Μία Πρώτη Προσέγγιση

Fotis Branikas

Mar 12, 2021

Άσκηση 1η

Σκοπός της πρώτης σειράς ασκήσεων είναι, αφ' ενός η εξοικείωση με το προγραμματιστικό περιβάλλον της Python, αφ' ετέρου, η εισαγωγή στους τρόπους παράστασης και επεξεργασίας τηλεπικοινωνιακών σημάτων στη συγκεκριμένη γλώσσα προγραμματισμού.

Jupyter notebook

Το Project Jupyter είναι ένας μη κερδοσκοπικός οργανισμός με αποστολή τη συγγραφή ανοικτού λογισμικού για διαδραστικές εφαρμογές. Ξεκίνησε από ipython, ωστόσο σήμερα προσφέρει προγράμματα σε πολλές γλώσσες προγραμματισμού.

Το jupyter notebook είναι μια πλατφόρμα web ανοικτού λογισμικού για την ανάπτυξη διαδραστικών εφαρμογών, κυρίως για επεξεργασία (επιστημονικών) δεδομένων και μηχανικής μάθησης.

Μέρος 1: Εξάσκηση στην Python

Μην ξεχνάτε ότι η IPython μας δίνει τη δυνατότητα να 'εξερευνήσουμε' το περιεχόμενο ενός package, χρησιμοποιώντας τη δυνατότητα του tab-completion, ή τη χρήση του ? για help/documentation: Π.χ., για να δούμε όλα τα περιεχόμενα του signal namespace δίνουμε:

In [3]: signal?

και για να καλέσουμε την ενσωμετωμένη τεκμηρίωση της numpy, δίνουμε:

In [4]: np?

Περισσότερες πληροφορίες μπορείτε να πάρετε από το http://www.numpy.org.

```
# Δημιουργήστε ένα βαθμωτό (μονοδιάστατο) μέγεθος

s=2
print('s =',s)
```

```
s = 2
```

```
# Δημιουργείστε ένα διάνυσμα πραγματικών τιμών:
# Στο MATLAB: v = [1,5,9] ή v = [1 5 9]

v=np.array([1,5,9])
print('v =',v)
```

```
v = [1 5 9]
```

```
# Πρόσβαση στα στοιχεία ενός numpy array
# το πρώτο στοιχείο ξεκινάει στο 0

print(v[0], end=" ")
print(v[1])

# υπάρχει και η δυνατότητα πρόσβασης στοιχείων από το τέλος με αρνητικούς δείκτες
# το τελευταίο στοιχείο έχει δείκτη -1 το προτελευταλιο -2 κ.ο.κ.
```

(continues on next page)

(continued from previous page)

```
print(v[-1], end=" ")
print(v[-2])
```

```
1 5
9 5
```

```
# Τα numpy arrays προσφέρουν και δυνατότητες τεμαχισμού (slicing)
# το απλό slicing u[start:end] ξεκινάει από το στοιχείο
# στη θέση start και φτάνει στη θέση end (χωρίς να την περιέχει)
# αν η αρχή ή το τέλος παραληφθεί, αυτά λαμβάνονται η αρχή ή το τέλος του πίνακα
# επίσης μπορεί να χρησιμοποιηθεί και βήμα με την μορφή u[start:end:step]

u=np.array([1,5,9,7,3,2])
print(u[:3])
print(u[2:])
print(u[1:5])

print(u[::2])
```

```
[1 5 9]
[9 7 3 2]
[5 9 7 3]
[1 9 3]
```

```
# Προσσοχή, όταν θέλουμε να εξάγουμε από έναν πίνακα ένα συγκερκιμένο τεμάχιο
# όπως παρακάτω, τα δύο numpy arrays είναι σενδεδεμένα, δηλ. ο,τι αλλάζει στο ένα
# αλλάζει και στο άλλο
u_slice1 = u[2:5]
print("τεμάχιο:                    ", u_slice1)
print("αρχικός πίνακας:            ", u)
u_slice1[0] = 8
print("τεμάχιο μετά από αλλαγή:    ", u_slice1)
print("αρχικός πίνακα μετά από αλλαγή:", u)

# Όποτε θέλουμε να το αποφύγουμε αυτό χρησιμοποιούμε το .copy()
u_slice2 = u[2:5].copy()
u_slice2[0] = 9
print("\nτεμάχιο με .copy() μετά από αλλαγή:        ", u_slice2)
print("αρχικός πίνακα μετά από αλλαγή με .copy():", u)
```

```
τεμάχιο:                       [9 7 3]
αρχικός πίνακας:               [1 5 9 7 3 2]
τεμάχιο μετά από αλλαγή:       [8 7 3]
αρχικός πίνακα μετά από αλλαγή: [1 5 8 7 3 2]

τεμάχιο με .copy() μετά από αλλαγή:         [9 7 3]
αρχικός πίνακα μετά από αλλαγή με .copy(): [1 5 8 7 3 2]
```

```
# Δημιουργείστε έναν πίνακα πραγματικών τιμών:
# Στο MATLAB: a = a=[[1,2,3];[4,5,6];[7,8,9]] ή a=[1,2,3;4,5,6;7,8,9]
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print('a =',a)
```

```
a = [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
# Αθροίστε

a+5
```

```
array([[ 6,  7,  8],
       [ 9, 10, 11],
       [12, 13, 14]])
```

```
#Πολλαπλασιάστε

b=s*v*2
print('b =',b)
```

```
b = [ 4 20 36]
```

```
# Πολλαπλασιάστε στοιχείο-προς-στοιχείο (elementwise)
# MATLAB: v.*b

np.multiply(v,b)
```

```
array([  4, 100, 324])
```

```
#Ελέγξτε το μήκος ενός διανύσματος
# MATLAB: length(v)

len(v)
```

```
3
```

```
# Ελέγξτε το μέγεθος ενός πίνακα
# MATLAB: size(a)

a.shape   # για array: np.array(a.shape)
```

```
(3, 3)
```

```
# Προσπελάστε συγκεκριμένα στοιχεία ενός πίνακα
# Η δεικτοδότηση αρχίζει από το 0.
# MATLAB: a(1,2)
# --- ΠΡΟΣΟΧΗ, στο MATLAB η δεικτοδ΄ότηση αρχ΄ίζει από το 1!

a[0,1]
```

```
2
```

```
# Προσπελάστε συγκεκριμένα στοιχεία ενός πίνακα (συνέχεια)
# Αρνητικές τιμές μετρούν από το τέλος, π.χ. το -1
# αναφέρεται στο τελευταίο στοιχείο)
```

(continues on next page)

(continued from previous page)

```
a[1,-1]
```

```
6
```

```
# Προσπελάστε συγκεκριμένο τμήμα ενός διανύσματος
# MATLAB: v(1:9)

v[1:3]

# ΠΡΟΣΟΧΗ: τα στοιχεία [2ο,3ο] δίνονται ως 1:3 και όχι ως 1:2
# Δοκιμάστε το v[1:2]...
```

```
array([5, 9])
```

```
# Προσπελάστε συγκεκριμένα τμήματα ενός πίνακα

a[0:2,:]

# Ομοίως: οι γραμμές 1 & 2 δίνονται ως 0:2 και όχι ως 0:1
```

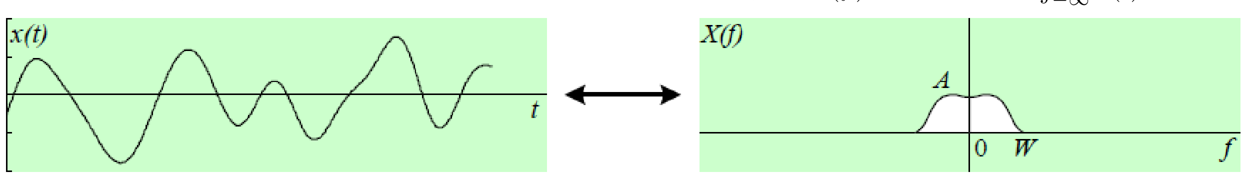
```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
# Δημιουργήστε ένα διάνυσμα με στοιχεία από το 0 έως το 0.5 και βήμα 0.1
# MATLAB: t=(0:0.1:0.4)

t=np.arange(0,0.5,0.1)
print('t=',t)
```

```
t= [0.  0.1 0.2 0.3 0.4]
```

Μέρος 2: Δειγματοληψία - Ψηφιοποίηση

Τα πρωτογενή σήματα είναι κυρίως αναλογικά (συνεχούς χρόνου). Για να τα παραστήσουμε και επεξεργαστούμε στον υπολογιστή μας (ή άλλη ψηφιακή μηχανή) θα πρέπει πρώτα να τα ψηφιοποιήσουμε. Υποθέστε ένα σήμα συνεχούς χρόνου $x(t)$ με μετασχηματισμό Fourier (Continuous Time Fourier Transform – CTFT): $X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt$



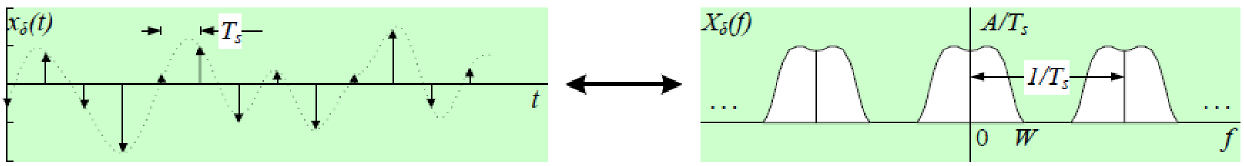
Λαμβάνοντας δείγματα του $x(t)$ με ρυθμό $f_s = 1/T_s$ παράγεται σήμα διακριτού χρόνου $x(nT_s)$. Μαθηματικά το αναπαριστάνουμε ως σειρά συναρτήσεων δέλτα

$$x_\delta(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t-nT_s) = x(t\sum_{n=-\infty}^{\infty} \delta(t-nT_s)$$

με μετασχηματισμό Fourier

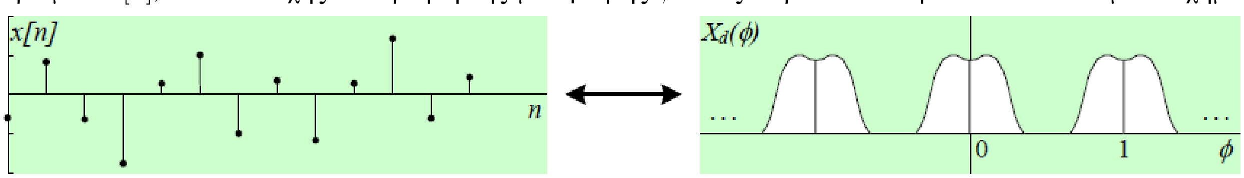
$$X_\delta(f) = \sum_{n=-\infty}^{\infty} x(nT_s)e^{-j2\pi fnT_s} = X(f) * 1/T_s \sum_{n=-\infty}^{\infty} \delta(f - k/T_s) = 1/T_s \sum_{n=-\infty}^{\infty} X(f - k/T_s)$$

που είναι περιοδική συνάρτηση.



Για βαθυπερατά σήματα $x(t)$ εύρους ζώνης W, με την υπόθεση ότι ο ρυθμός δειγματοληψίας $fs{\geq}2W$, ισχύει ότι $X(f) = T_s X_\delta(f)$, $0{\leq}f{\leq}W$, δηλαδή, το σήμα $X(f)$ προκύπτει μετά από διάβαση του δειγματοληπτημένου $x_\delta(t)$ μέσω ιδανικού βαθυπερατού φίλτρου κέρδους T_s. Από το προηγούμενο σχήμα γίνεται φανερό ότι εάν η δειγματοληψία γίνει με συχνότητα μικρότερη του διπλασίου της ανώτερης συχνότητας W του σήματος (υποδειγμάτιση – undersampling), τότε εμφανίζονται στην περιοχή συχνοτήτων του σήματος «είδωλα» φάσματος από ανώτερες συχνότητες που δεν επιτρέπουν την ακριβή αποκατάσταση του αρχικού σήματος συνεχούς χρόνου. Το φαινόμενο αυτό ονομάζεται **αναδίπλωση** ή **επικάλυψη** (aliasing), το δε σφάλμα κατά την αποκατάσταση του αρχικού σήματος αποκαλείται σφάλμα αναδίπλωσης (aliasing error). Η δειγματοληψία στο πεδίο του χρόνου αποτελεί τη βάση για τον ορισμό του μετασχηματισμού Fourier διακριτού χρόνου (Discrete Time Fourier Transform – DTFT). Για μια σειρά διακριτών αριθμών $x[n]$, ο μετασχηματισμός Fourier διακριτού χρόνου ορίζεται ως: $X_\delta(\phi) \triangleq \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi n\phi}$

Ο DTFT είναι περιοδική συνάρτηση με περίοδο 1, επομένως, αρκεί ο υπολογισμός του στο διάστημα συχνοτήτων $[0, 1]$ ή ισοδύναμα $[-½, ½]$. Να σημειωθεί ότι ο DTFT, παρότι προκύπτει από μια σειρά διακριτών αριθμών $x[n]$, είναι συνεχής συνάρτηση της μεταβλητής ϕ όπως παραστατικά φαίνεται στο επόμενο σχήμα.



Με τη σειρά των διακριτών αριθμών να προκύπτει ως αποτέλεσμα δειγματοληψίας, $x[n] = x(nT_s)$, ο DTFT και ο μετασχηματισμός Fourier $X_\delta(f)$ του δειγματοληπτημένου σήματος συνδέονται μέσω της αντιστοιχίας $\phi{\leftrightarrow}f/f_s$. Η συνήθης πρακτική είναι να παριστάνουμε τον λόγο f/f_s ως κανονικοποιημένη συχνότητα ϕ (f_D, στις σημειώσεις σας) και οι πραγματικές συχνότητες να προκύπτουν ως πολλαπλάσιά της (συνήθως κλασματικά). Για τη σύνδεση του DTFT με τον μετασχηματισμό Fourier $X(f)$ του σήματος πρέπει επιπλέον να γίνει αναγωγή στην περίοδο δειγματοληψίας με πολλαπλασιασμό επί T_s (ή διαίρεση με f_s). Κατ΄ αναλογία με τη δειγματοληψία σημάτων στο χρόνο μπορούμε να κάνουμε δειγματοληψία στο πεδίο της συχνότητας λαμβάνοντας διακριτές τιμές $X(kf_o)$ του μετασχηματισμού Fourier που αντιστοιχούν σε ανάλυση συχνότητας $f_o = 1/T_o$. Αυτό ισοδυναμεί με περιοδική επανάληψη του σήματος συνεχούς χρόνου $x(t)$ κάθε , αφού το περιοδικό σήμα $x_p(t) = \sum_{n=-\infty}^{\infty} x(t - nT_o)$

έχει μετασχηματισμό Fourier

$$X(f) \sum_{n=-\infty}^{\infty} e^{-j2\pi fnT_o} = X(f)\frac{1}{T_o} \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T_o}) = \frac{1}{T_o} \sum_{k=-\infty}^{\infty} X(\frac{k}{T_o})\delta(f - \frac{k}{T_o})$$

Επομένως, $X[k] = X(kf_o)/_o$ είναι οι συντελεστές του αναπτύγματος σε σειρά Fourier.του περιοδικού σήματος $x_p(t)$. Προφανώς, για σήματα $x(t)$ πεπερασμένης διάρκειας, όπου $x(t) = 0$ για $|t|{\geq}T$, με την υπόθεση ότι η περίοδος $T_o{\geq}2T$, ισχύει ότι $x(t) = x_p(t)$ για $|t|{\leq}T$. Στην πράξη, τα σήματα έχουν πολύ μεγάλη διάρκεια για να μπορέσουμε να τα αναλύσουμε στην ολότητά τους. Έτσι εφαρμόζουμε ένα ορθογωνικό χρονικό παράθυρο, ώστε να διατηρήσουμε μόνο το πιο σημαντικό τους μέρος για το διάστημα παρατήρησης και $x(t) = 0$, αλλού. Κατά τον υπολογισμό του DTFT $X_d(\phi)$ ενός τέτοιου ακρωτηριασμένου σήματος, αντί του απείρου αθροίσματος, περιοριζόμαστε σε μια πεπερασμένου μήκους L σειρά αριθμών $x[n]$, οπότε

$$X_d(\phi) = \sum_{n=0}^{L-1} x[n] e^{-j2\pi n\phi}$$

Η δειγματοληψία του $X_d(\phi)$ στο πεδίο συχνότητας σε ισαπέχουσες κανονικοποιημένες συχνότητες 0, $1/$, $2/$, …, $(- 1)/$, δίνει

$$X[k] = X_d(\frac{k}{N}) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi n\frac{k}{N}}, 0 \leq k \leq N-1$$

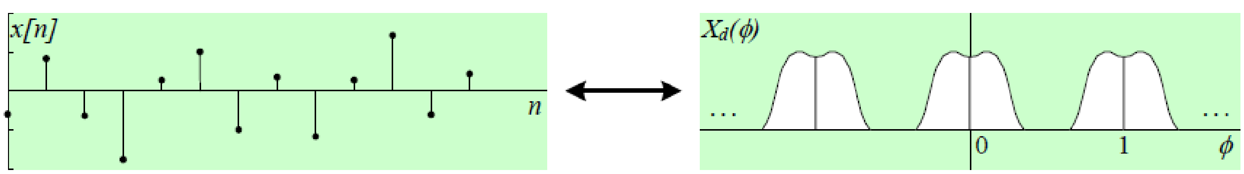
όπου, εάν $N{\geq}L$, θέτουμε $x[n] = 0$ για $n{\geq}L$. Η τελευταία σχέση αναγνωρίζεται ως ο διακριτός μετασχηματισμός Fourier (Discrete Fourier Transform – DFT), ο οποίος για μια πεπερασμένη σειρά xn, $n = 0$, 1, …, $N - 1$, ορίζεται ως:

$$X_k \triangleq \sum_{n=0}^{N-1} x_n e^{-j2\pi n\frac{k}{N}}, 0 \leq k \leq N-1$$

και ο αντίστροφός του είναι

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{-j2\pi n\frac{k}{N}}, 0 \leq n \leq N-1$$

Η $X_d(\phi)$ ως DTFT είναι περιοδική συνάρτηση και εάν η αρχική σειρά xn ήταν περιοδική (και δεν εφαρμόζαμε το παράθυρο), τότε η $X_d(\phi)$ θα ήταν μηδέν παντού εκτός των σημείων της δειγματοληψίας $k/$. Δηλαδή, εάν θεωρήσουμε μια πεπερασμένου μήκους σειρά αριθμών που επαναλαμβάνεται περιοδικά, ο διακριτού χρόνου μετασχηματισμός Fourier της (DTFT) είναι και αυτός περιοδικός και διακριτός. Επιπλέον, ο DFT και ο αντίστροφός του IDFT, εάν δεν περιορίζαμε τους δείκτες n και k μεταξύ 0 και $N - 1$, θα ήταν περιοδικές συναρτήσεις. Άρα η πεπερασμένη σειρά xn μπορεί να θεωρηθεί ως ένα περιοδικό σήμα διακριτού χρόνου ιδωμένο μόνο κατά τη διάρκεια μιας περιόδου και ο DFT, η σειρά X_k, ως τα δείγματα με ανάλυση $1/$ του DTFT $X_d(\phi)$ στο πεδίο κανονικοποιημένων συχνοτήτων $[0, 1]$, όπως φαίνεται στο επόμενο σχήμα.



