MT3802 Project 2: Report

Question 1: Polynomial interpolation

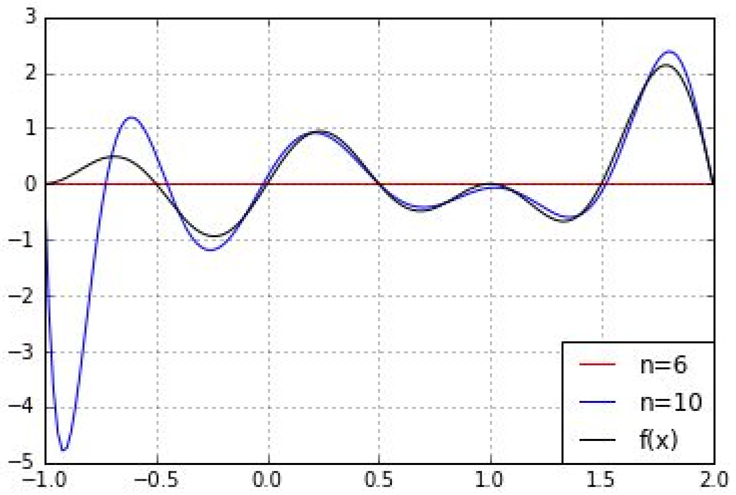
Method

At n=6, we have $n+1$=7 equally spaced points across the interval [-1,2]. The intervals are $h = (b-a)/n = ½$. So $(x = a+kh, f(x)) = \{(-1,0), (-0.5,0), (0,0), (0.5,0), (1,0), (1.5,0), (2,0)\}$ where $k = \{0, \ldots, n\}$. Similarly for n=10 we find 11 uniformly spaced coordinates. Our $f(x) = (1-x^2)sin(2\pi x)$.

Hence we construct the polynomial of order n: $p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0 x^0$. This gives us the polynomials $p_6(x) = a_6 x^6 + a_5 x^5 + \ldots + a_0 x^0$ and $p_{10}(x) = a_{10}x^{10} + a_9 x^9 + \ldots + a_0 x^0$ which can be solved using the n-dimensional matrices from lectures:

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

After defining both matrices for polynomials n=6 and n=10, they were plotted against f(x). This produced the plot:



As shown in the legend, plotting the fitted polynomial n=6 is almost equivalent to x = 0. It is, in fact, very nearby. At n=10, the fitted polynomial is visibly similar to f(x).

Error Analysis

Then, the errors were calculated between each polynomial n=6, n=10 and f(x), respectively. We expect n=10 to have a smaller error than for n=6.

To do this, I used the formula from lectures:

$$\max_{x_0 \le x \le x_1} |f(x) - p_n(x)| \le \frac{1}{(n+1)!} \max_{x_0 \le x \le x_1} \left| \prod_{k=0}^{n} (x - x_k) \right| \max_{x_0 \le x \le x_1} \left| f^{(k+1)}(x) \right|$$

This formula gives the maximum polynomial interpolation error, which is less than what it would be for the maximum linear interpolation error if that were to be calculated. The maximum polynomial interpolation error is greater than the theoretical error, which as shown is the maximum difference between the given function and the calculated polynomial over the interval.

For n=6, the maximum error was approximated to 118.538 and the theoretical error was 2.1315 on the interval. The theoretical difference was assumed to be the absolute maximum difference between f(x) and the fitted polynomial calculated earlier. This makes sense intuitively by looking at the graph earlier.

