Project 2: Polynomial Interpolation

Harper Wang, Cameron Stephens

October 2022

Contents

1 Introduction **3**

2 Mathematical Issues **4**
2.1 Construction of Polynomial Interpolator 4
2.2 Investigating Runge's Phenomenon 4
2.2.1 The Behavior of P_n as n Increases 4
2.3 Solving Runge's Phenomena 8
2.4 Interpolating Even Functions With Odd Polynomials 9
2.4.1 $P_7(x)$ Versus $P_7(-x)$ 10
2.4.2 Is $P_7(x)$ an Even or Odd Polynomial? 11
2.4.3 How Does the Selection of Nodes Preserve or Destroy 'Evenness?' 12

3 Programming Issues **13**
3.1 Construction of Polynomial Interpolator 13
3.2 Spline Interpolation Versus Newton's Formulation 14

4 Conclusion **15**

1 Introduction

Say you are given a set of data points with no noise, and you want to find the polynomial that best fits this set of nodes. Then the you would want to perform polynomial interpolation. The difference between polynomial interpolation and other forms of function approximation is that an interpolated polynomial must be equal to the original function exactly at each node, rather than only approximately equaling the polynomial at any given data point. As a result, interpolated polynomials that are not piece wise must be of degree one less than the number of nodes used in the interpolation. Higher degree interpolated polynomials can behave in wildly different ways depending on how you select the nodes over which they will be interpolated, as will be explored later in this paper. Additionally, there is a computationally optimal way to compute and evaluate these interpolated polynomials, which will be discussed as well.

2 Mathematical Issues

2.1 Construction of Polynomial Interpolator

An algorithm that can efficiently take a set of nodes and interpolate those nodes into a polynomial is given via Newton's formulation. Newton's formulation is more computationally efficient than the Lagrange formulation of polynomial interpolation; but, due to the uniqueness theorem, yields an equivalent polynomial.

2.2 Investigating Runge's Phenomenon

For this section, we will be looking at the following function over the interval $[-1, 1]$:

$$f(t) = \frac{1}{1 + 25t^2}.$$

2.2.1 The Behavior of P_n as n Increases

We took equal spacing between all nodes in the range and plotted $P_n(x)$ for degrees 6, 12, and 20.

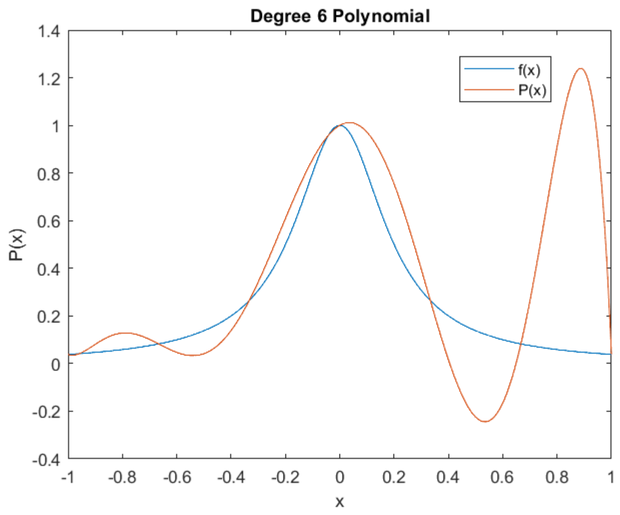

Figure 1: Plot of the degree 6 interpolated polynomial.

