Math 125 **Problem Set 3** **Due: Tuesday 10/25 at 11:59 p.m. EST**

Instruction: Read the assignment policy. For problems 2 and 3, include a printout your code with your homework submission. You should submit your assignment on Gradescope.

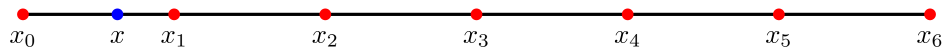
1. Consider a uniform distribution of $n+1$ nodes for interpolation. In particular, given x_0 and $h > 0$, $x_i = x_{i-1} + h$ for $i = 1, ..., n$.

(a) Prove the following inequality

$$\left| \prod_{i=0}^{n} (x - x_i) \right| \leq \frac{n! h^{n+1}}{4},$$

where $x \in (x_0, x_n)$.

[**Hint**: Without loss of generality, you can assume that $x \in (x_0, x_1)$. An example of such a grid with $n = 6$ is shown below].



(b) Define the interval I as $I = (x_0, x_n)$. Let f be continuously differentiable up to $n+1$ times and P_n be the Lagrange interpolant polynomial of order n using the data points $(x_0, f(x_0)), (x_1, f(x_2)), ..., (x_n, f(x_n))$. Show that

$$E(x) = |P_n(x) - f(x)| \leq \max_{x \in I} \frac{|f^{n+1}(x)|}{4(n+1)} h^{n+1},$$

where x is any point in I.

(c) Is it always true that $E(x) \to 0$ as $n \to \infty$?

Solution:

(a) We first consider the simple case where $x \in (x_0, x_1)$. We note the following inequalities

$$|x - x_2| \leq 2h$$
$$|x - x_3| \leq 3h$$
$$\vdots$$
$$|x - x_n| \leq nh$$

Next, we consider the maximum of $(x - x_0)(x - x - 1)$. Let $a = x - x_0$. Then, $x - x_1 = h - a$. With that, $f(a) = (x - x_0)(x - x - 1) = a(h - a) = -a^2 + ha$. This is maximum when $f'(a)$ vanishes which happens at $a = \frac{h}{2}$. Therefore, the maximum value of $(x - x_0)(x - x - 1)$ is $\frac{h^2}{4}$. Combining all the results we have

$$\left| \prod_{i=0}^{n} (x - x_i) \right| = \prod_{i=0}^{n} \left| (x - x_i) \right| = |x - x_0| \cdot |x - x_1| \cdot \left| \prod_{i=0}^{n} \left| (x - x_i) \right| \leq \frac{h^2}{4} (2h \cdot 3h \ldots \cdot nh) = \frac{n! h^{n+1}}{4}$$

General case: We consider the case where x could lie in any of the intervals. Let's assume that $x \in (x_i, x_{i+1})$ where $i \in \{2, .., n-1\}$. Arguing similarly to the simple case,

$$|x - x_i||x - x_{i+1}| \leq \frac{h^2}{4}.$$

Next, we note the following inequality

$$|x - x_j| = \begin{cases} (i - j + 1)h & \text{if } j < i \\ (j - i)h & \text{if } j > i + 1 \end{cases}$$

Using this, we obtain

$$\left|\prod_{i=0}^{n}(x - x_i)\right| \leq \frac{h^2}{4} \cdot \left(\prod_{k=2}^{i+1} kh\right) \cdot \left(\prod_{k=2}^{n-i} kh\right) = \frac{h^2}{4}(i+1)!h^i \cdot (n-i)!h^{n-i-1} = \frac{1}{4}(i+1)!(n-i)!h^{n+1}.$$

It now suffices to consider a bound of $(i+1)!(n-i)!$. We note that

$$(i+1)!(n-i)! < \begin{cases} \left((i+1)! \prod_{k=i+2}^{n} k!\right) = n! & \text{if } i + 1 > n - i \\ \left((n-i)! \prod_{k=n-i+1}^{n} k!\right) = n! & \text{if } i + 1 < n - i \end{cases}$$

