1. **[Space of polynomials P_n, 2+2+2pts]** Let P_n be the space of functions defined on $[-1, 1]$ that can be described by polynomials of degree less of equal to n with coefficients in $\mathbb{R}$. P_n is a linear space in the sense of linear algebra, in particular, for $p, q \in P_n$ and $a \in \mathbb{R}$, also $p + q$ and ap are in P_n. Since the monomials $\{1, x, x^2, \ldots, x^n\}$ are a basis for P_n, the dimension of that space is $n + 1$.

   (a) Show that for pairwise distinct points $x_0, x_1, \ldots, x_n \in [-1, 1]$, the Lagrange polynomials $L_k(x)$ are in P_n, and that they are linearly independent, that is, for a linear combination of the zero polynomial with Lagrange polynomials with coefficients α_k, i.e.,

$$\sum_{k=0}^{n} \alpha_k L_k(x) = 0 \text{ (the zero polynomial)}$$

   necessarily follows that $\alpha_0 = \alpha_1 = \ldots = \alpha_n = 0$. Note that this implies that the $(n + 1)$ Lagrange polynomials also form a basis of P_n.

(a)

First of all, it is straight forward that $l_k(x) \in P_n$

Since $l_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} (x - x_i) \Big/ \prod_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i)$ is a function of degree n.

Then we show linear independency.

If $\sum_{k=0}^{n} \alpha_k l_k(x) = 0$.

then. $\sum_{k=0}^{n} \alpha_k l_k(x_i) = \alpha_i = 0$ (as $l_j(x_i) = \delta_{i,j}$)

It implies $\alpha_i = 0 \quad \forall i \in \{0, \ldots, n\}$

It follows linearly independency.

(b) Since both the monomials and the Lagrange polynomials are a basis of P_n, each $p \in P_n$ can be written as linear combination of monomials as well as Lagrange polynomials, i.e.,

$$p(x) = \sum_{k=0}^{n} \alpha_k L_k(x) = \sum_{k=0}^{n} \beta_k x^k, \tag{1}$$

with appropriate coefficients $\alpha_k, \beta_k \in \mathbb{R}$. As you know from basic matrix theory, there exists a basis transformation matrix that converts the coefficients $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_n)^T$ to the coefficients $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_n)^T$. Show that this basis transformation matrix is given by the so-called Vandermonde matrix $V \in \mathbb{R}^{n+1 \times n+1}$ given by

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} & x_n^n \end{pmatrix},$$

i.e., the relation between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in (1) is given by $\boldsymbol{\alpha} = V\boldsymbol{\beta}$. An easy way to see this is to choose appropriate x in (1).

Proof. We can interpret 1 in the 1st column as x_i^0

Therefore, it suffices to show that.

$$\alpha_i = \sum_{k=0}^{n} x_i^k \beta_k$$

However, this is pretty clear.

We can choose x to be $\{x_i\}_{i=0}^{n}$

and we can get $p(x_i) = \sum \alpha_k L_k(x_i) = \sum_k \beta_k x_i^k$

$$= \alpha_i = \sum_k \beta_k x_i^k$$

(As is shown in the last subquestion, $L_j(x_i) = \delta_{j,i}$)

(c) Note that since V transforms one basis into another basis, it must be an invertible matrix. Let us compute the condition number of V numerically.[1] Compute the 2-based condition number $\kappa_2(V)$ for $n = 5, 10, 20, 30$ with uniformly spaced nodes $x_i = -1 + (2i)/n$, $i = 0, \ldots, n$. Based on the condition numbers, can this basis transformation be performed accurately?

2. **[Hermite interpolation, 1+2+1+2pts]** We are given distinct interpolation points x_i, $i = 0, \ldots, n$. In class we introduced the Hermite interpolation polynomials $H_k(x)$ and $K_k(x)$ as follows:

$$H_k(x) = [L_k(x)]^2(1 - 2L_k'(x_k)(x - x_k)), \qquad K_k(x) = [L_k(x)]^2(x - x_k),$$

where L_k are the Lagrange polynomials.

(a) Show that $H_k, K_k \in P_{2n+1}$, i.e., they are polynomials of degree $2n+1$ or lower.

(a) H_k Here l_k is a polynomial with degree n

$1 - 2 l_k'(x_k) \in \mathbb{R}$

$\Rightarrow H_k(x)$ has degree $2n+1$

For K_k, l_k has degree n, $x - x_k$ as degree 1

Therefore, $K_k(x)$ has degree $n+1$.

