ME565 HW4

Ex 4-1

$$f(x) = \begin{cases} 0 & x < -1 \\ 1+x & -1 \le x < 0 \\ 1-x & 0 \le x < 1 \\ 0 & x > 1 \end{cases}$$

Fourier Series :

$$f(x) = \frac{a_0}{2} + \sum_{n=0}^{\infty} \left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right)$$

$$\Rightarrow \quad a_n = \frac{1}{L} \int_{-L}^{L} f(x) \cos\left(\frac{n\pi x}{L}\right) dx$$

where $L = 2$

$$\Rightarrow \quad b_n = \frac{1}{L} \int_{-L}^{L} f(x) \sin\left(\frac{n\pi x}{L}\right) dx$$

$$a_0 = \frac{1}{2} \int_{-2}^{2} f(x)\, dx$$

$$= \frac{1}{2} \int_{-1}^{0} (1+x)\, dx + \frac{1}{2} \int_{0}^{1} (1-x)\, dx$$

$$= \frac{1}{2} \left(x + \frac{1}{2} x^2 \Big|_{-1}^{0} \right) + \frac{1}{2} \left(x - \frac{1}{2} x^2 \Big|_{0}^{1} \right)$$

$$= \frac{1}{2} \left(-\left(-\frac{1}{2}\right) + \frac{1}{2} \right) = \frac{1}{2}$$

$$\frac{dv}{dx} = \cos\left(\frac{n\pi x}{2}\right)$$

$$v = \frac{2}{n\pi} \sin\left(\frac{n\pi x}{2}\right)$$

$$a_n = \frac{1}{2} \int_{-2}^{2} f(x) \cos\left(\frac{n\pi x}{2}\right) dx$$

$$= \frac{1}{2} \Bigg[\underbrace{\int_{-1}^{0} \underbrace{(1+x)}_{u} \underbrace{\cos\left(\frac{n\pi x}{2}\right) dx}_{dv}}_{①} + \underbrace{\int_{0}^{1} \underbrace{(1-x)}_{u} \underbrace{\cos\left(\frac{n\pi x}{2}\right) dx}_{dv}}_{②} \Bigg]$$

$$① = (1+x) \frac{2}{n\pi} \sin\left(\frac{n\pi x}{2}\right) \Big|_{-1}^{0} - \int_{-1}^{0} \frac{2}{n\pi} \sin\left(\frac{n\pi x}{2}\right) dx$$

$$= -\left[\left(\frac{2}{n\pi}\right)^2\left(-\cos\left(\frac{n\pi x}{2}\right)\right)\Big|_{-1}^{0}\right]$$

$$= \frac{4}{n^2\pi^2}\left[1-\cos\left(-\frac{n\pi}{2}\right)\right] = \underline{\frac{4}{n^2\pi^2}\left[1-\cos\frac{n\pi}{2}\right]}$$

② $= (1-x)\frac{2}{n\pi}\sin\left(\frac{n\pi x}{2}\right)\Big|_0^1 - \int_0^1 \frac{2}{n\pi}\sin\left(\frac{n\pi x}{2}\right)(-dx)$

$$= \left[\left(\frac{2}{n\pi}\right)^2\left(-\cos\left(\frac{n\pi x}{2}\right)\right)\Big|_0^1\right]$$

$$= \frac{4}{n^2\pi^2}\left[-\cos\frac{n\pi}{2}+1\right] = \underline{\frac{4}{n^2\pi^2}\left[1-\cos\frac{n\pi}{2}\right]}$$

$$a_n = \frac{1}{2}(① + ②) = \frac{4}{n^2\pi^2}\left(1-\cos\frac{n\pi}{2}\right)$$

$\frac{dv}{dx} = \sin\left(\frac{n\pi x}{2}\right)$

$v = -\frac{2}{n\pi}\cos\left(\frac{n\pi x}{2}\right)$

$$b_n = \frac{1}{2}\int_{-2}^{2} f(x)\sin\left(\frac{n\pi x}{2}\right)dx$$

$$= \frac{1}{2}\Bigg[\underbrace{\int_{-1}^{0}\overset{u}{(1+x)}\,\overset{dv}{\sin\left(\frac{n\pi x}{2}\right)dx}}_{③} + \underbrace{\int_0^1 \overset{u}{(1-x)}\,\overset{dv}{\sin\left(\frac{n\pi x}{2}\right)dx}}_{④}\Bigg]$$