Combining all the results we have

$$\left|\prod_{i=0}^{n}(x - x_i)\right| < \frac{n!h^{n+1}}{4}.$$

(b) Recalling the Lagrange interpolation error formula, we obtain

$$E(x) = |P_n(x) - f(x)| \leq \max_{x \in I} \frac{|f^{n+1}(x)|}{(n+1)!}h^{n+1} \cdot \left|\prod_{i=0}^{n}(x - x_i)\right|$$

Using the result in (a), the bound simplifies to

$$E(x) \leq \max_{x \in I} \frac{|f^{n+1}(x)|}{(n+1)!}h^{n+1} \cdot \frac{n!h^{n+1}}{4} = \max_{x \in I} \frac{|f^{n+1}(x)|}{4(n+1)}h^{n+1}.$$

(c) No. This will depend on the the growth of $f^{n+1}(x)$ as $n \to \infty$. For an example, see solution of problem 2.

2. We explore the Runge phenomenon by considering the function $f(x) = \dfrac{1}{1 + 25x^2}$. We approximate $f(x)$ using $n + 1$ equally spaced points $x_0, x_1, ..., x_n$ with $x_i = (2i/n) - 1$.

(a) Plot the Lagrange interpolant of degree n for $n = 4, 8, 16, 32$.

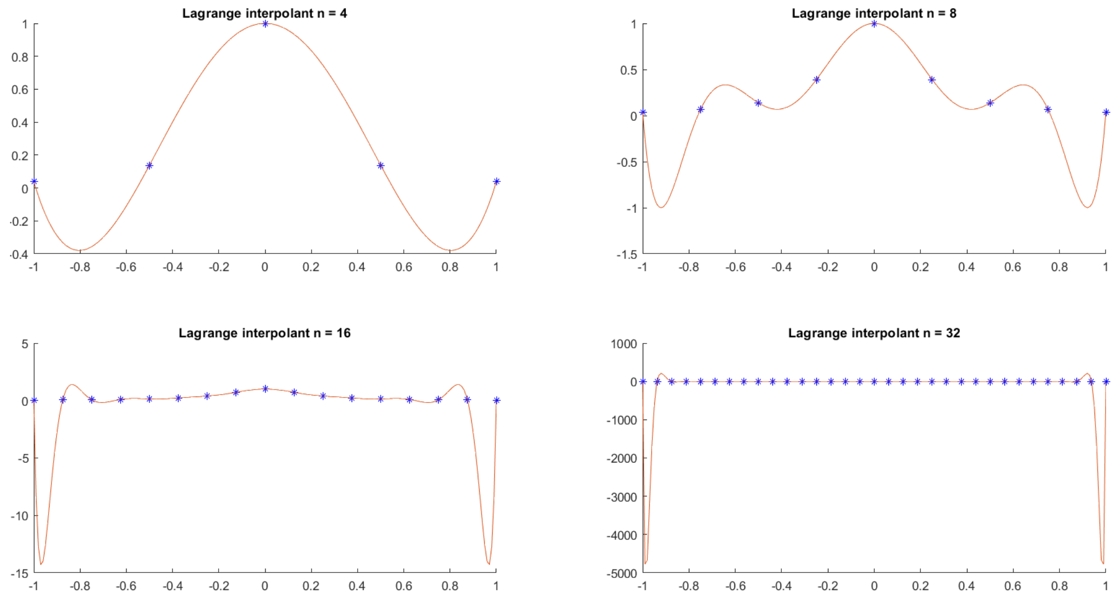
(b) For each interpolant in (a), plot the error $E(x) = f(x) - P_n(x)$.

(c) Does increasing n improve the quality of the approximation? Discuss the implication of this result.

