Math 151AH HW 4 Andrea Barette

1) From lecture,

$$\|f - P_n\|_\infty \le \frac{1}{(n+1)!} 2^{-n} \|f^{(n+1)}\|_\infty$$
$$= \frac{1}{(n+1)!} 2^{-n} \|e^{-x}\|_\infty$$
$$= \frac{1}{(n+1)!} 2^{-n} e$$

We want this to be bounded by 10^{-10}, so

$$\|f - P_n\|_\infty \le \frac{1}{(n+1)!} 2^{-n} e \le 10^{-10}$$
$$\frac{1}{(n+1)!} 2^{-n} \le \frac{1}{e} 10^{-10}$$

Solving this numerically, $n = 10$ is the lowest number of points guaranteeing $\|f - P_n\|_\infty \le 10^{-10}$.

2) a. $T_{n+1}(x) = \cos(n\cos^{-1}(x) + \cos^{-1}(x))$

$$= \cos(n\cos^{-1}(x))\cos(\cos^{-1}(x)) - \sin(n\cos^{-1}(x))\sin(\cos^{-1}(x))$$

$$T_{n-1}(x) = \cos(n\cos^{-1}(x) - \cos^{-1}(x))$$
$$= \cos(n\cos^{-1}(x))\cos(-\cos^{-1}(x)) - \sin(n\cos^{-1}(x))\sin(-\cos^{-1}(x))$$
$$= \cos(n\cos^{-1}(x))\cos(\cos^{-1}(x)) + \sin(n\cos^{-1}(x))\sin(\cos^{-1}(x))$$

Then

$$T_{n+1}(x) + T_{n-1}(x) = 2\cos(n\cos^{-1}(x))\,x = 2x\,T_n(x)$$

So

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Then

$$T_2(x) = 2x(x) - 1 = 2x^2 - 1$$
$$T_3(x) = 2x(2x^2 - 1) - x = 4x^3 - 2x - x$$
$$= 4x^3 - 3x$$
$$T_4(x) = 2x(4x^3 - 3x) - (2x^2 - 1) = 8x^4 - 6x^2 - (2x^2 - 1)$$
$$= 8x^4 - 8x^2 + 1$$

b. By induction: Base case: for $T_1(x) = x$, the leading coefficient is clearly $2^{n-1} = 2^{1-1} = 1$. Also $T_1 \in P_1$, $T_0 \in P_0$.

Induction step: Assume the leading coefficient of $T_n(x)$ is 2^{n-1}, and that $T_n \in P_n$, $T_{n-1} \in P_{n-1}$.

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$
$$= 2x(2^{n-1}x^n + q(x)) - T_{n-1}(x)$$

where $q(x) \in P^{n-1}$, $T_{n-1} \in P^{n-1}$. Then

$$T_{n+1}(x) = 2^n x^{n+1} + 2x\,q(x) - T_{n-1}(x)$$

where $2x\,q(x) \in P^n$. Then the leading term of $T_{n+1}(x)$ is $2^n x^{n+1}$, and so the leading coefficient is 2^n, as desired.

c. $\{s_k\}_{k=0}^{n}$ $\quad s_k = \cos\left(\frac{2k-1}{2n}\pi\right)$

$$\begin{aligned} T_n(s_k) &= \cos\left(n \cos^{-1}\left(\cos\left(\frac{2k-1}{2n}\pi\right)\right)\right) \\ &= \cos\left(n \frac{2k-1}{2n}\pi\right) \\ &= \cos\left((2k-1)\frac{\pi}{2}\right) \end{aligned}$$