③ $= \left[(1+x)\left(-\frac{2}{n\pi}\right)\cos\frac{n\pi x}{2}\Big|_{-1}^{0}\right] - \int_{-1}^{0}\left(-\frac{2}{n\pi}\right)\cos\frac{n\pi x}{2}\,dx$

$$= \left(-\frac{2}{n\pi}\right) + \left[\left(\frac{2}{n\pi}\right)^2\sin\left(\frac{n\pi x}{2}\right)\Big|_{-1}^{0}\right]$$

$$= -\frac{2}{n\pi} + \left(-\frac{4}{n^2\pi^2}\sin\left(-\frac{n\pi}{2}\right)\right)$$

$$= -\frac{2}{n\pi} + \frac{4}{n^2\pi^2}\sin\left(\frac{n\pi}{2}\right)$$

$$④ = \left[(1-x)\left(-\frac{2}{n\pi}\right)\cos\left(\frac{n\pi x}{2}\right) \Big|_0^1 \right] - \int_0^1 \left(-\frac{2}{n\pi}\right)\cos\frac{n\pi x}{2}(-dx)$$

$$= -\left(-\frac{2}{n\pi}\right) - \left[\left(\frac{2}{n\pi}\right)^2 \sin\left(\frac{n\pi x}{2}\right) \Big|_0^1 \right]$$

$$= \frac{2}{n\pi} - \left(\frac{4}{n^2\pi^2} \sin\frac{n\pi}{2} \right)$$

$$b_n = \frac{1}{2}(③ + ④) = 0$$

$$\boxed{\begin{aligned} &\therefore f(x) = A_0 + \sum_{n=0}^{\infty} A_n \cos\left(\frac{n\pi x}{2}\right) + B_n \sin\left(\frac{n\pi x}{2}\right) \\ &\text{where, } A_n = \frac{4}{n^2\pi^2}\left(1 - \cos\frac{n\pi x}{2}\right), \quad A_0 = \frac{1}{4}, \quad B_n = 0 \end{aligned}}$$

#

<u>Fourier Series</u>: represting a periodic function using discrete sum of orthogonal basis (Sine, Cosine)

<u>Fourier Transform</u>: representing a non-periodic function using the integral (Riemann Sum) of orthogonal basis (Sine, Cosine)

#

Exercise 4-1

Plot the approximation using n = 10 modes on top of the true triangle wave.

Code reference: [databook_python]

In [1]:

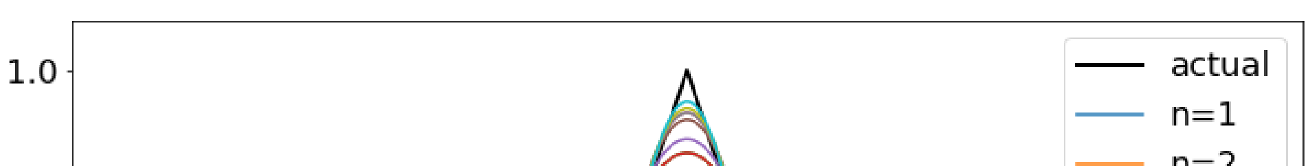
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.cm import get_cmap
```

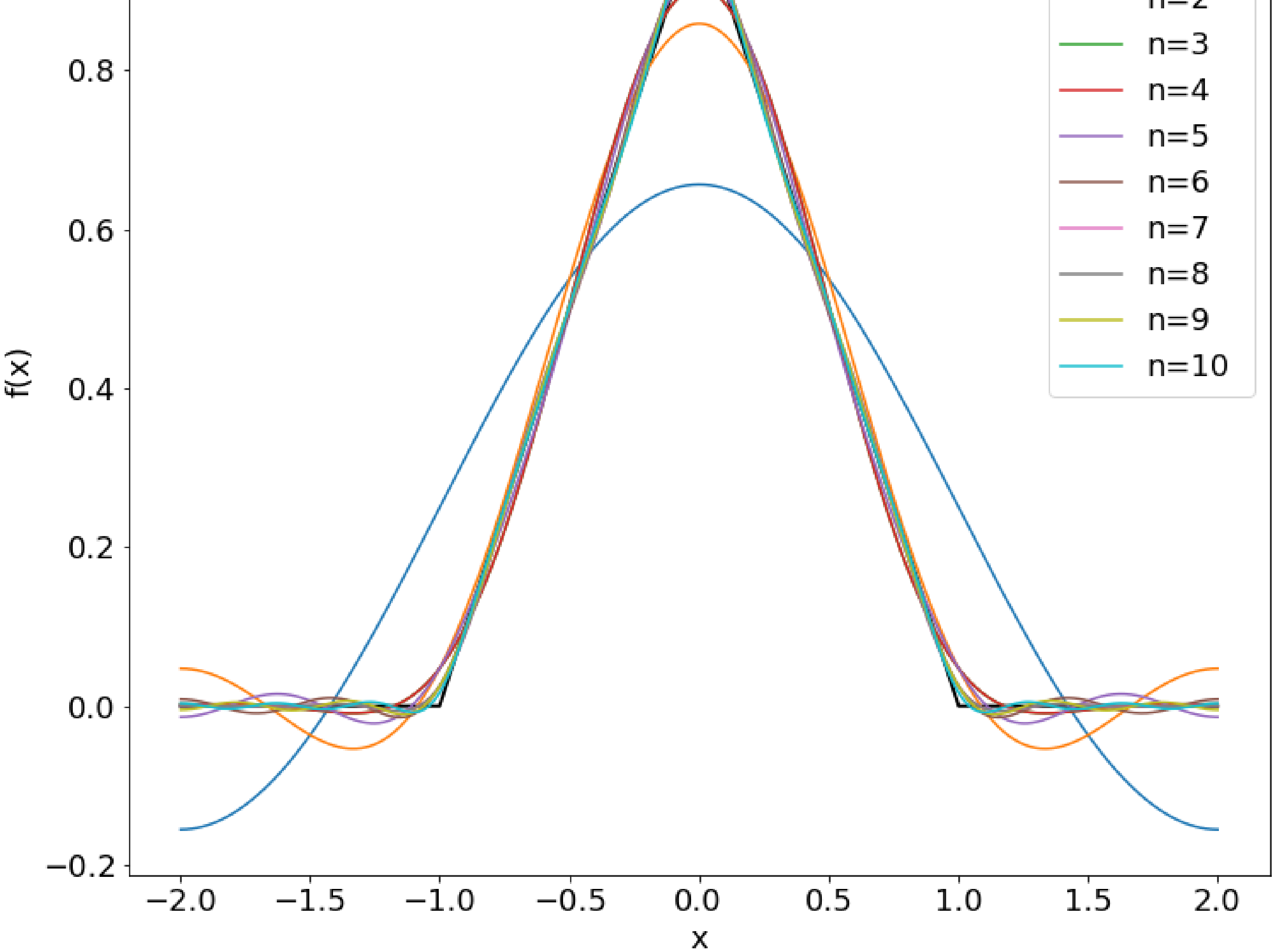
In [2]:

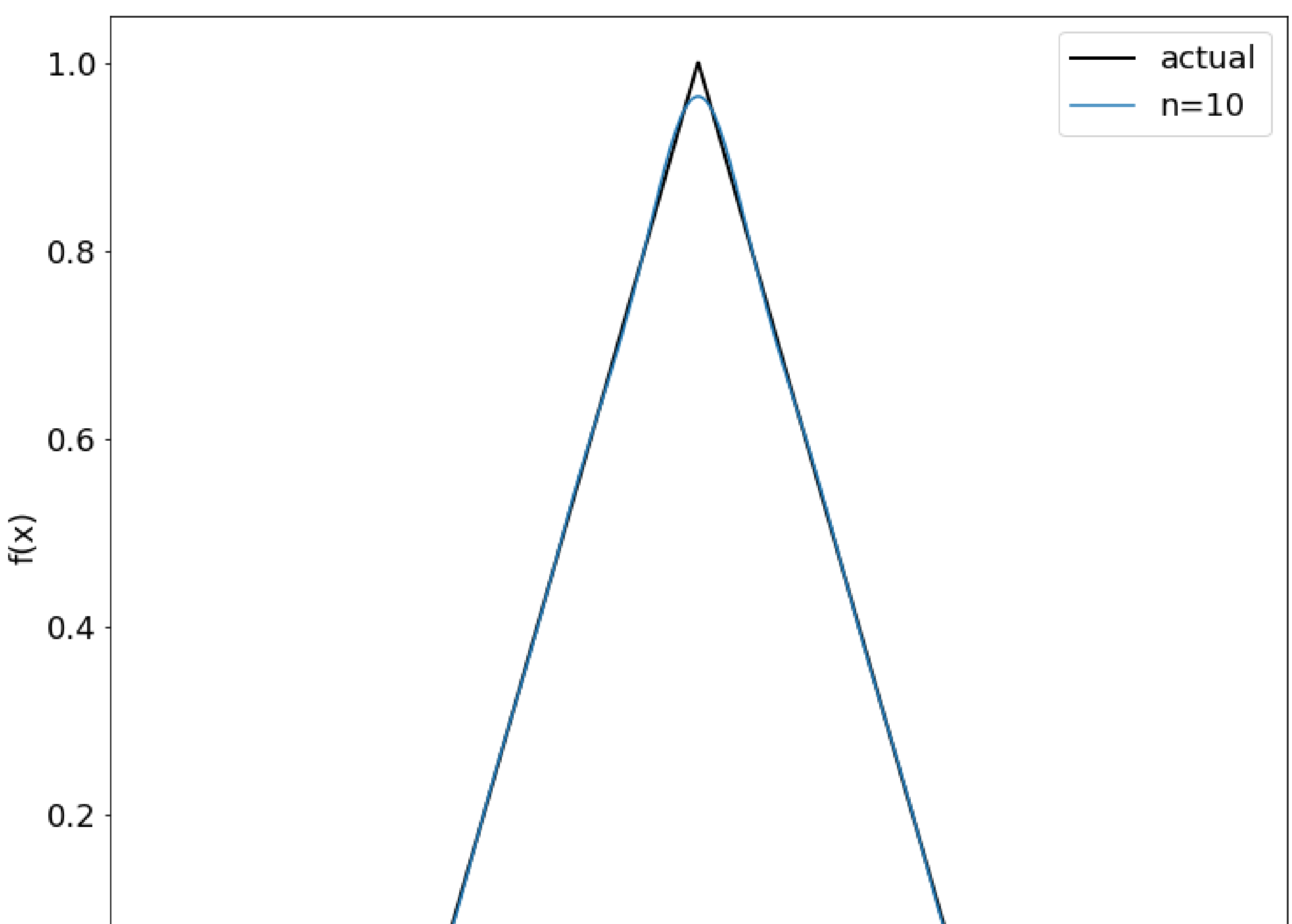
```
plt.rcParams['figure.figsize'] = [12, 24]
plt.rcParams.update({'font.size': 18})