Observations

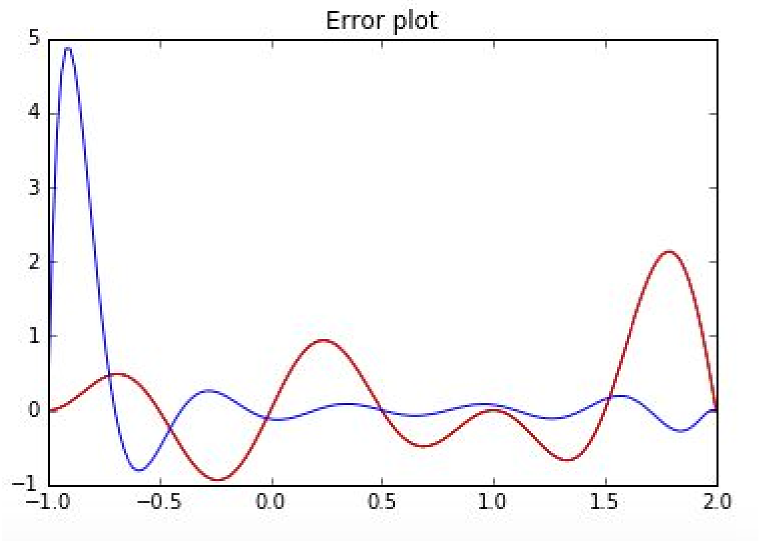
For n=10, I expected a smaller error for both the theoretical and maximum errors because the fitted polynomial calculated above appeared to match f(x) more accurately. Except for in the interval [-1,-0.5] where there appears to be a large outlier. The theoretical error I found was 4.8629, which reflects this outlier. The maximum error I found was 69.4152.

The maximum error at n=10 was, as expected, smaller than the error at n=6. This is because the degree of the polynomial was increased and more points were taken. We can see that increasing n improves polynomial interpolation in this case. In both cases, the above equation is confirmed: the theoretical error is less than the maximum polynomial interpolation error.

For both maximum errors, the values at the derivative were the most dominant in the equation. For example, the maximum $(n+1)^{st}$ derivative for n=6 is 797,887.99 and the $(n+1)^{st}$ derivative at n=10 is 3,762,740,836.19. Below is a plot of the theoretical error, f(x) - $p_n(x)$, for both cases on the interval. The error for n=6 is in red and for n=10 is in blue, as usual.

For n=6, the quality of the fit is particularly bad because of the nature of our f(x). At each of our equally spaced points on the interval with h=½, f(x) is zero because the $sin(2\pi x)$ factor has zeros at each of the given points. After solving the scheme Ax=b to find the roots of our polynomials, it was correctly observed that all the roots were near zero. Therefore we didn't get a good fit at all because it was nearly the line y=0.

An interpolating polynomial of degree less than seven will never provide a good approximation to this f(x). This is because f(x) has 7 turning points on our interval.



Question 2: Interpolation using Chebyshev polynomials

In this question, we try to reduce the error produced in polynomial interpolation by avoiding the dominance of the $(n+1)^{st}$ derivative.

Method

For n=6, we require $n+1$=7 points, or Chebyshev nodes of the Chebyshev polynomial $T_{n+1}(x) = T_7(x)$. $T_7(x)$ follows from the formula for a Chebyshev polynomial, namely $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$. Hence $T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x$.

After the roots are found by solving $T_7(x)$=0, we can calculate the nodes as

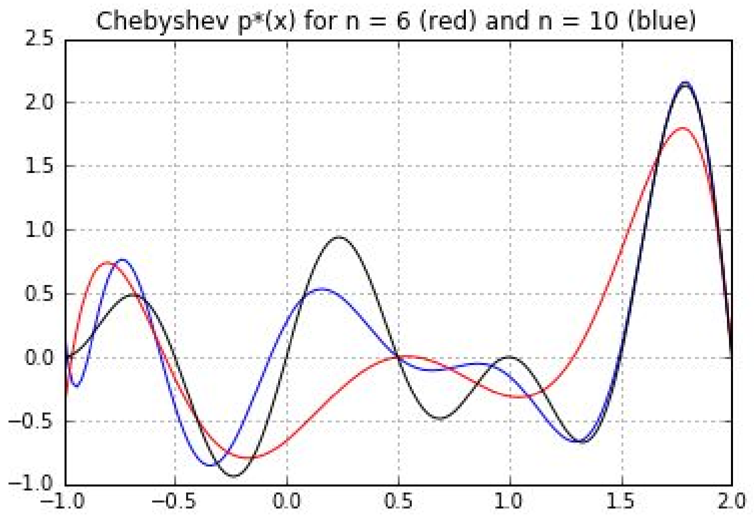
$$x_k = \frac{(a+b)}{2} + \frac{(b-a)}{2}x_{ck}$$