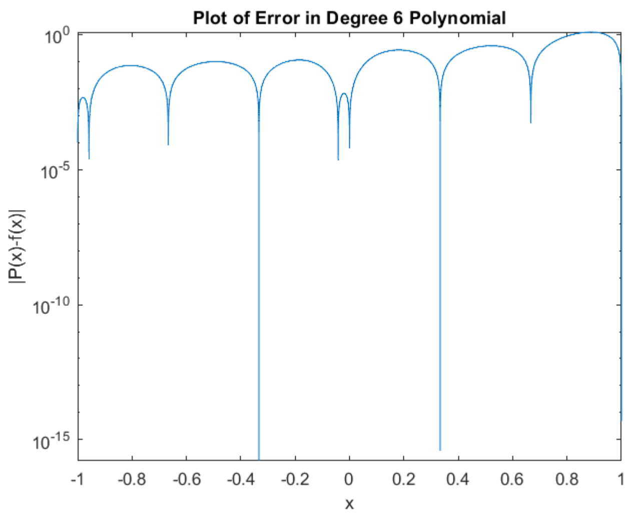

Figure 2: Plot of the error in the degree 6 interpolated polynomial.

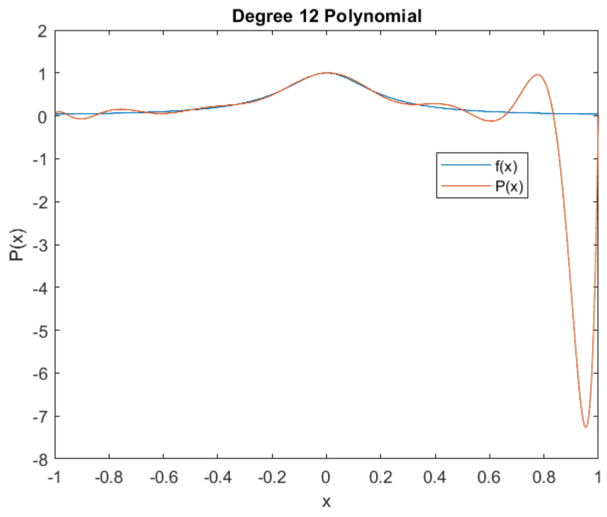

Figure 3: Plot of the degree 12 interpolated polynomial.

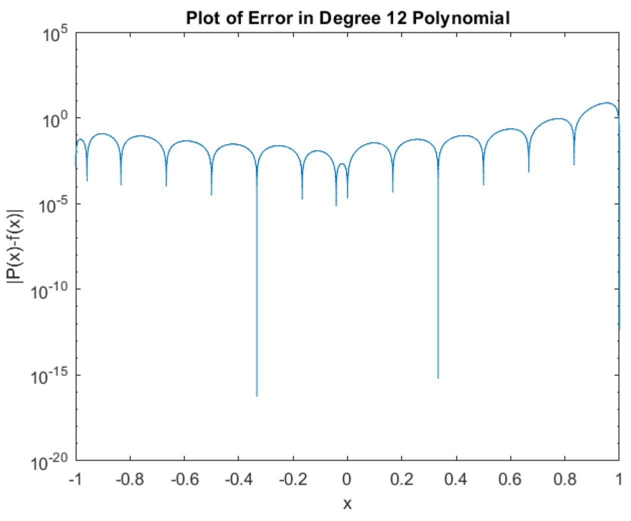

Figure 4: Plot of the error in the degree 12 interpolated polynomial.

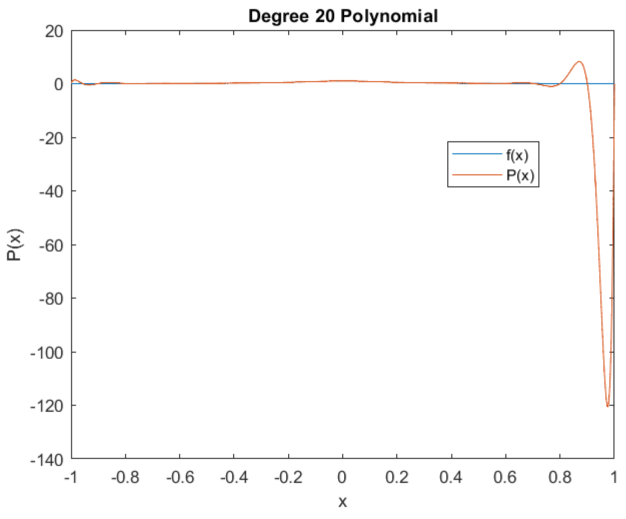

Figure 5: Plot of the degree 20 interpolated polynomial.

