Digital Signal Processing Laboratory
Department of Electrical & Computer Engineering

Ψηφιακή Επεξεργασία Σήματος

Εργαστήριο 3

Ψηφιακά Φίλτρα

Ηράκλειο 2025

Δρ. Κωνσταντίνος Καραμπίδης

Γραμμική Συνέλιξη (Linear Convolution)

Με το **άθροισμα της γραμμικής συνέλιξης** μπορούμε να βρούμε την απόκριση ενός συστήματος διακριτού χρόνου για είσοδο x(n), αν γνωρίζουμε την κρουστική του απόκριση h(n).

- Η έξοδος y(n) του συστήματος θα ισούται με την συνέλιξη της εισόδου x(n) και της κρουστικής h(n) του συστήματος και ορίζεται ως εξής:

$$y(n) = x(n) * h(n)$$

όπου το σύμβολο * αντιστοιχεί στο τελεστή της συνέλιξης.

Το Άθροισμα της Συνέλιξης

Το Άθροισμα της Συνέλιξης μπορεί να γραφεί και ως εξής:

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Όταν το σύστημα και η ακολουθία εισόδου είναι αιτιατά (δηλαδή h(n)=x(n)=0, για κάθε n<0 και έχουν μήκος δείγματος M και N αντίστοιχα) τότε αλλάζουν τα όρια του αθροίσματος και η εξίσωση της συνέλιξης παίρνει τη μορφή:

$$y(n) = \sum_{k=0}^{n} x(k)h(n-k) \qquad \text{Για } n=0,\ldots,N+M-2$$

Επομένως η απόκριση y έχει μέγεθος N+M-1

Η συνάρτηση convolve της Python

Για τον υπολογισμό του αθροίσματος της συνέλιξης μπορείτε να χρησιμοποιήσετε την συνάρτηση της Python.

```
import numpy as np
np.convolve (A, B)
np.convolve (A, B, SHAPE)
    Convolve two vectors A and B.
```

np.convolve (A, B, mode='full') Return the full convolution. The result is a vector with length equal to 'length (A) + length (B) - 1'.

When A and B are the coefficient vectors of two polynomials, the convolution represents the coefficient vector of the product polynomial.

Παράδειγμα

Έστω ότι έχουμε το παρακάτω σήμα εισόδου x(n) και την κρουστική του απόκριση h(n). Να υπολογίσετε την συνέλιξη χρησιμοποιώντας την συνάρτηση της Python.

$x(n) = 3n^3 + n^2 + 2n + 1$
$h(n) = n^2 + 2n + 3$

Αρχικά βρίσκουμε τα διανύσματα a και b όπου θα πρέπει να εισάγουμε ως ορίσματα στην συνάρτηση convolve.

```
a = [3, 1, 2, 1];
b = [1, 2, 3];
y = np.convolve(a, b)
```

Output

y = 3 7 13 8 8 3

Παράδειγμα

Πραγματοποιήστε τον υπολογισμό της συνέλιξης για ίδια σήματα x(n) και h(n) στο χαρτί.

$y(n) = x(n)*h(n) =$

$(3n^3 + n^2 + 2n + 1)(n^2 + 2n + 3) = 3n^5 + 6n^4 + 9n^3 + n^4 + 2n^3 + 3n^2 + 2n^3 + 4n^2 + 6n + n^2 + 2n + 3 = \mathbf{3}n^5 + \mathbf{7}n^4 + \mathbf{13}n^3 + \mathbf{8}n^2 + \mathbf{8}n + \mathbf{3}$

Κυκλική Συνέλιξη (Circular Convolution)

Η κυκλική συνέλιξη χρησιμοποιείται συνήθως στο πλαίσιο περιοδικών σημάτων διακριτού χρόνου και του διακριτού μετασχηματισμού Fourier (DFT). Περιλαμβάνει την άθροιση σε μια κυκλική ή περιοδική μετατόπιση ενός από τα σήματα.

Για δύο ακολουθίες διακριτού χρόνου $x[n]$ και $h[n]$, και τα δύο μήκους N, η κυκλική συνέλιξη $y[n]$ ορίζεται ως εξής:

$$y[n] = (x \circledast h)[n] = \sum_{k=0}^{N-1} x[k]h[(n-k) \mod N]$$

Κυκλική Συνέλιξη (Circular Convolution)

Βήματα κυκλικής συνέλιξης

Εξασφάλιση ίδιου μήκους: Βεβαιωθείτε ότι και οι δύο ακολουθίες $x[n]$ και $h[n]$ έχουν το ίδιο μήκος N. Εάν όχι, συμπληρώστε τη μικρότερη ακολουθία με μηδενικά.

