# Interpolation of Discretized Data

Milos Micik

June 14, 2022

## Abstract

Add abstract

# Chapter 1

# Introduction

Add intro
duction.

# Chapter 2

# Interpolation in 1D

In one dimensional interpolation we have decided to compare interpolation methods using Bezier curves and cubic polynomial splines. Both methods use piecewise polynomials to trace a curve defined by knot points and information about the first derivatives.

Possibly add a bit of theory to splines

In 1D, the interpolation task is nearly straightforward. The only issue that can arise is when the data to be interpolated is discretized. This could occur for example when collecting data from a measurement where the measuring device has a finite precision and can only detect change in the observed quantity in steps.

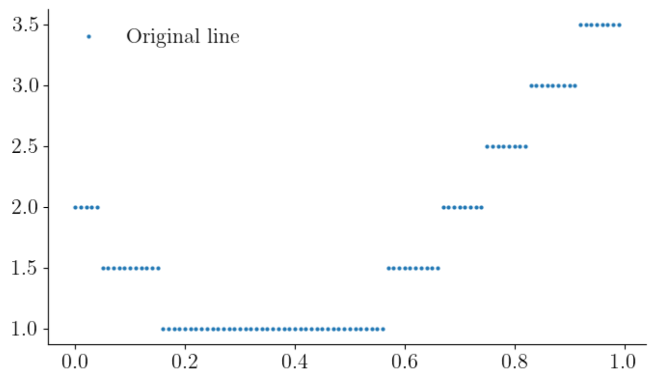

Figure 2.1: An example of discretized 1D data.

If we were to run the interpolation on such data without any adjustments, the polynomials would try to fit every single point in the collinear segments, producing a curve not too different from what we started with. Therefore in this case pre-processing of the data is required.

The pre-processing we have chosen assumes that the discretized data is obtained by rounding the real-data observation. For example, in Figure 2.1, the collinear segment from $x \approx 0.15$ to $x \approx 0.55$ is represented in the data with $y$ value 1.0, however by our assumption we know that the real values could have been anywhere in the interval $[0.75, 1.25)$. The pre-processing algorithm simply isolates a midpoint from a collinear region of points. Where the collinear region is a local maximum or minimum, we isolate two point instead, 1/3 and 2/3 along the collinear segment.

Using this algorithm, we can now interpolate any discretized curves in 1D. An example of it working in action can be seen in Figure 2.2:

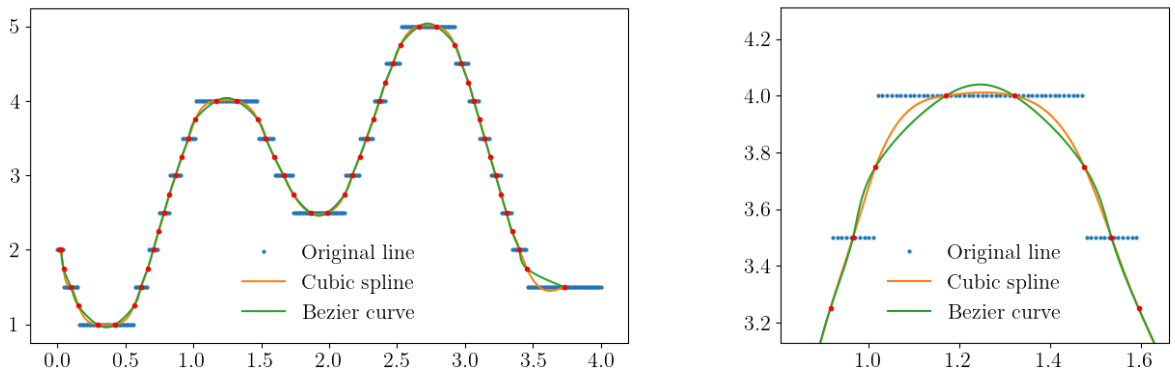

Figure 2.2: An example of curves interpolated from discretized 1D data.

Note the subtle difference between the Bezier curve and the cubic spline. In segments which are relatively straight, their shape is very similar. However, the difference at places where the curve takes sharper turn is more noticeable. The cubic spline seems to achieve much better results, introducing less irregularities and copying the original data curve more smoothly. This does not come as a surprise, as cubic splines are expected to have $C^2$ continuity, whereas Bezier curves are only $C^1$ continuous.