# Define domain
dx = 0.001
L = 2.0
x = L * np.arange(-1+dx,1+dx,dx)
n = len(x)
nquart = int(np.floor(n/4))

# Define hat function
f = np.zeros_like(x)
f[nquart:2*nquart] = (4/n)*np.arange(1,nquart+1)
f[2*nquart:3*nquart] = np.ones(nquart) - (4/n)*np.arange(0,nquart)

fig, axs = plt.subplots(2,1)
axs[0].plot(x,f,'-',color='k',linewidth=2)
axs[1].plot(x,f,'-',color='k',linewidth=2)

# Compute Fourier series
name = "Accent"
cmap = get_cmap('tab10')
colors = cmap.colors
axs[0].set_prop_cycle(color=colors)

A0 = np.sum(f * np.ones_like(x)) * dx
fFS = A0/2

A = np.zeros(10)
B = np.zeros(10)
for k in range(10):
    A[k] = np.sum(f * np.cos(np.pi*(k+1)*x/L)) * dx # Inner product
    B[k] = np.sum(f * np.sin(np.pi*(k+1)*x/L)) * dx
    fFS = fFS + A[k]*np.cos((k+1)*np.pi*x/L) + B[k]*np.sin((k+1)*np.pi*x/L)
    axs[0].plot(x,fFS,'-')
axs[1].plot(x,fFS,'-')

# settings for the plots
axs[0].legend(['actual','n=1','n=2','n=3','n=4','n=5','n=6','n=7','n=8','n=9','n=10'])
axs[0].set_xlabel('x')
axs[0].set_ylabel('f(x)')
axs[1].legend(['actual','n=10'])
axs[1].set_xlabel('x')
axs[1].set_ylabel('f(x)')
```

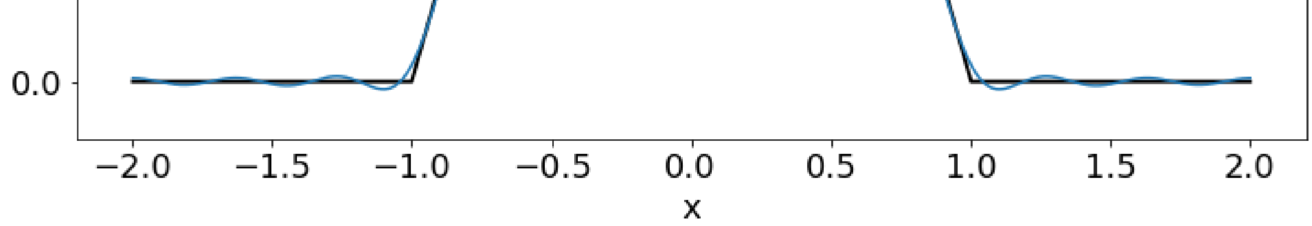
Out[2]:

```
Text(0, 0.5, 'f(x)')
```



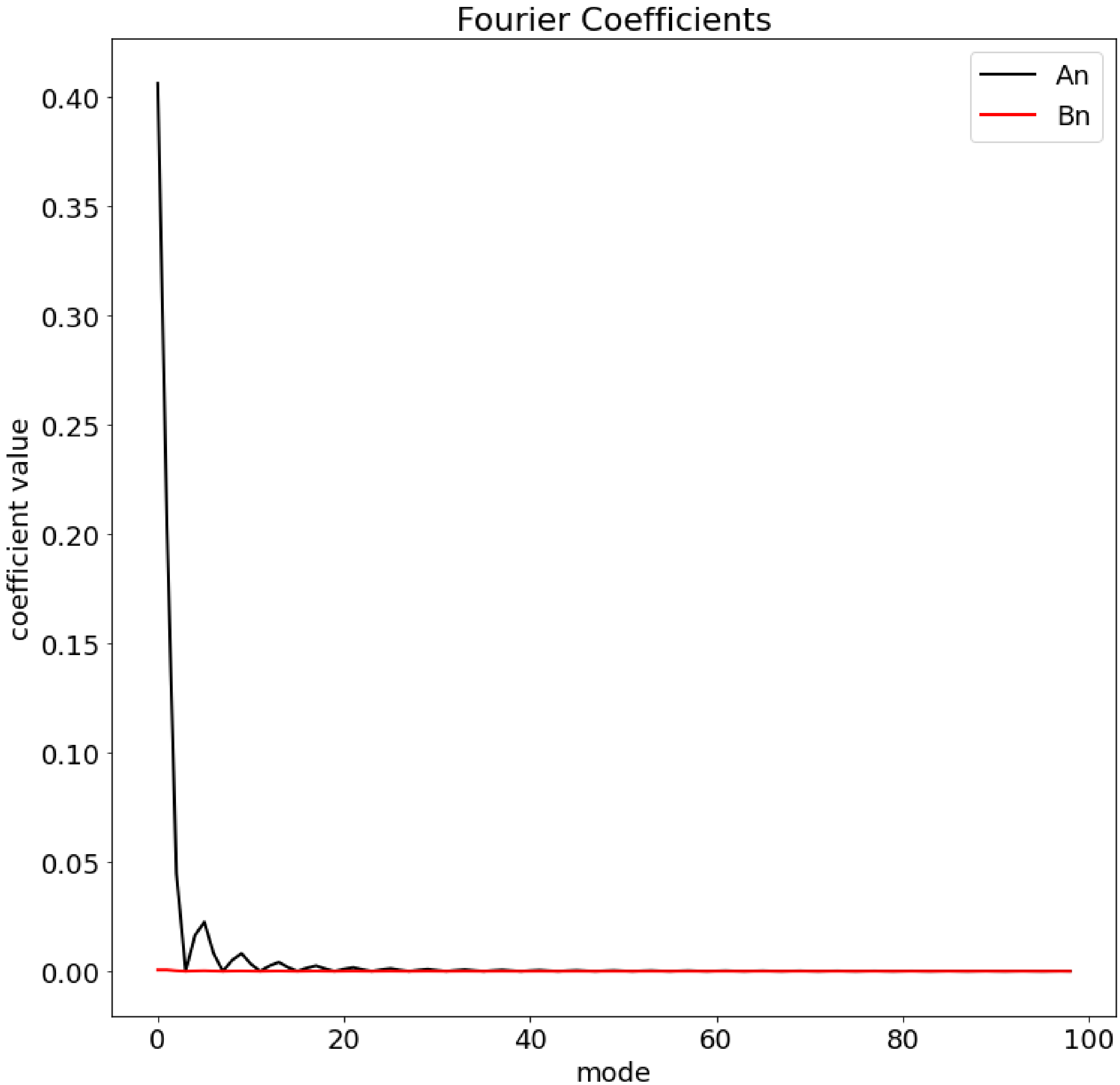

n=3
n=4
n=5
n=6
n=7
n=8
n=9
n=10
f(x)
x
0.8
0.6
0.4
0.2
0.0
−0.2
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


actual
n=10
f(x)
1.0
0.8
0.6
0.4
0.2



Also, plot the mode coefficients an and bn for the first 100 cosine and sine modes

In [3]:
```
plt.rcParams['figure.figsize'] = [12, 12]
plt.rcParams.update({'font.size': 18})

fFS = (A0/2) * np.ones_like(f)
kmax = 100
A = np.zeros(kmax)
B = np.zeros(kmax)

A[0] = A0/2

for k in range(1,kmax):
    A[k] = np.sum(f * np.cos(np.pi*k*x/L)) * dx
    B[k] = np.sum(f * np.sin(np.pi*k*x/L)) * dx
    fFS = fFS + A[k] * np.cos(k*np.pi*x/L) + B[k] * np.sin(k*np.pi*x/L)

