# Curvature recovery using splines

Bachelor Thesis

*Author:*
Oscar Kromhof
4496906

*Supervisor:*
Dennis den Ouden
- van der Horst

August 30, 2020

## Preface

This paper was written for the completion of my bachelor's degree in Applied Mathematics at Delft University of Technology. I want to thank my supervisor Dennis den Ouden-van der Horst for his excellent guidance and support during this project.

# Contents

## 1 Introduction

Phase transition is a phenomena that is ubiquitous in nature and industry. Think for example of ice passing to water or steel that is cooling from the austenite phase to the ferrite phase. The dynamics of such a transition are often formulated as a moving boundary problem (also known as a Stefan Problem). The problem can be considered in 1D, 2D and 3D. In this project we will limit ourself to the two dimensional case. It follows from the physics of this problem that the boundary between two phases can be modelled as a closed two-dimensional smooth curve which we will name $\Gamma$.
An important characteristic of $\Gamma$ is it's curvature $k$. This quantity is of interest to us because the propagating speed of the boundary is dependent on it's curvature, see for example [1]. If we have a general parametric representation of the curve then the curvature is given by

$$k = \frac{x'y'' - y'x''}{((x')^2 + (y')^2)^{\frac{3}{2}}}. \tag{1}$$

Where $x := x(t)$ and $y := y(t)$ are the parameterized $x, y$-coordinates of the curve $\Gamma$ on some interval $[a, b]$. From Equation (1) we see that we need to have at least that $x(t)$, $y(t) \in C^2[a, b]$ such that $k(t)$ can be calculated and is continuous. In many modelling applications the curve $\Gamma$ is described implicitly and explicit expressions such as $x$ and $y$ are unknown to us.
One widely used numerical modelling technique for the moving boundary problem is to use the level-set method mentioned in [2] and [3]. This method describes the boundary $\Gamma(t)$ between the phases with the aid of a signed distance formula $\phi(x, y, t)$, where $x$ and $y$ are special coordinates and t represents time. The curve is at time $t$ then implicitly described as $\Gamma(t) = \{(x, y) : \phi(x, y, t) = 0\}$, hence the name levelset method. The movement of the boundary is then captured by evolving the signed-distance function $\phi(x, y, t)$ using a convection equation as described by Den Ouden in [4]. Now to actually calculate the motion a finite difference technique is used. We will omit the technicalities of this method, but in this numerical process we need to know the value of the curvature at each time point to be able to calculate the propagating speed of the boundary and move to the next time point.
The method will return to us an ordered set of discrete points $\{(x_i, y_i)\}_{i=0}^{n}$ in the plane which represent the discretized boundary curve $\Gamma$ between two phases for some time instance. *Our main goal in this project is to find, apply and analyze a method to recover the curvature from data sets generated by the level set method.* The main technique we will use for curvature recovery will be spline interpolation. This is a well known method to interpolate smoothly between data points by making use of piece-wise defined polynomials. We will start in Chapter 2 with cubic splines adapted in such a way that it can be used for closed curve interpolation. We will then test and

analyze our method on multiple data sets, to come to the conclusion that we need to improve our methodology. This will be investigated in Chapter 3 and 4. We will close this project with Chapter 5 that contains a discussion on the results and suggestions for further research on the subject.

## Derivation of the parametric expression of the curvature

Since the curvature plays a central role in this project we will give a derivation of Equation (1) in this section.

First let $\mathbf{\Gamma} : [a, b] \to \mathbb{R}^2$ be a parameterization for a smooth regular curve $C$ given by $\mathbf{\Gamma}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$ (By regular we mean that $\mathbf{\Gamma}'(t) \neq \mathbf{0}$ for all $t \in [a, b]$). Then we can define the arc length parameter as

$$s(t) = \int_a^t \|\mathbf{\Gamma}'(u)\| du, \tag{2}$$

where $|| \cdot ||$ denotes the Euclidean norm.

We will first show that the reparameterization of the curve $C$ with respect to arc length given by $\hat{\mathbf{\Gamma}}(s)$ satisfies $||\hat{\mathbf{\Gamma}}'(s)|| = 1$. We will need this result later in our derivation.

*Proof.* Since we assumed that the curve is regular and $\|\mathbf{\Gamma}'(u)\| > 0$ we know that $s$ is strictly monotonically increasing so $s(t)$ has an inverse $t(s)$ and

$$\frac{dt}{ds} = \frac{1}{\frac{ds}{dt}} = \frac{1}{\|\mathbf{\Gamma}'(t)\|}. \tag{3}$$

Where we used the inverse function theorem on the composition $(t \circ s)(x) = x$ where $x \in [0, s(b)]$, and the fundamental theorem of calculus on Equation (2). Now let $\hat{\mathbf{\Gamma}}(s) := \mathbf{\Gamma}(t(s))$ be the special reparameterization of the same cuve $C$ in terms of it's arclength. Now it follows from the chain rule that

$$\hat{\mathbf{\Gamma}}'(s) = \mathbf{\Gamma}'(t(s))\frac{dt}{ds}. \tag{4}$$

If we now take the norm on both sides of Equation (4) and substitute Equation (3) we arrive at the desired conclusion that,

$$\|\hat{\mathbf{\Gamma}}'(s)\| = \|\mathbf{\Gamma}'(t(s))\| \frac{1}{\|\mathbf{\Gamma}'(t(s))\|} = 1.$$

□

If we define $\mathbf{T}(s) := \hat{\mathbf{\Gamma}}'(s)$ to be the constant unit tangent vector to the curve or intuitively the 'unit speed', then the curvature is defined in the following way

$$k(s) := \left\| \frac{d\mathbf{T}(s)}{ds} \right\| = \|\mathbf{T}'(s)\|. \tag{5}$$

Since $\mathbf{T}(s)$ has unit length for all $s$ we have $\mathbf{T} \cdot \mathbf{T} = \|\mathbf{T}\|^2 = 1$. If we differentiate both sides of the equation with the help of the product rule for scalar products we get

$$2\mathbf{T} \cdot \frac{d\mathbf{T}}{ds} = 0 \implies \mathbf{T} \cdot \frac{d\mathbf{T}}{ds} = 0.$$

This means that $\mathbf{T}$ and $\frac{d\mathbf{T}}{ds}$ are orthogonal for all $s$. Because $\mathbf{T}$ is tangent to the curve we have the following relationship for the curvature

$$\mathbf{T}'(s) = k(s)\mathbf{N}(s). \tag{6}$$

Where $\mathbf{N}(s)$ is the unit normal obtained by rotating $\mathbf{T}(s)$ $\pi/2$ radians. Figure 1 gives a nice visualisation of this.

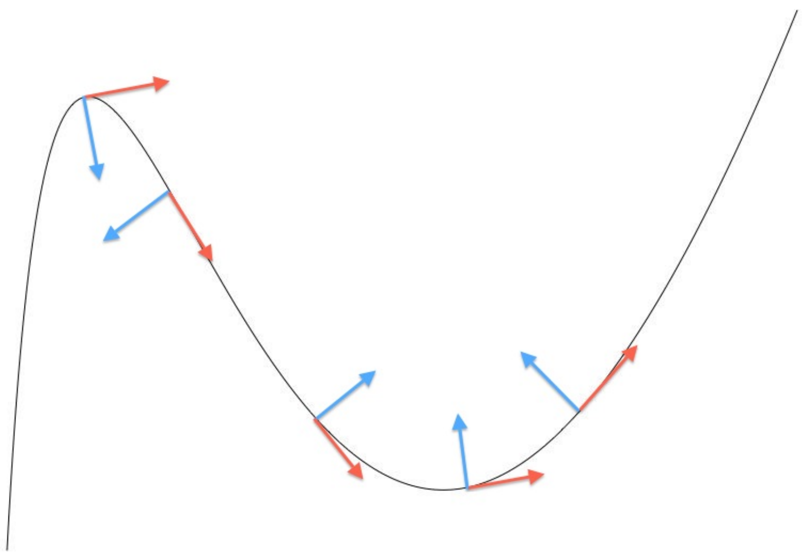


Figure 1: Visualisation of $\mathbf{T}$ and $\mathbf{N}$, where $\mathbf{N}$ is represented by the blue arrow and $\mathbf{T}$ is represented by the orange arrow.

Now we want to eliminate the parameter $s$ and find the curvature in terms of $t$. We start by differentiating $\mathbf{\Gamma}(t(s))$ by the chain rule we have

$$\frac{d\mathbf{\Gamma}}{dt} = \frac{ds}{dt}\frac{d\hat{\mathbf{\Gamma}}}{ds} = \frac{ds}{dt}\mathbf{T}. \tag{7}$$

If we differentiate again we find by the product and the chain rule that

$$\frac{d^2\mathbf{\Gamma}}{dt^2} = \frac{d^2s}{dt^2}\frac{d\hat{\mathbf{\Gamma}}}{ds} + \Big(\frac{ds}{dt}\Big)^2\frac{d^2\hat{\mathbf{\Gamma}}}{ds^2} = \frac{d^2s}{dt^2}\mathbf{T} + \Big(\frac{ds}{dt}\Big)^2 k\mathbf{N}. \tag{8}$$

Where we substituted $\frac{d^2\mathbf{\Gamma}}{ds^2}$ by Equation (7). We take the scalar product of both sides of Equation (8) with respect to $\mathbf{N}$. Since $\mathbf{T}$ and $\mathbf{N}$ are orthogonal and $\mathbf{N}\cdot\mathbf{N} = 1$ this will yield

$$\frac{d^2\mathbf{\Gamma}}{dt^2}\cdot\mathbf{N} = k\Big(\frac{ds}{dt}\Big)^2. \tag{9}$$

By differentiating Equation (2) and the fundamental theorem of calculus we find $\frac{ds}{dt} = [(x')^2 + (y')^2]^{\frac{1}{2}}$ where $x' = \frac{dx}{dt}$ and $y' = \frac{dy}{dt}$. So $\left(\frac{ds}{dt}\right)^2 = (x')^2 + (y')^2$. Since $t(s) \in [a, b]$ we will use the slight abuse of notation $t := t(s)$. From Equation (9) we find the result,

$$\begin{aligned} k =& \frac{1}{(x')^2 + (y')^2}\Big((x'', y'')\cdot\frac{(-y', x')}{[(x')^2 + (y')^2]^{\frac{1}{2}}}\Big) && (10)\\ =& \frac{1}{(x')^2 + (y')^2}\Big(\frac{x'y'' - x''y'}{[(x')^2 + (y')^2]^{\frac{1}{2}}}\Big) && (11)\\ =& \frac{x'y'' - x''y'}{[(x')^2 + (y')^2]^{\frac{3}{2}}}. && (12)\end{aligned}$$

Where we used that $\mathbf{N}$ is the $\pi/2$ radians rotation of $\mathbf{T}$ in Equation (10).

# 2 Curvature recovery with cubic splines

This chapter presents a way to smoothly interpolate between data points through the method of cubic splines. The analysis in [5] on pages 20-23 is followed with a few adaptations to meet the desired periodicity conditions. This will be followed by two sections on the application of this method for curvature recovery on test data-sets. The chapter is concluded with a short discussion on the results.

## 2.1 Definition of a spline

Let $f \in C[a,b]$, and let $a = t_0 < t_1 < ... < t_n = b$ be the $n+1$ nodes where the function values $f(t_k)$ are known. (In our specific case $f$ will either be $x(t)$ or $y(t)$). A spline, $s : [a,b] \to \mathbb{R}$, is a piece-wise polynomial that is connected smoothly in the function values $f(t_k)$. So the spline has in general the following form

$$s(t) = \begin{cases} s_0(t), & t \in [t_0, t_1], \\ s_1(t), & t \in [t_1, t_2], \\ \vdots \\ s_{n-1}(t), & t \in [t_{n-1}, t_n]. \end{cases}$$

Note that the spline must go smoothly through the points $(t_k, f(t_k))$ so we will for sure have that $s_k(t_{k+1}) = s_{k+1}(t_{k+1}) = f(t_k)$. So the function $s$ is well defined on $[a,b]$.

## 2.2 Splines of degree 3 (periodic)

A cubic spline $s$ has the following properties:

a On each subinterval $[t_k, t_{k+1}]$, $s$ is a third degree polynomial $s_k$, for $k = 0, \dots, n-1$.

b The values at the nodes satisfy $s(t_k) = f(t_k)$, for $k = 0, \dots, n$.
Note: since we are interpolating a *periodic* function we automatically have $s(t_0) = f(t_0) = f(t_n) = s(t_n)$.

c The smoothness conditions

$$\begin{aligned} s_k(t_{k+1}) &= s_{k+1}(t_{k+1}),\ k = 0, \dots, n-2, \\ s_k'(t_{k+1}) &= s_{k+1}'(t_{k+1}),\ k = 0, \dots, n-2, \\ s_k''(t_{k+1}) &= s_{k+1}''(t_{k+1}),\ k = 0, \dots, n-2. \end{aligned}$$

(This ensures that $s \in C^2[a,b]$ which we are going to need if we want to calculate the curvature given by Equation (1)).

d The periodic boundary conditions

$$\begin{aligned} s_0(t_0) &= s_{n-1}(t_n), \\ s_0'(t_0) &= s_{n-1}'(t_n), \\ s_0''(t_0) &= s_{n-1}''(t_n). \end{aligned}$$

We also introduce the standard form for each of the $n$ piece-wise polynomials of order 3 on the intervals $[t_k, t_{k+1}]$,

$$s_k(t) = a_k(t-t_k)^3 + b_k(t-t_k)^2 + c_k(t-t_k) + d_k \text{ for } k = 0,\dots,n-1. \quad (13)$$

With the conditions given in b, c and d we can uniquely determine the coefficients $a_k$, $b_k$, $c_k$ and $d_k$ which will define the spline $s$.

### 2.2.1 Determining the coefficients

We start by defining:

- $h_k := t_{k+1} - t_k$ for $k = 0,\dots,n-1$,

- $f_k := f(t_k)$ for $k = 0,\dots,n$.

From the condition $s(t_k) = f_k$ we find,

$$d_k = f_k \text{ for } k = 0,\dots,n-1. \quad (14)$$

We find an expression for our $a_k$ coefficients by looking at the condition $s_k''(t_{k+1}) = s_{k+1}''(t_k)$. From (30) we find $s_k''(t) = 6a_k(t-t_k) + 2b_k$ so we have

$$6a_k(t_{k+1} - t_k) + 2b_k = s_k''(t_{k+1}) = s_{k+1}''(t_k) = 2b_{k+1} \text{ for } k = 0,\dots,n-2.$$

If we solve for the $a_k$'s we find immediately that,

$$a_k = \frac{1}{3h_k}(b_{k+1} - b_k) \text{ for } k = 0,\dots,n-2. \quad (15)$$

Now for finding the last coefficient, $a_{n-1}$, we deviate from [5] and find by the condition $s_0''(t_0) = s_{n-1}''(t_n)$ that we have $6a_{n-1}h_{n-1} + 2b_{n-1} = 2b_0$. Solving for $a_{n-1}$ yields

$$a_{n-1} = \frac{1}{3h_{n-1}}(b_0 - b_{n-1}). \quad (16)$$

From the second condition $s_k(t_{k+1}) = s_{k+1}(t_{k+1})$ it follows that $a_k h_k^3 + b_k h_k^2 + c_k h_k + d_k = f_{k+1}$. If we now substitute our result from (14) and (15). We find for $c_k$

$$c_k = \frac{f_{k+1} - f_k}{h_k} - h_k \frac{2b_k + b_{k+1}}{3}, \text{ for } k = 0, \dots, n-2. \tag{17}$$

To find an expression for $c_{n-1}$ we make use of the periodicity condition $s_0(t_0) = s_{n-1}(t_n)$

$$c_{n-1} = \frac{f_0 - f_{n-1}}{h_{n-1}} - h_{n-1} \frac{2b_{n-1} + b_0}{3}. \tag{18}$$

Finally we want to find the $b_k$, for this we use that for the first derivative $s'_k(t_{k+1}) = s'_{k+1}(t_{k+1})$ holds. This will give us $3a_k h_k^2 + 2b_k h_k + c_k = c_{k+1}$. substituting (14), (15), (17) and simplifying yields.

$$h_k b_k + 2(h_k + h_{k+1}) b_{k+1} + h_{k+1} b_{k+2} = 3\Big( \frac{f_{k+2} - f_{k+1}}{h_{k+1}} - \frac{f_{k+1} - f_k}{h_k} \Big) \tag{19}$$

for $k = 0, \dots, n-3$.