Φασματική Ανάλυση

Για τον υπολογισμό της ενέργειας ή ισχύος της κυματομορφής $x(t)$, ανάλογα με την περίπτωση σήματος, ισχύει

$$E_x = \int_{-\infty}^{\infty} x^2(t)dt = \int_{-\infty}^{\infty} |X(f)|^2 df$$

$$P_x = \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x^2(t)dt = \int_{-\infty}^{\infty} S_X(f)df$$

όπου για σήματα ισχύος $S(f)$ είναι η πυκνότητα φάσματος ισχύος (Power Spectral Density – PSD) της $x(t)$. Για σήματα διακριτού χρόνου που προκύπτουν από δειγματοληψία της $x(t)$ με περίοδο T_s, οι αντίστοιχες σχέσεις υπολογισμό της ενέργειας ή ισχύος γίνονται

$$E_x = T_s \sum_{n=-\infty}^{\infty} x^2[n]$$

$$P_x = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x^2[n]$$

Ένας απλός τρόπος να εκτιμηθεί η πυκνότητα φάσματος ισχύος της κυματομορφής $x(t)$ είναι να ληφθεί ο DTFT των δειγμάτων του σήματος και μετά να υψωθεί στο τετράγωνο το μέτρο του αποτελέσματος. Αυτός ο εκτιμητής αποκαλείται περιοδόγραμμα (periodogram). Το περιοδόγραμμα ενός πεπερασμένου μήκους L σήματος $x[n]$ ορίζεται ως

$$P_{xx}(f) \triangleq \frac{|X_d(\frac{f}{f_s})|^2}{f_s L}$$

όπου $X_d(\phi)$ ο DTFT του σήματος. Με το μήκος L να τείνει στο άπειρο, το περιοδόγραμμα $P_{xx}(f)$ τείνει στην πυκνότητα φάσματος ισχύος $S(f)$. Ο υπολογισμός του περιοδογράμματος σε πεπερασμένο πλήθος συχνοτήτων $kf_s/, k = 0, 1, \dots ,$ δίνει

$$P_{xx}[k] = \frac{|X_k|^2}{f_s L}, k = 0, 1, ..., N-1$$

όπου X_k και ο DFT της πεπερασμένου μήκους L σειράς δειγμάτων του σήματος. Η ισχύς του σήματος είναι τότε

$$P_X = \frac{1}{f_s L} \sum_{k=0}^{N-1} |X_k|^2 f_o = \frac{1}{NL} \sum_{k=0}^{N-1} |X_k|^2 = \frac{1}{L} \sum_{n=0}^{L-1} |x_n|^2$$

όπου η τελευταία ισότητα προκύπτει από το θεώρημα Parseval, που για την περίπτωση του DFT εκφράζεται ως:

$$\sum_{n=0}^{N-1} |x_n|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X_k|^2$$

Στην ειδική περίπτωση περιοδικών σημάτων έχουμε

$$S_X(f) = \sum_{k=-\infty}^{\infty} |X[k]|^2 \delta(f - \frac{k}{T_o})$$

$$P_X(f) = \sum_{k=-\infty}^{\infty} |X[k]|^2$$

όπου $X[k]$ οι συντελεστές του αναπτύγματος σε σειρά Fourier και T_o η περίοδος του σήματος.

Μέρος 3: Εφαρμογή Α

```
Fs = 1000  # συχνότητα δειγματοληψίας 1000 Hz
Ts = 1 / Fs  # περίοδος δειγματοληψίας
L = 1000  # μήκος σήματος (αριθμός δειγμάτων)
T = L * Ts  # διάρκεια σήματος
t = np.arange(0, (L - 1) * Ts, Ts)  # χρονικές στιγμές υπολογισμού του σήματος

x = np.sin(2 * np.pi * 30 * t) \
    + 0.8 * np.sin(2 * np.pi * 80 * (t - 2)) \
    + np.sin(2 * np.pi * 60 * t)  # συνιστώσα 60 Hz

# Σχεδιάστε το σήμα στο πεδίο του χρόνου
```

(continues on next page)

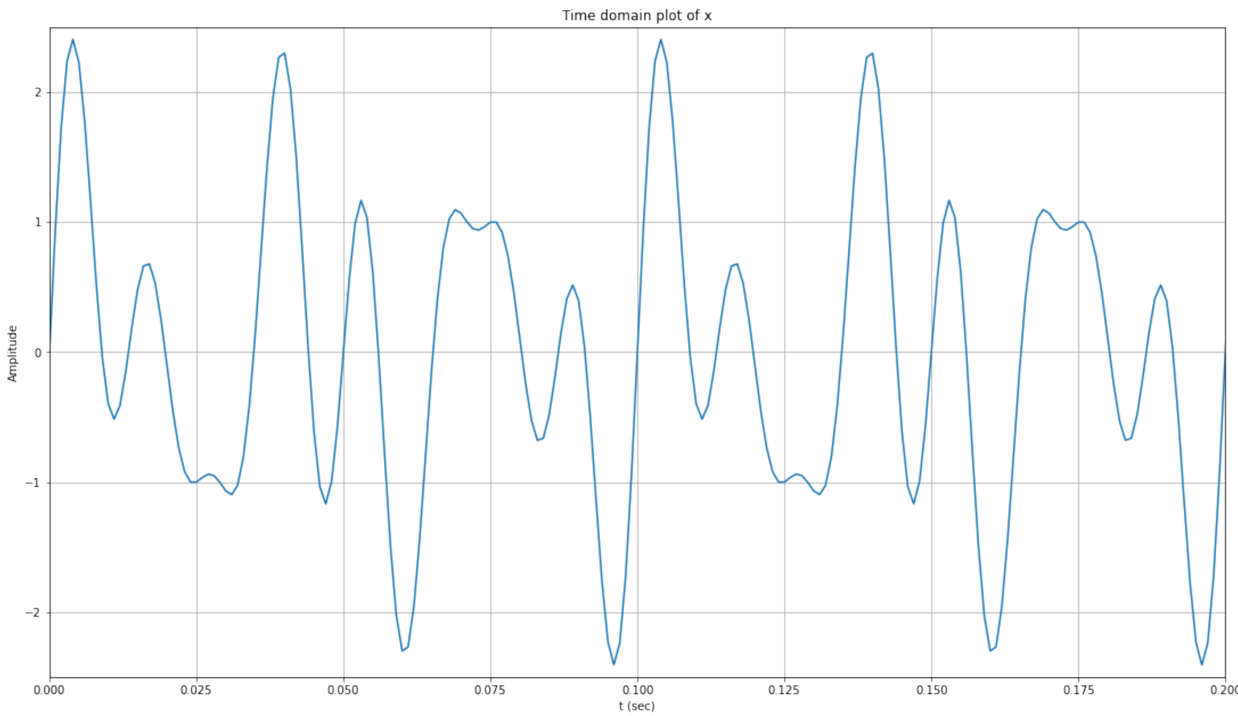
(continued from previous page)

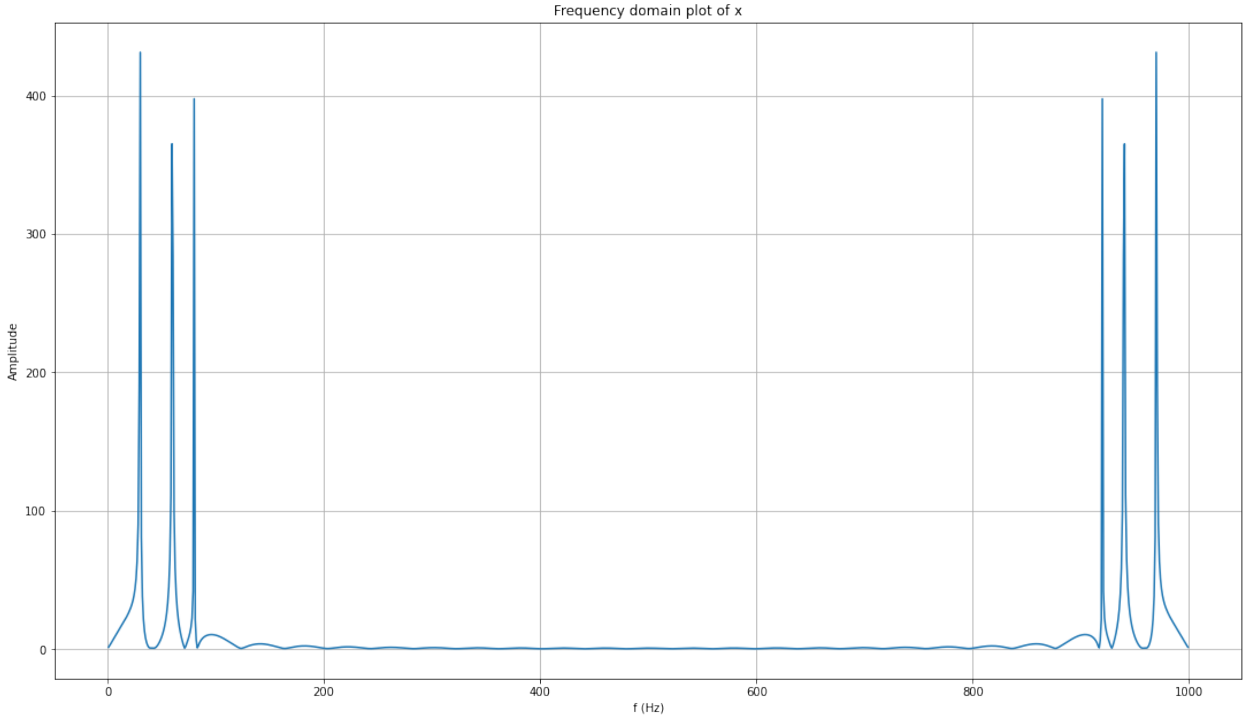
```
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(t, x)
ax.set(xlabel='t (sec)', ylabel='Amplitude',
       title='Time domain plot of x')
ax.grid()
ax.axis([0, 0.2, -2.5, 2.5])
plt.savefig('Time domain plot of x')
plt.show()


# Υπολογίστε τον διακριτό μετασχηματισμό Fourier

def nextpow2(i):  # επιστρέψει το n, τέτοιο ώστε 2^n >= L
    n = 0

    while 2 ** n < i:
        n += 1

    return n


N = 2 ** nextpow2(L)  # μήκος μετασχηματισμού Fourier.
# η nextpow2 βρίσκει τη δύναμη του 2 που
# είναι μεγαλύτερη ή ίση από το όρισμα L
Fo = Fs / N  # ανάλυση συχνότητας
f = np.arange(0, N) * Fo  # διάνυσμα συχνοτήτων
X = np.fft.fft(x, N)  # αριθμητικός υπολογισμός του διακριτού μετασχηματισμού Fourier␣
↪(DFT) για N σημεία

# Σχεδιάστε το σήμα στο πεδίο συχνότητας

# Αφού το σήμα είναι πραγματικό μπορείτε να σχεδιάσετε μόνο τις θετικές συχνότητες
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(f[np.arange(1, N)], abs(X[np.arange(1, N)]))
ax.set(xlabel='f (Hz)', ylabel='Amplitude',
       title='Frequency domain plot of x')
ax.grid()
plt.savefig('Frequency domain plot of x')
plt.show()

f = f - Fs / 2  # ολίσθηση συχνοτήτων προς τα αριστερά κατά -Fs/2
X = np.fft.fftshift(X)  # ολίσθηση της μηδενικής συχνότητας στο κέντρο του φάσματος
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 12)
ax.plot(f, abs(X))
ax.set(xlabel='f (Hz)', ylabel='Amplitude',
       title='Two sided spectrum of x')
ax.grid()
plt.savefig('Two sided spectrum of x')
plt.show()

# Υπολογίστε την ισχύ

power = np.multiply(X, np.conj(X)) / N / L  # υπολογισμός πυκνότητας ισχύος
fig, ax = plt.subplots()
```

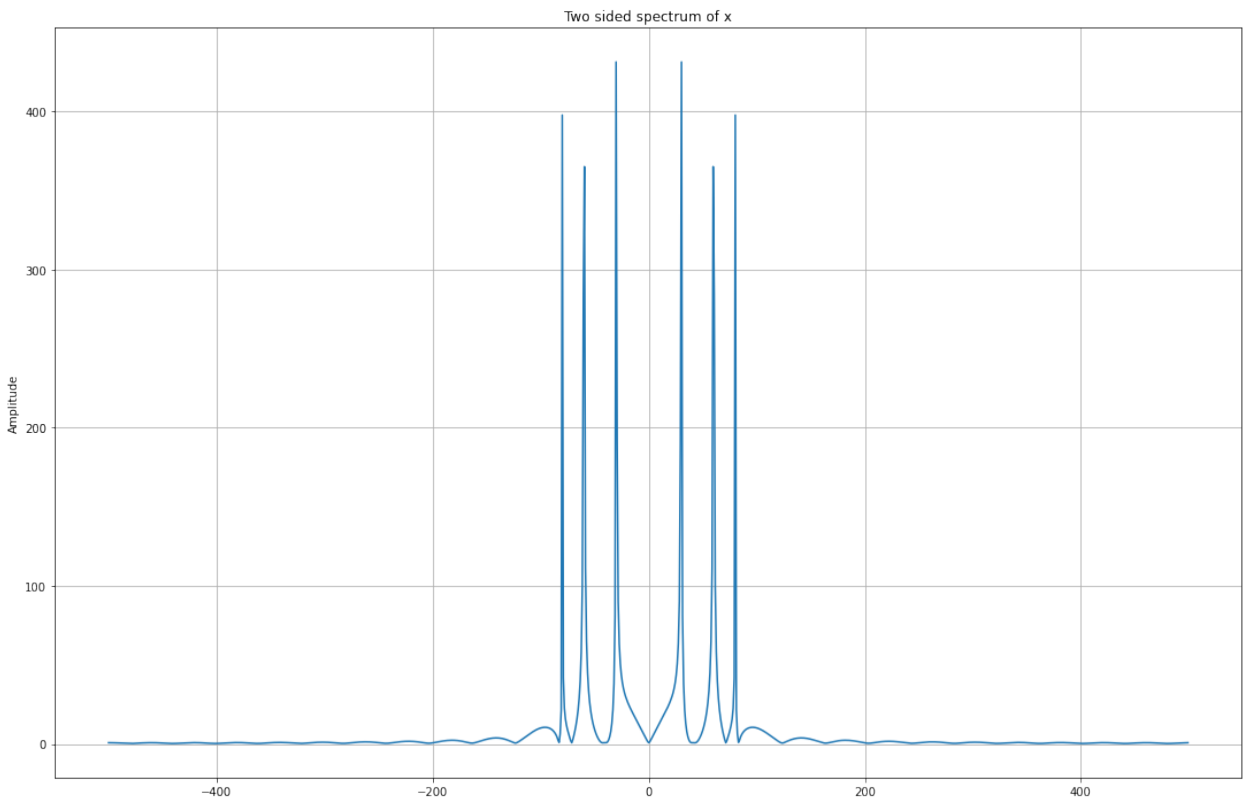
(continues on next page)

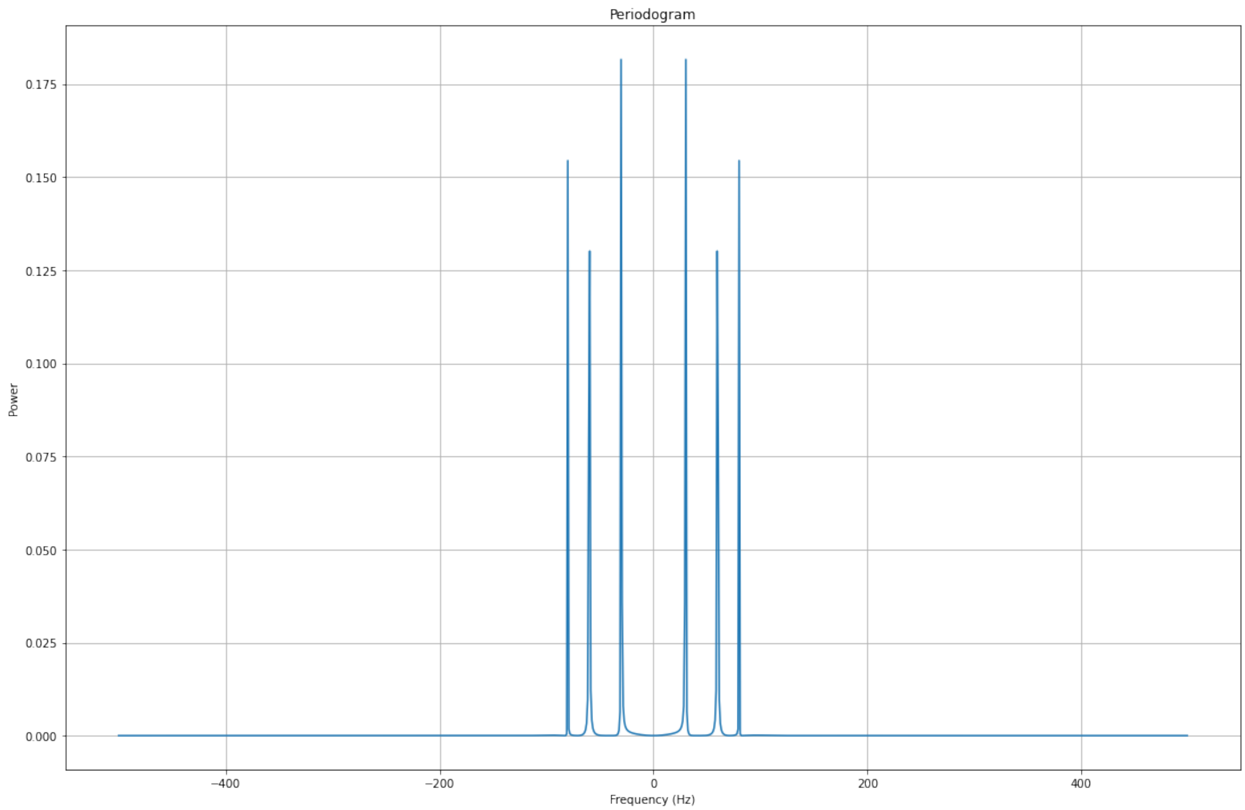
(continued from previous page)

```
fig.set_size_inches(18.5, 12)
ax.plot(f, power)
ax.set(xlabel='Frequency (Hz)', ylabel='Power',
       title='Periodogram')
ax.grid()
plt.savefig('Periodogram')
plt.show()
```