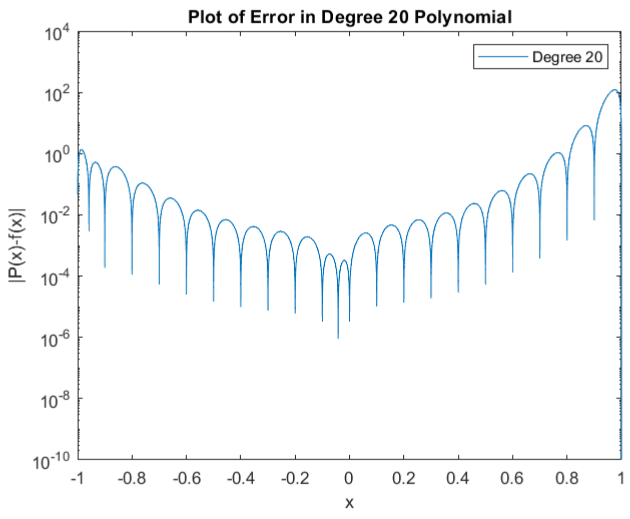

Figure 6: Plot of the error in the degree 12 interpolated polynomial.

We would expect increasing the number of nodes to increase the accuracy of an interpolated polynomial. But what we see is that with evenly spaced nodes, the error actually tends to increase as the number of nodes increases. This is known as Runge's phenomena.

2.3 Solving Runge's Phenomena

Runge's phenomena is caused by not having enough nodes placed towards the edges of the domain over which we are interpolating. Equally spacing the nodes nodes while increasing the number of nodes and the degree of the interpolated polynomial causes the error in the polynomial interpolated to increase dramatically. As seen in the above figures, most of the error occurs at the right end of the graph. Because polynomial interpolation guarantees that the interpolated polynomial will be equal to the original function at each node, it is natural to wonder what happens when we shift more of the nodes to the end of the graph where most of the error is occurring. We have done this shifting according to the Chebyshev nodes for a degree 20 polynomial.

$$x_k = \cos\frac{2k-1}{2n}\pi, k = 1, ..., 21.$$

Choosing these nodes significantly reduces the error and creates the following polynomial.

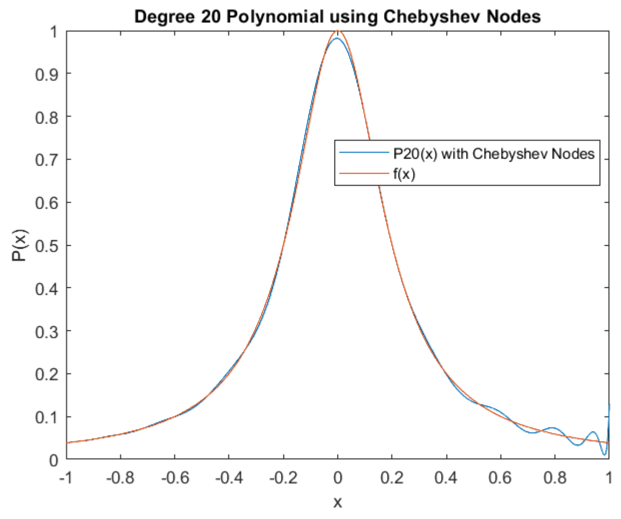

Figure 7: Plot of a degree 20 interpolated polynomial using Chebyshev nodes.

As shown in the figure, placing nodes towards the outer section of the graph reduces error due to oscillations as x approaches 1.