Κυκλική μετατόπιση: Κυκλική μετατόπιση (προς τα δεξιά ή τα αριστερά) μιας ακολουθίας και πολλαπλασιασμός της με την άλλη ακολουθία (element wise).

Άθροισμα γινομένων: Για κάθε μετατόπιση, αθροίστε τα γινόμενα για να λάβετε την ακολουθία εξόδου.

Κυκλική Συνέλιξη (Circular Convolution)

Παράδειγμα

Ας θεωρήσουμε δύο ακολουθίες:

$x[n]$={1,2,3}

$h[n]$={4,5,6}

Για να υπολογίσουμε την κυκλική συνέλιξη ακολουθούμε τα παρακάτω βήματα:

Εξασφάλιση ίδιου μήκους (Zero Padding): δεν είναι απαραίτητο εδώ, καθώς και οι δύο ακολουθίες έχουν μήκος 3:

$x[n]$={1,2,3}

$h[n]$={4,5,6}

Κυκλική Συνέλιξη (Circular Convolution)

Κυκλική μετατόπιση (Circular shift):

1) Αρχική ακολουθία h[0]= [4,5,6]
2) Μία θέση δεξιά μετατόπιση: h[1]=[6,4,5]
3) Δύο θέσεις δεξιά μετατόπιση: h[2]=[5,6,4]

Έπειτα υπολογίζουμε το άθροισμα των γινομένων για κάθε μετατόπιση.

- $y[0] = x[0]h_0[0] + x[1]h_0[1] + x[2]h_0[2] = 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 31$
- $y[1] = x[0]h_1[0] + x[1]h_1[1] + x[2]h_1[2] = 1 \cdot 6 + 2 \cdot 4 + 3 \cdot 5 = 29$
- $y[2] = x[0]h_2[0] + x[1]h_2[1] + x[2]h_2[2] = 1 \cdot 5 + 2 \cdot 6 + 3 \cdot 4 = 29$

Επομένως το αποτέλεσμα της κυκλικής συνέλιξης θα είναι y[n]=[31,29,29]

Κυκλική Συνέλιξη (Circular Convolution)

```
# Circular shift visualization
import numpy as np
# Define the signal
h = np.array([4, 5, 6])

# Circular right shift by 1
h_right_1 = np.roll(h, 1)
print("Right Shift by 1:", h_right_1)

# Circular right shift by 2
h_right_2 = np.roll(h, 2)
print("Right Shift by 2:", h_right_2)

# Circular left shift by 1
h_left_1 = np.roll(h, -1)
print("Left Shift by 1:", h_left_1)

# Circular left shift by 2
h_left_2 = np.roll(h, -2)
print("Left Shift by 2:", h_left_2)
```

Κυκλική Συνέλιξη (Circular Convolution)

1ος τρόπος

```
import numpy as np

def circular_convolution(x, h):
    N = len(x)
    y = np.zeros(N)
    for n in range(N):
        for k in range(N):
            y[n] += x[k] * h[(n - k) % N]

    return y

# Define the signals
x = np.array([1, 2, 3])
h = np.array([4, 5, 6])
# Compute the circular convolution
y = circular_convolution(x, h)

print("Circular Convolution Result:", y)
```

Κυκλική Συνέλιξη (Circular Convolution)

2ος τρόπος

```
import numpy as np
# Define the signals
x = np.array([1, 2, 3])
h = np.array([4, 5, 6])
# Compute the circular convolution using numpy
y = np.fft.ifft(np.fft.fft(x) * np.fft.fft(h))
print("Circular Convolution Result:", np.real(y))
```

Κυκλική Συνέλιξη (Circular Convolution)

y = np.fft.ifft(np.fft.fft(x) * np.fft.fft(h)) : Υπολογίζει την κυκλική συνέλιξη των x και h χρησιμοποιώντας τον Fast Fourier Transform (FFT). Η συνάρτηση np.fft.fft υπολογίζει τον διακριτό μετασχηματισμό Fourier (DFT) για τις ακολουθίες εισόδου x και h.