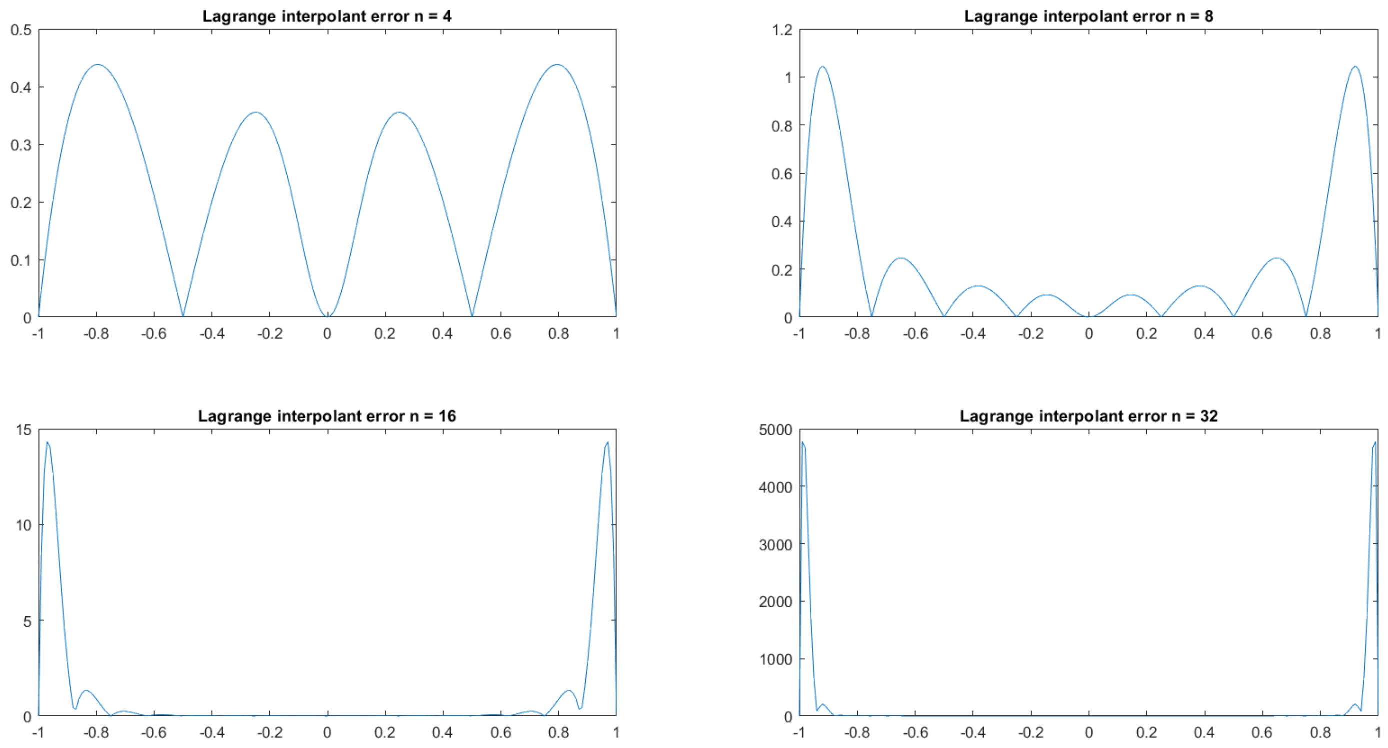
(d) Repeat the exercises in (a)-(c) by using Chebyshev interpolation.

(e) If we do not have the freedom to choose the nodes, what alternatives ways are to fix the problem observed in (a)-(c)?

Solution: You can find the codes that reproduce all the results for this problem in the `HW3_Soln` folder. The codes are `Lagrange_interpolant`, `runge_test`, `runge_test_chebyshev`, `error-lagrange` and `error_chebyshev`.



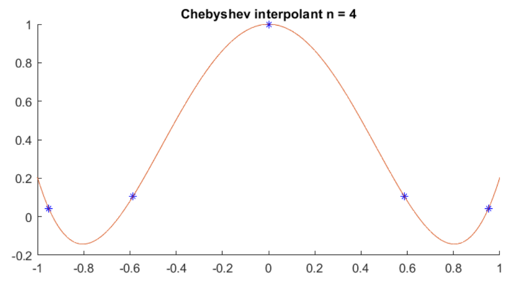
(a)