# n=1~100
fig, ax = plt.subplots()
ax.plot(np.arange(kmax-1),A[1:],color='k',linewidth=2)
ax.plot(np.arange(kmax-1),B[1:],color='r',linewidth=2)
plt.title('Fourier Coefficients')
plt.legend(['An','Bn'])
plt.ylabel('coefficient value')
plt.xlabel('mode')
plt.show()
```


Fourier Coefficients
An
Bn
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
coefficient value
0
20
40
60
80
100
mode

Exercise 4-2

Load the image recorder.jpg. Convert to grayscale and compress the image using the FFT.

Code reference: [databook_python]

(a)

Design a compression threshold to keep exactly 10% of the original Fourier coefficients. Compute the L2 norm of the error between the new compressed image and the original image. Also compute the L2 norm of the Fourier transformed versions of the compressed and original images.

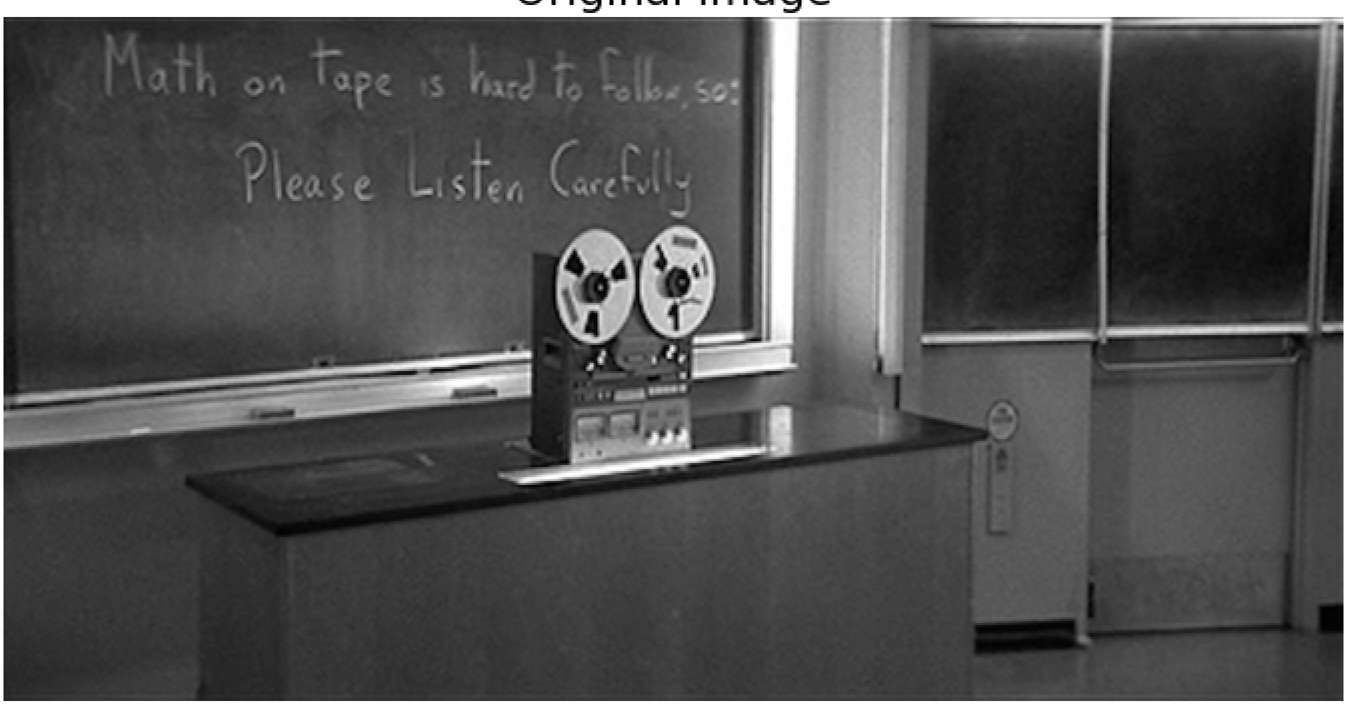
In [1]:
```
from numpy import linalg as LA
from matplotlib.image import imread
import numpy as np
import matplotlib.pyplot as plt
import os
```

Load the originanl image and gray-scaled it

In [2]:
```
plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams.update({'font.size': 18})

A = imread("recorder.jpeg")
B = np.mean(A, -1); # Convert RGB to grayscale
n = B.shape[0]*B.shape[1] # of samples in the original image