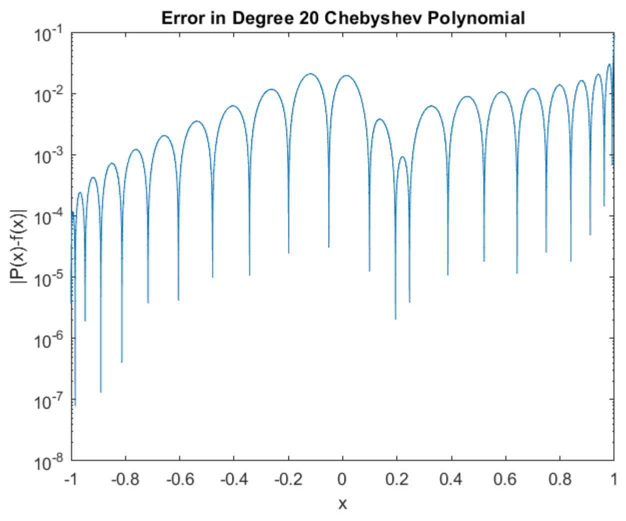

Figure 8: Plot of the error in the degree 20 interpolated polynomial constructed from Chebyshev nodes.

2.4 Interpolating Even Functions With Odd Polynomials

When interpolating an even function over a symmetric interval, does the interpolated polynomial inherit the trait of symmetry even if $P_n(x)$ is an odd polynomial?

2.4.1 $P_7(x)$ Versus $P_7(-x)$

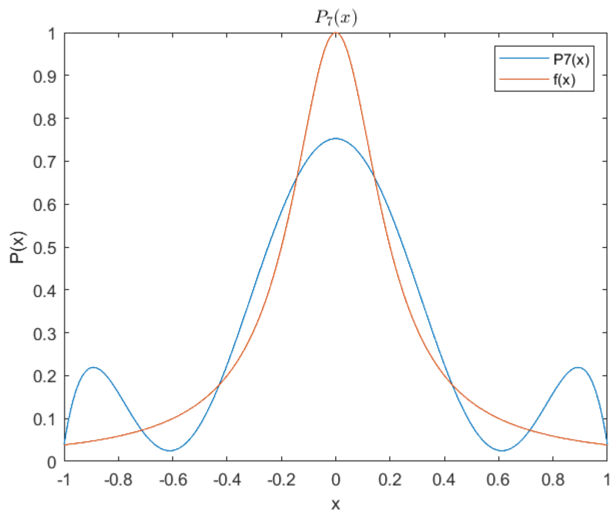

Figure 9: Plot of $P_7(x)$.

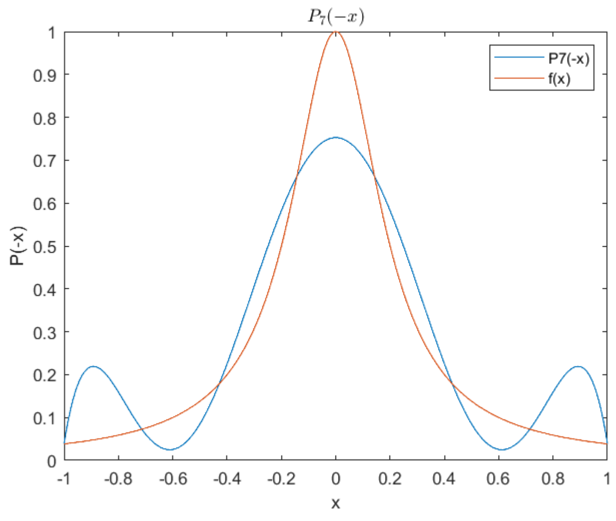

Figure 10: Plot of $P_7(-x)$.

When constructing $P(x)$ from symmetric data, it is nearly impossible to tell the difference between $P(x)$ and $P(-x)$.

2.4.2 Is $P_7(x)$ an Even or Odd Polynomial?

To determine if $P_7(x)$ is even or odd, we plotted the difference between $P_7(x)$ and $P_7(-x)$.