(b) Show that H_k, K_k as defined above satisfy the following conditions:

$$H_k(x_i) = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k, \end{cases} \quad \text{and} \quad H_k'(x_i) = 0, \quad i = 0, \ldots, n.$$

$$K_k'(x_i) = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k, \end{cases} \quad \text{and} \quad K_k(x_i) = 0, \quad i = 0, \ldots, n.$$

(b) If $i=k$, $H_k(x_i) = [l_k(x_i)]^2 (1 - 2 l_k'(x_k)(x_k - x_k))$ ($l_k(x_i) = 1$, $x_k - x_k = 0$)

$= 1$

If $i \neq k$ $H_k(x_i) = [l_k(x_i)]^2 (1 - 2 l_k'(x_k)(x_k - x_k))$ ($l_k(x_i) = 0$)

$= 0$

$H_k'(x) = 2 l_k(x) \cdot l_k'(x) (1 - 2 l_k'(x_k)(x - x_k)) - 2 l_k^2(x) \cdot l_k'(x_k)$

If $i = k$, $H_k(x_i) = 2 \underbrace{l_k(x)}_{=1} l_k'(x) - 2 \underbrace{l_k^2(x)}_{=1} \cdot l_k'(x)$

$= 0$

If $i \neq k$, $H_k'(x_i) = 2 \underbrace{l_k(x)}_{=0} \cdot l_k'(x) \left(1 - 2 l_k'(x_k)(x - x_k)\right)$

$- 2 \underbrace{l_k^2(x)}_{=0} \cdot l_k'(x_k) = 0$

$K_k(x_i) = l_k^2(x_i)(x - x_i) \begin{cases} i = k & x - x_i = 0 \\ i \neq k & l_k(x_i) = 0 \end{cases} \Rightarrow K_k(x_i) = 0$

$K_k'(x_i) = 2 l_k(x_i) \cdot l_k'(x_i)(x - x_i) + l_k^2(x_i)$

If $i = k$, $2 l_k(x_i) \cdot l_k'(x_i) \underbrace{(x - x_i)}_{=0} + \underbrace{l_k^2(x_i)}_{=1} = 1$

If $i \neq k$ $2 \underbrace{l_k(x_i)}_{1'0} \cdot l_k'(x_i)(x - x_i) + \underbrace{l_k^2(x_i)}_{1'0} = 0$

(c) Argue that H_k, K_k are the unique polynomials in P_{2n+1} satisfying the conditions in (b).

For H_k, Assme $\widehat{H}_k = \sum_{i=0}^{2n+1} \beta_i x^i$ satisfies $\widehat{H}_k(x_j) = \delta_{jk}$

$\widehat{H}_k(x_j) = 0$

$\widehat{H}_k' = \sum_{i=0}^{2n+1} (i\beta_i) x^{i-1}$

Therefore, it satisfies $\sum_{i=0}^{2n+1} \beta_i x_j^i = \delta_{jk}$

$\sum_{i=0}^{2n+1} (i\beta_i) x^{i-1} = 0$

we can write the linear system into the language of matrix.

$$\begin{bmatrix} 1 & x_0^1 & x_0^2 & \cdots & x_0^{2n+1} \\ \vdots & & & & \\ 1 & x_n^1 & x_0^2 & \cdots & x_0^{2n+1} \\ 0 & 1 & 2x_0^1 & 3x_0^2 & \cdots (2n+1)x_0^{2n} \\ \vdots & & & & \\ 0 & 1 & 2x_n^1 & 3x_n^2 & \cdots (2n+1)x_n^{2n} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_{2n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 1 \leftarrow k^{th} \text{ entry} \\ \vdots \\ 0 \end{bmatrix}$$

$A \qquad \beta \qquad \delta$

This is not a good solution.
A better solution is on the NEXT PAGE

Note that, Since $\{x_i\}$ are distinct. A is invertible

Therefore, $A\beta = \delta$ has a unique solution β, which are coefficients of $\widehat{H}_k(x)$. But we know $H_k(x)$ is such a polynomial. This gives us $H_k(x) = \widehat{H}_k(x)$ and uniqueness follows.

The story is almost the same for $K_k(x)$.

we assme $\exists\ \widehat{K}_k(x) = \sum_{i=0}^{2n+1} \gamma_i x^i$ s.t. $\widehat{K}_k(x)$ satisfies the condition.

It gives us the linear system, in the language of matrix.

$$\begin{bmatrix} 1 & x_0^1 & x_0^2 & \cdots & x_0^{2n+1} \\ \vdots & & & & \\ 1 & x_n^1 & x_0^2 & \cdots & x_0^{2n+1} \\ 0 & 1 & 2x_0^1 & 3x_0^2 & \cdots (2n+1)x_0^{2n} \\ \vdots & & & & \\ 0 & 1 & 2x_n^1 & 3x_n^2 & \cdots (2n+1)x_n^{2n} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \vdots \\ \gamma_{2n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 1 \rightarrow 2n+2+k^{th} \text{ entry} \\ \vdots \\ 0 \end{bmatrix} \Rightarrow$$

A

(k^{th} entry of x^{2n+1} term).

