Signal & System 2022 期末考 手做答案可以手寫或打字拍照。 程式考題寫出程式碼以及列印執行結果。 一並彙整為一個 pdf 檔上傳。 將請助教開一個 E-Learning 作業於考試當天(6 月 17 日)限時(3 小時)之內上傳。 正式考試為第 17 周上課時間(6 月 17 日 15:00~18:00)

Question (1)

1.1) What is DFT (Discrete Fourier Transform), and IDFT (Inverse Discrete Fourier Transform), write down their mathematical definition. if you get the answer from internet, please keep the address of the website.

答案

定義 [編輯]

對於N點序列$\{x[n]\}_{0\leq n<N}$，它的離散傅立葉轉換（DFT）為

$$\hat{x}[k] = \sum_{n=0}^{N-1} e^{-i\frac{2\pi}{N}nk} x[n] \qquad k = 0, 1, \ldots, N-1.$$

其中e是自然對數的底數，i是虛數單位。通常以符號$\mathcal{F}$表示這一轉換，即

$$\hat{x} = \mathcal{F}x$$

離散傅立葉轉換的逆轉換（IDFT）為：

$$x\,[n] = \frac{1}{N}\sum_{k=0}^{N-1} e^{i\frac{2\pi}{N}nk} \hat{x}[k] \qquad n = 0, 1, \ldots, N-1.$$

可以記為：

$$x = \mathcal{F}^{-1}\hat{x}$$

實際上，DFT和IDFT轉換式中和式前面的歸一化係數並不重要。在上面的定義中，DFT和IDFT前的係數分別為1和$\frac{1}{N}$。有時會將這兩個係數都改成$\frac{1}{\sqrt{N}}$。

https://zh.wikipedia.org/zh-tw/%E7%A6%BB%E6%95%A3%E5%82%85%E9%87%8C%E5%8F%B6%E5%8F%98%E6%8D%A2

1.2) I have written a function called myDFT(), it can compute the DFT (discrete Fourier Transform) of any signal x[n].
π= 3.1415926 def myDFT(x): N= len(x) n= np.arange(N) k= np.arange(N) kn= np.outer(k, n) M= np.exp(-1j * 2 * π * kn) X= M.dot(x) return X For the following

signal x, using myDFT(x) to compute its DFT. x = [1, 1, 1, 1] X= myDFT(x) = ? Keep your answer only in 3-digit precision. (四捨五入至小數點後 3 位。)

答案

```
π = 3.1415926
def myDFT(x):
    N= len(x)
    n= np.arange(N)
    k= np.arange(N)
    kn= np.outer(k, n)
    M= np.exp(-1j * 2 * π * kn)
    X= M.dot(x)
    return X
x = [1, 1, 1, 1]
X= myDFT(x)
X
array([4.+0.00000000e+00j, 4.+6.43077516e-07j, 4.+1.28615503e-06j,
       4.+1.92923255e-06j])
```

四捨五入後

```
[4.+0.j 4.+0.j 4.+0.j 4.+0.j]
```

```
In [39]: print(np.round(X,2))
         [4.+0.j 4.+0.j 4.+0.j 4.+0.j]
```

1.3) Using numpy.fft.fft() to do the DFT again, what is the answer?

答案

```
x = np.array([1,1,1,1])
X = np.fft.fft(x)
X
array([4.+0.j, 0.+0.j, 0.+0.j, 0.+0.j])
```

```
In [19]: x = np.array([1,1,1,1])
         X = np.fft.fft(x)
         X
Out[19]: array([4.+0.j, 0.+0.j, 0.+0.j, 0.+0.j])
```

1.4) If the answers are not the same, what is their mean squared error ?

答案

In [43]:
```
target = [4.+0.j, 0.+0.j, 0.+0.j, 0.+0.j]
prediction = [4.+0.j, 4.+0.j, 4.+0.j, 4.+0.j]
error = []
for i in range(len(target)):
    error.append(target[i] - prediction[i])

#print("Errors: ",error)
#print(error)

squaredError = []
absError = []

for val in error:
    squaredError.append(val * val)#target-prediction之差平方
    absError.append(abs(val))#誤差絕對值

#print("Square Error: ",squaredError)
#print("Absolute Value of Error: ",absError)
print("MSE = ",sum(squaredError) / len(squaredError))#均方誤差MSE
```

```
MSE =  (12+0j)
```

```
MSE =  (12+0j)
```

1.5) Could you debug myDFT() such that it becomes correct compared with numpy.fft.fft()?

