Lecture 11: Linear Interpolation Again - Bézier Curves

Let's say that we are given two points, for example the points $(1, 1)$ and $(5, 4)$ shown in Figure 1. The objective of *linear interpolation* is to define a linear function that includes the two points in its graph. In other words, to define the line that passes through the points. The word *interpolation* means that the function's graph must contain the points. Figure 2 shows just the part of the line

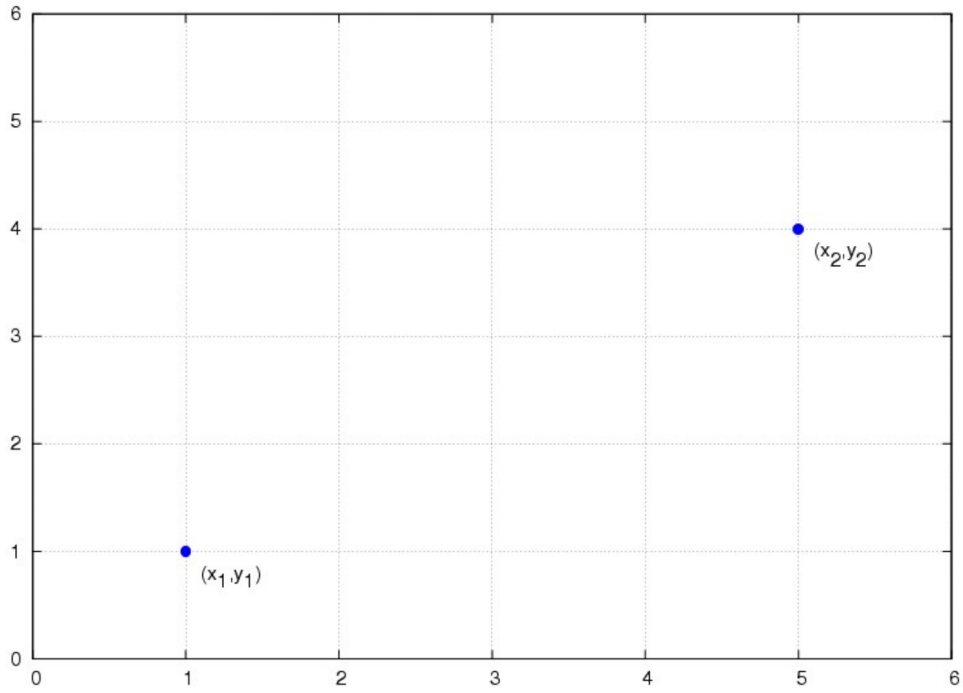

Figure 1: Two points.

passing through the two points that lies between them. Focusing on just this part of the line's graph is called *piecewise* linear interpolation. There are many ways to come up with the formula for the line passing through the two points. For example, we can compute the slope from the given points:

$$m = \frac{y_2 - y_1}{x_2 - x_1},$$

and plug that into the point-slope formula:

$$y - y_1 = m(x - x_1). \tag{1}$$

Equation (1) is one way to define the linear interpolant between the points. The next section discusses another way.

Parameterized Linear Interpolation

Imagine that the two points are places on a map. We will take an imaginary journey along a straight path starting at the point (x_1, y_1) and ending at (x_2, y_2). Figure 3 illustrates the journey and a few way points. For example, we pass through the point $(3, 2.5)$ exactly half-way through our journey.