Since A is invertible given by different $\{x_i\}$.

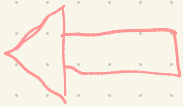
The solution is unique

and $K_k(x)$ is the only possible solution.

(c) Argue that H_k, K_k are the unique polynomials in P_{2n+1} satisfying the conditions in (b).

We Assume there's $\widehat{H_k} \in P_{2n+1}$ that meets the requirements above.

Let $h = H_k - \widehat{H}_k \in P_{2n+1}$



want to show $h = 0$

Indeed, since $H_k(x_i) = \widehat{H_k}(x_i) = 0$ for $i \neq k$

and $H_k(x_i) = \widehat{H_k}(x_i) = 1$ for $i = k$.

we have $h(x_i) = 0 \quad \forall i \in \{0, \cdots, n\}$

Also, $h'(x_i) = H_k'(x_i) - \widehat{H}_k'(x_i) = 0 \quad \forall i \in \{0, \cdots, n\}$.

It implies each x_i is at least a double root.

Therefore, $h \in P_{2n+1}$ has at least $2(n+1)$ roots (counting multiplicity)

However, according to fundemental Thm of Algebra, h has at most $2n+1$ roots. The only possible case is h is constantly zero.

It implies that $\widehat{H_k} = H_k$, and uniqueness follows.

The uniqueness of K_k is almost the same.

The punchline is using the fundemental Thm of Algebra.

(d) Find a (Hermite) polynomial $p_3 \in \boldsymbol{P}_3$ that interpolates $f(x) := 3\exp(x)$ and f' in $x_0 = 0, x_1 = 1/2$. Give the polynomial p_3 in the Hermite basis, plot f and p_3 in the same graph, and plot the four Hermite basis functions in another graph.

(d)

$f(x) = 3\exp(x)$

$y_0 = 3 \qquad z_0 = 3$

$y_1 = 3e^{\frac{1}{2}} \qquad z_1 = 3e^{\frac{1}{2}}$

$l_0 = \frac{(x-\frac{1}{2})}{-\frac{1}{2}} = -2(x-\frac{1}{2}) \qquad l_0'(0) = -2$

$l_1 = \frac{x}{\frac{1}{2}} = 2x \qquad l_1'(\frac{1}{2}) = 2$

$H_0(x) = 4(x-\frac{1}{2})^2(1+4x)$

$H_1(x) = 4x^2(1-4(x-\frac{1}{2})) = 4x^2(3-4x)$

$K_0(x) = 4(x-\frac{1}{2})^2 x$

$K_1(x) = 4x^2(x-\frac{1}{2})$

Basis functions

$\Rightarrow p_3 = 3\cdot 4(x-\frac{1}{2})^2(1+4x) + 3\cdot 4x(x-\frac{1}{2})^2$

$+ 3\cdot e^{\frac{1}{2}} 4x^2(3-4x) + 3\cdot e^{\frac{1}{2}} 4x^2(x-\frac{1}{2})$

3. **[Newton-Cotes Rules, 3+2+2pts]**

(a) For $f(x) = x^2 + x + 1$ on $[0, 2]$, we are interested in numerical approximations of the integral $I := \int_0^2 f(x)\,dx$. By splitting the integral into two,

$$\int_0^2 f(x)\,dx = \int_0^1 f(x)\,dx + \int_1^2 f(x)\,dx,$$

and using the trapezoidal rule on the subintervals $[0, 1]$ and $[1, 2]$, find an approximation of I (i.e., use the composite Trapeziodal rule for two subintervals). Also, use the Simpson's rule on $[0, 2]$ to approximate I. Which of the two approximations is more accurate, and why?

(a) First we do the Analytical Solution of the integral.

$$\int_0^2 f(x)\,dx = \frac{1}{3}x^3 + \frac{1}{2}x^2 + x \Big|_0^2 = \frac{8}{3} + 2 + 2 = \frac{20}{3}$$

Trapezoidal rule:

$$\int_0^2 f(x)dx \approx 1 \cdot \left(\frac{1}{2}f(0) + \frac{1}{2}f(1)\right) + 1 \cdot \left(\frac{1}{2}f(1) + \frac{1}{2}f(2)\right)$$

$$= \frac{1}{2} + \frac{3}{2} + \frac{3}{2} + \frac{7}{2} = 7$$

Simpson's rule

$$\int_0^2 f(x)\,dx \approx \frac{1}{6}\left(f(0) + 4f(1) + f(2)\right)$$

$$= \frac{1}{6}(1 + 12 + 7) = \frac{20}{3}$$

Simpson's Rule is a more accurate Approximation of the Integral. As, for polynomial of degree ≤ 2. Simpson's rule is exact.

