Spring 2018

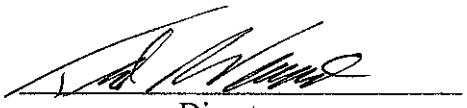
More than one way to skin a cat: Interpolation techniques in one-dimension

Jesica Bauer
Carroll College, Helena, MT

SIGNATURE PAGE

This thesis for honors recognition has been approved for the

Department of Mathematics .



Director

4-27-18

Date

Ted Wendt

Print Name

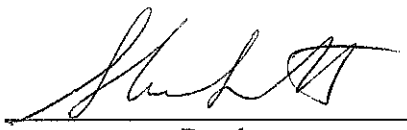
Reader

4-27-18

Date

Eric R. Sullivan

Print Name



Reader

4-27-18

Date

Shawn Scott

Print Name

More than one way to skin a cat:

Interpolation techniques in one-dimension

Jesica Bauer

May 1, 2018

Abstract

In this paper, we discuss and develop several one-dimensional interpolation techniques. Interpolation is a process for generating functions that pass through specified points in space. In general, given a set of points $P = \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, interpolation provides a function $f(x)$ such that $f(x_i) = y_i$ for $i = 0, \ldots, n$. We start by discussing common techniques used for interpolation, including polynomial, piecewise linear, cubic splines, and Bézier curves. We also develop two new interpolation techniques: one based on a refinement of quadratic interpolation and the other based on bending properties of physical materials. We examine quantitative and qualitative errors between the existing methods and our new techniques. Finally, we discuss how the new techniques could be generalized and extended into higher dimensions.

Contents

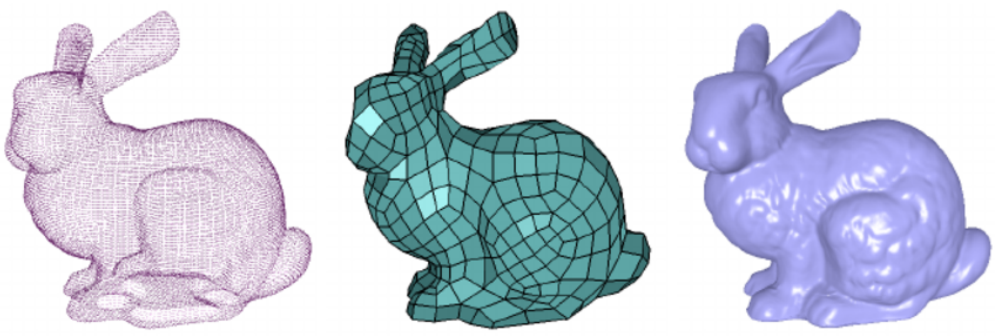


Figure 1: An example of skinning a point cloud of a rabbit. Left to right: original point cloud, control mesh, and interpolation result [1].

1 Introduction

Modern computers are able to create complex imagery with only a small set of information. For example, the fonts on your computer are saved as a set of points and the computer is told how to connect them. Many 3D animations are generated the same way, where the animation starts as a collection of points before the rest of the shape is systematically filled in. The challenge is to take a handful of points and turn them into something that doesn't look like it was drawn from the points.

The general idea of interpolation is, given a finite set of points $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, find a function f such that $f(x_i) = y_i$ for all $i = 0, 1, \ldots, n$. An interpolation technique can force the function to follow other constraints, such as continuity or restrictions on the derivatives. The restrictions that we force on the system alter how the final interpolation behaves.

In this paper, we will discuss several interpolation techniques which provide mathematical functions that describe how an arbitrary set of points in 2-dimensions are to be connected. While we can do these calculations by hand, this paper focuses on derivations of techniques and their computer implementations. We discuss polynomial and piecewise polynomial interpolation, as well as the method of Bézier curves. We also create two novel approaches to interpolation. One method, the "quadrubic", is a two-stage hybrid model that uses elements of both quadratic and cubic splines. Our other method is based upon the physical construction of cubic splines, utilizing the beam bending problem commonly studied in statics.

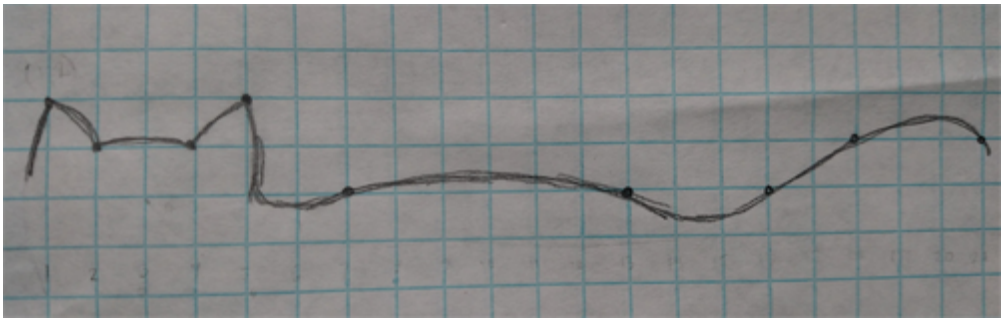


Figure 2: The original hand-drawn Carroll Cat.

2 1-Dimensional Interpolation

The simplest setting for interpolation is one-dimension. The dimension of the interpolation is usually associated with the domain of the interpolating function. Single dimensional interpolation is how we determine how to connect a series of ordered-pairs to create a curve. Depending on the restrictions, we could end up with a single high-degree polynomial, a "connect-the-dots" method, or a collection of smooth curves that go through all of the points.

In this section, we discuss four common methods of interpolation: polynomial, piecewise linear, cubic splines, and Bézier curves. We describe how to generate the curves and how they perform against each other to re-create two test functions: a sine curve and the top-half of a cat (the Carroll Cat) shown in Figure 2.

2.1 Polynomial Interpolation

Given a finite list of points, $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, there are several common techniques for generating a interpolating polynomial. For this paper, we discuss the Lagrange and Newton form of the interpolation polynomial to give the reader some insight on how the interpolation process works.

The Lagrange form of the interpolation polynomial utilizes weights $\varphi_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$. Notice that $\varphi_i(x_i) = 1$ and $\varphi_i(x_j) = 0$ when $i \neq j$, ensuring that polynomial p interpolates every point. That is, $p(x_i) = y_i$ for every $(x_i, y_i) \in P$. Then the Lagrange form of the interpolation polynomial is expressed as

$$p(x) = \sum_{i=0}^{n} y_i \varphi(x) = \sum_{i=0}^{n} y_i \left(\prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \right) \tag{1}$$

where $(x_i, y_i) \in P$.

