Homework 2 Solutions

1. (17 points) (T) Suppose that $x_0 = \frac{1}{4}, x_1 = \frac{1}{2}, x_2 = 1,$ and $x_3 = 2.$

   (a) Use divided differences to compute the newton form of the third degree polynomial such that $p(x_0) = 1, p(x_1) = 2, p(x_2) = 5,$ and $p(x_3) = -1.$

   The following is the table of divided differences.

Divided Differences				
x_i	$\{f[x_i]\}_{i=0}^3$	$\{f[x_i,x_{i+1}]\}_{i=0}^2$	$\{f[x_i,x_{i+1},x_{i+2}]\}_{i=0}^1$	$f[x_0,x_1,x_2,x_3]$
$\frac{1}{4}$	1			
$\frac{1}{2}$	2	$\frac{2-1}{1/2-1/4} = 4$		
1	5	$\frac{5-2}{1-1/2} = 6$	$\frac{6-4}{1-1/4} = \frac{8}{3}$	
2	-1	$\frac{-1-5}{2-1} = -6$	$\frac{-6-6}{2-1/2} = -8$	$\frac{-8-8/3}{2-1/4} = \frac{-128}{21}$

   Using this, the Newton form of the interpolating polynomial is

$$p_N(x) = 1 + 4\left(x - \frac{1}{4}\right) + \frac{8}{3}\left(x - \frac{1}{4}\right)\left(x - \frac{1}{2}\right) - \frac{128}{21}\left(x - \frac{1}{4}\right)\left(x - \frac{1}{2}\right)\left(x - 1\right)$$

   (b) For the same data, write the Lagrange interpolating polynomial.

   The lagrange interpolating polynomial is given by

$$p_L(x) = \frac{\left(x - \frac{1}{2}\right)\left(x - 1\right)\left(x - 2\right)}{\left(\frac{1}{4} - \frac{1}{2}\right)\left(\frac{1}{4} - 1\right)\left(\frac{1}{4} - 2\right)} + \frac{2\left(x - \frac{1}{4}\right)\left(x - 1\right)\left(x - 2\right)}{\left(\frac{1}{2} - \frac{1}{4}\right)\left(\frac{1}{2} - 1\right)\left(\frac{1}{2} - 2\right)} + \frac{5\left(x - \frac{1}{4}\right)\left(x - \frac{1}{2}\right)\left(x - 2\right)}{\left(1 - \frac{1}{4}\right)\left(1 - \frac{1}{2}\right)\left(1 - 2\right)}$$
$$- \frac{\left(x - \frac{1}{4}\right)\left(x - \frac{1}{2}\right)\left(x - 1\right)}{\left(2 - \frac{1}{4}\right)\left(2 - \frac{1}{2}\right)\left(2 - 1\right)}$$

   or by simplifying,

$$p_L(x) = \frac{-64}{21}\left(x - \frac{1}{2}\right)\left(x - 1\right)\left(x - 2\right) + \frac{32}{3}\left(x - \frac{1}{4}\right)\left(x - 1\right)\left(x - 2\right)$$
$$- \frac{40}{3}\left(x - \frac{1}{4}\right)\left(x - \frac{1}{2}\right)\left(x - 2\right) - \frac{8}{21}\left(x - \frac{1}{4}\right)\left(x - \frac{1}{2}\right)\left(x - 1\right)$$

   (c) Show that these two forms are the same polynomial.

$$p_N(x) = 1 + 4x - 1 + \frac{8}{3}\left(x^2 - \frac{3}{4}x + \frac{1}{8}\right) - \frac{128}{21}\left(x^2 - \frac{3}{4}x + \frac{1}{8}\right)\left(x - 1\right)$$
$$= 4x + \frac{8}{3}x^2 - 2x + \frac{1}{3} - \frac{128}{21}\left(x^3 - \frac{7}{4}x^2 + \frac{7}{8}x - \frac{1}{8}\right)$$
$$= 2x + \frac{8}{3}x^2 + \frac{1}{3} - \frac{128}{21}x^3 + \frac{32}{3}x^2 - \frac{16}{3}x + \frac{16}{21}$$

$$= -\frac{128}{21}x^3 + \frac{40}{3}x^2 - \frac{10}{3}x + \frac{23}{21}$$

Then, $p_L(x) = \frac{-64}{21}\left(x^2 - \frac{3}{2}x + \frac{1}{2}\right)\left(x-2\right) + \frac{32}{3}\left(x^2 - \frac{5}{4}x + \frac{1}{4}\right)\left(x-2\right)$

$$- \frac{40}{3}\left(x^2 - \frac{3}{4}x + \frac{1}{8}\right)\left(x-2\right) - \frac{8}{21}\left(x^2 - \frac{3}{4}x + \frac{1}{8}\right)\left(x-1\right)$$

$$= \frac{-64}{21}\left(x^3 - \frac{7}{2}x^2 + \frac{7}{2}x - 1\right) + \frac{32}{3}\left(x^3 - \frac{13}{4}x^2 + \frac{11}{4}x - \frac{1}{2}\right)$$

$$- \frac{40}{3}\left(x^3 - \frac{11}{4}x^2 + \frac{13}{8}x - \frac{1}{4}\right) - \frac{8}{21}\left(x^3 - \frac{7}{4}x^2 + \frac{7}{8}x - \frac{1}{8}\right)$$

$$= \left(-\frac{64}{21} + \frac{32}{3} - \frac{40}{3} - \frac{8}{21}\right)x^3 + \left(\frac{32}{3} - \frac{104}{3} + \frac{110}{3} + \frac{2}{3}\right)x^2$$

$$+ \left(-\frac{32}{3} + \frac{88}{3} - \frac{65}{3} - \frac{1}{3}\right)x + \left(\frac{64}{21} - \frac{16}{3} + \frac{10}{3} + \frac{1}{21}\right)$$

$$= -\frac{128}{21}x^3 + \frac{40}{3}x^2 - \frac{10}{3}x + \frac{23}{21} = p_N(x)$$

This demonstrates that the Newton and Lagrange forms of the interpolating polynomial are equivalent.

