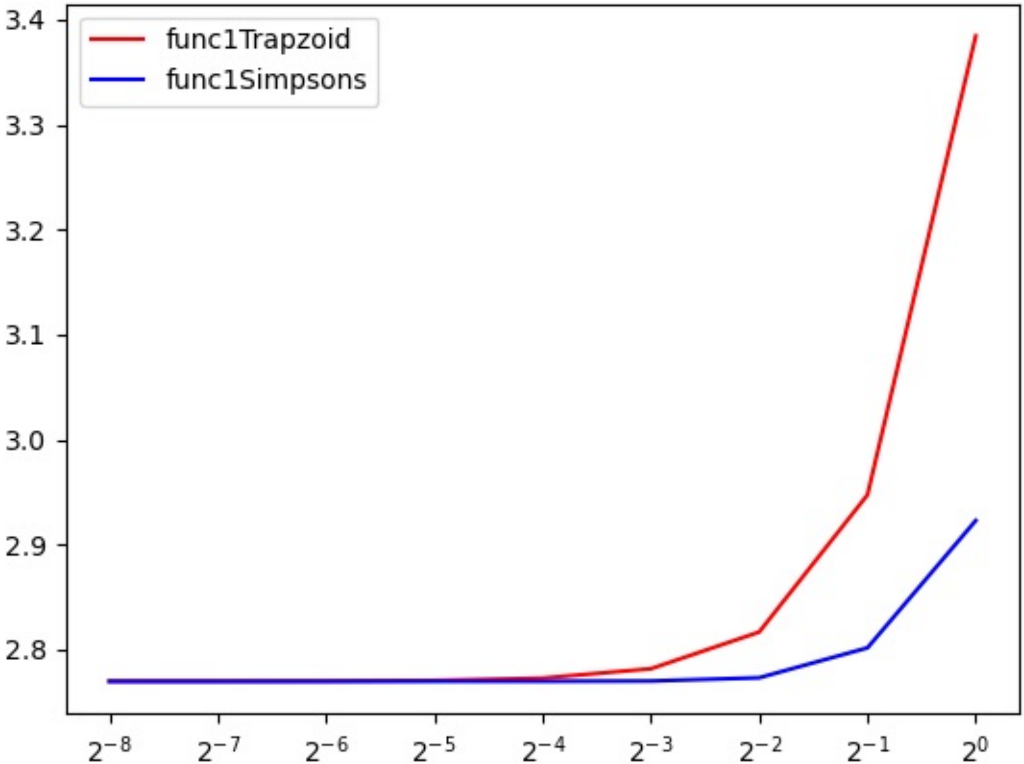
```python
import numpy as np
import matplotlib.pyplot as plt

def func1(x):
    return np.exp(np.power(x, 3)) + np.sin(np.power(x, 2)) + x

def func2(x):
    return np.cos(np.power(x, 5) + 4 * np.power(x, 4) + 3 * np.power(x, 3) + 2 * np.power(x, 
2) + x + 1)

def compositeTrapzoid(func, start, end, steps):
    trapzoid = np.zeros(9)
    for ind in range(steps.shape[0]):
        x = np.arange(start, end+steps[ind], steps[ind])
        trapzoid[ind] = steps[ind] * (func(x[0]) / 2 + func(x[-1]) / 2 + 
np.sum(func(x[1:-1])))
    return trapzoid

def compositeSimpsons(func, start, end, steps):
    simpsons = np.zeros(9)
    for ind in range(steps.shape[0]):
        x = np.arange(start, end+steps[ind], steps[ind])
        simpsons[ind] = steps[ind] * \
        (func(x[0]) + func(x[-1]) + \
         4* np.sum(func(x[1:-1:2])) + 2 * np.sum(func(x[2:-1:2])) \
         ) / 3
    return simpsons

pw = np.arange(0, 9)
steps = 0.5 ** pw

func1Trapzoid = compositeTrapzoid(func1, -1, 1, steps)
func1Simpsons = compositeSimpsons(func1, -1, 1, steps)

plt.plot(steps, func1Trapzoid, color="red", label="func1Trapzoid")
plt.plot(steps, func1Simpsons, color="blue", label="func1Simpsons")
plt.legend()
plt.xscale("log", base=2)
plt.savefig("hw6q1a.png")
plt.show()

func2Trapzoid = compositeTrapzoid(func2, 0, 2, steps)
func2Simpsons = compositeSimpsons(func2, 0, 2, steps)

plt.plot(steps, func2Trapzoid, color="red", label="func2Trapzoid")
plt.plot(steps, func2Simpsons, color="blue", label="func2Simpsons")
plt.legend()
plt.xscale("log", base=2)
plt.savefig("hw6q1b.png")
plt.show()
```

