Math 125

1 a) WLOG, let $x \in (x_0, x_1)$, so $(x-x_0)(x-x_1)$ as $x_1 = x_0 + h$, we can find the maximum to be at $\frac{x_0+h}{2}$ which we can say is $h/2$.

- If $x \in (x_0, x_1)$ then we can say

$|x - x_2| \leq 2h$ note we can do this as
$|x - x_3| \leq 3h$ x must be in one of the open intervals and if not, then $x = x_0$ or x_1 ..... and $\prod(x - x_i) = 0$

$|x - x_n| \leq nh$

So $\prod(x-x_i) \leq \frac{h^2}{4} \cdot \frac{2h \cdot 3h \cdot 4h \cdot \ldots \cdot nh}{1}$

$\leq \frac{h^2}{4} \cdot n! \cdot h^{n-1} = \frac{n!\, h^{n+1}}{4}$

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

b) $E(x) = P_n(x) - f(x) = \frac{f^{n+1}(\xi_x)}{(n+1)!} \prod_{i=1}^{n} (x-x_i)$

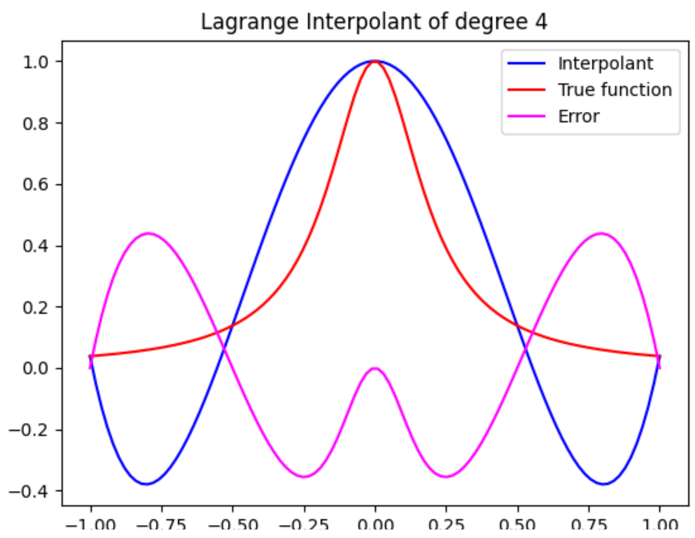
$|P_n(x) - f(x)| \leq \left| \max_{x \in (x_0, x_n)} \frac{f^{n+1}(x)}{(n+1)!} \cdot \frac{n!\, h^{n+1}}{4} \right|$