Now we need 2 more equations to be able to determine the $b_k$'s uniquely. The first one we get from $s'_{n-2}(x_{n-1}) = s'_{n-1}(x_{n-1})$. This implies

$$h_{n-1} b_0 + h_{n-2} b_{n-2} + 2(h_{n-2} + h_{n-1}) b_{n-1} = 3\Big( \frac{f_0 - f_{n-1}}{h_{n-1}} - \frac{f_{n-1} - f_{n-2}}{h_{n-2}} \Big). \tag{20}$$

We find the last equation by the periodicity condition $s'_0(t_0) = s'_{n-1}(t_n)$ this yields

$$c_0 = s'_0(t_0) = s'_{n-1}(t_n) = 3a_{n-1} h_{n-1}^2 + 2b_{n-1} h_{n-1} + c_{n-1}.$$

Now we substitute (16), (17), (18) which gives,

$$h_{n-1}(b_0 - b_{n-1}) + 2b_{n-1} h_{n-1} - h_{n-1} \frac{2b_{n-1} + b_0}{3} + h_0 \frac{2b_0 + b_1}{3} = \frac{f_1 - f_0}{h_0} - \frac{f_0 - f_{n-1}}{h_{n-1}},$$

and with further simplification we find

$$2b_0(h_{n-1} + h_0) + h_0 b_1 + b_{n-1} h_{n-1} = 3\Big( \frac{f_1 - f_0}{h_0} - \frac{f_0 - f_{n-1}}{h_{n-1}} \Big). \tag{21}$$

The results (19), (20) and (21) can be compactly summarized in matrix notation:

$$A = \begin{pmatrix} 2(h_{n-1} + h_0) & h_0 & 0 & 0 & 0 & \dots & 0 & 0 & h_{n-1} \\ h_0 & 2(h_0 + h_1) & h_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & h_{n-3} & 2(h_{n-3} + h_{n-2}) & h_{n-2} \\ h_{n-1} & 0 & 0 & 0 & 0 & \dots & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{pmatrix},$$

$$\mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix}, \ \mathbf{f} = \begin{pmatrix} 3\left(\frac{f_1-f_0}{h_0} - \frac{f_0-f_{n-1}}{h_{n-1}}\right) \\ 3\left(\frac{f_2-f_1}{h_1} - \frac{f_1-f_0}{h_0}\right) \\ \vdots \\ 3\left(\frac{f_{n-1}-f_{n-2}}{h_{n-2}} - \frac{f_{n-2}-f_{n-3}}{h_{n-3}}\right) \\ 3\left(\frac{f_0-f_{n-1}}{h_{n-1}} - \frac{f_{n-1}-f_{n-2}}{h_{n-2}}\right) \end{pmatrix}.$$

If we analyze the matrix we note that

$$\begin{aligned} |a_{11}| &= |2(h_{n-1}+h_0)| = 2(h_{n-1}+h_0) > h_0 + h_{n-1} = |a_{12}| + |a_{1n}|, \\ |a_{ii}| &= |2(h_{i-1}+h_i)| = 2(h_{i-1}+h_i) > h_{i-1}+h_i = |a_{ii-1}| + |a_{ii+1}| \text{ for } i \notin \{1,n\}, \\ |a_{nn}| &= |2(h_{n-2}+2h_{n-1})| = 2(h_{n-2}+h_{n-1}) > h_{n-1}+h_{n-2} = |a_{n1}| + |a_{nn-1}|. \end{aligned}$$

Thus the $n \times n$ matrix $A$ satisfies the property of diagonal dominance:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \ \forall i \in \{1, \ldots, n\}.$$

This means we can apply Levy-Desplanques theorem [6] which states that matrices of this type have $\det(A) \neq 0$, so the system is non-singular.
If we solve $A\mathbf{b} = \mathbf{f}$ we retrieve the $b_k$ coefficients which can be used to determine the $a_k$ and $c_k$ coefficients with Equations (15), (16), (17) and (18).

With the change from natural boundary conditions to periodic boundary conditions we lose the tridiagonal structure of the matrix $A$ (the bottom left and upper right elements are non-zero). But for this specific almost tridiagonal structure are still fast solvers available that can solve $A$ in $\mathcal{O}(n)$ operations. For example the parallel solvers mentioned in [7] by T. Chen.

## 2.3 Performance analysis: cubic spline curvature recovery on uniformly generated test data

This section presents test results obtained by the cubic spline interpolation with the method that was discussed in the previous paragraph on data that was uniformly generated from certain closed curves. The notation that will be used in this paragraph is listed below.

- $\mathbf{\Gamma}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$ with $t \in [a, b]$ denotes the unknown parametric expression for the curve that generated the data.
- The retrieved cubic spline parameterization of the curve is denoted as $\mathbf{S}_c$.
- $k_s(t)$ will be the curvature of $\mathbf{S}_c(t)$: *the recovered curvature.*

- $\mathbf{K}_0$ the vector that stores the theoretical curvature $k_i$ at the given data points.
- $\mathbf{K}_s$ the vector that stores the recovered curvature $k_s(t_i)$ at the given data points.
- $N$ denotes the amount of interpolation nodes, so $N = n + 1$.

We will also define 3 metrics to measure the error in the recovered curvature with respect to the theoretical curvature. Based on the 1-norm, the 2-norm and the $\infty-$norm.

$$||\mathbf{K_0} - \mathbf{K_s}||_1 := \frac{1}{N}\sum_{i=0}^{N-1} |k_i - k_s(t_i)|, \tag{22}$$

$$||\mathbf{K_0} - \mathbf{K_s}||_2 := \Big(\frac{1}{N}\sum_{i=0}^{N-1} (k_i - k_s(t_i))^2\Big)^{\frac{1}{2}}, \tag{23}$$

$$||\mathbf{K_0} - \mathbf{K_s}||_\infty := \max_{0\le i\le N-1} |k_i - k_s(t_i)|. \tag{24}$$

We have 4 different test data-sets[1] called:

- Function I (Circle) - curvature known
- Function II (Starfish) - curvature known
- Function III (Cat) - curvature unknown
- Function IV (Camel) - curvature unknown

Function I and II are generated from closed curves that have a simple parametric form. So at each $t_k$ we know the curvature given by Equation (1). Function III, IV are generated from closed curves that don't have a simple parametric form describing them so the curvature will here be unknown. It is given that the function values in our data set are generated at uniform spaced nodes $a = t_0, t_0 + h, t_0 + 2h, \ldots, t_0 + hN = b$ in $[a, b]$. It is also given that the generating curve $\Gamma(t)$ is smooth.

The parameterization nodes $t_k$ for $x(t)$ and $y(t)$ haven been chosen uniformly in $[0, 1]$. Note that it isn't necessary to know the original parameter interval $[a, b]$ since the intrinsic geometry of the spline $s$ won't change if we scale or translate the parameter interval which is proved in [8]. Since we know that the data is uniformly generated from the original curves, we choose the nodes for $x(t_k)$ and $y(t_k)$ as $t_k = k/n$ for $k = 0, \ldots, n$. This implies that $h_0 = h_1 = \cdots = h_{n-1} = h$, so this will simplify the matrix

[1]The data can be obtained by request.

$A$ and the vector $\mathbf{f}$. We will label the splines for $x(t)$, $y(t)$ respectively as $s^x(t)$, $s^y(t)$. (The superscript is used to avoid confusion with the piece-wise defined polynomials $s_k$). For the interpolated parametric representation of the curve we have the following expression:

$$\mathbf{S}_c(t) = \begin{pmatrix} s^x(t) \\ s^y(t) \end{pmatrix} \text{ with } t \in [0,1], \tag{25}$$

and the *recovered curvature* will then be given by,

$$k_s = \frac{(s^x)'(s^y)'' - (s^y)'(s^x)''}{[((s^x)')^2 + ((s^y)')^2]^{\frac{3}{2}}}. \tag{26}$$

The parameter variable $t$ is omitted here for readability but of course Equation (26) is defined for all $t \in [0,1]$.

**Function I: The Circle**

A Python script based on the method presented in Section 2.2.1 was written to perform the analysis. Figure 2 displays the recovered circle with cubic spline interpolation. (For plots of the interpolated geometric figures of test data the $x$-axis is always implicitly labeled by $s^x(t)$ and the $y$-axis by $s^y(t)$.) For the lowest amount of data points $N = 25$ we already see a nice circle appearing going smoothly through the data points. In Figure 3 and 4 we

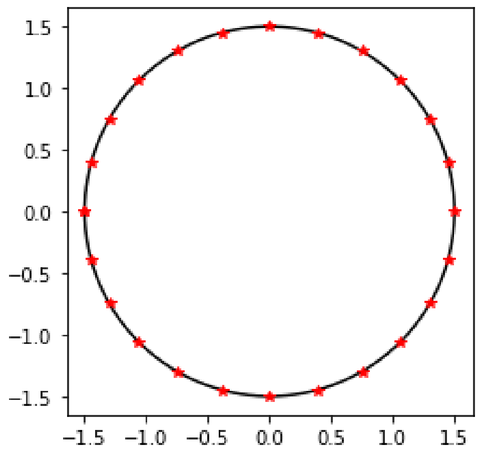
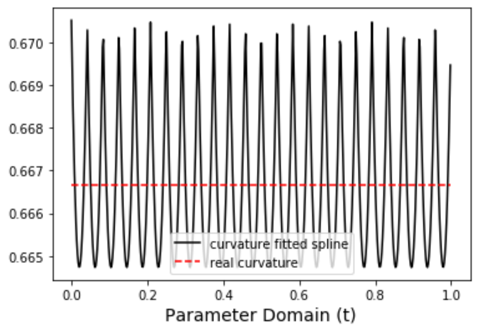


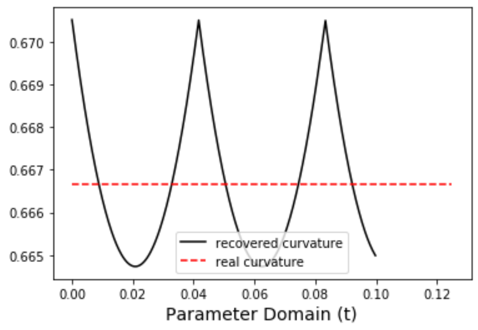
Figure 2: Plot of the cubic-spline interpolation of Function I with $N = 25$ data points. The red stars represent the data points $\{(x_i, y_i)\}_{i=0}^{n=24}$, and the black curve represents $\mathbf{S}_c(t)$ with $t \in [0,1]$.

see plots of the recovered $k_s$ curvature on different scales of the parameter domain. The recovered curvature oscillates around the theoretical value of the curvature represented by the straight red dotted line. However the oscillations are very small in amplitude, at points where the deviations are maximal the relative deviation error is only about $\frac{0.00385}{0.66667} \times 100 \approx 0.58\%$ of the theoretical value and this will only decrease as we increase $N$ which can be seen in Figure 3. The oscillatory behaviour is probably caused by small errors building up in the numerical simulations and rounding errors in the used data.

In Figure 3b we can clearly see sharp peaks in the recovered curvature,



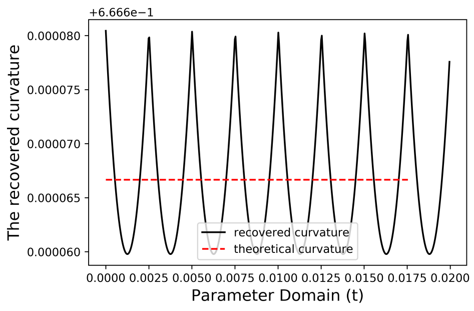
(a) Plot of the recovered curvature $k_s$ for the circle in black, and the theoretical/real curvature in dotted red.



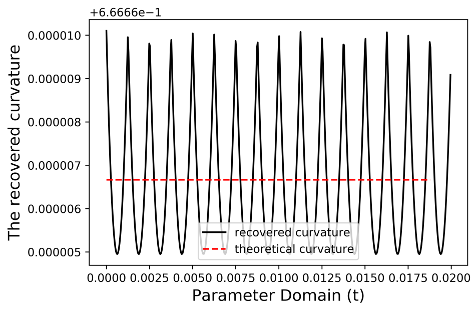
(b) Zoom on a subset of the parameter domain $[0, 0.1]$.

Figure 3: Plots of the recovered curvature on different parameter domains with $N = 25$ interpolation nodes.

this is undesirable behaviour since we know that the circle is an infinitely smooth geometric object so its curvature should be smooth as well. If we take a look at the recovered curvature for higher values of $N$ as can been seen in Figure 4b. The oscillatory behaviour still exists only with a much smaller amplitude and higher frequency.



(a) The recovered curvature for the circle for $N = 400$ nodes on the parameter subset $[0, 0.02]$



(b) The recovered curvature for the circle for $N = 800$ nodes on the parameter subset $[0, 0.02]$.

Figure 4: Plots of the recovered curvature on $[0, 0.2]$.

The decrease in the error for the recovered curvature is further confirmed by Table 1 and Figure 5.

| $N$ | 25 | 50 | 100 | 200 | 400 | 800 |
|---|---|---|---|---|---|---|
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_1$ | 0.00385 | 0.000916 | 0.000224 | $5.54 \times 10^{-5}$ | $1.38 \times 10^{-5}$ | $3.44 \times 10^{-6}$ |
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_2$ | $7.41 \times 10^{-6}$ | $4.2 \times 10^{-7}$ | $2.5 \times 10^{-8}$ | $1.53 \times 10^{-9}$ | $9.49 \times 10^{-11}$ | $5.90 \times 10^{-12}$ |
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_\infty$ | 0.00385 | 0.000916 | 0.000224 | $5.54 \times 10^{-5}$ | $1.38 \times 10^{-5}$ | $3.44 \times 10^{-6}$ |

Table 1: Error in the recovered curvature with cubic splines for the circle. $N$ denotes the amount of interpolation nodes in the data.

For a doubling in the amount of interpolation nodes the 1- and infinity norm error decrease with a factor of about 4. For example $\frac{0.000916}{0.00385} = 0.24 \approx \frac{1}{4}$. This gives some empirical evidence for

$$||\mathbf{K_0} - \mathbf{K_s}||_1 = \mathcal{O}(N^{-2}),$$
$$||\mathbf{K_0} - \mathbf{K_s}||_\infty = \mathcal{O}(N^{-2}).$$

For the mean square norm we find that for a doubling of interpolation nodes the error reduces by about a factor of 16. which implies

$$||\mathbf{K_0} - \mathbf{K_s}||_\infty = \mathcal{O}(N^{-4}).$$

Figure 5 displays the log-log plot of Table 1 including reference order lines $\mathcal{O}(N^{-2})$, and $\mathcal{O}(N^{-4})$. Figure 5 confirms our findings for the order of convergence of $k_s$ to the theoretical curvature $k$ with cubic spline interpolation for the circle.

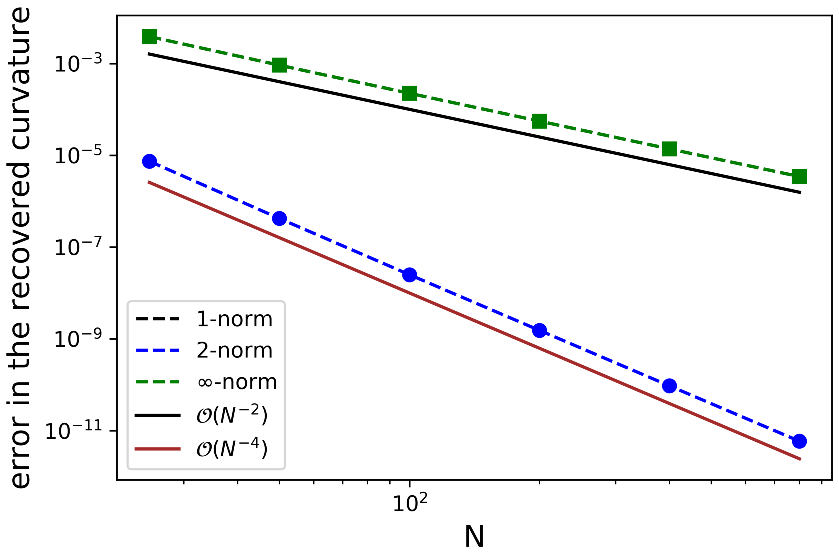
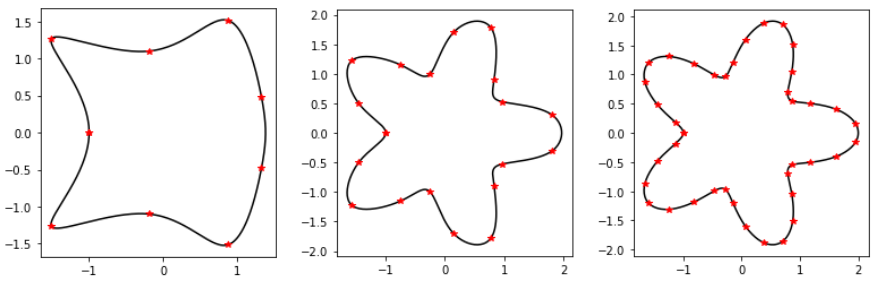


Figure 5: Log-log plot of the errors in the recovered curvature measured with different metrics. It also includes order lines $\mathcal{O}(N^{-2})$ and $\mathcal{O}(N^{-4})$ for reference. The 1-norm coincides with the $\infty$-norm.

## Function II: The Starfish

Figure 6 displays plots of $\mathbf{S}_c$ for different values of $N$. For $N = 20$ the recovered curve already seems to assume it's desired shape. If we then double the amount of data points visually we don't see the shape change anymore. So we expect the error in the recovered curvature for $N = 10$ to be high and after that to see quick convergence to the theoretical curvature.



(a) $N = 10$ (b) $N = 20$ (c) $N = 40$

Figure 6: Plots of the cubic interpolated Function II (starfish). With interpolation nodes $N = 10$, $N = 20$, $N = 40$.

The errors in the recovered curvature $k_s$ in Table 2 and Figure 7 seem to confirm our conjecture based on the plots of the recovered curves.

| $N$ | 10 | 20 | 40 | 80 | 160 | 320 |
|---|---|---|---|---|---|---|
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_1$ | 7.803 | 0.723 | 0.150 | 0.036 | 0.0089 | 0.0022 |
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_2$ | 75.2343 | 0.388 | 0.0228 | 0.0015 | 9.47 $\times 10^{-5}$ | 5.875 $\times$ $10^{-6}$ |
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_\infty$ | 24.53 | 1.751 | 0.683 | 0.196 | 0.050 | 0.0127 |

Table 2: Errors in the recovered curvature with cubic splines for the starfish. $N$ denotes the amount of interpolation nodes in the data.