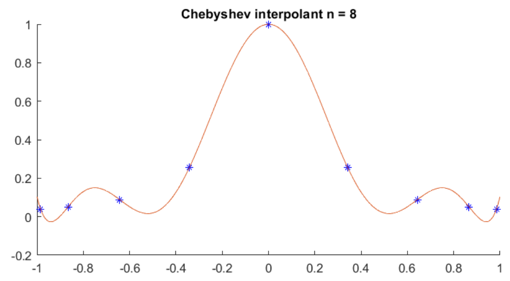


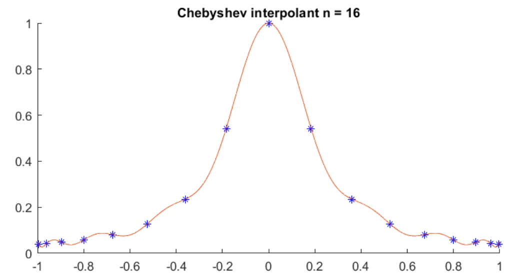
(b)

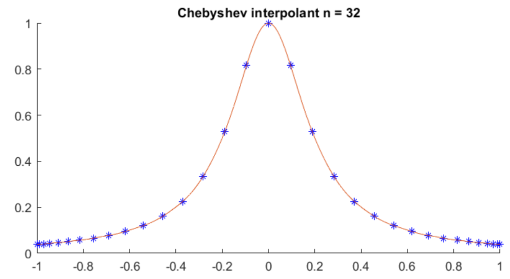
(c) No. In fact, as n increases, we see oscillations that get larger at the end of the intervals. This indicates that using higher degree polynomials does not improve the approximation error.

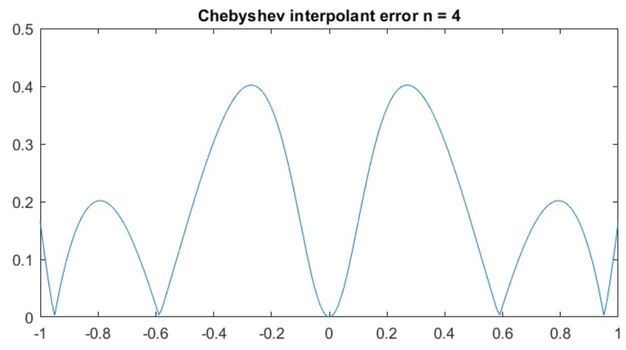
(d) Figure is shown below. We see that in contrast to using uniform nodes, the Lagrange interpolation based on the Chebyshev nodes is not susceptible to oscillations at the end. In addition, as n increases, we see that the approximation error decreases. This shows that Chebyshev nodes help us avoid the Runge phenomenon.

