature than p and the correct slope sign
`if` $i \leq \lfloor n/2 \rfloor - 1$ `then`
`if` $p'_R(ih)\, d \geq 0$ and $|p''_R(ih)| < |D^2 f_i|$ `then`
$Df_i := p'_R(ih)$
$D^2 f_i := p''_R(ih)$
`end if`
`else`
`if` $p'_R(ih)\, d \geq 0$ and $|p''_R(ih)| \leq |D^2 f_i|$ `then`
$Df_i := p'_R(ih)$
$D^2 f_i := p''_R(ih)$
`end if`
`end if`
`end if`
finalize estimates:
`if` $D^2 f_i = M$ `then`
$Df_i := 0$
$D^2 f_i := 0$

```
    end if
end for
return (Df, D²f)
```

Note that all algorithms and code used to generate results relating to the MQSI method in this thesis are based on the original paper by Lux et al. [8]. In particular, the original source code of [10] was ported from Fortran 2003 to C, taking into account the modifications mentioned above. When describing the experiments using the adaptation described here, reference is made to the original article [8] or to the source code [10].

3.4 Evaluation of the Approximation Order

As a conclusion of this chapter and as a validation of the implementation, the following experiments aim at estimating the approximation order of the different methods. The experiments also include the cubic C^2 spline from Section 2.1 as a comparison. In order to test the approximation order, as defined in Section 3.2, we use a different definition for this experiment. Consider the sample points $(x_0, f_0), \ldots, (x_n, f_n)$ of a smooth function $f : [a, b] \to \mathbb{R}$, where $a = x_0 < \ldots < x_n = b$. Let

$$\Phi_{\boldsymbol{x},\boldsymbol{f}} \coloneqq \Phi((x_0, f_0), \ldots, (x_n, f_n)) : [a, b] \to \mathbb{R}$$

be the interpolation method such that

$$\Phi_{\boldsymbol{x},\boldsymbol{f}}(x_i) = f_i$$

for $0 \leq i \leq n$. A polynomial q is interpolated exactly by Φ if

$$\Phi_{\boldsymbol{x},\boldsymbol{q}} = q$$

for any $n > \deg(q)$ and

$$\boldsymbol{q} = (q(x_0), \ldots, q(x_n)).$$

Now the approximation order of Φ can be defined as the maximal $p \geq 1$ for which Φ interpolates every polynomial q of degree $\leq p - 1$ exactly. This definition is motivated by the Taylor-series. In the Taylor-expansion of a polynomial of degree $p - 1$, each term with order $\geq p$ vanishes, such that an order p accurate interpolant is exact.

In the experiments, the function $\Phi_{\boldsymbol{x},\boldsymbol{f}}$ is the spline function S from Chapter 2, where the specific piecewise polynomials s_i that the spline is composed of depend on which method is used to estimate the derivatives.

The interpolation methods that we intend to test result in interpolants that are monotone on each interval $[x_i, x_{i+1}]$ for $0 \leq i \leq n$. Thus, it is reasonable to assume that the function we aim to approximate is monotone on each interval as well. Otherwise, the approximation of the monotone methods would be less than or equal to one. To demonstrate this, a parabola that does not include its extreme value would not be interpolated accurately with a monotone method, as depicted in Figure 3.4. Thus, according to the previous definition, the approximation order of any monotone method would be less than two. To simulate the monotonicity, polynomials can be constructed with sample points including the extrema. The test polynomial q of degree p is constructed to have extrema only at integer positions. To do this, we define the derivative of the polynomial

$$q'(x) \coloneqq \prod_{i=1}^{p-1}(x - z_i)$$

for some $z_1, \dots z_{p-1} \in [a, b] \cap \mathbb{Z}$. These are then exactly the zeros of the derivative of the polynomial q. Thus the polynomial q only has extreme values in $[a, b] \cap \mathbb{Z}$.
Now we sample the polynomial q at positions

$$x_i \coloneqq a + ih \quad \text{for } 0 \leq i \leq m\,(b - a),$$

where $h \coloneqq \frac{1}{m}$ for some $m \in \mathbb{N}$. This ensures that all integer positions within the interval $[a, b]$, and therefore all extrema of the polynomial q, are included in the samples.

We test polynomials with a degree of 7 or lower and set the interval boundaries to $a \coloneqq 0$ and $b \coloneqq 5$. This seems adequate for representing the various potential distributions of zeros z_i within the interval $[a, b]$. The zeros can be uniformly distributed throughout the entire interval without any excessive spacing between or at the endpoints. This will help avoid significant fluctuations that are unsuitable for the purpose of plotting. To derive an analytical expression for q, we can begin by expanding q' and then integrating it analytically. The following algorithm outlines the procedure for testing the reproduction order.

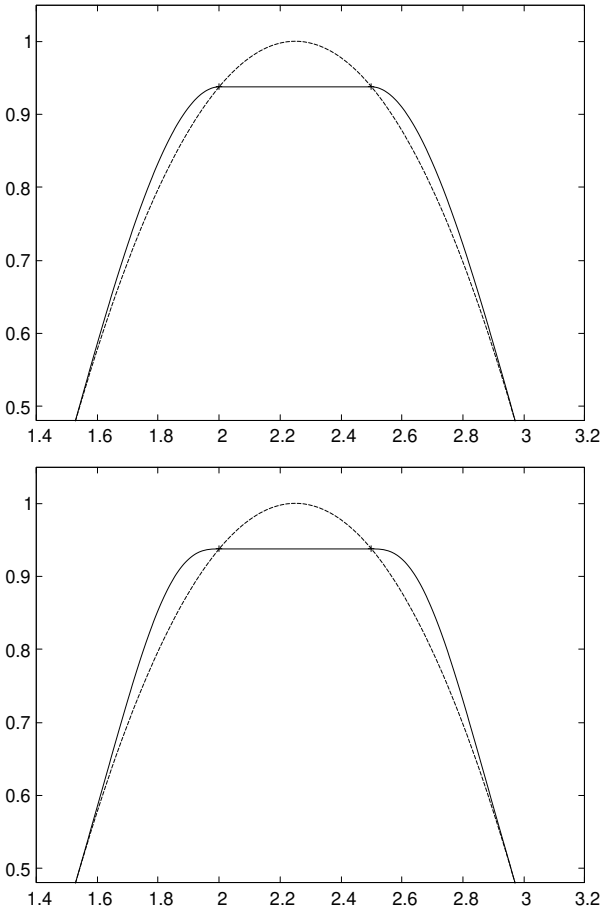


Fig. 3.4. Interpolants of a sampled parabola. **Dashed line:** cubic C^2 spline. **Continuous line:** monotone interpolants with the following monotone spline interpolation methods. **Top:** Fritsch-Carlson method [5] or Extended two-sweep [12] (identical in this case). **Bottom:** MQSI [8].

Algorithm 8. `test_reproduction_order` (d, Φ)
where $d \geq 2$ is the degree of the tested polynomials, Φ is the interpolation method. Returns `true` if Φ interpolates all test polynomials exactly.
$N \coloneqq$ sufficiently large sample number
`for` $k \coloneqq 1$ `step` 1 `to` N `do`
  $z \coloneqq$ random sample set $\{z_1, \ldots, z_{d-1}\} \subseteq \{a, \ldots, b\}$
  $q'(x) \coloneqq \prod_{i=1}^{d-1}(x - z_i)$
  expand q' such that $q'(x) = \sum_{i=0}^{d-1} a_i\, x^i$ for coefficients a_i

```
    integrate q' analytically to obtain q
    y := sample of q at positions a + i · 1/2 for i = 0, ... 2(b − a)
    f := interpolation of y with the given method Φ
    if (f ≠ q)
      return false
    end if
  end for
  return true
```

The condition if $f \neq q$ in Algorithm 8 was tested up to an accuracy of 10^{-14} by evaluating the polynomials at six positions within each interval. This number of evaluations is sufficient for both the cubic and quintic cases to determine the polynomial uniquely, according to the interpolation polynomial [14]. If all six evaluations are accurate on some interval $[x_i, x_{i+1}]$, then $f \equiv q$ on $[x_i, x_{i+1}]$.

The maximal value of p for which `test_reproduction_order` $(p-1, \Phi)$ returns `true` determines the approximation order of Φ according to this test. The Fritsch-Carlson method and the extended two-sweep method both initiate with the cubic C^2 spline interpolant. As the sample points $\{(x_i, q_i)\}$ in the tests always include the extrema of q, the test polynomial q satisfies the required monotonicity condition. The cubic C^2 spline is fourth-order accurate, meaning it reproduces polynomials of degree 3. If q has degree ≤ 3 then the cubic C^2 spline interpolant of the points (x_i, q_i) is equal to q. Thus, it satisfies the monotonicity condition. As a consequence it is not modified by the monotone cubic Hermite spline interpolation methods. This means that the test function should always return `true` for degree $d \leq 3$ if Φ is one of the cubic interpolation methods from Section 3.1, Section 3.2 or the cubic C^2 spline itself. Note that in practice this is generally only the case when using the "not-a-knot" boundary conditions for the cubic C^2 spline.

For the monotone quintic spline interpolation of Section 3.3 we would expect a third-order accurate interpolation because the derivatives are estimated by a quadratic interpolant. So the initial derivatives are second-order accurate, which means that the interpolant is third-order accurate.

The expected results were observed in the tests. All the polynomials of degree 2 that were tested were reproduced exactly by each of the methods.

All cubic methods, not the quintic method, interpolated the tested polynomials of degree ≤ 3 exactly. None of the methods interpolated polynomials of degree ≥ 4 exactly. Figure 3.5 shows plots of some polynomials that have been interpolated with a small error. The error seems to increase as the

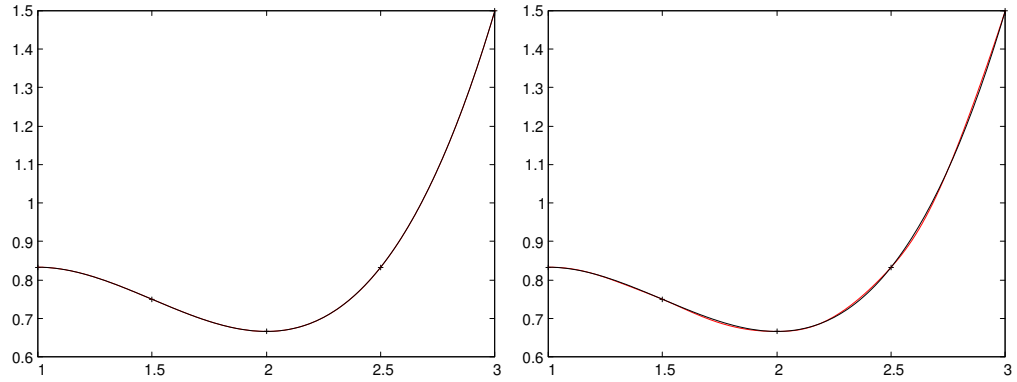
degree of the polynomial increases, as expected. Here, the quintic method with the quadratic derivative estimation [9] leads to more pronounced deviations from the original curve than the cubic methods with the estimates of the cubic derivative. Here, the Extended two-sweep method [12] acts as a representative for the monotone cubic methods, since the differences between the monotone cubic methods are relatively small. Therefore, the cubic C^2 spline and the extended Fritsch-Carlson method [5] are not explicitly shown in the plots.

The results validate the implementation of the cubic C^2 spline and to some extent the monotone cubic interpolation methods. If the initial cubic C^2 spline interpolant preserves monotonicity of the data, then the methods should not change the interpolant, which is consistent with the results.

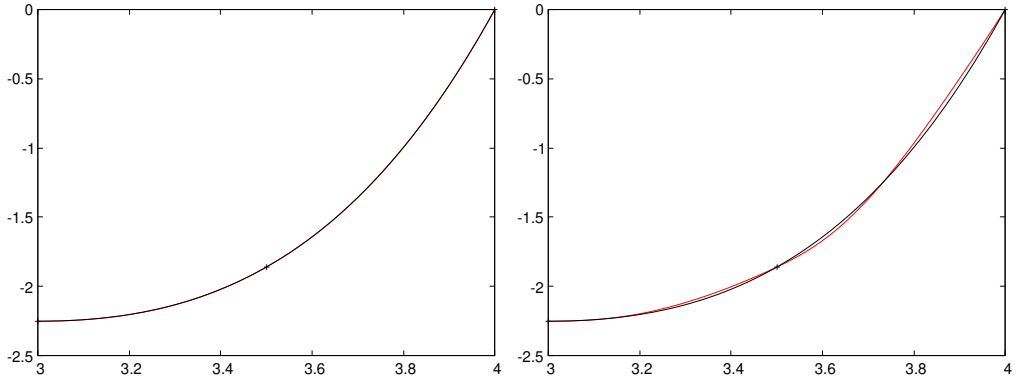
All tested polynomials of degree ≤ 3 have been reproduced by the cubic spline. This also confirms that the boundary conditions have been implemented correctly. The "not-a-knot" boundary conditions assume that the spline is three times differentiable at the second and second-last sample position. This results in the exact reproduction of the polynomials of degree ≤ 3, without any deviations at the boundaries.

The results also show that the quintic method with quadratic estimates is at most third-order accurate. Polynomials of degree 3 could not be reproduced by the method, while polynomials of degree 2 were reproduced exactly.

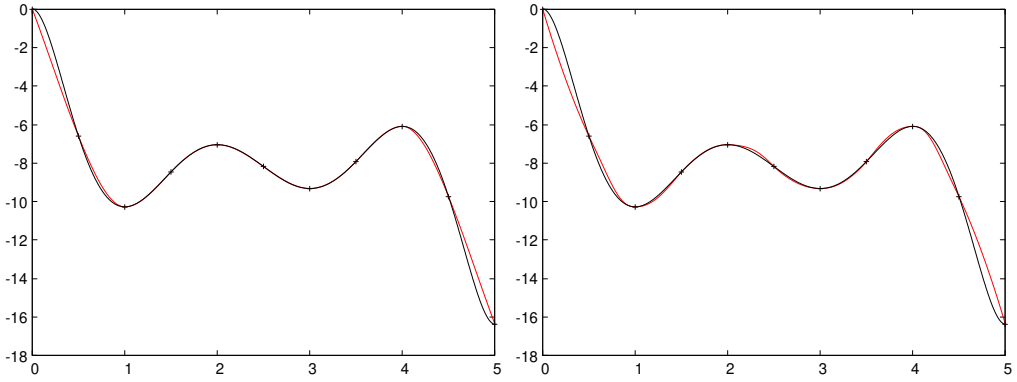
The examples of higher polynomial degrees in Figure 3.5 indicate that the error remains relatively small. This is one of the reasons why splines are so commonly used.



(a) $q'(x) = x - 2$, $\deg(q) = 3$



(b) $q'(x) = (x - 1)(x - 2)$, $\deg(q) = 4$



(c) $q'(x) = x(x - 1)(x - 2)(x - 3)(x - 4)(x - 5)$, $\deg(q) = 7$

Fig. 3.5. Interpolation of the given polynomials q as described above with the respective methods. **Black line:** exact polynomial. **Red line:** interpolated samples. **Left:** Extended two-sweep method [12]. **Right:** MQSI [8].

4 Separable Application to Image Upsampling

In this chapter, we apply the spline interpolation to images. For our use cases it is sufficient to interpolate separately in x and y direction of the image.

To test the image interpolation methods, we can use them to resize images (see Algorithm 12). Given an input image u of dimensions $n \times m$ and a desired output image size of $N \times M$, the image is scaled accordingly, and a resized image of dimensions $N \times M$ is returned. In order to estimate the new image values, a one dimensional piecewise Hermite spline interpolation method is used. Algorithm 9 returns the coefficients $c_{i,k}$ of the piecewise polynomials

$$s_i = \sum_{k=0}^{p-1} c_{i,k} \, (x - x_i)^k$$

from Section 2.1. This allows us to describe the horizontal and vertical image interpolation as follows. Note that for images we can assume that the data is sampled equidistantly.

