MA 323 Geometric Modelling
Course Notes: Day 10
Higher Order Polynomial Curves

David L. Finn

December 14th, 2004

Yesterday, we introduced quintic Hermite curves as a higher order variant of cubic Hermite curves. The advantage of quintic curves that we were concerned with is that they provide us with a method for joining curves in a manner that is twice differentiable, that is in a C^2 manner. This is useful for animation and motion planning as we can prescribe the position, velocity and acceleration at a series of data points and smoothly interpolate between the given values.

Today, we want to consider in some detail the problem of interpolation and data fitting with higher order polynomial curves. This involves generalizing the methods introduced for quadratic and cubic curves. However, we will discuss why these methods are not used often in geometrically modelling, in terms of what properties we can infer about the curve from the controls. We also look at how stable the methods are to small perturbations in the variations of the points and times to be interpolated.

The general difference with the interpolation problem with higher order polynomial curves is that you need to use more information. To interpolate with a nth degree polynomial one needs to supply $n+1$ data points, as there are $n+1$ unknowns in the equation for an nth degree polynomial,

$$a_0 + a_1\, t + a_2\, t^2 + \cdots + a_n\, t^n.$$

In the next subsection, we consider the method of interpolation based on the work of the French mathematician Joseph Louis Lagrange. This provides a theoretical answer to the interpolation problem, and generalizes the method used to perform quadratic interpolation and cubic interpolation.

10.1 Lagrange Interpolation

Given $n+1$ points $\{\mathbf{p}_i\}$ and $n+1$ parameter values t_i with $i = 0, 1, \cdots, n$ and $t_0 < t_1 < \cdots < t_n$, finding a polynomial curve $\mathbf{c}(t)$ with $\mathbf{c}(t_i) = \mathbf{p}_i$ can be accomplished by the methods presented in the previous sections. One can set up a system of $n+1$ equations and $n+1$ unknowns and solve or one can design the appropriate basis functions that construct the curve. However, we note that the matrix method exploited in the section on cubic curves does not generalize nicely to variable degree problems. Especially, if you want to add new points interactively and generate the curve automatically. The reason for this is simple. You need to resolve the system of equations each time increasing the degree of the polynomial. In general, matrix algebra methods are only appropriate when the degree of the polynomial is fixed before hand. It is theoretically possible, but potentially time-consuming given that

solving a system of linear equations is a n^3 algorithm, meaning that we need to perform roughly 1000 operations to determine the coefficients with 10 data points and we need to perform roughly 1000000 operations to determine the coefficients with 100 data points. We will thus examine the basis function approach, that is known as Lagrange interpolation, which is n^2 method, meaning we need to perform 100 operations to determine the coefficients 10 data points and we need to perform roughly 10000 operations to determine the coefficients with 100 data points.

Lagrange interpolation is built in an inductive manner. We consider the base case of linear polynomial interpolation, and then consider how this extends to quadratic polynomial interpolation. We then continue the process inductively with the goal being to define $n+1$ nth degree polynomial functions $L_i^n(t)$ (i=0,1,..n) that satisfy

$$L_i^n(t_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

The interpolating polynomial curve is then $c(t) = \sum_{i=0}^{n} L_i^n(t)\, p_i$.

Recall the linear function that passes through the points p_0 and p_1 at times t_0 and t_1 could be written as

$$\frac{t - t_1}{t_0 - t_1}\, p_0 + \frac{t - t_0}{t_1 - t_0}\, p_1.$$

The important observation of this linear interpolant is the form. This interpolant can be written as $L(t) = L_0(t)\, p_0 + L_1(t)\, p_1$. The functions $L_0(t)$ and $L_1(t)$ are designed so that $L_0(t)$ satisfies $L_0(t_0) = 1$, $L(t_1) = 0$ and $L_1(t)$ satisfies $L_1(t_0) = 0$, $L_1(t_1) = 1$. Notice the form of the basis function as a quotient.

From the section on interpolation with parabolic arcs solution by quadratics, we found that the quadratic Lagrange basis functions are

$$\begin{aligned} L_0^2(t) &= \frac{(t - t_1)(t - t_2)}{(t_0 - t_1)(t_0 - t_2)} \\ L_1^2(t) &= \frac{(t - t_0)(t - t_2)}{(t_1 - t_0)(t_1 - t_2)} \\ L_2^2(t) &= \frac{(t - t_0)(t - t_2)}{(t_2 - t_0)(t_2 - t_1)} \end{aligned}$$

Again notice the form of the basis function as a quotient. Furthermore, notice that the basis functions for quadratic interpolation can be viewed as a product of the linear interpolation functions. It is this that allows the inductive construction, though the inductive construction is hidden in the algebra.