2. (27 points) Given the points (x_i, y_i) for $i = 0 \ldots n$ the Newton form of the interpolating polynomial is

$$p(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}).$$

The textbook provides code for computing the coeffients by a divided difference table and evaluating the polynomial. I provided this code on the class website.

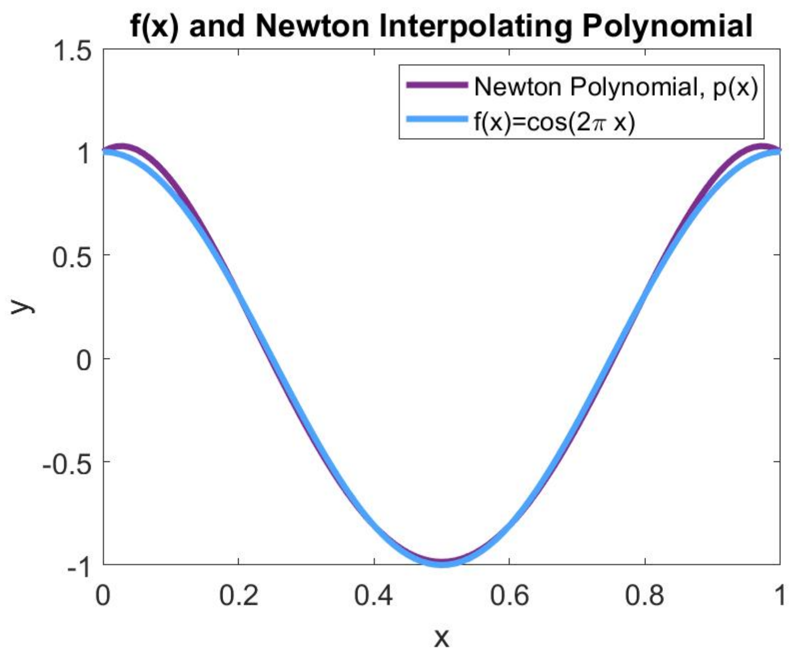
(a) Compute the coefficients of the Newton form of the interpolating polynomial for the fifth degree polynomial that interpolates $f(x) = \cos(2\pi x)$ at the points $x_j = j/5$ for $j = 0, \ldots, 5$. Display the results in a table.

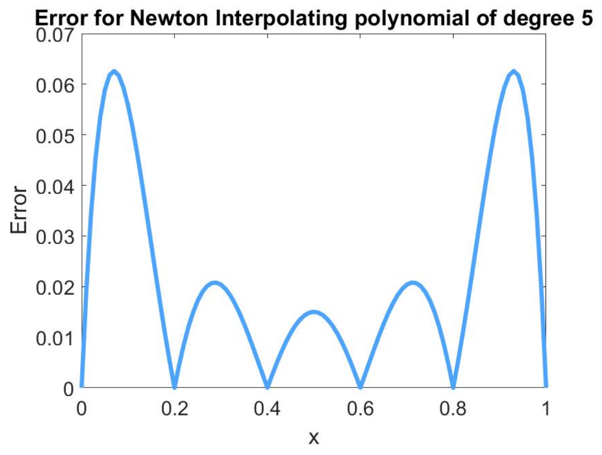
Here are the resulting coefficients of the 5th degree Newton interpolating polynomial.

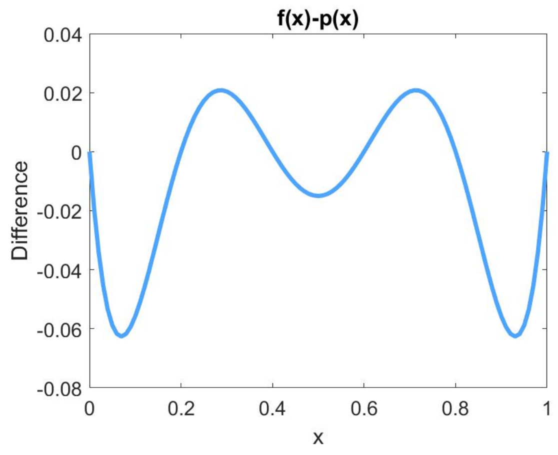
j	Coefficient
0	1
1	-3.45491502812526
2	-5.33813728906053
3	32.1892702473904
4	-40.236587809238
5	$4.9737991503207(10)^{-14}$

(b) Make a plot of $f(x)$ and the interpolating polynomial, $p(x)$, from part (a) on the same axes for $0 \le x \le 1$. Plot the difference of f and p for $0 \le x \le 1$.

Below are the resulting plots. Because of the wording, it is acceptable to plot either the difference or the absolute value of the difference (i.e. the error).







(c) Estimate the maximum of $|f(x) - p(x)|$ on the interval $[0, 1]$ by evaluating f and p for a large number of points between 0 and 1.

Based on the error plot, I found the errors for a a finer-spaced selection of points between $x = 0$ and $x = 0.2$. Based on that info, the error appears to be bounded by approximately 0.0626121.

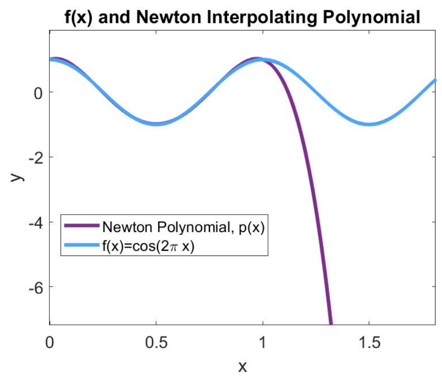
(d) Evaluate $p(1.5)$, $p(2.0)$, and $p(2.5)$ and report the results. Why are the values of p so different from the values of f at these points compared to points in $[0, 1]$?