Algorithm 9. `interpolate_piecewise_Hermite` (f, n, `method`)
where $f_0, \ldots, f_n$ are the equidistantly sampled values to interpolate. Returns the coefficients $c_{i,k}$ for $i = 0; \ldots, n-1$ and $k = 0, \ldots p-1$, where $p-1$ is the degree of the splines in the used `method`. The `method` argument can be an index specifying, which of the previous spline methods to use, and will be omitted in the following.

Algorithm 10. `interpolate_image_horizontal` (u, n, m)
where $u_{i,j}$, $i = 1, \ldots n$, $j = 1, \ldots, m$. Returns the coefficients $c_{j,i}$, $j = 1, \ldots, m$, $i = 1, \ldots, n-1$ of the Hermite polynomials in x direction of the image.

```
for j := 1 step 1 to m do
    v := (u_{1,j}, ..., u_{n,j})
    c_j := interpolate_piecewise_Hermite (v, n)
end for
return c
```

Algorithm 11. `interpolate_image_vertical` (u, n, m)
where $u_{i,j}$, $i = 1, \ldots n$, $j = 1, \ldots, m$ are the original image values including reflecting boundary conditions. Returns the coefficients $c_{i,j}$, $i = 1, \ldots, n$, $j = 0, \ldots, m-1$ of the Hermite polynomials in y direction of the image.

```
for i := 1 step 1 to n do
    v := (u_{i,1}, ..., u_{i,m})
    c_i := interpolate_piecewise_Hermite (v, m)
end for
return c
```

Now we can combine these separate horizontal and vertical interpolation steps to the image resize algorithm. To achieve this we can first interpolate the image in horizontal direction. Then for each image row we can sample the interpolated row in horizontal direction where the number of samples is the new image dimension in x direction, i.e. N. Now we have an image v of size $N \times m$. In the next step we interpolate the image v in y direction. Then by sampling each interpolated column at M equidistant positions we obtain the new image of size $N \times M$.

Algorithm 12. `resize_image` (u, n, m, N, M) where $u_{i,j}$, $i = 1, \ldots n$, $j = 1, \ldots, m$ are the original image values. Returns image w of size $M \times N$ obtained by interpolating the image u separately in x and y direction.

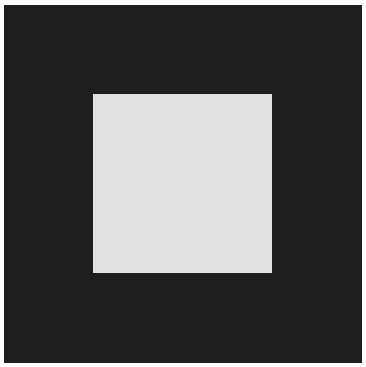
```
1.  c_x := interpolate_image_horizontal (u, n, m)
2.  for j := 1 step 1 to m do
3.      for i := 1 step 1 to N do
4.          v_{i,j} := eval_piecewise_Hermite (1 + ((i-1) n)/(N-1), c_{x,j}, n)
5.      end for
6.  end for
7.  c_y :=interpolate_image_vertical (v, N, m)
8.  for i := 1 step 1 to N do
9.      for j := 1 step 1 to M do
10.         w_{i,j} := eval_piecewise_Hermite (1 + ((j-1) m)/(M-1), c_{y,i}, m)
11.     end for
```

12. `end for`
13. `return` w

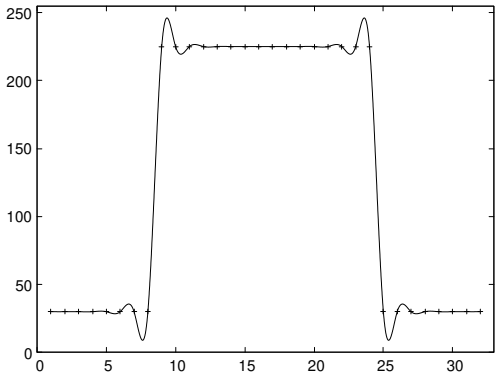
In Line 4 of Algorithm 12 the piecewise Hermite spline is evaluated at $1 + \frac{(i-1)\,n}{N-1}$. This ensures equidistant sampling with symmetric distribution over the image. If $i = 1$ then we sample at position 1, which is the position of the first pixel of the image. If $i = N$ then the the piecewise Hermite spline is evaluated at n which is the position of the last pixel of the image. The expression for the sample position is linear in i, and i is incremented by 1 at each iteration. So the piecewise Hermite spline is sampled equidistantly between 1 and n. Similarly, the sampling in Line 10 of the algorithm samples M equidistant points of each column. Another possibility is to include the boundaries of the image. In this way we can start sampling from 0 to pixel $n + 1$, or from 0 to $m + 1$ in the vertical case. Then the sampling positions must be adjusted accordingly.

Figure 4.1 shows the interpolation of an image row of the synthetic square image with Algorithm 10. Instead of a binary image with values 0 and 255, the grey values have been adjusted to 30 and 225 to account for the overshoot and undershoot of the cubic C^2 spline, which can be observed at the edges of the square. For this test image, all monotone cubic spline interpolants lead to the same results. The derivatives must be zero at any sample position, otherwise monotonicity would be violated. This leaves no room for variation of the cubic Hermite interpolant, assuming we do not introduce new nodes in between the discontinuities. For this reason, the graph in Figure 4.1c obtained by the Fritsch-Carlson method is representative of all monotone cubic Hermite interpolants in this case.

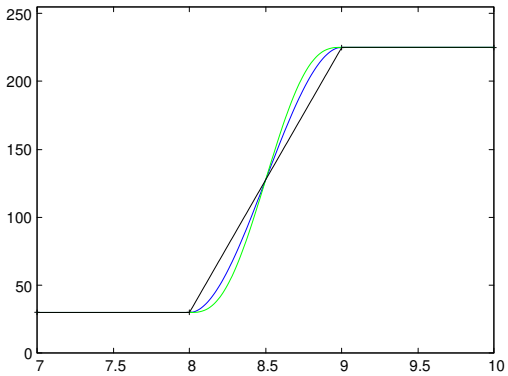
In Figure 4.2 the square image of size 32×32 is resized to size 256×256 with Algorithm 12. Again, we observe perturbations of the cubic C^2 spline at rapidly increasing image values such as the edge of the square. There is a slight increase in sharpness from the linear interpolant to the monotone cubic interpolant. The difference between the cubic methods and the MQSI [8] method is not visible in this case. Also, there is no difference between the monotone cubic methods, which is consistent with our expectations.



(a) original image (32×32)



(b) Cubic C^2 Spline



(c) Zoomed-in view of the graph for monotone interpolants. **Black:** linear spline. **Blue:** Fritsch-Carlson method [5]. **Green:** MQSI [9].

Fig. 4.1. Interpolation of the grey values of row 16 of the given image with the respective methods.

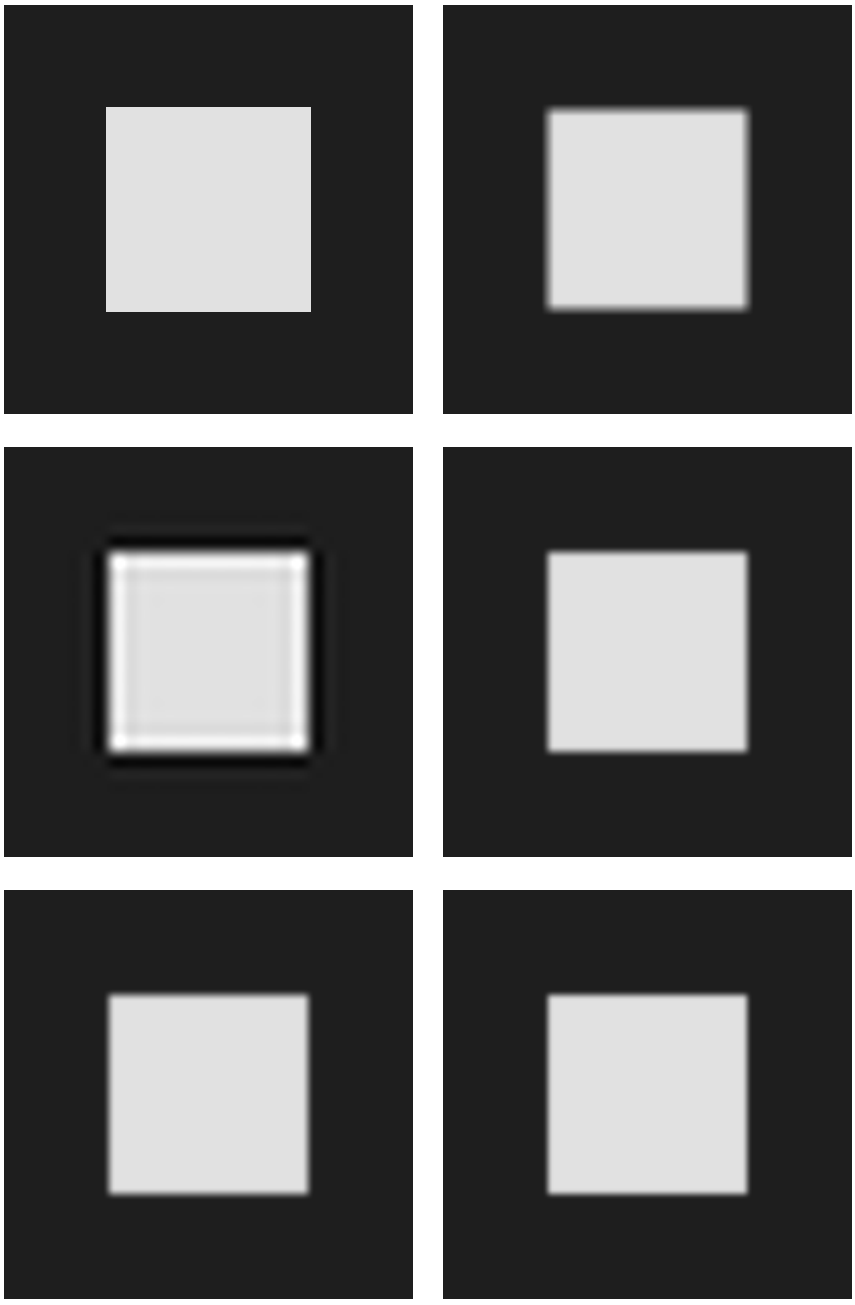


Fig. 4.2. Resized square image (32 × 32) to size (256 × 256) with the respective interpolation method as in Algorithm 12. **Top left:** original image. **Top right:** Linear spline. **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8].

5 Application to Mean Curvature Motion

Mean Curvature Motion (MCM) can be described as an anisotropic second-order PDE [2]

$$u_t = u_{\xi\xi} \tag{5.1}$$

where

$$\xi = \frac{1}{|\nabla u|}(-u_y, u_x)^\top$$

is a normalized vector perpendicular to the gradient of u. It points in the direction of the isophotes which are also called the level lines of the image u. A grey scale image f is regarded as initial condition

$$u(t = 0) = f.$$

The PDE (5.1) can also be written as

$$u_t = |\nabla u| \operatorname{curv}(u) \tag{5.2}$$

where $\operatorname{curv}(u) = \operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right)$ is the (mean) curvature of u. This representation is not used directly here, but it is a basic property that gives the PDE its name mean curvature motion (MCM), even if we are not strictly referring to (5.2)[16]. In the following we are going to focus on mean curvature motion as a second-order anisotropic PDE as in (5.1). Algorithms that implement MCM generally suffer from dissipative artefacts such as blurry edges [13]. In terms of stability, it is desirable for algorithms to satisfy a formal stability criterion, such as the maximum-minimum principle [1]. This also prevents over- and undershoots of the image values during filtering. The following section briefly introduces the δ-scheme [13, 18]. Then we introduce an interpolation-based approach that satisfies the maximum-minimum principle. Other aspects of the method, such as the sharpness and accuracy of the approximation, are then evaluated.

5.1 The Delta-Scheme

The aim of the following subsection is to set the focus on the finite difference approximation in the four principal directions. It is intended as a motivation for the idea of the proposed interpolation method without going into further details of the δ-scheme such as the directional weights. Along with the δ-stencil definition itself, these will be introduced subsequently.

5.1.1 Directional Splitting

In order to obtain a general class of 3×3 stencils with $\mathcal{O}(h^2)$ accuracy we can split the mean curvature motion PDE (5.1) into four weighted 1D diffusions

$$u_t = \sum_{n=0}^{3} w_n \, u_{\boldsymbol{e}_n \boldsymbol{e}_n} \tag{5.3}$$

where e_n are the four principal directions of the underlying grid and w_n are the directional weights. The underlying grid is assumed to have equal size h in x and y direction. The directions $\boldsymbol{e}_n$ are given by the two standard vectors in x and y direction and the normalized diagonal directions. This splitting makes it possible to approximate the directional derivatives $u_{\boldsymbol{e}_n \boldsymbol{e}_n}$ by a standard central difference approximation [13, 18]. For the principle direction vectors

$$\boldsymbol{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{e}_1 = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \boldsymbol{e}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{e}_3 = \tfrac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \tag{5.4}$$

we can approximate

$$(u_{\boldsymbol{e}_0 \boldsymbol{e}_0})_{i,\,j} = \frac{u_{i+1,\,j} - 2u_{i,\,j} + u_{i-1,\,j}}{h^2} + \mathcal{O}(h^2), \tag{5.5a}$$

$$(u_{\boldsymbol{e}_1 \boldsymbol{e}_1})_{i,\,j} = \frac{u_{i+1,\,j+1} - 2u_{i,\,j} + u_{i-1,\,j-1}}{(\sqrt{2}\,h)^2} + \mathcal{O}(h^2), \tag{5.5b}$$

$$(u_{\boldsymbol{e}_2 \boldsymbol{e}_2})_{i,\,j} = \frac{u_{i,\,j+1} - 2u_{i,\,j} + u_{i,\,j-1}}{h^2} + \mathcal{O}(h^2), \tag{5.5c}$$

$$(u_{\boldsymbol{e}_3 \boldsymbol{e}_3})_{i,\,j} = \frac{u_{i-1,\,j+1} - 2u_{i,\,j} + u_{i+1,\,j-1}}{(\sqrt{2}\,h)^2} + \mathcal{O}(h^2). \tag{5.5d}$$

Figure 5.1 shows the pixels that are involved in the approximation of $(u_{\boldsymbol{e}_1, \boldsymbol{e}_1})_{i,\,j}$ for example.

