ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 5

ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ FOURIER ΚΑΙ ΣΥΝΕΛΙΞΗ ΣΤΟ ΠΕΔΙΟ ΤΗΣ ΣΥΧΝΟΤΗΤΑΣ

Στην εργαστηριακή άσκηση 4 μελετήσαμε τη σχέση εισόδου–εξόδου ενός γραμμικού χρονικά αμετάβλητου (ΓΧΑ) συστήματος, που, στο πεδίο του χρόνου περιγράφεται ως η συνέλιξη της εισόδου, $x(t)$, και της κρουστικής απόκρισης, $h(t)$, του συστήματος:

$$y(t) = h(t) * x(t) = x(t) * h(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau$$

Αν και η διαδικασία εκτέλεσης της παραπάνω πράξης που περιγράψαμε είναι συγκεκριμένη και μπορεί να υλοποιηθεί στο MATLAB μέσω μιας προσέγγισης από συνέλιξη διακριτού χρόνου (**conv**), ο αντίστοιχος υπολογισμός στο πεδίο της συχνότητας, αυτός δηλαδή που περιλαμβάνει τους μετασχηματισμούς Fourier (MF) των εμπλεκόμενων σημάτων, είναι σημαντικά απλούστερος, τουλάχιστον στην περιγραφή του:

$$Y(\Omega) = H(\Omega)X(\Omega)$$

Η πράξη $*$, δηλαδή, ***μετασχηματίζεται σε πολλαπλασιασμό***. Θα πρέπει όμως πρώτα να υπολογιστούν οι MF $H(\Omega), X(\Omega)$.

Πριν πάμε να μελετήσουμε το πώς μπορεί ένας MF να υπολογιστεί στο MATLAB, αξίζει πρώτα να δούμε μέσω παραδειγμάτων τη φυσική σημασία του πολλαπλασιαστικού παράγοντα $H(\Omega)$.

Ας θυμηθούμε ότι η απόκριση του συστήματος σ' ένα μιγαδικό εκθετικό σήμα, $x(t) = e^{j\Omega_0 t}$, είναι

$$y(t) = H(\Omega_0)x(t)$$

όπου $$H(\Omega_0) = \int_{-\infty}^{\infty} h(t) e^{-j\Omega_0 t} dt$$

Το παραπάνω μας λέει ότι τέτοια σήματα, που αποτελούνται δηλαδή από μια μόνο συχνότητα, Ω_0 (και λέγονται σήματα *απλής* συχνότητας), «περνούν» από το ΓΧΑ με μόνο τη μεταβολή στο πλάτος τους, το οποίο είναι πλέον πολλαπλασιασμένο με το (γενικά μιγαδικό) παράγοντα $H(\Omega_0)$.

Αυτό, σε συνδυασμό με το γεγονός ότι οποιοδήποτε σήμα $x(t)$ μπορεί να ειδωθεί ως άθροισμα σημάτων απλής συχνότητας, συνεπάγεται ότι με τη βοήθεια ενός ΓΧΑ μπορούμε να αλλοιώσουμε το πλάτος κάθε συχνότητας του $x(t)$.

Για **<u>παράδειγμα</u>**, αν πρέπει να μηδενιστεί η συχνότητα Ω_0 στο $x(t)$, αρκεί να το «περάσουμε» μέσα από ένα ΓΧΑ σύστημα με *απόκριση συχνότητας* $H(\Omega)$ τέτοια ώστε $H(\Omega_0) = 0$. Αν δεν θέλουμε να πειράξουμε τις άλλες συχνότητες, θα πρέπει $H(\Omega) = 1, \forall \Omega \neq \Omega_0$.

Γενικότερα, μπορούμε, με κατάλληλη επιλογή της μορφής της $H(\Omega)$ (άρα και της $h(t)$), να αλλοιώσουμε το MF του $x(t)$ (άρα και το ίδιο το $x(t)$) σύμφωνα με τις ανάγκες μας. Στο παραπάνω παράδειγμα, το ΓΧΑ σύστημα «αφήνει» να περάσουν απ' αυτό όλες οι συχνότητες εκτός μιας, που την «κόβει».[1] Κάτι τέτοιο μας θυμίζει ένα φίλτρο, που σκοπό έχει να κατακρατήσει μέρος αυτού που του βάζουμε και ν' αφήσει να περάσει το υπόλοιπο. Σύμφωνα μ' αυτή την αναλογία, ένα ΓΧΑ σύστημα καλείται και *φίλτρο* (χωρίς απαραίτητα να «κόβει» κάποιες συχνότητες). Διακρίνουμε τέσσερα είδη φίλτρων:

1. **Κατωπερατό** (Lowpass): Περνάει χαμηλές συχνότητες και κόβει υψηλές
2. **Ανωπερατό** (Highpass): Περνάει υψηλές και κόβει χαμηλές συχνότητες
3. **Ζωνοπερατό** (Bandpass): Περνάει ένα διάστημα συχνοτήτων και κόβει όλες τις άλλες
4. **Ζώνης αποκοπής** (Bandstop): Κόβει ένα διάστημα συχνοτήτων και αφήνει όλες τις άλλες να περάσουν.

