Math104A Homework3

November 24, 2024

Brandon Su

1. (a) We know that P_1 is the linear interpolation of f, so we can write

$$f - P_1 = \frac{1}{2}f''(\xi)(x - x_0)(x - x_1)$$

for some $\xi \in [x_0, x_1]$. And the error is maximized when x is the midpoint of the interval $[x_0, x_1]$ Then

$$\begin{aligned}\|f - P_1\|_\infty &= \max_{x\in[x_0,x_1]} |f - P_1| \\ &= \left|\frac{1}{2}f''(\xi)\left(\frac{x_0 + x_1}{2} - x_0\right)\left(\frac{x_0 + x_1}{2} - x_1\right)\right| \\ &\leq \frac{1}{2}\left|f''(\xi)\right|\left(\frac{x_1 - x_0}{2}\right)^2\end{aligned}$$

Since $|f''^{(x)}| \leq M_2$ for all $x \in [x_0, x_1]$, so

$$\begin{aligned}\|f - P_1\|_\infty &= \max_{x\in[x_0,x_1]} |f - P_1| \\ &\leq \frac{1}{2}M_2\left(\frac{x_1 - x_0}{2}\right)^2 \\ &= \frac{1}{8}(x_1 - x_0)^2 M_2\end{aligned}$$

(b) The interpolation for $f(x) = \sin(x)$ is

$$P_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) = 0 + \frac{1 - 0}{\frac{\pi}{2} - 0}(x - 0) = \frac{2x}{\pi}$$

For $f(x) = \sin x$, then $f(0) = \sin 0 = 0$ and $f\left(\frac{\pi}{2}\right) = \sin\frac{\pi}{2} = 1$

Also, $f'(x) = \cos x$ and $f''(x) = -\sin x$, so $|f''(x)| \leq M_2 = 1$ for all $x \in [0, \frac{\pi}{2}]$, then

$$\|f - P_1\|_\infty = \max_{x\in[0,\frac{\pi}{2}]} |f - P_1| \leq \frac{1}{8}\left(\frac{\pi}{2} - 0\right)^2 M_2 = \frac{\pi^2}{32} \approx 0.308$$

The actual error at $x = \frac{\pi}{4}$ is

$$\left|f(\frac{\pi}{4}) - P_1(\frac{\pi}{4})\right| = \left|\frac{\sqrt{2}}{2} - \frac{1}{2}\right| \approx 0.207 < 0.308$$

The actual error is less than the maximum error bound.

2. (a) We know that

$$P_n(x) = \sum_{j=0}^{n} f(x_j) l_j(x)$$

where

$$l_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

and the leading coefficient of $l_j(x)$ is

$$\lambda_j = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{1}{x_j - x_k}$$

Hence, the leading coefficient of the Lagrange form is

$$\sum_{j=0}^{n} f(x_j) \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{1}{x_j - x_k} = \sum_{j=0}^{n} \frac{f(x_j)}{\prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)}$$

We know that the interpolation by Newton's form is

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0)(x - x_1) + \cdots + f[x_0, x_1, \ldots, x_n] \prod_{k=0}^{n-1} (x - x_k)$$

Therefore, the leading coefficient is

$$f[x_0, x_1, \ldots, x_n] = \sum_{j=0}^{n} \frac{f(x_j)}{\prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)}$$

(b) From part (a), we can conclude that $f[x_0, x_1, \ldots, x_n]$ is expressed as:

$$f[x_0, x_1, \ldots, x_n] = \sum_{j=0}^{n} \frac{f(x_j)}{\prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)}.$$

This only depends on the values $x_0, x_1, \ldots, x_n$ through symmetric products and sums. The denominator $\prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{1}{x_j - x_k}$ is invariant under permutations of $x_0, x_1, \ldots, x_n$, since it involves all the differences between the points. The numerator $f(x_j)$ is just the result of the function at x_j, and the summation includes all possible terms. Therefore, any permutation of $x_0, x_1, \ldots, x_n$ leaves the corresponding divided difference unchanged.

3. (a) Compute the coefficients

[1]:
```
import numpy as np

def newton_coeffs(xs, ys):
    n = len(xs)
    coeffs = np.copy(ys)
    for i in range(1, n):
```

```
        for j in range(n-1, i-1, -1):
            coeffs[j] = (coeffs[j] - coeffs[j-1]) / (xs[j] - xs[j-i])
    return coeffs

def newton_polynomials(coeffs, xs, x):
    n = len(xs)
    p = coeffs[-1]
    for k in range(n-2, -1, -1):
        p = coeffs[k] + (x - xs[k]) * p
    return p
```