Chapter 3

# Interpolation in 2D

## 3.1 Parametric Curves

Interpolating parametric curves in 2D (technically in any dimension) is very similar to interpolating in 1D. It might require different functions when implementing these in a programming language, but the algorithm is the same - if the data is discretized, we need to first isolate points from collinear segments.

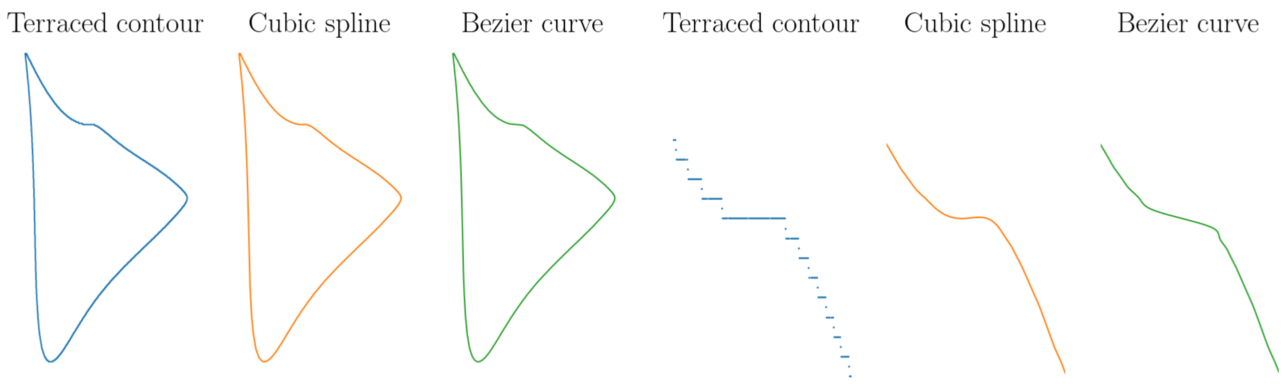

Figure 3.1: An example of parametric curves interpolated from discretized parametric 1D curve in 2D.

The same observation about the continuity holds in this case - cubic splines produce curves which look more natural.

## 3.2 Countour lines

Interpolating contour lines of 2D functions can be done by representing the contours as parametric curves. As an example, we will use digital elevation maps of regions from Scotland.

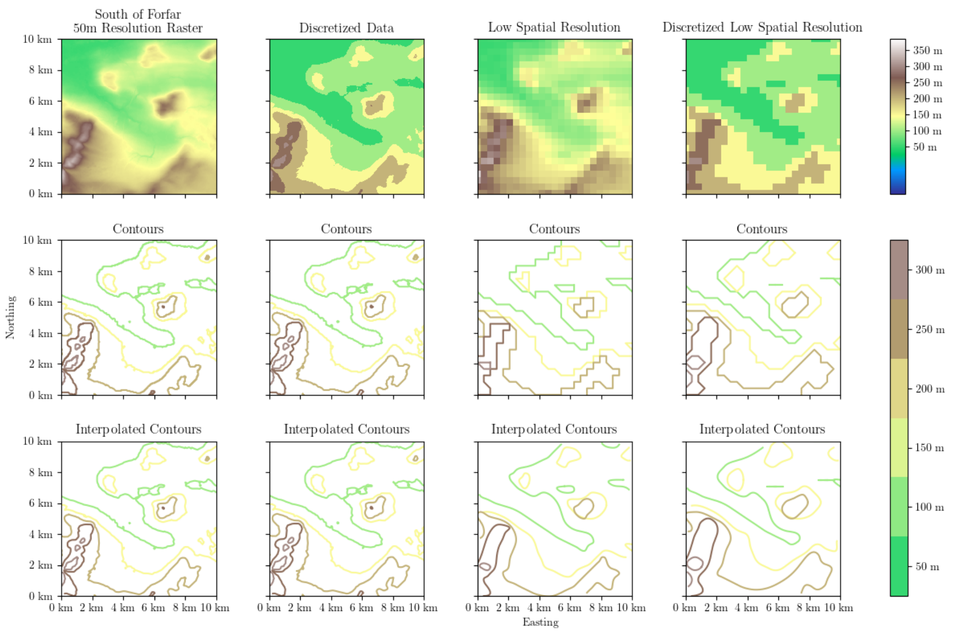

Figure 3.2: Contour interpolation from data with various resolution.