Since $2k-1$ is always odd, $(2k-1)\frac{\pi}{2} \in \{-\frac{\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{2}, \dots\}$, so

$$T_n(s_k) = \cos\left((2k-1)\frac{\pi}{2}\right) = 0$$

d. $T_n(x) = \cos(n \cos^{-1}(x))$

$$\frac{d}{dx} T_n(x) = -\sin(n\cos^{-1}(x))\, n \frac{-1}{\sqrt{1-x^2}} \qquad x \neq 1, -1$$

$$= \sin(n\cos^{-1}(x)) \frac{n}{\sqrt{1-x^2}} = 0$$

$$n \cos^{-1}(x) = \sin^{-1}(0) = k\pi \quad \text{for any } k \in \mathbb{N}$$

$$x = \cos\left(\frac{k\pi}{n}\right)$$

Since $x \in (-1, 1)$, we have to bound k by $(0, n) \cap \mathbb{N}$. Then, for $t_k = \cos(\frac{k\pi}{n})$, $0 < k < n$, each t_k is a local extrema of $T_n(x)$, and every local extrema to $T_n(x)$ is one of the t_k's. Further

$$\begin{aligned} T_n(t_k) &= \cos\left(n \cos^{-1}\left(\frac{k\pi}{n}\right)\right) \\ &= \cos(k\pi) = (-1)^k \end{aligned}$$

For $k = 0$: $t_0 = \cos(0) = 1$, and

$$\begin{aligned} T_n(t_0) &= \cos\left(n \cos^{-1}\left(\cos\left(\frac{0}{n}\right)\right)\right) \\ &= \cos(0) = 1 = (-1)^0 \end{aligned}$$

For $k = n$: $t_n = \cos(\pi) = -1$, and

$$\begin{aligned} T_n(t_n) &= \cos(n \cos^{-1}(\cos(\pi))) \\ &= \cos(n\pi) = (-1)^n \end{aligned}$$

Then the $\{t_k\}_{k=0}^{n}$ is the set of absolute extremas of $T_n(x)$.

1. Suppose for contradiction there exists $P \in \hat{\Pi}_n$ such that $\|P\|_\infty < \|T_n\|_\infty$ on $x \in [-1, 1]$. Since $P, \tilde{T}_n \in P_n$ are both monic polynomials, their leading term is the same, so $Q = T_n - P \in P_{n-1}$, so Q has $n-2$ roots. However, since $\|T_n\|_\infty > \|P\|_\infty$, and T_n achieves its absolute extrema at each $\{t_k\}_{k=0}^{n}$, $Q = T_n - P$ has sign $(-1)^k$ at each $k = 0, \dots, n$. Then, by the intermediate value theorem, Q has $n-1$ roots, a contradiction.

Homework4

March 10, 2024

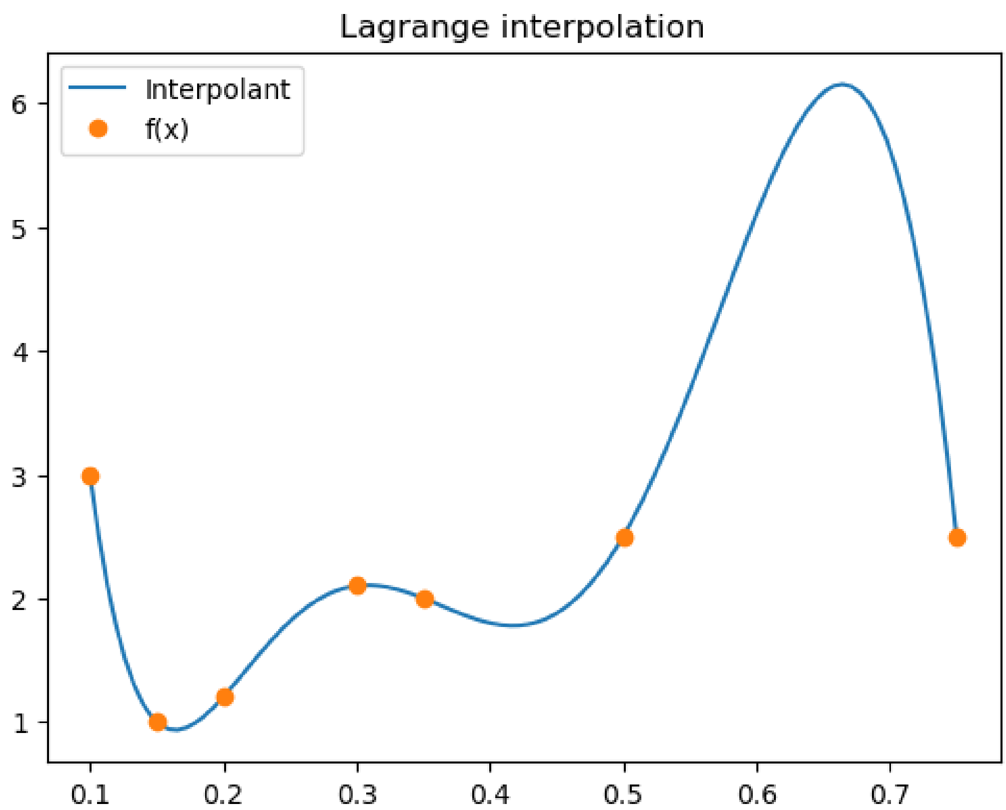
[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```python
def list_prod(l):
    ret = 1
    for x in l: ret *= x
    return ret