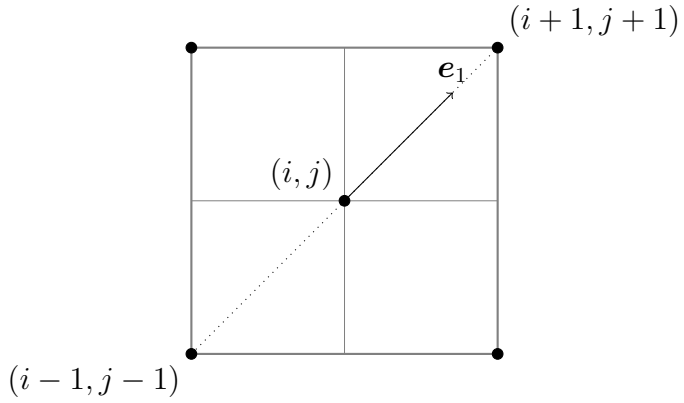


Fig. 5.1. The central point (i, j) and two points $(i-1,\ j-1)$ and $(i+1,\ j+1)$ for the finite difference approximation of $u_{\boldsymbol{e}_1\boldsymbol{e}_1}$

5.1.2 Directional Weights

The details of the directional weights and the resulting delta-stencil with the free parameter δ are important for writing down the δ-stencil and the resulting algorithm. In order to obtain the directional weights w_n we can rewrite (5.1) as

$$\begin{aligned} u_t &= \frac{u_y^2 u_{xx} - 2u_x u_{xy} + u_x^2 u_{yy}}{u_x^2 + u_y^2} \\ &= a\, u_{xx} + 2b\, u_{xy} + c\, u_{yy} \end{aligned}$$

with functions a, b, c that can easily be derived from the previous equation. Comparing this with (5.3) we obtain

$$\begin{aligned} a\, u_{xx} + 2b\, u_{xy} + c\, u_{yy} &= \sum_{n=0}^{3} w_n\, u_{\boldsymbol{e}_n\boldsymbol{e}_n} \\ &= \sum_{n=0}^{3} w_n\, \boldsymbol{e}_n^\top\, \mathbf{Hess}(u)\, \boldsymbol{e}_n \\ &= w_0\, u_{xx} + w_1\, \frac{1}{2}(u_{xx} + 2u_{xy} + u_{yy}) \\ &\quad + w_2\, u_{yy} + w_3\, \frac{1}{2}(u_{xx} - 2u_{xy} + u_{yy}). \end{aligned}$$

Solving the corresponding linear system for the weights w_n leaves one free parameter δ. The weights [13] in dependence of δ are

$$w_0 = a - \delta, \tag{5.6}$$
$$w_1 = \delta + b, \tag{5.7}$$
$$w_2 = c - \delta, \tag{5.8}$$
$$w_3 = \delta - b. \tag{5.9}$$

5.1.3 The Delta-Stencil

With the approximation of

$$u_t = u_{\xi\xi} \approx a\, u_{xx} + 2b\, u_{xy} + c\, u_{yy}$$

with the weights $w_0, \ldots, w_3$ from (5.6) we obtain a second-order accurate 3×3 stencil [13], shown in Figure 5.2 For the initial estimation of the

$$\frac{1}{h^2} \cdot \begin{array}{|c|c|c|} \hline \frac{1}{2}(\delta - b)_{i,j} & (c - \delta)_{i,j} & \frac{1}{2}(\delta + b)_{i,j} \\ \hline (a - \delta)_{i,j} & \begin{array}{c} -2(a - \delta)_{i,j} - (\delta + b)_{i,j} \\ -2(c - \delta)_{i,j} - (\delta - b)_{i,j} \end{array} & (a - \delta)_{i,j} \\ \hline \frac{1}{2}(\delta + b)_{i,j} & (c - \delta)_{i,j} & \frac{1}{2}(\delta - b)_{i,j} \\ \hline \end{array}$$

Fig. 5.2. The δ-stencil, taken from [13]

direction ξ, the Sobel stencils from Figure 5.3 are used, which provide good rotation invariance [13]. The parameter δ can be adapted locally. Following the approach of Weickert et al. [18], we consider

$$\delta = \alpha\,(a + c) + \gamma\,(1 - 2\alpha)\,|b| \tag{5.10}$$

with $\alpha \leq \frac{1}{2}$ and $\gamma \in [-1, 1]$. By selecting $\alpha = \frac{1}{2}$, the maximum sharpness can be achieved, and the use of γ becomes obsolete. Nevertheless, it is possible in this scenario to encounter high-frequency checkerboard-shaped artefacts [13].

$$\partial_x \approx \frac{1}{4}\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \frac{1}{2h}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} = \frac{1}{8h}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\partial_y \approx \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} * \frac{1}{2h}\begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array} = \frac{1}{8h}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Fig. 5.3. Sobel stencils, taken from [13]

5.1.4 The Explicit Delta-Scheme

The time derivative u_t from equation (5.1) is approximated by an explicit forward difference scheme

$$u_t \approx \frac{u^{(k+1)} - u^{(k)}}{\tau} \tag{5.11}$$

where $u^{(k)}$ is the time discretisation of the image u at $t = k\,\tau$. Although the delta scheme does not satisfy a maximum-minimum principle due to possibly negative weights in (5.6), empirical results have shown that the time step size

$$\tau \leq \frac{h^2}{4(1-\alpha)}. \tag{5.12}$$

ensures stability and decreasing variance [13].

Figure 5.4 shows the implementation of the explicit δ-scheme.

```
for (i=1; i<=nx; i++)
 for (j=1; j<=ny; j++)
     {
     /* compute first order derivatives with Sobel stencils */
     fx =   (f[i+1][j+1] - f[i-1][j+1]) / (8.0 * h)
          + (f[i+1][j]   - f[i-1][j]  ) / (4.0 * h)
          + (f[i+1][j-1] - f[i-1][j-1]) / (8.0 * h);
     fy =   (f[i+1][j+1] - f[i+1][j-1]) / (8.0 * h)
          + (f[i]  [j+1] - f[i]  [j-1]) / (4.0 * h)
          + (f[i-1][j+1] - f[i-1][j-1]) / (8.0 * h);

     /* compute (slightly regularised) squared norm of gradient */
     grad_sqr = fx * fx + fy * fy + 1.0e-10;

     /* compute the functions a, b, c, and delta */
     a =   fy * fy / grad_sqr;
     b = - fx * fy / grad_sqr;
     c =   fx * fx / grad_sqr;
     delta = alpha * (a + c) + gamma * (1 - 2.0 * alpha) * fabs(b);

     /* compute second order derivatives in the four principal directions */
     f00 = (f[i+1][j]   - 2.0 * f[i][j] + f[i-1][j])   / (h * h);
     f22 = (f[i][j+1]   - 2.0 * f[i][j] + f[i][j-1])   / (h * h);
     f11 = (f[i+1][j+1] - 2.0 * f[i][j] + f[i-1][j-1]) / (2.0 * h * h);
     f33 = (f[i-1][j+1] - 2.0 * f[i][j] + f[i+1][j-1]) / (2.0 * h * h);

     /* explicit update step for MCM */
     u[i][j]  = f[i][j] + tau * ( (a - delta) * f00 + (delta + b) * f11
                                + (c - delta) * f22 + (delta - b) * f33 );
     }
```

Fig. 5.4. delta-stencil implementation [17] given in [13]

5.2 The Interpolation Scheme

We now present the idea of an explicit interpolation scheme to approximate the mean curvature motion PDE. In order to obtain image values on the grid lines, we interpolate the image separate in x and y direction as depicted in Figure 5.5, similar to the resize algorithm in Chapter 4. This allows us to evaluate neighbouring image values directly in the direction of ξ, and makes a directional splitting as in Subsection 5.1.1 obsolete. Instead, we can use the interpolation to approximate the derivative $u_{\xi\xi}$ directly by a single finite different scheme.

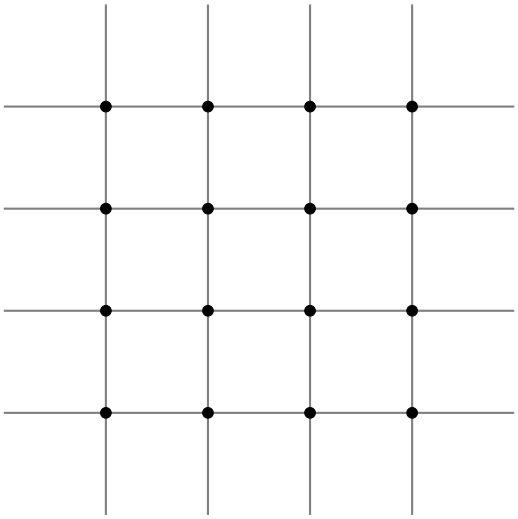


Fig. 5.5. Illustration that we interpolate along the grid lines in x and y direction of the image. Dots represent the pixels of the image.

5.2.1 Approximation by Interpolation

Let $u_{i,j}$ be the discrete image values for $i = 1, \ldots, n$, $j = 1, \ldots, m$. We assume constant pixel size h in both x and y direction. Let Ω be the image domain and Φ be the image interpolation on Ω such that

$$\Phi(u) : \Omega \to \mathbb{R}$$

is a continuous function that interpolates u along the grid lines, i.e.

$$\Phi(u)(i, j) = u_{i,j}$$

for $i = 1, \ldots, n$, $j = 1, \ldots, m$. Note that we normalize the grid size to 1 for this representation as it makes no difference for the image values itself. Now we need to evaluate the interpolated grid lines at the correct positions. For this, let (i, j) now be the pixel at which we want to appxominate $(u_{\xi\xi})_{i,j}$. Then we need the intersection points of the line through the central pixel (i, j) in the direction of ξ with the surrounding grid square. This is shown in Figure 5.6 for the case where the line intersects with the vertical grid lines.

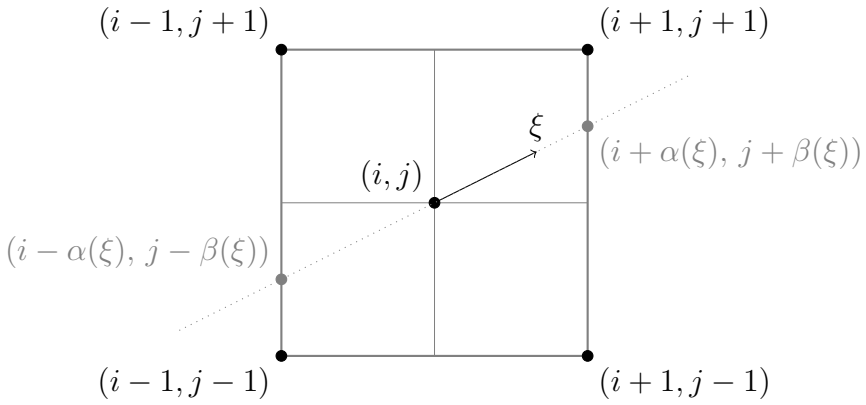


Fig. 5.6. The central point (i, j) and two intersection points $(i-\alpha(\xi),\ j-\beta(\xi))$ and $(i+\alpha(\xi),\ j+\beta(\xi))$ for the finite difference approximation of $u_{\xi\xi}$. In this example, the line intersects with the adjacent vertical grid lines. The case of intersection with the adjacent horizontal grid lines is treated similarly in a case distinction that is reflected in the definition of the α and β functions.

Here, the functions

$$\begin{aligned}
\alpha : \{\xi \in \mathbb{R}^2 \,|\, \|\xi\|_2 = 1\} &\to [-1, 1] \\
(x, y) &\mapsto \begin{cases} 1, & |x| \geq |y| \\ \frac{x}{y}, & |x| < |y| \end{cases} \\
\beta : \{\xi \in \mathbb{R}^2 \,|\, \|\xi\|_2 = 1\} &\to [-1, 1] \\
(x, y) &\mapsto \begin{cases} \frac{y}{x}, & |x| > |y| \\ 1, & |x| \leq |y|. \end{cases}
\end{aligned}$$

are constructed such that for any $\xi \in \mathbb{R}^2$ with $\|\xi\|_2 = 1$ the points $(\alpha(\xi), \beta(\xi))$ and $(-\alpha(\xi), -\beta(\xi))$ are the intersection points of the line $\{\xi\, t \,|\, t \in \mathbb{R}\}$ and the square

$$S^1 := \{(x, y) \in \mathbb{R}^2 \,|\, \max(|x|, |y|) = 1\}. \tag{5.13}$$

Now we can approximate the neighbouring image values in ξ direction by

$$v_{i,j} := \Phi(u)(i - \alpha(\xi), j - \beta(\xi) \tag{5.14}$$

$$w_{i,j} := \Phi(u)(i + \alpha(\xi), j + \beta(\xi) \tag{5.15}$$

and the pixel-dependant grid distance

$$h_{i,j} \coloneqq h\,\sqrt{\alpha(\xi)^2 + \beta(\xi)^2}. \tag{5.16}$$

for $i = 1, \ldots, n$, $j = 1, \ldots, m$. Since $(\alpha(\xi),\, \beta(\xi))$ lies on the square S^1 (5.13) we have

$$h \leq h_{i,j} \leq \sqrt{2}\,h \tag{5.17}$$

for any i, j. With this we approximate $u_{\xi\xi}$ at pixel (i, j) by the final difference scheme

$$(u_{\xi\xi})_{i,j} \approx \frac{v_{i,j} - 2u_{i,j} + w_{i,j}}{h_{i,j}^2} \tag{5.18}$$

including the central pixel (i, j) and the neighbouring interpolated image values in the direction of ξ, as depicted in Figure 5.6. The distance between the central pixel (i, j) and the interpolated pixel $(i + \alpha(\xi), j + \beta(\xi))$ is $h\,\sqrt{(\alpha^2 + \beta^2)}$ which explains the division by $h^2\,(\alpha(\xi)^2 + \beta(\xi)^2)$ in (5.18).

To approximate u_t from equation (5.1), we use an explicit forward difference scheme

$$u_t \approx \frac{u^{(k+1)} - u^{(k)}}{\tau} \tag{5.19}$$

where $u^{(k)}$ is the time discretisation of the image u at $t = k\,\tau$ with the time step size

$$\tau \leq \frac{h^2}{2}. \tag{5.20}$$

The approximation of u_t (5.19) together with the approximation in space with image interpolation (5.18) leads to the explicit interpolation scheme

$$u_{ij}^{(k+1)} = u_{ij}^{(k)} + \tau\,\frac{v_{i,j}^{(k)} - 2u_{i,j}^{(k)} + w_{i,j}^{(k)}}{h_{i,j}^2} \tag{5.21}$$

5.2.2 Maximum-Minimum Principle

The image sequence $\{u^{(k)}\}$ with $u^{(0)} = f$ satisfies the maxium-minimum principle [1] if for any $k \geq 0$ we have

$$\min_{n,m} f_{n,m} \leq u_{i,j}^{(k+1)} \leq \max_{n,m} f_{n,m} \tag{5.22}$$

for any i, j. In the following we write $\max u^{(k)} := \max_{n,m} u^{(k)}_{n,m}$ and $\min u^{(k)} := \min_{n,m} u^{(k)}_{n,m}$. Let $\{u^k\}$ now be the image sequence resulting from the interpolation scheme (5.21). Furthermore, let the interpolation itself now be monotonicity preserving. In particular this implies that

$$v^{(k)}_{i,j} \leq \max u^{(k)},\ w^{(k)}_{i,j} \leq \max u^{(k)} \tag{5.23}$$

$$v^{(k)}_{i,j} \geq \min u^{(k)},\ w^{(k)}_{i,j} \geq \min u^{(k)} \tag{5.24}$$

for any i, j and $k \geq 0$. Thus we have

$$\begin{aligned} u^{(k+1)}_{ij} &= u^{(k)}_{ij} + \tau \frac{v^{(k)}_{i,j} - 2u^{(k)}_{i,j} + w^{(k)}_{i,j}}{h^2_{i,j}} \\ &\overset{(5.23)}{\leq} u^{(k)}_{ij} + \tau \frac{\max u^{(k)} - 2u^{(k)}_{i,j} + \max u^{(k)}}{h^2_{i,j}} \\ &\leq u^{(k)}_{ij} + \tau \frac{2(\max - u^{(k)}_{i,j})}{h^2} \\ &\leq u^{(k)}_{ij} + \max u^{(k)} - u^{(k)}_{i,j}. \end{aligned}$$

In the second last step we used the fact that $h_{i,j} \geq h$ (see 5.17). The last step follows from the time step limitation (5.20) and the resulting estimation against $\tau = \frac{h^2}{2}$. In total we obtain

$$u^{(k+1)}_{ij} \leq \max u^{(k)} \tag{5.25}$$

for any $k \geq 0$. On the other hand we have

$$\begin{aligned} u^{(k+1)}_{ij} &= u^{(k)}_{ij} + \tau \frac{v^{(k)}_{i,j} - 2u^{(k)}_{i,j} + w^{(k)}_{i,j}}{h^2_{i,j}} \\ &\overset{(5.24)}{\geq} u^{(k)}_{ij} + \tau \frac{\min u^{(k)} - 2u^{(k)}_{i,j} + \min u^{(k)}}{h^2_{i,j}} \\ &\overset{(5.17)}{\geq} u^{(k)}_{ij} + \tau \frac{2(\min u^{(k)} - u^{(k)}_{i,j})}{(\sqrt{2}h)^2} \\ &= u^{(k)}_{ij} + \tau \frac{(\min u^{(k)} - u^{(k)}_{i,j})}{h^2} \\ &\overset{(5.20)}{\geq} u^{(k)}_{ij} + \frac{(\min u^{(k)} - u^{(k)}_{i,j})}{2} \\ &= \frac{\min u^{(k)} + u_{ij}}{2}. \end{aligned}$$

In the second last step we used the fact that

$$\frac{(\min u^{(k)} - u_{i,j}^{(k)})}{h^2} \leq 0$$

together with (5.20). With the estimation $u_{i,j}^{(k)} \geq \min u^{(k)}$ we obtain

$$u_{ij}^{(k+1)} \geq \min u^{(k)} \tag{5.26}$$

for any $k \geq 0$. In particular, (5.26) together with (5.25) implies that the maximum-minimum principle is satisfied by the interpolation scheme (5.21) if the interpolation method is monotone.