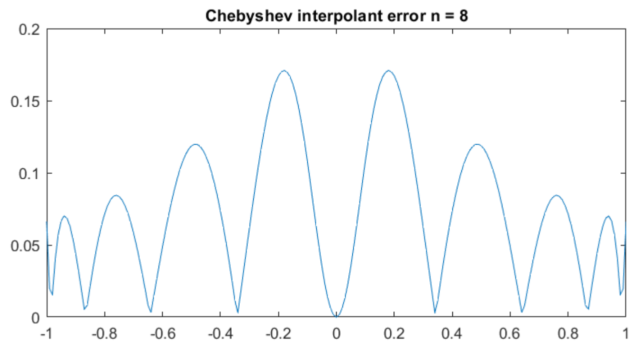


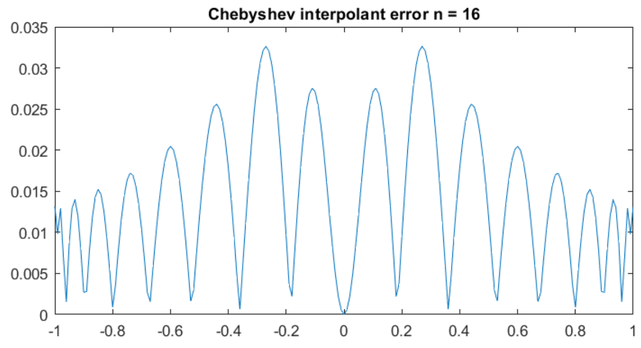


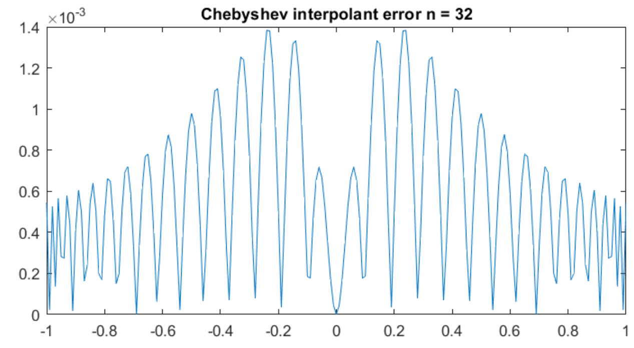












(e) One solution is to use splines e.g. cubic splines.

3. Assume that you have collected data $(t_1, y_1), (t_2, y_2), ..., t_{n+1}, y_n)$ and want to interpolate the data using cubic splines. Let p_i denote the cubic polynomial between t_i and t_{i+1}.

$$p_i(t) = a_i + b_i \frac{t - t_i}{t_{i+1} - t_i} + c_i \frac{(t - t_i)^2}{(t_{i+1} - t_i)^2} + d_i \frac{(t - t_i)^3}{(t_{i+1} - t_i)^3} \qquad t \in [t_i, t_{i+1}]$$

(a) Use a change of variable and argue that $p_i(u) = a_i + b_i u + c_i u^2 + d_i u^3$ for $u \in [0, 1]$.

(b) **Extra credit**: Let D_i, $1 \leq i \leq n+1$, denote the value of the first derivative of p_i at the nodes. In lecture, we discussed that the coefficients a_i, b_i, c_i, d_i can be determined from $\{y_i\}_{i=1}^{n+1}$ and $\{D_i\}_{i=1}^{n+1}$. In particular, we have

$$\begin{aligned} a_i &= y_i \\ b_i &= D_i \\ c_i &= 3(y_{i+1} - y_i) - 2D_i - D_{i+1} \\ d_i &= 2(y_i - y_{i+1}) + D_i + D_{i+1} \end{aligned}$$

Following the derivation in class and using natural boundary conditions, prove that the $\{D_i\}_{i=1}^{n+1}$ can be obtained from solving the following linear system

$$\begin{pmatrix} 2 & 1 & & & & 0 \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ 0 & & & & 1 & 2 \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_n \\ D_{n+1} \end{pmatrix} = \begin{pmatrix} 3(y_2 - y_1) \\ 3(y_3 - y_1) \\ 3(y_4 - y_2) \\ \vdots \\ 3(y_{n+1} - y_{n-1}) \\ 3(y_{n+1} - y_n) \end{pmatrix}$$

(c) Prove that the matrix in (c) is invertible.

(d) Download the dataset `spline_data.mat` from HW3 folder. If you use Python for programming, download `spline_data.csv`. Use the linear system in (c) to find the cubic spline that interpolates the data points. Plot the cubic spline. When you plot the cubic between two knots t_i and t_{i+1}, use a uniform grid with spacing 10^{-4}.

[**Remark**: To solve the linear system, you can use `np.linalg.solve` in Python or the command `A\z` (to solve $\mathbf{Ax} = \mathbf{z}$) in MATLAB.]

Solution:

(a) Let $u = \dfrac{t - t_i}{t_{i+1} - t_i}$. When $t = t_i$, $u = 0$ and when $t = t_{i+1}$, $u = 1$. Therefore, $p_i(u)$ can be written as

$$p_i(u) = a_i + b_i u + c_i u^2 + d_i u^3 \text{ for } u \in [0, 1]$$

(b) We first note the following equations

$$\begin{aligned} a_i &= y_i && (1) \\ b_i &= D_i && (2) \\ c_i &= 3(y_{i+1} - y_i) - 2D_i - D_{i+1} && (3) \\ d_i &= 2(y_i - y_{i+1}) + D_i + D_{i+1} && (4) \end{aligned}$$