[1] Στην πράξη, βέβαια, κάτι τέτοιο δεν μπορεί να υλοποιηθεί ακριβώς. Αυτό που θα γίνει είναι να μηδενιστεί η Ω_0, αλλά και να μειωθούν τα πλάτη των γειτονικών της συχνοτήτων.

Ας δούμε ένα παράδειγμα φίλτρου ζώνης αποκοπής. Έχει την ακόλουθη απόκριση συχνότητας:

$$H(\Omega) = \frac{0.64 - \Omega^2}{(0.64 - \Omega^2) + j1.6\Omega} \cdot \frac{4 - \Omega^2}{(4 - \Omega^2) + j4\Omega}$$

ή γράφοντας την ως ρητή συνάρτηση του $j\Omega$,

$$H(\Omega) = \frac{0.64 + (j\Omega)^2}{0.64 + 1.6j\Omega + (j\Omega)^2} \cdot \frac{4 + (j\Omega)^2}{4 + 4j\Omega + (j\Omega)^2}\,.$$

Η παραπάνω μορφή της απόκρισης συχνότητας, δηλαδή το γινόμενο ρητών παραγόντων, είναι πολύ συχνή στην πράξη και υπαγορεύει μια υλοποίηση του αντίστοιχου φίλτρου ως σύνδεσης σε σειρά των δύο φίλτρων που αντιστοιχούν στους δύο παράγοντές της. Μια τέτοια υλοποίηση λέγεται ***σύνδεση σε σειρά* ή *cascade.*** Ο κάθε παράγοντας λέγεται και *βαθμίδα (stage).*

Για να υπολογίσουμε την $H(\Omega)$ στο MATLAB, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **freqs**, που επιστρέφει την απόκριση συχνότητας δοσμένου *ρητού* συστήματος.

Πώς θα δώσουμε το σύστημα; Προσδιορίζοντας τους συντελεστές των πολυωνύμων ως προς $j\Omega$ στον αριθμητή και τον παρονομαστή του.

Έτσι, ο αριθμητής της πρώτης βαθμίδας έχει συντελεστές 1, 0 και 0.64, που πολλαπλασιάζουν το $(j\Omega)^2$, το $j\Omega$, και το $(j\Omega)^0$, αντίστοιχα. Παρόμοια για τα υπόλοιπα πολυώνυμα. Προκειμένου να βρούμε τον ολικό αριθμητή και τον ολικό παρονομαστή θα χρειαστούμε και τη συνάρτηση **conv** για να υλοποιήσουμε τον πολλαπλασιασμό κατά μέλη. Ας δούμε πώς:

Παρατήρηση: Γράφετε μόνο τις εντολές και όχι τα σχόλια που ακολουθούν με το σύμβολο %

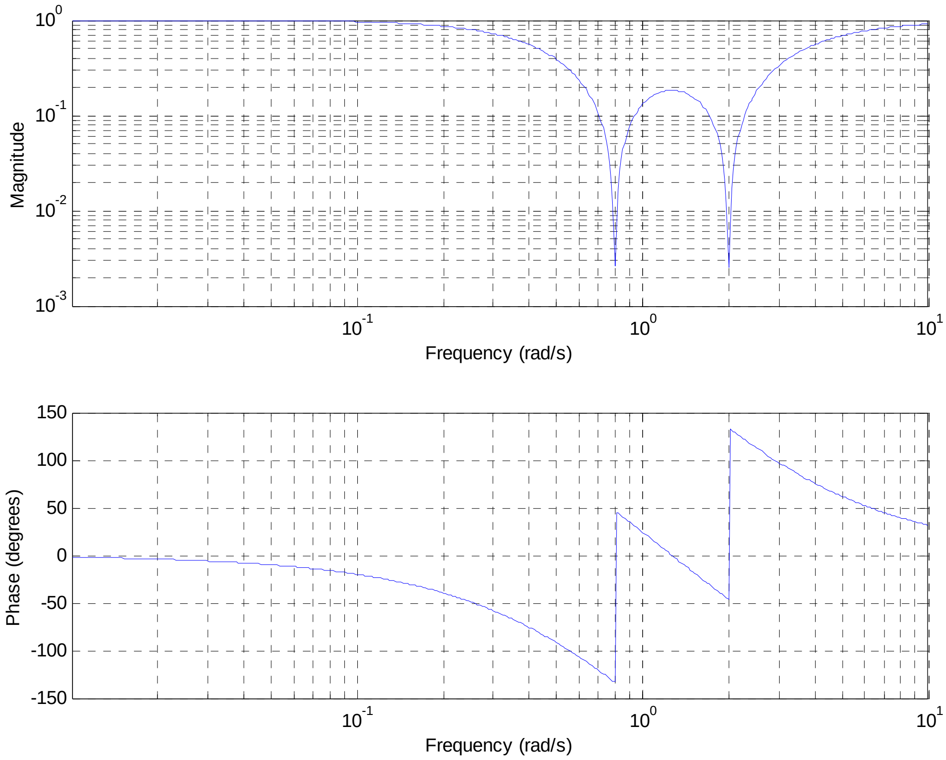
```
>> num1=[1 0 0.64]; % Αριθμητής πρώτης βαθμίδας

>> num2=[1 0 4]; % Αριθμητής δεύτερης βαθμίδας

>> den1=[1 1.6 0.64]; % Παρονομαστής πρώτης βαθμίδας
```

```
>> den2=[1 4 4]; % Παρονομαστής δεύτερης βαθμίδας

>> num=conv(num1,num2); % Συνολικός αριθμητής

>> den=conv(den1,den2); % Συνολικός παρονομαστής

>>  [H,w]=freqs(num,den,400);  %  Υπολογισμός  της
H(Ω)

                              % σε 400 συχνότητες w
```