5.2.3 Order of Accuracy

The order of accuracy in terms of the grid size or pixel distance h depends on several approximation steps. In general we aim at $\mathcal{O}(h^2)$ accuracy in space as it is the case with the δ-scheme [13, 18]. Thus, each approximation in space has to be at least $\mathcal{O}(h^2)$ accurate. It follows directly from equation (5.1) that these requirements are equivalent to approximating $u_{\xi\xi}$ with $\mathcal{O}(h^2)$ accuracy. The standard central difference scheme in (5.18) as in [13] is generally $\mathcal{O}(h^2)$ accurate in space. Assuming the order of accuracy of the interpolated image values $v_{i,j}$ and $w_{i,j}$ from (5.15) is $\mathcal{O}(h^4)$ we obtain

$$\begin{aligned}(u_{\xi\xi})_{ij} &= \frac{(v_{i,j} + \mathcal{O}(h^4)) - 2u_{i,j} + (w_{i,j} + \mathcal{O}(h^4))}{h_{i,j}^2} + \mathcal{O}(h^2) \\ &= \frac{v_{i,j} - 2u_{i,j} + w_{i,j}}{h_{i,j}^2} + \frac{\mathcal{O}(h^4)}{h_{ij}^2} + \mathcal{O}(h^2) \\ &= \frac{v_{i,j} - 2u_{i,j} + w_{i,j}}{h_{i,j}^2} + \mathcal{O}(h^2).\end{aligned}$$

In the last step we used the fact that $h_{i,j}^2$ is of $\mathcal{O}(h^2)$ which follows from (5.17). This demonstrates that the approximation of $u_{\xi\xi}$ in (5.18) is second-order accurate in space if the interpolated image values are fourth-order accurate in space. Similarly, if the interpolation method is third order accurate, then the resulting interpolation scheme is first order accurate. The linear spline interpolation leads to inconsistent approximations.

The table in Figure 5.8 illustrates the findings w.r.t. maximum-minimum principle and approximation order for the different methods. In the case of

the interpolation-scheme, they are a consequence of the monotonicity properties and approximation order of the interpolation methods, as shown in Figure 5.7.

Interpolation method	monotone	approx. order	spline order	smoothness degree
Linear spline	yes	2	2	0
Cubic spline	no	4	4	2
Fritsch-Carlson method [5]	yes	3	4	1
Ext. Two-sweep [12]	yes	4	4	1
MQSI [8]	yes	3	6	2

Fig. 5.7. Basic properties of the different spline methods used for the MCM-interpolation-scheme.

MCM-scheme	maximum-minimum principle	approximation order
Delta-scheme [17]	no	2
Linear spline	yes	0
Cubic spline	no	2
Fritsch-Carlson method [5]	yes	1
Ext. Two-sweep [12]	yes	2
MQSI [8]	yes	1

Fig. 5.8. Properties of the delta-scheme compared with the interpolation-scheme with given interpolation methods.

5.2.4 Implementation

The following pseudocode outlines the implementation of the interpolation based MCM-scheme with an explicit time discretisation. It is derived from the original delta scheme implementation [17] of Subsection 5.1.4 and adapted to include the interpolation scheme in the approximation of $u_{\xi\xi}$. In the approximation of ξ itself, the Sobel stencil from Figure 5.3 is used again for good rotation invariance.

Algorithm 13. *The peudocode is based on the original code from [17], adapted to include interpolation scheme*

`mcm_step` (u, n, m, τ, h, Φ)
where u is the input image of size $n \times m$, τ is the time step size and h the pixel distance. Φ is the 1D-interpolation method that is being used. Performs one iteration of the MCM interpolation scheme (5.21) on the image u.
$f \coloneqq$ copy of the image u
$\Phi_u \coloneqq$ `interpolate_image` (f, n, m, Φ)
`for` $i \coloneqq 1$ `step` 1 `to` n `do`
    `for` $j \coloneqq 1$ `step` 1 `to` m `do`
        Approximate f_x and f_y by Sobel derivatives as in Figure 5.4
        `if` $(f_x^2 + fy^2 = 0)$ `continue`
        $\xi \coloneqq \frac{1}{|\nabla f|}(-f_x, f_y)$
        $v_{i,j} \coloneqq \Phi_u(i - \alpha(\xi), j - \beta(\xi))$
        $w_{i,j} \coloneqq \Phi_u(i + \alpha(\xi), j + \beta(\xi))$
        $h_{i,j} \coloneqq h\sqrt{\alpha(\xi)^2 + \beta(\xi)^2}$
        $u_{i,j} \coloneqq f_{i,j} + \tau\,(v_{i,j} - 2u_{i,j} + w_{i,j})/h_{i,j}$
    `end for`
`end for`

Note that all the algorithms and code used in this thesis to generate results relating to mean curvature motion (MCM) are based on the documented adaptation of the original code from [17].

5.2.5 Running Time Estimations

The running time of the interpolation-based methods should be of the same order as the delta-scheme in terms of image size. In each iteration of the method, computations are made that are of the order $n\,m$ where n and m are the image dimensions. This is true for both the delta-scheme and the interpolation-based approach. However, the interpolation scheme should be slower because the image interpolation is computationally more expensive than the computation of the δ-stencil and the resulting derivatives.

6 Experimental Results

Mean curvature motion can be interpreted as morphological evolution that smoothes the image anisotropically. During this smoothing process, the shape of the image is further simplified and becomes more disk-shaped until there is only a single point in the centre [16]. When interpreted as a morphological filter, it theoretically works on binary images. However, during the filtering process, we interpret the image as a grey scale image. The results will therefore contain grey values from 0 to 255. This allows us to visualize disspative artefacts that occur during filtering.

The tested methods result from combining the interpolation scheme with one of the monotone spline interpolation methods, the cubic C^2 spline or the linear spline. They are compared with each other and with the δ-scheme.

6.1 Running Time Experiment

In Figure 6.1, a comparison is made between different interpolation methods combined with the interpolation scheme in Algorithm 13, with regards to the mean run time per iteration for various image sizes. The results are consistent with the expectations in Subsection 5.2.5. The delta-scheme is faster than any interpolation-based approach. However, all of the methods show a comparable increase in running time as image dimensions increase. The monotone cubic methods are slightly slower than the cubic spline, since they include the cubic spline as initial approximations. The extended two-sweep method [12] is slightly slower than the Fritsch-Carlson method [5], because the projection of the derivatives is more elaborate. Finally, the quintic method is the most expensive computationally. This is probably mostly due to the fact that the quintic spline has a higher order than the cubic spline.

n	δ-scheme [17],	CS	FC	ETS	MQSI
64	0.043	0.205	0.210	0.228	0.496
128	0.158	0.647	0.766	0.845	1.493
192	0.362	1.471	1.799	2.005	2.686
256	0.689	2.552	3.094	3.301	3.714
320	1.062	3.989	4.855	4.988	5.627
384	1.516	5.900	7.118	7.349	7.859
448	2.093	8.362	9.694	9.91	10.14
512	2.763	10.439	12.45	12.71	13.61

Fig. 6.1. Average running time per iteration in milliseconds on a test image of $n \times n$ pixels. Interpolation schemes with the following interpolation methods. **CS:** cubic C^2 spline. **FC:** Fritsch-Carlson method [5], **ETS:** Extended two-sweep [12], MQSI [12].

6.2 Shape Simplification

First, we compare the results at different time steps in the filtering process until the shape largely disappears. The shape simplification of the different methods is shown in Figure 6.2 to Figure 6.7. The original image, the time step values and the parameters of the δ-scheme are taken from [13]. All methods that use a monotone interpolation method satisfy the maximum-minimum principle in all the tests. The greyscale range of the filtered image using monotone cubic and monotone quintic interpolation does not exceed that of the original image. This is not the case for the cubic C^2 spline interpolation method.

All cubic methods lead to results that are visually quite similar to the results of the δ-scheme. There are slight differences between all these different methods at each time step of the evolution, although they may appear to be identical.

The monotone quintic spline method from [8], on the other hand, produces much sharper results in this test.

An important question is what may cause this drastic increase in sharpness. Experiments have shown that the sharpness increase could not reproduced in this amount with the cubic splines using the quadratic facet model [6], but slight increases could be noticed. Using the quintic spline with derivatives from the cubic C^2 spline did not improve sharpness.

Assuming that the quadratic facet model plays a significant role in the

sharpness increase, the question is why the cubic methods do not profit from it in a similar way. Here it is important to notice that the quadratic facet model includes five pixels in each derivative estimation. Even though the derivative is a quadratic estimate that includes three points, in general five points are involved in deciding which facet is being picked. The resulting minimum curvature then determines the second derivatives of the quintic spline.

The cubic spline on the other hand is already specified by the first derivatives, so its curvature cannot be explicitly optimized. Another important aspect to notice is that the monotonicity preservation leads to zero derivatives at local extrema. Since the cubic Hermite spline is only defined by the first derivatives, this leaves no room for estimating the curvature at local extrema. In the quintic case on the other hand, we estimate the second derivatives as well. At an extremum, the quadratic facet model approximates the curvature by a local quadratic fit that includes the zero derivative which could lead to a better approximation of the local curvature.

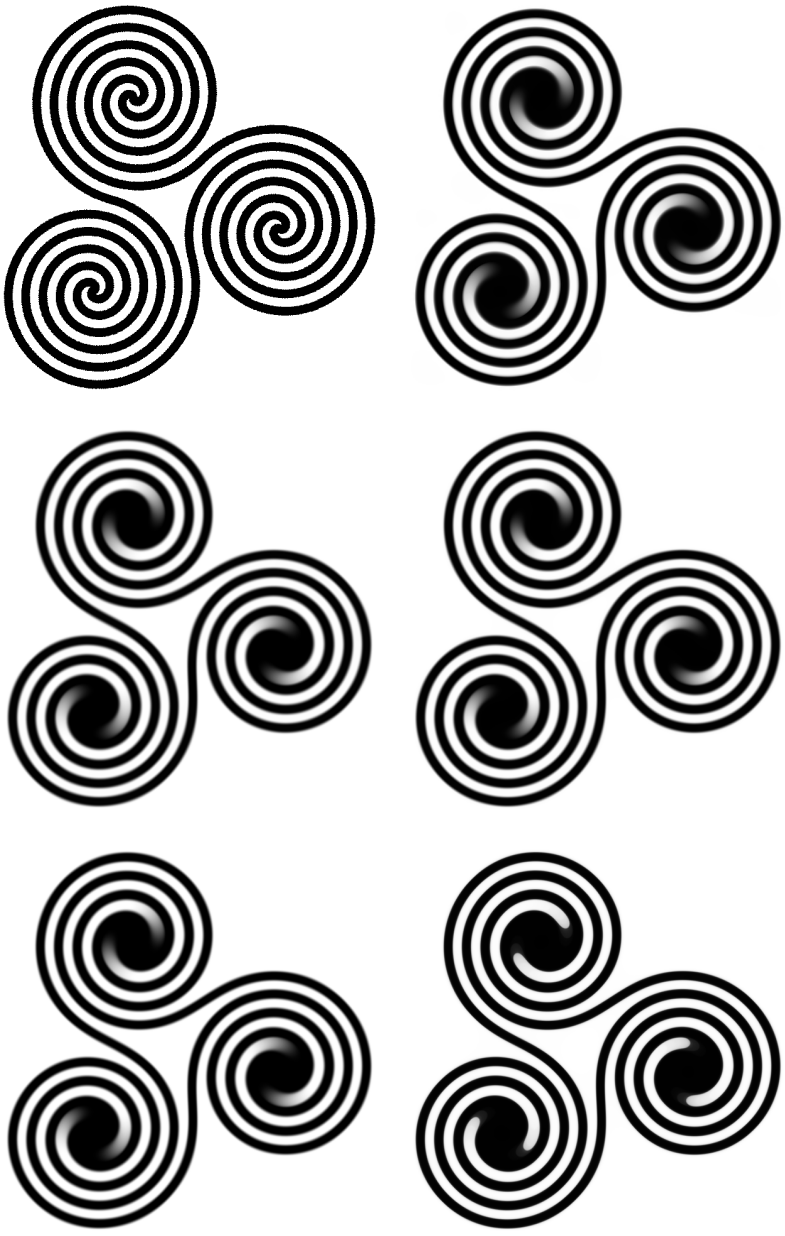


Fig. 6.2. MCM-filtering [17] at $t = 900$. **Top left:** original image. **Top right:** δ-scheme [17] ($\alpha = 0.5$). Interpolation schemes with: **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

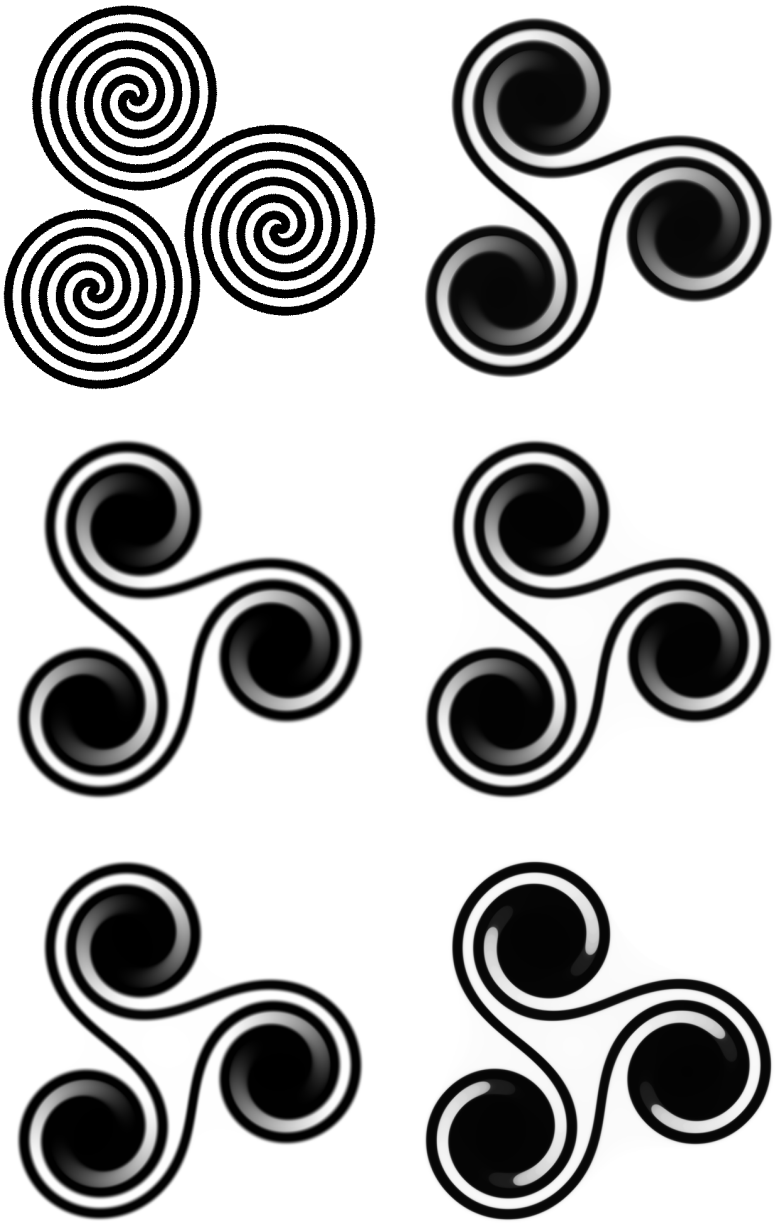


Fig. 6.3. MCM-filtering [17] at $t = 3100$. **Top left:** original image. **Top right:** δ-scheme [17] ($\alpha = 0.5$). Interpolation schemes with: **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