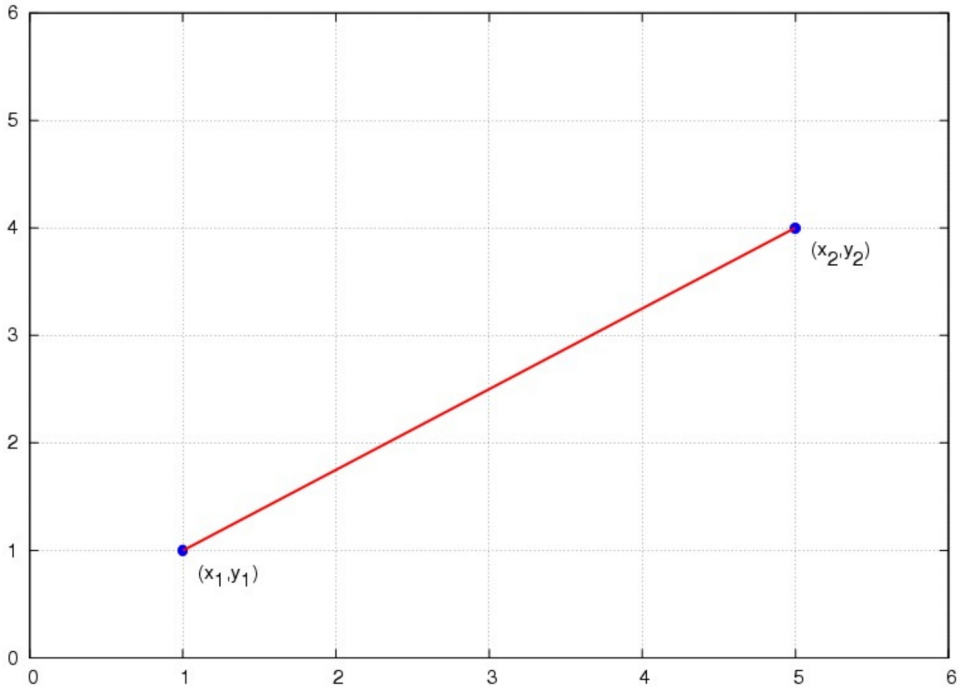

Figure 2: A line segment connecting the two points from Figure 1.

The variable t shown in Figure 3 refers to how far along the journey we are at any given point. Starting out at point (x_1, y_1), $t = 0$. When we are half-way, $t = 0.5$, and $t = 1$ when we've arrived at our destination (x_2, y_2). Given a value of $t, 0 \leq t \leq 1$, we want a formula that returns the corresponding x and y coordinates that lie on the path.

First, consider only the x-values. The total distance travelled along the path in the x-coordinates (including a sign for direction) is $x_2 - x_1$. When we are $t\%$ of the way along the path, that distance corresponds to $(x_2 - x_1)t$, and adding that to the starting point, we arrive at:

$$\begin{aligned} x \equiv x(t) &= x_1 + (x_2 - x_1)t \\ &= x_1(1 - t) + x_2 t. \end{aligned}$$

A formula for the y coordinate can be arrived at in the same way:

$$y \equiv y(t) = y_1(1 - t) + y_2 t.$$

The functions $f(t) = 1 - t$ and $g(t) = t$ that appear in the above formulas are interesting enough to have a special name: Bernstein polynomials of order 1. A picture of their graphs is displayed in Figure 4. We can use these functions to think about linear interpolation in a new way: as a weighted-average of translations of the graphs of the two lines displayed in Figure 4.

Quadratic Bézier Interpolation

Previously, we used the linear basis functions

$$f(t) = 1 - t,$$

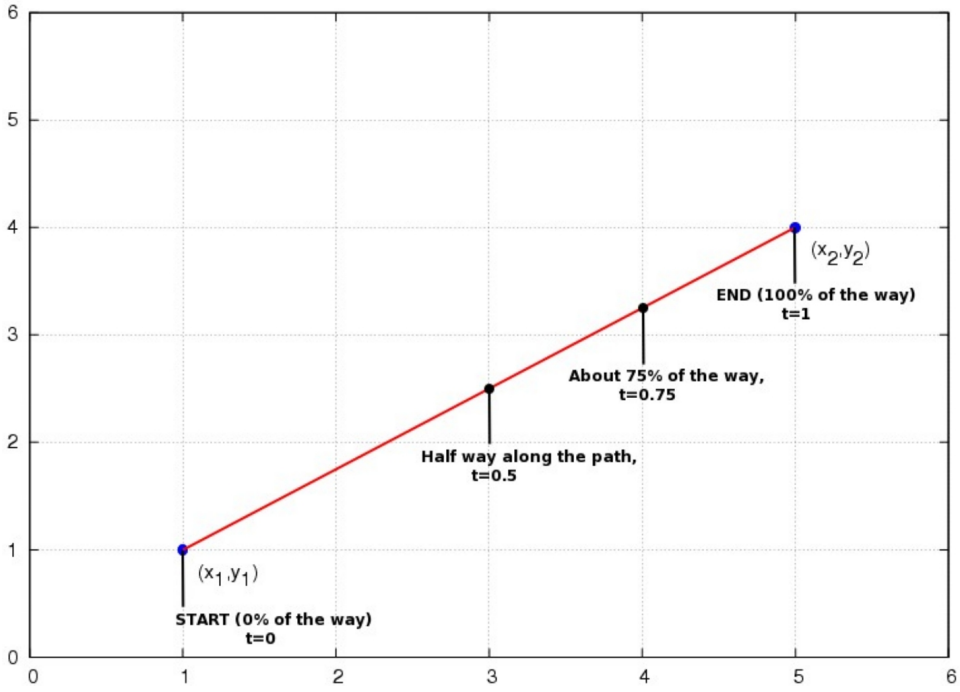

Figure 3: Traveling between the points along a straight path.

