449 Undergrad Thesis: work so far

Theory of polynomial interpolation

Given a set of data points yi, obtained by querying an unknown function at n data points xi, the goal of polynomial interpolation is to construct an approximating function that we can analyse in lieu of the queried function. To do this we define a polynomial interpolant, a polynomial function p of degree less than or equal to n that satisfies the following criteria:

p(xi) = yi i = 1 to n, yi = f(xi)

A piece-wise polynomial interpolant can be defined by dividing the domain xi into subdomains, and constructing a polynomial interpolant of a specified degree for each subdomain. Additional constraints can be satisfied to ensure continuity across the domain.

For example, a polynomial interpolant of degree 3, also known as a cubic spline, would satisfy the following:

pi(x) = ai + bi(x-xi) + ci(x-xi)^2 + di(x-xi)^3

where pi is the cubic polynomial from xi to xi+1, that passes through both of those points.

The constraints for continuity would be the following:

pi(xi+1) = fi(xi+1) = pi+1(xi+1) for continuity

pi’(xi+1) = pi+1’(xi+1) for continuity of the first derivative

pi’’(xi+1) = pi+1’’(xi+1) for continuity of the second derivative

Additional constraints govern the endpoints, determined by boundary conditions.

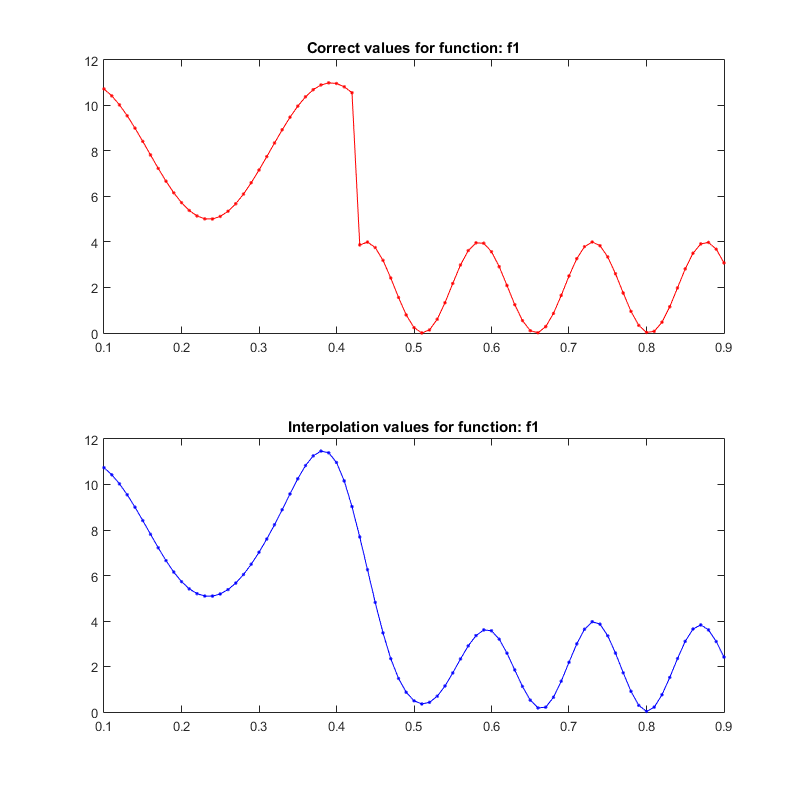
Part of the goal of this project involves calculating the order of accuracy of a polynomial interpolant, if we break certain continuity conditions. The order of accuracy here refers to the highest level of derivative that determines the maximum error bound of the interpolant. For example, if we were given a cubic spline interpolation, which is governed by the following error formula:

|f – p | <= (5/384) \* | f’’’’ | \* h4 - *A First Course In Numerical Methods*, Ascher and Grief