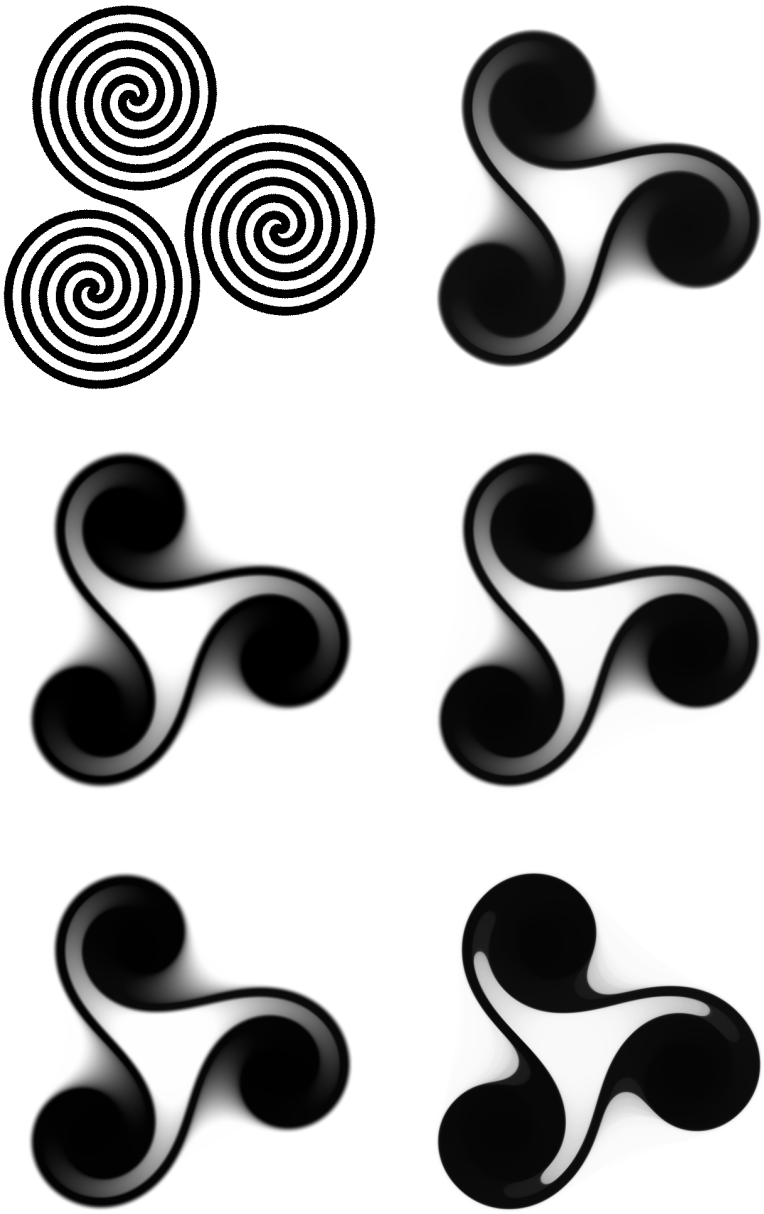


Fig. 6.4. MCM-filtering [17] at $t = 5300$. **Top left:** original image. **Top right:** δ-scheme [17] ($\alpha = 0.5$). Interpolation schemes with: **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

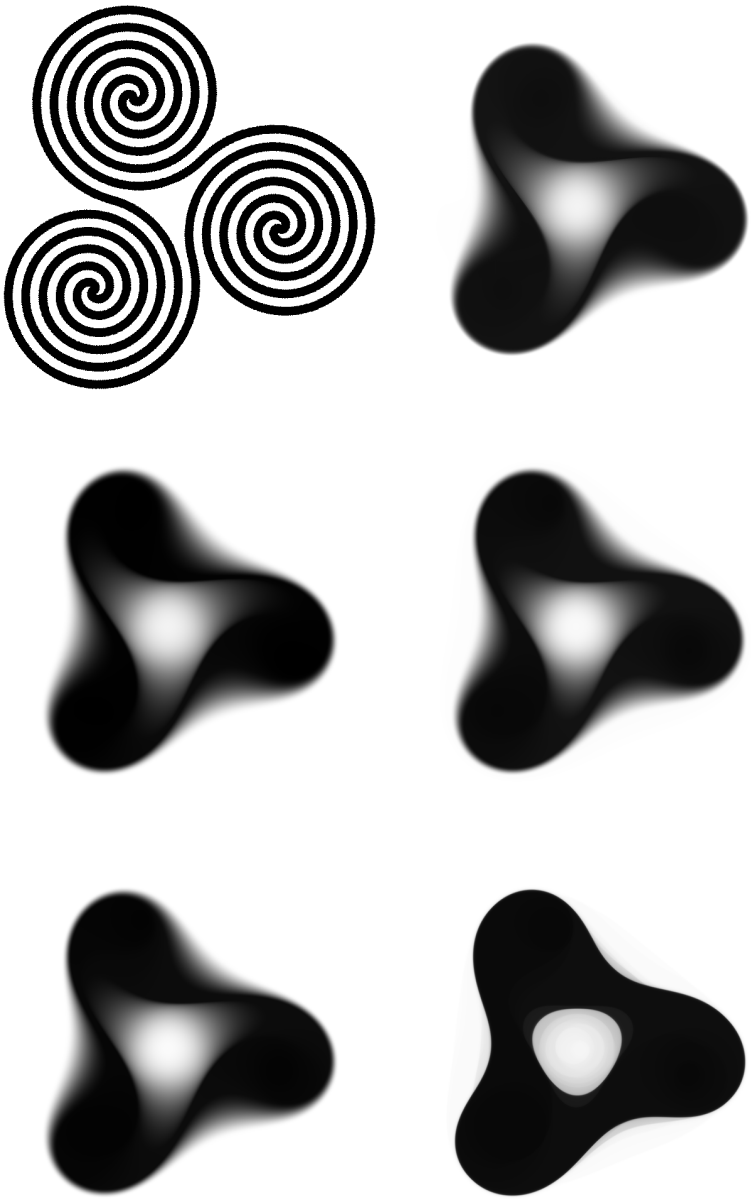


Fig. 6.5. MCM-filtering [17] at $t = 7800$. **Top left:** original image. **Top right:** δ-scheme [17] ($\alpha = 0.5$). Interpolation schemes with: **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

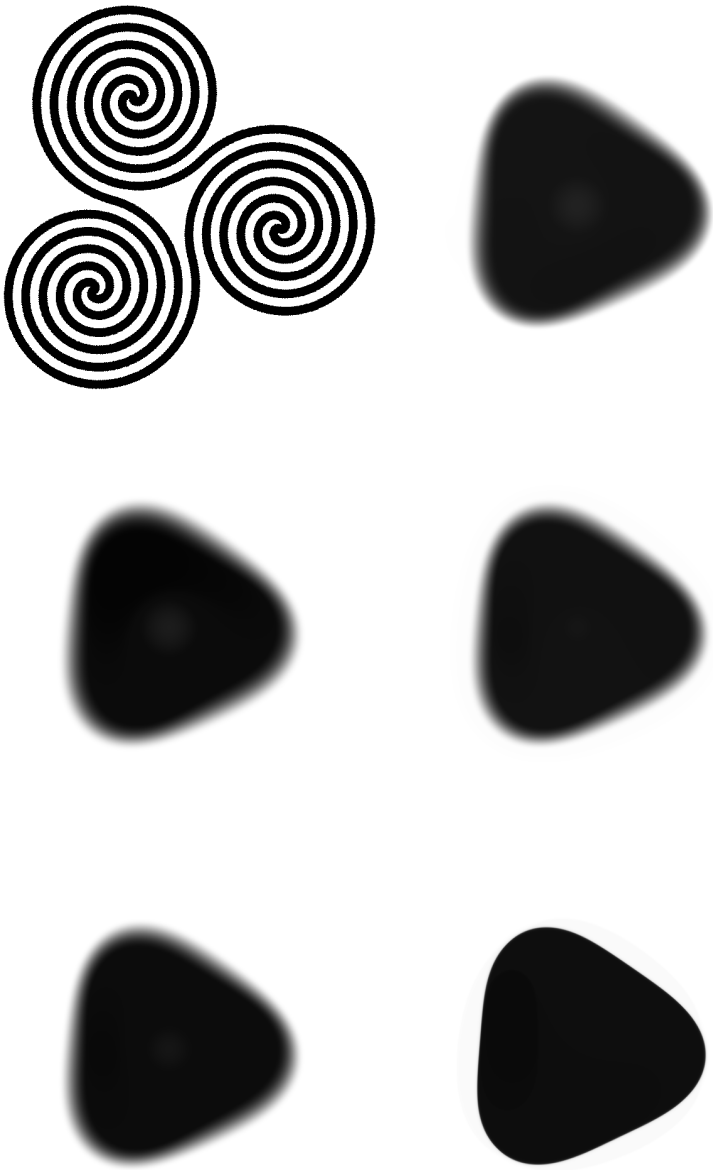


Fig. 6.6. MCM-filtering [17] at $t = 13000$. **Top left:** original image. **Top right:** δ-scheme [17] ($\alpha = 0.5$). Interpolation schemes with: **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

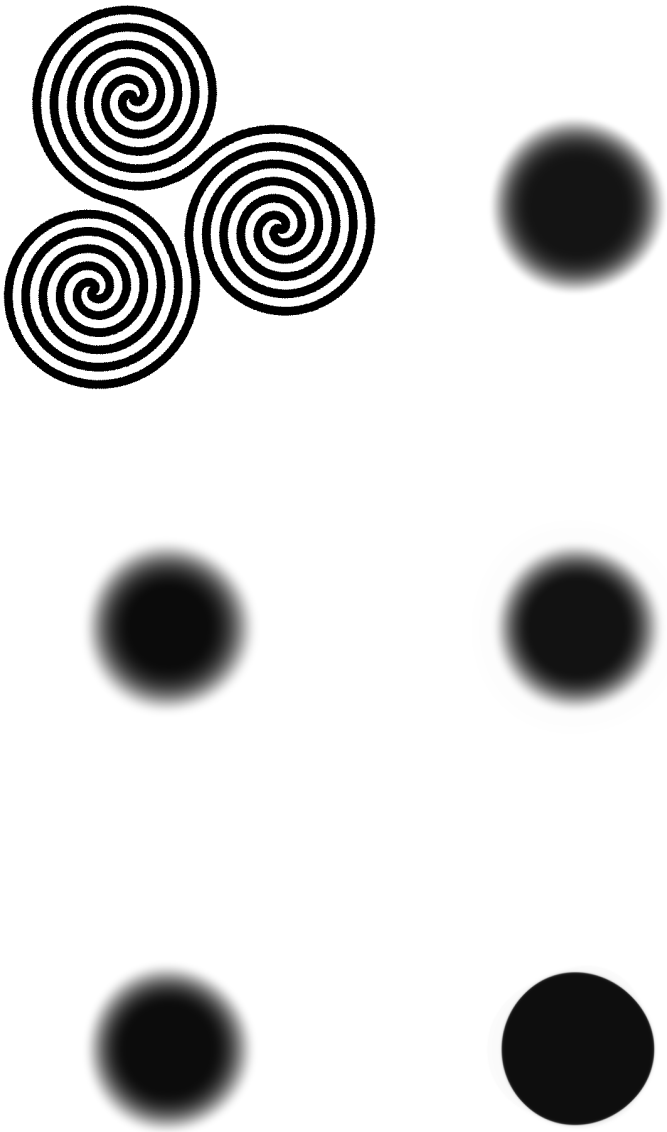


Fig. 6.7. MCM-filtering [17] at $t = 22000$. **Top left:** original image. **Top right:** δ-scheme [17] ($\alpha = 0.5$). Interpolation schemes with: **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

6.3 Rotational Invariance

In order to evaluate symmetry aspects of the methods, we test them on a rotationally invariant image input u depicted in Figure 6.8. The grey

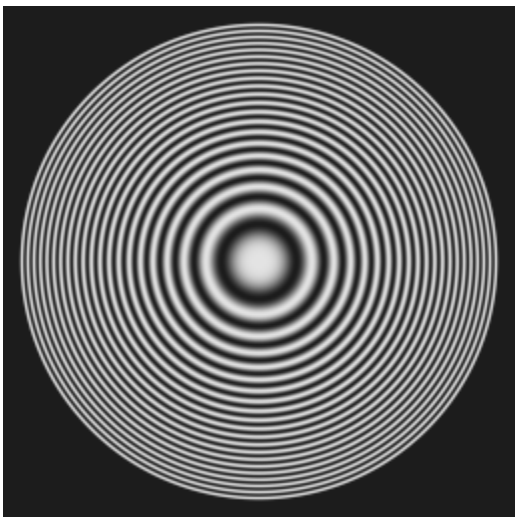


Fig. 6.8. Rotationally invariant synthetic image with rapidly varying image values.

values of u oscillate rapidly, depending on the distance from the centre of the image. In polar coordinates (r, φ) with the centre of the image as origin, the idealized version of the image could be described as a smooth function of the radius

$$u(r, \varphi) = f(r)$$

where f is an oscillating function which increases its frequency with increasing distance to the centre, such as

$$f(r) = A \cos(B\, r^2) + C$$

for parameters A, B and B such that the image values are in the range $[0, 255]$.

First we test the δ-scheme with different parameters. Results for $\gamma = 1$ are shown in Figure 6.9 for different α-values. There are several aspects of the quality of the results than can be examined here. First we notice that the frequency of the oscillation in the original image increases with increasing radius. This leads to more artefacts and fluctuations with increasing radius.

So the first quality in the assessment of the result is at which distance to the centre the images start to produce noticeable deviations from a constant function of the angle φ.

All methods produce a star like pattern at some point. Another aspect is how pronounced the artefact become at the edges. This can be observed with the δ-stencil as well as with the interpolation based methods. For the δ-scheme there is a checkerboard pattern artefact that can be observed in Figure 6.9 along the diagonals. It is most pronounced for $\alpha = 0.5$ [18], which can be observed.