and,

$$g(t) = t$$

to define linear interpolants by forming weighted averages of their graphs. An engineer named Pierre Bézier who worked for the French car company Renault wondered what interpolants made from quadratic and higher-order polynomials might look like. The resulting interpolating curves that he developed in the 1960's bear his name. Bézier curves are widely used in computer graphics, and in particular computer font-rendering technologies like True Type, Adobe PDF, and Postscript.

Note that $f(t)+g(t) = (1-t)+t = 1$. Higher-degree Bernstein polynomials (from which we will make Bézier curves) are derived from this observation as follows:

$$\begin{aligned} 1 &= (1-t)+t, \\ 1^2 &= ((1-t)+t)^2 \\ &= (1-t)^2 + t^2 + 2t(1-t). \end{aligned}$$

Quadratic Bézier curves interpolate two points by forming weighted averages of the functions $(1-t)^2$, t^2, and $2t(1-t)$. Those functions are called Bernstein polynomials of order 2.

Note that we are now working with *three* functions. Since we only have two points to interpolate, we need some extra information to deal with the third term. That information can be provided by choosing a third point (x_3, y_3), usually called a *control point*, that helps determine the shape of the resulting Bézier curve. That curve is defined by:

$$\begin{aligned} x(t) &= x_1(1-t)^2 + x_2t^2 + x_3 2t(1-t), \\ y(t) &= y_1(1-t)^2 + y_2t^2 + y_3 2t(1-t). \end{aligned}$$

Note that $(x(0), y(0)) = (x_1, y_1)$, and $(x(1), y(1)) = (x_2, y_2)$, satisfying the interpolation condition. Also note the similarity to the liner example discussed in the last section.

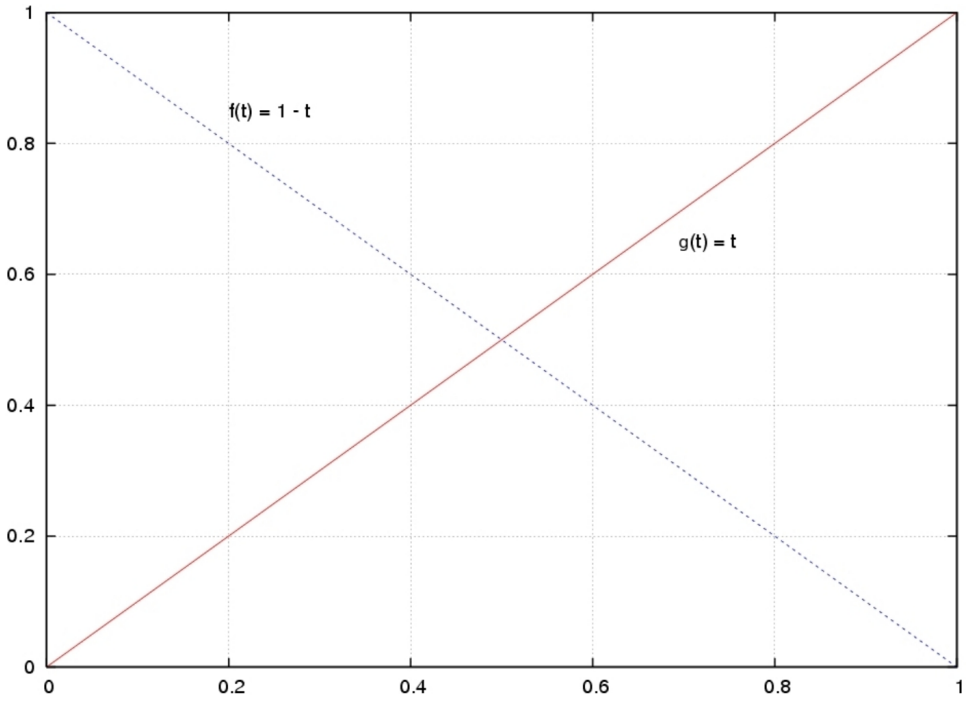

Figure 4: First-order Bernstein polynomial basis functions.