np.fft.fft(x): Υπολογίζει τον DFT της ακολουθίας x.
np.fft.fft(h): Υπολογίζει τον DFT της ακολουθίας h.
np.fft.fft(x) * np.fft.fft(h): Πολλαπλασιάζει τα αποτελέσματα των δύο DFT στοιχείο προς στοιχείο (element-wise). Αυτό το βήμα αντιστοιχεί στον πολλαπλασιασμό στο πεδίο της συχνότητας, ο οποίος είναι ισοδύναμος με τη συνέλιξη στο πεδίο του χρόνου.
np.fft.ifft(...): Υπολογίζει τον αντίστροφο DFT του γινομένου των DFTs για να λάβουμε το αποτέλεσμα της κυκλικής συνέλιξης στο πεδίο του χρόνου. Επιστρέφει έναν μιγαδικό πίνακα όπου το πραγματικό μέρος αντιπροσωπεύει το αποτέλεσμα.

Απόκριση Συχνότητας

Η απόκριση συχνότητας χαρακτηρίζει τον τρόπο με τον οποίο το πλάτος και η φάση ενός σήματος εξόδου μεταβάλλονται με τη συχνότητα του σήματος εισόδου. Δείχνει πώς ενισχύονται ή εξασθενούν οι διάφορες συνιστώσες συχνότητας από το σύστημα και πώς μετατοπίζονται οι φάσεις τους.

Βασικά συστατικά της απόκρισης συχνότητας

- **Magnitude response (**Απόκριση μεγέθους): $|H(f)|$
  Δείχνει το πλάτος του σήματος εξόδου ως συνάρτηση της συχνότητας.
- **Phase Response** (Απόκριση φάσης): $\angle H(f)$

Απόκριση Συχνότητας

Για ένα διακριτό σύστημα με κρουστική απόκριση h(n), η απόκριση συχνότητας του θα είναι ο μετασχηματισμός Fourier της κρουστικής απόκρισης.

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}$$

Η $H(e^{j\omega})$ είναι στη γενική περίπτωση, μιγαδικός αριθμός και εξαρτάται από τη συχνότητα ω της μιγαδικής εκθετικής συνάρτησης.
Επομένως, χαρακτηρίζει πλήρως την y(n) από το πεδίο ορισμού του χρόνου σε αυτό των συχνοτήτων.

Απόκριση Συχνότητας - Παράδειγμα

Θεωρήστε ένα χαμηλοπερατό φίλτρο:

- **Magnitude Response**: Επιτρέπει τη διέλευση συνιστωσών χαμηλών συχνοτήτων, ενώ εξασθενεί τις συνιστώσες υψηλών συχνοτήτων.
- **Phase Response**: Δείχνει πώς το φίλτρο επηρεάζει τη φάση των διαφόρων συνιστωσών συχνότητας.

Διάγραμμα Bode

Ένα διάγραμμα Bode είναι μια γραφική αναπαράσταση της απόκρισης συχνότητας ενός συστήματος. Αποτελείται από δύο διαγράμματα:
Magnitude Plot: Δείχνει την απόκριση μεγέθους $|H(f)|$ έναντι της συχνότητας (συχνά σε λογαριθμική κλίμακα).

Διάγραμμα φάσης: Δείχνει την απόκριση φάσης $\angle H(f)$ συναρτήσει της συχνότητας.

Η Συνάρτηση freqz της Python

Η συνάρτηση υπολογισμού την απόκρισης συχνότητας της Python είναι η freqz.

```
import scipy
 H, W = scipy.signal.freqz(B, A,N)
```

Return the complex frequency response H of the rational IIR filter whose numerator and denominator coefficients are B and A, respectively.
If N is omitted, a value of 512 is assumed. For fastest computation, N should factor into a small number of small primes.

Άσκηση 1

Για το σύστημα με εξίσωση διαφορών:
$y(n) = 1.1y(n-1)-0.5y(n-2)-0.3y(n-4)+0.5x(n)-0.2x(n-1)$
να υπολογίσετε:

a) Την κρουστική απόκριση για το διάστημα [0,10].
b) Την απόκριση του συστήματος για την παρακάτω είσοδο χρησιμοποιώντας τη συνάρτηση convolve.
x = [5 1 1 1 0 0 1 1 1 0]
c) Την απόκριση για την παραπάνω είσοδο χρησιμοποιώντας τη συνάρτηση lfilter.

Λύση

Από την εξίσωση διαφορών παρατηρούμε ότι για να υπολογίσουμε την κρουστική απόκριση με δικό μας κώδικα θα πρέπει να υπολογίσουμε τις αρχικές συνθήκες για n=[0,4].