Example 2.1. Consider $P = \{(1, 2), (2, 3)\}$. Then the Lagrange interpolation polynomial is

$$p(x) = 2 \cdot \frac{(x-2)}{(1-2)} + 3 \cdot \frac{(x-1)}{(2-1)} = x + 1.$$

However, if we wanted to add another point (e.g. $(3, 6)$), then we have to recreate the interpolation polynomial from scratch. For the set $P = \{(1, 2), (2, 3), (3, 6)\}$, the polynomial would be

$$p(x) = 2 \cdot \frac{(x-2)(x-3)}{(1-2)(1-3)} + 3 \cdot \frac{(x-1)(x-3)}{(2-1)(2-3)} + 6 \cdot \frac{(x-1)(x-2)}{(3-1)(3-2)} = x^2 - 2x + 3.$$

Having to re-create the interpolation polynomial from scratch each time isn't ideal if we're creating the polynomials by hand. Enter the Newton polynomial. The Newton form of the interpolation polynomial takes the form of

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n \prod_{j=0}^{n-1}(x - x_j) \tag{2}$$

where a_0, a_1, ..., a_n are chosen so that $p(x_i) = y_i, i = 0, ..., n$ [2]. Unlike the Lagrange form, the Newton form easily allows the introduction of new ordered-pairs to the interpolation.

To find a_n, we just apply the condition $p(x_i) = y_i$ to each point [2]. That is

$$\begin{aligned}
p(x_0) =& a_0 = y_0, \\
p(x_1) =& a_0 + a_1(x_1 - x_0) = y_1 \Rightarrow a_1 = \frac{y_1 - a_0}{x_1 - x_0}, \\
p(x_2) =& a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = y_2 \Rightarrow \\
& a_2 = \frac{y_2 - a_0 - a_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)}, \\
& \vdots \\
p(x_n) =& a_0 + a_1(x_n - x_0) + a_2(x_n - x_0)(x_n - x_1) + \ldots + a_n \prod_{j=0}^{n-1}(x_n - x_j)
\end{aligned}$$

This system can be written as a lower triangular linear system for $a_0, \ldots, a_n$ [2]:

$$\begin{bmatrix} 1 & & & & \\ 1 & x_1 - x_0 & & & \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & & \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \dots & \prod_{j=0}^{n-1}(x_n - x_j) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \tag{3}$$

Example 2.2. Consider $P = \{(1, 2), (2, 3)\}$. Then we would write the Newton form of the interpolation polynomial as

$$p(x) = a_0 + a_1(x - 1).$$

By solving the lower matrix from Equation 3, we get $a_0 = 2$ and $a_1 = 1$. So, our final interpolation is

$$p(x) = 2 + 1(x - 1) = x + 1$$

which is the same as the Lagrange interpolation polynomial. However, if we want to add the third point $(3, 6)$, then we simply add onto the polynomial

$$p(x) = a_0 + a_1(x - 1) + a_2(x - 1)(x - 2)$$

and we can easily find $a_2 = 1$, which gives

$$p(x) = 2 + 1(x - 1) + 1(x - 1)(x - 2) = x^2 - 2x + 3.$$

As shown in Example 2.1 and 2.2, these techniques give rise to the same interpolation polynomial. In fact, we can guarantee that any method we use to generate a minimal degree interpolating polynomial will be identical.

Theorem 1. The interpolation polynomial of minimal order that passes through n distinct points is unique.

Proof. Let $P = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ and let $p_1(x)$ and $p_2(x)$ both interpolate P (that is, $p_1(x_i) = p_2(x_i) = y_i$ for $i = 1, \dots, n$).

Let $f(x) = p_1(x) - p_2(x)$. Since $p_1(x)$ and $p_2(x)$ have order at most $n - 1$, then f has order at most $n - 1$. (Because we know the Newton form of interpolation polynomial has order at most $n - 1$, then the minimal order of the polynomial cannot exceed $n - 1$.)

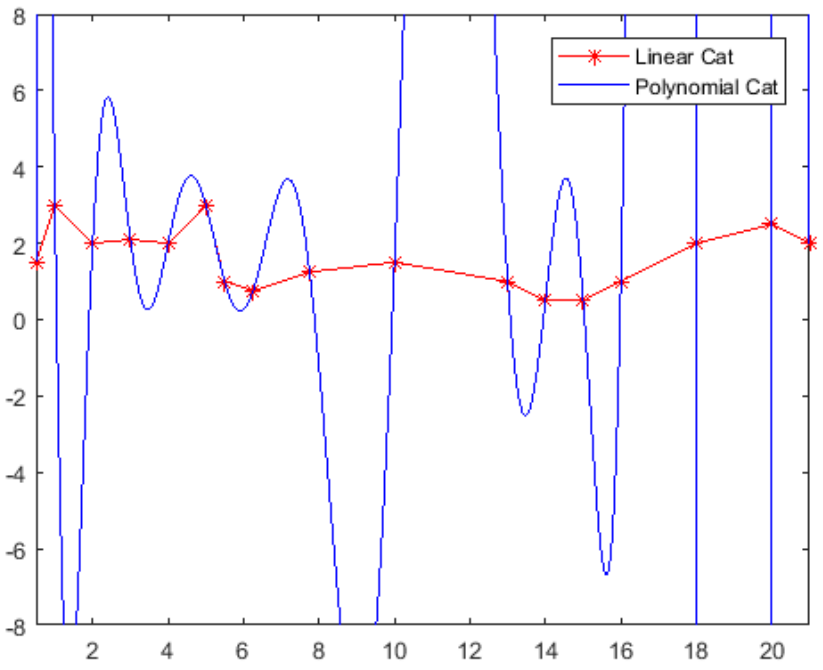


Figure 3: In blue, the interpolation polynomial of the Carroll Cat, with a piecewise linear interpolation shown in red for reference. The polynomial interpolation is not ideal for re-creating the Carroll Cat, while the piecewise linear appears to be more appropriate.

Since $p_1(x)$ and $p_2(x)$ interpolate P, then $p_1(x_i) = y_i$ and $p_2(x_i) = y_i$ for each i. Then

$$f(x_i) = 0 \text{ for all } x_i.$$

So, $f(x)$ has n distinct real roots, which means that $f(x) \equiv 0$ by the Fundamental Theorem of Algebra. Then, $p_1(x) - p_2(x) = 0$ for all x. Therefore, $p_1(x) = p_2(x)$ and the interpolation polynomial is unique. □

Since our interpolation polynomial is unique, we can then use Newton or Lagrange to generate an interpolation polynomial for the Carroll Cat from Figure 2. Figure 3 shows what this form of interpolation would generate. The figure demonstrates that we do not maintain any of the structure of the Carroll Cat. While a high-order polynomial provides us with a smooth, single-piece curve, it does not re-create the original shape. This means that we need to examine alternative methods if we want to accurately skin the Carroll Cat.

2.2 Piecewise Linear Interpolation

As seen in Figure 3, a polynomial interpolation may result in large oscillations. A more appropriate way to represent the Carroll Cat might be to use a piecewise interpolation method.

Given a set of ordered pairs, $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, define the interpolation between two consecutive points (x_i, y_i) and (x_{i+1}, y_{i+1}) in point-slope form. That is:

$$f_i(x) = \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)(x - x_i) + y_i \text{ for } x \in [x_i, x_{i+1}] \tag{4}$$

If we use a piecewise linear method of interpolation for each adjacent pair of points in the domain of the Carroll Cat, we get a collection of functions that can be used to generate the red interpolation in Figure 3.