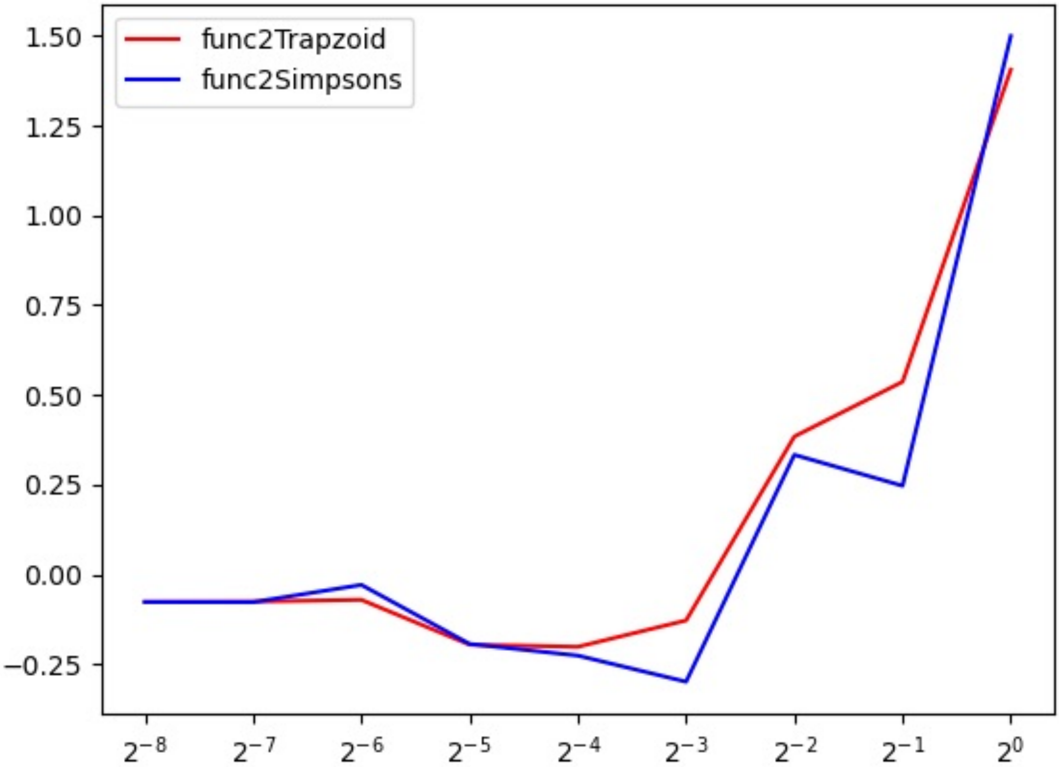
Question 1:

Question 1



func1Trapzoid
func1Simpsons
3.4
3.3
3.2
3.1
3.0
2.9
2.8
2^{-8}
2^{-7}
2^{-6}
2^{-5}
2^{-4}
2^{-3}
2^{-2}
2^{-1}
2^{0}