(b) The error estimate for the Simpson's rule is given by

$$|E_2(f)| \leq \frac{(b-a)^5}{2880} M_4,$$

where $M_4 = \max_{x\in[a,b]} |f^{(iv)}(x)|$. Here, $f^{(iv)}$ denotes the 4-th derivative of f. Use the error estimate to explain which functions f are integrated exactly by the Simpson's rule.

As we can see from the expression,

Once $M_4 = 0$, then the approximation is exact.

The functions satisfies $M_4 = 0$ are

P_3 polynomials with degree 3 or less

(c) Let $f(x) = \frac{1}{4}x^4 + \sin(x)$. According to the error estimate, what is the maximal error you will make when integrating f over $[0, \pi]$? You do not need to calculate the approximate integral.

$f'(x) = x^3 + \cos(x)$

$f''(x) = 3x^2 - \sin(x)$

$f'''(x) = 6x - \cos(x)$

$f^{(iv)}(x) = 6 + \sin(x)$

$\Rightarrow \quad M_4(f) = \max_{x\in[0,\pi]} |f^{(iv)}(x)| = 7$

Therefore, the error is bounded by

$$|E_2 f| \leq \frac{7\pi^5}{2880}$$

4. **[Errors in polynomial interpolation, extra credit, 3pt]** Interpolate the function

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0, \end{cases}$$

on the domain $[-1, 1]$ using Lagrange polynomials with Chebyshev points.[2] You can use the following MATLAB function `lagrange_interpolant` to compute the values of the Lagrange interpolants p_n.

Describe qualitatively what you see for $n = 2, 4, 8, 16, 32, 64, 128, 256$ interpolation points. Provide a table of the maximum errors[3]

$$||p_n - f||_\infty = \max_{x \in [-1,1]} |p_n(x) - f(x)|,$$

and the L_2-errors[4]

$$||p_n - f||_2 = \sqrt{\int_{-1}^{1} (p_n(x) - f(x))^2 dx}$$

for each $n = 2, 4, 8, 16, 32, 64, 128, 256$. Do you expect convergence in the maximum norm? How about in the L_2 norm?

5. **[Composite trapezoidal and Simpson sum, extra credit, 2+2+2pt]** Write codes[5] to approximate integrals of the form

$$I(f) = \int_a^b f(t)\,dt$$

using the trapezoidal and Simpson's rule on the sub-intervals $[x_{i-1}, x_i]$, $i = 1, \ldots, m$, where $x_i = a + ih$, $i = 0, \ldots, m$ with $h = (b-a)/m$.[6]

(a) Hand in listings of your codes, and use them to approximate the integral

$$\int_{0.1}^{1} \sqrt{x}\,dx.$$

Compare the numerical errors $\mathcal{E}$ for both quadrature rules (the exact value of the integral is $\frac{2}{3} - \frac{1}{15\sqrt{10}}$). Try different m (e.g., $m = 10, 20, 40, 80, \ldots$) and plot the quadrature errors versus m in a double-logarithmic plot.

(b) To numerically study how the errors $\mathcal{E}$ decrease with m, we assume that the errors behaves like Cm^{κ}, with to-be-determined $C, \kappa \in \mathbb{R}$. Applying the logarithm to $\mathcal{E} = Cm^{\kappa}$ results in

$$\log(\mathcal{E}) = D + \kappa \log(m), \tag{2}$$

where $D = \log(C)$. Use the values for m and $\log(\mathcal{E})$ you computed in (a) to find the best-fitting values for D and κ in (2) by solving a least squares problem. Compare your findings for κ with the theoretical estimates for the composite trapezoidal and Simpson's rules.[7]

(c) Repeat steps (a) and (b) using $a = 0$ instead of $a = 0.1$ as lower integration bound. Can the theoretical estimates for the composite rules still be applied and why/why not?

It Cannot be applied.