$h(n) = 1.1h(n-1)-0.5h(n-2)-0.3h(n-4)+0.5\delta(n)-0.2\delta(n-1)$

$h(0) = 0 - 0 - 0 + 0.5 - 0 = 0.5$

$h(1) = 1.1(0.5) - 0 - 0 + 0 - 0.2 = 0.55 - 0.2 = 0.35$

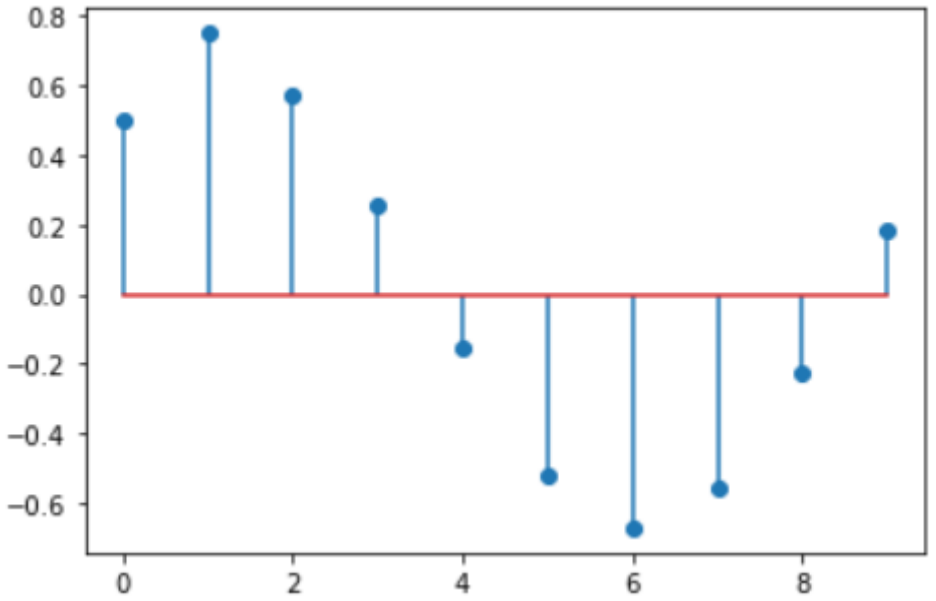
$h(2) = 1.1(0.35) - 0.5(0.5) - 0 + 0 - 0 = 0.385 - 0.25 = 0.135$

$h(3) = 1.1(0.135) - 0.5(0.35) - 0 + 0 - 0 = 0.1485 - 0.175 = -0.0265$

$h(4) = 1.1(-0.0265) - 0.5(0.135) - 0.3(0.5) + 0 - 0 = -0.0215 - 0.0675 - 0.15$
$= -0.24665$

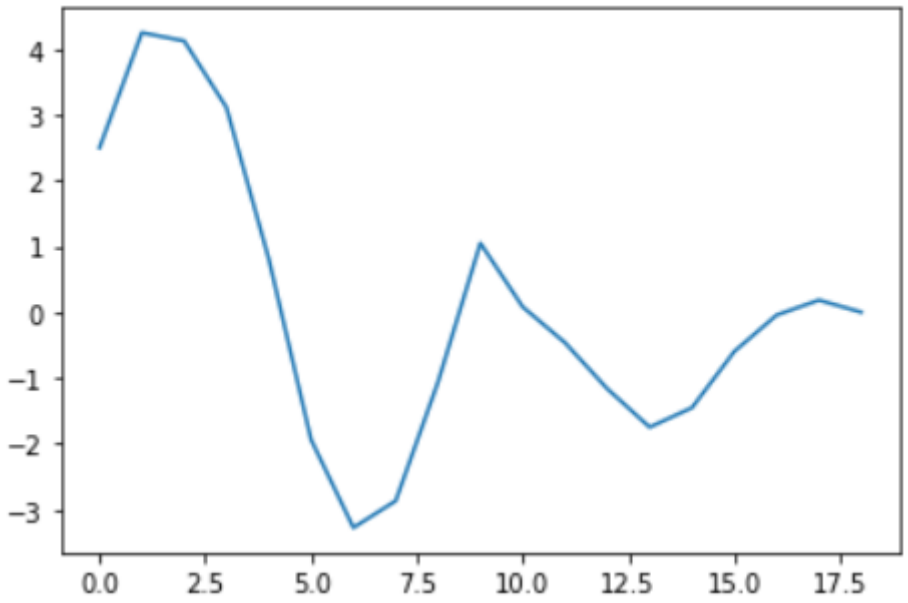
Χρησιμοποιώντας την συνάρτηση lfilter της Python

```
delta = lambda n: n==0
a = [1, -1.1, 0.5, 0, 0.3];
b = [0.5, -0.2, 0, 0, 0];
x=np.zeros(10)
for n in range(0,10,1):
    temp=int(delta(n))
    x[n]=temp
n=range(0,10,1)
h= scipy.signal.lfilter(b,a,x)
plt.stem(n,h)
```