We consider (3) and the fact that the second derivatives match at the knots to obtain

$$c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1} \tag{5}$$

$$2c_{i-1} + 6d_{i-1} = 2c_i \text{ for } 2 \leq i \leq n \tag{6}$$

Using the above two equations, we get

$$2c_{i-1} + 6d_{i-1} = 2\Bigg[3(y_{i+1} - y_i) - 2D_i - D_{i+1}\Bigg]$$

We note that we know the form of c_{i-1} and d_{i-1} from (3) and (4). Plugging this in the LHS of the above equation leads to

$$2\Bigg[3(y_i - y_{i-1}) - 2D_{i-1} - D_i\Bigg] + 6\Bigg[2(y_{i-1} - y_i + D_{i-1} + D_i\Bigg] = 2\Bigg[3(y_{i+1} - y_i - 2D_i - D_{i+1}\Bigg]$$

Rearranging the above, we have

$$D_{i+1} + 4D_i + D_{i-1} = 3(y_{i+1} - y_{i-1}) \text{ for } 2 \leq i \leq n \tag{7}$$

We now consider the boundary conditions. Since $p_1''(0) = 0, p_n''(1) = 0$, we have $c_1 = 0$ and $2c_n + 6d_n = 0$. Using (3) and $c_1 = 0$, we have

$$2D_1 + D_2 = 3(y_2 - y_1) \tag{8}$$

Similarly, using $c_n = -3d_n$ and the fact that the second derivative must match at the knots

$$\begin{aligned} &a_n + b_n + c_n + d_n = y_n + 1 \\ &\to y_n + D_n - 2(y_n - y_{n+1}) + D_n + D_{n+1}) = y_{n+1} \end{aligned}$$

Re-arranging the last equation, we get

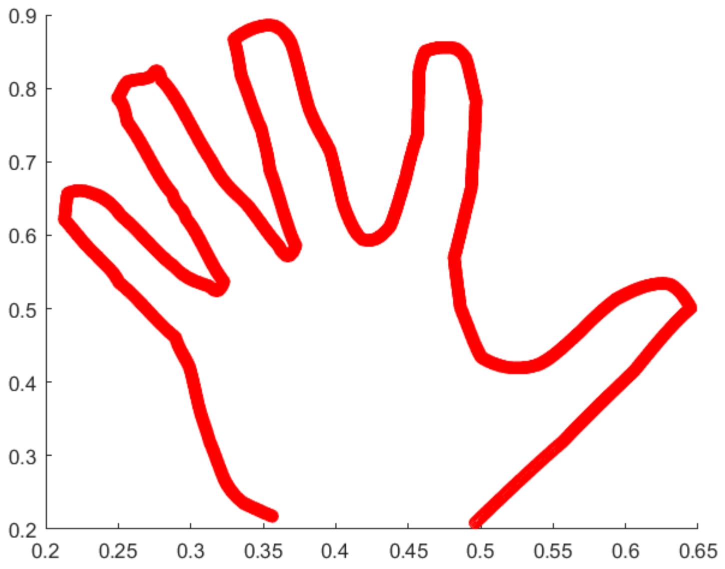
$$D_n + D_{n+1} = 3(y_{n+1} - y_n) \tag{9}$$

Using (7), (8) and (9), we can set up a linear system to solve for $\{D_i\}_{i=1}^{n+1}$.

$$\begin{pmatrix} 2 & 1 & & & & 0 \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ 0 & & & & 1 & 2 \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_n \\ D_{n+1} \end{pmatrix} = \begin{pmatrix} 3(y_2 - y_1) \\ 3(y_3 - y_1) \\ 3(y_4 - y_2) \\ \vdots \\ 3(y_{n+1} - y_{n-1}) \\ 3(y_{n+1} - y_n) \end{pmatrix}$$

(c) There are many ways to do this. A simple way is to use the Gershgorin circle theorem. We note that the minimum eigenvalue is at least 2 which is away from zero. Since the matrix does not contain a zero eigenvalue, it is invertible.

(d) The figure is shown below. For details of the implementation, refer to `spline_interpolation` script in the `HW3_Soln` folder.


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65