In the general construction, we first note that an nth degree polynomial that satisfies $L_i^n(t_j) = 0$ for $i \neq j$ is given by

$$(t - t_0)(t - t_1) \cdots (t - t_{i-1})(t - t_{i+1}) \cdots (t - t_n).$$

The other constraint $L_i^n(t_i) = 1$ is obtained by dividing by the value of the above function when $t = t_i$. Therefore, the basis functions are

$$L_i^n(t) = \frac{(t - t_0)(t - t_1) \cdots (t - t_{i-1})(t - t_{i+1}) \cdots (t - t_n)}{(t_i - t_0)(t_i - t_1) \cdots (t_i - t_{i-1})(t_i - t_{i+1}) \cdots (t_i - t_n)}.$$

A typical graph of basis function looks like

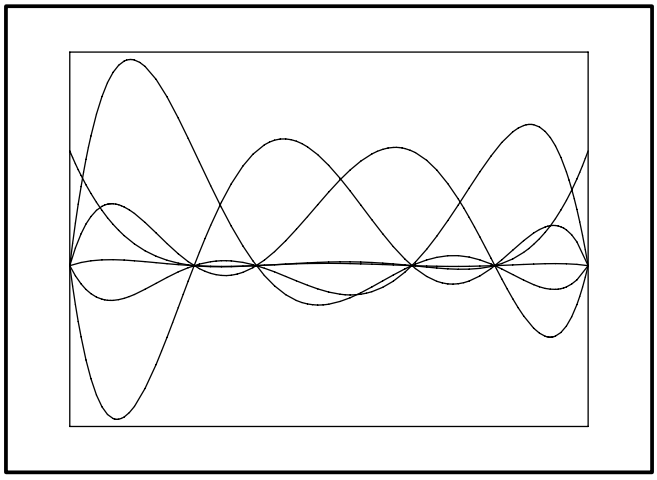


Figure 1: Graph of Typical Lagrange Basis Function

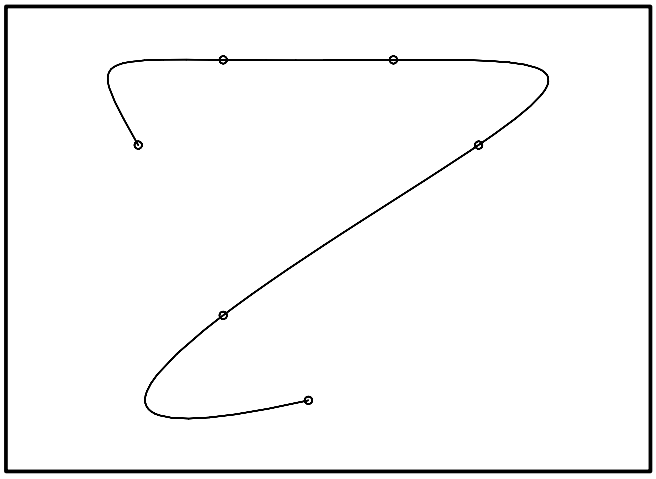


Figure 2: Graph of Typical Interpolants

Using the Lagrange basis function pictured in Figure 1, we obtain the following curves with various points.

For our purposes, there is a large problem with Lagrange interpolation. The interpolant can have wild variation. The key element that controls this oscillation is the spacing of the parameter values. The functional values of the basis functions are in fact not between 0 and 1 (an important property to be discussed later). In fact, another major problem is the fact that in this manner of interpolation the values of the Lagrange basis function $L_i^n(t)$ is not necessarily close to zero when $|t - t_j| < \epsilon$. We do have $L_i^n(t_j) = 0$ for $i \neq j$, but the value $L_i^n(t)$ is not uniformly close to zero when t is close to t_j. The derivative of the basis functions can be very large at $t = t_j$.

To demonstrate the possibility of wild variation, we show an example of a set of basis functions that generate some wild variation and an example of interpolating with the same data points as in 2.