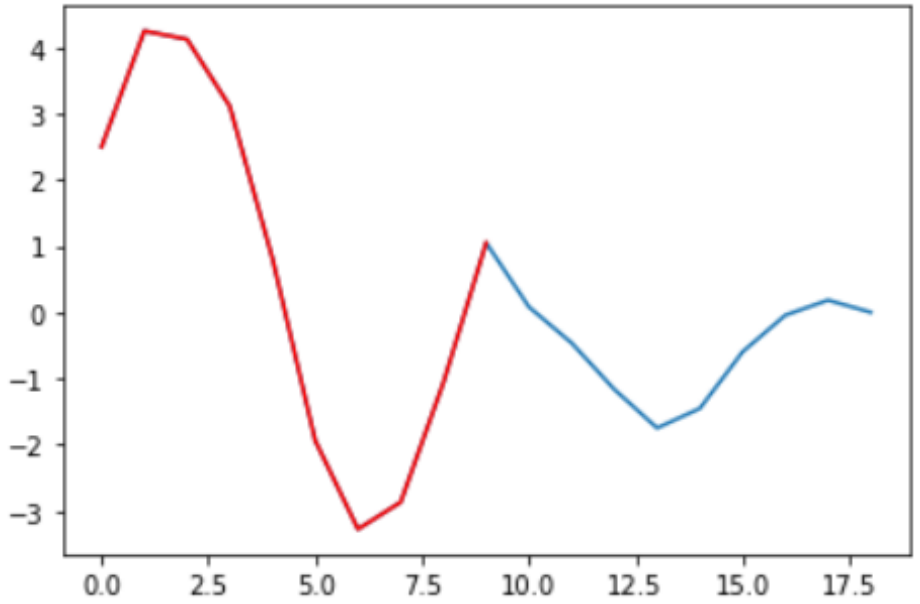
Για τον υπολογισμό της απόκρισης (χρησιμοποιώντας την συνάρτηση convolve) προσθέτουμε στο τέλος του προηγούμενου κώδικα:

```
x=[5, 1, 1, 1, 0, 0, 1, 1, 1, 0];

y=np.convolve(x, h);

plt.plot(y)

plt.show()
```



Για τον υπολογισμό της απόκρισης (χρησιμοποιώντας την συνάρτηση **lfilter**) προσθέτουμε στο τέλος του προηγούμενου κώδικα:

```
y2=scipy.signal.lfilter(b, a, x);
plt.plot(y);
plt.plot(y2,'r');
```

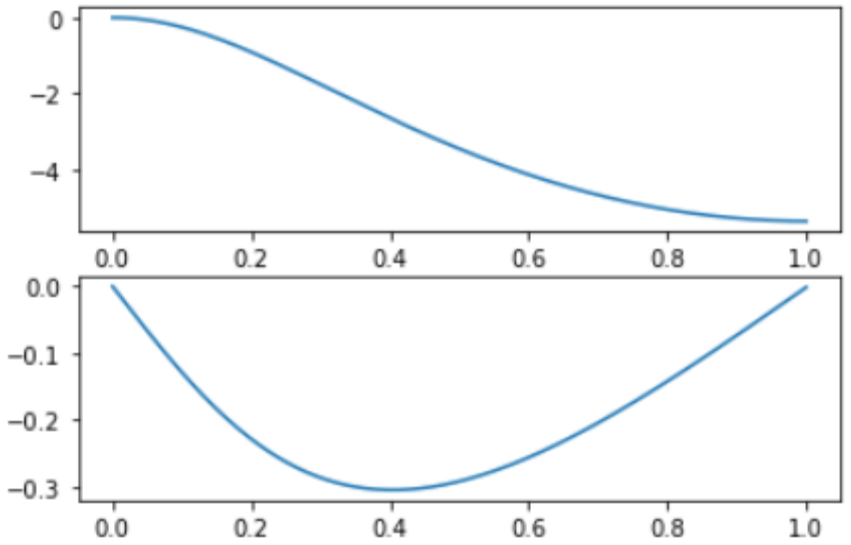


Άσκηση 2

Να υπολογιστεί και να παρασταθεί γραφικά η απόκριση συχνότητας (μέτρο και φάση) του παρακάτω φίλτρου.
y(n) = 0.3y(n-1) + 0.7x(n) για n>=0