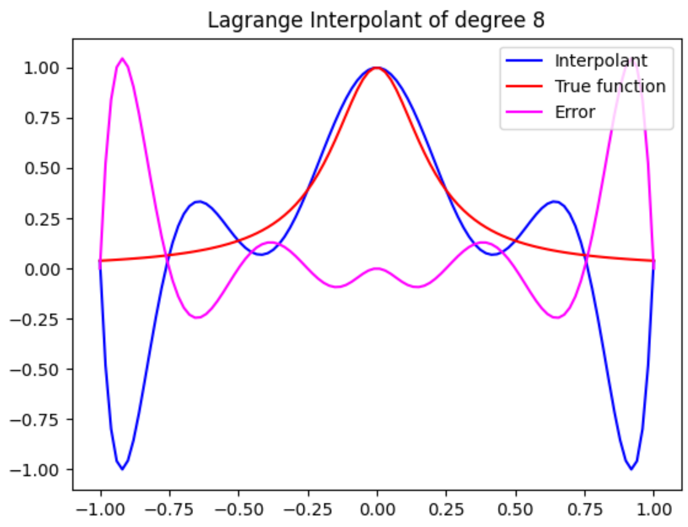
Let $(x_0, x_n) = I$

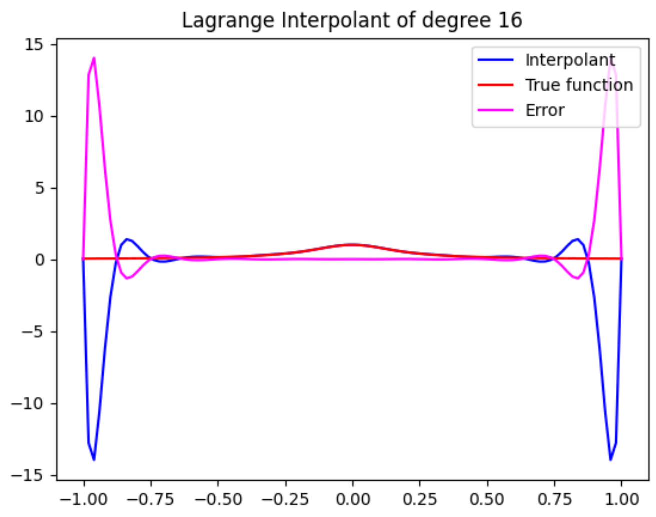
$|P_n(x) - f(x)| \leq \max_{x \in I} \frac{|f^{n+1}(x)|}{4(n+1)} \cdot h^{n+1}$ as $h \geq 0$

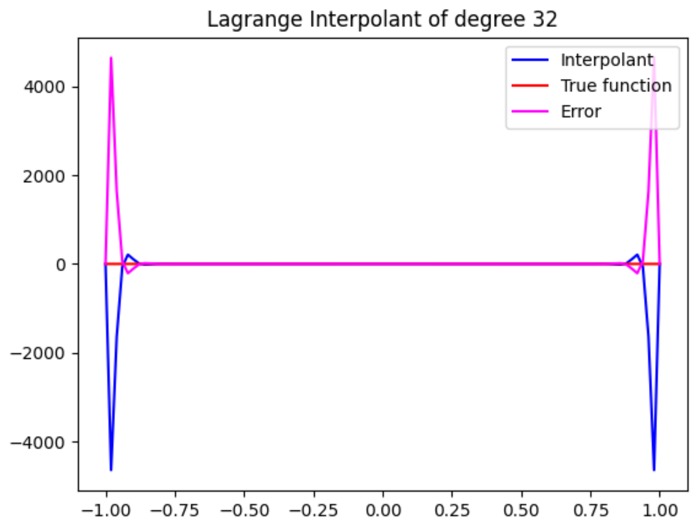
c) No, we cannot guarantee f is always differentiable, we can only guarantee f is differentiable n+1 times

2a and 2b)
Here are my graphs for 2a and 2b, code is at the end of my answers.