Frequency domain plot of x
Amplitude
f (Hz)
0
100
200
300
400
0
200
400
600
800
1000


Two sided spectrum of x
Amplitude
f (Hz)
0
100
200
300
400
−400
−200
0
200
400

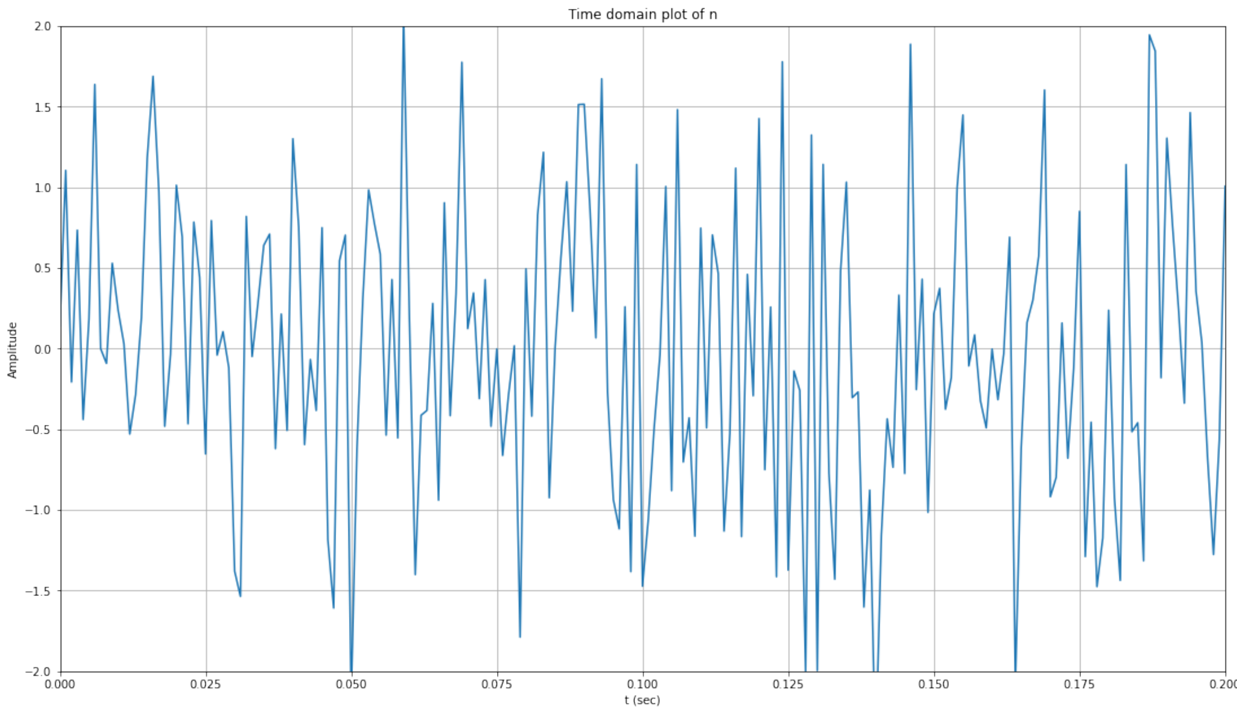


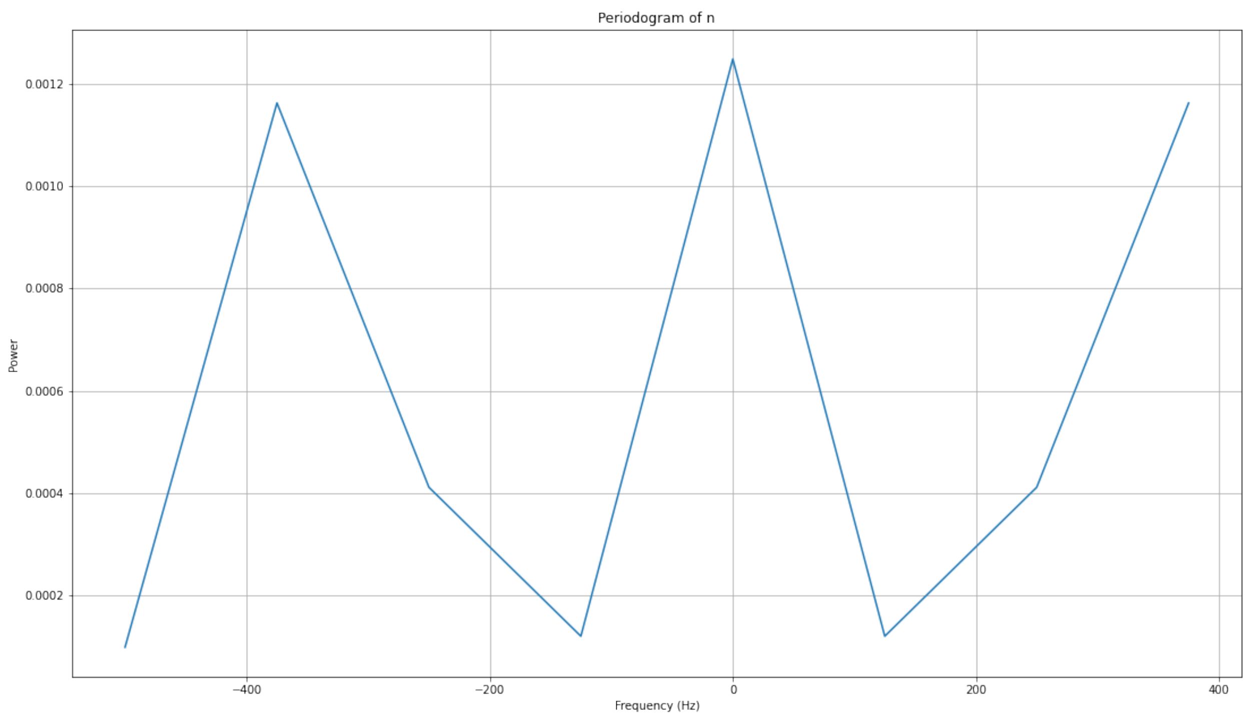

```
# Part 2 Προσθέστε θόρυβο στο σήμα

# Συμπληρώστε τον κώδικα για τη δημιουργία του σήματος θορύβου n με τη βοήθεια της␣
↪συνάρτησης randn.
# Το διάνυσμα θορύβου n θα πρέπει να είναι του ίδιου μεγέθους με αυτό της␣
↪ημιτονοειδούς κυματομορφής x του πρώτου μέρους.
# Σχεδιάστε το σήμα θορύβου στο διάστημα από 0 έως 0.2 sec και κλίμακα σε από -2 έως␣
↪2.
# Υπολογίστε το περιοδόγραμμα του n και σχεδιάστε την πυκνότητα φάσματος ισχύος του␣
↪σήματος θορύβου.
# Προσθέστε το σήμα θορύβου και το x για να λάβετε το σήμα με θόρυβο s.
# Σχεδιάσατε το σήμα με θόρυβο s στο πεδίο του χρόνου στην περιοχή 0 έως 0.2 sec
# και κλίμακα από -2 έως 2 καθώς και το αμφίπλευρο φάσμα του.

rand_n = random.randn(np.size(x))
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(t,rand_n)
ax.set(xlabel='t (sec)', ylabel='Amplitude',
       title='Time domain plot of n')
ax.axis([0, 0.2, -2, 2])
ax.grid()
plt.show()

N = 2^nextpow2(L)
Fo=Fs/N
f=(np.arange(0,N))*Fo
rand_N=np.fft.fft(rand_n,N)
```

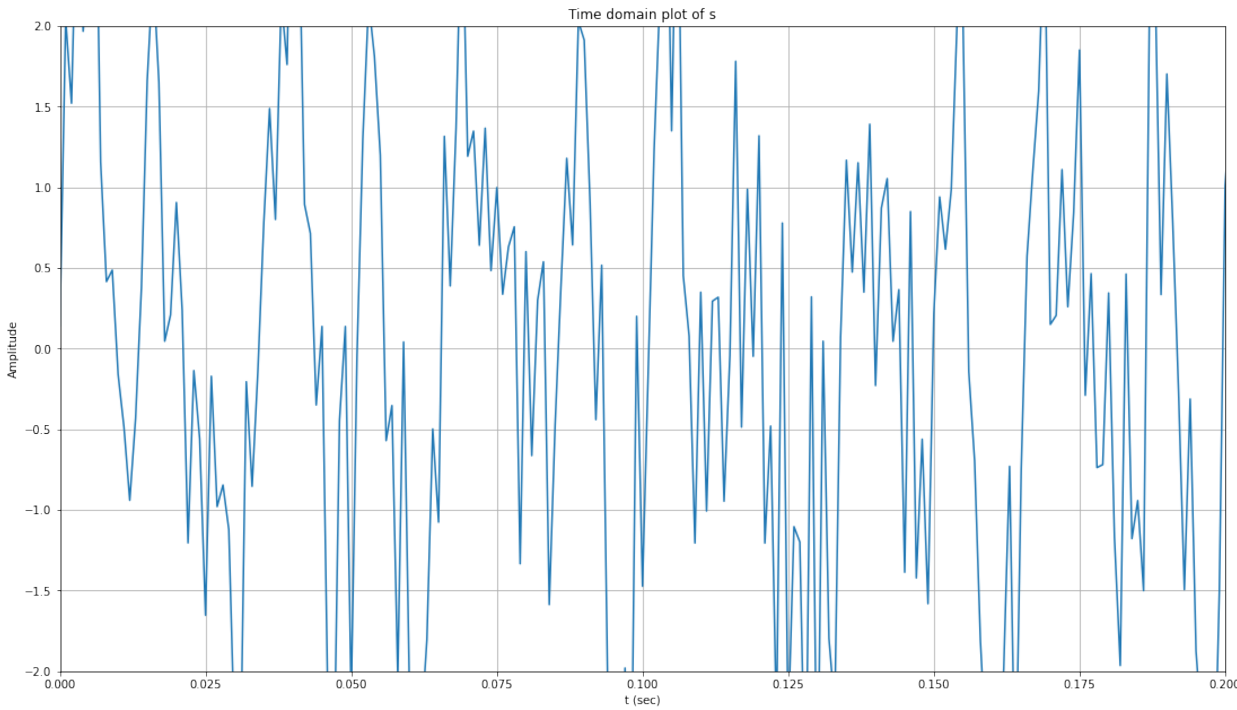
(continues on next page)

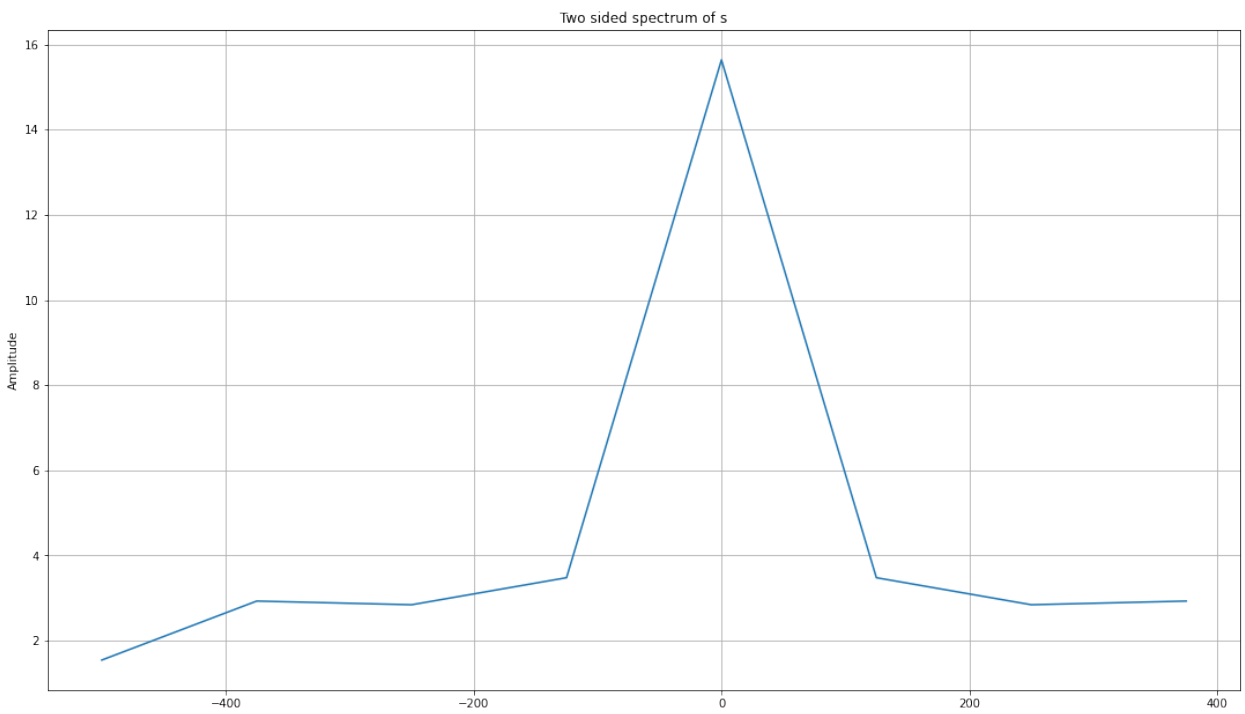
(continued from previous page)

```
f=f-Fs/2
rand_N=np.fft.fftshift(rand_N)

power_n=np.multiply(rand_N,np.conj(rand_N))/N/L
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(f,power_n)
ax.set(xlabel='Frequency (Hz)', ylabel='Power',
       title='Periodogram of n')
ax.grid()
plt.show()

s = x + rand_n

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(t,s)
ax.set(xlabel='t (sec)', ylabel='Amplitude',
       title='Time domain plot of s')
ax.axis([0, 0.2, -2, 2])
ax.grid()
plt.show()

N = 2^nextpow2(L)

Fo=Fs/N
f=(np.arange(0,N))*Fo
S=np.fft.fft(s,N)

f=f-Fs/2
S=np.fft.fftshift(S)

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(f,abs(S))
ax.set(xlabel='f (Hz)', ylabel='Amplitude',
       title='Two sided spectrum of s')
ax.grid()
plt.show()
```


Time domain plot of n
Amplitude
t (sec)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200


Periodogram of n
Power
Frequency (Hz)
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
−400
−200
0
200
400