Question 1



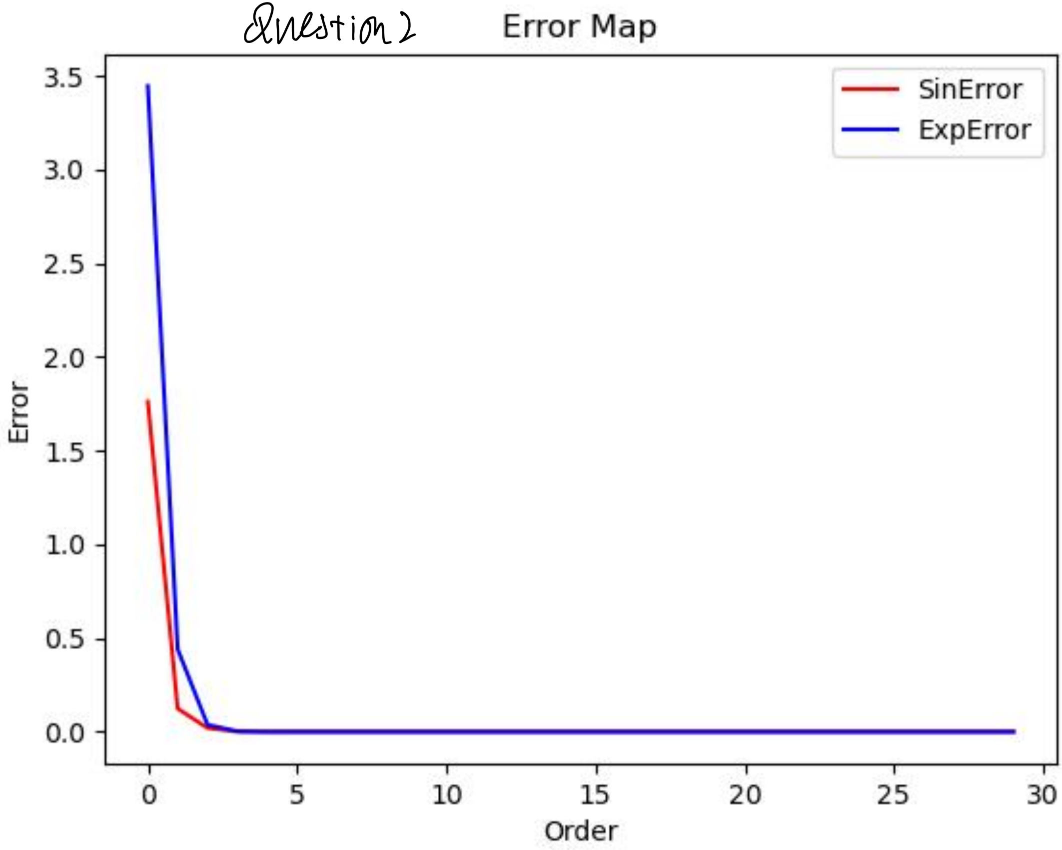
func2Trapzoid
func2Simpsons
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
2^{-8}
2^{-7}
2^{-6}
2^{-5}
2^{-4}
2^{-3}
2^{-2}
2^{-1}
2^{0}

```python
import numpy as np
from scipy.integrate import quad
import matplotlib.pyplot as plt

def Exp(x):
    return np.exp(x)

def Sin(x):
    return np.sin(x)

def integrand(x, f, i):
    return f(x) * np.power(x, i)

def interpolate(func, n):
    x1 = np.arange(0, n+1)
    x2 = (np.arange(0, n+1)).reshape(-1, 1)
    A = 1/(x1 + x2 + 1)
    b = np.array([quad(integrand, 0, 1, args=(func, i))[0] for i in range(n+1)])
    cis = np.linalg.solve(A, b)
    return cis

def pnOutput(coff, inputx):
    pw = np.arange(coff.shape[0])
    mid = np.power(inputx.reshape(-1, 1), pw)
    return mid @ coff

def getOutput(func, n, funcName="GroundTruth", savefig=False):
    coff = interpolate(func, n)
    inputx = np.arange(0, 10, 0.2)
    outputy = pnOutput(coff, inputx)
    groundTruth = func(inputx)
    error = np.abs(groundTruth - outputy)
    errorNorm = np.linalg.norm(error)
    plt.plot(inputx, outputy, label=f"approximation order {n}", color="red")
    plt.plot(inputx, groundTruth, label=f"{funcName}", color="blue")
    plt.title(f"n = {n}, error = {errorNorm}")
    plt.legend()
    if savefig:
        plt.savefig(f"hw6q2_{funcName}_order{n}.jpg")
    plt.show()

def getError(func, n):
    coff = interpolate(func, n)
    inputx = np.arange(0, 1, 0.02)
    outputy = pnOutput(coff, inputx)
    groundTruth = func(inputx)
    error = np.abs(groundTruth - outputy)
    errorNorm = np.linalg.norm(error)
    # plt.plot(inputx, outputy, label=f"approximation order {n}", color="red")
    # plt.plot(inputx, groundTruth, label=f"groundTruth", color="blue")
    # plt.title(f"n = {n}, error = {errorNorm}")
    # plt.legend()
    # plt.show()
    return errorNorm


def hw6q2():
    n = np.arange(30)
    sinError = np.array([getError(Sin, i) for i in n])
    expError = np.array([getError(Exp, i) for i in n])
    plt.plot(n, sinError, label = "SinError", color="red")
    plt.plot(n, expError, label = "ExpError", color="blue")
    plt.legend()
    plt.xlabel("Order")
    plt.ylabel("Error")
    plt.title("Error Map")
    plt.savefig("hw6q2.jpg")
```

```
    plt.show()

hw6q2()
```

Question 2:

Question 2



Error Map
SinError
ExpError
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Error
0
5
10
15
20
25
30
Order

Question 3:

(a). Let $i, j \in \mathbb{Z}^+$

$$\int_0^{2\pi} \cos(ix)\cos(jx)\,dx$$

$$= \int_0^{2\pi} \frac{1}{2}(\cos((i+j)x) + \cos((i-j)x))\,dx$$

$$= \frac{1}{2(i+j)}\sin((i+j)x)\Big|_0^{2\pi} + \frac{1}{2(i-j)}\sin((i-j)x)\Big|_0^{2\pi}$$

$$= 0 + 0 = 0$$

(b) Statement: $p_i(x) = (a_i x + b_i)p_{i-1}(x) + c_i p_{i-2}(x)$, $i \geq 2$

Let: $p_0 = 0$ $\quad p_1 = 1$

choose $a_i := 1$ $\quad b_i := -\dfrac{\langle p_{i-1}, xp_{i-1}\rangle}{\langle p_{i-1}, p_{i-1}\rangle}$ $\quad c_i := -\dfrac{\langle p_{i-1}, p_{i-1}\rangle}{\langle p_{i-2}, p_{i-2}\rangle}$

Let $\psi(x) := p_k - (x + b_k)p_{k-1}(x) - c_k p_{k-2}(x)$ for some $k \in \mathbb{Z} - \mathbb{Z}^-$

We can see $\langle \psi(x), p_j\rangle$ for some $j \in \{0, \dots, k-3\}$

$$\langle \psi(x), p_j(x)\rangle = \langle p_k, p_j(x)\rangle - \langle (x+b_k)p_{k-1}(x), p_j(x)\rangle - \langle c_k p_{k-2}(x), p_j(x)\rangle$$

where $p_k, p_j, p_{k-1}, p_{k-2}$ are vectors in the othogonal basis of polynomial space, so $\langle \psi(x), p_j(x)\rangle = 0$

Since under the construction of a_i, b_i, c_i, the coefficient of the term of the highest order of the components in the orthogonal basis is 1.