The following table gives the results.

x	$f(x)$	$p(x)$	Error
1.5	-1	-23.2224883409374	22.2224883409374
2	1	-164.184771765594	165.184771765594
2.5	-1	-572.774054558612	571.774054558612

The errors are much worse for these x values because we have left the interval $[0, 1]$, which is the interval containing all of our interpolation points. As you can see in the plot below, the polynomial veers off to negative infinity starting soon after 1. This is because any polynomial will have end behavior of ∞ or $-\infty$ that occurs after it has completed all of its "oscillations." On the other hand, $\cos 2\pi x$ continues oscillating indefinitely. In order to keep representing a $\cos x$ function with a polynomial into another interval, we would have to continue adding points to "tie down" the polynomial to all of the osciallations in the new interval. In general, it is best to always have interpolation points throughout any interval in which you hope to approximate a function, but with data, this may not be the case. There are functions that may continue on the trajectiory of your interpolating polynomial, but that is not possible with a periodic function like this.

Another thing we can think about to answer this question is the error bound theorem we have used. For this fifth degree polynomial, using the fact that the n^{th} derivative of f will be bound in magnitude by $(2\pi)^n$, we know that an error bound is given by $\frac{(2\pi)^6|x(x-1/5)(x-2/5)(x-3/5)(x-4/5)(x-1)|}{6!} \leq \frac{(2\pi)^6(0.0011)}{6!} \approx 0.094 \underline{\textit{for } x \in [0,1]}$, which does bound the error, as we can see by comparing it to the value in part (c). On the other hand, if $x \in [1, 2.5]$, this error bound becomes $\frac{(2\pi)^6(58.5034)}{6!} \approx 4999.51$, which tells us we cannot guarantee a decent approximation.



3. (18 points) (T) Suppose you wish to find a polynomial p that satisfies the conditions

$$p(0)=0, \quad p'(0)=0, \quad p''(0)=0, \quad p(1)=1, \quad p'(1)=0, \quad p''(1)=0$$

What degree polynomial satisfies these conditions? Prove your claim. (It is not necessary that you $p(x)$, but finding it would be one way to prove it.)

With 6 pieces of information, I expect to have a 5th degree polynomial. To prove it, let's prove that $\exists$ coefficients a,b,c,d,e,f $s.t.$ $p(x)=a+bx+cx^2+dx^3+ex^4+fx^5$ satisfies the above conditions.

$p(x)=a+bx+cx^2+dx^3+ex^4+fx^5$

$p(0)=0 \implies a=0$

$p'(x)=b+2cx+3dx^2+4ex^3+5fx^4$

$p'(0)=0 \implies b=0$

$p''(x)=2c+6dx+12ex^2+20fx^3$

$p''(0)=0 \implies c=0$

Now we have eliminated some of the variables, so $p(x)=dx^3+ex^4+fx^5$ and we need to find d,e,f such that:

$1=d+e+f, \quad 0=3d+4e+5f, \quad$ and $0=6d+12e+20f$.

As a matrix problem, this gives us: $\begin{pmatrix}1 & 1 & 1\\ 3 & 4 & 5\\ 6 & 12 & 20\end{pmatrix}\begin{pmatrix}d\\ e\\ f\end{pmatrix}=\begin{pmatrix}1\\ 0\\ 0\end{pmatrix}$

Then, $\begin{vmatrix}1 & 1 & 1\\ 3 & 4 & 5\\ 6 & 12 & 20\end{vmatrix}=(80-60)-(60-30)+(36-24)=2\neq 0$. Therefore, there exists a unique solution for our coefficients, and hence a fifth degree polynomial exists that satisfies the conditions.

4. (42 points) Given the nodes

$$a=x_0<x_1<\cdots<x_n=b,$$

let S be the piecewise Hermite cubic interpolant of f at these nodes. That is S interpolates f and f' at these nodes, and on each subinterval $[x_i,x_{i+1}]$, S coincides with a cubic polynomial $S_i(x)$.

(a) Let $h=\max_i(x_{i+1}-x_i)$ and assume that f has at least four continuous derivatives on $[a,b]$. Show that

$$|f(x)-S(x)|\leq \frac{h^4}{384}\max_{\xi\in[a,b]}\left|f^{(4)}(\xi)\right|$$

Let's begin by looking at a Hermite cubic interpolant, $H_3(x)$ that interpolates two points $(x_0, f(x_0)), (x_1, f(x_1))$ and also satisfies $H_3'(x_0) = f'(x_0), H_3'(x_1) = f'(x_1)$.

As we were able to find an error bound for non-Hermite interpolating polynomial by using the Newton form, let's write $H_3(x)$ in a form from class that was similar to Newton form.

Then $H_3(x) = f(x_0) + f'(x_0)(x - x_0) + \alpha(x - x_0)^2 + \beta(x - x_0)^2(x - x_1)$,

We are interested in the error from using $H_3(x^*)$ to approximate $f(x^*)$.

Following an idea from class, we can create a fourth degree polynomial, which I will call $P_4(x)$ that will interpolate f at x^*. To do this, we will add on a fourth degree polynomial that has 0 function values and derivatives at x_0 and x_1. In other words, we get

$P_4(x) = f(x_0) + f'(x_0)(x - x_0) + \alpha(x - x_0)^2 + \beta(x - x_0)^2(x - x_1) + \gamma(x - x_0)^2(x - x_1)^2$,

where γ is chosen so that $P_4(x^*) = f(x^*)$.