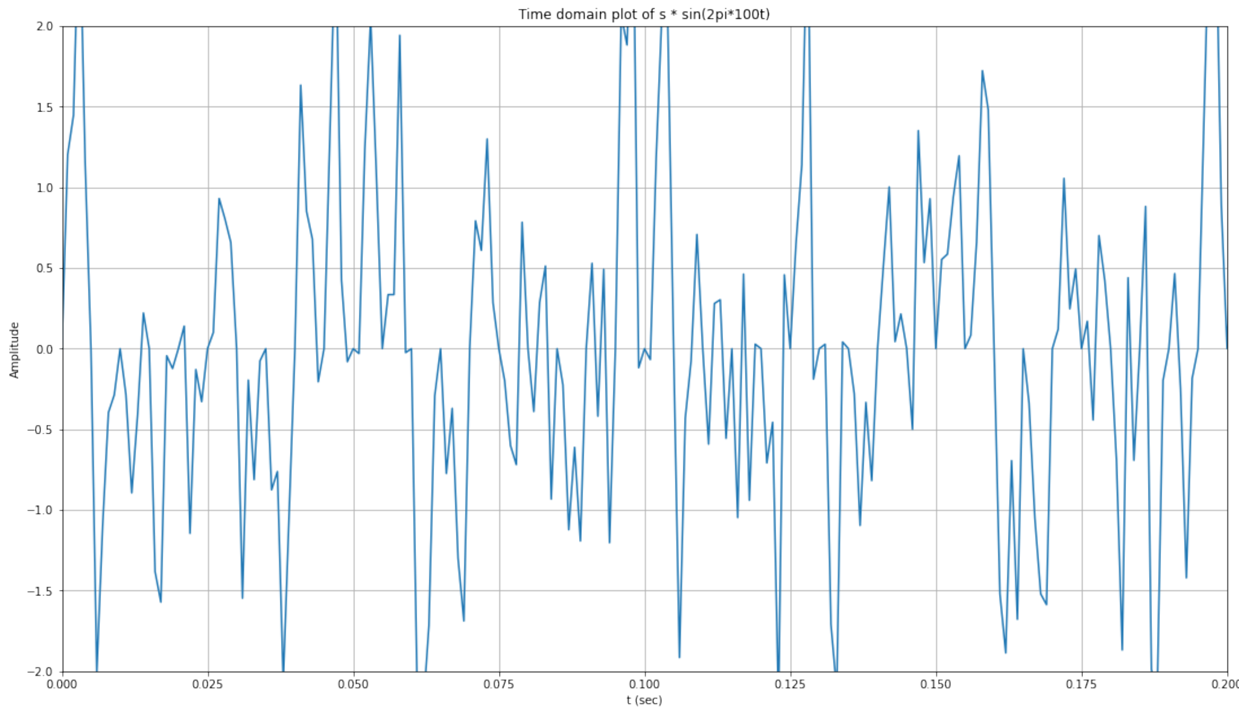


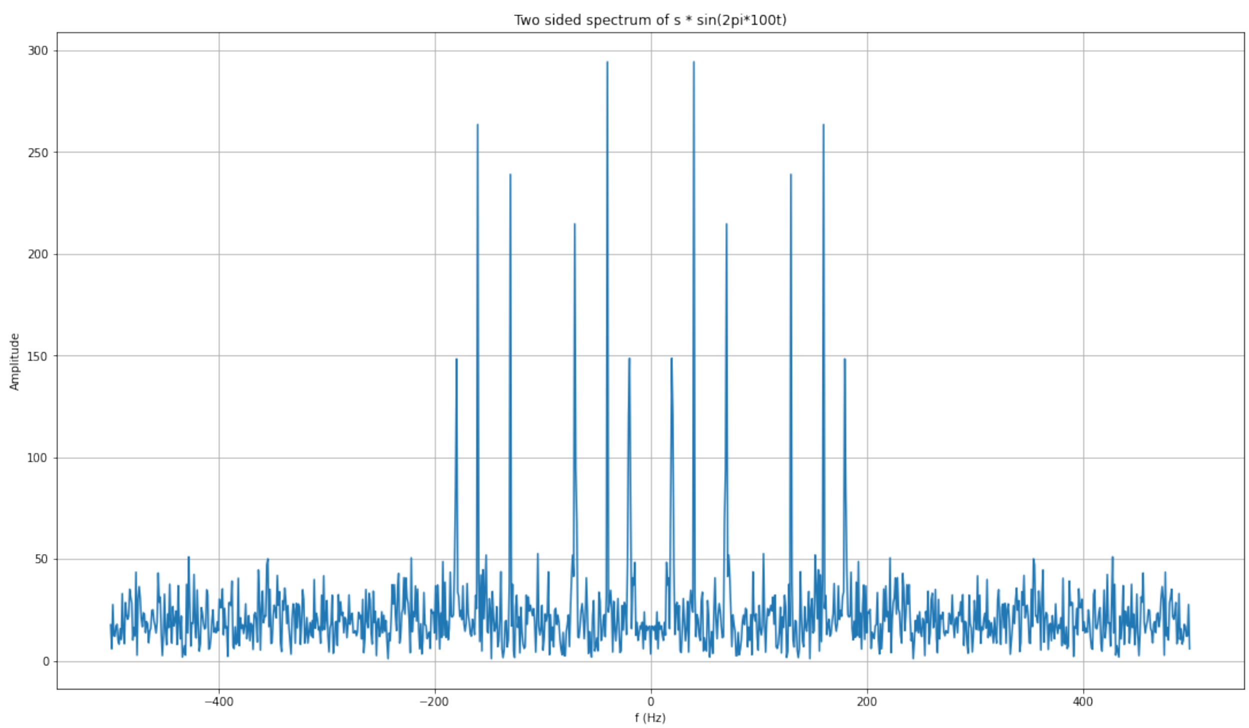

```
# Part 3. Πολλαπλασιασμός σημάτων

# Συμπληρώστε τον κώδικα δημιουργίας ενός ημιτονοειδούς σήματος συχνότητας
# 100 Hz και πολλαπλασιάστε με το προηγούμενο σήμα s.
# Τα δύο σήματα θα πρέπει να είναι του ίδιου μεγέθους.
# Σχεδιάστε το αποτέλεσμα στο πεδίο του χρόνου στην περιοχή 0 έως 0.2 sec
# και κλίμακα από -2 έως 2 καθώς και στο πεδίο της συχνότητας
```

(continues on next page)

(continued from previous page)

```
# χρησιμοποιώντας τη συνάρτηση fftshift.


s_mul = np.sin(2 * np.pi * 100 * t)
s_final = s * s_mul

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(t, s_final)
ax.set(xlabel='t (sec)', ylabel='Amplitude',
       title='Time domain plot of s * sin(2pi*100t)')
ax.axis([0, 0.2, -2, 2])
ax.grid()
plt.savefig('Time domain plot of s * sin(2pi*100t)')
plt.show()

N = 2 ** nextpow2(L)

Fo = Fs / N
f = (np.arange(0, N)) * Fo
S_final = np.fft.fft(s_final, N)

f = f - Fs / 2
S_final = np.fft.fftshift(S_final)

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.plot(f, abs(S_final))
ax.set(xlabel='f (Hz)', ylabel='Amplitude',
       title='Two sided spectrum of s * sin(2pi*100t)')
ax.grid()
plt.savefig('Two sided spectrum of s * sin(2pi*100t)')
plt.show()
```


Time domain plot of s * sin(2pi*100t)
Amplitude
t (sec)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200


Two sided spectrum of s * sin(2pi*100t)
Amplitude
f (Hz)
300
250
200
150
100
50
0
−400
−200
0
200
400

Μέρος 4: Εφαρμογή Β

Να γραφεί σε Python συνάρτηση φασματικής ανάλυσης, παρόμοια με την `signal.welch()`: θα δέχεται ως είσοδο διάνυσμα πραγματικού σήματος καθώς και τη συχνότητα δειγματοληψίας, F_s, και θα σχεδιάζει τη μονόπλευρη φασματική πυκνότητα του σήματος στην περιοχή $[0 - F_s/2)$. Το σήμα θα τεμαχίζεται σε τμήματα μήκους ίσου με τη δύναμη του 2 την πλησιέστερη στο $1/8$ του συνολικού του μήκους, αλλά όχι μικρότερου από 256. Τα τμήματα θα είναι επικαλυπτόμενα κατά 50%. Το τελευταίο τμήμα, εάν υπολείπεται σε μήκος των άλλων, θα αγνοείται. Θα υπολογίζεται με FFT το φάσμα κάθε τμήματος και θα λαμβάνεται η μέση τιμή όλων των τμημάτων. Η συνάρτηση να δοκιμαστεί με το σήμα του παραδείγματος 1.1 και να συγκριθεί το αποτέλεσμα με το αντίστοιχο της `signal.welch()`.

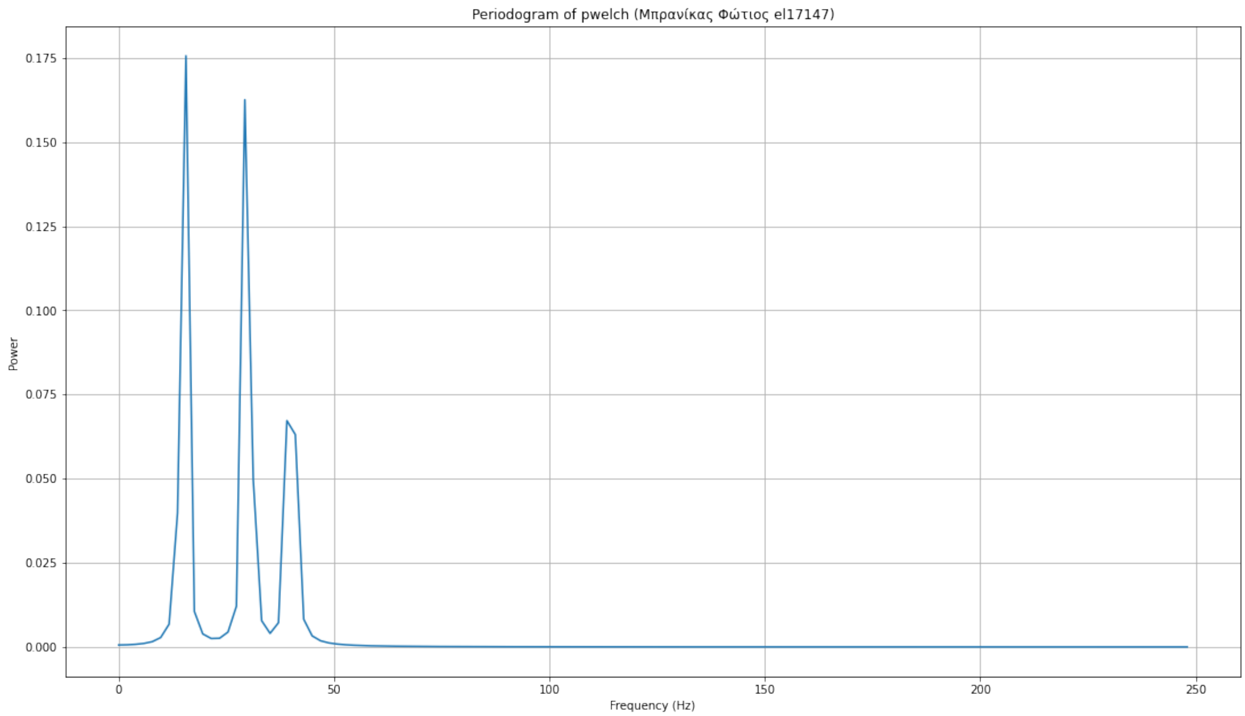
```
def pwelch(x, Fs):
    part_size = max(2 ** nextpow2(np.size(x) // 8), 256)
    N = 2 ** nextpow2(part_size)

    Fo = Fs / N
    f_welch = np.arange(0, N) * Fo

    part_start = 0
    part_end = part_start + part_size
    cntr = 0
    fft_sum = np.zeros(N)

    while part_start + part_size < np.size(x):

        part = x[int(part_start):int(part_end)]
        temp = np.fft.fft(part, N)

        for i in range(0, N):
            fft_sum[i] += abs(temp[i])
            i += 1

        part_start += part_size / 2
        part_end = part_start + part_size
        cntr += 1

    fft_avg = np.divide(fft_sum, cntr)
    avg_pwr = np.multiply(fft_avg, np.conj(fft_avg)) / N / part_size

    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    ax.plot(f_welch[0:N // 2], avg_pwr[0:N // 2])
    ax.set(xlabel='Frequency (Hz)', ylabel='Power', title='Periodogram of pwelch␣
↪(Μπρανίκας Φώτιος el17147)')
    ax.grid()
    plt.savefig('Periodogram of pwelch')
    plt.show()

    return f_welch[np.arange(0, N // 2)], avg_pwr[np.arange(0, N // 2)]
```

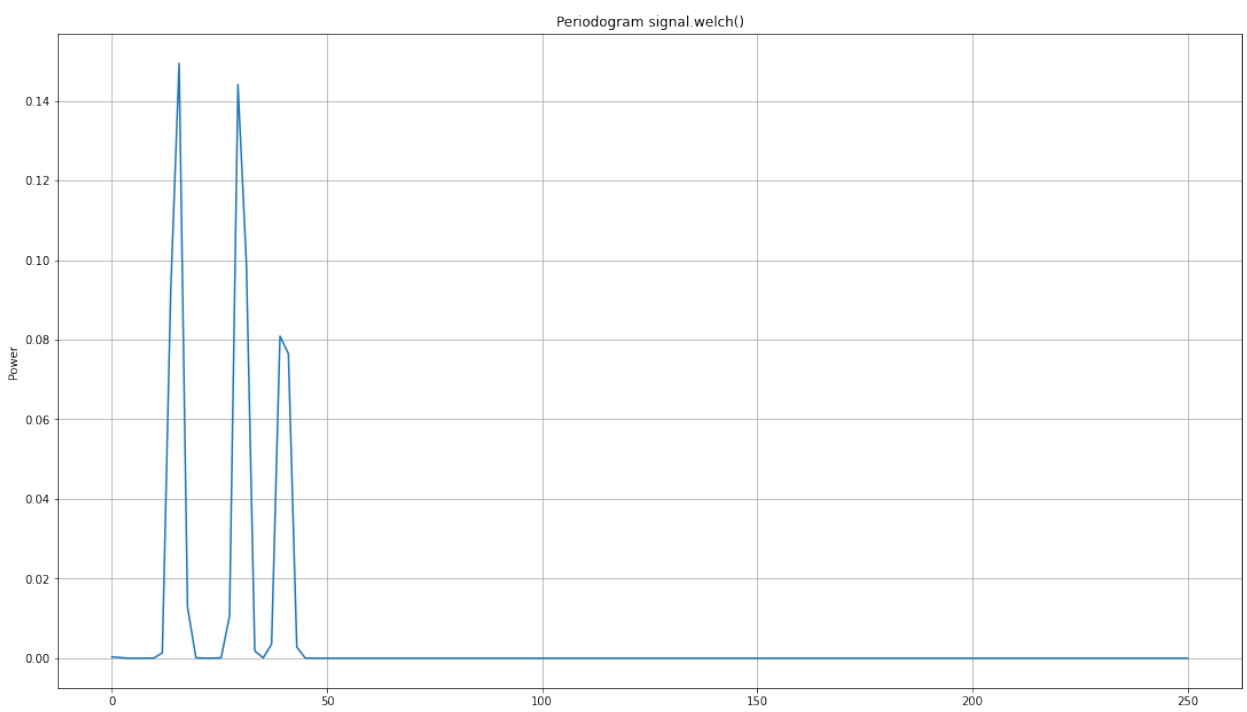
```
Fs=500
f1,Pxx1 = pwelch(x,Fs)
f2,Pxx2 = signal.welch(x,fs=Fs)

fig, ax = plt.subplots()
```

(continues on next page)

(continued from previous page)

```
fig.set_size_inches(18.5, 10.5)
ax.plot(f2,Pxx2)
ax.set(xlabel='Frequency (Hz)', ylabel='Power',
       title='Periodogram signal.welch()')
ax.grid()
plt.show()
```





Άσκηση 2η

Θα ασχοληθούμε με το Παράδειγμα 1.2 της παραγράφου 1.5 του τεύχους Μαθήματος. Το παράδειγμα αυτό παρουσιάζει δύο εναλλακτικές μεθόδους σχεδιασμού FIR φίλτρων: α) τη μέθοδο των παραθύρων και β) τη μέθοδο των ισοϋψών κυματώσεων τις οποίες εφαρμόζει στην περίπτωση βαθυπερατών φίλτρων.

Στο παράδειγμα, τα φίλτρα δοκιμάζονται σε ένα πραγματικό σήμα, s, το οποίο είναι αποθηκευμένο στο αρχείο sima.mat (binary αρχείο MATLAB). Πρόκειται για ένα σήμα sonar με φάσμα που εκτείνεται μέχρι περίπου τα 4 KHz και συχνότητα δειγματοληψίας Fs=8192 (είναι και αυτή αποθηκευμένη στο αρχείο sima.mat, μαζί με το σήμα).

Μέρος 1 Σχεδιασμός και υλοποίηση φίλτρων

Εδώ θα πειραματιστούμε με δύο σήματα: (i) το sonar του παραδείγματος, το οποίο εδώ διαβάζεται από ένα .txt αρχείο (έχει προέλθει με εξαγωγή του s από το MATLAB) και (ii) ένα σήμα μουσικής, το violin.wav (σήμα από μουσική βιολιού), το οποίο περιέχει υψηλότερες συχνότητες και έχει προέλθει με δειγματοληψία στα Fs_viol=44100 Hz.

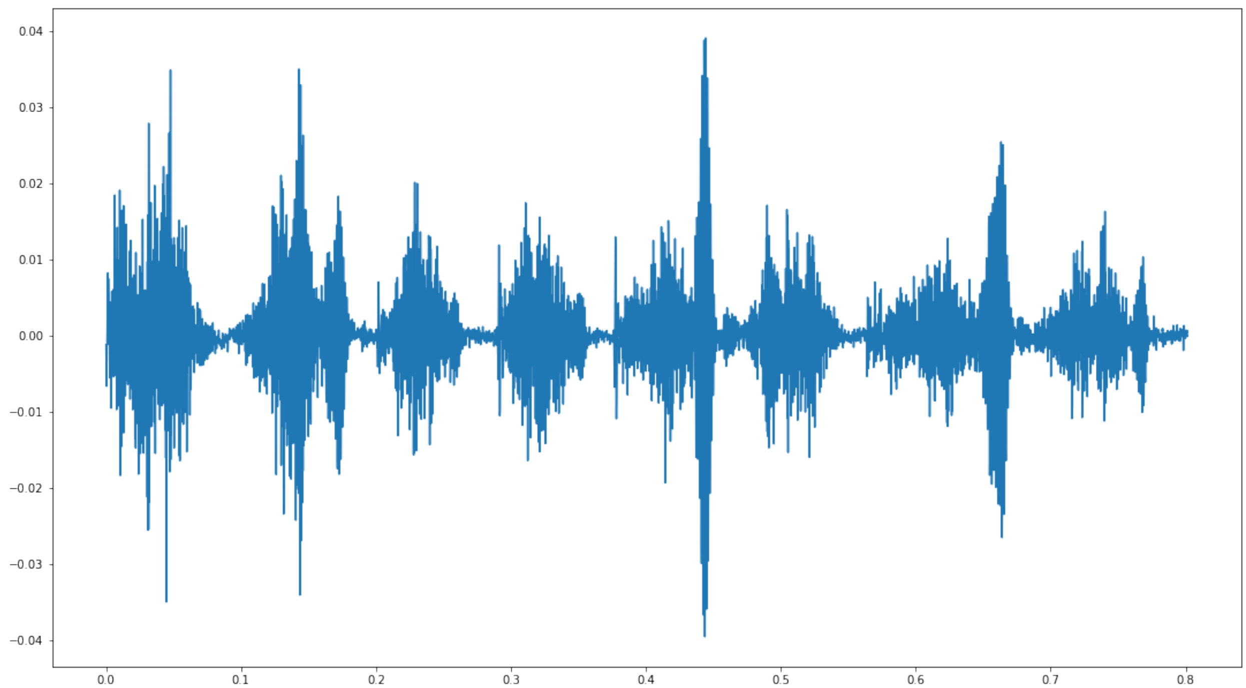
Σήμα sonar

```
# Ανάγνωση δειγμάτων σήματος από txt file
with open('sima.txt') as f:
    s = [float(x) for x in f]
s=np.array(s)
print('μέγεθος σήματος =', s.shape)
Fs=8192
```

```
μέγεθος σήματος = (6565,)
```

Στο πεδίο του χρόνου

```
t=np.arange(0,len(s))/Fs
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

ax.plot(t,s)
plt.show()
```