where x_{ck} are our calculated roots to $T_7(x)$ and [a,b] is our given interval [-1,2]. These nodes were: [[1.9624, 0.6674], [1.6728, 1.5904], [1.1508, -0.2634], [0.5, 0.0], [-0.1508, -0.7936], [-0.6728, 0.4842], [-0.9624, 0.01728]]. In the code, this can be proven by typing 'z' into the console. These nodes can be plotted by the definition "chebyshevnodes()" for $n+1$=7 in my code. Clearly, these nodes lie on the interval.

I did the exact same procedure for n=10. $T_{11}(x) = 1024x^{11} - 2816x^9 + 2816x^7 - 1232x^5 + 220x^3 - 11x$. $T_{11}(x)$=0 was solved and nodes were calculated using the formula for x_k (shown above). These nodes were: [[-0.9847, 0.002902], [-0.8645, 0.1902], [-0.6336, 0.4455], [-0.3110, -0.8379], [0.07740, 0.4646], [0.5, 0.0], [0.9226, -0.06955], [1.3110, -0.6666], [1.6336, 1.2422], [1.8645, 1.8630], [1.9847, 0.2815]].

We then use the same matrix formula from Question 1 but replace the evenly spaced points with the new Chebyshev nodes.

The new fitted Chebyshev polynomials using the Chebyshev nodes for n=6 and n=10 calculated above produce the plot below. More precisely defined, we say that they are the improved polynomials which interpolate f(x) on the zeros of $T_{n+1}(x)$. The function f(x) is plotted in black as usual.



Error Analysis

The theoretical and maximum errors were calculated. Note that in Question 1, we expected the theoretical error at each n to be less than the maximum error. We also observed that as n increases, the maximum error decreases. We can prove that Chebyshev interpolation decreases the error.

To do this, I used the formula from the lectures:

$$\|f(x) - p_n(x)\|_\infty \leq \frac{1}{2^n(n+1)!} \max_{x_0 \leq u \leq x_n} \left| f^{(n+1)}(u) \right| , \qquad x \in [-1, 1] .$$

Note that this is for x on the interval [-1,1]. This derives from property 1 of Chebyshev polynomials. Since $T_n(x) = cos(n\theta)$, $T_n(x)$ must lie on the interval [-1,1] as cos(n*theta) is never larger or smaller than [-1,1]. Chebyshev polynomials $T_{n+1}(x)$ have n+1 distinct roots symmetrically placed on the interior of this interval. Furthermore, the third property is also satisfied, that being that $T_{n+1}(x)$ attains a maximum of 1 $n+1$ times on the interval.

Also, the infinity norm of the difference between f(x) and a polynomial of degree n is essentially just the maximum difference at point x. We took that to be the theoretical error. This will always be less than the maximum error on the right hand side of the equation.

Following on from the theory outlined above, we got the following answers. At n=6, the theoretical error was approximately 1.2391, with a maximum Chebyshev interpolation error 5.8419. This confirms that indeed the theoretical error is less than the maximum error. At n =10, the theoretical error was approximately 0.4871 and the maximum Chebyshev interpolation error was found to be 2.29596. Again, the obvious is true, and the same idea from Question 1 follows, that increasing n gives a better approximation to the function. The maximum error will decrease as more points are observed.

Observations

In this case, n=6 is particularly bad because the fitted Chebyshev polynomial only has 5 turning points (i.e. maxima and minima) on the interval, whereas f(x) and the fitted Chebyshev polynomial for n=10 both have 7 turning points. This gives the fitted Chebyshev polynomial for n=6 an awkward look about it, as shown in the previous graph. Hence, n=6 gives a poor fit to our function.