Then the error of estimating $f(x^*)$ with $H_3(x^*)$ is:

$|f(x^*) - H_3(x^*)| \ = |P_4(x^*) - H_3(x^*)| = |\beta|(x^* - x_0)^2(x^* - x_1)^2$

Now, I claim that $\exists \xi \in (x_0, x_1)$ such that $\gamma = \frac{f^{(4)}(\xi)}{4!}$.

Proof:

Let $P_4(x) = f(x_0) + f'(x_0)(x - x_0) + \alpha(x - x_0)^2 + \beta(x - x_0)^2(x - x_1) + \gamma(x - x_0)^2(x - x_1)^2$

be the 4th degree polynomial formed as discussed before, starting with the Hermite cubic polynomial and then selecting γ to interpolate another point, x^*.

Then Let $r(x) = f(x) - P_4(x)$, which has 3 zeros, x_0, x^*, x_1 .

According to Rolle's theorem, there exist $\xi_1 \in (x_0, x_1)$ and $\xi_2 \in (x_1, x_2)$ such that

$r'(\xi_1) = r'(\xi_2) = 0$.

Then, by the design of $P_4(x)$, we know that $r'(x)$ has 2 more zeros, at x_0 and x_1.

Therefore, $r'(x)$ has 4 zeros, and using the Generalized Rolle's Theorem, we then have that $\exists \xi \in (x_0, x_1)$ such that $r^{(4)}(\xi) = 0$.

Then, $0 = r^{(4)}(\xi) = f^{(4)}(\xi) - P^{(4)}(\xi) = f^{(4)}(\xi) - 4!\gamma$

$\implies 4!\gamma = f^{(4)}(\xi) \implies \gamma = \frac{f^{(4)}(\xi)}{4!}$. $\square$

Then, we have $|f(x) - H_3(x)| = |\gamma|(x - x_0)^2(x - x_1)^2 = \dfrac{|f^{(4)}(\xi)|(x - x_0)^2(x - x_1)^2}{4!}$ for some $\xi \in (x_0, x_1)$.

Then, our error bound for this Hermite cubic polynomial is

$$|f(x) - H_3(x)| \leq \frac{[\max_{[x_0,x_1]} |f^{(4)}(x)|] \ \ [\max_{[x_0,x_1]}(x - x_0)^2(x - x_1)^2]}{4!}$$

Then the max of the latter portion occurs at $x = \frac{1}{2}(x_0 + x_1)$, where

$(x - x_0)^2(x - x_1)^2 \quad = (\frac{1}{2}(x_0 + x_1) - x_0)^2(\frac{1}{2}(x_0 + x_1) - x_1)^2$

$= (\frac{1}{2}(x_1 - x_0))^2(\frac{1}{2}(x_0 - x_1))^2 \quad = \frac{1}{16}(x_1 - x_0)^4$

Now our error bound is

$$|f(x) - H_3(x)| \leq \frac{(x_1 - x_0)^4}{16(4!)}[\max_{[x_0,x_1]} |f^{(4)}(x)|]$$

or $|f(x) - H_3(x)| \leq \dfrac{(x_1 - x_0)^4}{384}[\max_{[x_0,x_1]} |f^{(4)}(x)|]$.

That is the error bound for one Hermite cubic polynomial. Now we must look at a piecewise cubic Hermite polynomial.

On the j^{th} interval, using the above we have

$$|f(x) - S_j(x)| \leq \frac{h_j^4}{384}[\max_{[x_j,x_{j+1}]} |f^{(4)}(x)|].$$

Then, $\dfrac{h_j^4}{384}[\max_{[x_j,x_{j+1}]} |f^{(4)}(x)|] \quad \leq \quad \dfrac{h^4}{384}[\max_{[a,b]} |f^{(4)}(x)|]$

And we have $|f(x) - S(x)| \leq \dfrac{h^4}{384} \max_{[a,b]} \left|f^{(4)}(\xi)\right|$

□

(b) (T) On the interval $[x_i, x_{i+1}]$ let S_i be represented as

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3.$$

Derive expressions for a_i, b_i, c_i, and d_i in terms of $h_i = x_{i+1} - x_i$, $f_i = f(x_i)$, $f'_i = f'(x_i)$, $f_{i+1} = f(x_{i+1})$, and $f'_{i+1} = f'(x_{i+1})$.

(1) $S_i(x_i) = a_i = f_i$

(2) $S'_i(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$

$\implies S'_i(x_i) = b_i = f'_i$

(3) $S_i(x_{i+1}) = a_i + b_ih_i + c_ih_i^2 + d_ih_i^3 = f_{i+1}$

(4) $S'_i(x_{i+1}) = b_i + 2c_ih_i + 3d_ih_i^2 = f'_{i+1}$

- Using (1), (2), and (3), we get

  $f_i + h_if'_i + c_ih_i^2 + d_ih_i^3 = f_{i+1}$

  $\implies c_i = \frac{1}{h_i^2}(f_{i+1} - f_i) - \frac{1}{h_i}f'_i - h_id_i$

- Then (4) becomes $f'_i + \frac{2}{h_i}(f_{i+1} - f_i) - 2f'_i - 2h_i^2d_i + 3h_i^2d_i = f'_{i+1}$

  $\implies d_i = \frac{1}{h_i^2}(f'_{i+1} + f'_i) - \frac{2}{h_i^3}(f_{i+1} - f_i)$

- And $c_i = \frac{1}{h_i^2}(f_{i+1} - f_i) - \frac{1}{h_i}f'_i - \frac{1}{h_i}(f'_{i+1} + f'_i) + \frac{2}{h_i^2}(f_{i+1} - f_i)$