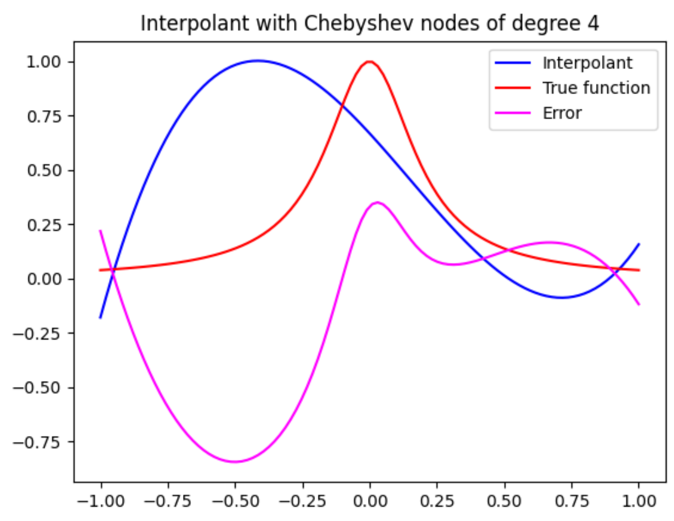


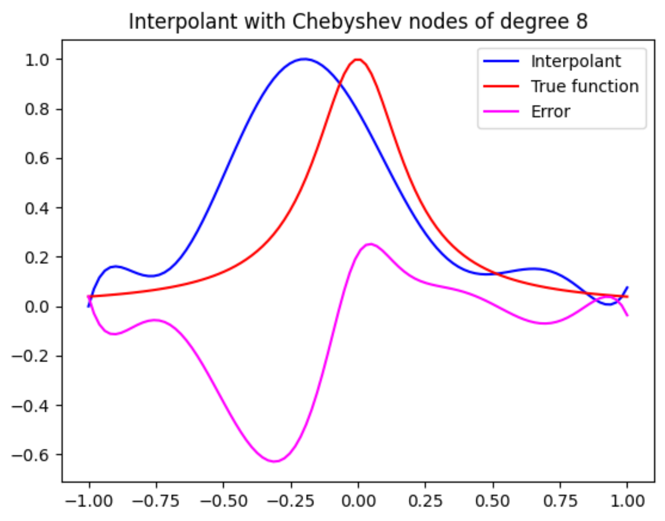


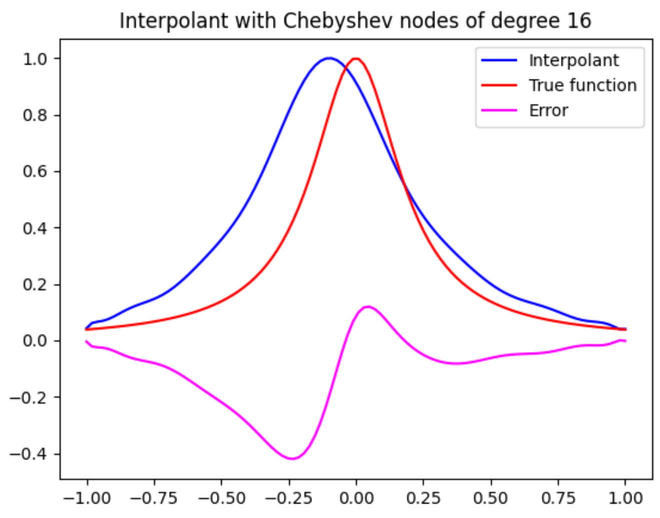


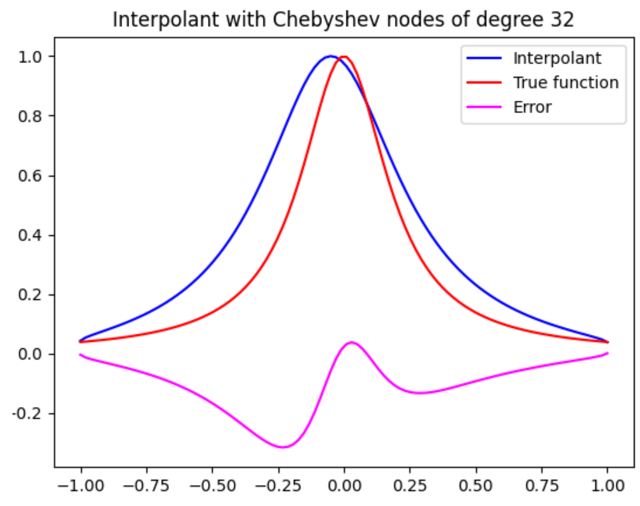
2c) We can see that as the number of nodes increases, generally, the approximation improves. This is hard to visually notice at degrees 16 and 32, as due to the Runge effect, at the end points, error greatly increases and distorts the graph.

2d)









As the number of nodes increases, we can see that the error appears to decrease and the interpolant gets closer to the true function. We also notice that at the endpoints, error is small showing that we've minimized the Runge effect.

2e)
If you don't have the freedom to pick nodes, then you could use spline interpolation instead.