plt.figure()
plt.imshow(B,cmap='gray')
plt.axis('off')
plt.title('Original image')
```

Out[2]:
```
Text(0.5, 1.0, 'Original image')
```

Original image

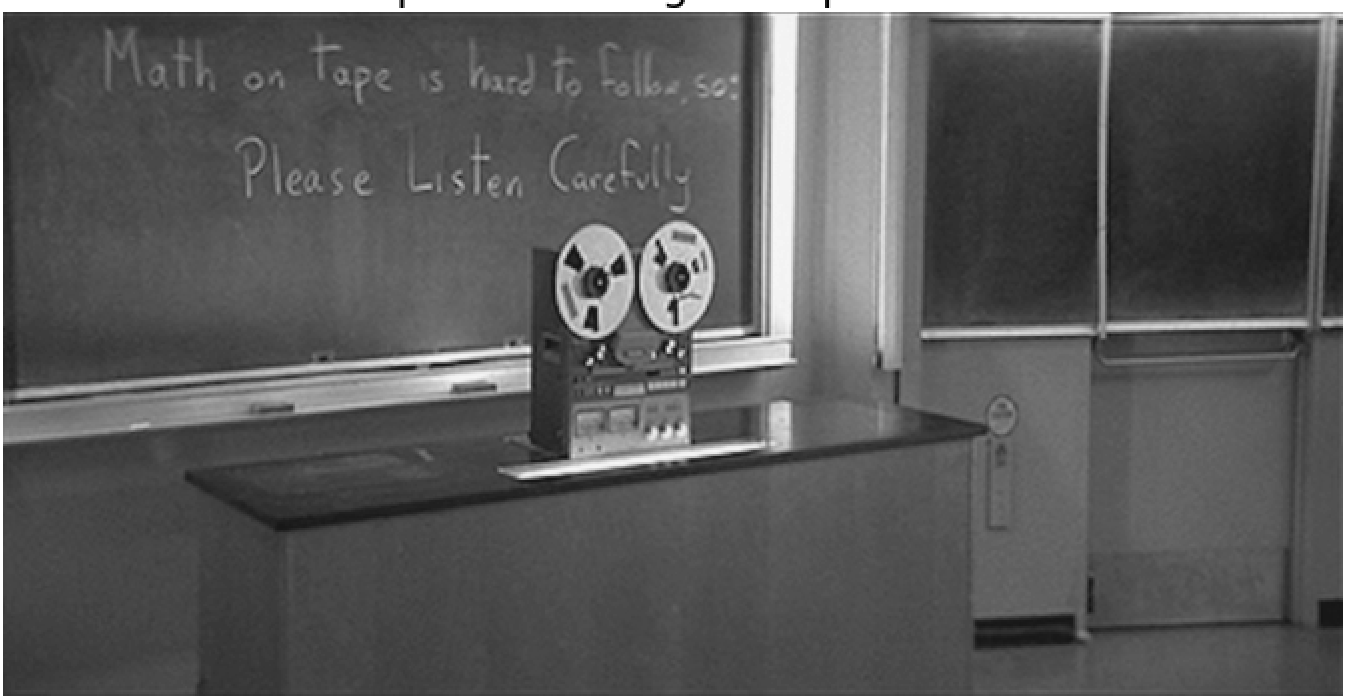


Do 2D FFT and commpressed the image

In [3]:
```
Bt = np.fft.fft2(B)
Btsort = np.sort(np.abs(Bt.reshape(-1))) # sort by magnitude

# Zero out all small coefficients and inverse transform
keep = 0.1

thresh = Btsort[int(np.floor((1-keep)*len(Btsort)))]
ind = np.abs(Bt)>thresh          # Find small indices
Btlow = Bt * ind                 # Threshold small indices
Blow = np.fft.ifft2(Btlow).real  # Compressed image

# Print the commpressed image
plt.figure()
plt.imshow(Blow,cmap='gray')
plt.axis('off')
plt.title('Compressed image: keep = ' + str(keep*100) + '%')
```

Out[3]:
```
Text(0.5, 1.0, 'Compressed image: keep = 10.0%')
```

Compressed image: keep = 10.0%



Compute L2 norm an check the error using L2 norm

In [4]:
```
# L2 norm
NB = LA.norm(B, ord=2) # gray-scaled original img
NBt = LA.norm(Bt, ord=2)/np.sqrt(n)  # FFT img
NBtlow = LA.norm(Btlow, ord=2)/np.sqrt(n)  # compressed FFT img
NBlow = LA.norm(Blow, ord=2)  # compressed img
print('L2 norm of gray-scaled original img:  ', NB)
print('L2 norm of gray-scaled compressed img:', NBlow)
print('L2 norm of FFT img:                   ', NBt)
print('L2 norm of compressed FFT img         ', NBtlow)
print('-----------------------------------------------------------')

image_error = LA.norm((B-Blow), ord=2)/NB
FFT_image_error = LA.norm((Bt-Btlow), ord=2)/np.sqrt(n)/NBt
print('error between the compressed image and the original image:', image_error)
print('error between the FFT compressed image and the FFT image :', FFT_image_error)
```