Figure 5 illustrates a quadratic Bézier interpolant with the same points shown in in Figure 1, and a control point of $(x_3, y_3) = (2, 4)$. Lots of image processing programs like Adobe Photoshop and its popular open source alternative the Gimp provide B'ezier selection tools. Figure 6 shows a screenshot of a Gimp session using the selection tool. Note the similarity to Figure 5. Scalable True Type fonts are also drawn using quadratic Bézier curves.

Exercise 1

Make a plot to the one similar in Figure 4 of the second-order Bernstein polynomials used to form quadratic Bézier curves.

Exercise 2

Consider the vectors $x = (2, 6, 10, 10, 6, 2, 1, 2, 6, 10, 10, 10, 13)^T$, and $y = (10, 12, 10, 6, 7, 6, 4, 1, 1, 3, 6, 1, 1)^T$. They define a set of x- and y-coödinates of an approximation to the lower case letter "a" displayed in the figure.

Your assignment is to find a small number of quadratic Bézier curves that interpolate a subset of the points in the above picture and that give a much better looking rendering than the piecewise linear interpolants shown. You are free to choose your own set of curves and subset of points.

To complete this assignment, you will need to code at least four Matlab (or Octave) functions: one for each of the three Bernstein polynomial quadratic basis functions, and one that computes and plots a discretization of a Bézier curve given two endpoints and a control point. You'll then need to play with your new Bézier function to find control points that result in reasonable looking curves when plotted.

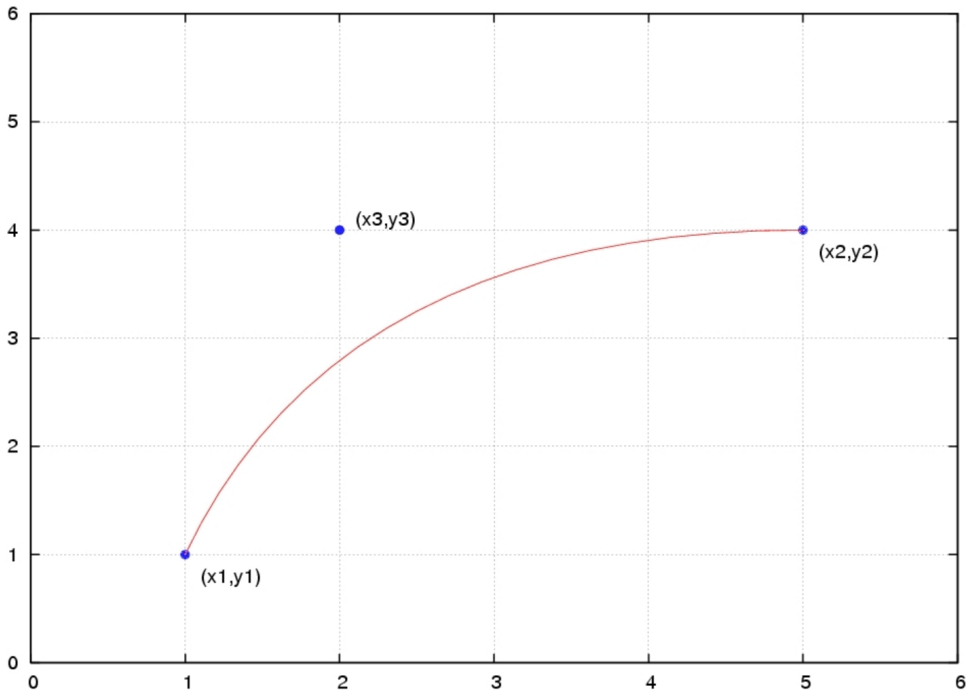

Figure 5: A quadratic Bézier interpolant and its control point.

This project involves just a bit of coding and lots of interactive experimentation and plotting.