  $\implies c_i = \frac{3}{h_i^2}(f_{i+1} - f_i) - \frac{1}{h_i}(f'_{i+1} + 2f'_i)$

Our coefficients are therefore :

- $a_i = f_i$
- $b_i = f'_i$
- $c_i = \frac{3}{h_i^2}(f_{i+1} - f_i) - \frac{1}{h_i}(f'_{i+1} + 2f'_i)$
- $d_i = \frac{1}{h_i^2}(f'_{i+1} + f'_i) - \frac{2}{h_i^3}(f_{i+1} - f_i)$

(c) Evaluating S requires the location of the nodes (x_j's) and the coefficients computed in the previous part. For easy evaluation, the coefficients can be stored in an n by 4 array. For example, let P be an n by 4 array with elements defined as $P_{i,1} = a_i$, $P_{i,2} = b_i$,$P_{i,3} = c_i$, and $P_{i,4} = d_i$ for $i = 0, 1, ..., n-1$.

Write a routine which takes as input the three arrays corresponding to the location of the nodes, the function values at these nodes, and the derivative values at the nodes, and it returns the array of coefficients, `P`.

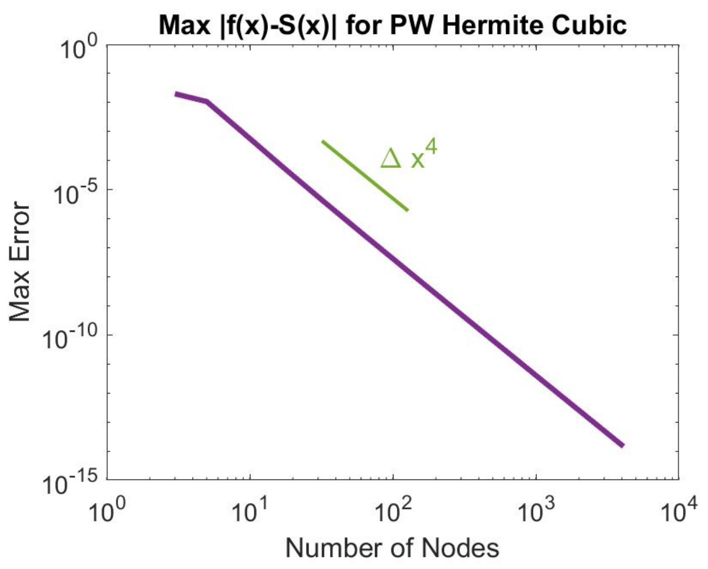
Use your routine to compute the coefficients defining S for the function $f(x) = \cos(2\pi x)$ at the nodes $x_i = i/5$ for $i = 0, \dots, 5$. Display the resulting coefficients in a table.

i	a_i	b_i	c_i	d_i
0	1	0	-21.9454037744634	23.3541431691854
1	0.3090169943749	-5.9756643294831	-5.6300875565064	23.3541431691854
2	-0.8090169943749	-3.6931636609809	18.4658183049046	$-5.5511151231258(10)^{-15}$
3	-0.8090169943749	3.6931636609809	17.0425908990174	-37.7877974258729
4	0.3090169943749	5.9756643294831	-7.9329178729521	-23.3541431691854

(d) Write a routine which takes as input the location of the nodes, the array of coefficients, and an evaluation point, x, and returns the value of $S(x)$.

Let $f(x) = \cos(2\pi x)$ and let $x_i = i/n$ for $i = 0, 1, \dots, n$ be the $n+1$ equally spaced points on $[0, 1]$. For a range of n values use your code to estimate $\max_{x\in[0,1]} |f(x) - S(x)|$ by evaluating f and S on a large set of values in $[0, 1]$. Display your results in a table and/or graph (use log-log plot). Discuss how your results demonstrate $\mathcal{O}(h^4)$ convergence.

The plot and table that follow show the estimates for the max-norm of the error function, $e_N(x) = |f(x) - S(x)|$. Let $E_N = \max_{x\in[0,1]} |e_n(x)|$ be the max norm of our error function. Then $\mathcal{O}(h^4)$ convergence would mean that $E_N \approx Ch^4$. Then, since we are using equally spaced points, $E_N/E_{2N} \approx \frac{h^4}{(h/2)^4} = 16$. Therefore, the ratios of the max norms should be about 16. We see that in the table that follows. Additionally, on a log-log plot, $E_N \approx Ch^4$ becomes $\log E_N \approx C' + 4\log h = C' - 4\log N$, which gives us a line of slope-4. This is again what we see in the plot.



$N+1$	Max Error Estimate	Ratio of Max Errors
3	0.020017013412594	
5	0.010790682002429	1.855027644044044
9	0.000906216248087	11.907402924199525
17	0.000060585551028	14.957629875683738
33	0.000003849578173	15.738231125494037
65	0.000000241587869	15.934484548057796
129	0.000000015114690	15.983646590357758
257	0.000000000944907	15.995952986305786
513	0.000000000059059	15.999291298765506
1025	0.000000000003691	15.999729307025987
2049	0.000000000000231	16.000000000000000
4097	0.000000000000015	15.862595419847329

5. (33 points) At the end of this assignment, code is given that takes as input vectors x and y, and returns an array, P, which contains the coefficients of the natural cubic spline through the points (x_i, y_i). A new one is also included in the homework Matlab code folder. The cubic spline is defined as $S(x) = S_i(x)$ for $x_i \leq x \leq x_{i+1}$, and

$$S_i(x) = P_{i1} + P_{i2}(x - x_i) + P_{i3}(x - x_i)^2 + P_{i4}(x - x_i)^3.$$

Note that you can used your code Problem 4c to evaluate the spline.