As $f'(x) = \frac{1}{2} x^{-\frac{1}{2}}$

HW_6_code

April 30, 2022

[1]:
```
import numpy as np
from scipy.interpolate import lagrange
from numpy.polynomial.polynomial import Polynomial
import matplotlib.pyplot as plt
```

0.1 1(c)

[2]:
```
# define the vander generator function
# note that to make the notation consistent
# np.flip is used
vander_gen = lambda n: np.flip(np.vander(np.linspace(-1, 1, n+1)))
for n in [5, 10, 20, 30]:
    V = vander_gen(n)
    kappa_2 = np.linalg.cond(V, 2)
    print(f"The condition number for n = {n:2d} is: kappa_2 = {kappa_2:.3f}")
```

```
The condition number for n =  5 is: kappa_2 = 63.827
The condition number for n = 10 is: kappa_2 = 13951.627
The condition number for n = 20 is: kappa_2 = 831377053.878
The condition number for n = 30 is: kappa_2 = 56415165097885.938
```

- As we can see, when n grows, its condition number κ_2 grows quickly as well. It implies the basis transformation cannot be performed accurately.

0.2 2(d)

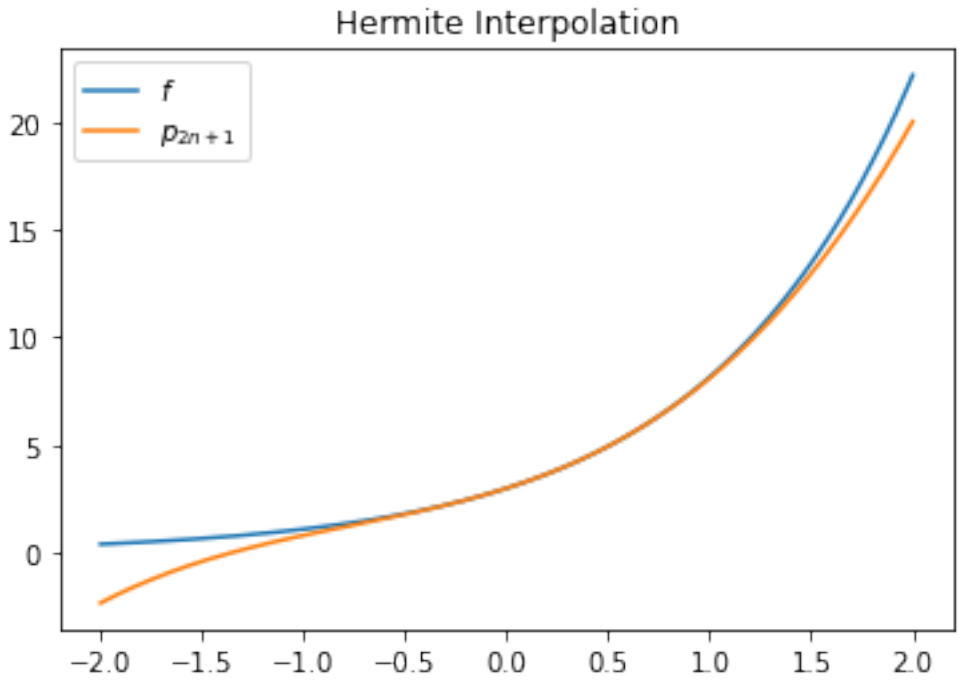
[3]:
```
H_0 = lambda x: 4*(x - 1/2)**2 * (1 + 4*x)
H_1 = lambda x: 4*x**2 * (3 - 4*x)
K_0 = lambda x: 4*(x - 1/2)**2 * x
K_1 = lambda x: 4*x**2 * (x - 1/2)

y_0, y_1, z_0, z_1 = 3, 3 * np.exp(1/2), 3, 3 * np.exp(1/2)

f = lambda x: 3 * np.exp(x)
p = lambda x: y_0 * H_0(x) + z_0 * K_0(x) + y_1 * H_1(x) + z_1 * K_1(x)
```