Therefore, $xp_{m-1} = p_m + q$ where $q \in \mathbb{P}_{m-1}$ (monicity of basis)

$$\langle \psi, p_{k-2}\rangle = \langle p_k, p_{k-2}\rangle - \langle (x+b_k)p_{k-1}, p_{k-2}\rangle - c_k\langle p_{k-2}, p_{k-2}\rangle$$

$$= 0 - \langle p_{k-1}, (x+b_k) p_{k-2} \rangle + \frac{\langle p_{k-1}, p_{k-1} \rangle}{\langle p_{k-2}, p_{k-2} \rangle} \langle p_{k-2}, p_{k-2} \rangle$$

$$= -\langle p_{k-1}, p_{k-1} \rangle - \langle p_{k-1}, q \rangle - b_k \langle p_{k-1}, p_{k-2} \rangle + \langle p_{k-1}, p_{k-1} \rangle$$

Here, by monocity of basis, $q \in \mathbb{P}_{k-2}$, so $\langle p_{k-1}, q \rangle = 0$

$$= 0$$

$$\langle \psi, p_{k-1} \rangle = \langle p_k, p_{k-1} \rangle - \langle (x+b_k) p_{k-1}, p_{k-1} \rangle - c_k \langle p_{k-2}, p_{k-1} \rangle$$

$$= 0 - \langle x p_{k-1} + b_k p_{k-1}, p_{k-1} \rangle - 0$$

$$= -\langle x p_{k-1}, p_{k-1} \rangle - b_k \langle p_{k-1}, p_{k-1} \rangle$$

$$= -\langle x p_{k-1}, p_{k-1} \rangle + \frac{\langle p_{i-1}, x p_{i-1} \rangle}{\langle p_{i-1}, p_{i-1} \rangle} \langle p_{k-1}, p_{k-1} \rangle$$

$$= 0$$

$$\psi = p_k - (x+b_k) p_{k-1} - c_k p_{k-2}$$

$$= p_k - x p_{k-1} - b_k p_{k-1} - c_k p_{k-2}$$

$$= p_k - p_k - q_{k-1} - b_k p_{k-1} - c_k p_{k-2}$$

$$= -q_{k-1} - b_k p_{k-1} - c_k p_{k-2} \quad \text{so } \psi \in \mathbb{P}_{k-1}$$

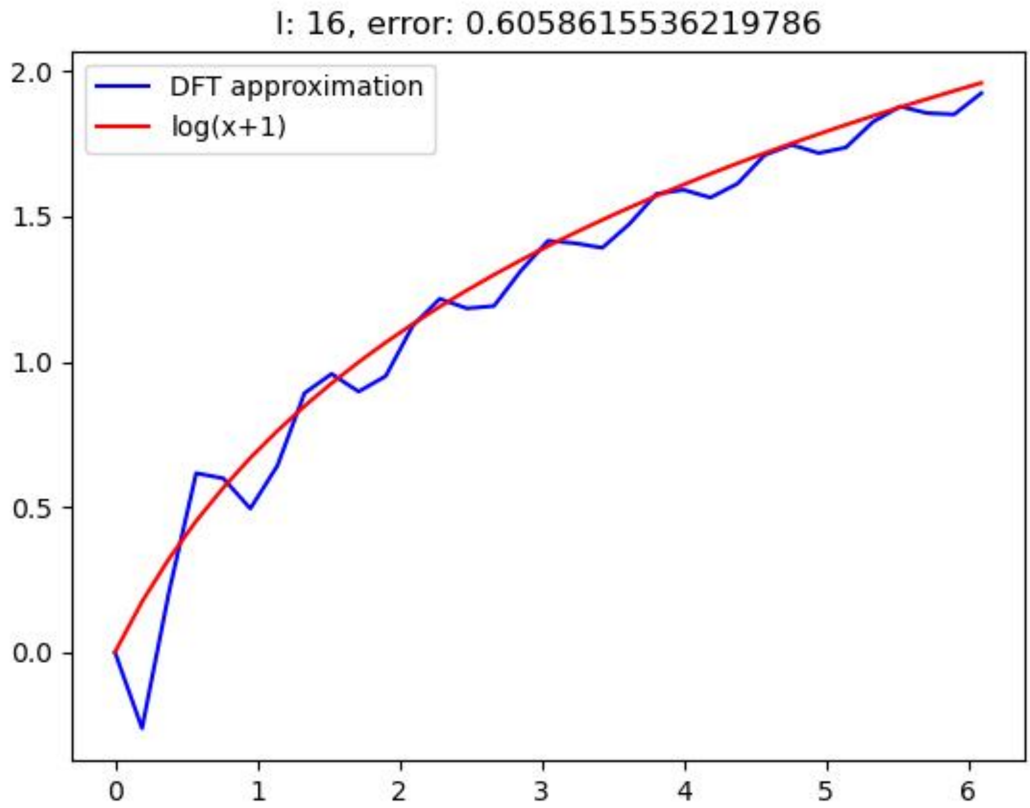
Since ψ is orthogonal to $p_0 \cdots p_{k-1}$, the orthogonal basis $\mathbb{P}_{k-1}$, ψ can only be 0, so