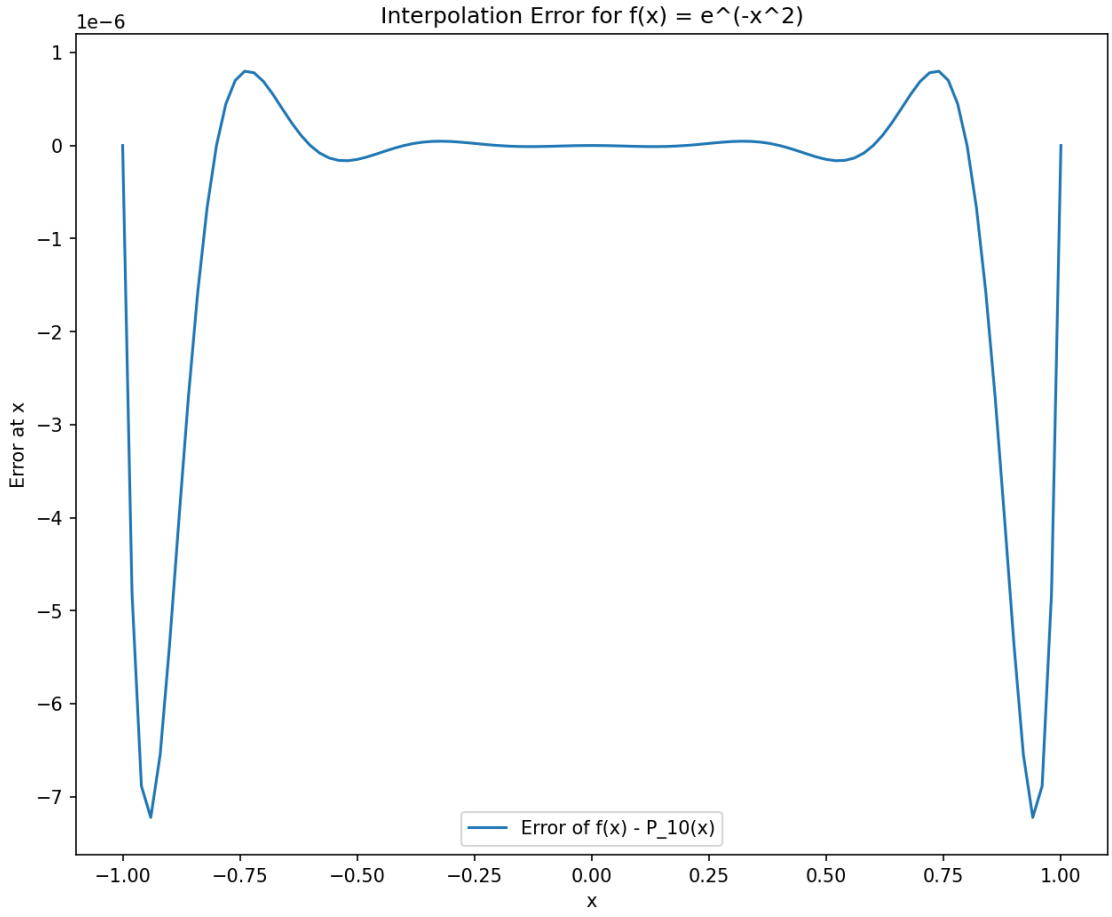
[7]:
```
# Test
nodes = np.array([1, 2, 3])
fnodes = np.array([2, 5, 10])

coefficients = newton_coeffs(nodes, fnodes)
print(f"The coefficients are: {coefficients}.")

x = 5/2
evaluation = newton_polynomials(coefficients, nodes, x)
print(f"The evaluation at x = 5/2 is {evaluation}.")
```

```
The coefficients are: [2 3 1].
The evaluation at x = 5/2 is 7.25.
```

(b). $f(x) = e^{-x^2}$

[14]:
```
import matplotlib.pyplot as plt

def f(x):
    return np.e**(-x**2)

x_nodes = np.linspace(-1, 1, 11)
fx_nodes = f(x_nodes)
coefficients_x = newton_coeffs(x_nodes, fx_nodes)

x_nodes_2 = np.linspace(-1, 1, 101)
fx_nodes_2 = f(x_nodes_2)

evaluations = []
for i in x_nodes_2:
    evaluation_i = newton_polynomials(coefficients_x, x_nodes, i)
    evaluations.append(evaluation_i)

errors = fx_nodes_2 - evaluations

plt.figure(figsize=(10,8), dpi=150)
plt.title("Interpolation Error for f(x) = e^(-x^2)")
```

```
plt.plot(x_nodes_2, errors, label = "Error of f(x) - P_10(x)")
plt.xlabel("x")
plt.ylabel("Error at x")
plt.legend()
plt.show()
```



4.

[20]:
```
def f_2(x):
    return x - np.exp(-x)

x_0 = 0.5
x_1 = 0.6
y_0 = f_2(x_0)
y_1 = f_2(x_1)

divdiff_inverse = (x_1 - x_0) / (y_1 - y_0)
```

```
P_1_0 = x_0 - y_0 * divdiff_inverse

print(f"The first approximation of the zero: P_1(0) is {P_1_0}")

x_2 = P_1_0
y_2 = f(x_2)

divdiff_inverse_01 = (x_1 - x_0) / (y_1 - y_0)
divdiff_inverse_12 = (x_2 - x_1) / (y_2 - y_1)
inv_diff_012 = (divdiff_inverse_12 - divdiff_inverse_01) / (y_2 - y_0)

P_2_0 = x_0 - y_0 * divdiff_inverse_01 + (0 - y_0) * (0 - y_1) *␣
 ↪divdiff_inverse_12
print(f"The second approximation of the zero P_2(0) is: {P_2_0}")
```

```
The first approximation of the zero: P_1(0) is 0.5675445848373014
The second approximation of the zero P_2(0) is: 0.5678073931099946
```