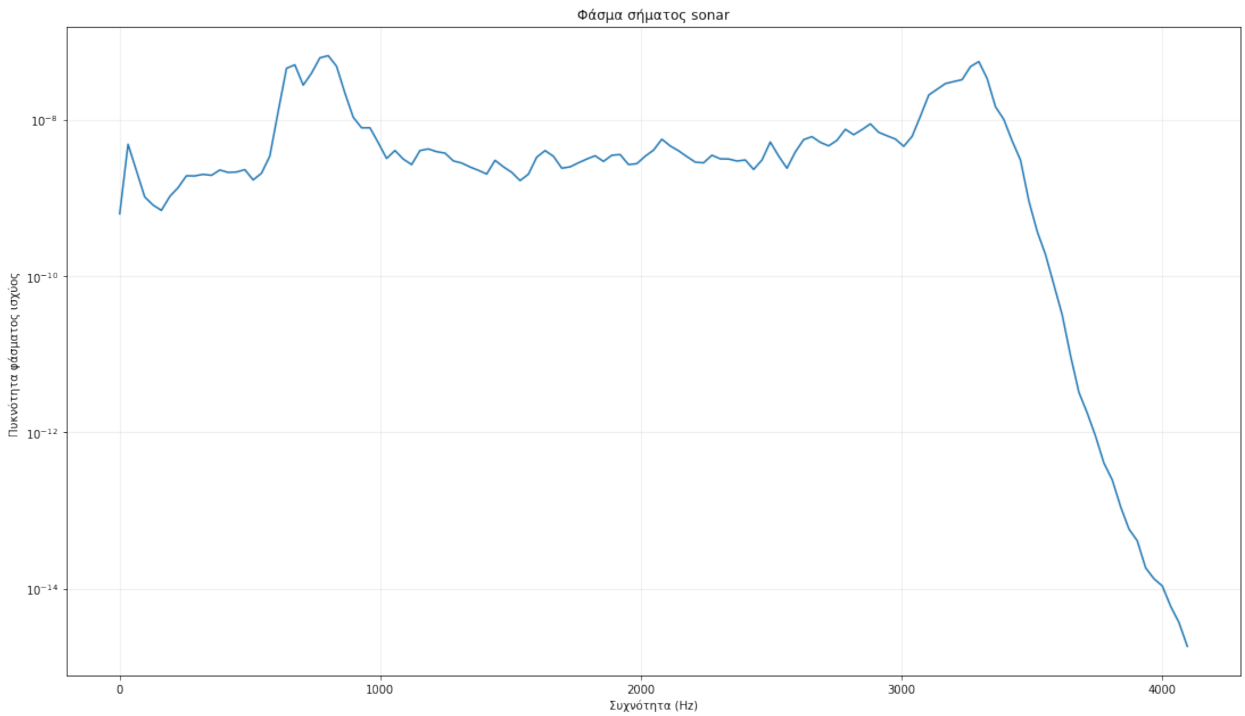
Ακούμε το σήμα

```
# Πρέπει να έχουμε εγκατεστημένη τη βιβλιοθήκη sounddevice
import sounddevice as sd
sd.play(20*s,Fs)
```

Φάσμα (spectrum)

```
f, Pxx_den = signal.welch(s, Fs, noverlap=128, nperseg=256)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

plt.title('Φάσμα σήματος sonar')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Πυκνότητα φάσματος ισχύος ')

ax.semilogy(f, Pxx_den)
```

```
[<matplotlib.lines.Line2D at 0x12361e520>]
```



Σήμα βιολιού

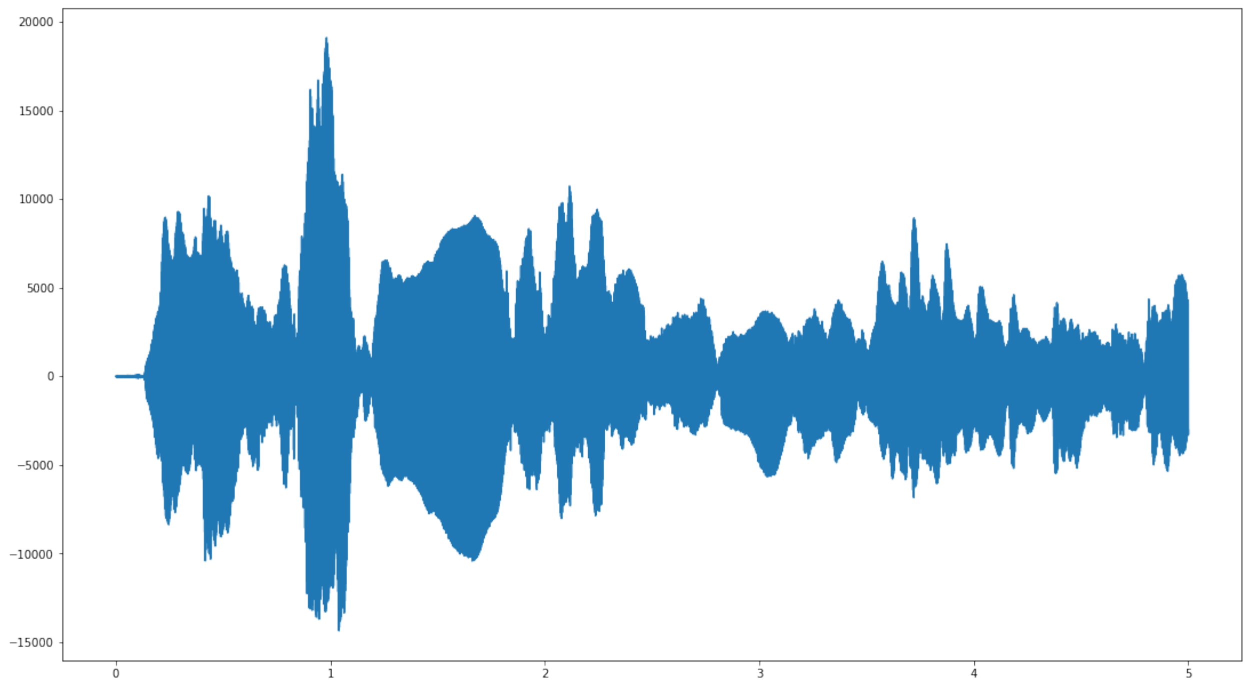
```
from scipy import signal
import scipy.io.wavfile
import numpy as np
import matplotlib.pyplot as plt
```

```
f=open('violin.wav', 'rb')
Fs_viol, s_viol = scipy.io.wavfile.read(f)
print('Fs_viol=',Fs_viol, ' number of samples=',len(s_viol))
f.close()
```

```
Fs_viol= 44100  number of samples= 220500
```

Στο πεδίο του χρόνου

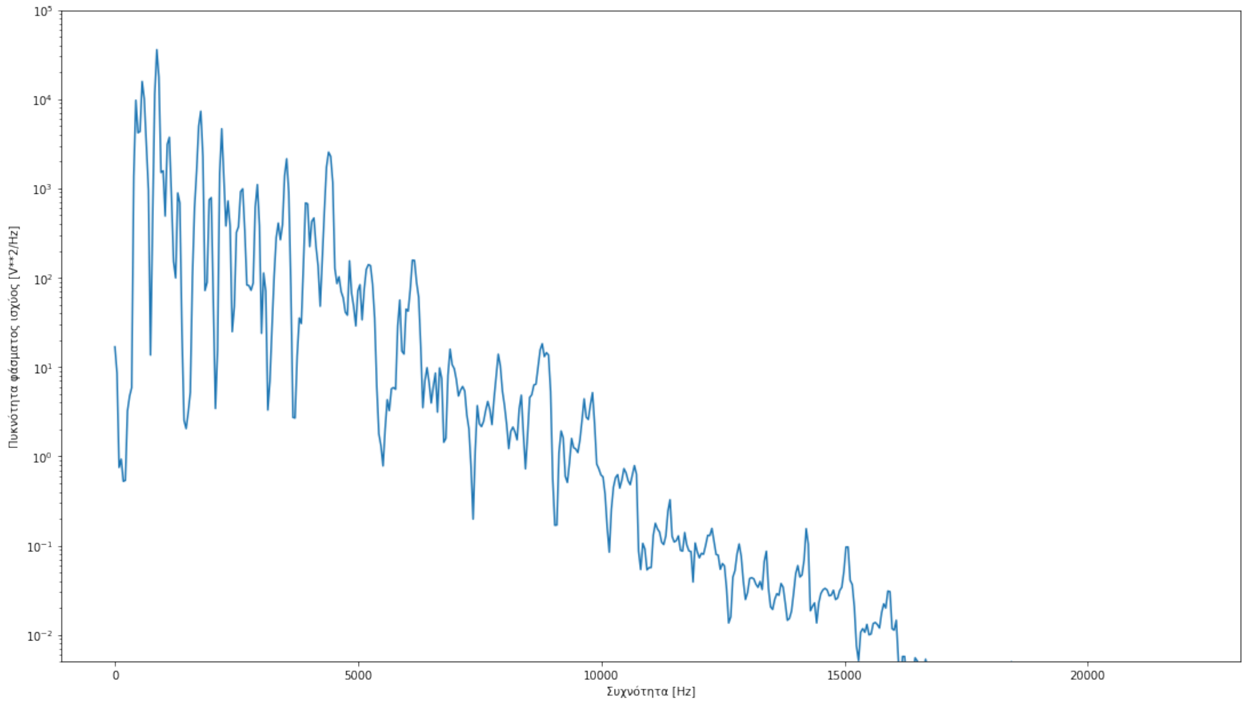
```
tvl=np.arange(0,len(s_viol))/Fs_viol
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

ax.plot(tvl,s_viol)
plt.show()
```



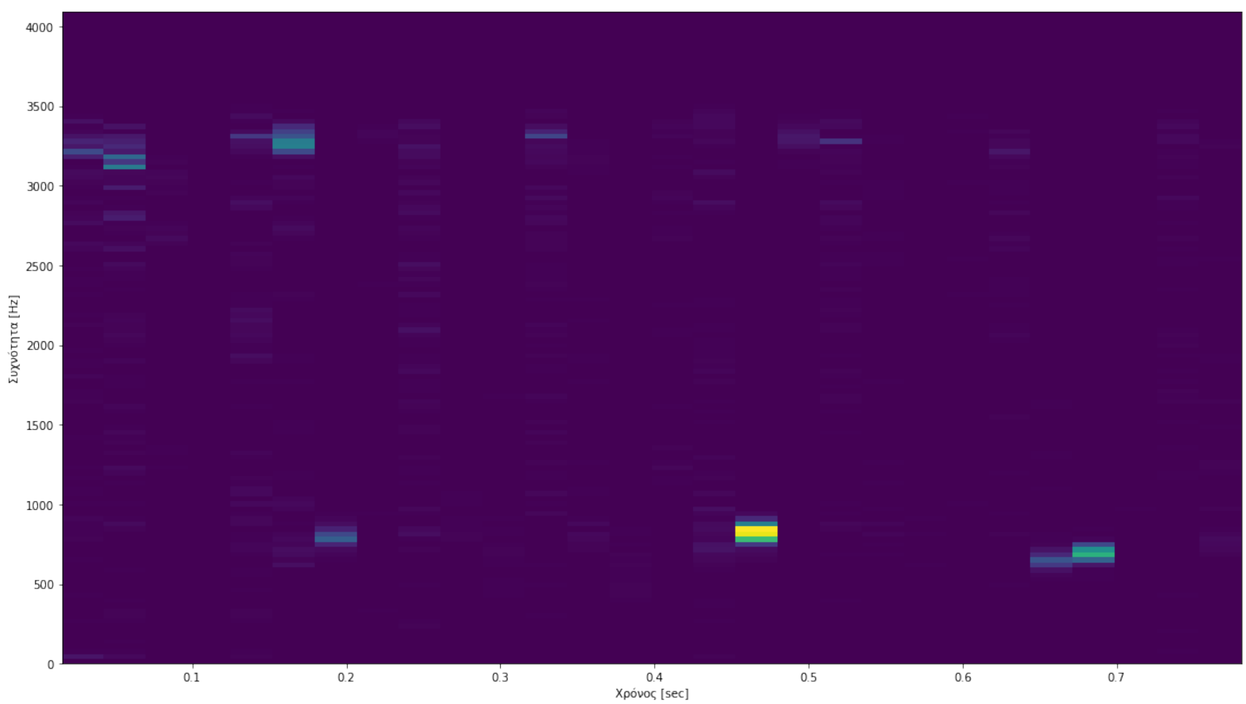
```
# Πρέπει να έχουμε εγκατεστημένη τη βιβλιοθήκη sounddevice
import sounddevice as sd
sd.play(s_viol,Fs_viol)
```

Φάσμα (spectrum) και Φασματόγραμμα (spectorgram)

```
f, Pxx_den = signal.welch(s_viol, Fs_viol, nperseg=1024, noverlap=256)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

ax.semilogy(f, Pxx_den)
plt.ylim([0.5e-2, 1e5])
plt.xlabel('Συχνότητα [Hz]')
plt.ylabel('Πυκνότητα φάσματος ισχύος [V**2/Hz]')
plt.show()
```



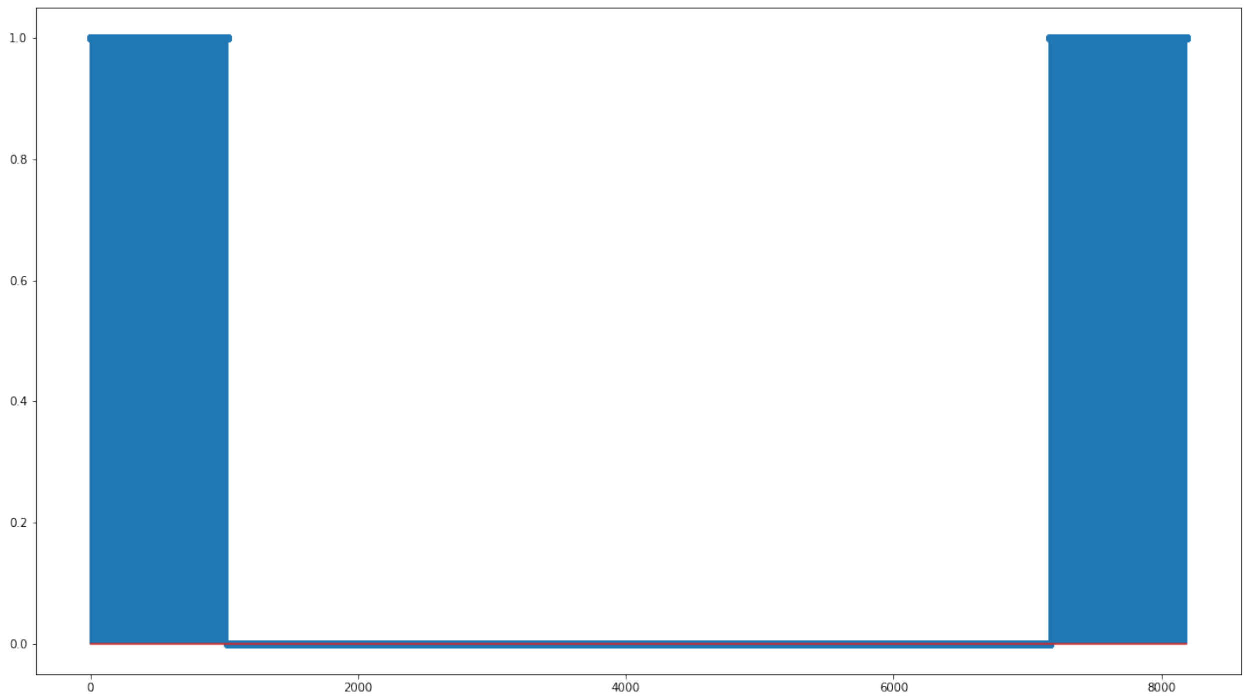

```
f, tsp, Sxx = signal.spectrogram(s, Fs)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

ax.pcolormesh(tsp, f, Sxx)
plt.ylabel('Συχνότητα [Hz]')
plt.xlabel('Χρόνος [sec]')
plt.show()
```



Βαθυπερατά φίλτρα

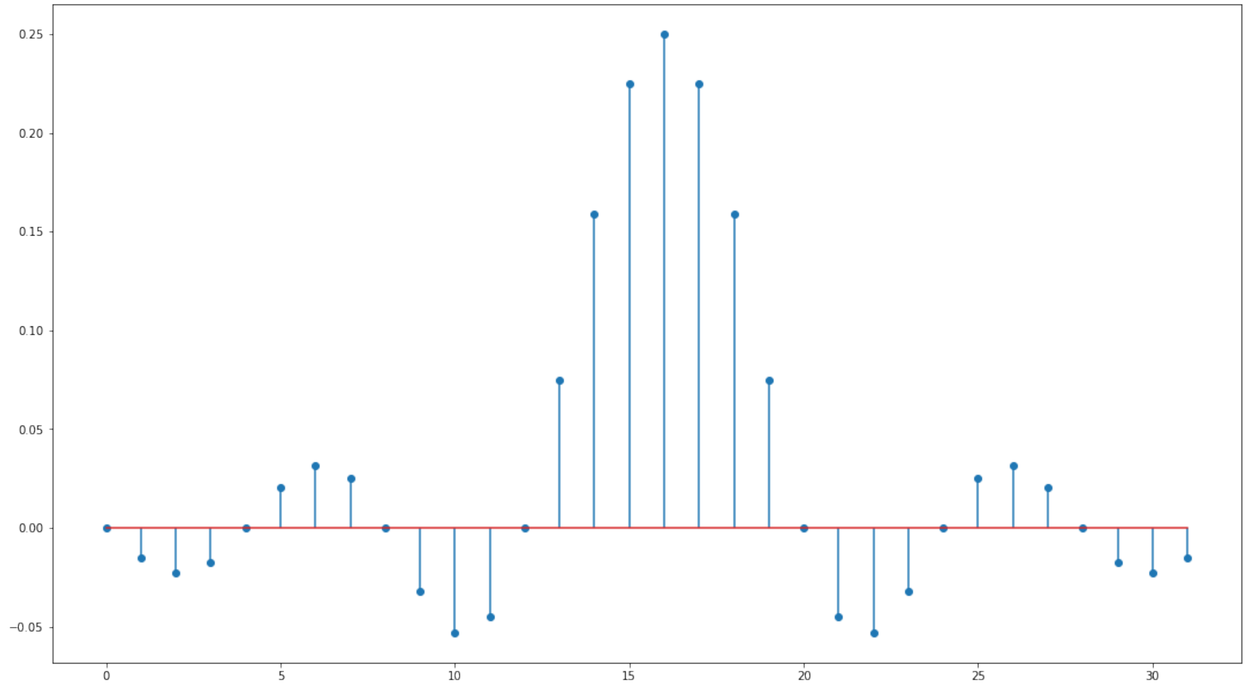
Η μέθοδος των παραθύρων

```python
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt
#
# Fs=8192
H=np.hstack((np.ones(int(Fs/8)), np.zeros(int(Fs-Fs/4)), np.ones(int(Fs/8))))
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

ax.stem(H)
plt.show()
# Το γράφημα αυτό αργεί... περιμένετε...
```