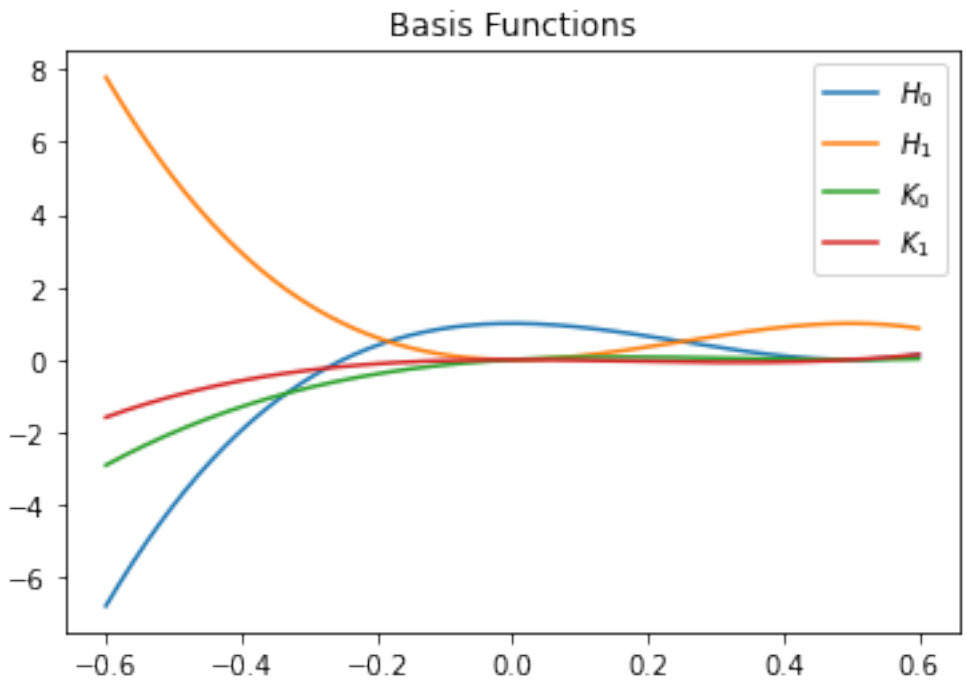
[4]:
```
x_space = np.linspace(-2, 2, 1000)
plt.plot(x_space, [f(x) for x in x_space], label=r"$f$")
```

```
plt.plot(x_space, [p(x) for x in x_space], label=r"$p_{2n+1}$")
plt.title("Hermite Interpolation")
plt.legend()
plt.show()
```



[5]:
```
x_space = np.linspace(-0.6, 0.6, 1000)
plt.plot(x_space, [H_0(x) for x in x_space], label=r"$H_0$")
plt.plot(x_space, [H_1(x) for x in x_space], label=r"$H_1$")
plt.plot(x_space, [K_0(x) for x in x_space], label=r"$K_0$")
plt.plot(x_space, [K_1(x) for x in x_space], label=r"$K_1$")
plt.title("Basis Functions")
plt.legend()
```

[5]:
```
<matplotlib.legend.Legend at 0x7fd7df168670>
```



0.3 4

[6]:
```
f = lambda x: 1 if x >= 0 else 0

def p(n, f):
    x_chebyshev = np.array([np.cos(((i + 1/2)*np.pi) / (n+1)) for i in␣
 ↪range(n+1)])
    y = np.array([f(x_i) for x_i in x_chebyshev])
    poly_coef = lagrange(x_chebyshev, y).coef[::-1]

    return Polynomial(poly_coef)

def l_infity(f, g, n):
    x_space = np.linspace(-1, 1, 10*n+1)
    candidates = [abs(f(x) - g(x)) for x in x_space]
    return max(candidates)

def l_2(f, g, n):
    x_space = np.linspace(-1, 1, 10*n)
    summation = sum([(f(x) - g(x))**2 for x in x_space])

    return ((2 * summation) / (10 * n)) ** (1/2)
```

[7]:
```
err_2_history = []
err_m_history = []
n_space = [2, 4, 8, 16, 32, 64, 128, 256]

for n in n_space:
    p_n = p(n, f)
    err_2 = l_2(f, p_n, n)
    err_m = l_infity(f, p_n, n)
    err_2_history.append(err_2)
    err_m_history.append(err_m)
```

[8]:
```
print("="*10 + " MAXIMAL ERROR " + "="*10)
for i, n in enumerate(n_space):
    print(f"n = {n:3d}: {err_m_history[i]}")

print()

print("="*12 + " L_2 ERROR "+"="*12)
for i, n in enumerate(n_space):
    print(f"n = {n:3d}: {err_2_history[i]}")
```

```
========== MAXIMAL ERROR ==========
n =   2: 0.9355983064143708
n =   4: 0.9425095430408871
n =   8: 0.9470005133845324
n =  16: 0.949636543244897
n =  32: 0.9510760906425554
n =  64: 524649557898087.44
n = 128: 1.2817169852842763e+47
n = 256: 5.814192150589475e+111

============ L_2 ERROR ============
n =   2: 0.602575825362048
n =   4: 0.4749343446137968
n =   8: 0.3582547960583235
n =  16: 0.2626019817271545
n =  32: 0.18945196425742922
n =  64: 61153226621720.484
n = 128: 1.0720035561706522e+46
n = 256: 3.464613188694958e+110
```

- The Maximal error doesn't converge in any senses.
- Analytically, we know the maximal error is bounded by $\frac{M_{n+1}}{(n+1)!}|\Pi_{n+1}|$. However, M_{n+1} is not finite in this case.
- The l_2 error seems to converge, but the error blows up when $n \geq 32$. From my point of view, thoeretically, the l_2 loss may converge. The blowing-up is due to some numerical error of python. According to visulization(ommited here), the lagrange interpolation is not exact at $x = 1$ when n grows large, which is not possible for lagrange interpolation by def.