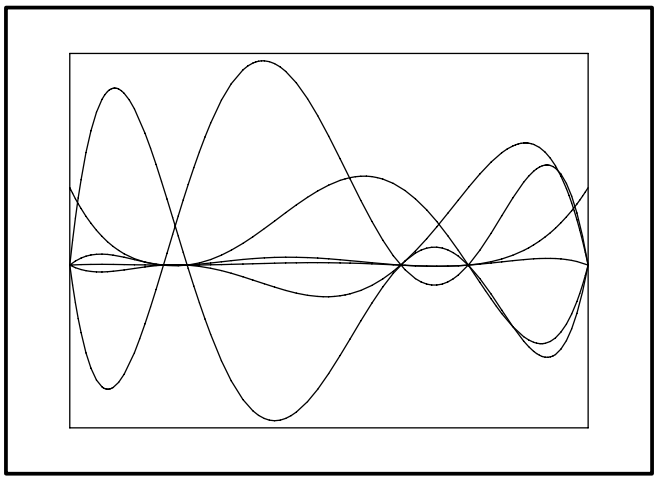


Figure 3: Wilder Variation in a Lagrange Basis Function

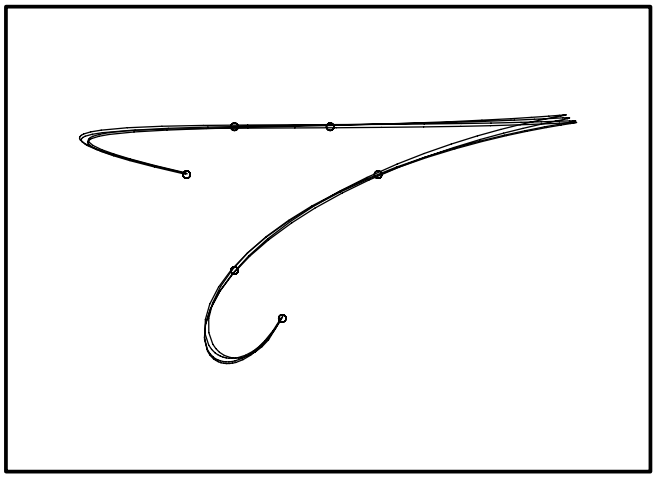


Figure 4: Interpolations with Wilder Variation Lagrange Basis Functions

We show another example of basis functions with wild variation and another set of examples of interpolation using the basis functions above with roughly the same data points as in 2.

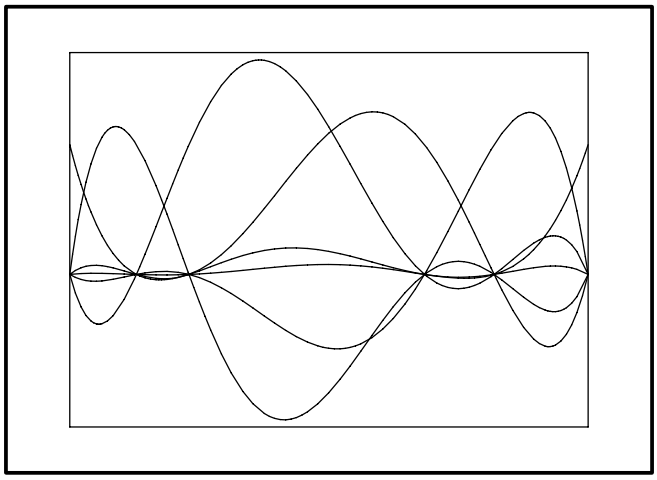


Figure 5: Wilder Variation in a Lagrange Basis Function

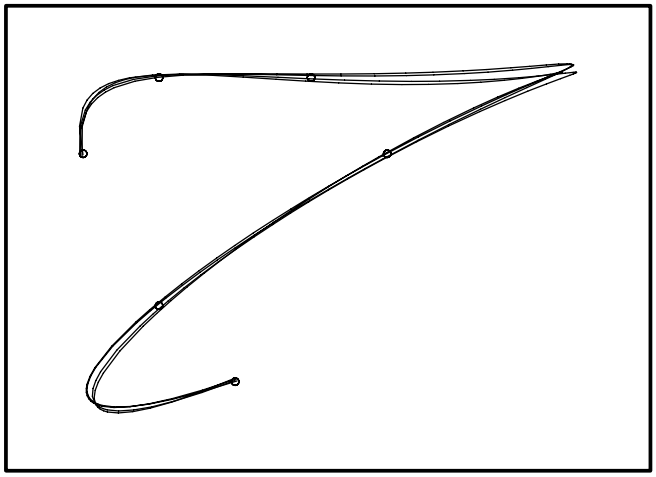


Figure 6: Interpolations with Wilder Variation Lagrange Basis Functions

Here is a third set of basis functions with wild variation and another set of examples of interpolation with roughly the same data points as in 2.