After $N = 40$ we note that if $N$ doubles the error decreases by a factor of 4, take for example $\frac{0.036}{0.150} = 0.24 \approx \frac{1}{4}$. This suggests that the 1- and $\infty$-norm converge to the theoretical value with order 2. With similar calculations we see that the 2-norm shows evidence for order 4 convergence. Also the slope of the order lines in Figure 7 are in agreement with this. So there is strong empirical evidence for

$$||\mathbf{K_0} - \mathbf{K_s}||_1 = \mathcal{O}(N^{-2}),$$
$$||\mathbf{K_0} - \mathbf{K_s}||_\infty = \mathcal{O}(N^{-2}),$$
$$||\mathbf{K_0} - \mathbf{K_s}||_\infty = \mathcal{O}(N^{-4}).$$

This means there is no significant difference in the performance of the method for the starfish in comparison with the circle, in the sense that their order of convergence seem to be the same.

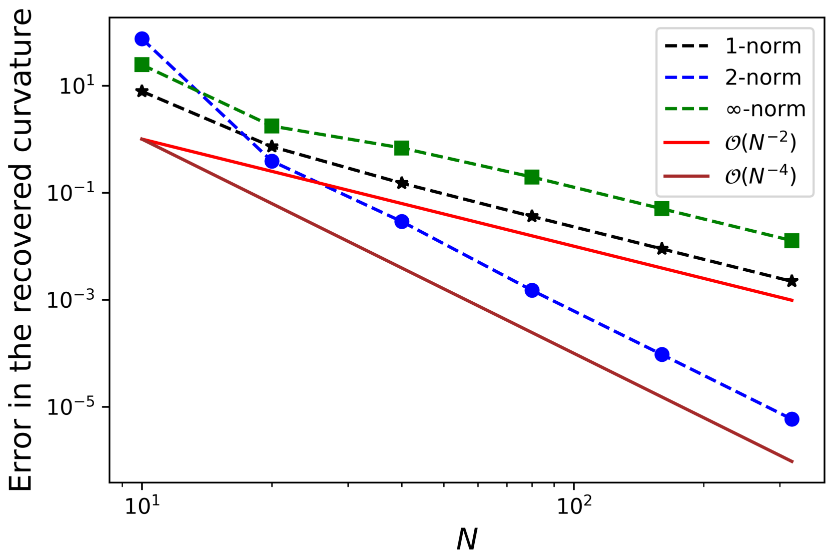


Figure 7: Log-log plot of the error in the recovered curvature measured with different metrics for the starfish.

A plot of the recovered curvature together with the theoretical curvature for the starfish is displayed in Figure 8. The function $k_s(t)$ is periodic which corresponds nicely with our obtained interpolated curves in Figure 6.

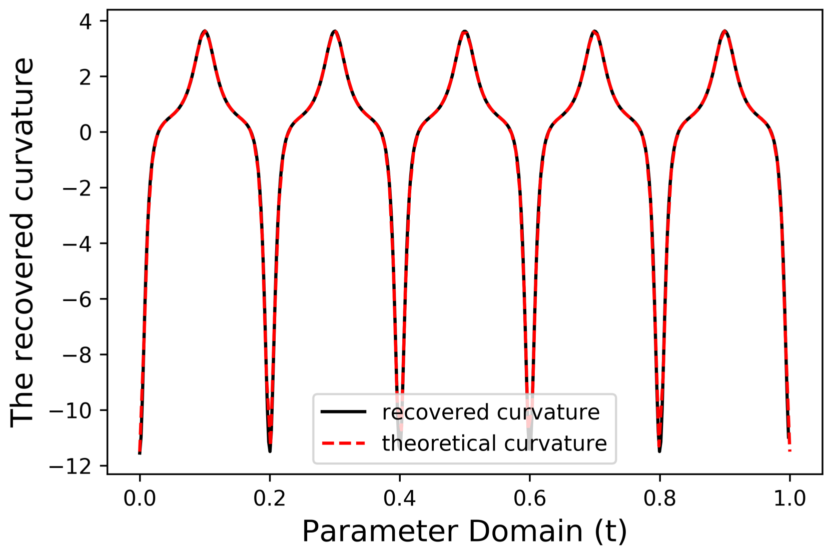
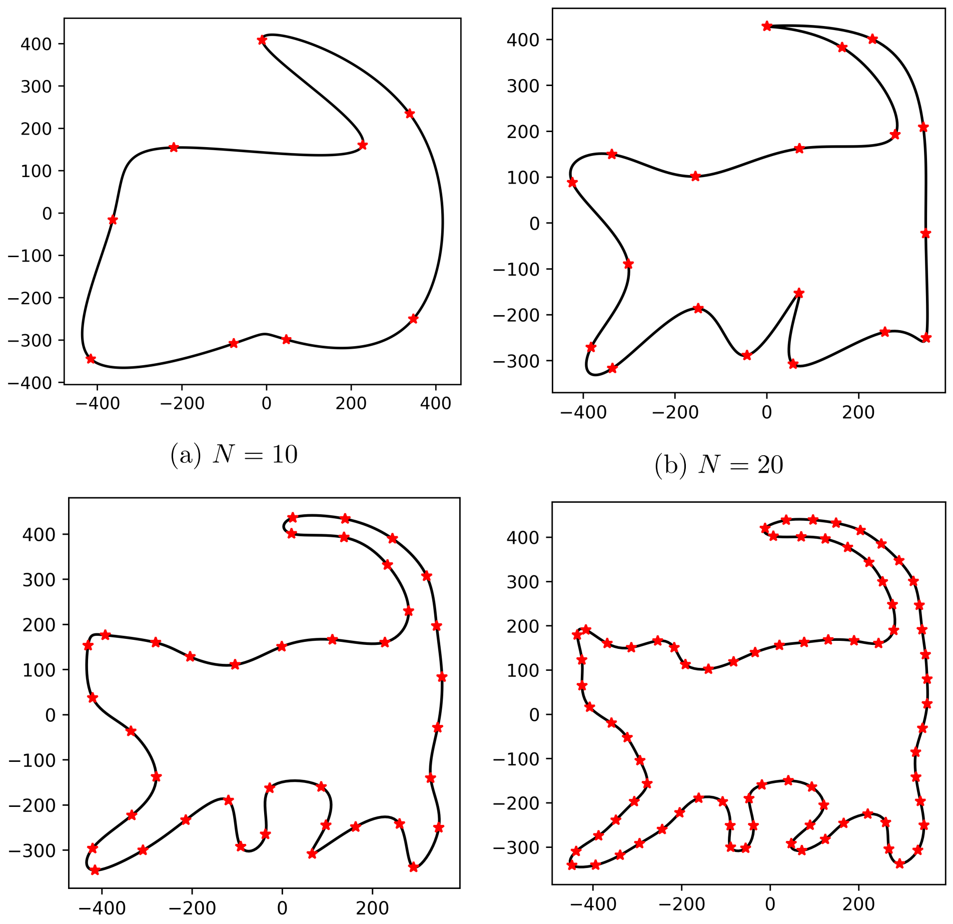


Figure 8: Recovered curvature for the starfish $N = 160$.

**Function III & IV: The Cat & The Camel**

Finally we consider Function III and Function IV. As stated before we don't have the theoretical curvature for these functions so we can only judge the performance of the cubic spline curvature recovery by visual inspection. In Figure 9 and 10 we see the cat and the camel emerge. In both cases the recovered curve starts to resemble the animal in such a way that it is recognizable at $N = 40$ interpolation nodes. In Figure 9b, 9c we see some sharp edges occur but these get smoothed for $N = 80$ in Figure 9d. For the camel in Figure 10 there remain some sharp edges in the figure even for $N = 80$.



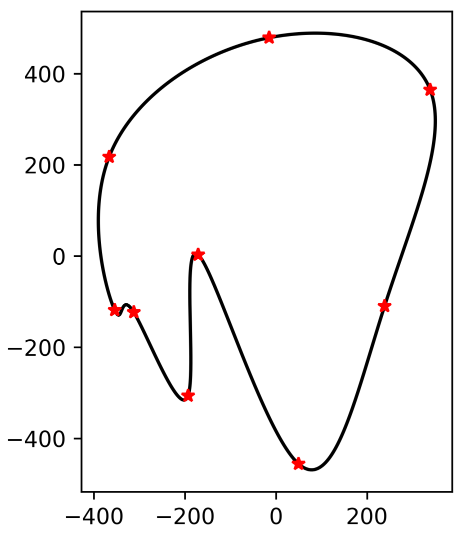
(a) $N = 10$

(b) $N = 20$

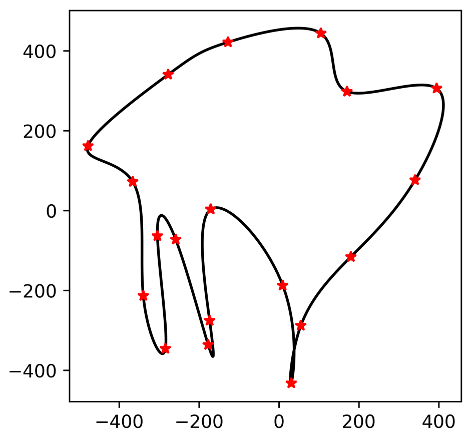
(c) $N = 40$

(d) $N = 80$

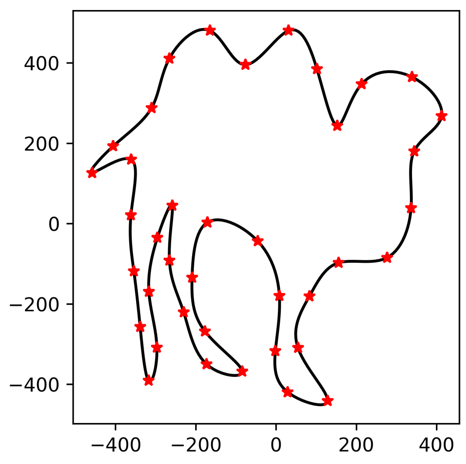
Figure 9: Plots of the recovered curve for the cat.



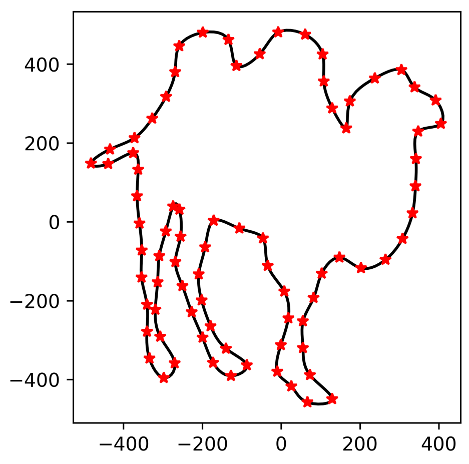
(a) $N = 10$



(b) $N = 20$



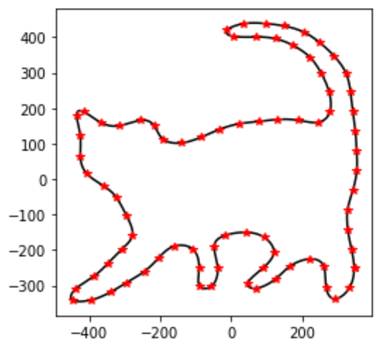
(c) $N = 40$



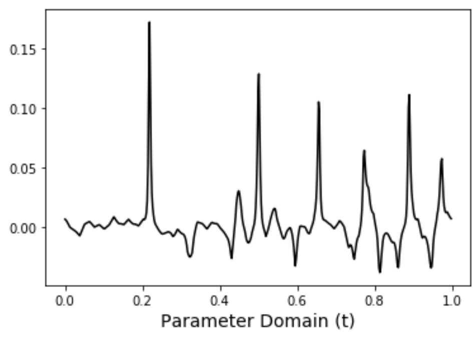
(d) $N = 80$

Figure 10: Plots of the recovered curve for the camel.

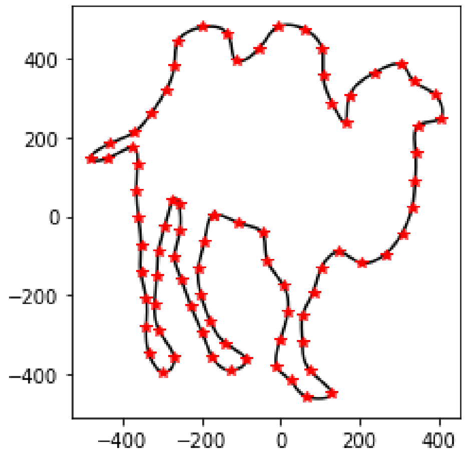
The interpolated curves in comparison with their recovered curvature can be found in Figure 11. For the cat we see 6 peaks in the positive direction corresponding to it's tail, legs and left ear. We also note that the scale of interpolated curves (ranging from -400 to 400) is much larger than for the circle and the starfish (ranging from -2 to 2) so we expect a much lower value in the curvature. This is indeed confirmed in Figure 11. The curvature only varies by about 0.3 from zero while for the starfish we saw the curvature assume values of -10. This also explains the sharp edges we see in the recovered curves. If we would zoom in on the sharp edges we would see a smooth turn but relative to the large scale they look non-smooth.



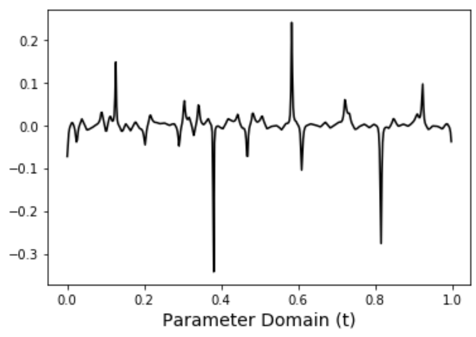
(a) Cubic interpolation Function III.



(b) Recovered curvature for the cat.



(c) Cubic interpolation Function IV.



(d) Recovered curvature for the camel.

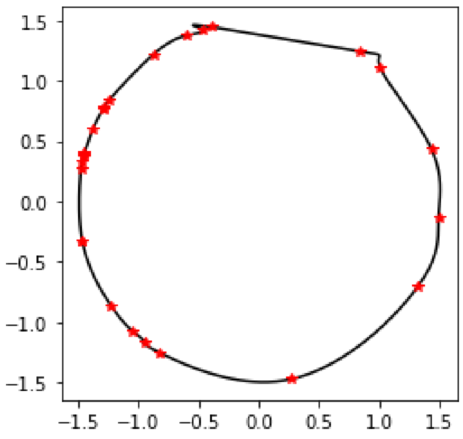
Figure 11: Interpolation and curvature recovery for the camel and the cat. $N = 80$

## 2.4 Performance analysis: curvature recovery with cubic splines on randomly sampled test data

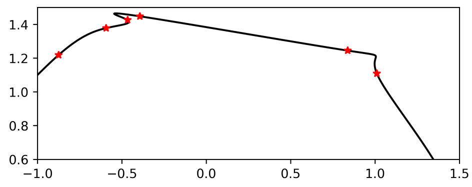
Thus far we only considered data that was uniformly generated from the parameter domain of the Functions I, II, III and IV. But in practice the data generated by the level set method is not uniformly generated. That's why we will now study the behaviour of the cubic spline curve interpolation with data sets from which the parameter values are randomly sampled from the parameter domain of the original functions. Note that the order of the data points is still preserved only the distance between the nodes in the original sample domain of the curve is now randomized and this distance is unknown to us. Throughout this paper we will use the term 'random data' as a short hand for the randomly generated data set from the test curves.

### 2.4.1 The circle with data generated from randomized parameter domain

In Figure 12 we see that the interpolation goes wrong which is not that surprising since we lost a nice property of our data set.



(a) The red stars represent the randomly generated data points and the black line the interpolated curve.



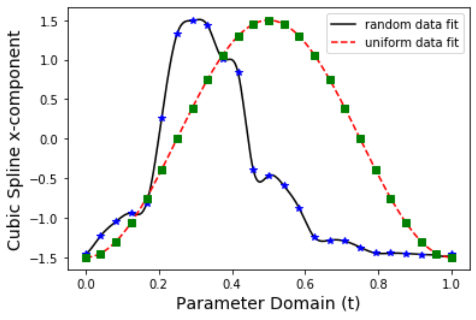
(b) Zoom on two obvious irregularities of the curve $\mathbf{S}_c$.

Figure 12: Interpolation on the random data set for the circle, with $N = 25$.

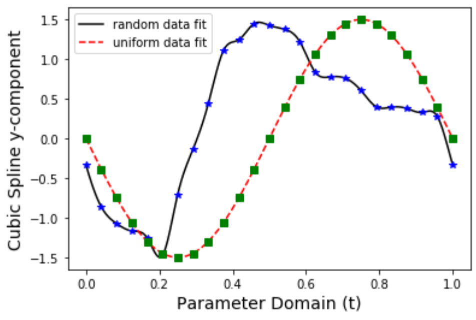
To get a better understanding on why exactly we get such a poor fit as displayed in Figure 12, we look at the individual $x$ and $y$ components for the circle given by $s^x$, $s^y$ and compare this to the uniform fit for which we have strong evidence that everything works properly. This simulation can be seen in Figure 13.

If we label the sampled nodes from the original curve $t_k^*$. Then it is clear that points $t_k^*$, $t_{k+1}^*$ that were sampled close to each other get stretched too far out by the uniform nodes $t_k$, and conversely points that were originally sampled far away get squeezed to close to each other. This causes the distortions that become visible in Figure 12. If we now look for example at the $x$-component as we increase the value of $N$ we expect that these distortions become smaller since on average the distance between $t_k^*$, $t_{k+1}^*$ becomes smaller. If we simulate this situation we obtain Figure 14 which displays that the plots of the random interpolations come closer to the uniform interpolation, so we do seem to have that the component $s^x$ converges to the $x$-component of the theoretical curve. For $s^y$ we find a similar result which implies $\mathbf{S_c} \to \Gamma$. However we still see a lot of jaggedness in the curve.