While we do get a good representation of the Carroll Cat, this method lacks the natural smoothness that we expect.

2.3 Cubic Splines

One way we can get a smooth drawing of the Carroll Cat is by ensuring the first and second derivative between connected interpolations are the same. This means the piecewise interpolation would have C^1 and C^2 continuity, and consequently smooth bending, in the interpolation it passes through the given points.

Given a set of ordered pairs, $P = \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, define the cubic spline interpolation between two consecutive points (x_i, y_i) and (x_{i+1}, y_{i+1}) as a cubic

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \text{ for } x \in [x_i, x_{i+1}] \tag{5}$$

such that

$$f_i(x_i) = y_i \text{ for } x_i \in \{x_0, x_1, \ldots, x_n\}$$

and

$$f_i(x_i) = f_{i+1}(x_i)$$
$$f_i'(x_i) = f_{i+1}'(x_i)$$
$$f_i''(x_i) = f_{i+1}''(x_i)$$

for all x_i such that $i \neq 0, n$. This ensures that the final curve will be continuous and twice differentiable over the piecewise interpolation. Since we define $f_i(x)$ as a symbolic cubic function,

then we can define the derivatives as

$$f_i'(x) = 3a_i x^2 + 2b_i x + c_i$$
$$f_i''(x) = 6a_i x + 2b_i.$$

Since the equations are linear in the coefficients, we can use linear algebra to solve for a_i, b_i, c_i, and d_i for each piecewise cubic $f_i(x)$.

With the constraints on position, first, and second derivatives, we have with $(n-1)$ cubic functions with $4(n-1)$ unknowns. Since we have $(n-2)$ equations enforcing position, $(n-2)$ equations enforcing continuity, $(n-2)$ enforcing differentiability, and $(n-2)$ enforcing concavity, we need 2 more equations to match the total unknowns. In order to create a well-defined system, we need to introduce an additional restriction on the piecewise cubics on $[x_0, x_1]$ and $[x_{n-1}, x_n]$. For cubic splines, there are two common methods:

- We can **clamp** the system, which means that the first derivatives on the endpoints are prescribed.
- We can create a **natural** spline which forces the second derivatives on the endpoints to be zero.

Figure 4 shows the Carroll Cat with a cubic spline interpolation. For this cat, we used a clamped spline and set the slopes at the ends to be equal to the piecewise linear slope.

We would also like to consider the error that the cubic spline has when re-creating a known curve. Figure 5 shows the cubic re-creation of a sine function given the end points and local minimum and maximum on the left, with the difference on the right.

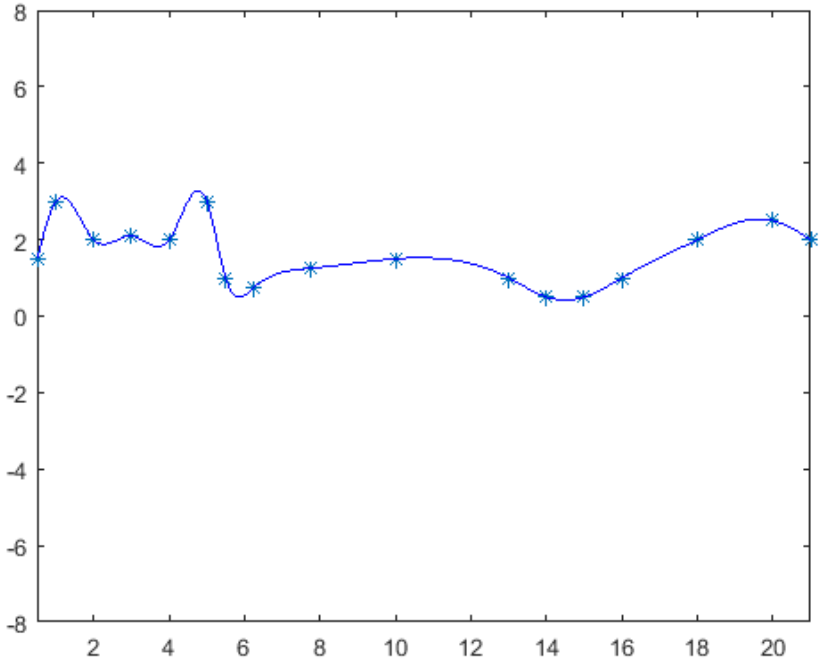


Figure 4: A representation of the Carroll Cat with a piecewise cubic interpolation, or a cubic spline. This re-creation is twice differentiable, so the curve has the same slope and concavity at each internal point.

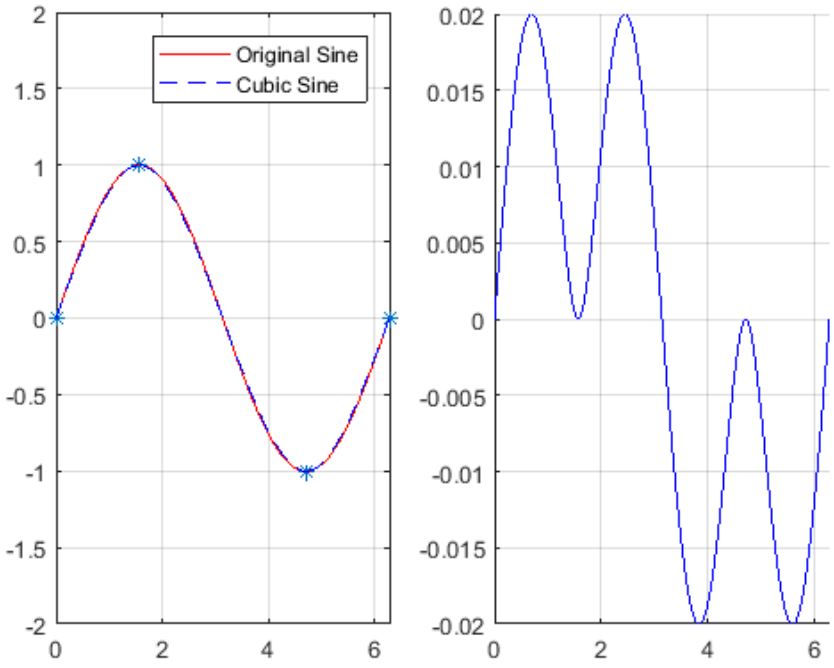


Figure 5: A plot showing cubic spline interpolation against a sine function. The given points were the end points as well as the maximum and minimum. The left shows the interpolation while the right shows the difference of the two functions.

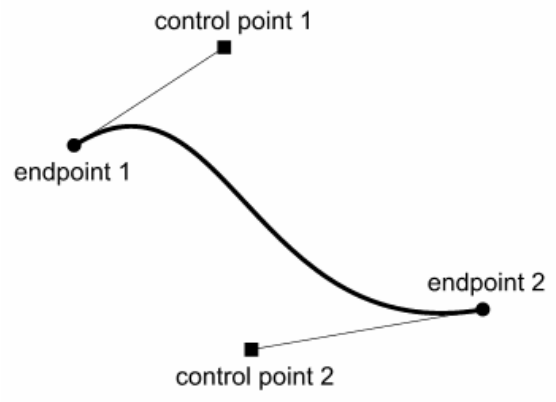


Figure 6: A basic representation of a Bézier curve. The endpoints control the location of the curve while the control points control the slopes near that point [3].