$$p_k = (x+b_k) p_{k-1} + c_k p_{k-2} \quad \square$$

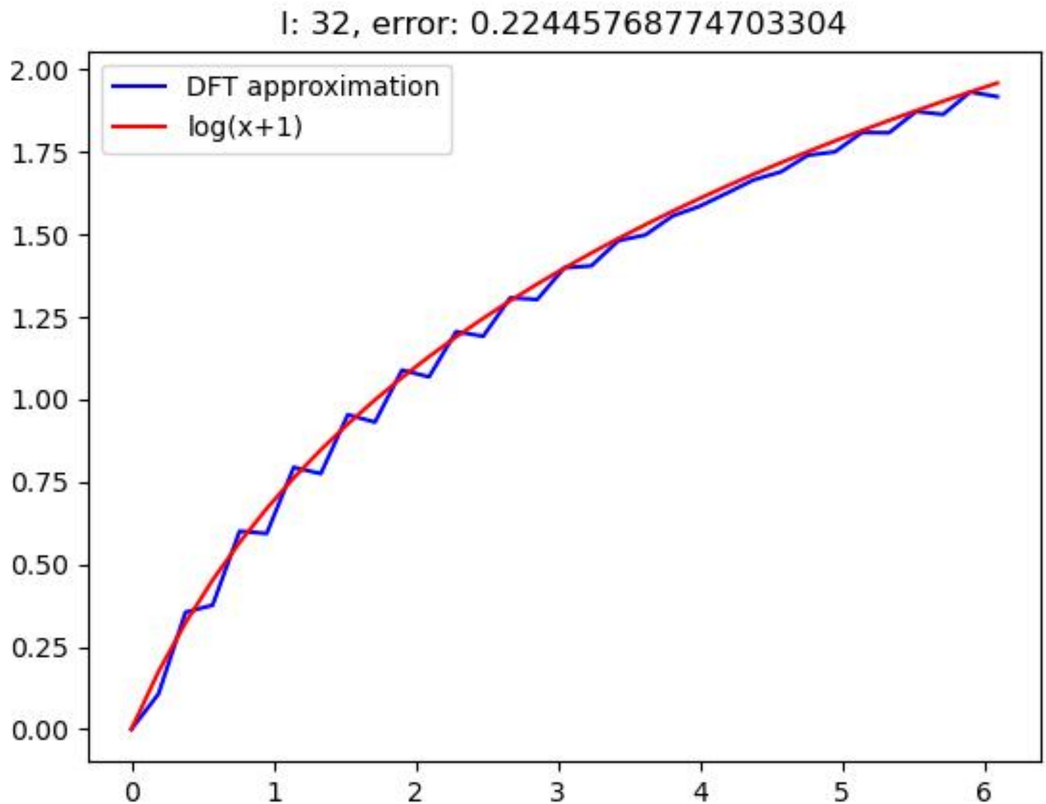
```python
import numpy as np
import matplotlib.pyplot as plt



def myDFT1(inputx, l, x, y):
    n = 2*l - 1
    acoff = np.arange(0, l+1).reshape(1, -1)
    bcoff = np.arange(1, l).reshape(1, -1)
    amid = (np.cos(x*acoff)).T
    bmid = (np.sin(x*bcoff)).T

    ak = 2 * (amid @ y) / (n + 1)
    bk = 2 * (bmid @ y) / (n + 1)

    akSum = np.array(ak[1:-1])
    coff = np.arange(1, l).reshape(-1, 1)
    inside = coff * inputx
    ainside = np.cos(inside).T
    binside = np.sin(inside).T
    asum = (ainside @ akSum).reshape(1, -1)
    print("asum1: \n", asum)

    bsum = (binside @ bk).reshape(1,-1)

    return asum + bsum + 0.5 * ak[0] + 0.5 * np.cos(l * inputx) * ak[-1]

def myDFT(inputx: np.array, l: int, x: np.array, y:np.array, endPoint=2*np.pi):
    # len(inputx) = m, len(x) = n+1, len(y) = n+1
    n = 2*l - 1
    DFTScalingFactor = 2 * np.pi / endPoint
    # assert len(x) == n + 1 and len(y) == n + 1
    acoffk = np.arange(0, l+1).reshape(-1, 1)  # shape of (l+1, 1)
    bcoffk = np.arange(1, l).reshape(-1, 1)    # shape of (l-1, 1)
    acoffk = acoffk * DFTScalingFactor
    bcoffk = bcoffk * DFTScalingFactor
    amid = np.cos(acoffk * x)                  # shape of (l+1, n+1)
    bmid = np.sin(bcoffk * x)                  # shape of (l-1, n+1)
    ak = 2* (amid @ y) / (n + 1)               # shape of (l+1)
    bk = 2* (bmid @ y) / (n + 1)               # shape of (l-1)
    # assert len(ak) == l+1 and len(bk) == l-1
    coffab = np.arange(1, l)
    coffab = coffab * DFTScalingFactor
    reshapeInputx = inputx.reshape(-1, 1)
    akSummation = np.array(ak[1:-1])
    bkSummation = bk
    # assert len(akSummation) == len(bk)
    asum = (np.cos(reshapeInputx * coffab) @ akSummation)
    bsum = (np.sin(reshapeInputx * coffab) @ bkSummation)
    # assert asum.shape == inputx.shape and bsum.shape == inputx.shape
    outputy = asum + bsum + 0.5 * (ak[0] + ak[-1] * np.cos(ak[l-1] * inputx))
    # assert len(outputy) == len(inputx)
    return outputy

def question1(l, endPoint= 2*np.pi, printB=False):
    n = 2*l - 1
    x = (endPoint * (np.arange(0, n + 1)) / ((n + 1)))
    y = np.log(x+1)
    print("question1 x: \n", x)

    inputx = (2 * np.pi * (np.arange(0, 32 + 1)) / (32 + 1))
    groundTruth = (np.log(inputx+1))

    myY = myDFT(inputx, l, x, y, endPoint)
    print("question1 myY: \n", myY)
```