So it might also be interesting to take a look at the behaviour of $||\mathbf{S}_c'||$, the ‘speed’. The plot of the speed is displayed in Figure 15. In Figure 15b

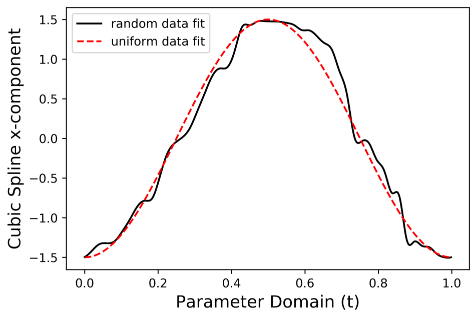


(a) $s^x(t)$ for $t \in [0, 1]$ random and uniform data.

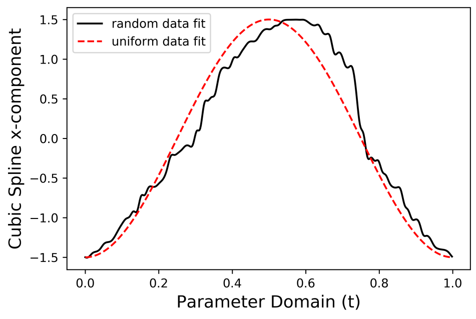


(b) $s^y(t)$ for $t \in [0, 1]$ random and uniform data.

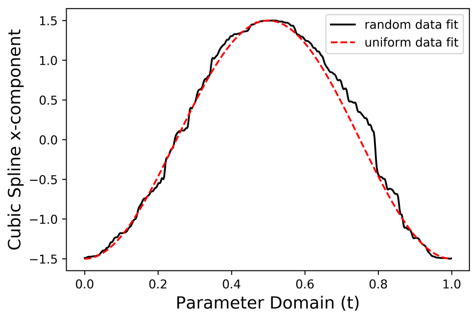
Figure 13: $s^x$, $s^y$ are both plotted twice once for the uniform and once for the random data $N = 25$. The blue stars and green squares indicate respectively the pairs $(t_k, x(t_k))$ for the random and uniform data.



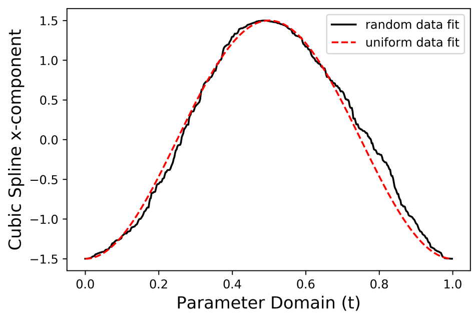
(a) $N = 50$



(b) $N = 100$



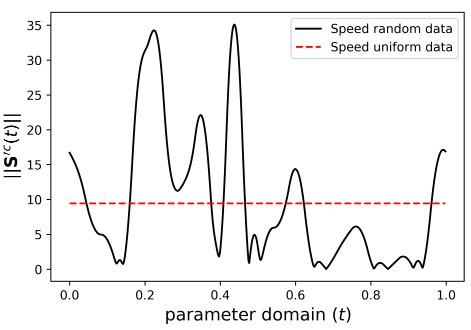
(c) $N = 200$



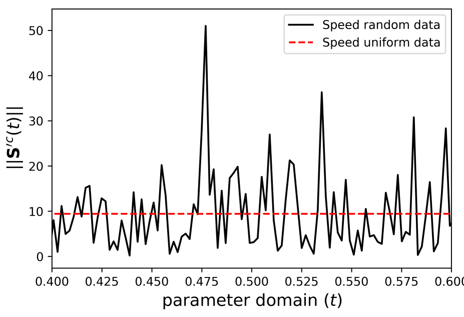
(d) $N = 400$

Figure 14: Plots of the $x$-component for uniform data (red) and random data (black).

with $N = 800$ data points $||\mathbf{S}'_c||$ still shows very wild behaviour. This is caused by the squeeze and stretch process discussed, big distances between the data points gets squeezed into a too short parameter interval which result in huge spikes in $||\mathbf{S}'_c||$, so there won't be any convergence for the speedvector $\mathbf{S}$ as we increase $N$.



(a) Plot of $||\mathbf{S}'_c||$ for $N = 25$



(b) Plot of $||\mathbf{S}'_c||$ for $N = 800$ on the parameter domain $[0.4, 0.6]$

Figure 15: Interpolation on the random data set for the circle, with $N = 25$.

## 2.5 Conclusions

### For the uniform data

The curvature recovery with cubic splines applied to Function I and Function II both show empirical evidence for second order convergence in the 1- and $\infty$-norm, and fourth order convergence in the 2-norm. Of course this is only an inductive argument for a finite amount of test cases, for a definitive proof for the method's convergence and it's order one would need to provide a rigorous mathematical proof. For the Functions III and IV we didn't have theoretical data available as a reference frame, but the interpolated curves look good and correspond nicely with their recovered curvature.
The only undesirable result that we obtained in our analysis for the uniform case is that the recovered curvature $k_s$ showed some sharp peaks were differentiability is lost, as we could for example see in Figure 3b. In theory this should not occur since we are dealing with smooth curves and that implies that it's curvature should also be smooth. But these results don't come as a surprise since the cubic spline only guarantees that $s^x$, $s^y \in C^2[0,1]$ so $(s^x)''$ and $(s^y)''$ can be non smooth functions. So it might be a good idea to smoothen the recovered curvature by using a higher order interpolation method.

### For the random data

For the random data we only limited our analysis to the case of the circle. Since the interpolation already doesn't work properly for this simple geometric object it will for sure not work for even more complicated closed curves. Choosing the interpolation nodes $t_k$ uniform for the random data causes huge swings in the recovered magnitude of the derivative $||\mathbf{S}_c'||$ compared to the uniform data, and this behaviour won't fade for higher values of $N$. So it is clear that we need to adapt our method so we can use it for randomly generated data sets, this will be topic of the the next chapter.

## 3 Different spacing for the interpolation nodes

With the discussion of the previous chapter in mind, one could imagine that choosing the parameter nodes $t_k$ to be uniformly in $[0, 1]$ is not the optimal configuration because it disregards the spacing of the data points $D_i = (x_i, y_i)$. There has been research conducted for picking the parameterization nodes $t_k$ which will be briefly discussed below. We will first list the relevant equation and definitions as compactly summarized by Shene in [11].

We define the 'length' of the polygon, that is the linearly interpolated data set, as

$$L := \sum_{i=1}^{n} ||D_i - D_{i-1}||^a.$$

Where $|| \cdot ||$ is the Euclidean norm and $a$ a parameter that can assume the values $0 \leq a \leq 1$.

The spacing of the nodes are given by the following equations.

$$t_0 := 0, \tag{27}$$

$$t_k := \frac{1}{L}\Big(\sum_{i=0}^{k} ||D_i - D_{i-1}||^a\Big) \text{ for } k = 1, \ldots, n-1, \tag{28}$$

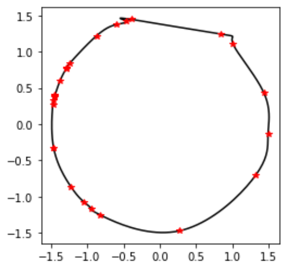
$$t_n := 1. \tag{29}$$

If $a = 0$ then the $t_k$ are just uniformly distributed in $[0, 1]$. The most common way to choose the nodes $t_k$ is to use so called *chordal parameterization* which corresponds to the case $a = 1$. The motivation for this choice is that the distance between two points on a curve is a reasonable approximation to the length of the associated curve segment. We might then expect that the 'speed', $||\mathbf{S}_c'(t)||$, is close to a constant speed for $t \in [0, 1]$ as stated in [8]. In this way the chordal parameterization kind of tries to approximate the arclength parameterization only the speed doesn't necessarily have to be unit. Finally if $a = \frac{1}{2}$ we speak of *centripetal parameterization.* Lee claims in [10] that the centripetal method for choosing the parameter nodes will in almost all cases yield better results then the previous two options. The method is called centripetal because Lee introduces the method by analogy of a car driving on a road with a sharp turn ahead (where the road represents the curve). The driver would not keep driving at a uniform speed but in anticipation slow down before making the turn, in order to reduce *centripetal force* such that the driver can maintain a comfortable posture or prevent the car from slipping. Lee suggests that the centripetal force should be proportional to the change in angle. The centripetal method is an approximation to this model. In fact all values $0 \leq a \leq 1$ can be chosen so $a$ can be seen as a kind of blending parameter.

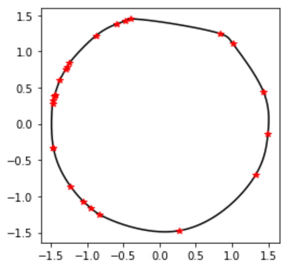
## 3.1 Results and Analysis

### Function I: the circle

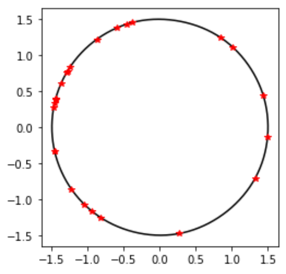
In Figure 16 it is clear that the chordal parameterization outperforms the uniform and the centripetal parameterization for the circle on $N = 25$ data points. Figure 16a is the same as Figure 12b and we already analyzed this case extensively in Section 2.4.1. In Figure 16b we see that the circle looks a bit more smooth than the uniformly spaced one, for example the sharp peak at the top of the circle from Section 2.4.1 is now gone. Figure 16c looks to be a perfect circle that goes smoothly true the data points.



(a) $a = 0$



(b) $a = \frac{1}{2}$



(c) $a = 1$

Figure 16: Plots of the cubic interpolated circle with randomly generated data. With uniform (a), chordal (b) and centripetal parameterization (c). $N = 25$

We can also inspect the plots of $||\mathbf{S}_c'||$ for the different values of $a$ displayed in Figure 17. We see indeed that the chordal parameterization produces a steady constant speed displayed by the blue line. The other two parameterizations meander around this speed. Thus the Chordal parameterization nicely emulates the same constant speed as we saw for the uniform data with uniform parameterization.

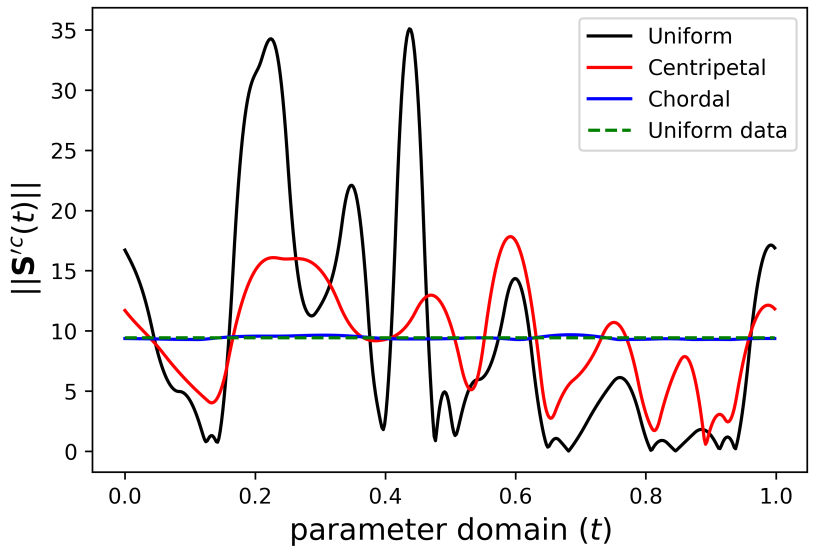


Figure 17: $||\mathbf{S}'_c||$ for the different types of parameterization: Uniform, Chordal and Centripetal. The green dotted line represents the uniform parameterization on uniform data.

Also if we inspect the errors measured according to the supnorm for the recovered curvature for the circle for the values $a = 0$, $a = \frac{1}{2}$, $a = 1$. We get the result displayed in Figure 18.

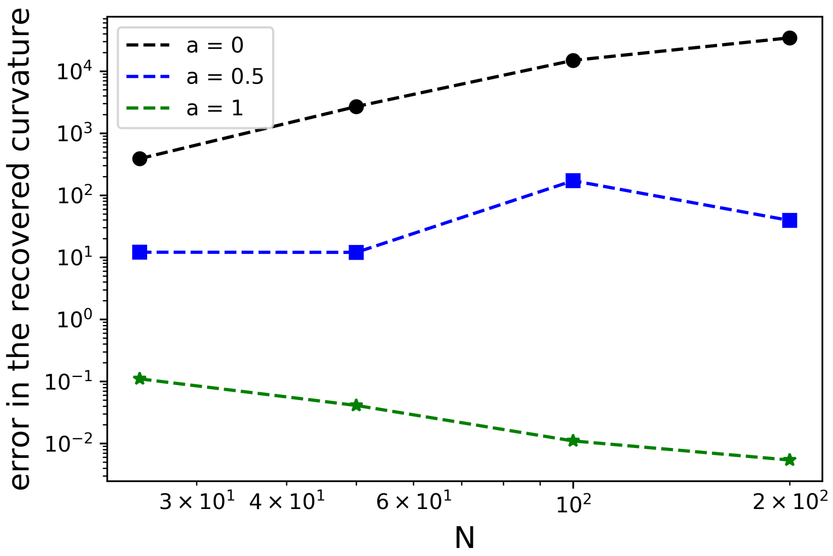


Figure 18: Log-log plot of the error with respect to the supnorm for the recovered curvature for different values for $a$.

We see that the errors in the recovered curvature for the uniform and centripetal parameterization don't show any evidence for convergence. We can conclude from Figure 18 and our previous discussions that only $a = 1$ shows some promising results and we will investigate this case further. Table 3 displays the errors in the recovered curvature with respect to the different metrics.

| $N$ | 10 | 20 | 40 | 80 | 160 | 320 |
|---|---|---|---|---|---|---|
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_1$ | 0.021 | 0.00542 | 0.0017 | 0.00045 | $9.96 \times 10^{-5}$ | $2.626 \times 10^{-5}$ |
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_2$ | 0.000637 | $5.3\times10^{-5}$ | $3.61 \times 10^{-6}$ | $3.8 \times 10^{-7}$ | $1.4 \times10^{-8}$ | $1.237 \times 10^{-9}$ |
| $\|\|\mathbf{K_0} - \mathbf{K_s}\|\|_\infty$ | 0.109 | 0.041 | 0.011 | 0.00544 | 0.00081 | 0.000492 |

Table 3: Errors in the recovered curvature with cubic splines for the circle with randomly generated data. ($a = 1$)

The results of Table 3 are plotted in Figure 19. Where we see again order 2 and order 4 convergence in the recovered curvature.

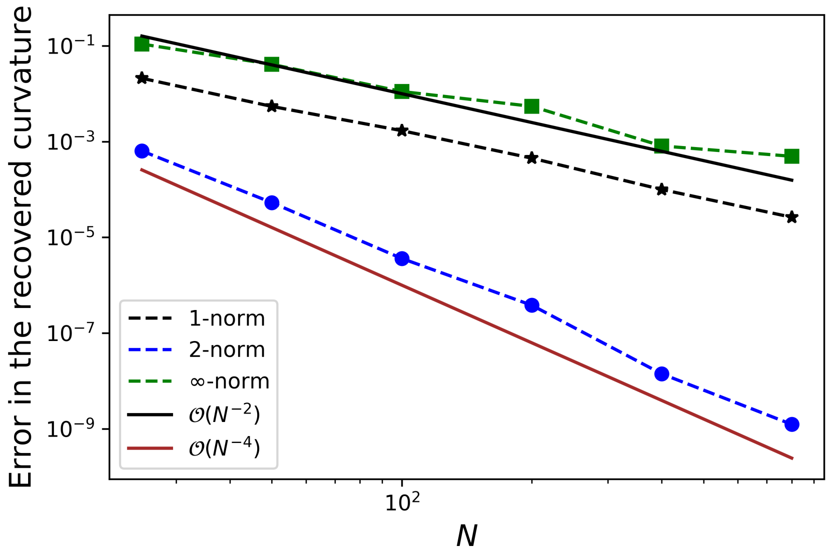


Figure 19: Log-log plot of the error in the recovered curvature for random sampled data with chordal parameterization ($a = 1$) for the circle.

**Function II: The Starfish**

For the starfish we will also start with a plot of the error in the recovered curvature for the uniform, centripetal and chordal parameterization methods measured with respect to the supnorm. This is shown in Figure 20. We see again the same pattern as in Figure 18. For $a = 0$, $a = \frac{1}{2}$ we see divergence in the recovered curvature, for $a = 1$ the error in the recovered curvature looks significantly better, so we will investigate this paramterization further.

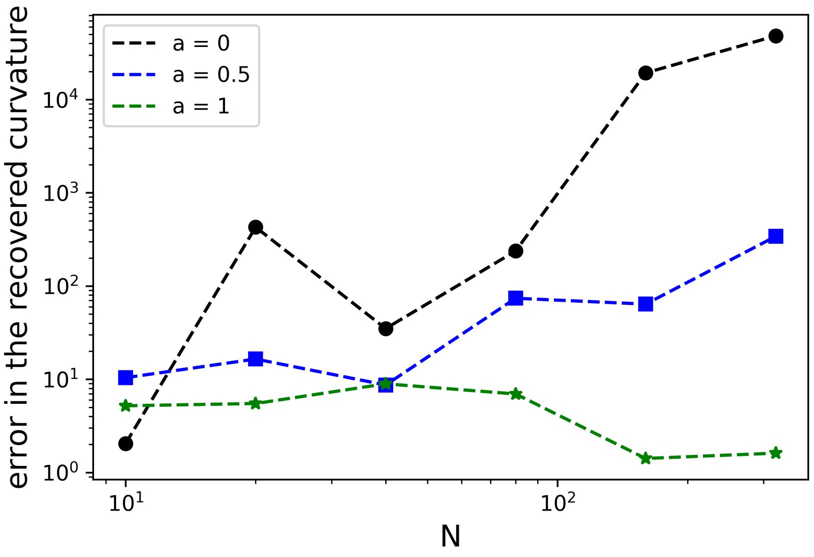


Figure 20: Log-log plot of the error with respect to the $\infty$-norm for the recovered curvature for different different values for $a$.