2.4 Bézier Curves

A Bézier curve is another method of interpolation. Bézier curves still pass through the interpolation points (called endpoints) but the slope is dictated by outside control points. Figure 6 shows an example of a Bézier curve.

Bézier curves are generated parametrically, so this method allows us to represent curves that only implicitly define y as a function of x. This means that the set of points can now create vertical lines and the ordered pairs can be listed in non-ascending x-coordinates.

One of most common uses of Bézier curves is for PostScript fonts. If computers only had one font which was always the same size, then it would be computationally inexpensive to store what each individual character looks like. However, most modern computers have hundreds of fonts which can range from 8 pixels to over 100 pixels large. Using Bézier curves means that the computer can instead scale the endpoints and control points by the appropriate factor and re-create the font to look smooth at all levels.

Given endpoints (x_1, y_1), (x_4, y_4) and control points (x_2, y_2), (x_3, y_3), we can define a parametrized Bézier curve defined on $0 \leq t \leq 1$ by the equations

$$x(t) = x_1 + b_x t + c_x t^2 + d_x t^3 \tag{6}$$

$$y(t) = y_1 + b_y t + c_y t^2 + d_y t^3. \tag{7}$$

By requiring that the parameterization pass through the appropriate points and have the appropriate slopes, we can find these parameters are

$$\begin{aligned} b_x &= 3(x_2 - x_1), \quad c_x = 3(x_3 - x_2) - b_x, \quad d_x = x_4 - x_1 - b_x - c_x, \\ b_y &= 3(y_2 - y_1), \quad c_y = 3(y_3 - y_2) - b_y, \quad d_y = y_4 - y_1 - b_y - c_y. \end{aligned} \tag{8}$$

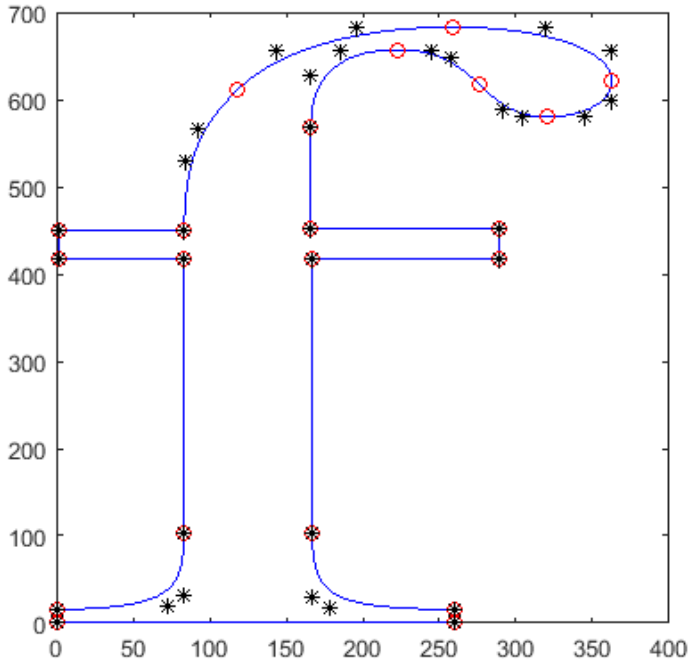


Figure 7: A PostScript Times-Roman letter 'f'. Red circles are the endpoints and black stars are the control points. Straight lines occur when the endpoints and control points coincide [5].

An example of a Bézier curve is shown in Figure 7. We leave the creation of the Carroll Cat using Bézier curves to the interested reader. The red circles are the endpoints, while the black stars are control points. When an endpoint and control point coincide, we see that the process creates a straight line. However, when the control points do not coincide, then the line is "pulled" towards the control points [4].

Theorem 2. When the control points coincide with the endpoints, Bézier curves create a line.

Proof. Consider the parameterization in Equations (6) (7). The slope can be represented as

$$\frac{dy}{dx} = \frac{b_y + c_y t + 3d_y t^2}{b_x + c_x t + 3d_x t^2}.$$

Substituting Equation (8) (and noting that $b_x = b_y = 0$ when $x_1 = x_2$ and $y_1 = y_2$),

$$\begin{aligned}\frac{dy}{dx} &= \frac{2(3(y_4 - y_1))t - 3(2y_4 - 2y_1)t^2}{2(3(x_4 - x_1))t - 3(2x_4 - 2x_1)t^2} \\ &= \frac{6(y_4 - y_1)}{6(x_4 - x_1)}\left[\frac{t - t^2}{t - t^2}\right] \\ &= \frac{y_4 - y_1}{x_4 - x_1}.\end{aligned}$$

Since $\frac{y_4 - y_1}{x_4 - x_1}$ is constant, then the slope from (x_1, y_1) to (x_4, y_4) is constant. Therefore, the Bézier curve is linear. □

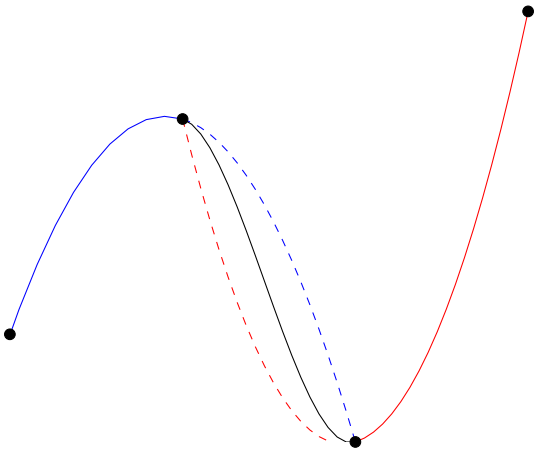


Figure 8: An illustration of a quadrubic curve. The blue is the fit quadratic for the first three points, the red is the fit quadratic for the last three points. The black curve between point 2 and point 3 is the moving average which creates the quadrubic.

3 New Techniques

Both cubic splines and Bézier curves are effective methods of interpolation. In the remainder of this paper, we define two new forms of interpolation techniques which retain desired properties from the previous methods, while not requiring the same types of work for generation. Our first method utilizes a quadratic function, the general form of which has 3 unknowns rather than the 4 required for a cubic, to create an interpolation polynomial.

Our second method considers a physical structure. We create an interpolation by taking a "beam" and bending it around the given points. This method was motivated by splines, which are thin pieces of material that were bent to help draw smooth curves between points. Our method utilizes a set of single dimensional beams and the forces required to create a continuous system.

3.1 Quadrubics

While piecewise linear and cubic splines are typical methods for interpolation, piecewise quadratic is usually ignored. The system of equations corresponding to quadratic interpolation has only one undetermined parameter (compared to the 2 we get with cubics). This means that the use of piecewise quadratics requires us to decide which endpoint we're going to apply the additional conditions to.