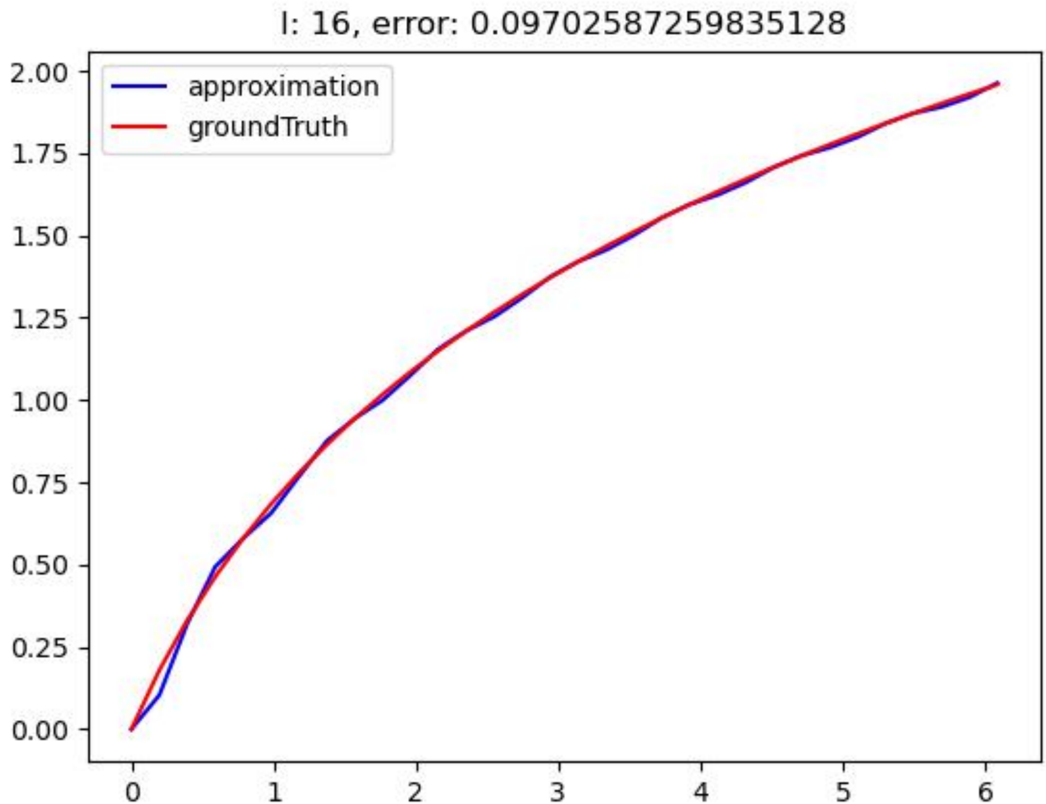
```
    error = myY - groundTruth
    errorNorm = np.linalg.norm(error)
    # print("myY: ", myY)
    # print("groundTruth: ", groundTruth)
    # print("error: ", error)
    plt.plot(inputx, myY, color="blue", label="DFT approximation")
    plt.plot(inputx, groundTruth, color="red", label="log(x+1)")
    plt.title(f"l: {l}, error: {errorNorm}")
    plt.legend()
    if printB:
        plt.savefig(f"question1_l={l}.jpg")
    plt.show()

# question1(16, True)

def f(x):
    return np.log(x + 1)

def g(x):
    return np.log(4 * np.pi - x + 1)

def getQuestion2Y(x):
    topX = x[x < 2 * np.pi]
    lowX = x[x >= 2 * np.pi]
    topY = f(topX)
    lowY = g(lowX)
    y = np.concatenate((topY, lowY))
    return y

def question2(l,endPoint= 2*np.pi, printB=False):
    n = 2*l - 1
    interpolatex = endPoint * np.arange(0, n + 1) / ((n + 1))
    interpolatey = getQuestion2Y(interpolatex)

    inputx = 2 * np.pi * np.arange(0, n + 1) / (n + 1)
    outputyTrue = getQuestion2Y(inputx)
    outputy = myDFT(inputx, l, interpolatex, interpolatey, endPoint)

    error = outputy - outputyTrue
    errorNorm = np.linalg.norm(error)
    plt.plot(inputx, outputy, color="blue", label="approximation")
    plt.plot(inputx, outputyTrue, color="red", label="groundTruth")
    plt.title(f"l: {l}, error: {errorNorm}")
    plt.legend()
    if printB:
        plt.savefig(f"question2_l={l}.jpg")
    plt.show()

question1(32, 4*np.pi, True)
```