(a) (T) In the absence of derivatives at the endpoints, one commonly uses the *not-a-knot* boundary condition rather than natural boundary conditions. If the points are indexed from $j = 0 \dots n$, then the not-a-knot condition requires that $S'''(x)$ be continuous at x_1 and x_{n-1}. For $S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$, in class we derived the system of linear equations for the values of c_i:

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$$

for $i = 1 \dots n-1$, where $h_i = x_{i+1} - x_i$. We need two more equations to be able to solve for the c values. Derive the two equations that come from the not-a-knot conditions involving only c values and h values.

Our continuity conditions are: $S_0'''(x_1) = S_1'''(x_1)$ and $S_{n-2}'''(x_{n-1}) = S_{n-1}(x_{n-1})$

Since $S_j'''(x) = 6d_j$, this says:

$d_0 = d_1$ and $d_{n-2} = d_{n-1}$

Then, using our equations for d_j from class, this tells us:

$$\frac{c_1 - c_0}{3h_0} = \frac{c_2 - c_1}{3h_1}$$

$$\implies \underline{-h_1 c_0 + (h_1 + h_0)c_1 - h_0 c_2 = 0}$$

And $\dfrac{c_{n-1} - c_{n-2}}{3h_{n-2}} = \dfrac{c_n - c_{n-1}}{3h_{n-1}}$

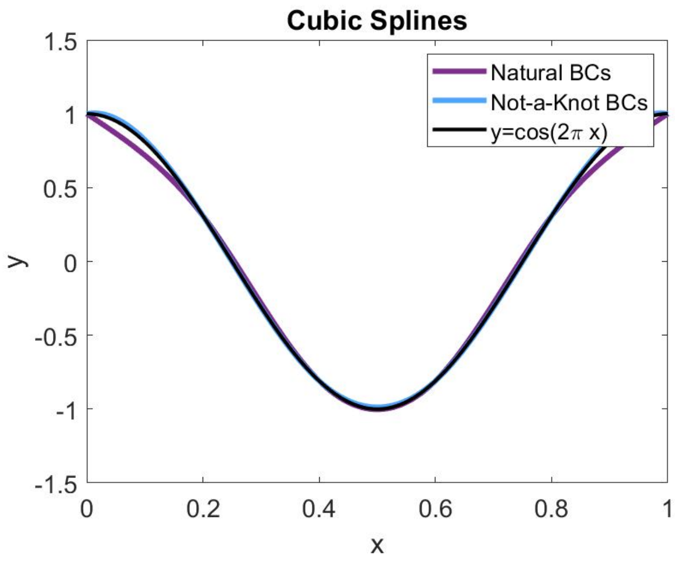
$$\implies \underline{-h_{n-1}c_{n-2} + (h_{n-1} + h_{n-2})c_{n-1} - h_{n-2}c_n = 0}$$

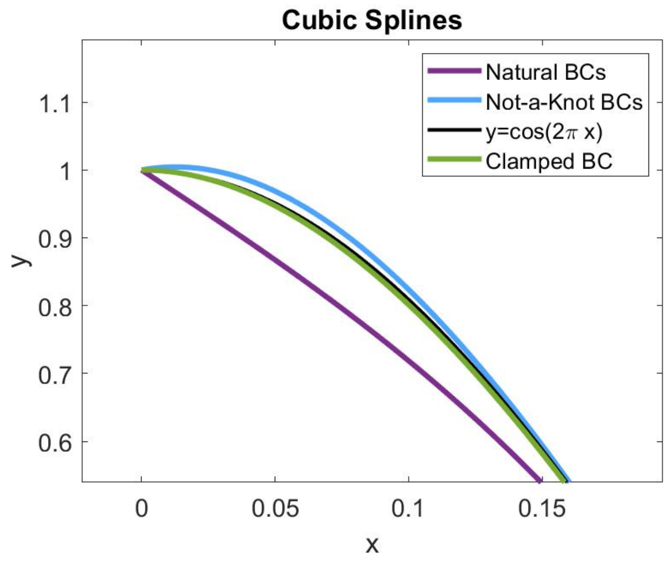
(b) Write a routine to compute the coefficients of a cubic spline with not-a-knot boundary conditions.

See not-a-knot-spline.m. Note: it requires using the full matrix from class, not the smaller one used in the original natural spline code. The new natural spline code uploaded is much more easily generalized. You just add the equations from part (a) to the top and bottom rows of the matrix.

(c) Plot the natural and not-a-knot splines that interpolate $f(x) = \cos(2\pi x)$ at the points $x_j = j/5$ for $j = 0, \dots, 5$. Which spline looks more like the cosine function?

Below is the plot of the Natural and Not-A Knot Cubic Splines. The Not-A-Knot Boundary conditions look better, managing to capture the concavity of the curve (whereas natural boundary conditions force there to be no concavity at the end points). I have also included a close up of a clamped BC cubic spline to demonstrate that it does an even better job of approximating the curve.





(d) For each spline, estimate the maximum error in using the spline to approximate $f(x)$ between $x = 0$ and $x = 1$. Do this by using your code to evaluate f and each spline at a large number of points on $[0, 1]$ compute the maximum absolute value of of the difference on these evaluation

points.

For the natural boundary conditions, I got a maximum error of ≈ 0.09517 and for the Not-a-knot BC's, I got a maximum error of ≈ 0.018098.

6. (a) (30 points) Use $N = 11$ equally spaced points between $x = -1$ and $x = 1$, and plot the

$N-1$-degree polynomial that interpolates

$$g(x) = \frac{1}{1+25x^2}$$

at these points for $-1 \le x \le 1$. Repeat for $N = 21$. What is the maximum error in using these interpolating polynomials to approximate $g(x)$.

I have summarized my results in a table in part (e)

(b) Repeat part (a), but use a piecewise Hermite cubic polynomial.