Figure 6.10 shows the results for the interpolation-based methods. All tested methods produce results that are axially symmetric for both the x and the y axis of the image with the centre of the image as origin. This validates that the methods preserve symmetry of the image along the x and the y axis. In this figure, the results are compared to the best results of the delta-stencil in terms of rotation invariance ($\alpha = 0.5$) and low amount of oscillatory checkerboard-pattern like artifacts ($\alpha = 0, \gamma = 1$). The considered interpolation-based methods lead to satisfactory results for the rotation invariance. Here, only the cubic C^2 spline interpolation method can compete with the δ-scheme. The image starts varying with the angle φ quite late in terms of the radius r compared to other interpolation-based methods. Artefacts at the edges of the image can be observed with any interpolation method. Here, the quintic method with the quadratic derivative estimation at the bottom right of Figure 6.10 leads to disk-shaped artifacts and wave-like patterns at the edges, which do not occur as much in the other images.

On the other hand, the interpolation methods do not have the checkerboard like oscillatory artefacts which is an important advantage over the delta-stencil.

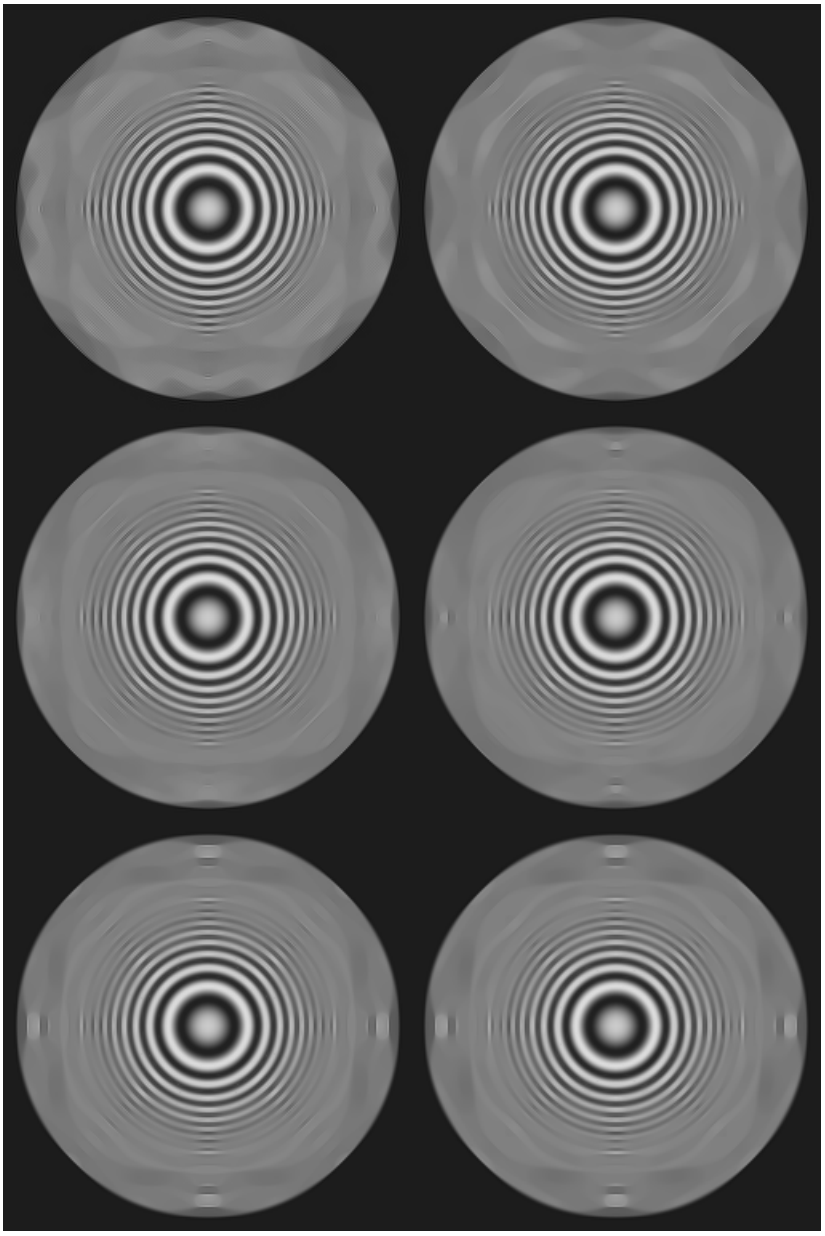


Fig. 6.9. MCM-filtering [17] with the δ-scheme [17] at $t = 40$ with $\gamma = 1$. **Top left:** $\alpha = 0.5$. **Top right:** $\alpha = 0.4$. **Middle left:** $\alpha = 0.3$. **Middle right:** $\alpha = 0.2$. **Bottom left:** $\alpha = 0.1$. **Bottom right:** $\alpha = 0$.

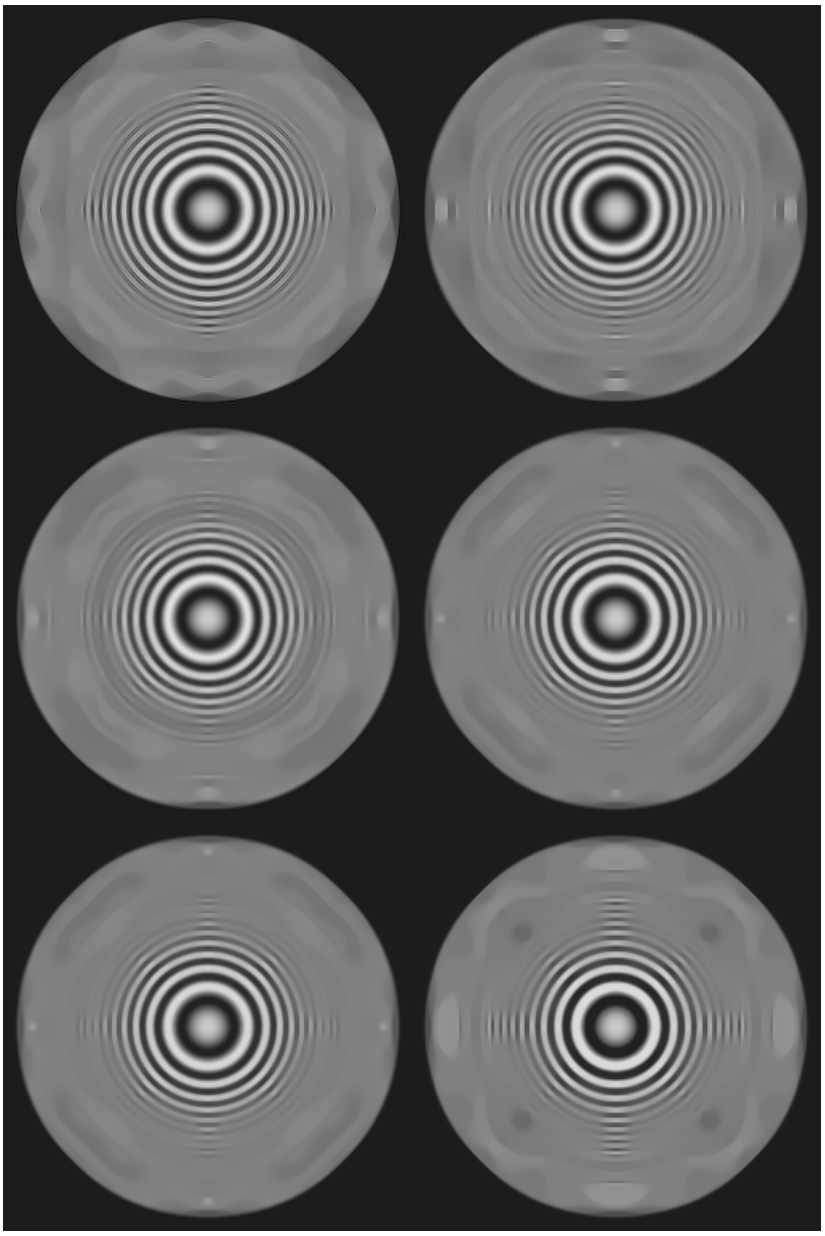


Fig. 6.10. MCM-filtering [17] $t = 40$. **Top left:** δ-scheme [17] ($\alpha = 0.5$). **Top right:** δ-scheme [17] ($\alpha = 0$, $\gamma = 1$). Interpolation schemes with: **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

6.4 Disk Shrinkage

During the shape simplification of the mean curvature motion PDE, the radius of the shapes shrink by a certain rate. For a disk of radius σ this rate is proportional to the area of the disk. If $A(0) = \pi\sigma^2$ is the initial area of the disk, then

$$T = \frac{1}{2}\sigma^2 \tag{6.1}$$

is the extinction time such that $A(T) = 0$ and $A(t) > 0$ for $t \in [0, T][16]$.

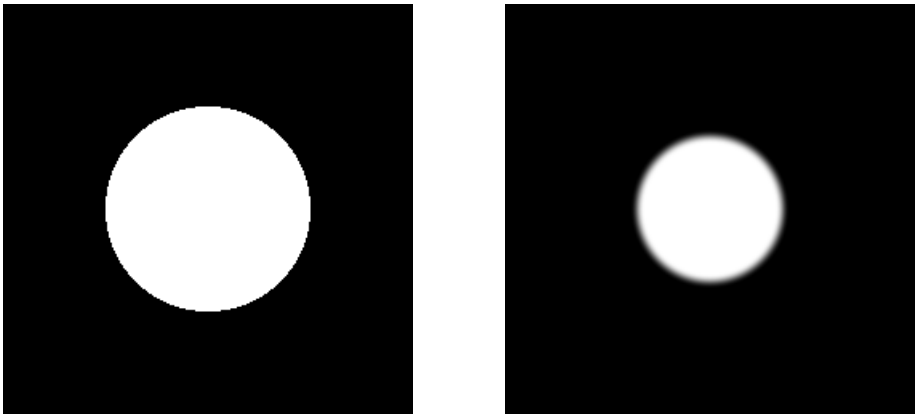


Fig. 6.11. MCM-filtering on a disk shaped image with the interpolation-scheme using the extended two-sweep method [12]. **Left:** original image of size 256×256 and disk radius 64. **Right:** result after $t = 1024$ which is half the extinction time in this case.

6.4.1 Basic Approximated Extinction Times

In order to estimate the extinction time for a given method, the initial disk image is shrunken by the corresponding MCM-scheme, as depicted in Figure 6.11, until the discretised version of the resulting image is effectively zero. The extinction time T_{approx} can then directly be approximated by

$$T_{\text{approx}} = k_{\max}\,\tau \tag{6.2}$$

where $k_{\max}$ is the maximal number of iterations until the image vanishes and τ is the time step size. The extinction times are measured for different

disk sizes in order to evaluate the behaviour of the methods for different resolutions. To simulate shrinking grid sizes h, we increase the disk sizes and therefore increase the resolution. During this, the actual grid size h in the algorithm remains constant. Doubling the resolution of the disk for example is equivalent to halving the sample step size h.

Since all tested methods except the interpolation method resulting from the linear spline should are at least $\mathcal{O}(h)$ accurate in space and first order accurate in time, we would expect that the errors of the approximated extinction times converge towards zero with increasing resolution. The results from the first test are in line with this assumption. As expected, the linear spline produces inconsistent results and will not be used in the following discussion.

Figure 6.12 shows the relative errors of the approximated extinction times for different methods and different initial disk sizes. For the delta scheme, the standard discretisation $\alpha = 0$ and $\gamma = 0$ is used, as this seems to give the most accurate results.

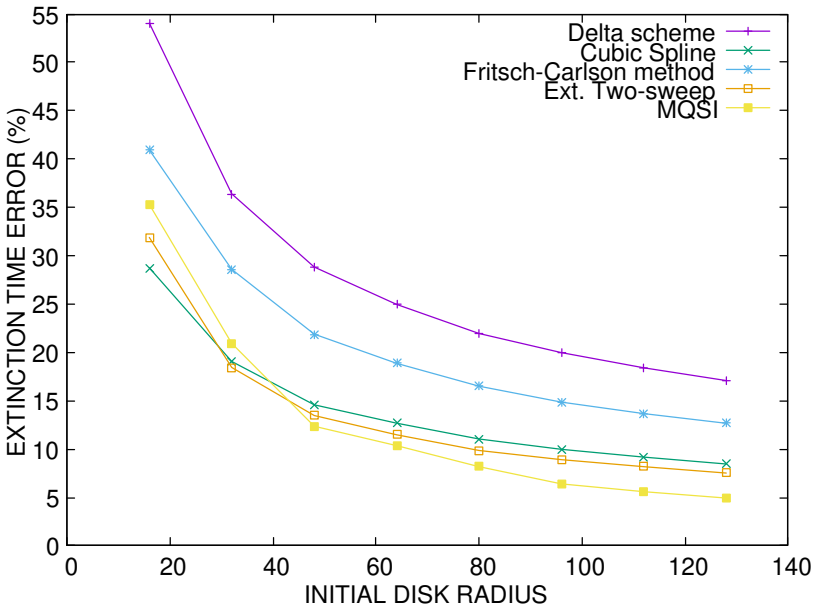


Fig. 6.12. Relative error of the approximated extinction time in dependence of the radius of the initial disk to the analytical solution. Delta scheme [17] parameters: $\alpha = 0$, $\gamma = 0$. No thresholding.

6.4.2 Approximated Extinction Times With Line Fitting

At the beginning of this section we saw that the area A of the circular disk shrinks linearly in time. Together with the initial value $A(0) = \pi\sigma^2$ and the extinction time $A(T) = 0$ we can write down an exact analytical solution for the shrinking area. We obtain

$$A(t) = \pi\sigma^2 - 2\pi t \tag{6.3}$$

for $t \in [0, T]$. On the other hand, we can approximate the area A_i of the shrinking disk at any given time t_i in the image evolution by

$$A_i \coloneqq A_{\text{approx}}(t_i) \coloneqq \frac{1}{255} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{i,j}. \tag{6.4}$$

This approximation also takes into account the blurred edges of the disc, as can be observed in Figure 6.11, which are given appropriate weight through their grey value.
To compare the approximated solution to the analytic solution, we first compute the sample values A_i for equidistant sample time steps t_i for $i = 1, \ldots, N$ and an appropriate sample size N. Then we calculate a line of best fit $A_{\text{fit}}(t)$ through the sample points A_i and compare it to the analytic solution $A(t)$. The best fit of the line A_{fit} is determined by minimizing the mean squared error

$$E = \sum_{i=1}^{N} \left| \frac{1}{t_i} (A_i - A_{\text{fit}}(t_i)) \right|^2. \tag{6.5}$$

Note, that the weights $\frac{1}{t_i}$ are used in order optimize the slope of the line A_{fit}. We know that the point $(t = 0, A_0)$ is exact, so with increasing time we allow an error that increases linear in time. With the exact value $A_0 = \pi\sigma^2$ we can write

$$A_{\text{fit}}(t) = \pi\sigma^2 + c_1\, t \tag{6.6}$$

for the slope c_1 that we want to estimate, which leads to

$$E = \sum_{i=1}^{N} \left| \frac{1}{t_i} (A(t_i) - \pi\sigma^2 - c_1\, t_i) \right|^2. \tag{6.7}$$

Minimizing E with respect to the slope c as variable leads to

$$\begin{aligned} 0 = \frac{\partial E}{\partial c} &= \sum_{i=1}^{N} \frac{2}{t_i}(A(t_i) - \pi\sigma^2 - c\,t_i)(-1) \\ &= 2\sum_{i=1}^{N} \left(\frac{1}{t_i}(-A(t_i) + \pi\sigma^2) + c \right) \end{aligned}$$

which results in the minimizing slope

$$c_1 = \frac{1}{N} \sum_{i=1}^{n} \frac{1}{t_i} \left(A(t_i) - \pi\sigma^2 \right). \tag{6.8}$$

Since

$$\frac{\partial^2 E}{\partial c^2} = 2N > 0,$$

this is indeed a minimum. Figure 6.13 to Figure 6.15 show an example fit for an initial disk of radius $\sigma = 64$. The results are consistent with the findings in Subsection 5.2.3. The linear spline should lead to an inconsistent MCM-scheme. Figure 6.13b shows that the shrinking area of the disk does not represent a line. The fact that the line fitting leads to a quite accurate result is due to the initial underestimation of the disk area which is balanced by the drastic overestimation at larger time steps. All other methods lead to shrinking area approximations that represent the shape of a line. This is consistent with the findings that all interpolation schemes are at least first order accurate approximations of MCM. The cubic C^2 spline leads to the most accurate results, followed by the monotone cubic interpolation methods and the MQSI [8] method. The delta-scheme [16] leads to similar results as the interpolation schemes (except for the linear spline). Since we know that the delta-stencil is second order accurate [13, 18], this also validates the results for the interpolation schemes.