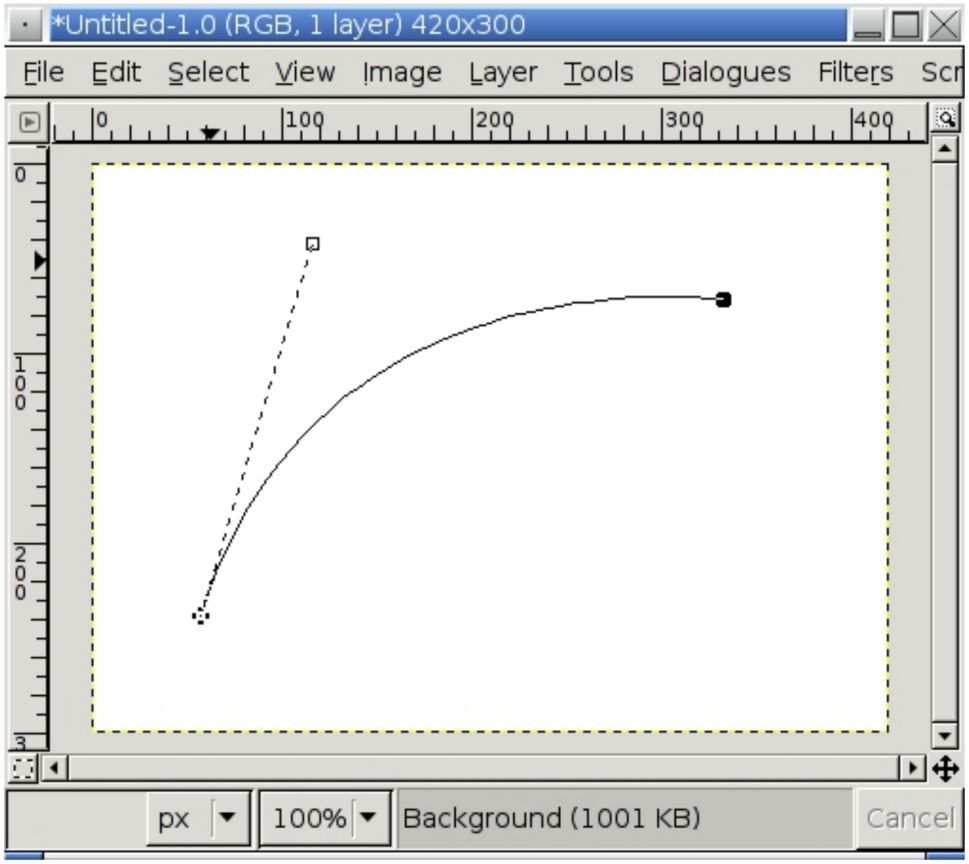

Figure 6: The selection tools in Photoshop and the Gimp use Bézier curves!

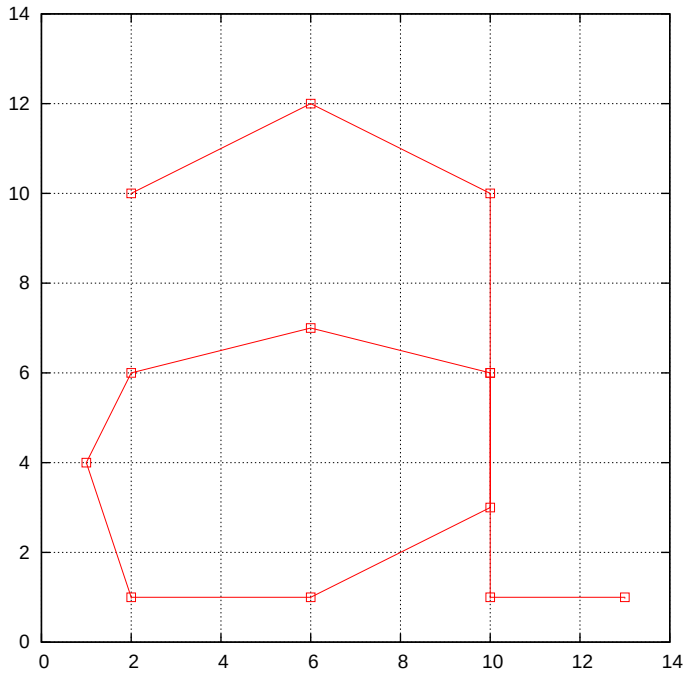

Figure 7: The points defined by x and y with piecewise linear interpolants between them. You can produce this plot with the Matlab/Octave command `plot (x,y,'-r+')`.