(c) Repeat part (a), but use a not-a-knot spline.

(d) Repeat part (a) using the points

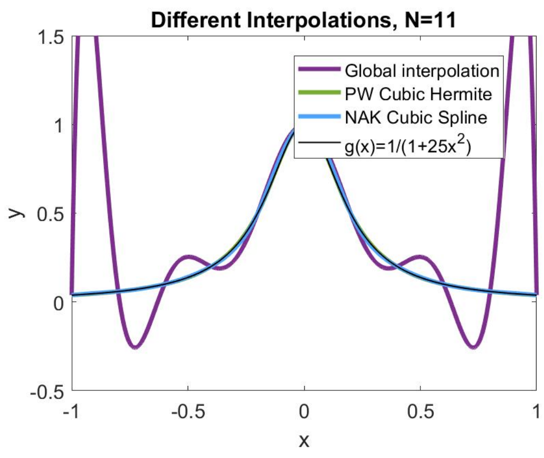
$$x_j = \cos\left(\frac{j}{n}\pi\right), \quad j = 0 \ldots n,$$

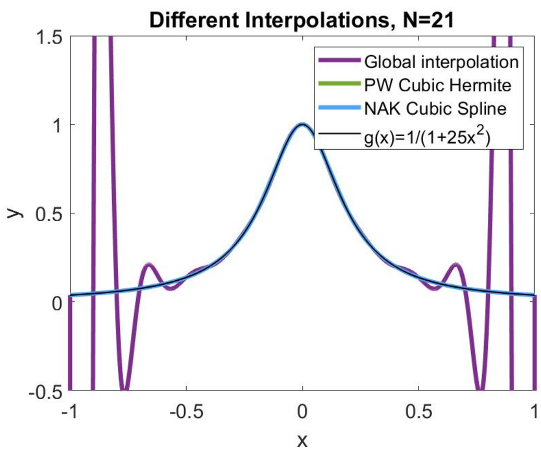
instead of equally spaced points.

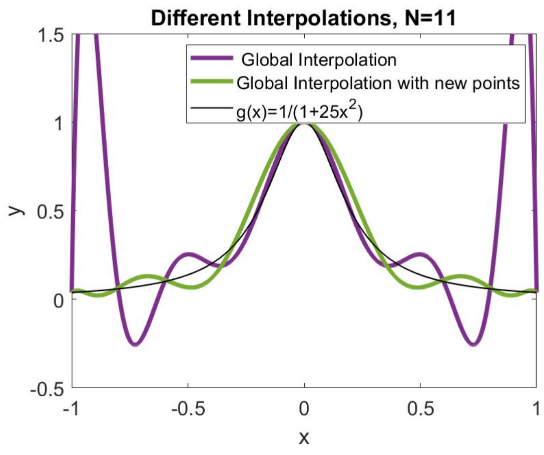
(e) Comment on your results.

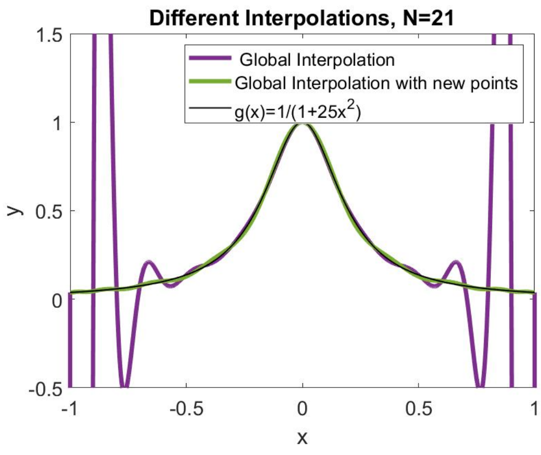
Below gives the table with the estimates for the max norms for parts (a)-(d), as well as some plots to illustrate these interpolants. First, we notice that the global interpolation shows Runge's phenomenon, giving us large oscillations near the boundaries of our interval. The piecewise interpolations, on the other hand, do a very good job of approximating this function. As we double the number of points, we see the oscillations in the global interpolation get worse, and hence our max error gets much worse, but our piecewise interpolants decrease in error. Therefore, we see that one way to avoid Runge's phenomenon is to do piecewise interpolations in order to utilize low degree polynomials. Part (d) gives us another option for reducing Runge's phenomenon - using unequally spaced points. These points are similar to the Chebyshev points, and we see that when we double our points, our error for this global interpolant does decrease. However, it still has higher error than the two piecewise interpolants. Using these points doesn't allow us to avoid the oscillations altogether, as the piecewise interpolants do, but the oscillations are definitely reduced.

Method	Max Norm Est. for N=11	Max Norm Est. for N=21
Global Interpolation	1.9157	59.822
PW Hermite Cubic	0.01294	0.001252
Not-A-Knot Cubic Spline	0.02197	0.003183
Global Interpolation with other points	0.1322	0.01774