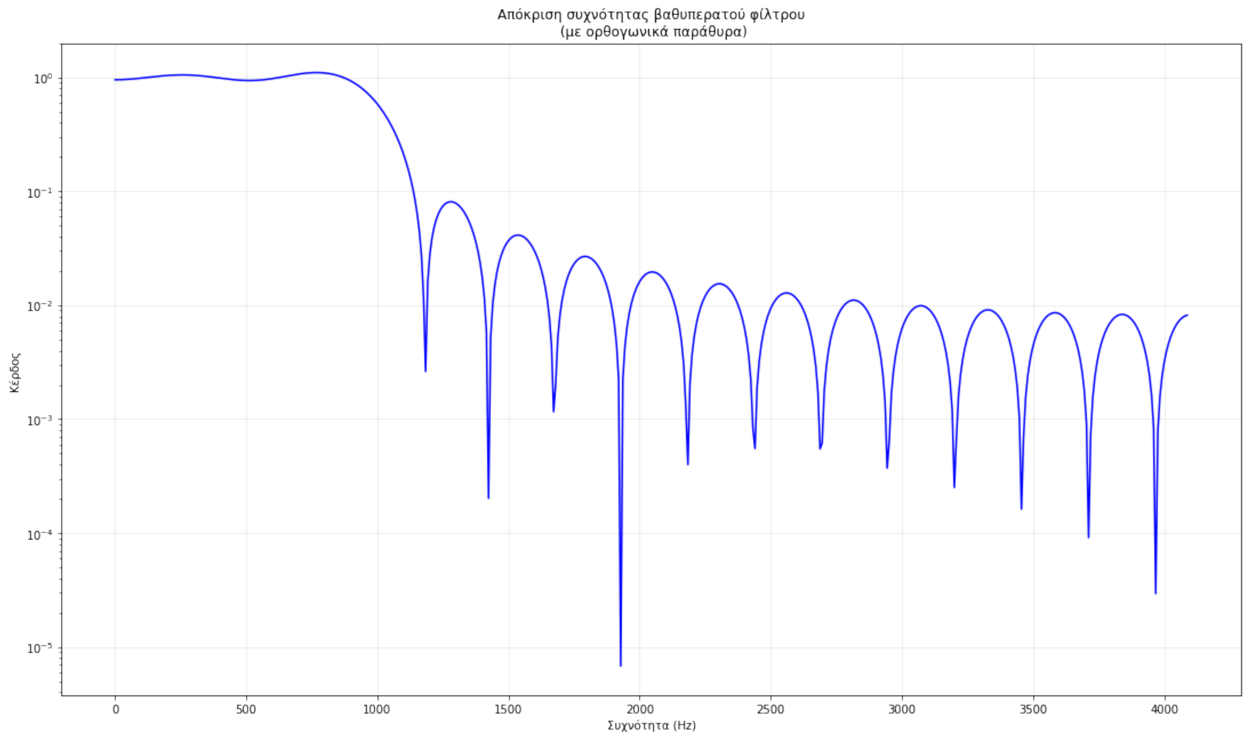


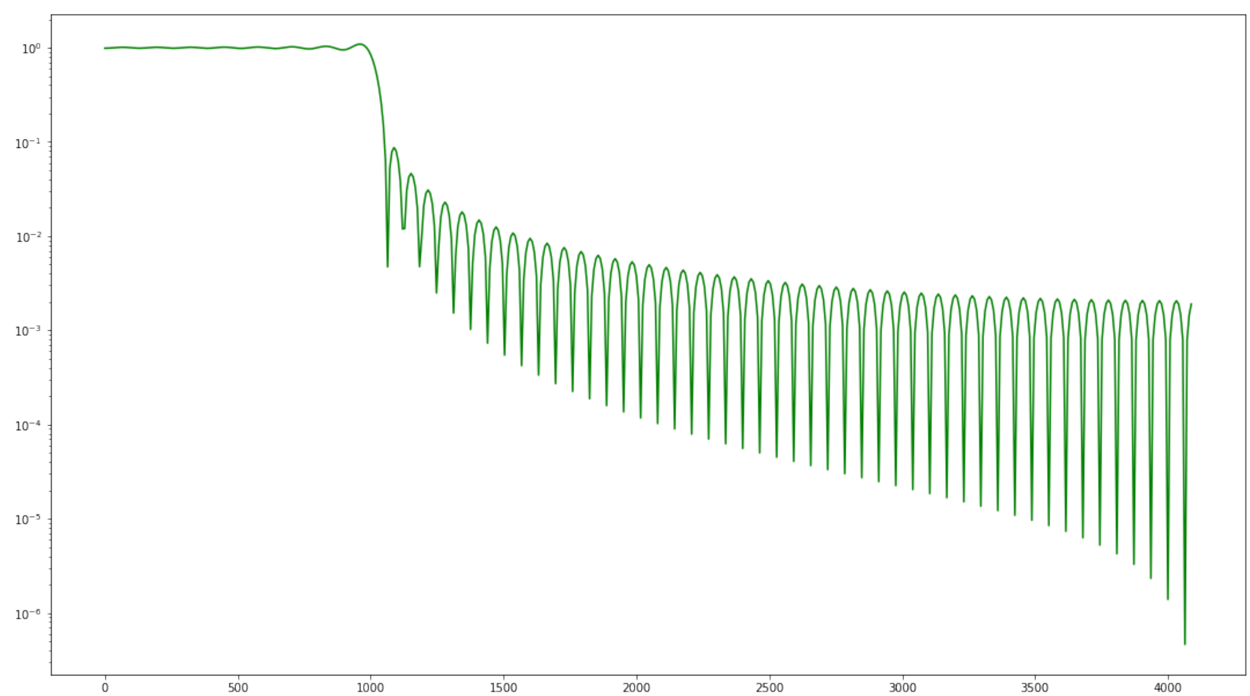
Ορθογωνικό παράθυρο (απλή περικοπή της h)

```python
h=np.real(np.fft.ifft(H));
middle=int(len(h)/2)
h=np.hstack((h[middle:],h[:middle]))
h32=h[middle-16:middle+16]
h128=h[middle-64:middle+64]
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

ax.stem(h32)
plt.show()
```



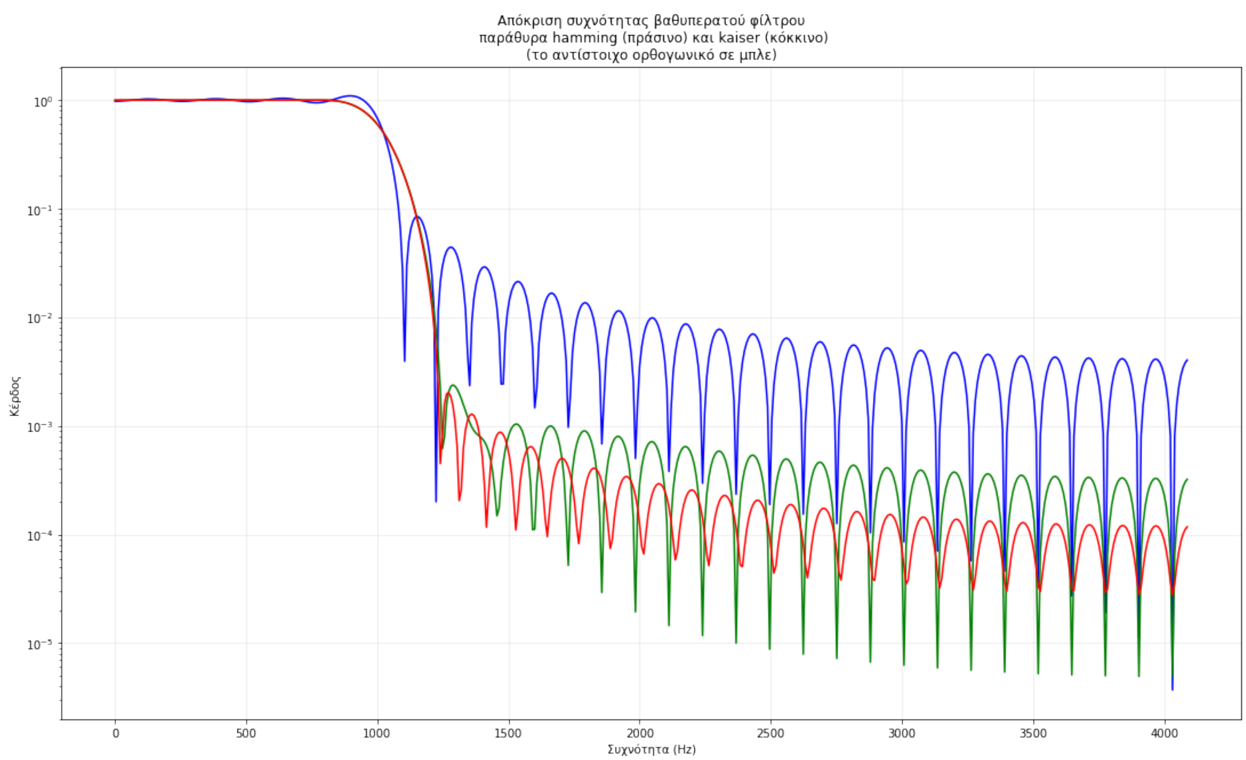

```
# Σχεδίαση απόκρισης συχνότητας (πλάτους)
#  ΜΠΟΡΕΙ ΝΑ ΓΙΝΕΙ ΣΥΝΑΡΤΗΣΗ !!!

freq,resp32 = signal.freqz(h32);


fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

plt.title('Απόκριση συχνότητας βαθυπερατού φίλτρου\n (με ορθογωνικά παράθυρα)')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')
ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp32), 'b-')
freq,resp128 = signal.freqz(h128);
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp128), 'g-')

plt.show()
```


Απόκριση συχνότητας βαθυπερατού φίλτρου
(με ορθογωνικά παράθυρα)
Κέρδος
Συχνότητα (Hz)



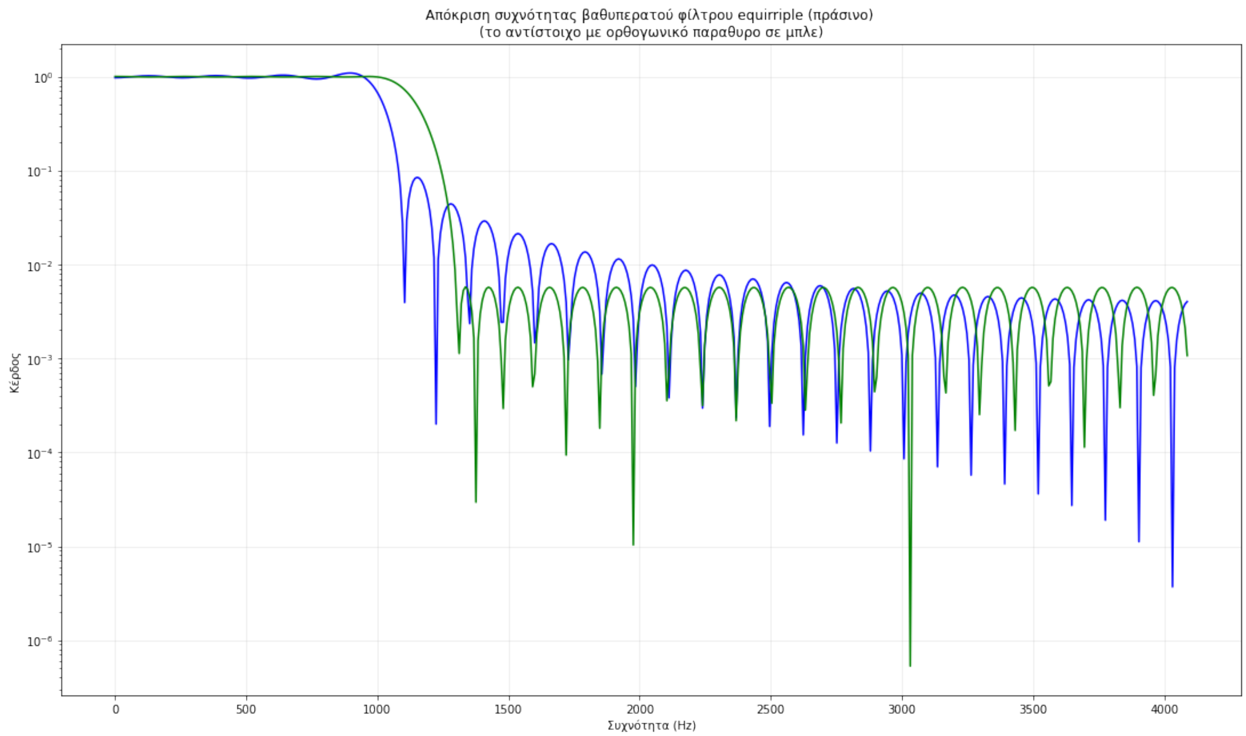
Παράθυρα Hamming και Kaiser

```
h64=h[middle-32:middle+32]
freq,resp64 = signal.freqz(h64);
w_hamming=signal.hamming(len(h64))
h64_hamming = np.multiply(h64,w_hamming)
w_kaiser=signal.kaiser(len(h64),5)
h64_kaiser = np.multiply(h64,w_kaiser)

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)


plt.title('Απόκριση συχνότητας βαθυπερατού φίλτρου\n παράθυρα hamming (πράσινο) και kaiser (κόκκινο)\n(το αντίστοιχο ορθογωνικό σε μπλε)')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')

ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp64), 'b-')
freq,resp64_hamming = signal.freqz(h64_hamming);
ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp64_hamming), 'g-')
freq,resp64_kaiser = signal.freqz(h64_kaiser);
ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp64_kaiser), 'r-')

plt.show()
```



Φίλτρα ισοϋψών κυματώσεων

```
lpass = signal.remez(64, [0, 1000, 1300, Fs/2], [1, 0], fs=Fs)

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

plt.title('Απόκριση συχνότητας βαθυπερατού φίλτρου equirriple (πράσινο) \n(το␣
↪αντίστοιχο με ορθογωνικό παραθυρο σε μπλε)')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')

ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp64), 'b-')
freq,resp_pm = signal.freqz(lpass);
ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp_pm), 'g-')

plt.show()
```



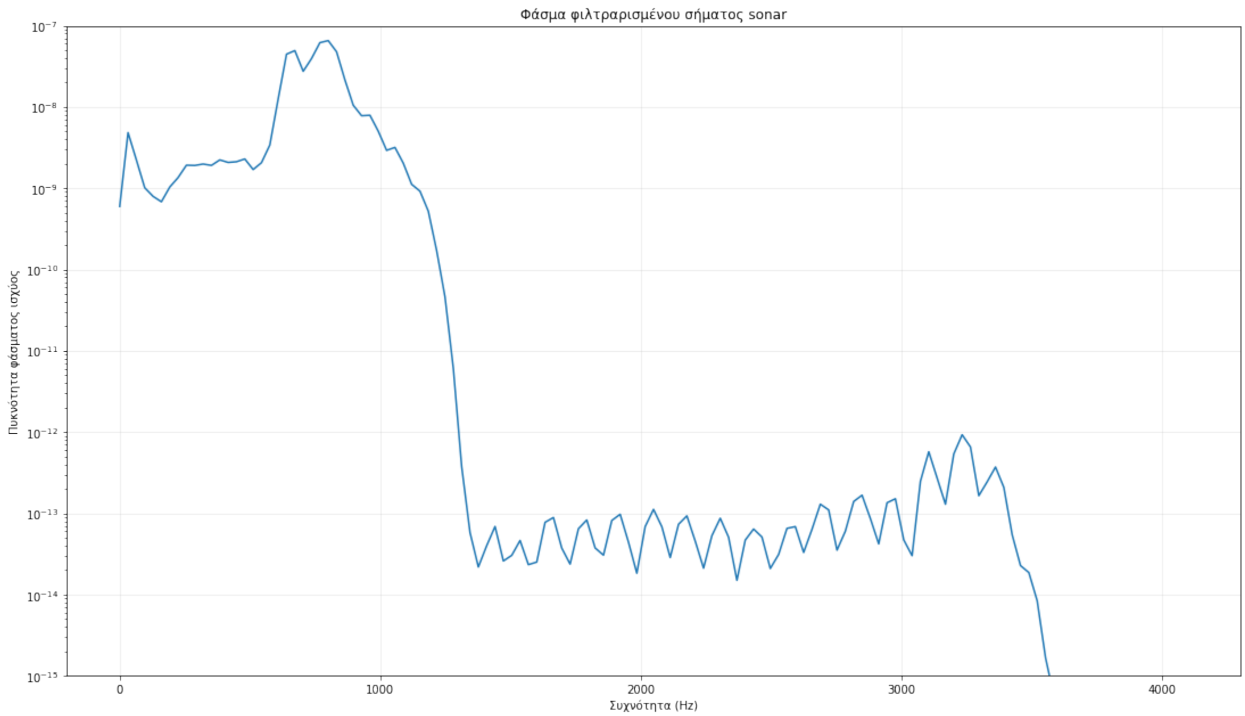
Εφαρμογή του φίλτρου

```
s_pm = signal.convolve(s,lpass,mode='same')/sum(lpass)

f, Pxx_den = signal.welch(s_pm, Fs, noverlap=128, nperseg=256)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

plt.title('Φάσμα φιλτραρισμένου σήματος sonar')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
```

(continues on next page)

(continued from previous page)

```
plt.ylabel('Πυκνότητα φάσματος ισχύος')
plt.ylim((1e-15,1e-7))

ax.semilogy(f, Pxx_den)
sd.play(s_pm,Fs)
```