def list_sum(l):
    ret = 0
    for x in l: ret += x
    return ret
```

[3]:
```python
x_hat = [0.1, 0.15, 0.2, 0.3, 0.35, 0.5, 0.75]
y_hat = [3.0, 1.0, 1.2, 2.1, 2.0, 2.5, 2.5]

def lagrange_interpolant(x_hat, y_hat):
    phis = []
    for i,x_i in enumerate(x_hat):
        tmps = [lambda x,x_i=x_i,x_j=x_j: (x - x_j)/(x_i - x_j) for j,x_j in enumerate(x_hat) if i != j]
        phis.append(lambda x, tmps=tmps: list_prod([g(x) for g in tmps]))

    L = lambda x, y_hat=y_hat, phis=phis: list_sum([y_hat[i] * phis[i](x) for i,phi in enumerate(phis)])
    return lambda x: L(x), phis

L_1, phis = lagrange_interpolant(x_hat, y_hat)
```

[4]:
```python
xs = np.linspace(0.1, 0.75, 100)
plt.plot(xs, [L_1(x) for x in xs], label="Interpolant")
plt.plot(x_hat, y_hat, 'o', label="f(x)")
plt.title("Lagrange interpolation")
plt.legend()
plt.plot()
```

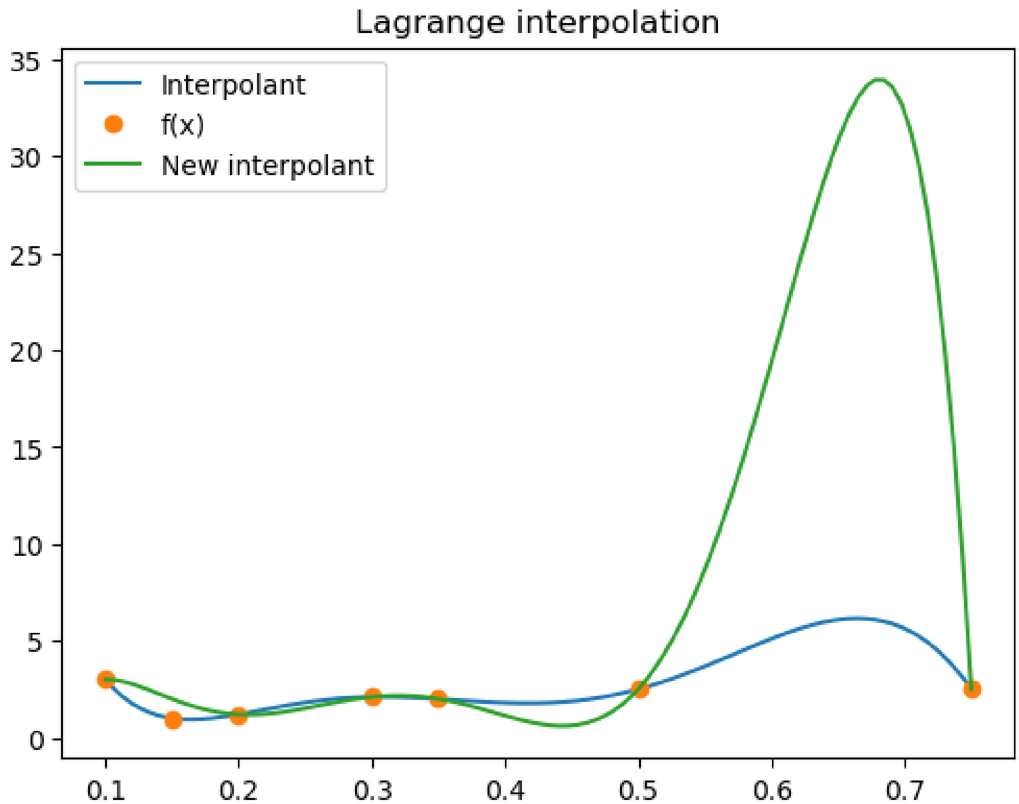
[4]: []



[5]:
```
x_hat_new = [0.1, 0.15, 0.2, 0.3, 0.35, 0.5, 0.75]
y_hat_new = [3.0, 2.0, 1.2, 2.1, 2.0, 2.5, 2.5]