Similar to the experiment in Subsection 6.4.1 we now consider different resolutions. The relative error of the extinction time is equal to the relative error of the approximated slope c_1 to the analytical slope which is -2π. Hence

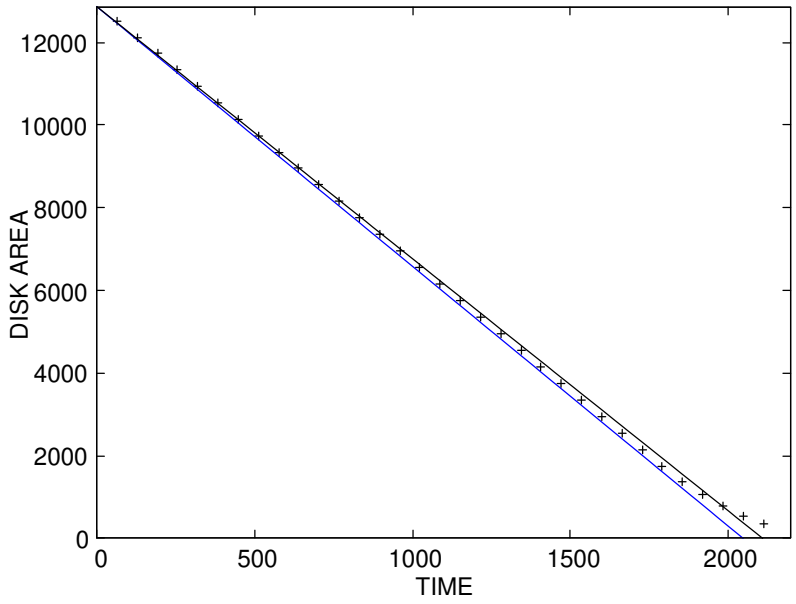
$$\frac{|T - T_{\text{approx}}|}{T} = \frac{|2\pi + c_1|}{2\pi} \tag{6.9}$$

In Figure 6.16 this error is shown for different initial disk sizes.

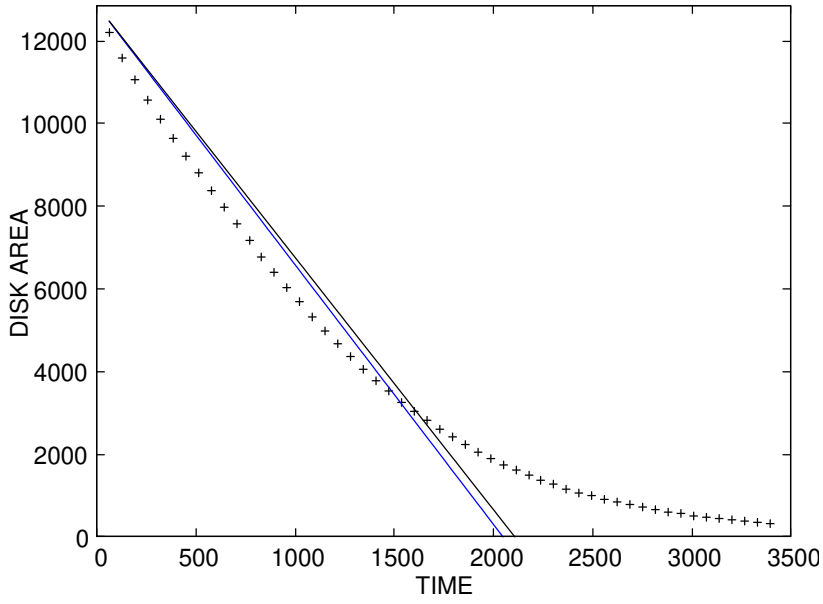
In the experiment, for small resolutions, all tested methods improve drastically. Among the interpolation scheme methods, the cubic C^2 spline is the most accurate method here, followed by the extended two-sweep [12] and the Fritsch-Carlson method [5] which are close together. The interpolation scheme using MQSI [8] shows similar results, whereas the delta-scheme [17] displays the least accuracy within the range of $\sigma = 16$ to $\sigma = 48$. At $\sigma = 48$, each method has a local minimum in the approximation error. Each method has an error increase in the next sample position of $\sigma = 64$. This is the first turning point from underestimating the disk area to overestimating it. The interpolation based approach with the quintic spline method seems to keep a residual error, whereas all other methods ultimately seem to tend towards zero in their errors. Experiments with $\sigma = 192$ and $\sigma = 256$ have shown that the error continues to increase slightly for the interpolation scheme with MQSI [8]. This is an indication that there may be problems with the consistency in this case.

6.4.3 Dissipative Artefacts

The disk shrinkage method can also be used to examine dissipative artefacts. Figure 6.17 illustrate how the sharpness of the border of the disk varies with different time steps using various methods. It shows the difference between the approximated solution and the analytic solution. For the delta-scheme [17], the parameter of $\alpha = 0.5$ was selected as this produce the sharpest results for this experiment. The results are consistent with the assumption that the linear spline is inconsistent. The quintic method with the quadratic facet model leads to a thinner blurred border than the other methods which leads to sharper results. On the other hand, we notice that this border seems to have a higher density compared to the other methods, as can be observed in Figure 6.19.

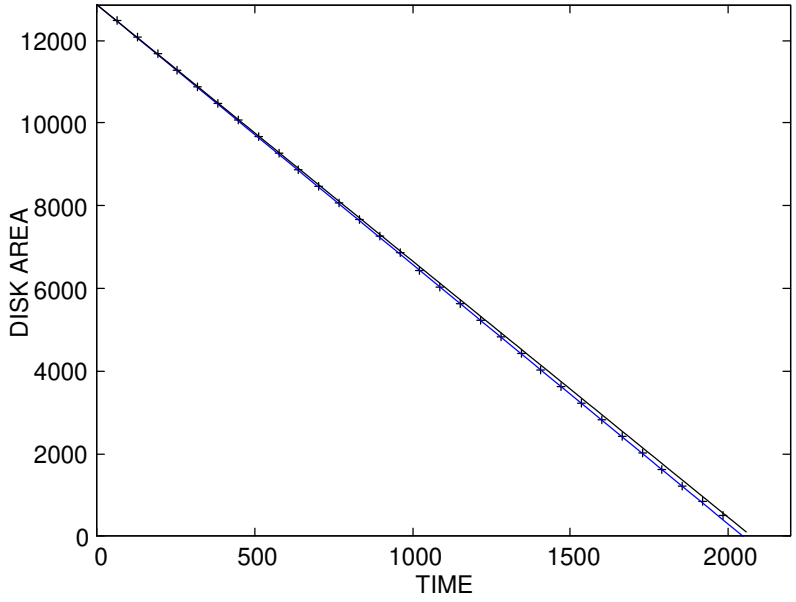


(a) Delta-Scheme [17] ($\alpha = 0, \gamma = 0$)

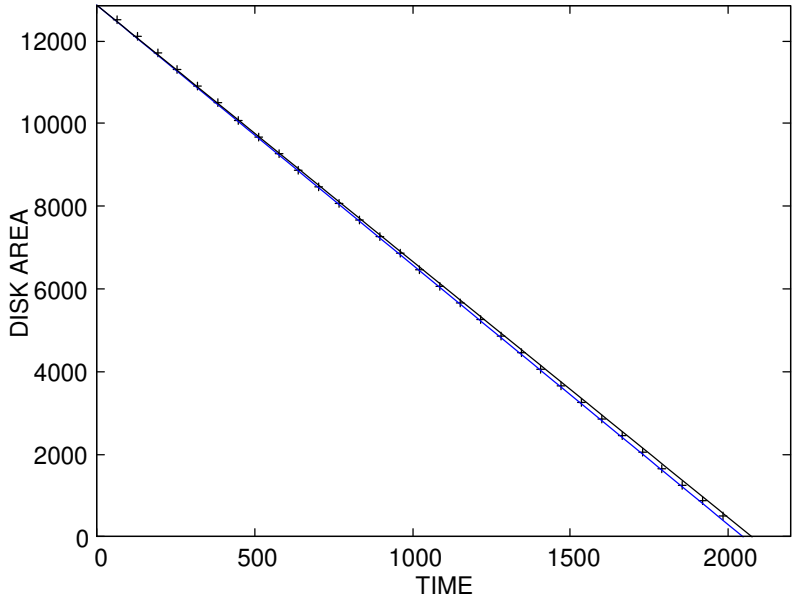


(b) Interpolation scheme with linear Spline

Fig. 6.13. Line fitting of the approximated disk areas $A(t_i)$ for the respective method for initial disk radius $\sigma = 64$. **Blue line:** analytic solution. **Black line:** fitted line through the areas $A(t_i)$.

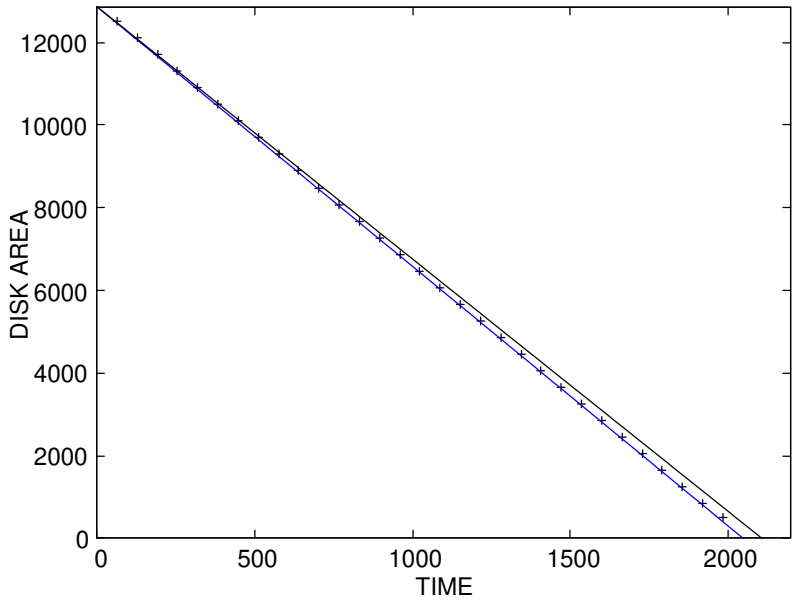


(a) Interpolation scheme with cubic C^2 spline

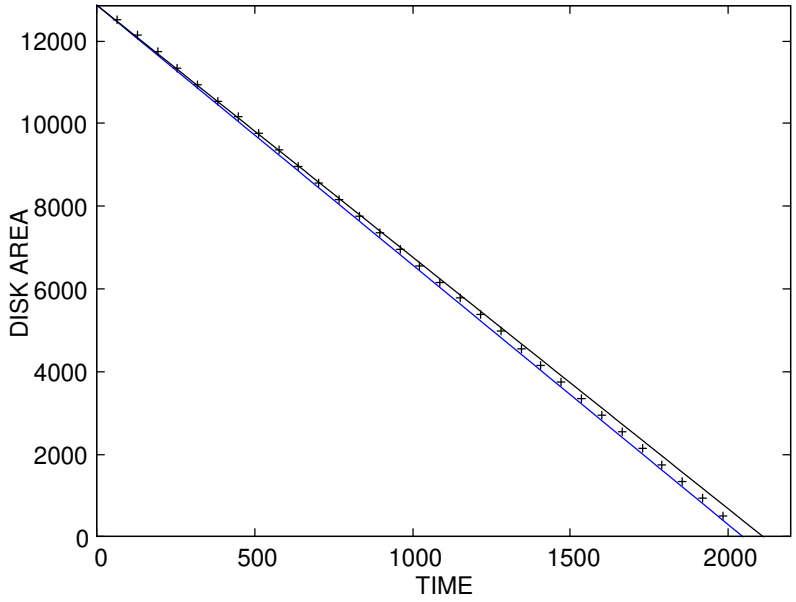


(b) Interpolation scheme with Fritsch-Carlson method [5]

Fig. 6.14. Line fitting of the approximated disk areas $A(t_i)$ for the respective method for initial disk radius $\sigma = 64$. **Blue line:** analytic solution. **Black line:** fitted line through the areas $A(t_i)$.



(a) Interpolation scheme with Extended two-sweep method [12]



(b) Interpolation scheme with MQSI [9]

Fig. 6.15. Line fitting of the approximated disk areas $A(t_i)$ for the respective method for initial disk radius $\sigma = 64$. **Blue line:** analytic solution. **Black line:** fitted line through the areas $A(t_i)$.

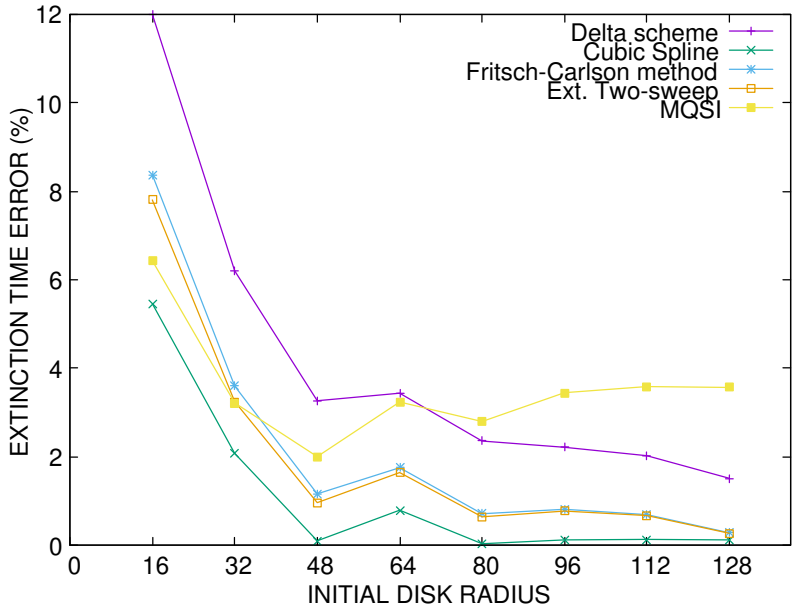


Fig. 6.16. Relative error of the approximated extinction time in dependence of the radius of the initial disk to the analytical solution. Approximation by line fitting. Delta scheme [17] parameters: $\alpha = 0$, $\gamma = 0$. Interpolation scheme with: Cubic C^2 spline, Fritsch-Carlson method [5], Extended two-sweep [12], MQSI [8].