答案

```
def myDFT(x, inverst=False):
    if not inverst:
        return np.fft.fft(x)
    return np.fft.ifft(x)
```

In [30]:
```
def myDFT(x, inverst=False):
  if not inverst:
    return np.fft.fft(x)
  return np.fft.ifft(x)
```

In [31]:
```
x=[1,1,1,1]
X=myDFT(x)
X
```

Out[31]:
```
array([4.+0.j, 0.+0.j, 0.+0.j, 0.+0.j])
```

1.6) After debugging, please verify your answer with that computed from numpy.fft.fft() by comparing their mean squared error.

答案

```
x=[1,1,1,1]
X=myDFT(x)
X
```

```
array([4.+0.j, 0.+0.j, 0.+0.j, 0.+0.j])
```

```
target = [4.+0.j, 0.+0.j, 0.+0.j, 0.+0.j]
prediction = [4.+0.j, 0.+0.j, 0.+0.j, 0.+0.j]
error = []
for i in range(len(target)):
    error.append(target[i] - prediction[i])

#print("Errors: ",error)
#print(error)

squaredError = []
absError = []

for val in error:
    squaredError.append(val * val)#target-prediction之差平方
    absError.append(abs(val))#誤差絕對值

#print("Square Error: ",squaredError)
#print("Absolute Value of Error: ",absError)
print("MSE = ",sum(squaredError) / len(squaredError))#均方誤差MSE
```

```
MSE =  0j
```

MSE=0j

1.7) How to modify myDFT(x) to become myIDFT(X), such that it can compute the Inverse DFT? Verify your answer by setting X= [4, 0, 0, 0] and compute the IDFT(X)=?

答案

```
def myDFT(x, inverst=False):
    if not inverst:
        return np.fft.fft(x)
    return np.fft.ifft(x)
x=[4,0,0,0]
X=myDFT(x,True)
```

```
array([1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j])
```

```
In [21]: def myDFT(x, inverst=False):
             if not inverst:
                 return np.fft.fft(x)
             return np.fft.ifft(x)

In [23]: x=[4,0,0,0]
         X=myDFT(x,True)
         X

Out[23]: array([1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j])
```

Question (2)

(2.1)

What is discrete convolution of 2 signals, x[n] and y[n]? Please write down their mathematical

definition. If you get the answer from internet, please keep the address of the website.

答案

- 作法：利用摺積的定義

$$y[n] = f[n] * g[n] = \sum_{m=0}^{M-1} f[n-m]g[m]$$

那我們這邊計算時

公式的 y[n]用 A[n]代入(表示答案)

f[n]用此題的 x[n]代入

g[n]用此題的 y[n]代入

https://zh.wikipedia.org/wiki/%E5%8D%B7%E7%A7%AF#%E7%A6%BB%E6%95%A3%E5%8D%B7%E7%A7%AF

(2.2)

if x= [1, 2, 3, 4, 5, 6, 7, 8], y= [1, 1, 1, 1], what is their convolution z = x*y ?

please compute the answer by hands (not using computer, you should write down the process.)

答案

from thinkdsp import Wave

w1 = Wave(np.array([1,2,3,4,5,6,7,8]))

w2 = Wave(np.array([1,1,1,1]))

w1.convolve(w2).ys

```
array([ 1,  3,  6, 10, 14, 18, 22, 26, 21, 15,  8])
```

```
In [33]: from thinkdsp import Wave
         w1 = Wave(np.array([1,2,3,4,5,6,7,8]))
         w2 = Wave(np.array([1,1,1,1]))
         w1.convolve(w2).ys

Out[33]: array([ 1,  3,  6, 10, 14, 18, 22, 26, 21, 15,  8])
```

(2.3)

write a function called

myCONV()

which can compute the convolution of two signals. please fill the code in
z= ????????,

such that it can run correctly, and write out the result of the program.

```
def myCONV(x,y):

    y= np.append(y, [0] * (len(x)-1)) # zero padding

    m= np.arange(len(y))

    zL= []
    for n in range(len(y)):

        z= ????????

        zL += [z]
    zL= np.array(zL)

    return zL

myCONV(x,y)
```