In Figure 21 we see the cubic interpolation attempts for the starfish with chordal spacing for the interpolation nodes $t_k$. For $N = 80$ the recovered curve seems to assume it's desired shape as displayed in Figure 22.

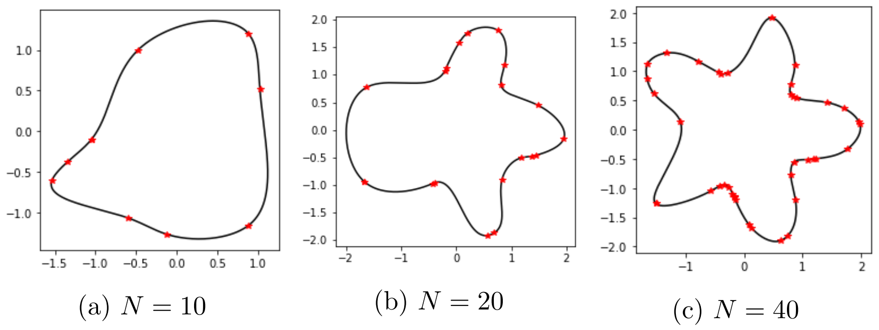

(a) $N = 10$ (b) $N = 20$ (c) $N = 40$

Figure 21: Plots of the cubic interpolated starfish with randomly generated data. With chordal spacing for the interpolation nodes ($a = 1$).

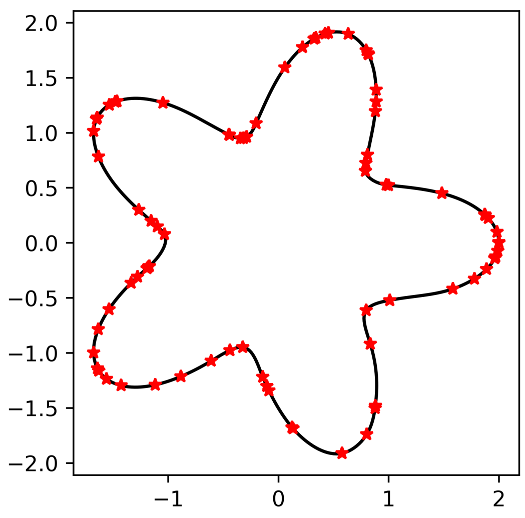


Figure 22: Cubic interpolated curve for $N = 80$ with chordal parameterization.

In Figure 23 and 24 we can see what happens with $||\mathbf{S}_c'||$ for different parameterizations. We also see that the speed for the uniform data oscillates as opposed to the constant speed which we saw earlier for the circle. Note that the the uniform and centripetal speeds change completely as we move from $N = 80$ to $N = 160$ but the uniform and chordal method remain the same. This gives more evidence for our conjecture that the problem with the the centripetal and chordal method is the non-converging speed vector.

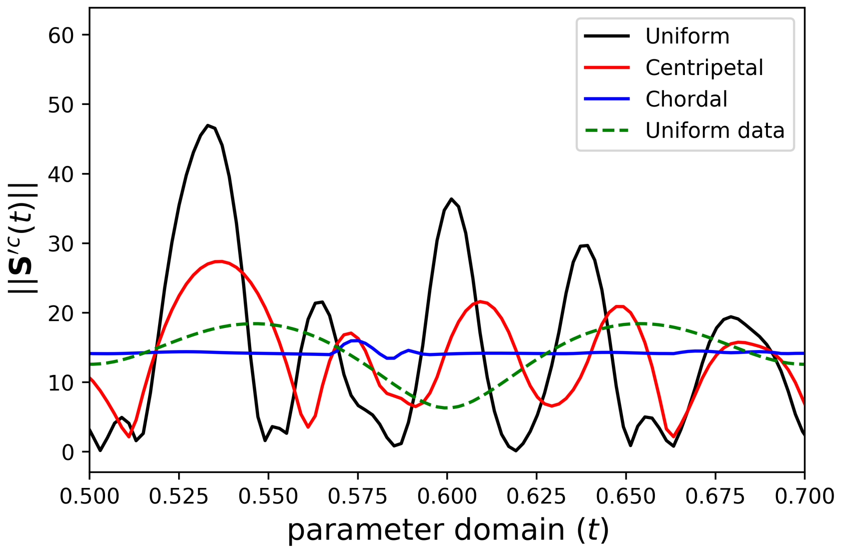


Figure 23: $||\mathbf{S}'_c||$ on the paramter domain $[0.5, 0.7]$ for the different types of parameterization: Uniform, Chordal and Centripetal. The green dotted line represents the uniform parameterization on uniform data. $N = 80$

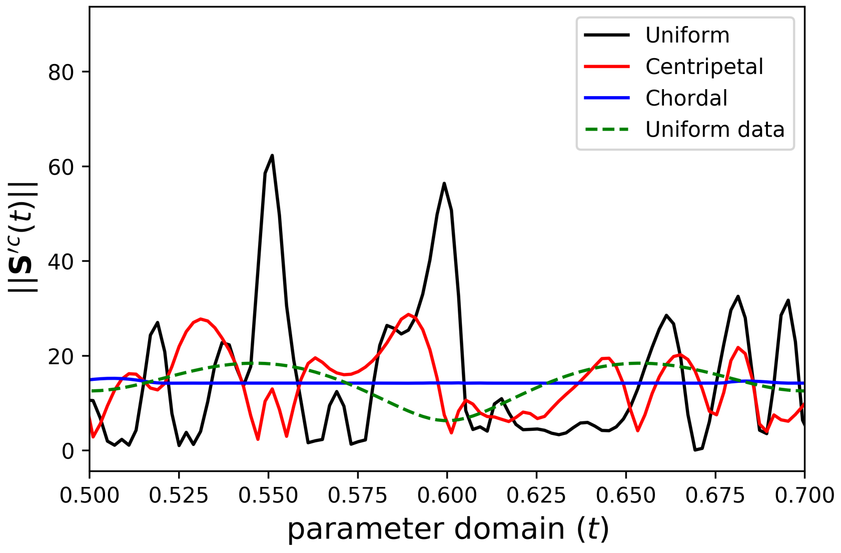


Figure 24: $||\mathbf{S}'_c||$ on the paramter domain $[0.5, 0.7]$ for the different types of parameterization: Uniform, Chordal and Centripetal. The green dotted line represents the uniform parameterization on uniform data. $N = 160$

Figure 25 displays the error in the recovered curvature with the chordal method. This is followed by Figure 26 where we can clearly see that the method starts to converge with order 2 and 4 with respect to their metrics.

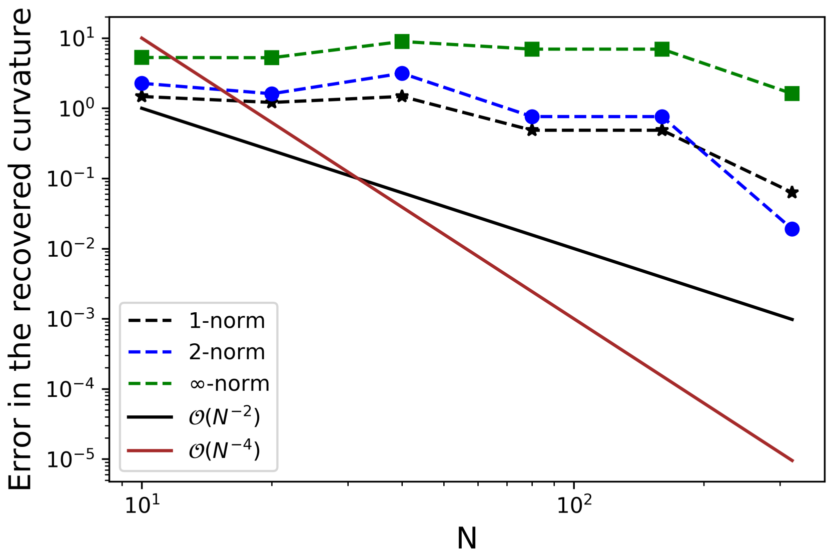


Figure 25: Error in the recovered curvature with cubic interpolation on random data for the starfish.

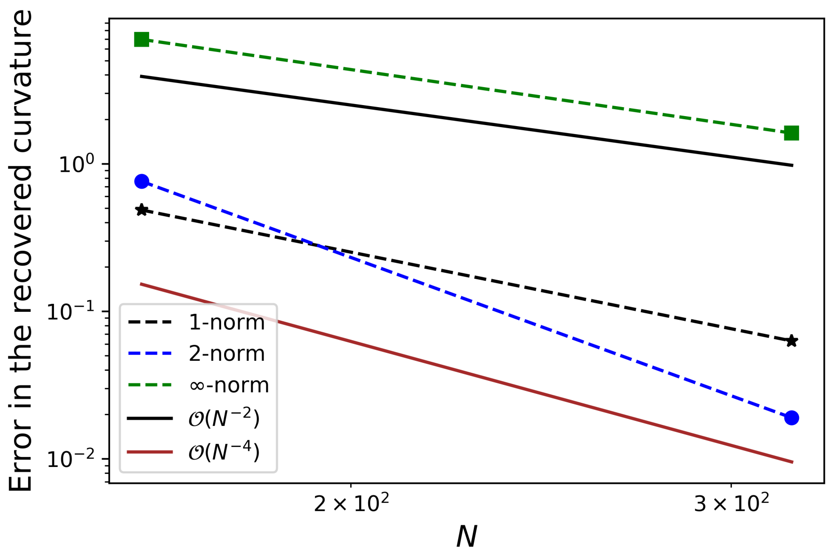
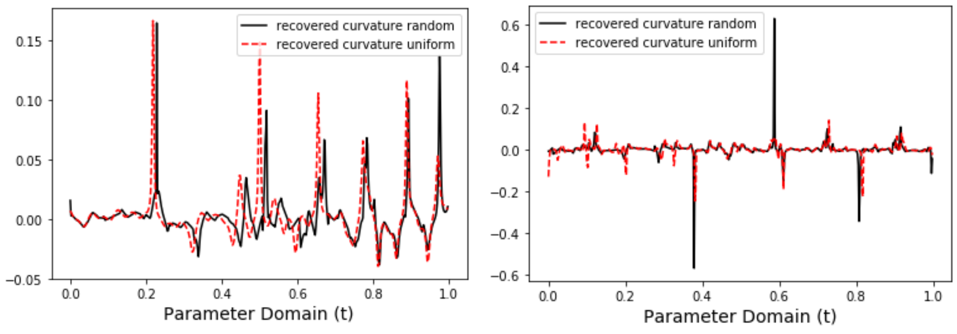


Figure 26: Close up on $N = 160$ and $N = 320$.

### Function III & IV: The Cat & The Camel

If we assume that the recovered curvature obtained for the uniform data is close to the real curvature of Function III and IV. Then we can use the uniformly recovered curvature as a reference for the random curvature recovery. From Figure 27 we can clearly see resembles between the two curves but still large errors are made. The fact that the uniform recovered curvature seems to be shifted compared to the random recovered curvature is caused by the change of the location of the parameterization nodes $t_k$.

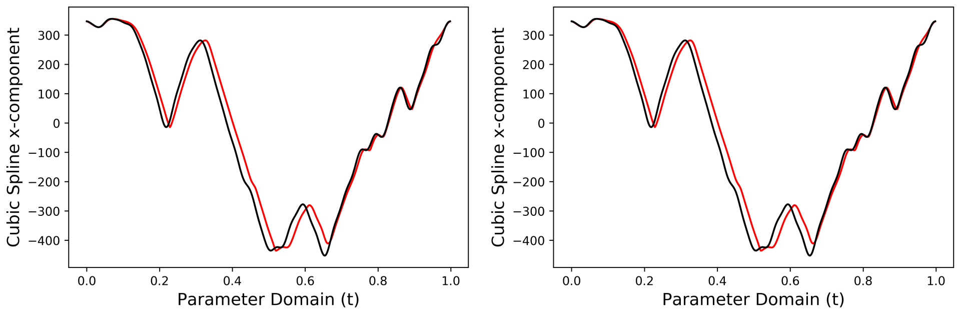


(a) Error in the recovered curvature for the sea star.

(b) Recovered curvature for uniform data and random data for the camel.

Figure 27

We can see the shift in the $x$- and $y$-component in Figure 28.



(a) Plot of the $x$-component $s^x$.

(b) Plot of the $y$-component $s^y$.

Figure 28: Small shift in the $x$ and $y$-component for $N = 160$.

## 3.2 Conclusions

From the simulations it is clear that the the chordal parameterization outperforms the uniform and centripetal parameterization on the random data sets for curvature recovery. Intuitively the chordal parameterization is a nice fix to the problems we encountered in Section 2.4.1 since it simply states: if two successive data points $D_k$ and $D_{k+1}$ are far away from each other measured by the chord connecting the two, place the corresponding parameterization nodes $t_k$ and $t_{k+1}$ farther away from each other and vice versa reducing the distortions in the recovered curvature. The reason that the other two methods don't work is probably caused by the not converging $||\mathbf{S}'(t)||$ we saw that even for high values $N$ we still have large erratic oscillations. The only downside to the chordal parameterization is that the linear approximation of arclength gets progressively worse at points where the curvature gets larger, this explains that the curvature recovery works better for the circle than for for the starfish, since the circle's curvature is constantly 2/3 and the curvature for the starfish can peak even to -10. In Figure 26 we saw some relatively large reduction in the error of the curvature, this can be explained with the same reasoning. Now the distance between the data points has become on average sufficiently small such that the chordal method approximates the actual underlying curve better.

# 4 Curvature recovery with quintic splines

## 4.1 Picking a spline of higher order

As discussed in Chapter 2 we want to have that $k_s(t) \in C^2[0,1]$. If we look at the curvature given by Equation (1) we see that if $x$, $y$ are in $C^4[0,1]$ then by the fact that compositions of continuous functions are again continuous and compositions of differentiable functions are again differentiable we have that $k(t) \in C^2[0,1]$. Which implies that $s^x$, $s^y$ should be in $C^4[0,1]$. The lowest order spline that guarantees this is of degree 5 (quintic).

## 4.2 Splines of degree 5 (periodic)

The conditions we impose on a spline of order 5 are similar as the ones used for the cubic splines. We only add $2(n-1)+2 = 2n$ conditions to obtain the required smoothness property such that $s \in C^4[0,1]$. And of course the piece-wise polynomials are now of order 5 instead of order 3.

a On each sub interval $[t_k, t_{k+1}]$, $s$ is a 5th degree polynomial $s_k, k = 0, \dots n-1$.

b The values at the nodes satisfy $s(t_k) = f(t_k),\ \text{for } k = 0, \dots, n$
Note: Since we are interpolating a *periodic* function we automatically have $s(0) = f(t_0) = f(t_n) = s(1)$.

c The smoothness and continuity conditions

$$\begin{aligned}
s_k(t_{k+1}) &= s_{k+1}(t_{k+1}),\ \text{for } k = 0, \dots, n-2,\\
s'_k(t_{k+1}) &= s'_{k+1}(t_{k+1}),\ \text{for } k = 0, \dots, n-2,\\
s''_k(t_{k+1}) &= s''_{k+1}(t_{k+1}),\ \text{for } k = 0, \dots, n-2,\\
s^{(3)}_k(t_{k+1}) &= s^{(3)}_{k+1}(t_{k+1}),\ \text{for } k = 0, \dots, n-2,\\
s^{(4)}_k(t_{k+1}) &= s^{(4)}_{k+1}(t_{k+1}),\ \text{for } k = 0, \dots, n-2.
\end{aligned}$$

d The periodic boundary conditions

$$\begin{aligned}
s_0(t_0) &= s_{n-1}(t_n),\\
s'_0(t_0) &= s'_{n-1}(t_n),\\
s''_0(t_0) &= s''_{n-1}(t_n),\\
s^{(3)}_0(t_0) &= s^{(3)}_{n-1}(t_n),\\
s^{(4)}_0(t_0) &= s^{(4)}_{n-1}(t_n).
\end{aligned}$$

The standard form for each polynomial on an interval $[t_k, t_{k+1}]$ with $k = 0, \dots, n$ is,

$$s_k(t) = \alpha_k(t-t_k)^5 + \beta_k(t-t_k)^4 + \gamma_k(t-t_k)^3 + \delta_k(t-t_k)^2 + \epsilon_k(t-t_k) + \zeta_k \quad (30)$$

As with the cubic spline the constant terms $\zeta_k$ can directly be obtained from condition b, we find

$$\zeta_k = f(t_k) \text{ for } k = 0, \ldots, n-1$$

To find a nice algorithm as for the cubic spline in Chapter 3 will be quite an algebraically demanding task. Since our main goal is curvature recovery and not algorithm optimisation we will directly substitute Equation (30) into the conditions mentioned in point c and d to retrieve the coefficients for the $s_k$.
We define again $h_k = t_{k+1} - t_k$ and $f(t_k) = f_k$ for more clarity in the notation. We will get the following set of equations from condition c,

$$\begin{aligned}
\alpha_k h_k^5 + \beta_k h_k^4 + \gamma_k h_k^3 + \delta_k h_k^2 + \epsilon_k h_k &= f_{k+1} - f_k \text{ for } k = 0, \ldots, n-2, \\
5\alpha_k h_k^4 + 4\beta_k h_k^3 + 3\gamma_k h_k^2 + 2\delta_k h_k + \epsilon_k - \epsilon_{k+1} &= 0 \text{ for } k = 0, \ldots, n-2, \\
20\alpha_k h_k^3 + 12\beta_k h_k^2 + 6\gamma_k h_k + 2(\delta_k - \delta_{k+1}) &= 0 \text{ for } k = 0, \ldots, n-2, \\
60\alpha_k h_k^2 + 24\beta_k h_k + 6(\gamma_k - \gamma_{k+1}) &= 0 \text{ for } k = 0, \ldots, n-2, \\
120\alpha_k h_k + 24(\beta_k - \beta_{k+1}) &= 0 \text{ for } k = 0, \ldots, n-2.
\end{aligned}$$

And the final 5 equations are obtained from point d,

$$\begin{aligned}
\alpha_{n-1}h_{n-1}^5 + \beta_{n-1}h_{n-1}^4 + \gamma_{n-1}h_{n-1}^3 + \delta_{n-1}h_{n-1}^2 + \epsilon_{n-1}h_{n-1} &= f_0 - f_{n-1}, \\
5\alpha_{n-1}h_{n-1}^4 + 4\beta_{n-1}h_{n-1}^3 + 3\gamma_{n-1}h_{n-1}^2 + 2\delta_{n-1}h_k + \epsilon_{n-1} - \epsilon_0 &= 0, \\
20\alpha_{n-1}h_{n-1}^3 + 12\beta_{n-1}h_{n-1}^2 + 6\gamma_{n-1}h_{n-1} + 2(\delta_{n-1} - \delta_0) &= 0, \\
60\alpha_{n-1}h_{n-1}^2 + 24\beta_{n-1}h_{n-1} + 6(\gamma_{n-1} - \gamma_0) &= 0, \\
120\alpha_{n-1}h_{n-1} + 24(\beta_{n-1} - \beta_0) &= 0.
\end{aligned}$$

So to summarize we have to solve a non-singular $5n$ by $5n$ linear system to retrieve all the coefficients for the $n$ polynomials.