Our technique, the "quadrubic", creates the unique quadratic functions for each set of 3-concurrent points within the domain. This creates overlap between the internal quadratics. Wherever this occurs, we do a weighted average to create a smooth cubic. This is shown in Figure 8.

The first step is to create a quadratic function between every set of three consecutive points. For a set of ordered pairs $P = \{(x_0, y_0), (x_0, y_0), \ldots, (x_n, y_n)\}$, create a quadratic of the form $q_i(x) = \alpha_i x^2 + \beta_i x + \gamma_i$ such that

$$q_i(x_{i-1}) = y_{i-1}, \quad q_i(x_i) = y_i, \quad q_i(x_{i+1}) = y_{i+1} \tag{9}$$

for $i = 1, \ldots, (n-1)$.

If we define B_i as

$$B_i = \begin{bmatrix} x_{i-1}^2 & x_{i-1} & 1 \\ x_i^2 & x_i & 1 \\ x_{i+1}^2 & x_{i+1} & 1 \end{bmatrix}$$

then we can write the system from Equation (9) as a matrix equation

$$B_i \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix} = \begin{bmatrix} y_{i-1} \\ y_i \\ y_{i+1} \end{bmatrix}.$$

Next, we create the smoothing function by taking a convex combination of the two quadratics defined on each interval $[x_i, x_{i+1}]$. This creates a function

$$f_i(x) = \frac{(x - x_i)q_{i+1}(x) + (x_{i+1} - x)q_i(x)}{x_{i+1} - x_i},$$

such that $q_i(x)$ refers to the quadratics defined across the domain. By construction, $f_i(x)$ is cubic. Expanding the terms in $f_i(x)$ to isolate the coefficients, we find

$$\begin{aligned} f_i(x) =& (\alpha_{i+1} - \alpha_i)x^3 + (\beta_{i+1} - \alpha_{i+1}x_i + \alpha_i x_{i-1} - \beta_i)x^2 \\ &+ (\gamma_{i+1} - \beta_{i+1}x_i + \beta_i x_{i+1} - \gamma_i)x + (\gamma_i x_{i+1} - \gamma_{i+1}x_i). \end{aligned} \tag{10}$$

We define a matrix C_i where

$$C_i = \begin{bmatrix} 1 & 0 & 0 \\ -x_i & 1 & 0 \\ 0 & -x_i & 1 \\ 0 & 0 & -x_i \end{bmatrix}.$$

We can find the coefficients for the quadrubic by evaluating

$$\begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = C_i \begin{bmatrix} \alpha_{i+1} \\ \beta_{i+1} \\ \gamma_{i+1} \end{bmatrix} - C_{i+1} \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix}.$$

If we wished to solve this as a single matrix system, we can then create a matrix A:

$$A = \begin{bmatrix} B_0 & & & & & & \\ C_1 & I_4 & -C_2 & & & & \\ & & B_1 & & & & \\ & & & \ddots & & & \\ & & & & B_{n-2} & & \\ & & & & C_{n-2} & I_4 & -C_{n-1} \\ & & & & & & B_{n-1} \end{bmatrix}$$

and apply standard linear algebra techniques to solve a corresponding system for the appropriate right-hand side.

Theorem 3. The collection of interpolating quadrubics are C^1 continuous on the domain.

Proof. Consider the quadrubic

$$f_i(x) = \frac{(x - x_i)q_{i+1}(x) + (x_{i+1} - x)q_i(x)}{x_{i+1} - x_i}.$$

Since this is a cubic function on $[x_i, x_{i+1}]$, then $f_i(x)$ is C^∞ on (x_i, x_{i+1}). It suffices to show that the function is C^1 at each x_i. Taking the derivative by applying the product rule gives

$$f_i'(x) = \frac{1}{x_{i+1} - x_i}[(x - x_i)q_{i+1}'(x) + q_{i+1}(x) + (x_{i+1} - x)q_i'(x) - q_i(x)].$$

If we evaluate the derivative at x_i, then we see

$$f_i'(x_i) = \frac{1}{x_{i+1} - x_i}[q_{i+1}(x_i) + (x_{i+1} - x_i)q_i'(x_i) - q_i(x_i)].$$

Since $q_i(x_i) = q_{i+1}(x_i) = y_i$, then it is clear that $f_i'(x_i) = q_i'(x_i)$. Similarly, $f_{i-1}'(x_i) = q_i'(x_i)$. Therefore, the collection of interpolating quadrubics are C^1 continuous. □

The interpolation of the Carroll Cat is shown in Figure 8. A comparison of this method against the cubic splines will be discussed in Section 4.

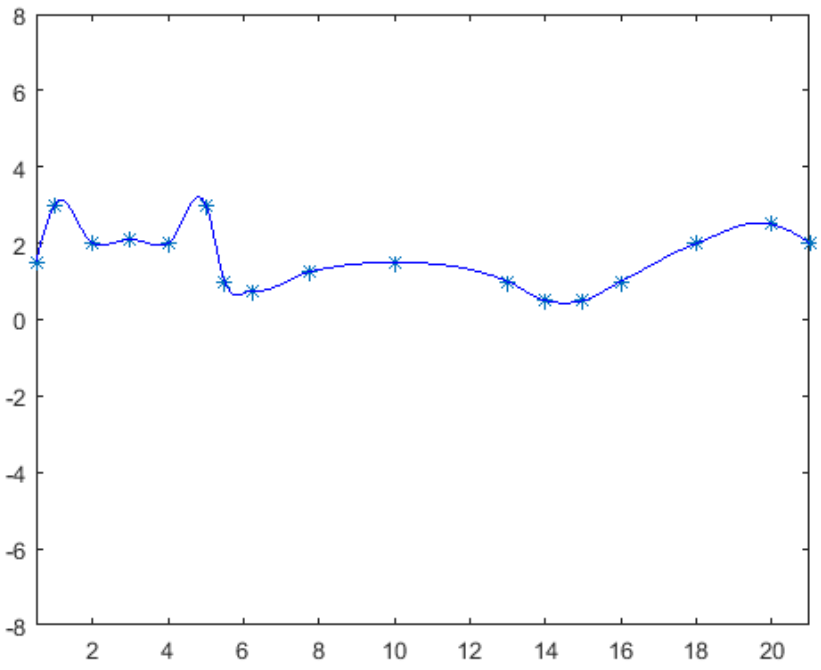


Figure 9: The interpolated cat using the quadrubic method. This method creates a flatter image of the cat than the cubic, with the peak between the ears and the dip after the ears much more shallower than before.

We also wanted to investigate our method against a known function. We chose the points $(0, 0)$, $(\pi/2, 1)$, $(3\pi/2, -1)$ and $(2\pi, 0)$ (the endpoints and local extrema) to re-create a sine function using the quadrubic method. Figure 10 shows quadrubic approximation of the sine function, as well as the difference between the original function and the interpolation. We will discuss the details of the error in Section 4.