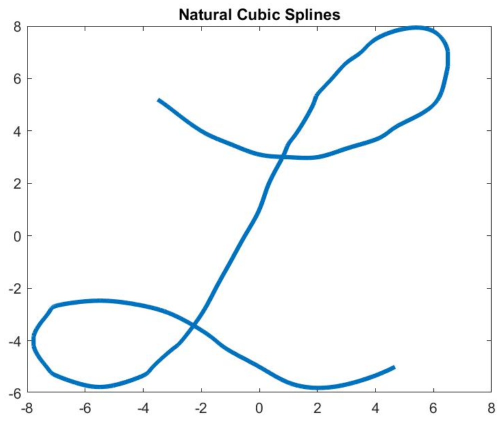




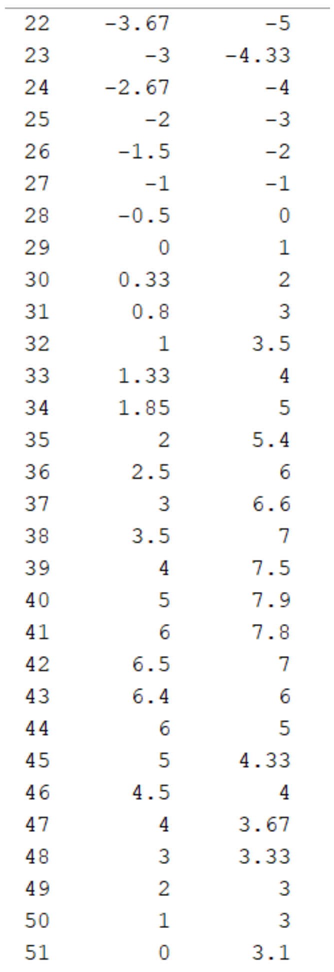


7. (33 points) Use cubic splines to represent a curve $(x(s), y(s))$ that is in the shape of the script letter L below. List the points you used to make the splines, and make a plot of your curve. You may use the provided grid to estimate the nodes by hand, or use software to help identify points on the curve. For example, in MATLAB the commands `imread,` `image`, and `ginput` could be used to aid in selecting your points.

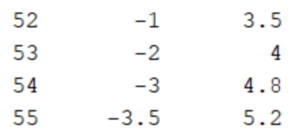
Using a crudely chosen 55 points, listed below, these are my resulting plots, using Natural Cubic Splines.



t	x	y
1	4.67	-5
2	4	-5.33
3	3	-5.67
4	2	-5.8
5	1	-5.6
6	0	-5
7	-1	-4.4
8	-1.5	-4
9	-2	-3.6
10	-3	-3
11	-4	-2.67
12	-5	-2.5
13	-6	-2.5
14	-7	-2.67
15	-7.3	-3
16	-7.8	-4
17	-7.33	-5
18	-7	-5.33
19	-6	-5.7
20	-5	-5.7
21	-4	-5.33

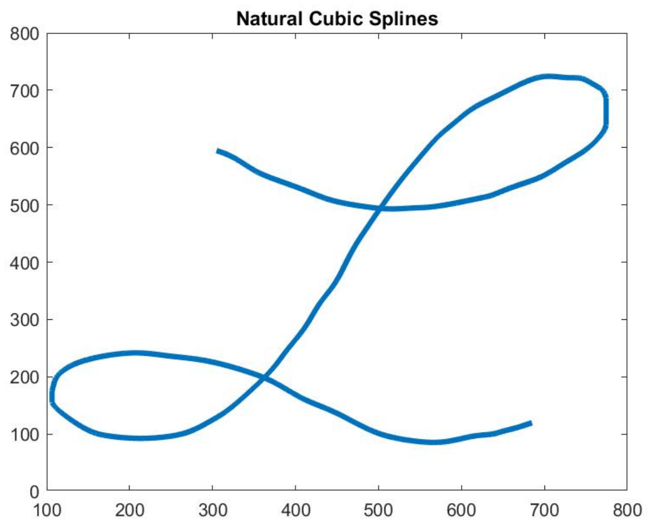


22	-3.67	-5
23	-3	-4.33
24	-2.67	-4
25	-2	-3
26	-1.5	-2
27	-1	-1
28	-0.5	0
29	0	1
30	0.33	2
31	0.8	3
32	1	3.5
33	1.33	4
34	1.85	5
35	2	5.4
36	2.5	6
37	3	6.6
38	3.5	7
39	4	7.5
40	5	7.9
41	6	7.8
42	6.5	7
43	6.4	6
44	6	5
45	5	4.33
46	4.5	4
47	4	3.67
48	3	3.33
49	2	3
50	1	3
51	0	3.1



52	-1	3.5
53	-2	4
54	-3	4.8
55	-3.5	5.2

Instead using 70 points chosen on Matlab, I got the following plot with Natural Cubic Splines.



t	x	y
1	304.81	594.76
2	326.02	582.02
3	350.49	560.18
4	376.6	543.8
5	410.87	525.6
6	445.13	507.4
7	485.92	496.48
8	515.29	492.84
9	543.03	494.66
10	574.03	498.3
11	619.72	511.04
12	639.3	518.32
13	665.41	532.88
14	696.41	549.26
15	724.15	572.92
16	750.26	596.58
17	768.2	622.06
18	774.73	642.08
19	774.73	665.74
20	774.73	683.95
21	769.84	700.33
22	760.05	709.43
23	745.36	720.35
24	725.78	722.17
25	699.67	723.99
26	671.93	711.25
27	644.2	691.23

28	614.83	669.38
29	591.98	643.9
30	574.03	622.06
31	557.72	596.58
32	536.51	560.18
33	516.93	523.78
34	498.98	487.38
35	482.66	452.79
36	466.34	414.57
37	448.4	365.43
38	428.81	327.21
39	410.87	285.34
40	388.02	243.48
41	366.81	205.26
42	327.65	152.48
43	304.81	128.82
44	281.96	110.62
45	255.86	97.875
46	215.07	92.415
47	177.54	96.055
48	149.8	106.98
49	120.43	134.28
50	105.74	157.94
51	107.37	179.78
52	112.27	199.8
53	130.22	219.82
54	162.85	234.38
55	206.91	241.66
56	247.7	236.2
57	295.02	227.1
58	345.6	207.08
59	378.23	187.06
60	409.23	161.58
61	441.87	141.56
62	474.5	117.9
63	507.14	97.875
64	547.93	86.954
65	582.19	86.954
66	613.19	96.055
67	636.04	99.695
68	650.72	105.16
69	665.41	110.62
70	684.99	119.72