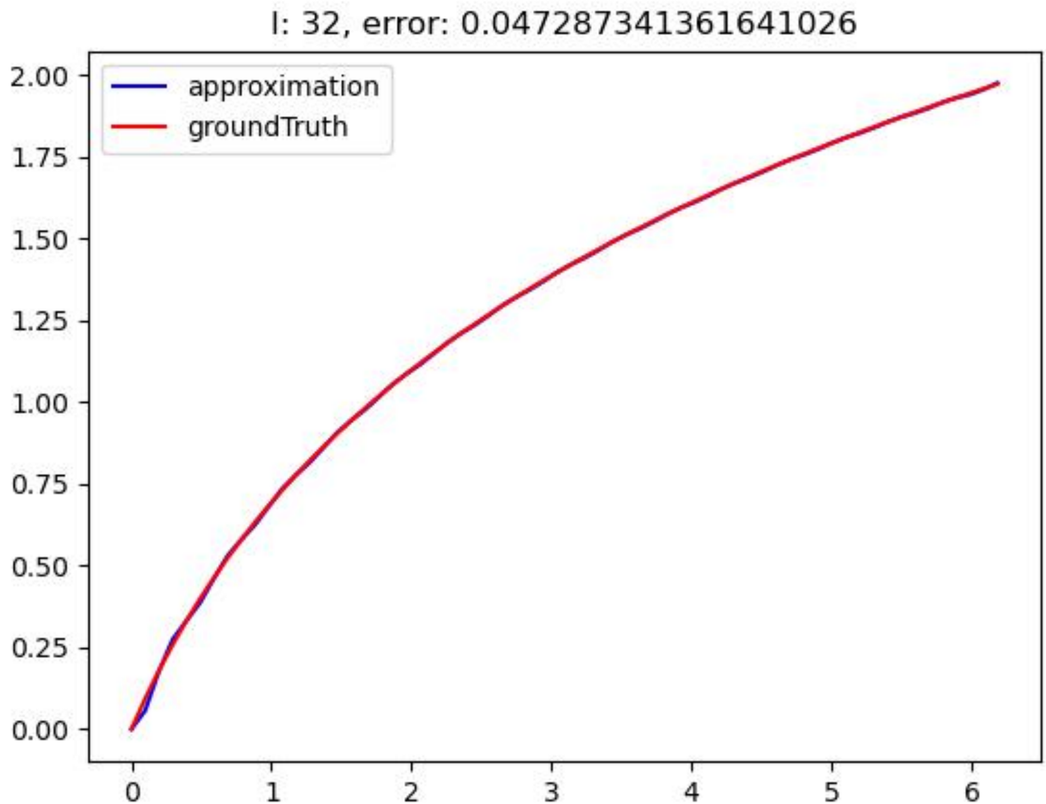
Question 4



l: 16, error: 0.6058615536219786
DFT approximation
log(x+1)
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6


l: 32, error: 0.22445768774703304
DFT approximation
log(x+1)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
2
3
4
5
6


l: 16, error: 0.09702587259835128
approximation
groundTruth
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
2
3
4
5
6



l: 32, error: 0.047287341361641026
approximation
groundTruth
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
2
3
4
5
6

Question 4 (2):

The results do look better. That is because we even extend the function so that the difference between begining and ending is smaller

Question 5: Let $z \in \mathbb{C}$ s.t. $|z|=1$, then $\exists\ \theta \in \mathbb{R}$ s.t. $z=e^{i\theta}$
so z is a root to $x^m=1$, and $z^m = e^{im\theta}=1$

Since $e^{2\pi i}=1$, so $e^{2\pi i k} = e^{im\theta}$ for some $k \in \mathbb{Z}-\mathbb{Z}^-$

Since the sum of all the roots to $x^m=1$ is 0.
we get $\sum_{j=0}^{m-1} e^{\frac{j 2\pi k i}{m}} = 0$

By Euler's Equation:

$$\sum_{j=0}^{m-1} \cos\left(\frac{jk}{m} 2\pi\right) + i \sum_{j=0}^{m-1} \sin\left(\frac{jk}{m} 2\pi\right) = 0$$