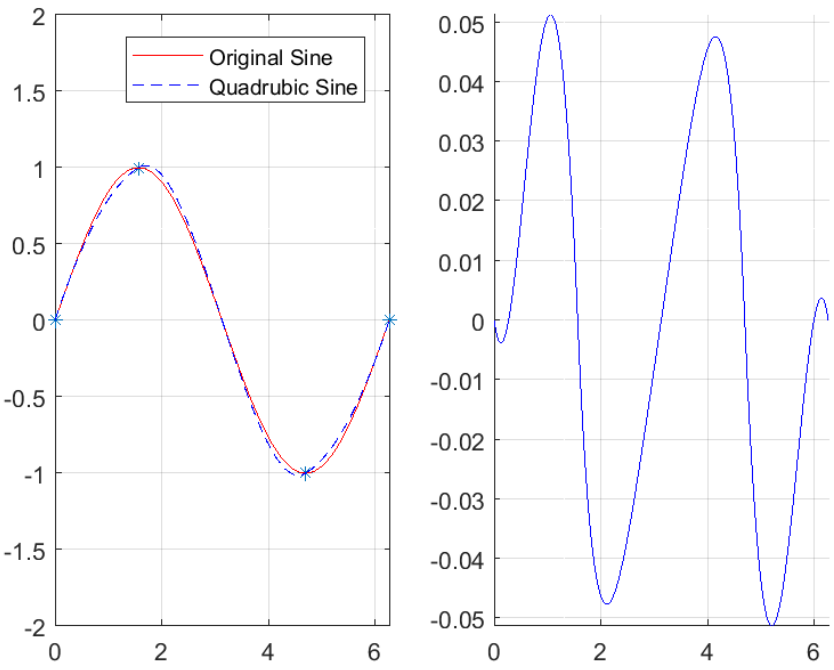


Figure 10: A plot showing quadrubic interpolation against a sine function. The given points were the end points as well as the maximum and minimum. The left shows the interpolation while the right shows the difference of the two functions.

3.2 Bending Beam

Our final method of interpolation attempts to mimic the physics of a bending beam. This method considers a series of beams which are pinned by the given points and bent towards each other to create a single, seamless beam. This method of interpolation considers overhanging beams with the same physical properties that are bent towards some undetermined point with equal and opposite force.

To do this, we examine the moment-curvature relationship established in statics [6], written as

$$\frac{1}{\rho} = -\frac{\epsilon}{y},$$

where ρ is the radius of curvature, ϵ is strain, and y is displacement. If we make the assumption that the beam is a Hookean solid, then we can apply Hooke's law. This allows us to make the substitution

$$\epsilon = \sigma/E$$

$$\sigma = -My/I$$

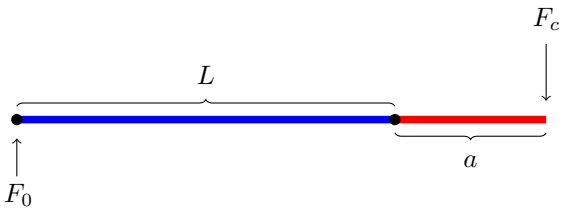


Figure 11: A bending beam diagram. The beam is held at the black points with an overhang of length a (shown in red). The force on the beam will cause the entire system to bend.

where σ is stress, E is the modulus of elasticity, M is the internal moment, and I is the moment of inertia. So, we can rewrite the moment-curvature relationship as

$$\frac{1}{\rho} = \frac{M}{EI}.$$

Hibbeler defines the equation of an elastic curve by the coordinates v and x and expresses the deflection of the beam as a function of $v = f(x)$, which can be represented by

$$\frac{1}{\rho} = \frac{d^2v/dx^2}{[1 + (dv/dx)^2]^{3/2}} = \frac{M}{EI}.$$

In many physical scenarios, we can restrict the maximum deflection of the beam to something small. This assumption allows us to simplify this differential equation to [6]

$$\frac{d^2v}{dx^2} = \frac{M}{EI}. \tag{11}$$

Since the deformations in the system are not "small", we acknowledge that we lose some of the physical reality by making this assumption.

Consider Figure 11. The interpolation methods considers a beam which is fixed at two points L units apart with an overhang of length a. A force, F_c, is then applied to the end of the beam.

Because the beam acts as a cohesive unit, the downwards force on the overhang creates an upwards force on the first pinned point. This force, F_0, is equal to

$$F_0 = F_c \frac{a}{L}.$$

With this information, we can then start to create a general solution for the differential equation for the displacement of the beam. We consider the beam in two parts: $v_1(x)$ is the displacement of the beam between the two pinned points (shown in blue in Figure 11) and $v_2(x)$ is the displacement in the overhanging part of the beam (shown in red). Following Hibbeler's lead, $v_2(0)$ will be the end of the overhang while $v_2(a)$ will be the pinned point [6]

Starting with Equation (11) and rearranging we can use the force F_0 to solve for $v_1(x)$

$$\begin{aligned} EI\frac{d^2v_1}{dx^2} &= F_0x \\ EI\frac{dv_1}{dx} &= \frac{F_0}{2}x^2 + C_1 \\ EIv_1 &= \frac{F_0}{6}x^3 + C_1x + C_2. \end{aligned} \tag{12}$$

Similarly, we can find $v_2(x)$ to be

$$EIv_2 = \frac{F_c}{6}x^3 + C_3x + C_4. \tag{13}$$

Since this is an interpolation technique, we must make sure that the displacement at the pinned points are 0. That is:

$$v_1(0) = 0 \tag{14}$$

$$v_1(L) = 0 \tag{15}$$

$$v_2(a) = 0. \tag{16}$$

While we consider the beam in two parts, this beam is a continuous piece of material. Therefore, we also restrict the beam such that

$$v_1'(L) = v_2'(a). \tag{17}$$

Equations (14), (15), (16), and (17) allow us to solve for the constants in Equations (12) and (13). Simple calculus and algebra show the solutions are

$$\begin{aligned} C_1 &= \frac{(EIy_1 - C_2)}{L} - \frac{F_0}{6}L^2 \\ C_2 &= EIy_0 \\ C_3 &= -(\frac{F_0}{2}L^2 + C_1) - \frac{F_c}{2}a^2 \\ C_4 &= -\frac{F_c}{6}a^3 - C_3a + EIy_1 \end{aligned}$$

where $y_0 = v_1(0)$ and $y_1 = v_1(L)$ are the prescribed y values for the given values. This system allows us to interpolate two points with a prescribed overhang a.