```
L2 norm of gray-scaled original img:   105526.92014109949
L2 norm of gray-scaled compressed img: 105526.83856644596
L2 norm of FFT img:                    105526.92014109944
L2 norm of compressed FFT img          105526.83856644598
----------------------------------------------------------
error between the compressed image and the original image: 0.003057801349858109
error between the FFT compressed image and the FFT image : 0.003057801349858109
```

(b)

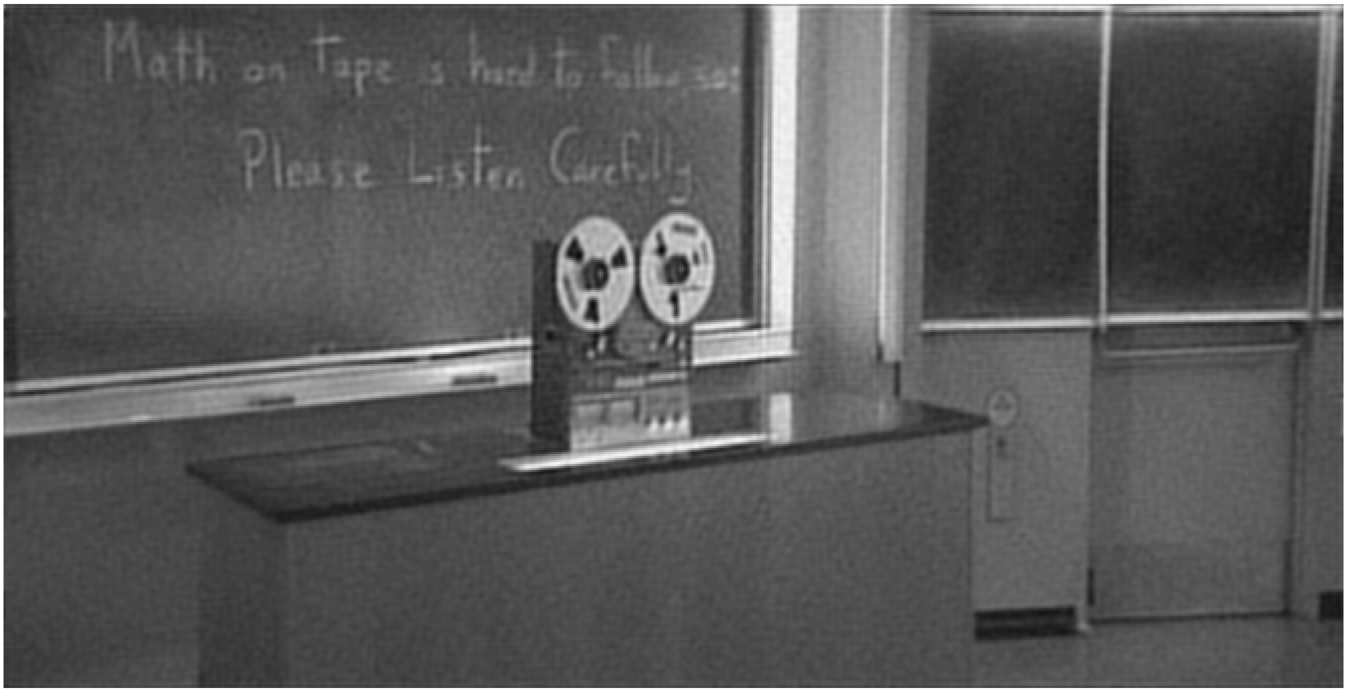
Repeat for a compression that only keeps 1% of the original Fourier coefficients.

In [5]:

```python
Bt = np.fft.fft2(B)
Btsort = np.sort(np.abs(Bt.reshape(-1))) # sort by magnitude

# Zero out all small coefficients and inverse transform
keep = 0.01

thresh = Btsort[int(np.floor((1-keep)*len(Btsort)))]
ind = np.abs(Bt)>thresh          # Find small indices
Btlow = Bt * ind                 # Threshold small indices
Blow = np.fft.ifft2(Btlow).real  # Compressed image

# Print the commpressed image
plt.figure()
plt.imshow(Blow,cmap='gray')
plt.axis('off')
plt.title('Compressed image: keep = ' + str(keep*100) + '%')

# L2 norm
NB = LA.norm(B, ord=2) # gray-scaled original img
NBt = LA.norm(Bt, ord=2)/np.sqrt(n) # FFT img
NBtlow = LA.norm(Btlow, ord=2)/np.sqrt(n) # compressed FFT img
NBlow = LA.norm(Blow, ord=2) # compressed img
print('L2 norm of gray-scaled original img:  ', NB)
print('L2 norm of gray-scaled compressed img:', NBlow)
print('L2 norm of FFT img:                   ', NBt)
print('L2 norm of compressed FFT img         ', NBtlow)
print('----------------------------------------------------------')

image_error = LA.norm((B-Blow), ord=2)/NB
FFT_image_error = LA.norm((Bt-Btlow), ord=2)/np.sqrt(n)/NBt
print('error between the compressed image and the original image:', image_error)
print('error between the FFT compressed image and the FFT image :', FFT_image_error)
```

```
L2 norm of gray-scaled original img:   105526.92014109949
L2 norm of gray-scaled compressed img: 105517.44733398104
L2 norm of FFT img:                    105526.92014109944
L2 norm of compressed FFT img          105517.44733398105
----------------------------------------------------------
error between the compressed image and the original image: 0.014370021272522945
error between the FFT compressed image and the FFT image : 0.014370021272522964
```

Compressed image: keep = 1.0%




```
In [ ]:
```