3 a) Let $w = \frac{t - t_i}{t_{i+1} - t_i}$

$p_i(w) = a_i + b_i x w + c_i w^2 + d_i w^3$

Domain of w: $\left[\frac{t_i - t_i}{t_{i+1} - t_i}, \frac{t_{i+1} - t_i}{t_{i+1} - t_i}\right] = [0, 1]$

- As transformation is linear only need to check bounds $\square$

b) Didn't do extra credit. :(

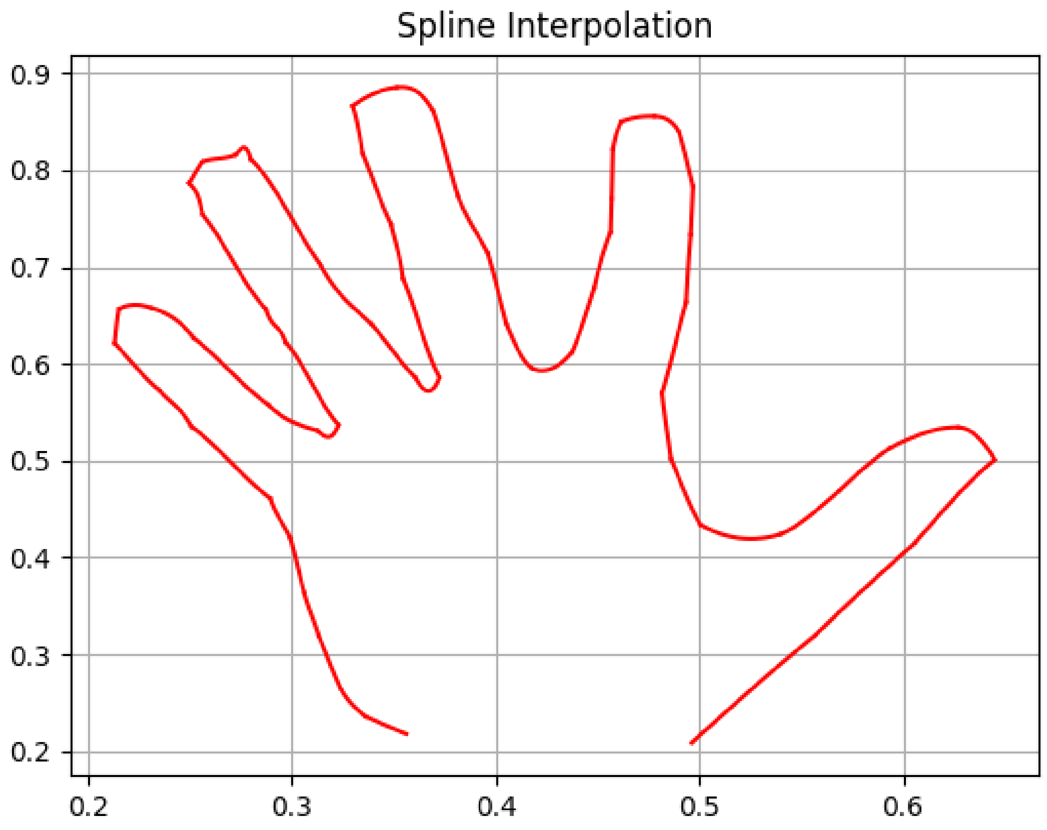
c) By the Circle theorem, for a matrix A,
$|\lambda - A_{ii}| \leq \sum_{j \neq i} |A_{ij}| = R \quad \forall i$

Let A be the matrix in part (b).
For the first and last row, A_{11} and $A_{ii} = 2$
$\sum_{i \neq j} |A_{ij}| = 1$, so by Circle theorem, $\lambda \in [1, 3]$
For all other rows, $A_{ii} = 4$ and $\sum_{i \neq j} |A_{ij}| = 2$ so by Circle theorem $\lambda \in [2, 6]$
All eigenvalues are nonzero, therefore, A is an invertible matrix.