Ζωνοπερατά φίλτρα

Με αναλυτικό υπολογισμό της κρουστικής απόκρισης και παράθυρο

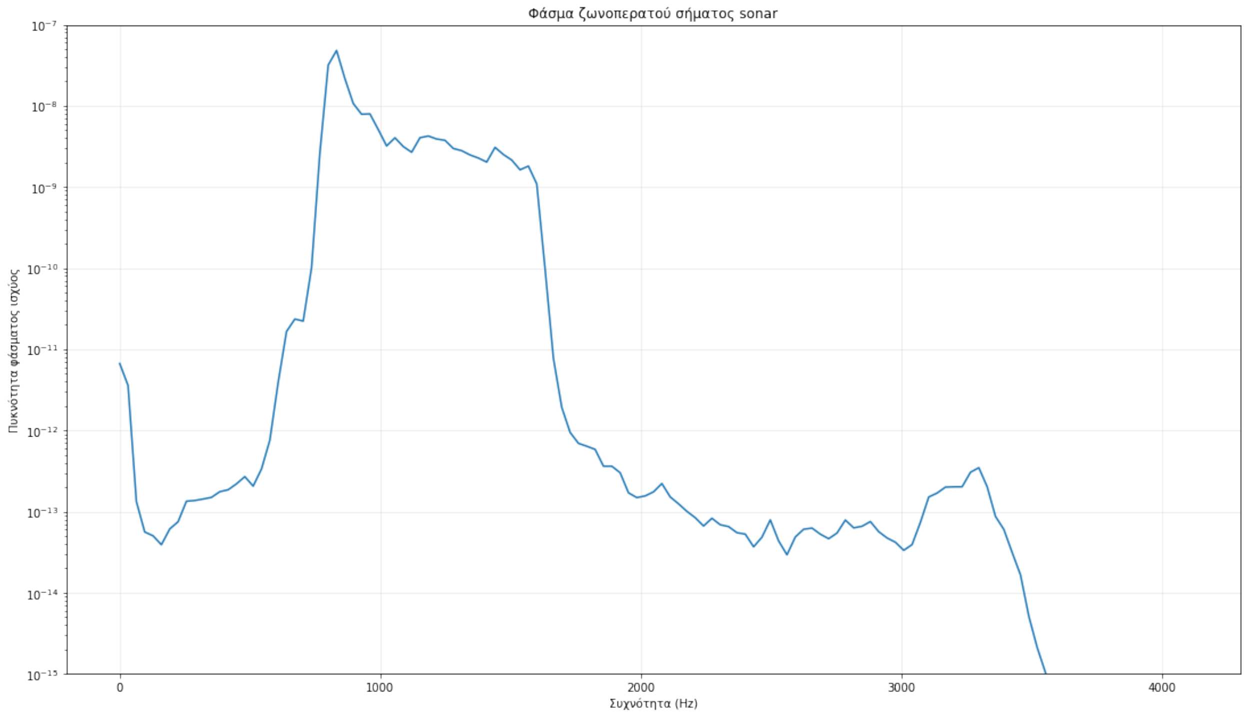
```
# Με αναλυτικό υπολογισμό της κρουστικής απόκρισης και παράθυρο kaiser
f1=800; f2=1600;
Ts=1/Fs;
f2m1=(f2-f1); f2p1=(f2+f1)/2; N=256
t=np.arange(-(N-1),N-1,2)*Ts/2
hbp=2/Fs*np.divide(np.multiply(np.cos(2*np.pi*f2p1*t),np.sin(np.pi*f2m1*t))/np.pi,t);
hbpw=np.multiply(hbp,signal.kaiser(len(hbp),5));

s_bp=signal.convolve(s,hbp,'same');
```

```
f, Pxx_den = signal.welch(s_bp, Fs, noverlap=128, nperseg=256)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)

plt.title('Φάσμα ζωνοπερατού σήματος sonar')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Πυκνότητα φάσματος ισχύος')
plt.ylim((1e-15,1e-7))
```

(continues on next page)

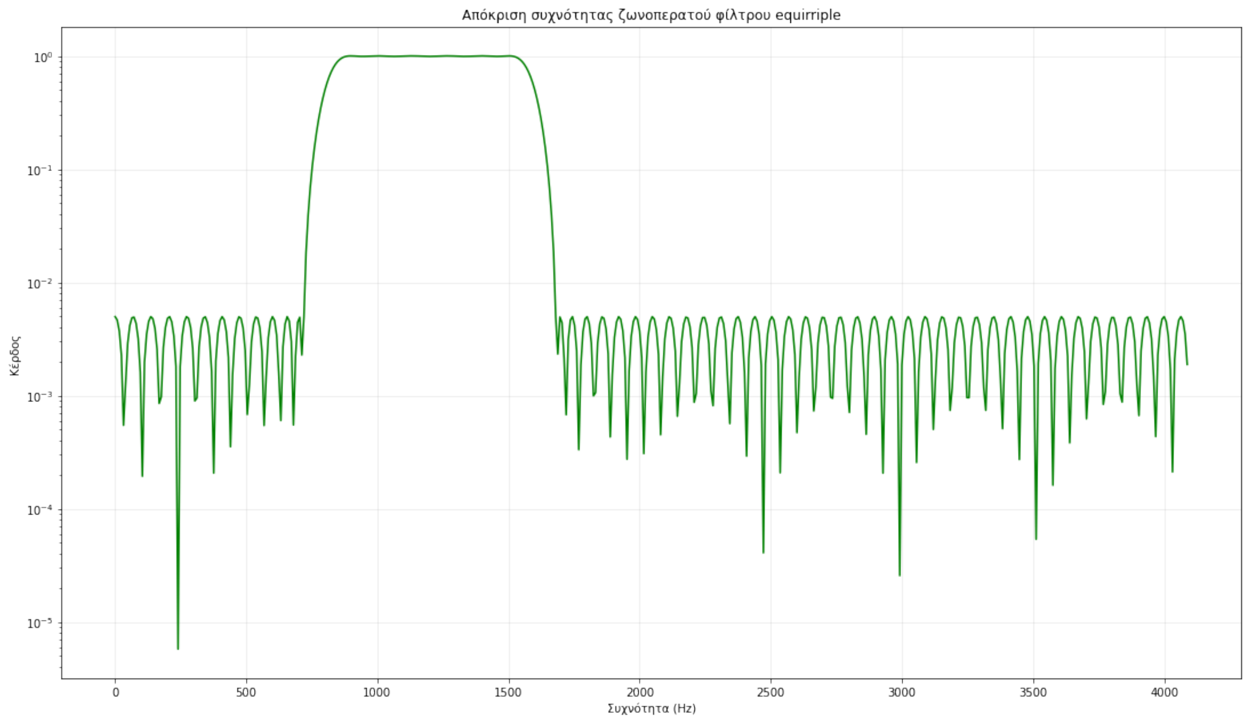
(continued from previous page)

```
ax.semilogy(f, Pxx_den)
sd.play(20*s_bp,Fs)
```



Ζωνοπερατό ισουψών κυματώσεων

```
bpass = signal.remez(128, [0, f1*0.9, f1*1.1, f2*0.95, f2*1.05, Fs/2], [0, 1, 0],␣
↪fs=Fs)

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)


plt.title('Απόκριση συχνότητας ζωνοπερατού φίλτρου equirriple')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')

freq,resp_pm = signal.freqz(bpass);
ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp_pm), 'g-')

plt.show()
```



Ζωνοπερατό φίλτρο με ζώνες διέλευσης (750 Hz, 950 Hz) και (3000 Hz, 3500 Hz)

Με αναλυτικό υπολογισμό της κρουστικής απόκρισης και παράθυρο

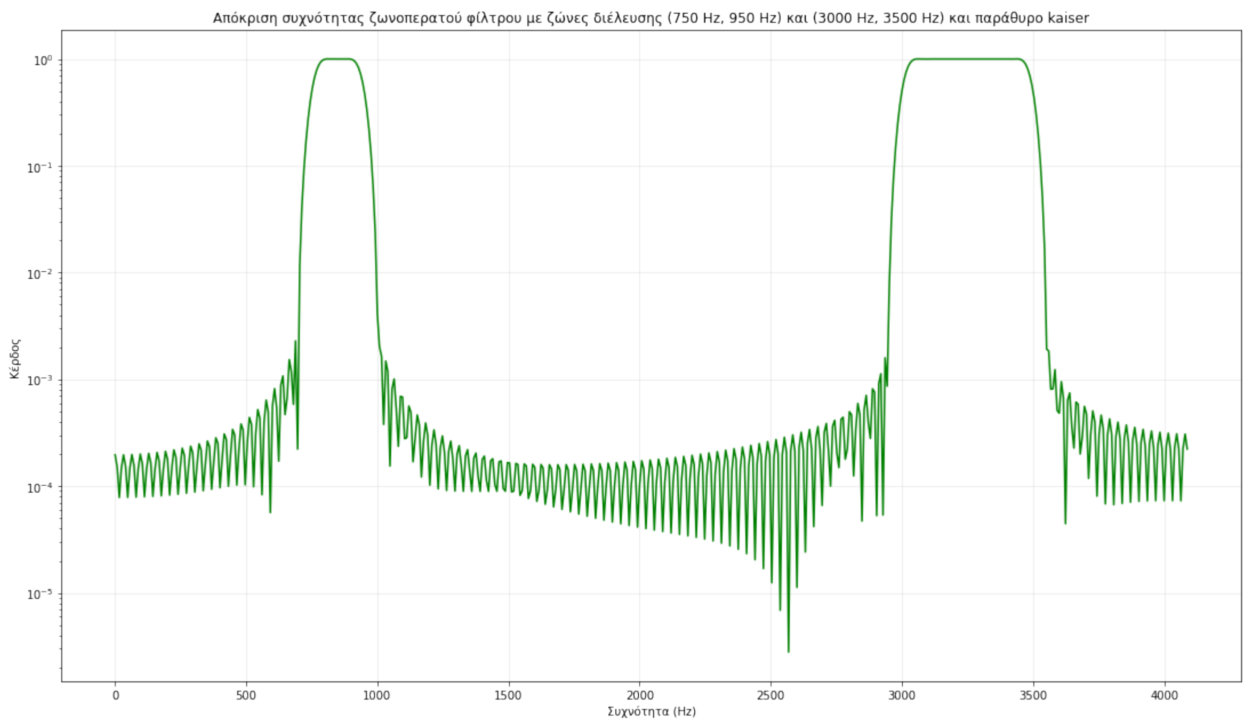
```
f1=750
f2=950
f3=3000
f4=3500
Ts=1/Fs
f2m1=(f2-f1)
f2p1=(f2+f1)/2
f4m3=(f4-f3)
f4p3=(f4+f3)/2
N=256
t=np.arange(-(N-1),N-1,2)*Ts/2;
band1=2/Fs*np.divide(np.multiply(np.cos(2*np.pi*f2p1*t),np.sin(np.pi*f2m1*t))/np.pi,t)
band2=2/Fs*np.divide(np.multiply(np.cos(2*np.pi*f4p3*t),np.sin(np.pi*f4m3*t))/np.pi,t)
hbp2=band1+band2
hbpw2=np.multiply(hbp2,signal.kaiser(len(hbp2),5));
s_bp2=signal.convolve(s,hbpw2,'same');
```

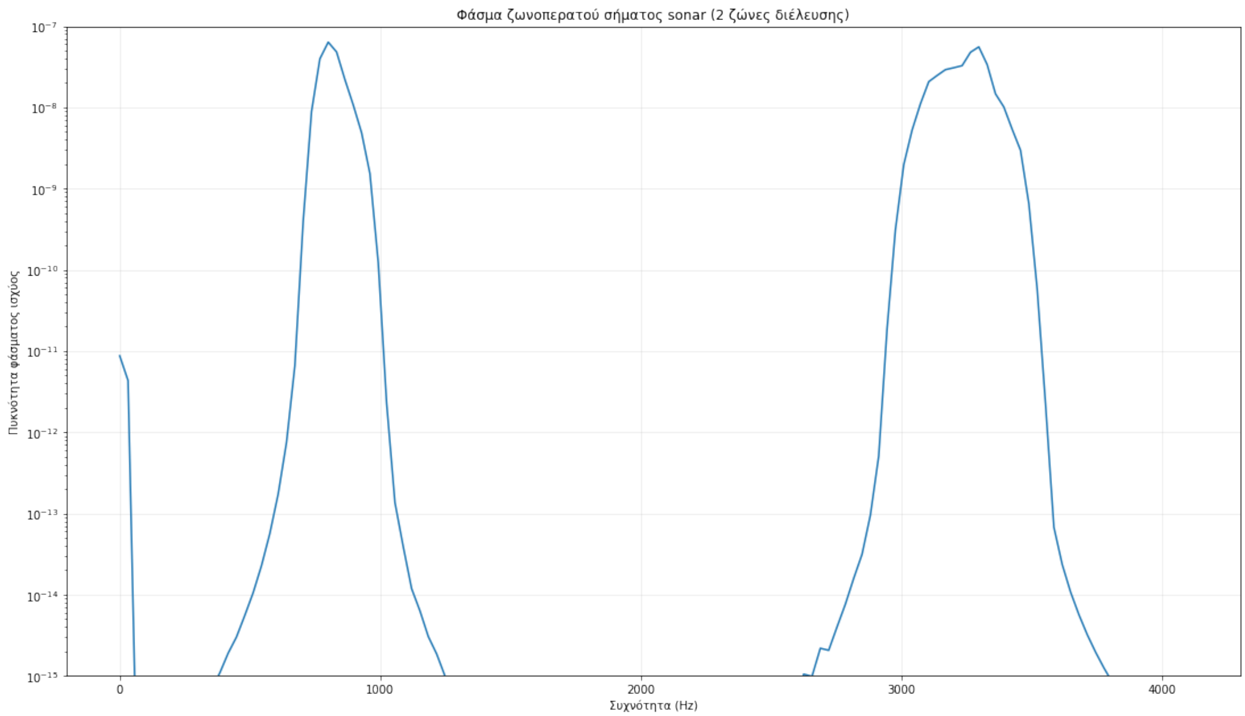
```
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)


plt.title('Απόκριση συχνότητας ζωνοπερατού φίλτρου με ζώνες διέλευσης (750 Hz, 950␣
↪Hz) και (3000 Hz, 3500 Hz) και παράθυρο kaiser')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')
```

(continues on next page)

(continued from previous page)

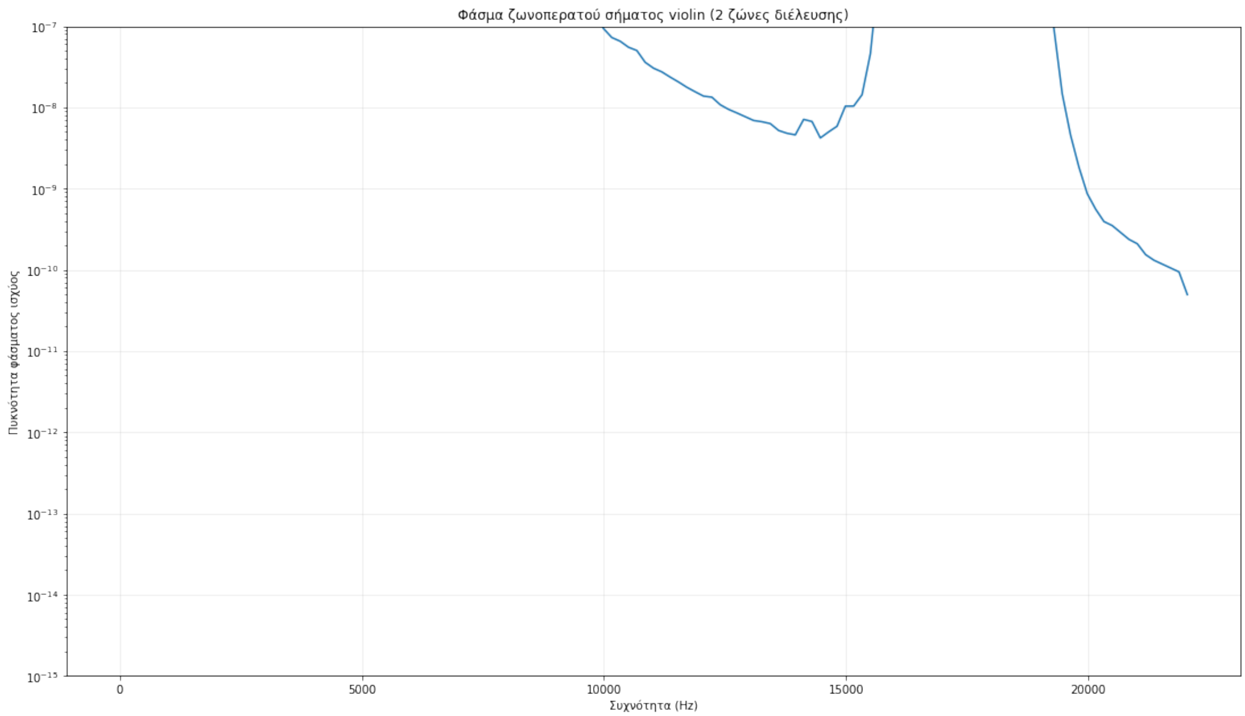
```
freq,resp_pm = signal.freqz(hbpw2);
ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp_pm), 'g-')

plt.show()

f, Pxx_den = signal.welch(s_bp2, Fs, noverlap=128, nperseg=256)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)


plt.title('Φάσμα ζωνοπερατού σήματος sonar (2 ζώνες διέλευσης)')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Πυκνότητα φάσματος ισχύος')
plt.ylim((1e-15,1e-7))

ax.semilogy(f, Pxx_den)
sd.play(20*s_bp2,Fs)
```