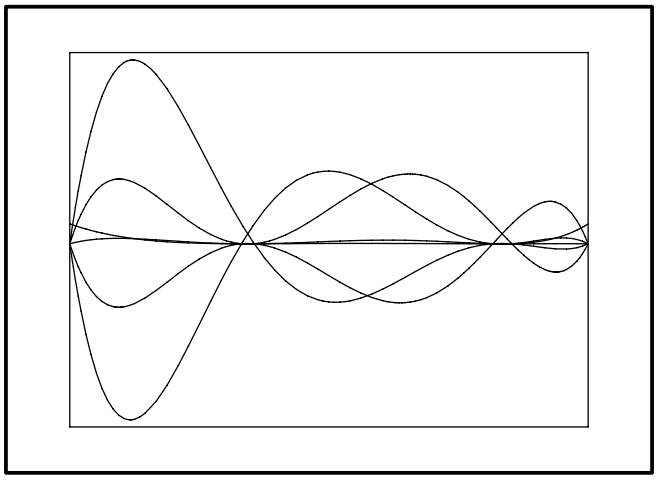


Figure 7: Wilder Variation in a Lagrange Basis Function

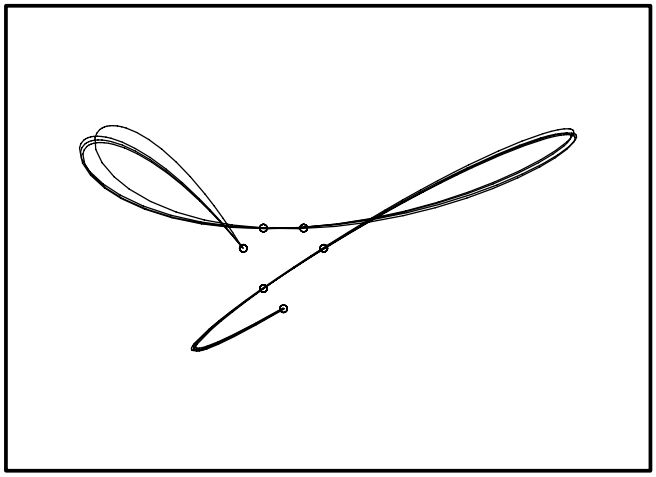


Figure 8: Interpolations with Wilder Variation Lagrange Basis Functions

The parameter values used above were chosen semi-randomly. You could argue that since the parameter values were not chosen to represent properties of the data we obtained possibly nonsensical results. The figures below represent using chord length parameter values, which can still have some problems.