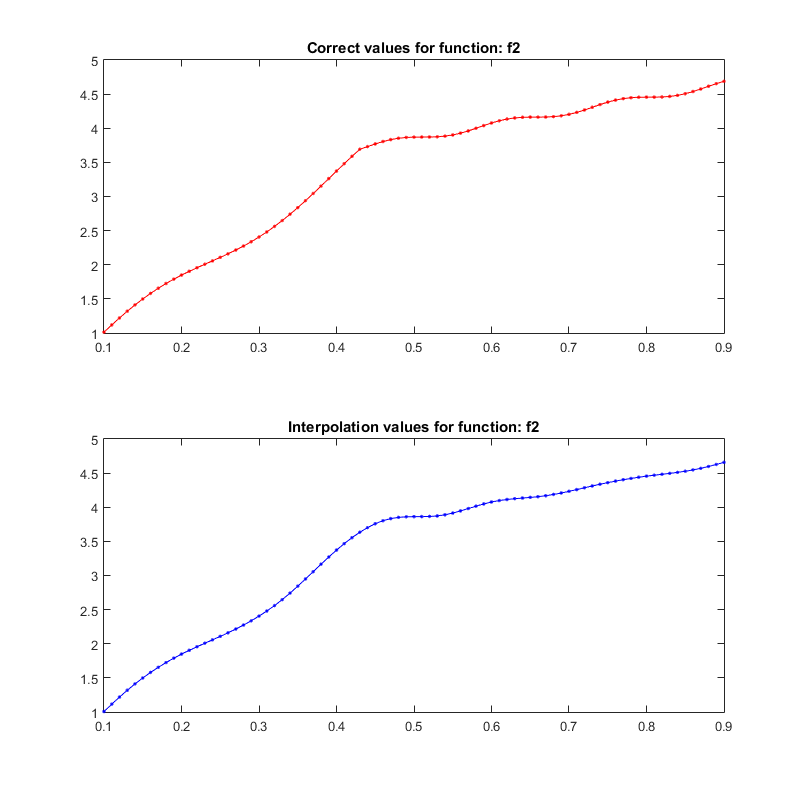
The order would be 4, as it’s the fourth derivative that determines the maximum error bound. If we were to plot a log log plot of the error against the discretization (as shown further below), the error would closely follow a line of slope -4.

One of the steps involved in determining how robust a given algorithm is is to apply it to functions with lower continuity than is required by the algorithm. The expected result is that the slope of the error will be the maximum of either the highest level continuity or the upper error bound of the algorithm. A cubic spline, for example, should give us slopes of 0, -1, and -2, for piecewise, first degree continuous, and second degree continuous functions. For functions of greater continuity, we should just see slopes of -3.

In order to generate functions with limited continuity, we define a piece-wise function with a jagged discontinuity, and integrate it repeatedly. For example, the following function is entirely discontinuous, and so we expect the order of any polynomial interpolant to have an accuracy of constant order.



Once we integrate this once, we have a continuous function with a discontinuous first derivative. We refer to this as first-order continuity. Any polynomial interpolant is expected to have an order of accuracy no greater than 1.



The ultimate goal of this project is to design a method of testing interpolation routines, to produce a measure of reliability beyond simply whether the routine converges or not. There are two main parts to this. The first is to calculate the order of a routine with reasonable certainty; we do this by first integrating a piecewise function to generate a set of functions with varying levels of continuity, and then running each of those through the interpolation routine to see how the error behaves with varying continuity and discretization. The second part, which is still in development, is to apply mutation testing to the routine and see how this effects with slopes derived in the first part.

The ideal outcome is that any perturbation results in a decreased order of accuracy, implying that the current version of the routine is the best we can get. Perturbations that result in the same or higher order of accuracy should point to errors or redundancies in the interpolation routine.

In order to find the order of a given interpolation routine, we currently run it through our matlab code, interpolateConvergence. This code performs the following steps:

1. Select a domain of query points: these points will be compared to the true solution to verify the accuracy of the method.
2. Iterate through a range of discretizations, using them to produce a set of sample points to pass to the interpolation.
3. Optionally, randomly perturb the domain of sample points.
4. Compare the output of the interpolation using the query points and discretizations to find sets of mean and max errors
5. Plot the min and max error against the discretization on a loglog plot

So far, we see the expected orders of accuracy for our methods, unless random perturbation of points is used. In that case, the two cubic spline methods (cublicSpline – my implementation, and spline – matlab’s implementation) drop to second order, or equal to piecewise linear’s order of accuracy.