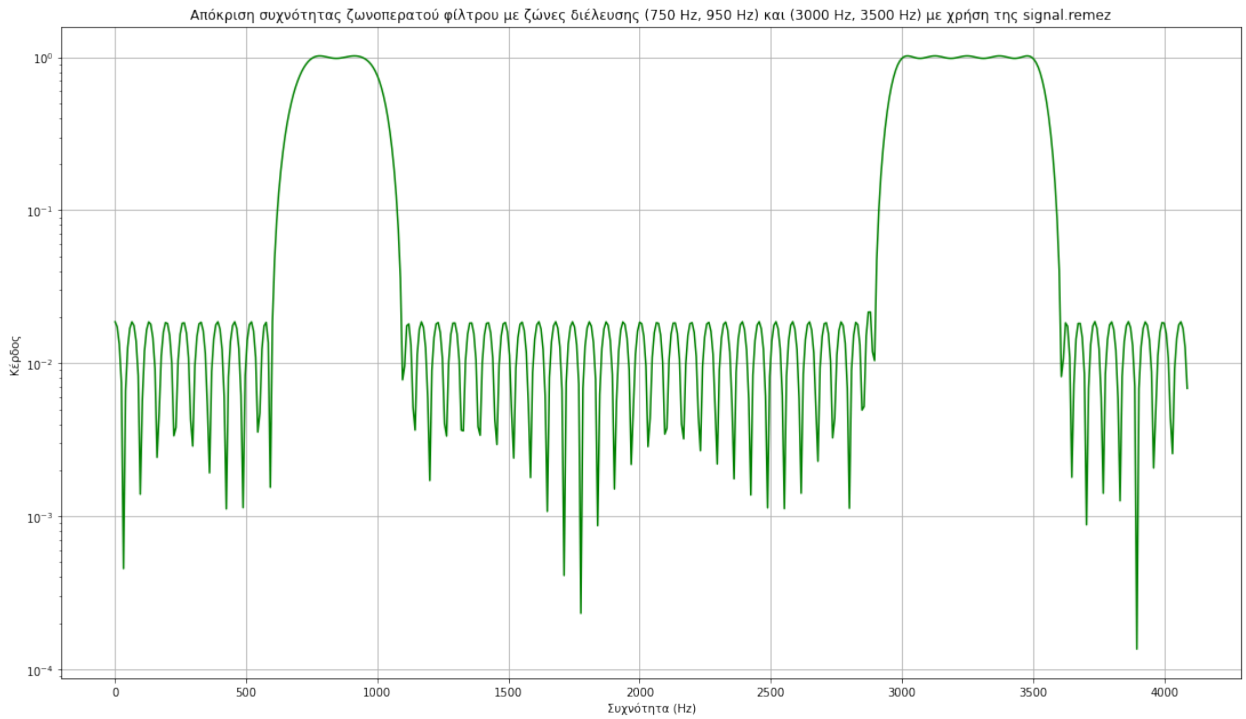
Εφαρμογή φίλτρου στο σήμα violin

```
s_viol_bp2=signal.convolve(s_viol,hbpw2,'same');

f, Pxx_den = signal.welch(s_viol_bp2, Fs_viol, noverlap=128, nperseg=256)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)


plt.title('Φάσμα ζωνοπερατού σήματος violin (2 ζώνες διέλευσης)')
plt.grid(alpha=0.25)
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Πυκνότητα φάσματος ισχύος')
plt.ylim((1e-15,1e-7))

ax.semilogy(f, Pxx_den)
sd.play(20*s_viol_bp2,Fs_viol)
```



Ζωνοπερατό ισουψών κυματώσεων με ζώνες διέλευσης (750 Hz, 950 Hz) και (3000 Hz, 3500 Hz)

```
bpass2 = signal.remez(128, [0, f1*0.8, f1, f2, 1.15*f2, f3*0.95, f3, f4, f4*1.03, Fs/
↪2],
                      [0, 1, 0, 1, 0], fs=Fs)
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)


plt.title('Απόκριση συχνότητας ζωνοπερατού φίλτρου με ζώνες διέλευσης (750 Hz, 950␣
↪Hz) και (3000 Hz, 3500 Hz) με χρήση της signal.remez')
plt.grid()
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')
freq,resp_pm = signal.freqz(bpass2)

ax.semilogy(0.5*Fs*freq/np.pi, np.abs(resp_pm), 'g-')


s_bpass2 = signal.convolve(s,bpass2,'same')
```



Μέρος 2 Ζωνοπερατό φίλτρο

Ζωνοπερατό φίλτρο με ζώνες διέλευσης (750 Hz, 950 Hz) και (3000 Hz, 3500 Hz)

Παρακάτω πραγματοποιείται ο σχεδιασμός ζωνοπερατού φίλτρου με ζώνες διέλευσης: (750 Hz, 950 Hz) και (3000 Hz, 3500 Hz) και η εφαρμογή του στο σήμα sonar. Χρησιμοποιήθηκαν δύο διαφορετικές μέθοδοι για τον σχεδιασμό αυτού του φίλτρου: η μέθοδος αναλυτικού υπλογισμού της κρουστικης απόκρισης με εφαρμογή παραθύρου και η μέθοδος ισοϋψών κυματώσεων. Κατά τον σχεδιασμό του φίλτρου με την μέθοδο ισοϋψών κυματώσεων, χρειάστηκε να γίνει κατάλληλη χρήση της συνάρτησης signal.remez για την οποία έπρεπε να οριστούν κατάλληλα τα όρια των ζώνων διέλευσης και αποκοπής ετσι ώστε να αποφευχθεί οποιοδήποτε πιθανό φαινόμενο απότομης μεταβολής της έντασης στην ζώνη διέλευσης.

```
import warnings
import sounddevice as sd
import scipy.io.wavfile
from scipy import signal
import scipy.io.wavfile
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
```

Σήμα sonar

```
# Ανάγνωση δειγμάτων σήματος από txt file
with open('sima.txt') as f:
    s = [float(x) for x in f]
s=np.array(s)
print('μέγεθος σήματος=', s.shape)
Fs=8192
sd.play(20*s,Fs)
```

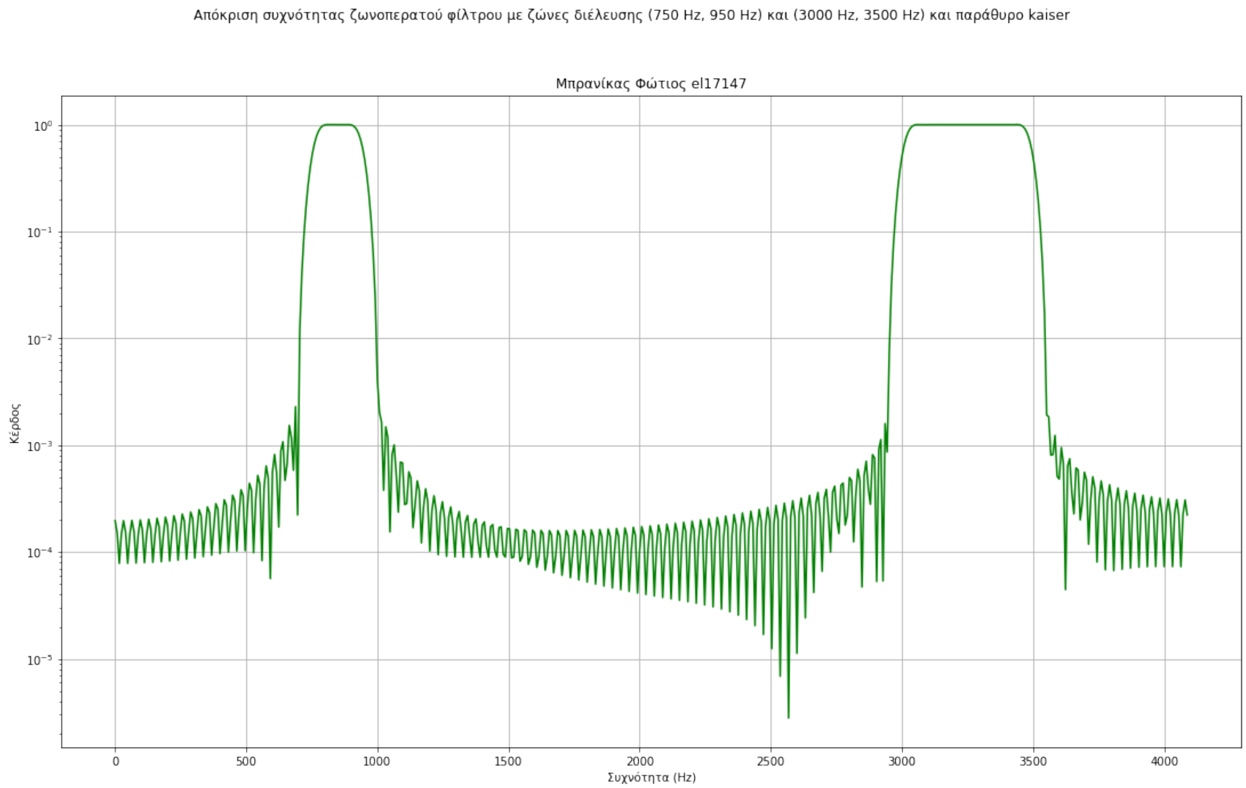
```
μέγεθος σήματος= (6565,)
```

Ζωνοπερατό φίλτρο με ζώνες διέλευσης (750 Hz, 950 Hz) και (3000 Hz, 3500 Hz)

```
f1=750
f2=950
f3=3000
f4=3500
Ts=1/Fs
f2m1=(f2-f1)
f2p1=(f2+f1)/2
f4m3=(f4-f3)
f4p3=(f4+f3)/2
N=256
```

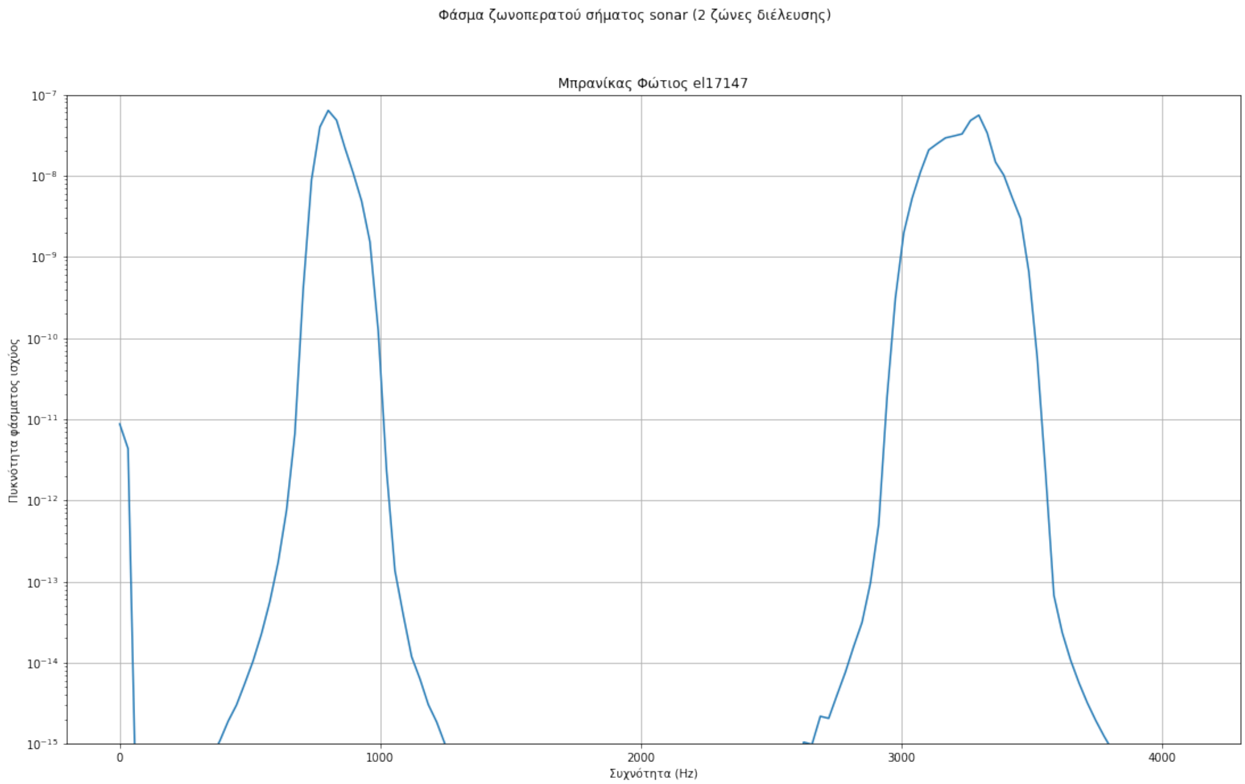
Σχεδιασμός φίλτρου με αναλυτικό υπολογισμό της κρουστικής απόκρισης και παράθυρο

```
t=np.arange(-(N-1),N-1,2)*Ts/2;
band1=2/Fs*np.divide(np.multiply(np.cos(2*np.pi*f2p1*t),np.sin(np.pi*f2m1*t))/np.pi,t)
band2=2/Fs*np.divide(np.multiply(np.cos(2*np.pi*f4p3*t),np.sin(np.pi*f4m3*t))/np.pi,t)
hbp2=band1+band2
hbpw2=np.multiply(hbp2,signal.kaiser(len(hbp2),5));
fig = plt.figure()
plt.suptitle('Απόκριση συχνότητας ζωνοπερατού φίλτρου με ζώνες διέλευσης (750 Hz, 950␣
↪Hz) και (3000 Hz, 3500 Hz) και παράθυρο kaiser')
plt.title('Μπρανίκας Φώτιος el17147')
plt.grid()
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')
fig.set_size_inches(18.5, 10.5)

freq,resp_pm = signal.freqz(hbpw2);
plt.semilogy(0.5*Fs*freq/np.pi, np.abs(resp_pm), 'g-')

plt.show()
```



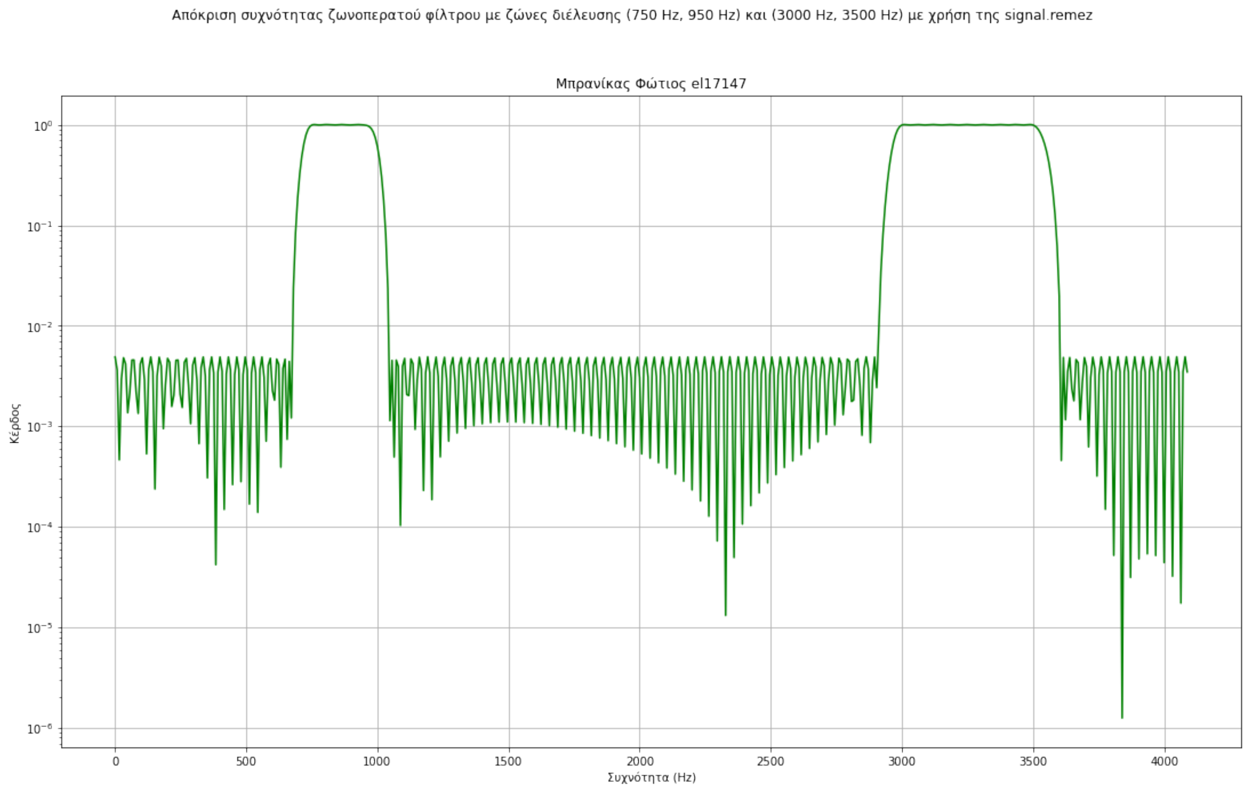
Εφαρμογή φίλτρου στο σήμα sonar

```
s_bp2=signal.convolve(s,hbpw2,'same');

f, Pxx_den = signal.welch(s_bp2, Fs, noverlap=128, nperseg=256)
fig = plt.figure()
plt.suptitle('Φάσμα ζωνοπερατού σήματος sonar (2 ζώνες διέλευσης)')
plt.title('Μπρανίκας Φώτιος el17147')
plt.grid()
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Πυκνότητα φάσματος ισχύος')
plt.ylim((1e-15,1e-7))
fig.set_size_inches(18.5, 10.5)

plt.semilogy(f, Pxx_den)
sd.play(20*s_bp2,Fs)
```



Σχεδιασμός Ζωνοπερατού φίλτρου ισουψών κυματώσεων με ζώνες διέλευσης (750 Hz, 950 Hz) και (3000 Hz, 3500 Hz)

```
bpass2 = signal.remez(256, [0, f1*0.9, f1, f2, 1.1*f2, f3*0.97, f3, f4, f4*1.03, Fs/2],
                      [0, 1, 0, 1, 0], fs=Fs)

fig = plt.figure()
plt.suptitle('Απόκριση συχνότητας ζωνοπερατού φίλτρου με ζώνες διέλευσης (750 Hz, 950 Hz) και (3000 Hz, 3500 Hz) με χρήση της signal.remez')
plt.title('Μπρανίκας Φώτιος el17147')
plt.grid()
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Κέρδος')
freq,resp_pm = signal.freqz(bpass2)
fig.set_size_inches(18.5, 10.5)

plt.semilogy(0.5*Fs*freq/np.pi, np.abs(resp_pm), 'g-')
plt.show()
```



Εφαρμογή φίλτρου στο σήμα sonar

```
s_bpass2 = signal.convolve(s,bpass2,'same')
f, Pxx_den = signal.welch(s_bpass2, Fs, noverlap=128, nperseg=256)
fig = plt.figure()
plt.suptitle('Φάσμα ζωνοπερατού σήματος sonar (2 ζώνες διέλευσης)')
plt.title('Μπρανίκας Φώτιος el17147')
plt.grid()
plt.xlabel('Συχνότητα (Hz)')
plt.ylabel('Πυκνότητα φάσματος ισχύος')
plt.ylim((1e-15,1e-7))
fig.set_size_inches(18.5, 10.5)

plt.semilogy(f, Pxx_den)
sd.play(20*s_bpass2,Fs)

for
```

```
  File "<ipython-input-7-9884b6f1db20>", line 15
    for
       ^
SyntaxError: invalid syntax
```