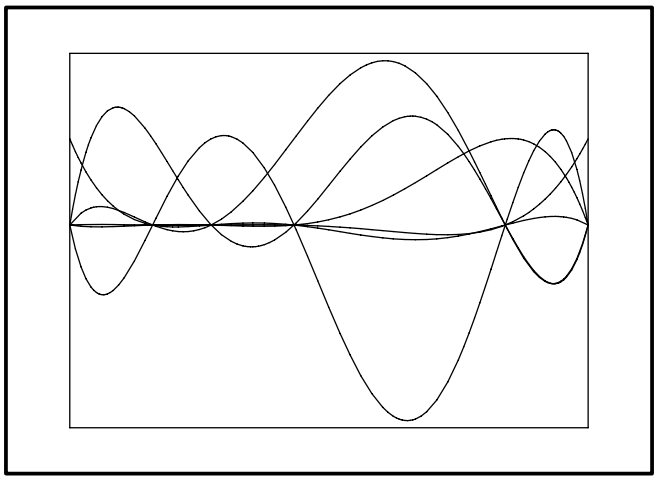


Figure 9: Lagrange Basis Function for Data with Chord Length Parameter Values

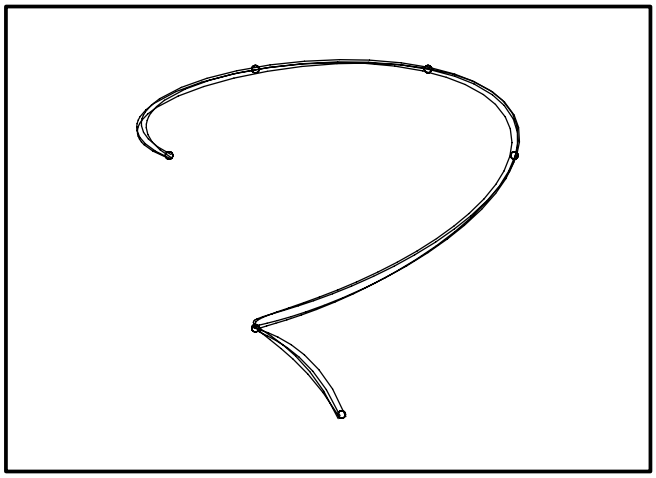
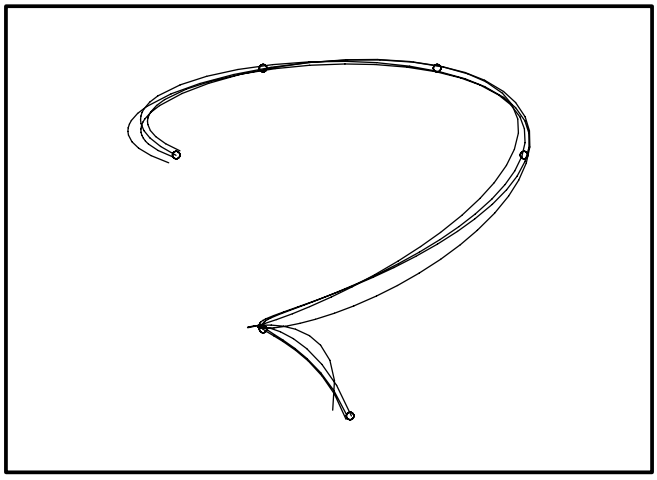


Figure 10: Interpolations with Chord Length Parameter Lagrange Basis Functions



10.2 Problems and Numerical Stability

One of the problems with Lagrange interpolation is that there is no clear indication as to where the curve will lie with the given data points. The parametrization of the curve greatly affects the location of the curve. Specifically, using roughly the same data points, we obtain vastly different curves in the figures 2, 4, 6 and 8. This is a great problem. We would like the data points to enforce on the curve some control without using extra structure as in chord length parameter values.

There are some problems with the methods presented so far in the course, especially when one starts using actual data instead of concocted examples. The principal problem with the polynomials methods we have introduced is that we are working with the monomial basis $\{1, t, t^2, \cdots, t^n\}$, which is numerically unstable. When we solve for the coefficients $\mathbf{a}_0$, $\mathbf{a}_1$, $\cdots$, $\mathbf{a}_n$ of a polynomial curve, the data supplied is frequently not going to be exact and calculations performed will not be done in exact arithmetic. This means that the data and the calculations are only done to a fixed decimal place accuracy. In an interpolation problem, this possibly causes the curve not to pass through the desired points. In addition, the curves produced by these methods can vary greatly under small variations of the data. To see this, run the Maple worksheet available on Angel and look at the effect of using different decimal place accuracy, also see figure below.

Other issues to consider are the design of the relevant basis functions. The monomial basis functions $\{1, t, t^2, \cdots, t^n\}$ are designed for problems where Taylor polynomial data is supplied, the value and derivatives at one point. Lagrange basis functions are designed for interpolation problems. Hermite basis functions are designed for interpolation problems involving function values and derivative values at different endpoints. Each of these types of basis functions are designed for different purposes and thus have different types of error propagation involved. For instance, polynomials written in the monomial basis have small errors (given small roundoff error in the coefficients) near $t = 0$, but the error grows as t becomes larger. The Lagrange basis functions have small error near each given parameter value $\{t_i\}$, but the error from the "true curve" can grow large in between the given parameter values, due to the oscillations in the basis functions. Hermite polynomials have similar problems with the monomial basis. The advantage is that the data at the second endpoint sort of corrects the error propagation.

If one is truly interested in error analysis of the methods introduced in this chapter, they

should consult a text on Numerical Analysis and take a course in Numerical Analysis. For our purposes, these methods while conceptually easy are not good for design problems, with the exception of Hermite polynomials which do lend themselves to design and creation of curves. In the next few days, we will introduce a better method at least for numerical stability and for design of curves.

Exercises

1. (Computational) Compare the basis functions and the interpolants for the data points

x	1.0	2.0	2.5	1.0	0.0
y	0.0	1.0	4.0	3.0	-1.0

   using equally spaced parameter values and chord length parameter values rounding values to three decimal point accuracy. Which in your opinion produces the better curve.

2. (Computational) Repeat the above exercise using the data points

x	1.0	2.1	2.4	1.0	0.2
y	0.0	0.9	4.1	2.9	-1.1

3. How do the interpolations and the basis functions compare in the previous two exercises.