However, for a system of four points, we adopt a different method. For the second pair of points, we solve a left hanging beam equation. We then ensure that the two beams meet at the same point and enforce that the force on the beams has to be equal and opposite. We can then

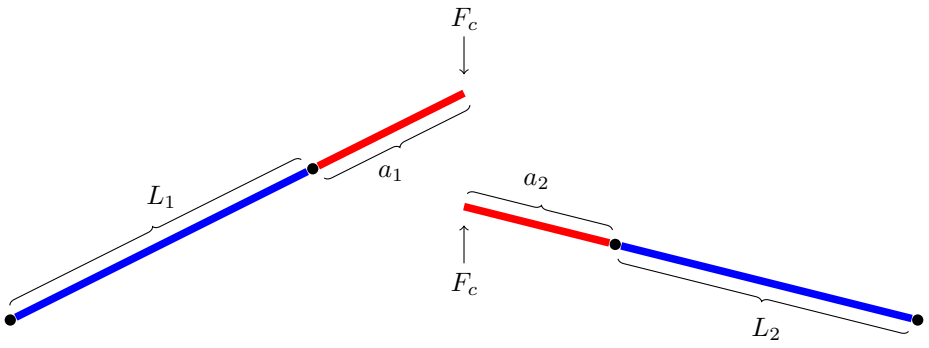


Figure 12: Two bending beams fixed at the interpolation points. By adjusting a_1 and a_2, we can find a point where F_c will cause the system of beams to be continuous and differentiable.

solve for the appropriate overhang a for each beam to ensure the system is smooth. A diagram of this is shown in Figure 12.

Figure 13 shows the interpolation method on the first four points of the Carroll Cat. We did not complete a method which interpolated more than 4-points, but we do discuss this process in Section 5.

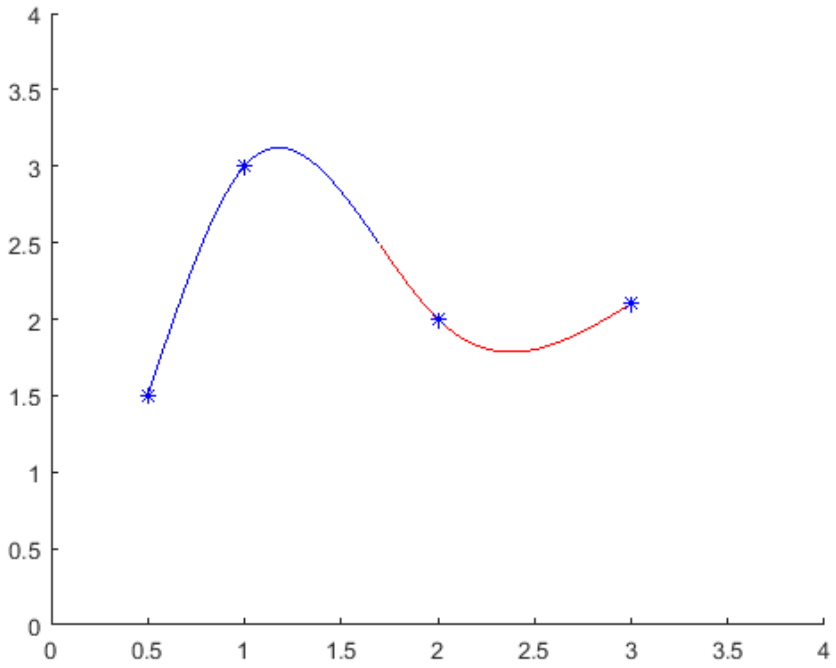


Figure 13: An interpolation using the bending beam method on the first four points of the cat. The small gap is where the phantom point exists for each of these beams, with the beam in red having a left overhang and the blue beam having a right overhang.

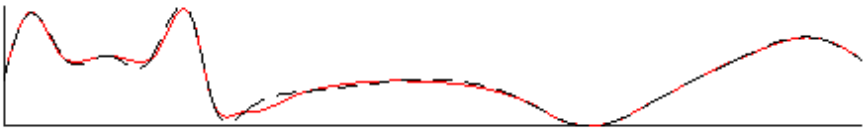


Figure 14: The Carroll Cat as interpolated by the quadrubic method (red) and the cubic spline method (black).

4 Method Comparisons and Error Analysis

In this section, we discuss the qualitative differences between our introduced interpolation methods as well as a quantitative analysis of the quadrubic and the cubic splines method against a given sine curve.

4.1 The Carroll Cat

While the cubic spline method is popular for interpolation, we wanted to see how it compared against the Carroll Cat for the quadrubic method since both functions contain cubic interpolation. Figure 14 shows the direct comparison on the interpolation with the quadrubic method (red) and the cubic splines method (black).

Qualitatively, we see that the quadrubic arguably makes a more realistic looking cat. The crown between the ears is less dramatic and the piecewise functions connecting the neck and back seem more physically possible. However, as we get to the tail-end of the cat, we see that there is not much difference between the cubic and quadrubic methods. This implies that the quadrubic can visually mimic a cubic spline under certain conditions.

4.2 Sine Interpolations

Recall in Figures 5 and 10, we compared the quadrubic and cubic spline interpolations against the sine function with the same given points. We chose to measure the error by taking the integral of the absolute difference between the original sine function and the interpolation. For the cubic spline, we found this error to be 0.0730 while the quadrubic has an error of 0.1772. This means that, with the chosen points, the cubic has a visually more accurate approximation of the original sine.

4.3 Method Comparisons

Finally, we wanted to see how our methods compare to each other. Figure 15 shows the linear, quadrubic, natural cubic, and the bending beam interpolation. As we discussed before, the

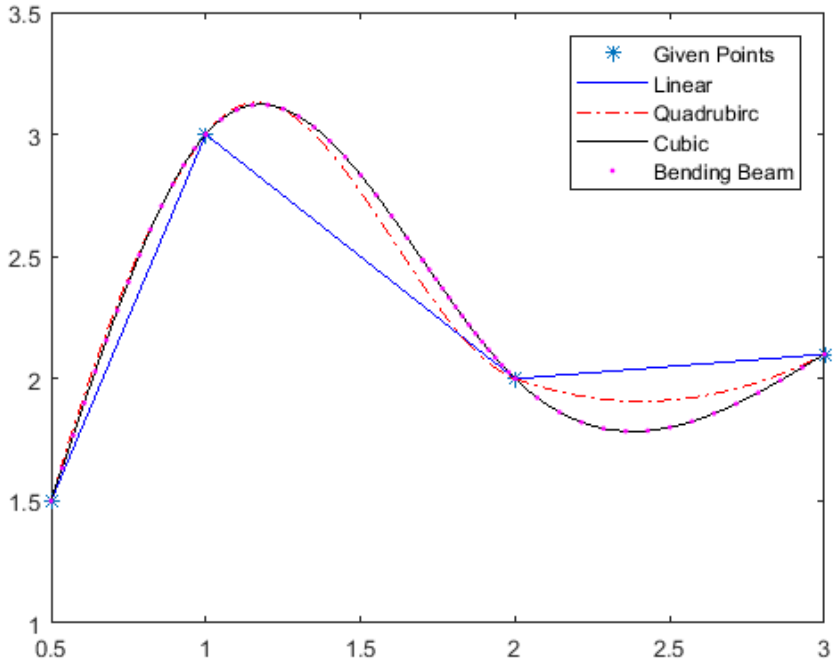


Figure 15: A comparison of the four major interpolation techniques we discussed–linear, natural cubic, quadrubic, and the bending beam method–on the first four points of the interpolation cat. It appears that the bending beam method creates an extremely similar curve to the natural cubic.

quadrubic technique appears to be a more mollified version of the cubic.