Question 3: Interpolation using cubic splines

Method

Cubic splines provide an even better approximation to the curve, and decrease the error.

We aim to find a uniform spline, $S(x)$, that is both differentiable and continuously differentiable to order $n-1$ at all our uniformly spaced points. We try to find n polynomials each within their own equal intervals to make up a better approximation to f(x).

We consider $n+1$ uniformly spaced points across the interval for each case. We know we have $n+1$ unknowns in $n+1$ equations. We use the same points $(x_k, f(x_k))$ as in Question 1 for both cases.

We were given the formula that $S(x) = \sum_{k=-1}^{n+1} a_k B_k$, where a_k represents the $n+1$ unknowns and B_k represents the B-splines on the interval. The B-splines are basis functions that satisfy the continuity condition. For example, if the given interval is [a, b] = [x_0, x_N] then we must solve the following system of equations for the B-spline at k=0:

$$B_0(x) = \begin{cases} 0 & x \le x_0 - 2h \\ \frac{1}{6}(2h + (x - x_0))^3 & x_0 - 2h \le x \le x_0 - h \\ \frac{2h^3}{3} - \frac{1}{2}(x - x_0)^2(2h + (x - x_0)) & x_0 - h \le x \le x_0 \\ \frac{2h^3}{3} - \frac{1}{2}(x - x_0)^2(2h - (x - x_0)) & x_0 \le x \le x_0 + h \\ \frac{1}{6}(2h - (x - x_0))^3 & x_0 + h \le x \le x_0 + 2h \\ 0 & x \ge x_0 + 2h \end{cases}$$

But first, we solve the equation to find values for our $n+1$ unknowns, $a_0, ..., a_n$:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & & \dots & 0 \\ 1 & 4 & 1 & 0 & & \dots & \vdots \\ 0 & 1 & 4 & 1 & 0 & \dots & 0 \\ \vdots & & & & & & \vdots \\ \vdots & & \dots & 0 & 1 & 4 & 1 \\ 0 & & \dots & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} f(0) \\ 6f(1) \\ \vdots \\ \vdots \\ 6f(n-1) \\ f(n) \end{bmatrix}$$

Then, we find a_{-1} and a_{n+1} with the given formulae: $a_{-1} = 2a_1 - a_0$ and $a_{n+1} = 2a_n - a_{n-1}$. We add this to an array we've made containing $a_0, ..., a_n$ and get an array $a_{-1}, a_0, ..., a_n, a_{n+1}$ and convert it to a list.

We then calculate the n number of splines that make up our equation for $S(x)$ by making use of the definition for the B-spline above. Small adjustments are made to ensure continuity, namely: we let $x = x - kh$ in each B-spline segment. This is because we want each $B_k(x_k)$ in the equation to be $B_0(x - kh)$. Finally, we plot them within their unique domains on the interval [-1,2].

Observations

For n=6, the result was a disappointing "straight line" along the x-axis, as we found in Question 1. In fact, it is not quite straight upon closer observation, but on this axis it appears to be pretty flat. For n=10, a more interesting result appeared (in rainbow coloured segments). The graph confirms that we get a much better fit for f(x) using cubic splines rather than by using the previous methods above. The outlier we saw before is gone. The fit is much tighter and near the actual f(x), which is denoted in black.

In this case, n =6 proves to be a particularly bad fit to f(x). There is a larger distance between points, making the cubic spline interpolation algorithm unstable. Again, f(x) will be zero or near zero for multiples of π, as demonstrated in the 'bmatrix' command in my code. Such a matrix is almost singular, which is why we got the near flat line in this question. Furthermore, the condition number of its system is 6.0, (where $K(A) = \|A\| \times \|A^{-1}\|$). We can also just read it off the initial matrix. This can be shown in my work under the definition "conditionnumber1()" This is quite big and causes the system to be ill-conditioned. Small errors in the right hand side of the equation outlined in the method can cause huge differences in the outcome for undefined $a_{-1}, a_0, ..., a_n, a_{n+1}$.