答案

```
def myCONV(x,y):
    y= np.append(y, [0] * (len(x)-1)) # zero padding
    m= np.arange(len(y))
    zL= []
    for n in range(len(y)):
        z=x[:-n]*y[m:]
        zL += [z]
    zL= np.array(zL)
    return zL

myCONV(x,y)
```

(2.4)
The following is a better implimentation, please fill the code in,
z= ????????,
and check out the answer.

```
def myCONV2(x,y):

    x=    np.array(x)
    y=    np.append(y, [0]*(len(x)-1)) # zero padding
    m= np.arange(len(x)).reshape( 1,-1) # row vector
    n= np.arange(len(y)).reshape(-1, 1) # column vector



    z= ????????

    return z

myCONV2(x,y)
```

(2.5) if the length of x[n] is Nx, the length of y[n] is Ny, what is the length of their convolution

z[n] = x[n] * y[n]

答案

Nx+Ny-1

(2.6) Convolution theorem describes the relationship of the spectrums for signals and theirconvolution. If

X[k] = DFT(x[n]), Y [k] = DFT(y[n]), Z[k] = DFT(z[n])

what is the relationship between?

都做了 DFT 轉換

(2.7) Verify the convolution theorem by using myDFT() and myCONV() , you can use x= [1, 2,3, 4, 5, 6, 7, 8], y= [1, 1, 1, 1] as the above. And check whether the relationship described in (2.6) really hold or not? You may need do "zero padding" to the end of x[n] and y[n], such that x[n], y[n] and z[n] all have the same length to do the DFT for comparison.

Question (3)

(3.1)

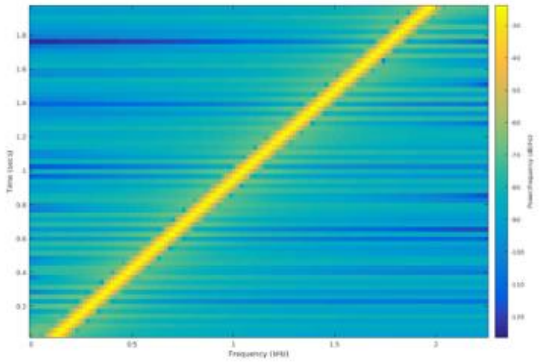
What is a linear chirp signal? Write down its mathematical function.

有種啁啾的瞬時頻率 $f(t)$ 呈線性變化，即有

$f(t) = f_0 + ct$，其中

$$c = \frac{f_1 - f_0}{T}$$

為常值。積分計算出，相位為 t 的二次函數：



線性啁啾的時頻譜，顯示頻率隨時間變化為線性，由0至7 kHz，每2.3秒重覆。圖中，色彩的鮮豔度表示訊號在指定時間和頻率所含的能量。

$$\begin{aligned}\phi(t) &= \phi_0 + 2\pi \int_0^t f(\tau)\,\mathrm{d}\tau \\ &= \phi_0 + 2\pi \int_0^t (c\tau + f_0)\,\mathrm{d}\tau \\ &= \phi_0 + 2\pi \left(\frac{c}{2}t^2 + f_0 t\right),\end{aligned}$$

從而信號在時域表示為

$$x(t) = A\cos\left(\phi_0 + 2\pi\left(\frac{c}{2}t^2 + f_0 t\right)\right).$$

此種訊號又稱**二次相位訊號**。[2]

https://zh.wikipedia.org/wiki/%E5%95%81%E5%95%BE

(3.2)

Write a function called myCHIRP(), it can generate a chirp signal.

input:

a time sequence: ts= [0, 0.001, 0.002, ..., T] in (sec)

a start frequency f0 (Hz),

a ending frequency f1 (Hz).

output:

the chirp sequence: ys= myCHIRP(ts)

```
def myCHIRP(ts, f0=100, f1=400):

    #
    # your code ...
    #
```

```
    return ys

# 1. generate a time sequence from 0 sec to 10 sec
# 2. test your function.

from thinkdsp import Chirp
def myChirp(ts,f0=100,f1=400):
    signal = Chirp(f0, f1)
    ys = signal.make_wave(duration=10)
    ys.make_audio()
    ys.segment(start=0, duration=0.1).plot()
    decorate(xlabel='Time (s)')
    return ys
T=10
ts=np.arange(0,T,0.001)
myChirp(ts)
```

```
In [78]: T=10

In [79]: ts=np.arange(0,T,0.001)

In [80]: myChirp(ts)

Out[80]: <thinkdsp.Wave at 0x21f18b86a00>
```