0.4 5

[9]:
```
def trapez(f, a, b, m):
    x_space = np.linspace(a, b, m+1)
    temp = 0

    for x in x_space[1:-1]:
        temp += f(x)
    temp += (f(x_space[0]) + f(x_space[-1]))/2

    return (temp * (b-a)) / m

def simpson(f, a, b, m):
    x_space = np.linspace(a, b, m+1)

    def sub_simpson(f, a, b):
        c = (a + b) / 2
        return ((b-a)/6) * (f(a) + 4*f(c) + f(b))

    summation = 0
    for i in range(m):
        a, b = x_space[i], x_space[i+1]
        summation += sub_simpson(f, a, b)

    return summation
```

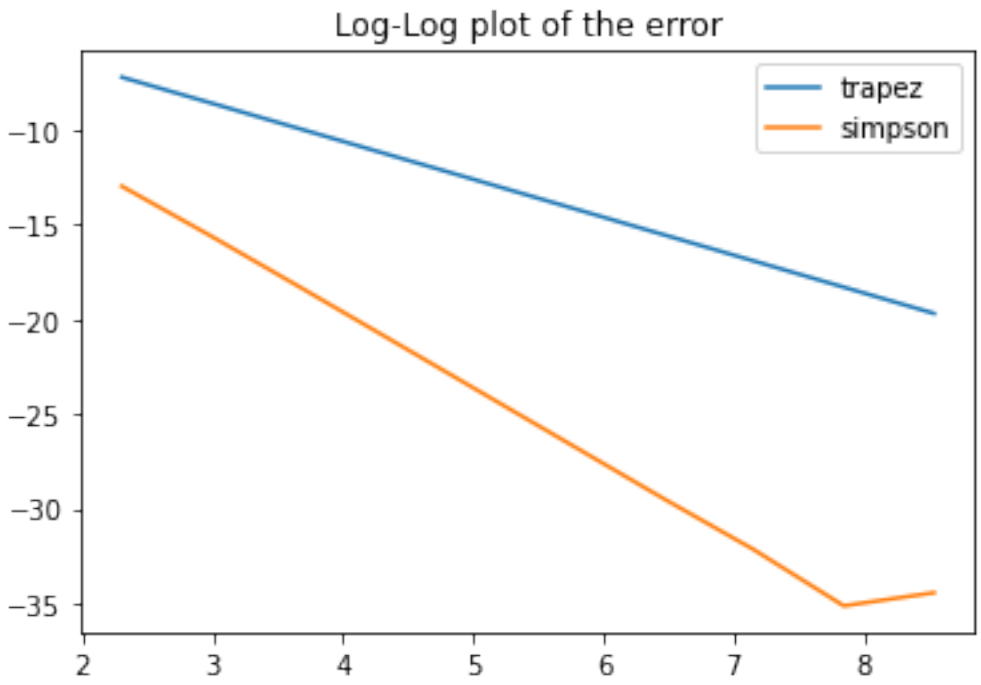
[10]:
```
f = lambda x: x ** .5
a, b= 0.1, 1
I_gt = 2/3 - 1/(15*10**.5)
err_simpson = []
err_trap = []

m_space = [10*2**i for i in range(10)]
for m in m_space:
    err_trap.append(abs(I_gt - trapez(f, a, b, m)))
    err_simpson.append(abs(I_gt - simpson(f, a, b, m)))

log_e_trap = np.log(err_trap)
log_e_simpson = np.log(err_simpson)
log_m = np.log(m_space)
```

[11]:
```
# Visualization
plt.plot(log_m, log_e_trap, label="trapez")
plt.plot(log_m, log_e_simpson, label="simpson")
plt.title("Log-Log plot of the error")
plt.legend()
```

[11]:
```
<matplotlib.legend.Legend at 0x7fd7df31f2b0>
```



want to find, such that D, κ that minimizes

$$\log(\epsilon) = D + \kappa \log(m)$$

It is equivalent to do least square estimation or linear regression.

[12]:
```
# Optimization
n = len(m_space)
A = np.ones((n, 2))
A[:, 1] = log_m
D_1, kappa_1 = np.linalg.solve(A.T @ A, A.T @ log_e_trap)
D_2, kappa_2 = np.linalg.solve(A.T @ A, A.T @ log_e_simpson)

print("Composite Trapezoidal Rule")
print(f"D: {D_1:.4f}, kappa: {kappa_1:.4f}")
print("Composite Simpson Rule")
print(f"D: {D_2:.4f}, kappa: {kappa_2:.4f}")
```

```
Composite Trapezoidal Rule
D: -2.6264, kappa: -1.9987
Composite Simpson Rule
D: -4.8010, kappa: -3.7244
```