Therefore:

$$\sum_{j=0}^{m-1} \cos\left(\frac{jk}{m} 2\pi\right) = \sum_{j=0}^{m-1} \sin\left(\frac{jk}{m} 2\pi\right) = 0$$

now set $n = m-1$

$$\sum_{j=0}^{n} \cos\left(\frac{jk}{n+1} 2\pi\right) = \sum_{j=0}^{n} \sin\left(\frac{jk}{n+1} 2\pi\right) = 0$$

Let $x_j = \frac{2\pi j}{n+1}$, we get $\sum_{j=0}^{n} \cos(kx_j) = \sum_{j=0}^{n} \sin(kx_j) = 0$

Given $n=2l-1$, $x_i = \frac{2\pi i}{n+1}$ for $i=0, \cdots, n$

Equation ① $\frac{2}{n+1} \sum_{i=0}^{n} \cos(kx_i)\cos(jx_i)$

$$= \frac{1}{n+1} \sum_{i=0}^{n} \cos((k+j)x_i) + \cos((k-j)x_i)$$

$$= \frac{1}{n+1}\left(\sum_{i=0}^{n} \cos\left(\frac{(k+j)}{n+1} i\, 2\pi\right) + \sum_{i=0}^{n} \cos\left(\frac{(k-j)}{n+1} i\, 2\pi\right)\right) \quad (1)$$

Equation ②:

$$\frac{2}{n+1}\left(\sum_{i=0}^{n} \sin(kx_i)\sin(jx_i)\right)$$

$$= \frac{2}{n+1}\left(\sum_{i=0}^{n} -\frac{1}{2}\left(\cos((k+j)x_i) - \cos((k-j)x_i)\right)\right)$$

$$= -\frac{1}{n+1}\left(\sum_{i=0}^{n} \cos\left(\frac{(k+j)i}{n+1}2\pi\right) - \cos\left(\frac{(k-j)i}{n+1}2\pi\right)\right) \quad (2)$$

Equation ③:

$$\frac{2}{n+1}\sum_{i=0}^{n} \sin(kx_i)\cos(jx_i)$$

$$= \frac{1}{n+1}\sum_{i=0}^{n} \sin\left(\frac{(k+j)i}{n+1}2\pi\right) - \sin\left(\frac{(k-j)i}{n+1}2\pi\right) \quad (3)$$

For equation ①:

Case ①, $k \neq j$:

$$\sum_{i=0}^{n} \cos\left(\frac{(k+j)i}{n+1}2\pi\right) = \sum_{i=0}^{n} \cos\left(\frac{(k-j)i}{n+1}2\pi\right) = 0$$

case ②: $1 < k = j < l$, $k-j=0$:

$$\frac{1}{n+1}\left(\sum_{i=0}^{n} \cos\left(\frac{(k+j)i}{n+1}2\pi\right) + \sum_{i=0}^{n} \cos\left(\frac{(k-j)}{n+1}i\,2\pi\right)\right)$$

$$= \frac{1}{n+1}\left(\sum_{i=0}^{n} 0 + \sum_{i=0}^{n} \cos(0)\right)$$

$$= \frac{1}{n+1}\cdot(n+1) = 1$$

case ③: $k = j = 0$

$$(1) = \frac{1}{n+1}\left(\sum_{i=0}^{n} \cos(0) + \sum_{i=0}^{n} \cos(0)\right) = \frac{1}{n+1}(n+1+n+1) = 2$$

case ④ : $k=j=l$

$$(1) = \frac{1}{n+1}\left(2\sum_{i=0}^{n} \cos\left(\frac{2li}{2l} n\pi\right)\right)$$

$$= \frac{2}{n+1}\left(\sum_{i=0}^{n} \cos(i 2\pi)\right) = \frac{2}{n+1}(n+1) = 2$$

For Equation ②:

$$(2) \quad -\frac{1}{n+1}\left(\sum_{i=0}^{n} \cos\left(\frac{(k+j)i}{n+1}2\pi\right) - \cos\left(\frac{(k-j)i}{n+1}2\pi\right)\right)$$

Case ①, $k \neq j$:

$$-\frac{1}{n+1}(0-0) = 0$$

case ②: $1 < k = j < l$

$$\sum_{i=0}^{n} \cos\left(\frac{(k+j)i}{n+1}2\pi\right) = 0$$

$$\sum_{i=0}^{n} \cos\left(\frac{(k-j)i}{n+1}2\pi\right) = \sum_{i=0}^{n} 1 = n+1$$

$$(2) = -\frac{1}{n+1}(0-n-1) = 1$$

For equation ③:

$$\frac{1}{n+1}\sum_{i=0}^{n} \sin\left(\frac{(k+j)i}{n+1}2\pi\right) - \sin\left(\frac{(k-j)i}{n+1}2\pi\right) \quad (3)$$

(3) is 0 for all cases because no matter $1<k=j<l$, $k \neq j$, $k=j=0$, $k=j=l$. $\sum_{i=0}^{n} \sin\left(\frac{(k+j)i}{n+1}2\pi\right) = \sum_{i=0}^{n} \sin\left(\frac{(k-j)i}{n+1}2\pi\right) = 0$

This concludes the proof. $\square$