## 4.3 Performance analysis: quintic spline curvature recovery on uniformly generated test data

This section presents some results obtained by the quintic spline interpolation method. There will first be a section on curvature recovery with uniform nodes (to test if everything behaves nicely). Followed by a section on the randomly generated data, where we also compare the performance of the quintic spline to the cubic spline. All the simulations were executed with Python.

### Function I, the circle

In Figure 29 we see that the quintic spline interpolation gives the expected result, a smooth circle trough the data points.

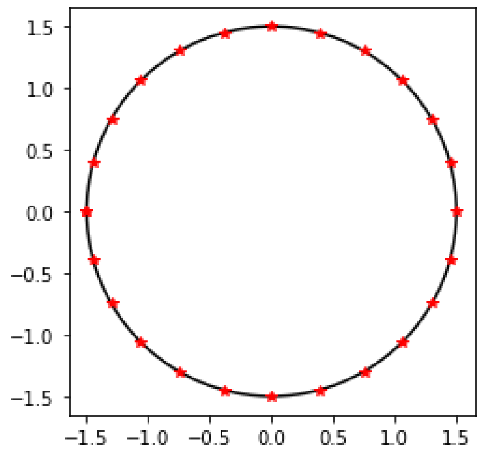


Figure 29: Plot of the quintic-spline interpolation of Function I with $N = 25$ data points. The red stars represent the data points $\{(x_i, y_i)\}_{i=0}^{n=24}$, and the black curve represents $\mathbf{S}_q(\mathrm{t})$ with $t \in [0, 1]$.

For higher values of $N$ we get exactly the same picture (only with more data points of course).
Figure 30 displays a plot of the recovered curvature $k_{\mathbf{s}}(t)$, on the interval $[0, 0.1]$. Just like in the recovered curvature with cubic spline we see oscillatory behaviour in the curvature, only now the oscillation is much smaller (note the scale on the $y$-axis). Also, we note that the oscillations are smooth as opposed to Figure 3b.

The plots of the error in the recovered curvature is displayed in Figure 31. The 1-and infinity norm decrease with the same order $\mathcal{O}(N^{-4})$. The 2-norm seems to converge even faster with order $\mathcal{O}(N^{-8})$. So the order of convergence has doubled compared to the cubic curvature recovery on uniform data.

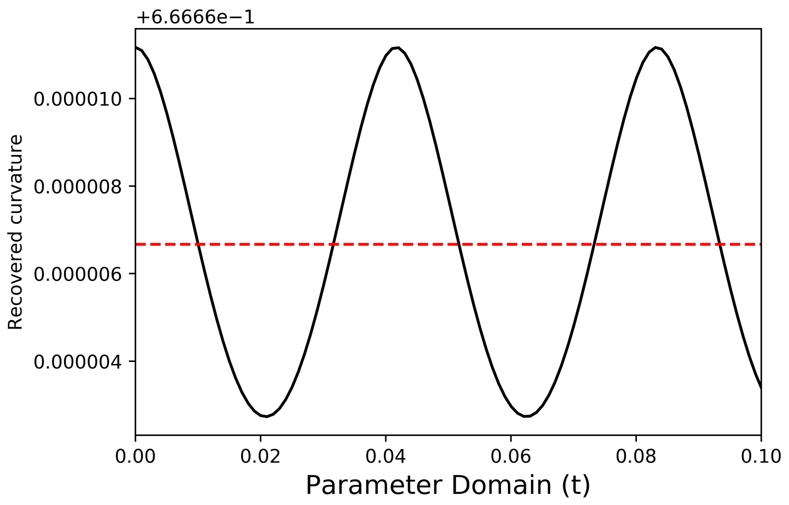


Figure 30: Recovered curvature with the method of quintic splines on the subset $[0, 0.1]$ of the parameter domain.

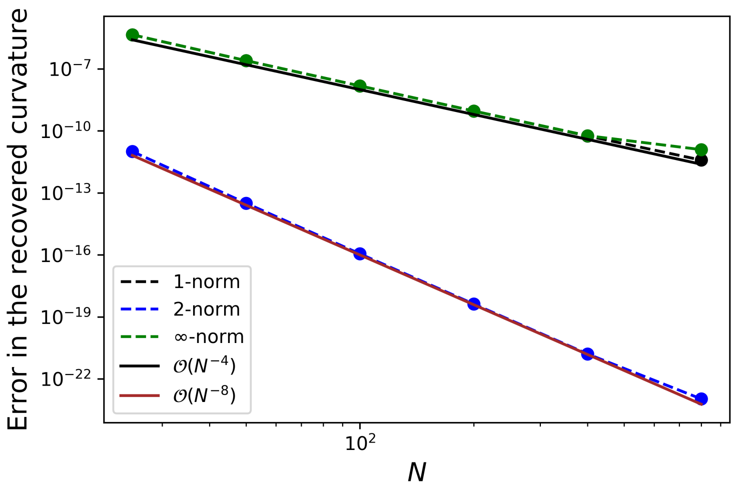
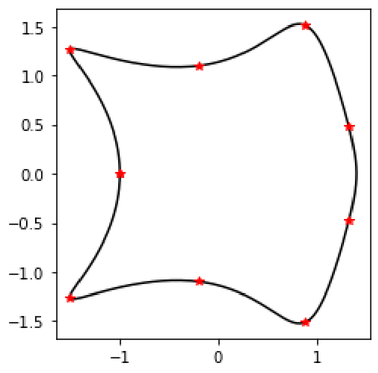


Figure 31: Log-log plot of the error of the recovered curvature for Function I, the circle. The black line represents the infinity norm, the blue line the 2-norm and the green line the 1-norm. The 1-norm and infinity norm coincide for the first 5 values of $N$.

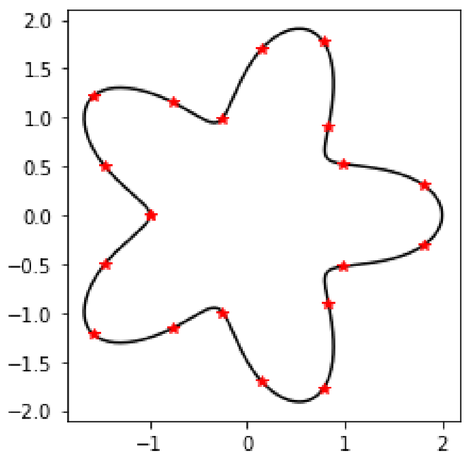
## Function II, the starfish

In Figure 32 we see the quintic spline interpolation for different amounts of interpolation nodes. After $N = 20$ the shape doesn't change visually anymore and we see the starfish emerge.

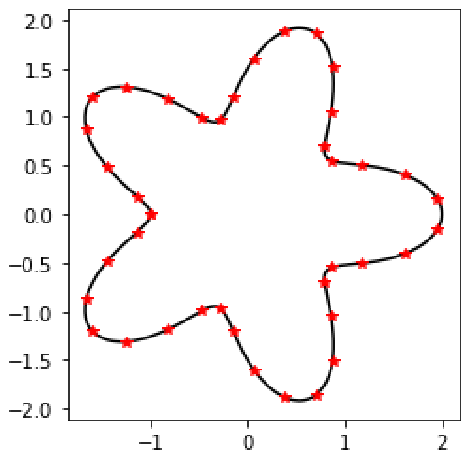
If we look at the plots of the recovered curvature itself then we get the



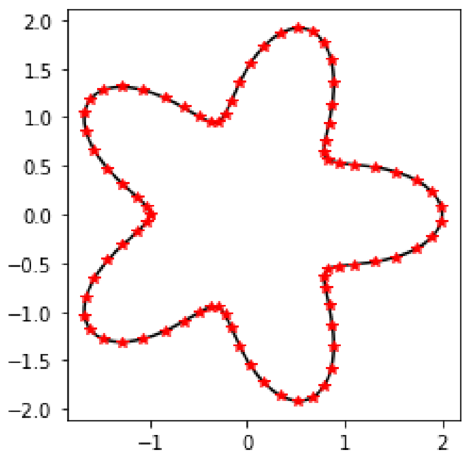
(a) $N = 10$



(b) $N = 20$



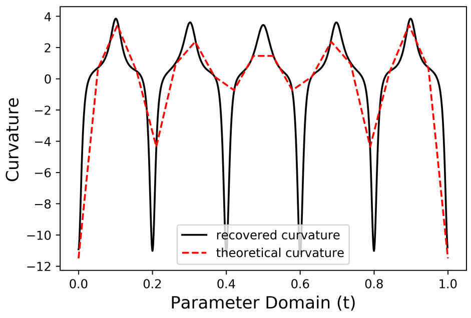
(c) $N = 40$



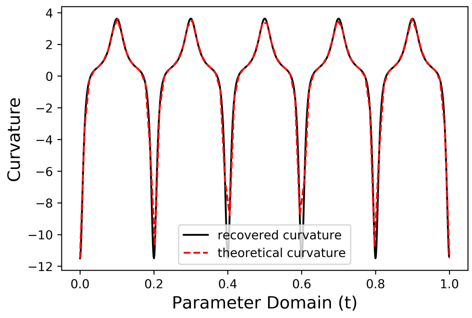
(d) $N = 80$

Figure 32: Plot of the quintic-spline interpolation of Function II with $N = 10$, $N = 20$, $N = 40$, $N = 80$ data points in Figures 32a, 32b, 32c, 32d respectively. The red stars represent the data points $\{(x_i, y_i)\}_{i=0}^{n}$, and the black curve represents $\mathbf{S}_q(\mathrm{t})$ with $t \in [0, 1]$.

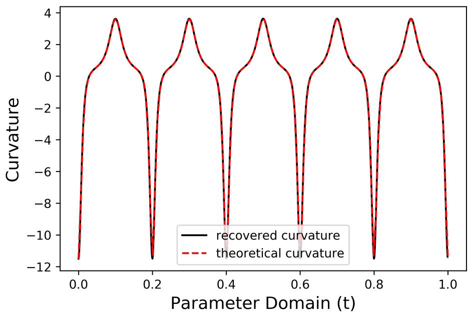
results displayed in Figure 33 which look to converge nicely to the theoretical curvature. If we go to the next figure, Figure 34, we see indeed that this is confirmed by the decreasing errors in the recovered curvature. It also indicates that the order of convergence for the sup and 1-norm is $\mathcal{O}(N^{-4})$ and for the 2-norm $\mathcal{O}(N^{-8})$. Thus far the results for the method with quintic splines seems to be analogous to the results obtained for the recovery method with cubic splines for the uniform data. Only the order of convergence has doubled.



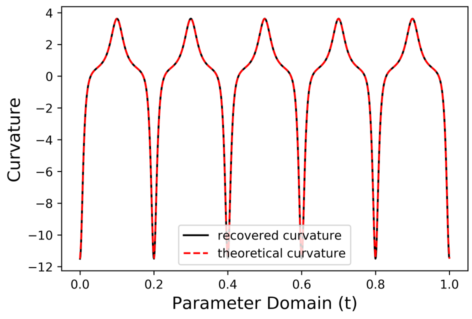
(a) $N = 50$



(b) $N = 100$



(c) $N = 200$



(d) $N = 400$

Figure 33: Plots of the recovered curvature and theoretical curvature for the starfish.

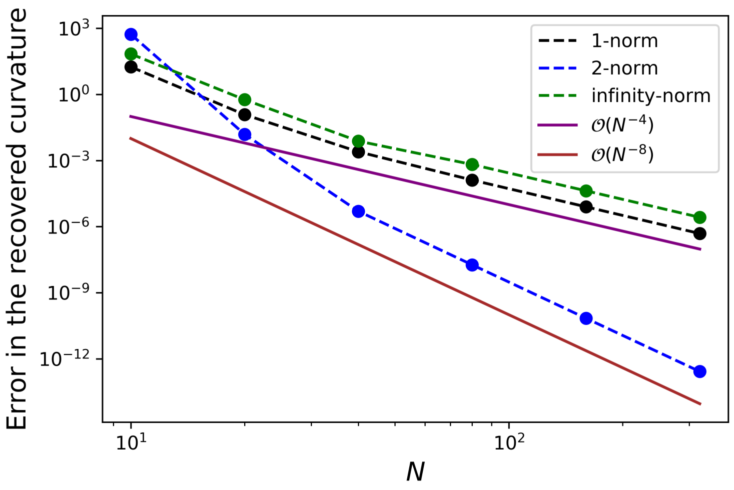
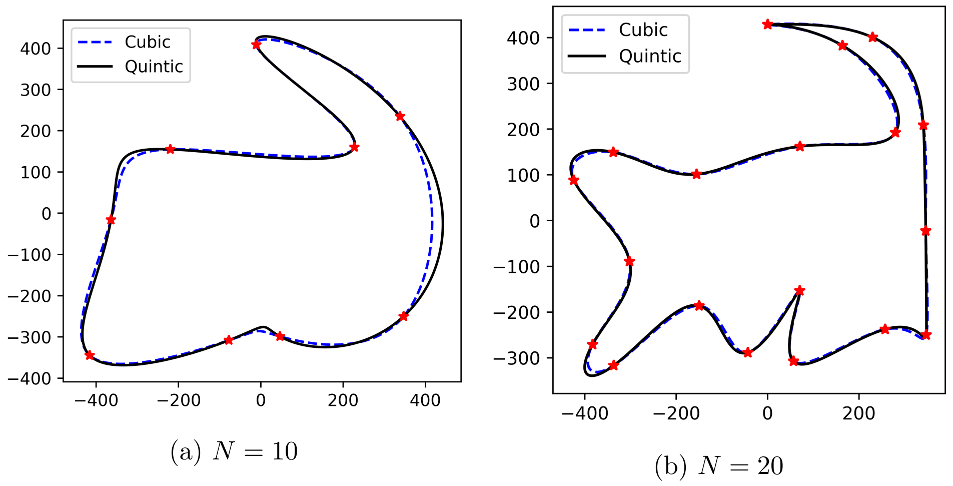


Figure 34: Log-log plot of the error of the recovered curvature for Function II, the starfish.

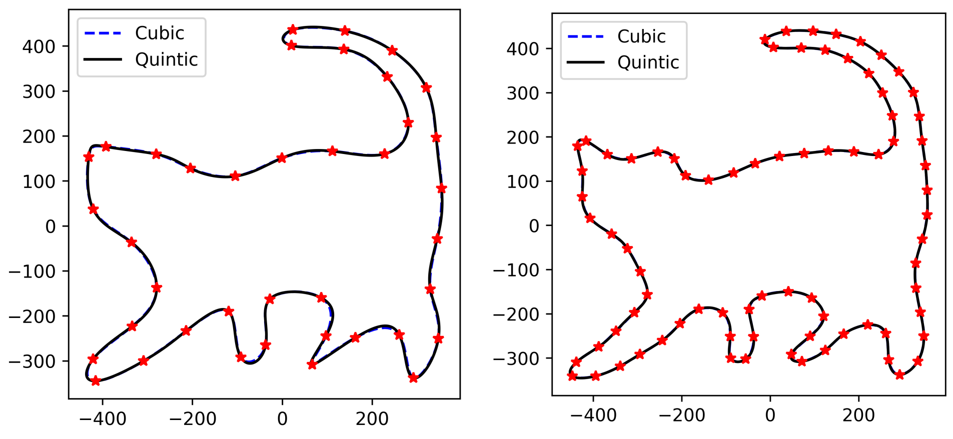
## The Cat & The Camel

In Figure 35 and 37 we see the recovered curves with quintic interpolation in black together with the cubic interpolated curve on the uniform data in dotted blue. For $N = 10$ and $N = 20$ there is still a significant visible difference between the two but after that the methods produce visually identical curves. This gives us strong evidence that the cubic and quintic interpolation converge to the original generating curve $\Gamma(t)$.