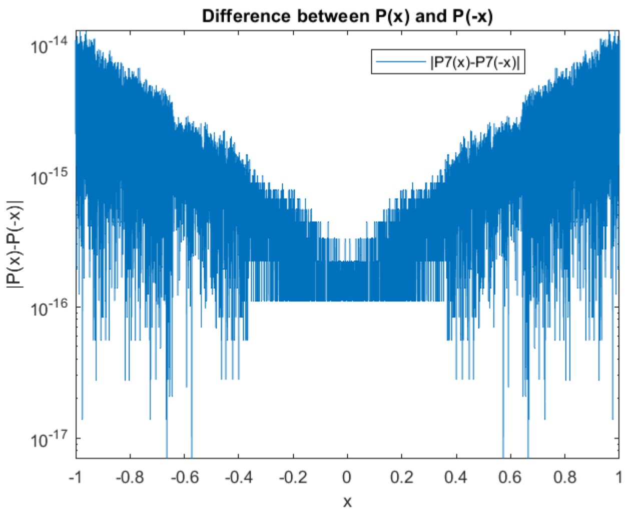

Figure 11: Plot of $|P_7(-x) - P_7(x)$.

The error between $P_7(x)$ and its reflection is very small, even on the order of machine error. This, however, is not enough evidence to prove that $P_7(x)$ is an even function.
When expanding $P_7(x)$ into its natural form, we find that the leading coefficient associated with the x_7 term is $7.35561\dot{0}^{-5}$. This coefficient is much smaller than all the other coefficients, so the x^7 term will not noticeably affect the values of $P_7(x)$ for small values of x, but it will still affect it a small amount. Since $P_7(x)$ has nonzero coefficients for odd exponents, we can conclude that $P_7(x)$ is an odd function.

2.4.3 How Does the Selection of Nodes Preserve or Destroy 'Evenness?'

As seen with $P_7(x)$ above, selecting symmetric data will cause an interpolated polynomial to act symmetric locally by having odd coefficients that are very small numbers. But how does selecting asymmetric data from the same even function influence the interpolated polynomial. To explore this, we selected asymmetric nodes for a degree 7 polynomial and plotted $P_7(x)$ and $P_7(-x)$. From the graph we can tell that the new polynomial, $P_7(x)$, did not inherit the

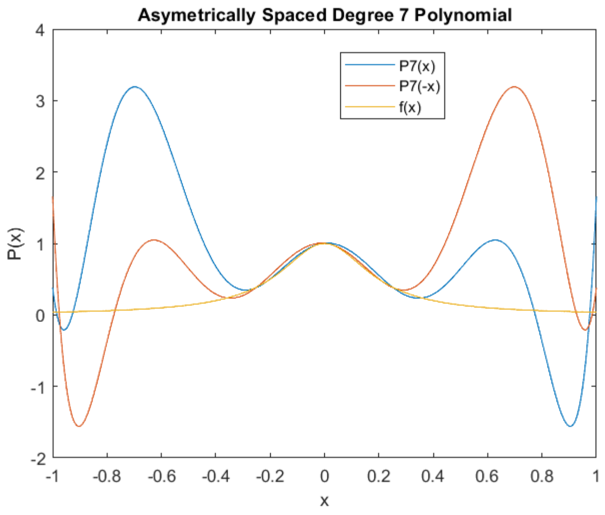

Figure 12: Plot of $P_7(-x)$ and $P_7(x)$ for asymmetric nodes.

symmetry of $f(x)$ when selecting asymmetric nodes.

3 Programming Issues

3.1 Construction of Polynomial Interpolator

The Newton formulation of polynomial interpolation was implemented as follows.

```
function [ylookup,Coeff] = table1(x,y,xlookup)
% define n as the number of nodes
n=length(x);

% array of coefficients first defined as array of f_i values
Coeff = y;

%implementation of Newton Backwards DD method
for j = 2:1:n
    for i = n:-1:j
        Coeff(i) = ( Coeff(i)-Coeff(i-1) )/ ( x(i)-x(i-j+1) );
    end
end

%Calculate P(x) from Horner's Alg
P = Coeff(n);

for k = n:-1:1
    P = P.*( xlookup' - x(k) ) + Coeff(k);
end
ylookup = P';
end
```

This function takes a set of n nodes, uses Newton's backwards divided difference method for polynomial interpolation to construct the coefficients (in Newton's form) for a polynomial of degree n-1. A variation of Horner's algorithm is then used to evaluate the polynomial at every x-value specified in the vector xlookup. The $P_{n-1}(x)$ corresponding to every xlookup value is then passed to the variable ylookup, which is then passed to the global variable outside the function. The vector of all the Newton's form coefficients is also passed outside the function for the case where we need to find the coefficients of a polynomial in its natural form.