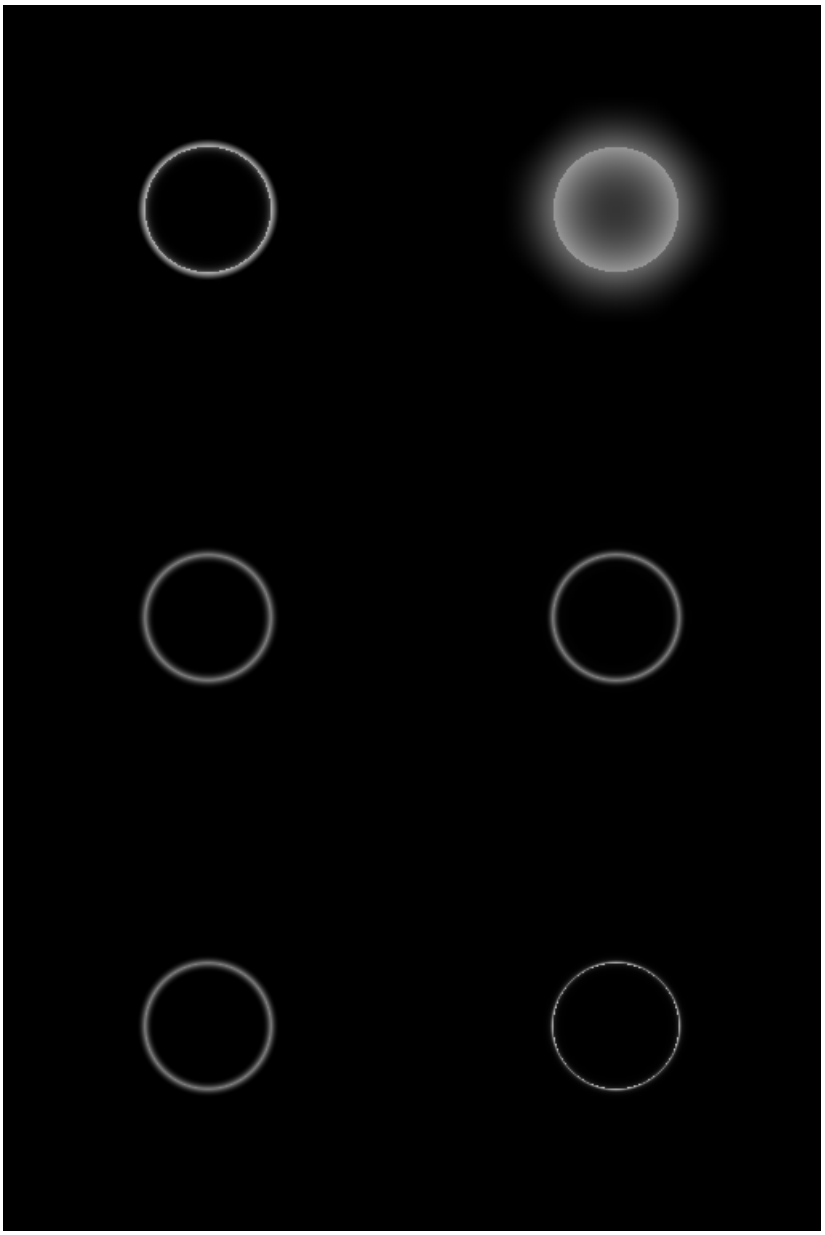


Fig. 6.17. MCM-filtering [17] on a disk shaped image with radius 64. Absolute difference between approximation and analytic solution at $t = 1280$. **Top left:** δ-scheme [17]($\alpha = 0.5$, $\gamma = 0$). Interpolation schemes with: **Top right:** Linear spline. **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

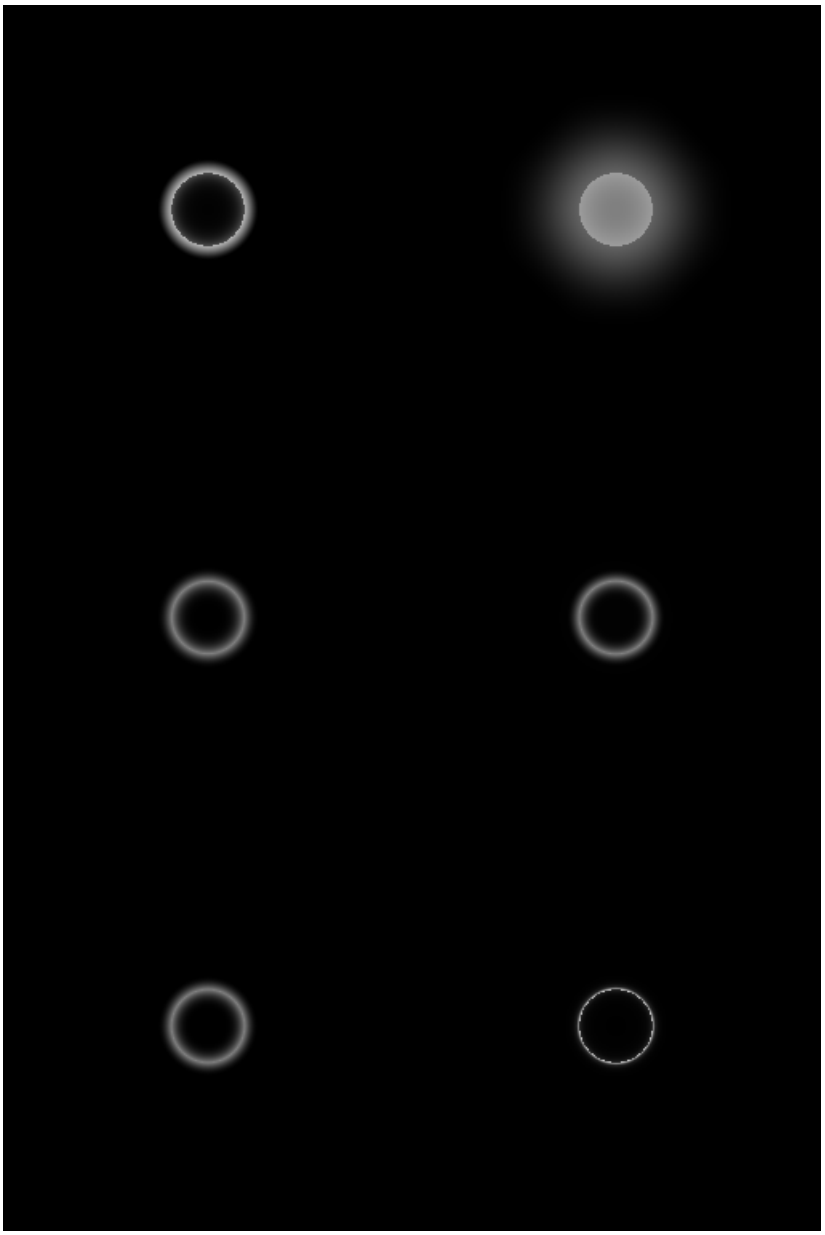


Fig. 6.18. MCM-filtering on a disk shaped image with radius 64. Absolute difference between approximation and analytic solution at $t = 1792$. **Top left:** δ-scheme ($\alpha = 0.5$, $\gamma = 0$). Interpolation schemes with: **Top right:** Linear spline. **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

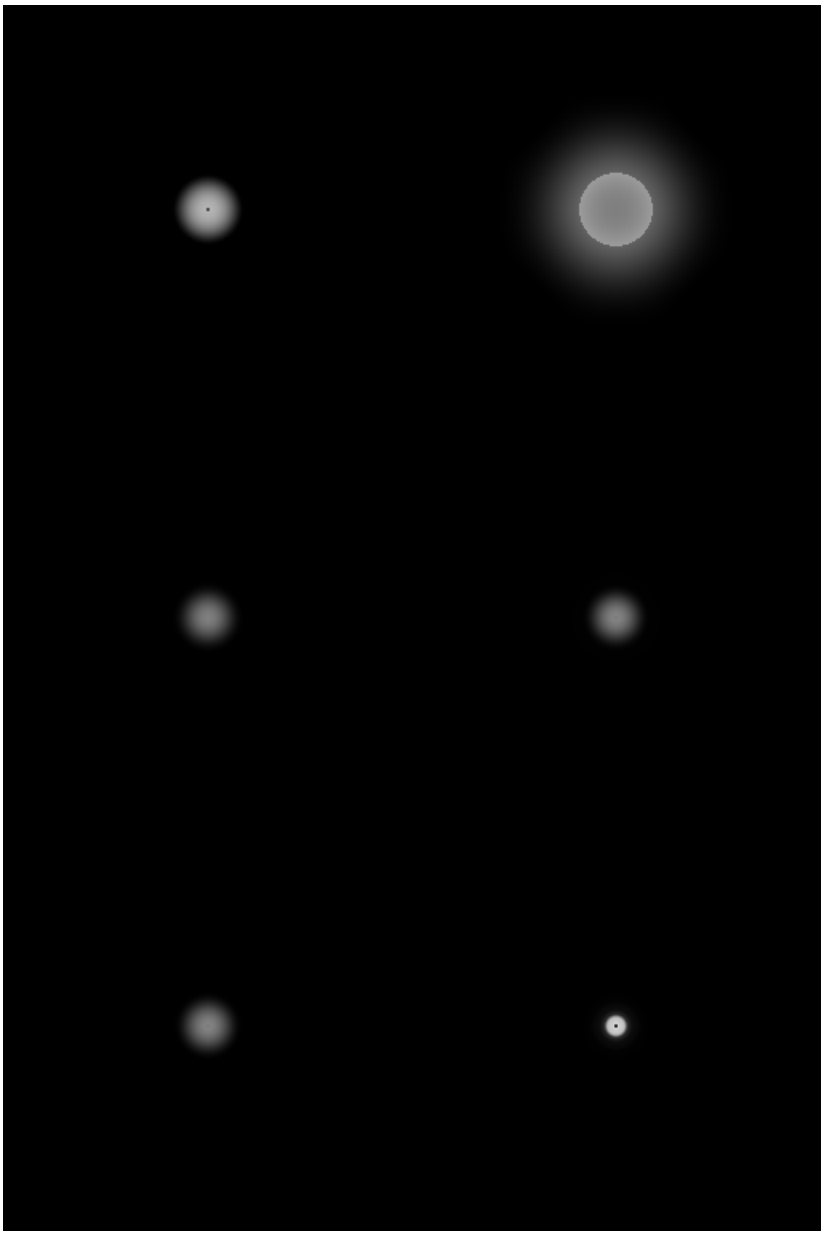


Fig. 6.19. MCM-filtering on a disk shaped image with radius 64. Absolute difference between approximation and analytic solution at $t = 2048$. **Top left:** δ-scheme ($\alpha = 0.5$, $\gamma = 0$). Interpolation schemes with: **Top right:** Linear spline. **Middle left:** Cubic C^2 spline. **Middle right:** Fritsch-Carlson method [5]. **Bottom left:** Ext. two sweep [12]. **Bottom right:** MQSI [8]

7 Conclusion

7.1 Summary

Monotone spline interpolation can be used for PDE-based numerical schemes in image processing. The interpolated image values approximate directional image derivatives directly through a one-dimensional finite difference approximation. It can be shown that the monotonicity preserving aspect of the interpolant ensures a maximum minimum principle. Concerning the approximation order, a fourth-order accurate interpolant is necessary for a second-order accurate approximation of the directional derivative.

These considerations lead to the extended two-sweep method by Eisenstat et al.[12] in the cubic spline case. Based on the monotonicity requirements of Fritsch and Carlson [5], they modify local derivatives for monotonicity while ensuring that the method is still fourth-order accurate such as the cubic C^2 splines.

With the goal of improving the rotational invariance of the results for the mean curvature motion PDE, we implement a monotone quintic spline interpolation method by Lux et al. [8]. While it is not able to improve the rotational invariance of the method, it drastically increases the sharpness of the resulting images. Possibly, this is due to the approximation of the local curvature in the quadratic facet model, that affects the spline via the second derivatives, which can not be set directly in the cubic case. Due to the third order accurate derivative approximation, this method gives only a first-order accurate approximation of MCM.

In a comparison with the explicit δ-scheme, the interpolation-based methods are of similar quality. In some aspects they can not quite reach the same quality such as rotational invariance. In terms of accuracy, the theoretical extinction time of a disk shaped initial condition is used as an overall estimate of the approximation quality. The cubic spline interpolation methods lead to the most accurate approximations. The dissipative

artefacts, on the other hand, are lowest with the quintic method.

7.2 Outlook

In future work, the quadratic facet model [6] from the monotone quintic spline interpolation [9] could be explored further in order to explain the drastic increase in sharpness of the resulting methods. In addition, it could be applied to other important PDEs in the field of image analysis such as anisotropic diffusion, provided that possible consistency issues can be resolved.

Furthermore, it may be worth exploring more effective numerical schemes with regards to time discretisation, such as implicit schemes. This could also potentially increase the accuracy of the method over the time.

In terms of monotone spline interpolation of images, there are several obvious ways to extend this approach. In the future, two-dimensional shape preserving spline interpolants could be studied. This would make the derivative approximation completely trivial, since there would be a differentiable analytic model of the image.

Bibliography

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

[2] Brakke, K.A.: The motion of a surface by its mean curvature (1978)

[3] Cheney, E.: Introduction to Approximation Theory. AMS Chelsea Pub. (1998)

[4] Datta, B.N.: Numerical Linear Algebra and Applications Philadelphia, vol. 116. SIAM (2010)

[5] Fritsch, F.N., Carlson, R.E.: Monotone piecewise cubic interpolation. SIAM Journal on Mathematical Analysis **17**, 238–246 (April 1980)

[6] Haralick, R.M., Watson, L.: A facet model for image data. Computer Graphics and Image Processing **15**(2), 113–129 (1981)

[7] Heß, W., Schmidt, J.W.: Positive quartic, monotone quintic C^2-spline interpolation in one and two dimensions. Journal of Computational and Applied Mathematics **55**(1), 51–67 (October 1994)

[8] Lux, T., Watson, L.T., Chang, T., Thacker, W.: Algorithm 1031: MQSI—monotone quintic spline interpolation. ACM Trans. Math. Softw. **49**(1) (mar 2023). https://doi.org/10.1145/3570157, `https://doi.org/10.1145/3570157`

[9] Lux, T.C.H.: Interpolants, Error Bounds, and Mathematical Software for Modeling and Predicting Variability in Computer Systems. Ph.D. thesis, Virginia Polytechnic Institute and State University (August 2020)

[10] Lux, T.C.H.: MQSI: Monotone quintic spline interpolation [Source code] (2020), https://github.com/tchlux/papers/tree/master/%5B2020-08%5D_ACMTOMS_(MQSI)

[11] Randall L. Dougherty, Alan Edelman, J.M.H.: Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic hermite interpolation. Mathematics of Computation **52**(186), 471–494 (April 1989)

[12] S. C. Eisenstat, K. R. Jackson, J.W.L.: The order of monotone piecewise cubic interpolation. SIAM Journal on Numerical Analysis **22**, 1120–1237 (December 1985)

[13] Schaefer, K., Weickert, J.: Mathematical morphology in image analysis. MIA Group, Faculty of Mathematics and Computer Science, Saarland University, Germany (lecture slides, summer term 2022)

[14] Sonar, T.: Angewandte Mathematik, Modellbildung und Informatik. Vieweg, Braunschweig (2001)

[15] Ulrich, G., Watson, L.T.: Positivity conditions for quartic polynomials. SIAM Journal on Scientific Computing **15**(3), 528–544 (May 1994)

[16] Weickert, J.: Anisotropic Diffusion in Image Processing. Teubner, Stuttgart (1998)

[17] Weickert, J.: Mean curvature motion, explicit scheme with delta stencil [Source code] (202 1)

[18] Weickert, J., Welk, M., Wickert, M.: L2-stable nonstandard finite differences for anisotropic diffusion. In: Kuijper, A., Bredies, K., Pock, T., Bischof, H. (eds.) Scale Space and Variational Methods in Computer Vision. pp. 380–391. Springer, Berlin (2013)