Λύση

```
#y(n)-0.3y(n-1) = 0.7x(n)
a=[1, -0.3]
b=[0.7, 0]
w,h= scipy.signal.freqz(b,a)
angles = np.unwrap(np.angle(h))


plt.subplot(2, 1, 1)
plt.plot(w/max(w), 20 * np.log10(abs(h)))
plt.subplot(2, 1, 2)
plt.plot(w/max(w), angles)
```

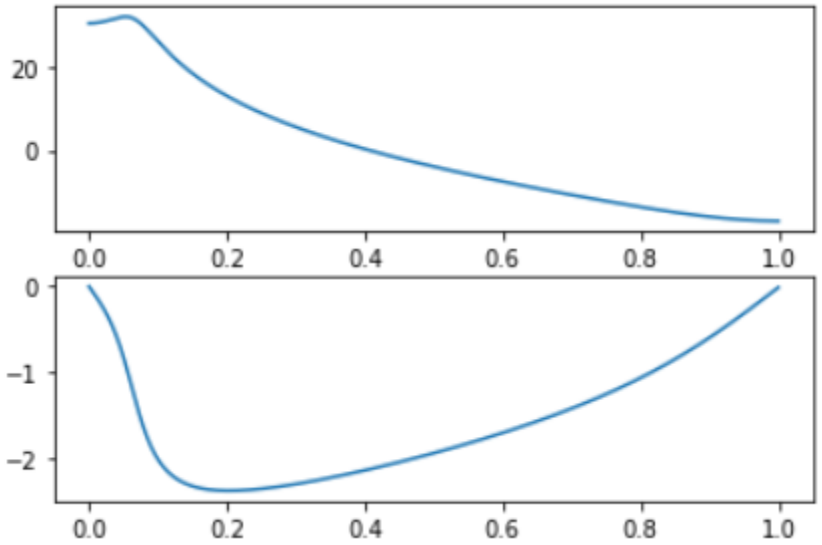


Άσκηση 3

Να υπολογίσετε και να παραστήσετε γραφικά την απόκριση συχνότητας (μέτρο και φάση) με 512 δείγματα στο άνω ήμισυ του μοναδιαίου κύκλου (ω=0-π) του φίλτρου με την παρακάτω συνάρτηση μεταφοράς.

$$H(z) = \frac{1 + 0.5z^{-1}}{1 - 1.8\cos(\frac{\pi}{16})z^{-1} + 0.81z^{-2}}$$

```
a=[1,-1.8*np.cos(np.pi/16), 0.81]
b=[1, 0.5, 0]
w, h = scipy.signal.freqz(b,a)
angles = np.unwrap(np.angle(h))

plt.subplot(2, 1, 1)
plt.plot(w/max(w), 20 * np.log10(abs(h)))
plt.subplot(2, 1, 2)
plt.plot(w/max(w), angles)
```



Άσκηση 4

Ας υποθέσουμε ότι έχουμε ένα φίλτρο FIR με την ακόλουθη κρουστική απόκριση: h[n]={0.25,0.5,0.25}
Υπολογίστε και σχεδιάστε την απόκριση συχνότητας.

Λύση

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

# Define the impulse response of the FIR filter
h = np.array([0.25, 0.5, 0.25])

# Compute the frequency response. The w array contains the frequencies (in radians per sample), and the H array contains the corresponding frequency response values.
w, H = freqz(h)
```

```
# Plot the magnitude response
plt.figure()
plt.subplot(2, 1, 1)
plt.plot(w, np.abs(H), 'b')
plt.title('Frequency Response')
plt.xlabel('Frequency
(rad/sample)')
plt.ylabel('Magnitude')
plt.grid()

# Plot the phase response
plt.subplot(2, 1, 2)
plt.plot(w, np.unwrap(np.angle(H)), 'b')
plt.xlabel('Frequency (rad/sample)')
plt.ylabel('Phase (radians)')
plt.grid()

plt.tight_layout()
plt.show()
```