It is also worth noting that the bending beam method appears to be identical to the natural cubic. We have not rigorously proven the two to be equivalent and discuss how we might go about this in Section 5.

5 Future Work

We would like to see if we can find a way to bound the error of the quadrubic. Quarteroni et al. describes how to bound the error of a cubic and we believe that we could use similar methods to find the bounds on the quadrubic [7]. If we were able to calculate this bound, we may be able to determine if our method might ever be worth utilizing over one of the standard methods of interpolation.

We would also like to continue the bending beam methods for more than 4 points. Figure 16 shows how we would set up the interpolation system, where each internal point would have force applied to both sides of the beam.

We would also like to determine if the natural cubic spline and the bending beam interpolation methods give us the same interpolation functions.

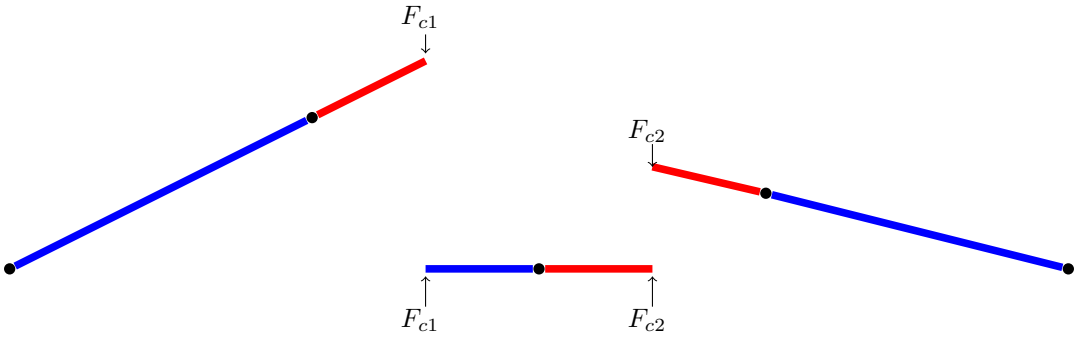


Figure 16: The bending beam interpolation method for five points.

5.1 2-Dimensional Interpolation

Finally, we would also like to adapt our new methods into 2-D interpolation. This would allow us to create surfaces and hopefully create a skin for a point-cloud system.

One method of 2-D interpolation techniques we examined was bilinear interpolation. This method creates a grid-like structure using linear interpolation and also utilizes the Lagrange form of the polynomial. Therefore, we can find the point (x_i, y_1) by solving [8]

$$f(x_i, y_1) = \frac{x_i - x_2}{x_1 - x_2} f(x_1, y_1) + \frac{x_i - x_1}{x_2 - x_1} f(x_2, y_1)$$

and the point at (x_i, y_2) by solving

$$f(x_i, y_2) = \frac{x_i - x_2}{x_1 - x_2} f(x_1, y_2) + \frac{x_i - x_1}{x_2 - x_1} f(x_2, y_2).$$

For example, Figure 17 shows the interpolation between four points. While the interpolation across the center gives a linear interpolation, the Lagrange form basis means that the full interpolation is not necessarily planar. Since bilinear interpolation utilizes linear interpolation between the given grid points, then we could apply a quadrubic interpolation between the points and establish a bi-quadrubic interpolation.

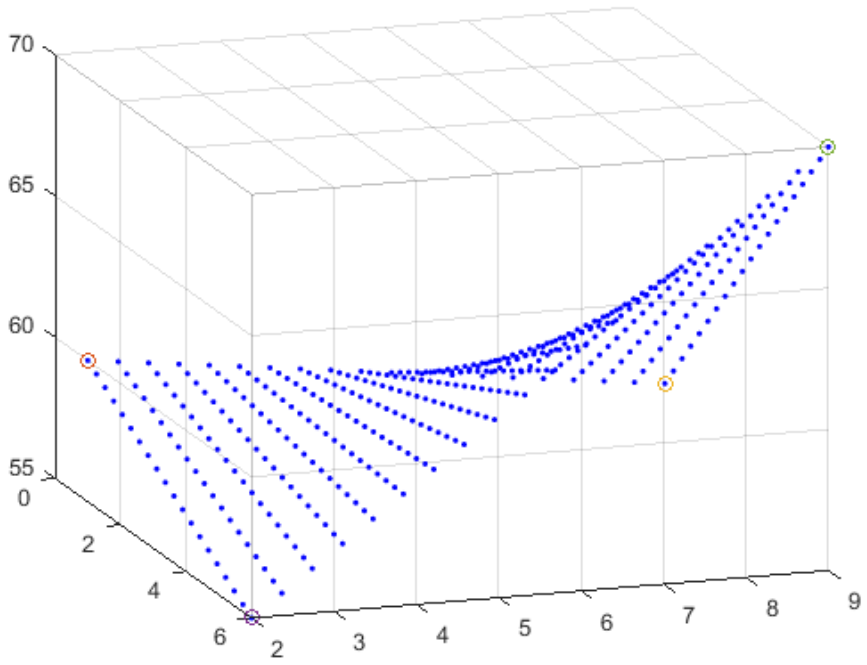


Figure 17: The bilinear interpolation for the surface between four given points (circled). You can see that the interpolations are very linear as they go across, but the surface itself is not planar.

6 Conclusion

We have discussed several one-dimensional interpolation techniques in this paper. While the techniques introduced in this paper are obviously more computationally expensive than the common methods, we found that they do have some qualities which may be desirable. The quadrubic, which was a refinement on quadratic interpolation, tends to be a more mollified version of the cubic spline. As we saw in the Carroll Cat, if the interpolation points are local extrema, we found the quadrubic is less likely to "overshoot" than the cubic spline. We also found that the bending beam method allows us to create an interpolation which is consistent with current models for elastic materials. If the purpose of the interpolation is to model reality (e.g. in a physically based simulation), then the bending beam method would be ideal.

References

[1] Seung-Hyun Yoon. A surface displaced from a manifold — https://www.researchgate.net/publication/221209194_A_Surface_Displaced_from_a_Manifold, 2016. [Accessed 10-April-2018].

[2] Anne Greenbaum and Timothy P. Chartier. *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*. Princeton University Press, Princeton, NJ, 2012.

[3] Unity class eight — https://holdmycat.com/2017/11/22/unity-class-five/, 2017. [Accessed 12-April-2018].

[4] Timothy Sauer. *Numerical Analysis*. Pearson Education, Boston, MA, 1st edition, 2006.

[5] Richard L. Burden, Douglas J. Faires, and Annette M. Burden. *Numerical Analysis*. Cengage Learning, Boston, MA, 10th edition, 2016.

[6] R.C. Hibbeler. *Statics and Mechanics of Materials*. Pearson Education, Hoboken, NJ, 5th edition, 2017.

[7] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer-Verlag, New York, NY, 2000.

[8] Steven C. Chapra and Raymond P. Canale. *Numerical Methods for Engineers*. McGraw-Hill, New York, NY, 6th edition, 2006.