3.2 Spline Interpolation Versus Newton's Formulation

Spline interpolation creates a piece-wise polynomial that fits several different polynomials to different subregions of the domain being interpolated over. This piece-wise function is then made smooth by matching both the values of $f(x)$ at each node as well as the the slope and curvature of each piece of the interpolated polynomial at each node. When using MATLAB's interp1 function and comparing it to the degree 20 polynomial we constructed from the Chebyshev roots, we get the following graph. From the graph, we can tell that the

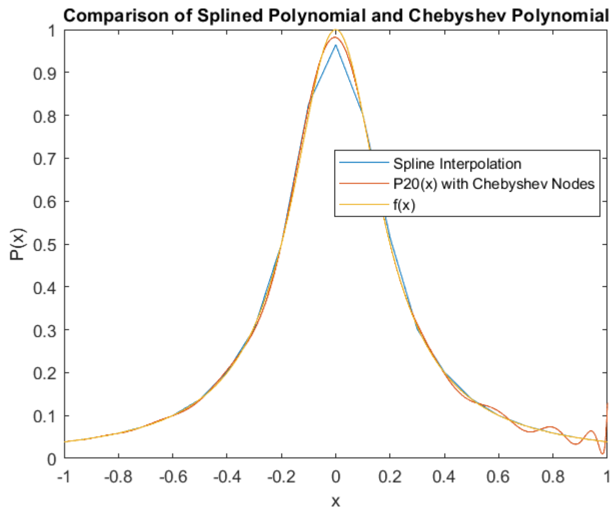

Figure 13: Plot of spline interpolated polynomial of 20 roots and the optimal $P_20(x)$.

spline interpolation does much better for larger values of x, but the polynomial interpolated using Newton's formulation does better in the region around x=0.

4 Conclusion

When interpolating polynomials, it is most efficient to use Newton's backward divided difference method while storing all coefficients into a 1D vector. Then when evaluating many points of the interpolated polynomial, Horner's algorithm should be applied to all domain values being evaluated in parallel through vectorization.
There are more than just computational obstacles to consider when approximating a function with polynomial interpolation. One also wants to consider how to optimally select nodes to minimize the overall error of the polynomial. Since an interpolated polynomial must exactly equal the original function at each node, error in a region can be reduced by placing more nodes in a region prone to error. This concept can be applied to tame any diverging solutions. The optimal selection of nodes is by selecting the Chebyshev nodes to interpolate over. The formula for Chebyshev nodes is

$$x_k = \cos\frac{2k-1}{2n}\pi, k = 1, ..., n.$$

Important things to note about this formula is that will only select x values on the range [-1,1], so any function must be scaled to be over that interval; and that the x_k values defined by the Chebyshev nodes are symmetric. Selecting symmetric nodes will allow the interpolated polynomial to inherit most of the symmetry of the function being interpolated (if the function has symmetry), this is important when choosing to approximate even functions with an odd degree polynomial.
Lastly, how does Newton's formulation for polynomial interpolation compare to the spline method for interpolation? In the region about $x = 0$, Newton's formulation does a better job of modeling f(x); but in the region outside of $x = 0$, the spline interpolation does a better job of modeling f(x). So a high degree polynomial is very good at modeling small sections of curves, but for larger domains it is better to construction a piece-wise function of many lower degree polynomials to avoid the oscillations that higher degree polynomials have towards the end of the domain of interpolation.