d) Next page has code

3d)

Implementing spline interpolation on spline_data.csv yields the following picture (all code on next page):



Question 2 Lagrange Interpolation Code:

```
import numpy as np
import matplotlib.pyplot as plt

n = 8 #Degree of polynomial
f = lambda x: 1/(1+25*(x**2)) #True function

def cheby(n): #Calculates Chebyshev nodes for given degree
    x = np.zeros(n)
    for i in range(0, n):
        temp = (2*(i+1)-1)/(2*n)
        x[i] = np.cos(temp*np.pi)
    return x

#Does Lagrange interpolation
def interpolate(x, y, domain):
    prod = 1
    sum = np.zeros(len(domain))
    #Generates each component of sum individually, adding them
iteratively
    #Follows Lagrange interpolation formula
    for i in range(0, len(x)):
        for j in range(0, len(x)):
            if i != j:
                prod = (domain - x[j])/(x[i]-x[j])*prod
        prod = prod*y[i]
        sum = sum + prod
        prod = 1
    return sum

x = np.zeros(n+1)
y = np.zeros(n+1)
for i in range(0,n+1): #For non-Chebyshev nodes, makes domain and
range
    x[i] = (2*i)/n-1
    y[i] = f(x[i])

x_2 = cheby(n)
y_2 = f(x)
domain = np.linspace(-1, 1, 100) #Graphing domain
#res = interpolate(x, y, domain) This line is uniform nodes
res = interpolate(x_2, y_2, domain) #This line is Chebyshev nodes
#Comment out the line you don't want

#Graph display, just change name between different graph
```

```
plt.title('Interpolant with Chebyshev nodes of degree ' + str(n))
plt.plot(domain, res, color='blue',label='Interpolant')
plt.plot(domain, f(domain), color='red', label= 'True function')
plt.plot(domain, f(domain) - res, color = 'magenta', label ='Error')
plt.legend(loc=1)
plt.show()
```

Question 3 Spline Interpolation Code:

```
import numpy as np
import csv
import matplotlib.pyplot as plt

x = np.zeros(59)
y = np.zeros(59)
count = 0

#Reads in csv file and data
with open('spline_data.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        x[count] = float(row[0])
        y[count] = float(row[1])
        count += 1

#Creates triangle matrix
arr = np.zeros((59, 59))
for i in range(0, 59): #Build triangular matrix
    if i == 0:
        arr[0, 0] = 2
        arr[0, 1] = 1
    elif i == 58:
        arr[58, 58] = 2
        arr[58, 57] = 1
    else:
        arr[i, i - 1] = 1
        arr[i, i] = 4
        arr[i, i + 1] = 1

y_new = np.zeros(59) #Creates y-vector
y_new[0] = 3*(y[1]-y[0])
for i in range(1, 58):
    y_new[i] = 3*(y[i+1] - y[i-1])
y_new[58] = 3*(y[58] - y[57])
```

```
D = np.matmul(np.linalg.inv(arr), y_new) #Finds vector of derivatives

a = y #Gets coefficients for each spline
b = D
c = np.zeros(59)
d = np.zeros(59)
for i in range(0, 58):
    c[i] = 3*(y[i+1]-y[i]) - 2*D[i]-D[i+1]
    d[i] = 2*(y[i] - y[i+1]) + D[i] + D[i+1]

t = x
for i in range(0, 58): #Converts to correct interval then graphs
    t_bar = 0
    if  t[i] < t[i+1]:
        t_bar = np.linspace(t[i], t[i+1], 100)
    else:
        t_bar = np.linspace(t[i+1], t[i], 100)
    z = (t_bar - t[i])/(t[i+1]-t[i])
    plt.plot(t_bar, a[i]+b[i]*z+c[i]*(z**2)+d[i]*(z**3), color =
'red')

plt.grid() #Display stuff
plt.title('Spline interpolation')
plt.show()
```