0.4.1 Repeat that again for $a = 1$

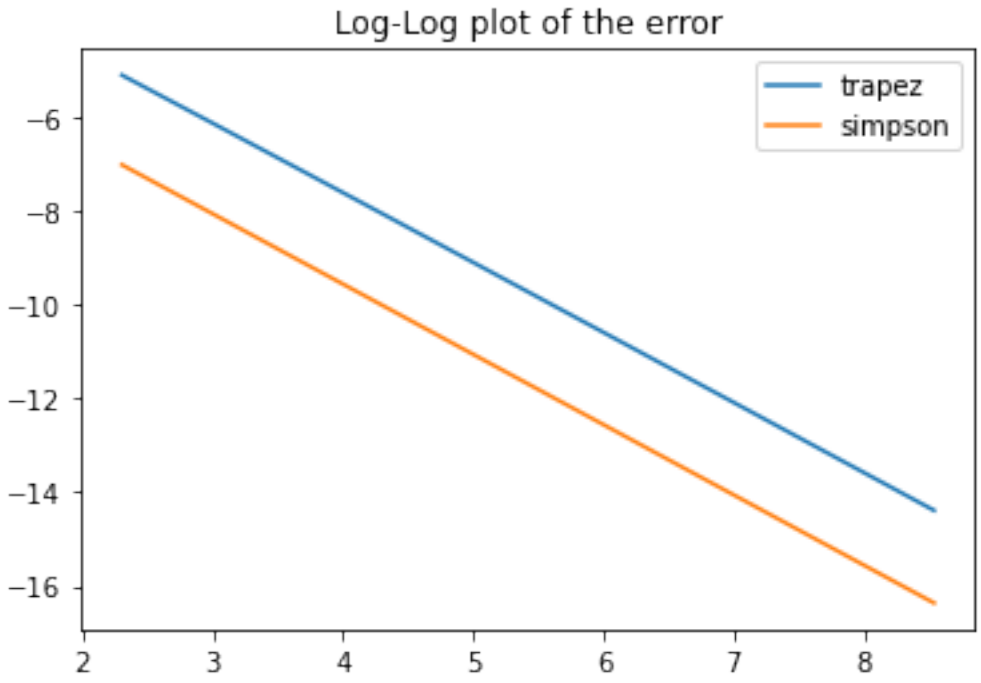
[13]:
```
f = lambda x: x ** .5
a, b= 0, 1
I_gt = 2/3
err_simpson = []
err_trap = []

m_space = [10*2**i for i in range(10)]
for m in m_space:
    err_trap.append(abs(I_gt - trapez(f, a, b, m)))
    err_simpson.append(abs(I_gt - simpson(f, a, b, m)))

log_e_trap = np.log(err_trap)
log_e_simpson = np.log(err_simpson)
log_m = np.log(m_space)
```

[14]:
```
# Visualization
plt.plot(log_m, log_e_trap, label="trapez")
plt.plot(log_m, log_e_simpson, label="simpson")
plt.title("Log-Log plot of the error")
plt.legend()
```

[14]: <matplotlib.legend.Legend at 0x7fd7df43a760>



[15]:
```
# Optimization
n = len(m_space)
A = np.ones((n, 2))
A[:, 1] = log_m
D_1, kappa_1 = np.linalg.solve(A.T @ A, A.T @ log_e_trap)
D_2, kappa_2 = np.linalg.solve(A.T @ A, A.T @ log_e_simpson)

print("Composite Trapezoidal Rule")
print(f"D: {D_1:.4f}, kappa: {kappa_1:.4f}")
print("Composite Simpson Rule")
print(f"D: {D_2:.4f}, kappa: {kappa_2:.4f}")
```

```
Composite Trapezoidal Rule
D: -1.6414, kappa: -1.4909
Composite Simpson Rule
D: -3.5508, kappa: -1.5000
```

0.4.2 Theoretical estimates

To answer this we need to look into the essence of theoretical estimates. Generally speaking, We have

$$\epsilon_1 \leq \frac{(b-a)^3}{12m^2} M_2$$

$$\epsilon_2 \leq \frac{(b-a)^5}{2880m^4} M_4$$

When M_2 and M_4 are bounded, we have

$$\log(err) = \kappa \log(m)$$

$$\kappa_1 \approx -2, \kappa_2 \approx -4$$

However, once we put the lowerbound of the integral to be 0, we encounter a problem. That is, the derivatives near $x = 0$ blows up. Therefore, the $\log(\epsilon)$ and $\log(m)$ is no longer in good linear relation since M is not bounded anymore, which fails the theoretical estimates

Though in practice we still get a linear relation, but we now don't have a theoretical base.