L_2, _ = lagrange_interpolant(x_hat_new, y_hat_new)
```

[6]:
```
xs = np.linspace(0.1, 0.75, 100)
plt.plot(xs, [L_1(x) for x in xs], label="Interpolant")
plt.plot(x_hat, y_hat, 'o', label="f(x)")
plt.plot(xs, [L_2(x) for x in xs], label="New interpolant")
plt.title("Lagrange interpolation")
plt.legend()
plt.plot()
```

[6]: []



[7]:
```
print("Difference at x* = 0.7 is", L_2(0.7)-L_1(0.7))
```

```
Difference at x* = 0.7 is 26.666666666666675
```

Since $L_n(x) = \sum_{i=1}^{n} f(x_i)\phi_i(x)$, and $y_2 = f(x_2)$,

$$\frac{\partial L_n(x)}{\partial y_2} = \phi_2(x)$$

Then, the gradient at $x^* = 0.7$ is $\phi_2(0.7)$, which evaluates to

[8]:
```
print(phis[1](0.7))
```

```
26.66666666666667
```

To check numerically,

[9]:
```
def center_diff(f, x, h):
    return (f(x+h) - f(x-h))/(2*h)

grad_old_y2 = center_diff(lambda y2:\
                          lagrange_interpolant(
                            [0.1, 0.15, 0.2, 0.3, 0.35, 0.5, 0.75],
```

```
                                [3.0, y2, 1.2, 2.1, 2.0, 2.5, 2.5])[0](0.7), 1.0,␣
   ↪1e-10)
print("Gradient of polynomial with respect to old y2:", grad_old_y2)

grad_new_y2 = center_diff(lambda y2:\
                             lagrange_interpolant(
                                [0.1, 0.15, 0.2, 0.3, 0.35, 0.5, 0.75],
                                [3.0, y2, 1.2, 2.1, 2.0, 2.5, 2.5])[0](0.7), 2.0,␣
   ↪1e-10)
print("Gradient of polynomial with respect to new y2:", grad_new_y2)
```

```
Gradient of polynomial with respect to old y2: 26.66666887307656
Gradient of polynomial with respect to new y2: 26.66666887307656
```

Since the gradient with respect to y_2 is constant, we have that

$$\frac{\Delta L_n(x)}{\Delta y_2} = \phi_2(x)\Delta L_n(x) = \phi_2(x)\Delta y_2$$

Since in parts b and c we have that $\Delta y_2 = 1$, we get

$$\Delta L_n(x) = \phi_2(x)\Delta L_n(x^*) = \phi_2(x^*) = 1$$

Which is exactly the change in $L(x^*)$ which we found in part c.