(a) $N = 10$

(b) $N = 20$

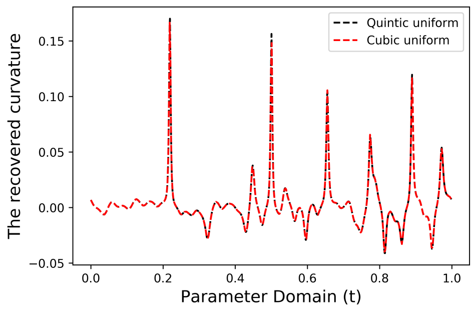


(c) $N = 40$

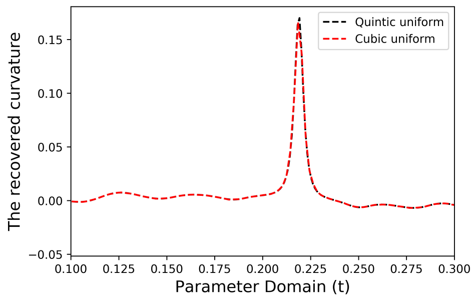
(d) $N = 80$

Figure 35: Comparison of the cubic and quintic interpolation of the cat on random data sets.

The results of the curvature recovery can be found in Figure 36. For $N = 160$ the recovered curvature nicely coincides with the cubic recovery. Also if we zoom in on the parameter domain we still see complete overlap of the two curves. This confirms that for the quintic and the cubic method on uniform data we have convergence in the recovered curvature.



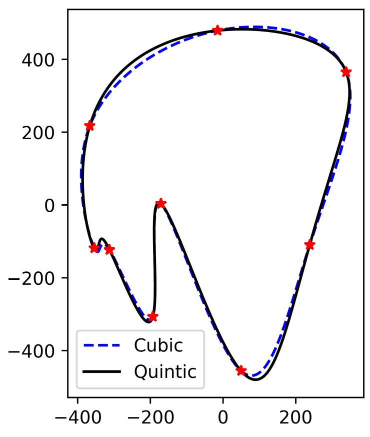
(a) Recovered curvature.



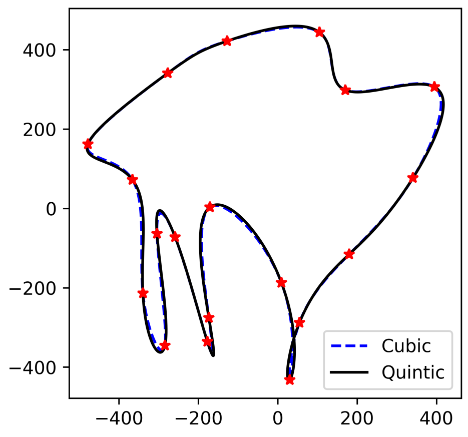
(b) Zoom on $[0.1, 0.3]$ parameter domain.

Figure 36: Comparison quintic and cubic recovered curvature.

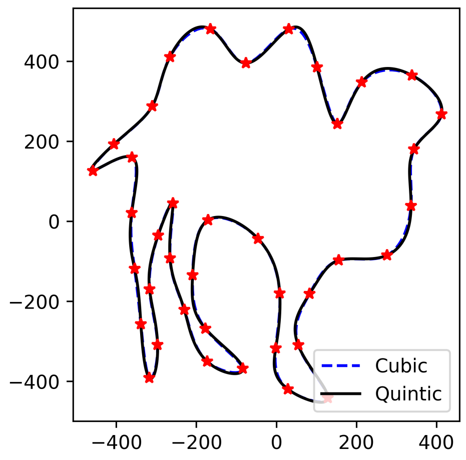
Figure 37 displays the comparison between the interpolation of the camel with the quintic splines and cubic splines. The dotted blue line represents the cubic curve while the black line represents the quintic curve.



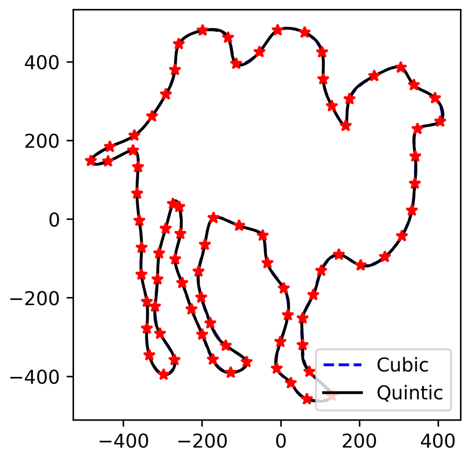
(a) $N = 10$



(b) $N = 20$



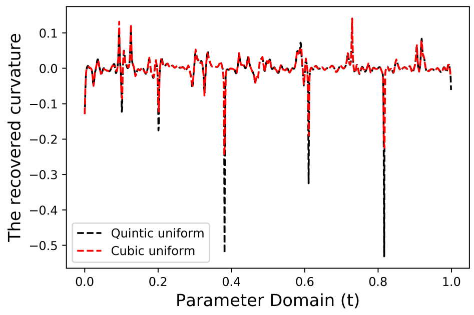
(c) $N = 40$



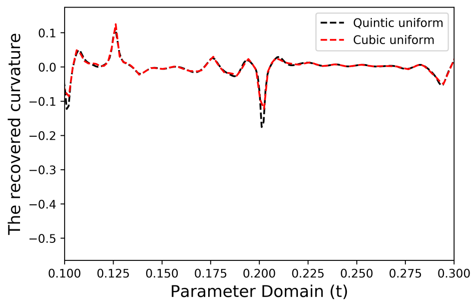
(d) $N = 80$

Figure 37: Comparison of the cubic and quintic interpolation on uniform data sets.

In Figure 38 we see the recovered curvature with quintic and cubic splines on uniform data sets with uniform parameterization. Note that the quintic recovery seems to pick up higher values in the curvature compared to the cubic curvature, but for the most part they are very close to each other.



(a) Recovered curvature.



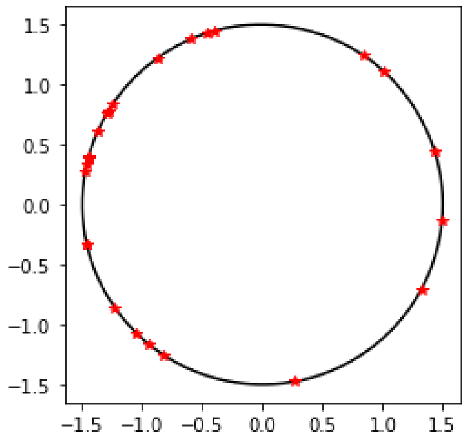
(b) Zoom on $[0.1, 0.3]$ parameter domain.

Figure 38: Comparison quintic and cubic recovered curvature for the camel.

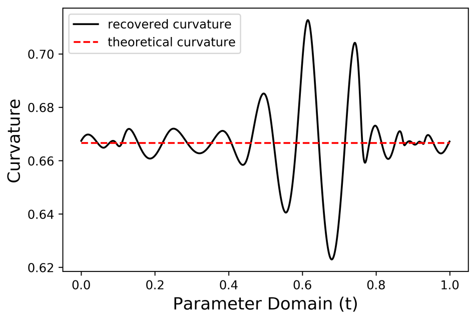
## 4.4 Performance analysis: curvature recovery with quintic splines on randomly sampled test data

### The Circle

As usual we start with our favorite geometric object. It is clear that we only need to consider the chordal spacing for the $t_k$, that is $a = 1$, since for the other two values of $a$ we know that we don't have convergence to the theoretical curvature. In Figure 39 we can see the interpolated curve and the recovered curvature retrieved with the quintic method for random data. The interpolation of the circle goes smoothly through the data points. The recovered curvature displays similar oscillations as for the uniform data only amplified on a subset of the parameter domain, but the order of magnitude of the amplitude is about the same. We also see that for random data we still have the desired smoothness in the curvature as can be seen in Figure 39b.



(a) Plot of the interpolated curve



(b) Recovered curvature

Figure 39: Results for quintic method for random data on the circle with $N = 25$ data points.

In Figure 40 the behaviour of the error in the recovered curvature is displayed. We see that the 1- and $\infty$-norm converge with $\mathcal{O}(N^{-2})$ and $\mathcal{O}(N^{-4})$. This is quite surprising since it shows that using quintic interpolation on randomly generated test data for the cricle doesn't seem to give a higher order of convergence to the theoretical curvature. This can be clearly seen in Figure 41, where we compare the cubic with the quintic curvature recovery. However we do notice that the quintic method always has a lower error in the recovered curvature than the cubic recovered curvature (with respect to their norms). So the quintic spline does perform better but there is no clear evidence that there is a higher order of convergence for the quintic spline in comparison with the cubic spline.

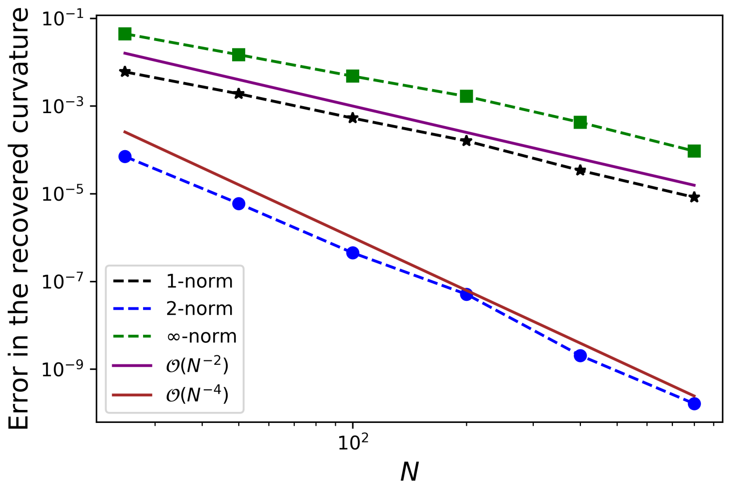


Figure 40: Log-log plot of the error in the recovered curvature for the circle for random data points and quintic curvature recovery.

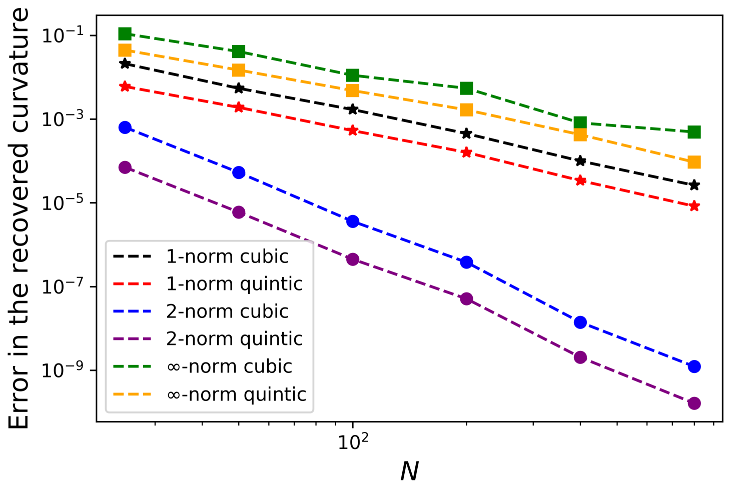
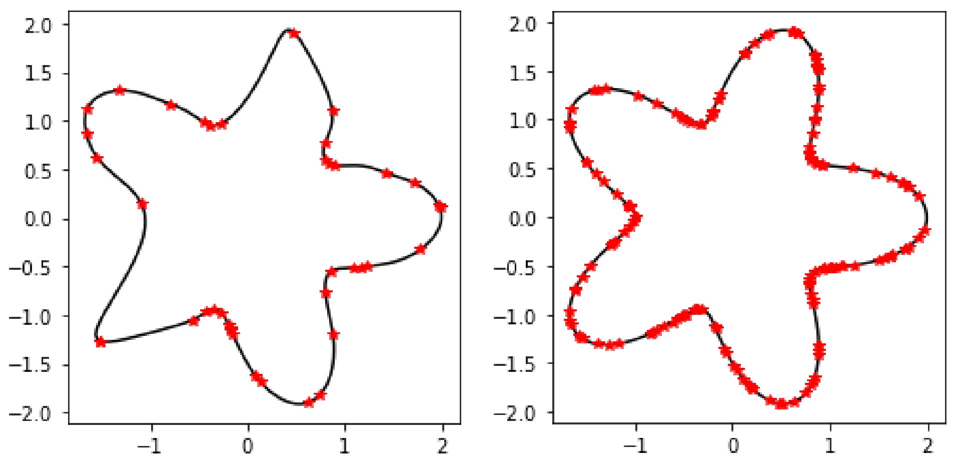


Figure 41: Comparison of the recovered curvature between the quintic method and the cubic method for random data and chordal parameterization.

**The starfish**

For the starfish we begin by looking at the convergence of the starfish visually in Figure 42. Note that only after $N = 40$ the starfish starts to take it's familiar shape. Also note that in Figure 43, after the third dot which corresponds to $N = 40$, the error seems to start to converge with a certain order. This can be understood if we look at Figure 42a, here we still see large distances between the data points on parts of the curve that have high curvature which cause the chordal method to approximate the actual curve poorly and this is why the errors remain big. If we now look at Figure 42b we see that on all the parts of the curve with high curvature are densely packed with data points which improve the chordal approximation by a lot. This explains the sudden decrease in the error of the recovered curvature in Figure 43.



(a) $N = 40$ (b) $N = 80$

Figure 42: plots of the quintic method for random data for the starfish.

In Figure 43 we also see the error in the quintic recovered curvature in comparison with the cubic method. The quintic spline interpolation always yields a lower error than the cubic method however their order of converge seem to be the same.

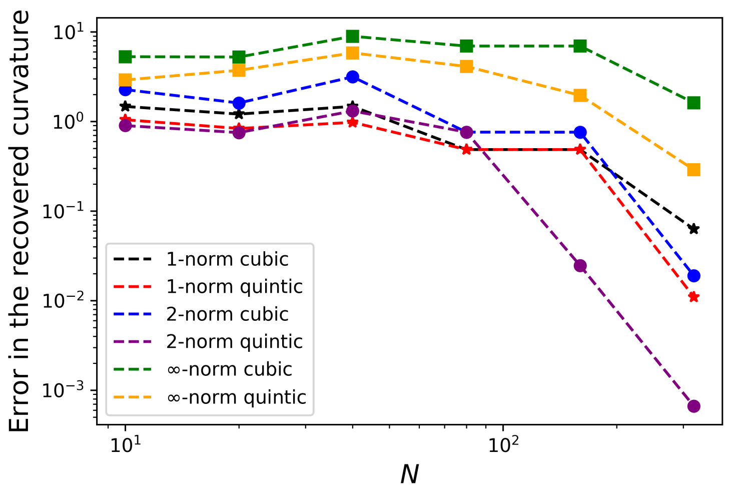
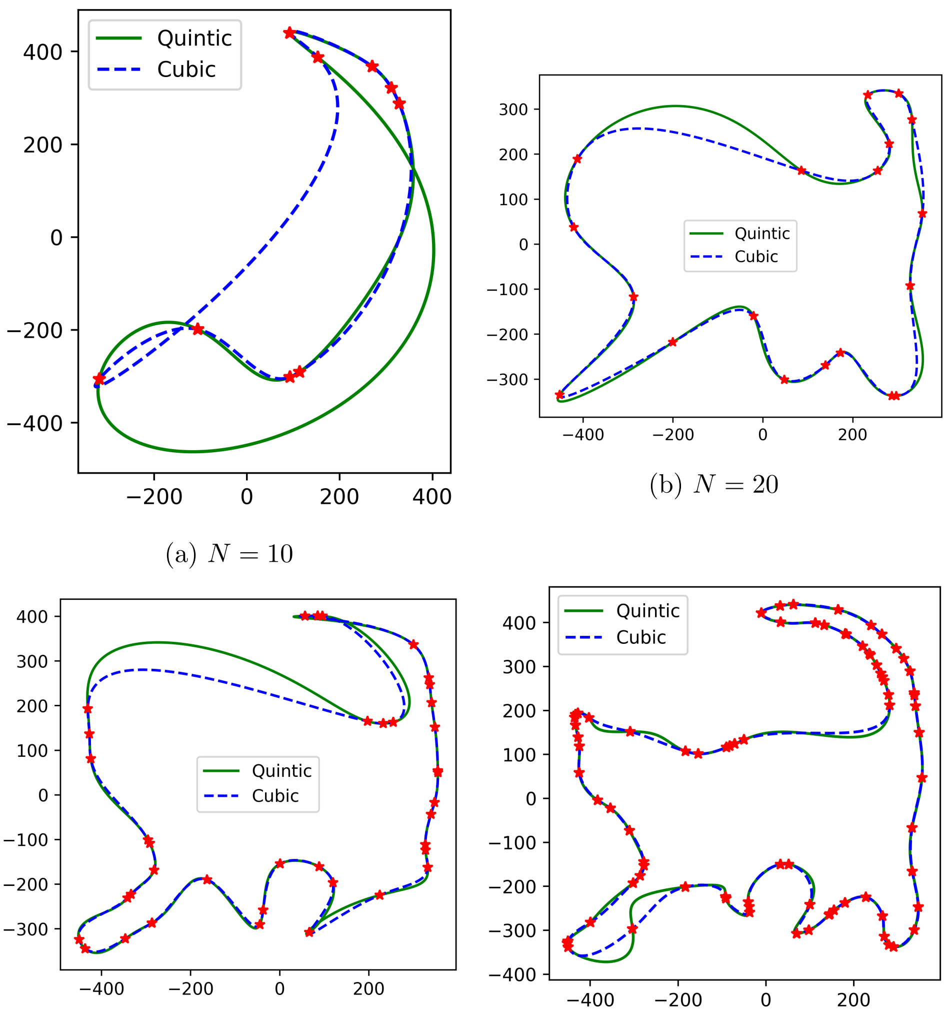


Figure 43: Comparison of the recovered curvature between the quintic method and the cubic method for random data and chordal parameterization.

## The Cat & The Camel

In this section we will compare the random quintic method with the random cubic method for the cat and the camel. In Figure 44 we can see the difference between the cubic interpolation on random data and the quintic interpolation. For small $N$ we see some big differences between the two recovered curves by the quintic and the cubic method. This difference becomes smaller as $N$ increases.



(a) $N = 10$

(b) $N = 20$

(c) $N = 40$

(d) $N = 80$

Figure 44: Comparison of the cubic and quintic interpolation of the cat on random data sets.

Note that if the difference between two successive data points $D_k$, $D_{k+1}$ is big the quintic spline seems to interpolate looser than the cubic interpolation. Look for example in Subfigure 44d we see that the left leg has more curvature for the quintic as for the cubic method. We see that in Figure 45 for $N = 160$ both interpolated curves nicely coincide.

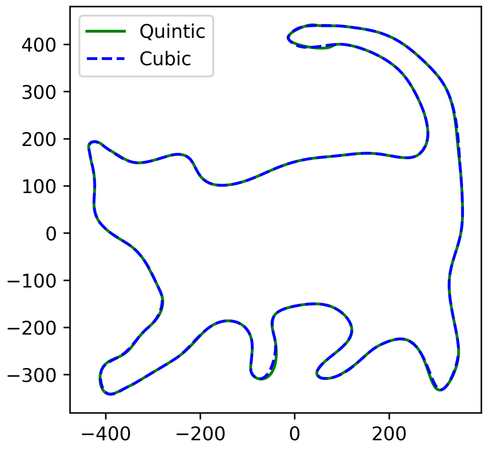
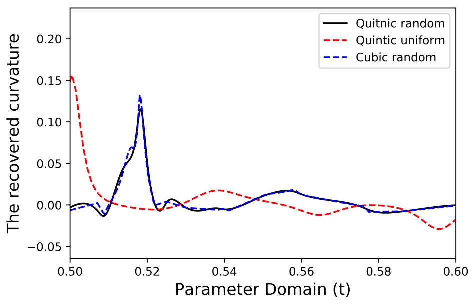
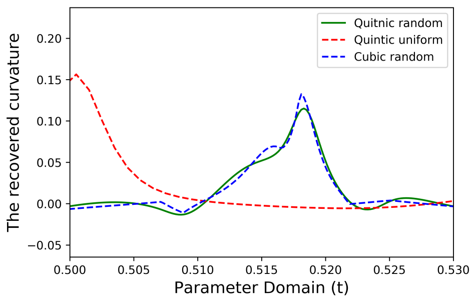


Figure 45: Plots of the quintic-spline interpolation for the cat on random data sets.

In Figure 46 the recovered curvature is displayed for random and uniform data with different methods. We also see the shift of the recovered curvature for the uniform parameterization on uniform data displayed by the dotted red line. Note that in Figure 46b we see that the quintic curvature on random data (the green line) is nicely smooth compared to the blue dotted cubic curvature.



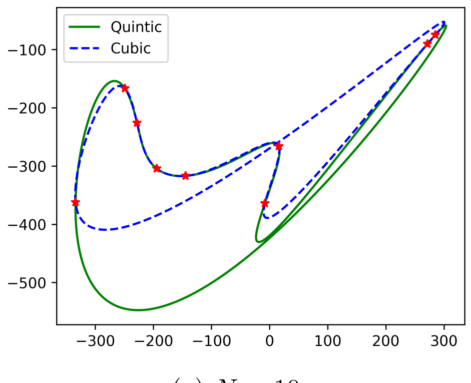
(a) Recovered curvature on the parameter domain $[0.5, 0.6]$



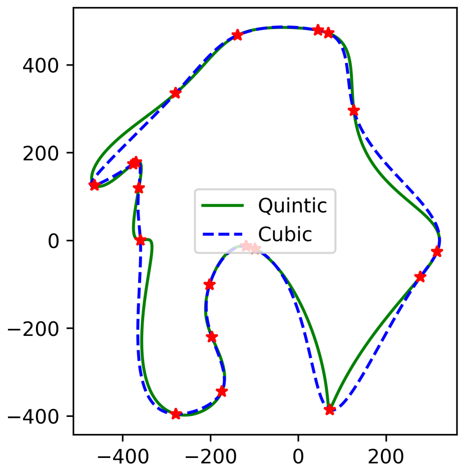
(b) Recovered curvature on the parameter domain $[0.50, 0.53]$

Figure 46: Comparison recovered curvature between quintic splies and cubic splines on random data and quintic splines on uniform data for $N = 160$.

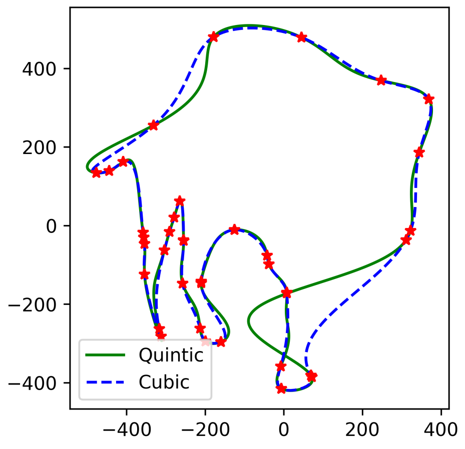
For the camel we find similar results for the interpolated curves, these are displayed in Figure 47.



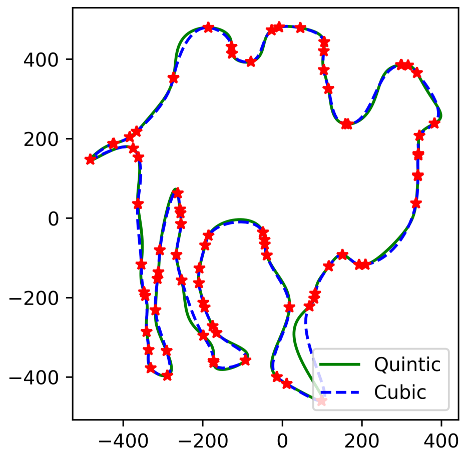
(a) $N = 10$



(b) $N = 20$



(c) $N = 40$



(d) $N = 80$

Figure 47: Comparison of the cubic and quintic interpolation of the cat on random data sets.

In Figure 48 we see the result for the highest amount of interpolation nodes $N = 160$. Again the cubic and quintic interpolation almost coincide in this case.

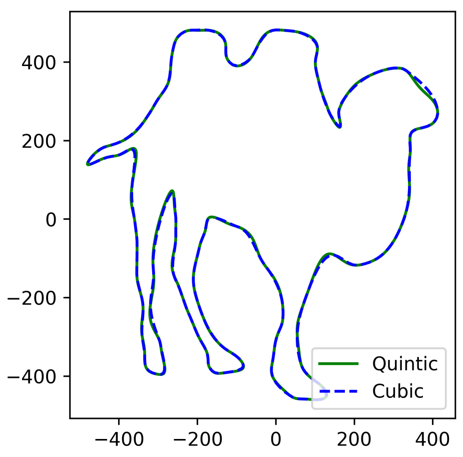
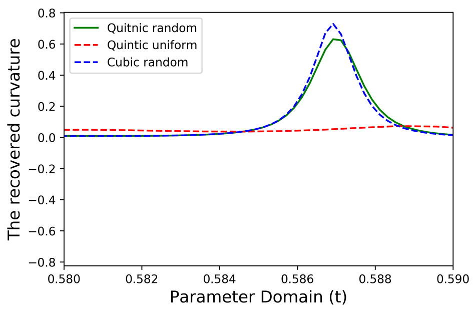
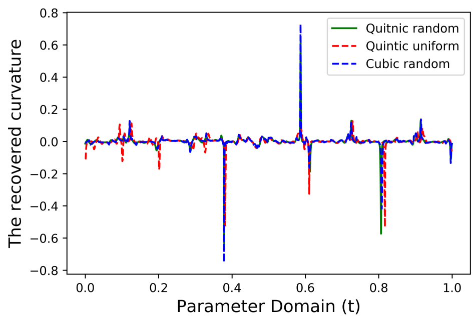


Figure 48: Recovered camel with $N = 160$ on the random data set.

We see again the shift in the curvature if we compare the chordal random data with the uniform paramterization on the uniform data in Figure 49. Note that again the green line is smooth compared to the blue dotted cubic curvature.



(a) Recovered curvature for different data and methods for the camel on the parameter domain $[0.58, 0.59]$.



(b) Recovered curvature for different data and methods for the camel on the parameter domain $[0, 1]$.

Figure 49: $N = 160$ interpolation nodes.

# 5 Conclusions, discussion and recommendations

## 5.1 Conclusions

The curvature recovery with quintic splines works universally better than with cubic splines for uniform data. This is due to the fact that we want to recover smooth curves and since quintic splines incorporate higher order smoothness conditions we have better modes of convergence. For the uniform data we collected some strong evidence that the cubic curvature recovery converges with order 2 and 4, and the quintic method with order 4 and 8 with respect to their relevant metrics. However the price that we pay is a higher order of computation time since the matrix we found for cubic interpolation has the almost-tridiagonal structure which can be solved in $\mathcal{O}(n)$ basic operations. For the matrix for quintic-splines we didn't find such a special form so here the computation time will be of order $\mathcal{O}(n^p)$ with $p \geq 2$ (depending on the algorithm that is used).

For the random data sets we first had to improve our method for placing the interpolation nodes $t_k$ because the uniform parameterization on randomly generated data points didn't show any promising results. We found that only with the chordal parameterization our simulation gave some decent to good results.

For the randomly generated data sets with the chordal paramterization we found the following results for the the quintic and cubic curvature recovery. For the circle we found evidence for order 2 and order 4 convergence (again with respect to the proper metrics) for both the cubic and quintic curvature recovery. This means that if we move to randomly generated data we lose the higher order of convergence in the curvature we had for the quintic method for the circle on uniform data. After inspecting the errors in the recovered curvature we saw that the error in the recovered curvature was smaller with quintic curvature recovery than with cubic curvature recovery but the order of convergence is the same. In the analysis of the starfish on randomly generated data we found that for the cubic and quintic curvature recovery more or less the same result. For small $N$ the error remained relatively big but after a sufficiently large $N$ the error started to decrease fast. As for the circle the quintic method did perform better but there is no indication that the quintic recovery has a higher order convergence than the cubic recovery. For the cat and the camel we saw that curvature recovery with quintic splines did indeed yield the smooth recovered curvature compared to it's cubic counterpart where the recovered curvature shows some sharp edges. So to summarize,

*the curvature recovery with splines performs good on uniform data, and* *decent to good on random data if the chordal parameterization is chosen and the chordal approximation for the arc length of the theoretical curve is sufficiently close to the theoretical arc length. The usage of quintic splines yields smaller errors in the recovered curvature than cubic splines but the improvement varies per data set.*

## 5.2 Discussion

We tested the curvature recovery methods on two types of datasets uniformly and randomly sampled from the original curve. These are two extreme cases of perfect structure in the data to no structure at all. Maybe it is possible to find something between these two types (for example uniform with noise) to make the test data fit some more realistic scenarios.
During the simulations on some of the random data sets we also looked at other values of $a$ then 0, $\frac{1}{2}$ and 1 where we started with $a_0 = 0$ and incremented the value of $a$ by $a_k = a_{k-1} + \frac{1}{10}$ to $a = 1$. For all the data sets and values that were tested $a = 1$ performed always better for the cubic and for the quintic curvature recovery. Since $a = 1$ is most likely the optimal value for curvature recovery we will give in Section 5.3 a potential method to improve the chordal method itself.

## 5.3 Recommendations for further research

### Richardson Extrapolation

For the camel and the cat we mostly used visual comparisons to see if the methods for curvature recovery worked. In successive research one might use Richardson's extrapolation [5] to get some practical quantitative error estimates for the recovered curvature for these geometric figures. This could however become quite tricky for the randomly sampled cases because for each increase in $N$ the points are freshly randomly sampled. As a suggestion to apply Richardson extrapolation to randomly sampled data sets one could start with a large amount of data points randomly sampled from the curve say $A_0 = \{D_i\}_{i=1}^{N}$ such that $N$ is a large power of 2. From this generated set one could sample at random $\frac{N}{2}$ points and create a data subset $A_1 = \{D_{i_j}\}_{j=1}^{N/2} \subset A_0$, and from $A_1$ a data set $A_2 \subset A_1 \subset A_0$ can be created with $N/4$ elements etc. Then Richardson extrapolation can be applied to these data sets for the cat and the camel to get some practical error estimates for the curvature recovery.

### Proving rigorously some convergence results for the curvature recovery

We collected evidence for the convergence of the recovered curvature to the theoretical curvature. As an idea for further research one could try to prove rigorously some of these convergence hypothesis for curvature recovery with splines, be it for quintic or the cubic method.

## Using 'Arc length parameteriztion'

In [8] a nice improvement is given for the chordal parameterization method to obtain higher order convergence to the theoretical curve for interpolation methods that are of higher degree than cubic interpolation. Instead of only using the chordal parameterization we take an extra step to improve the order of convergence. We start by using the chordal interpolation given by Equation (28) with $a = 1$ to retrieve the nodes $0 = t_0 < t_1 < \ldots < t_n = 1$ and apply cubic interpolation which will give our familiar cubic interpolated curve $\mathbf{S}^c(t)$ with $t \in [0,1]$. Now we can use the *arc length* of $\mathbf{S}^c(t)$ to determine our new parameterization nodes $\hat{t}_k$. Because we used the unit domain we take $\hat{t}_0 = t_0 = 0$ and $\hat{t}_n = t_n = 1$ and then normalize to have all $t_k$ correctly distributed in $[0,1]$ by

$$\hat{t}_k = \frac{L(\mathbf{S}^c|_{[0,t_k]})}{L(\mathbf{S}^c|_{[0,1]})} = \frac{\int_0^{t_k} ||(\mathbf{S}^c(u))'||du}{\int_0^1 ||(\mathbf{S}^c(u))'||du} \text{ for } k = 1, \ldots, n-1. \tag{31}$$

Now we can apply quintic interpolation on the *arc length parameterization* $0 = \hat{t}_0 < \hat{t}_1 < \ldots < \hat{t}_n = 1$. Since this yields a higher order of convergence to the theoretical curve, we except that it will also converge faster to the theoretical curvature. This scheme can be continued to gain even higher order of convergence according to Floater and Surazhky. For further research one might use this method to see if it indeed yields better results for the curvature recovery, especially for the random data sets.

## Curvature recovery for closed surfaces in $\mathbb{R}^3$

A natural question to ask is if it is possible to extend our method for curvature recovery from closed flat curves to closed surfaces in $\mathbb{R}^3$.
In this paper we essentially constructed a map $\mathbf{S} : [0,1] \to \mathbb{R}^2$ that tries to reconstruct the original curve with the help of spline interpolation, and from this map we can calculate the curvature. If we argue by analogy then for the case of a closed surface we need to reconstruct a map of the following form: $\mathbf{X} : [0,1] \times [0,1] \to \mathbb{R}^3$ so $\mathbf{X}(u,v) = (x(u,v), y(u,v), z(u,v))$ with $(u,v) \in [0,1] \times [0,1]$ from a set of data points $\{(x_i, y_i, z_i)\}_{i=0}^N$. It is not directly clear how to apply our usage of splines in this setting. In the literature *B-splines* are often mentioned for obtaining a surface from a set of data points, for example in Shene's online course notes [12].

## References

[1] S. Osher, J.A. Sethian (1988) *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations.* Journal of Computational Physics, 79(1), 12–49

[2] den Ouden D., Segal, A., Vermolen, F. J., Zhao, L., Vuik, C., & Sietsma, J. (2012). *Application of the level-set method to a mixed-mode driven Stefan problem in 2D and 3D.* Computing, 95(S1), 553–572

[3] Raees, F., van der Heul, D. R., & Vuik, C. (2015). *A mass-conserving level-set method for simulation of multiphase flow in geometrically complicated domains.* International Journal for Numerical Methods in Fluids, 81(7), 399–425.

[4] D. Den Ouden (2015) *Mathematical Modelling of Nucleating and Growing Precipitates:Distributions and Interfaces* 122-123

[5] C. Vuik, F.J. Vermolen, M.B. van Gijzen, M.J. Vuik. (2016) *Numerical Methods for Ordinary Differential Equations.* 20-23, 33-35

[6] A. Ambrosio. (2013) *Levy-Desplanques theorem.* Retrieved from `https://planetmath.org/levydesplanquestheorem`

[7] T. Chen, (1995) *Parallel Solvers for Almost-Tridiagonal Linear Systems.* 21

[8] M.S. Floater, T. Surazhky (2005) *Parameterization for curve interpolation.* Topics in Multivariate Approximation and Interpolation. 102-103, 108

[9] B. de Pagter, W. Groenevelt (2014) *Dictaat TW1070 Analyse 2.* 9, 18, 93, 169-171, 139-144

[10] E. T. Y. Lee (1989) *Choosing nodes in parametric curve interpolation.* Computer Aided Design. 363-370

[11] C.K. Shene *The Centripetal Method.* CS3621 Introduction to Computing with Geometry. Retrieved from `https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/INT-APP/PARA-centripetal.html`

[12] C.K. Shene *Surface Global Interpolation* CS3621 Introduction to Computing with Geometry. Retrieved from `https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/INT-APP/SURF-INT-global.html`