Αν καλέσουμε την **freqs** απλά ως

>> **`freqs(num,den,400)`**

χωρίς να της ζητήσουμε να επιστρέψει κάτι, τότε γίνεται αυτόματα η γραφική παράσταση του μέτρου και της φάσης (σε μοίρες) της $H(\Omega)$, όπως φαίνεται στο παρακάτω σχήμα:[2]

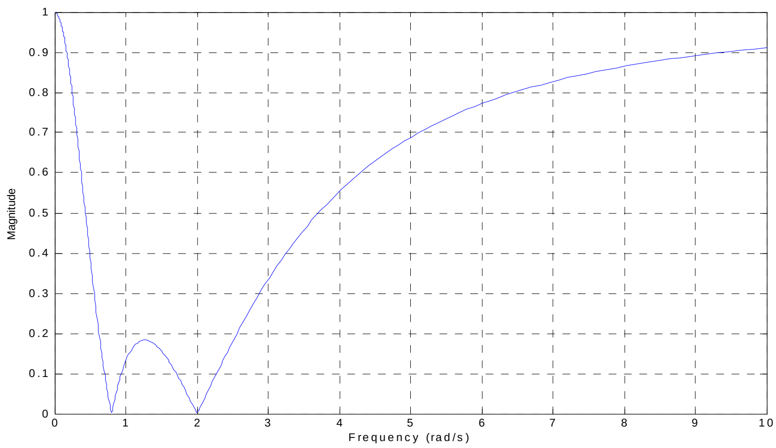


[2] Ένα τέτοιο σχήμα καλείται και *Bode plot* και μπορεί να παραχθεί και με τη βοήθεια της συνάρτησης `bode`.

Η συχνότητα είναι σε **ακτίνια/δευτερόλεπτο** (rad/s) και δίνεται σε λογαριθμική κλίμακα. Αν θέλουμε π.χ. μόνο το μέτρο, και μάλιστα με τη συχνότητα σε γραμμική κλίμακα, τότε μπορούμε να το παραστήσουμε με τις εντολές

```
>> plot(w,abs(H))

>> grid

>> xlabel('Frequency (rad/s)')

>> ylabel('Magnitude')
```

Το αποτέλεσμα φαίνεται παρακάτω:



Όπως βλέπουμε, η **ζώνη αποκοπής**[3] βρίσκεται **μεταξύ των 0.8 και των 2 rad/s.** Επίσης, όχι μόνο αυτές αλλά και γειτονικές τους συχνότητες εξασθενίζουν μ' αυτό το φίλτρο. Κάτι τέτοιο είναι αναπόφευκτο για φίλτρα που είναι πρακτικά υλοποιήσιμα.

Ένας δεύτερος τρόπος υλοποίησης είναι ο εξής :

```
>>  w=0:0.1:10               % ορισμός διανύσματος συχνοτήτων
>> H=((0.64+(j*w).^2).*(4+(j*w).^2)) ./ (  (0.64 +1.6*j*w+(j*w).^2) .*
(4+4*j*w + (j*w).^2)) ;   % ορισμός της απόκρισης συχνότητας H(jω)
>>  plot(w,abs(H)) % σχεδίαση μέτρου
>> plot(w,angle(H) % σχεδίαση φάσης
```

[3] Λέγεται και *notch*, ειδικά όταν είναι στενή.

Έχοντας υπολογίσει την απόκριση συχνότητας σ' ένα διάστημα συχνοτήτων, μπορούμε να δούμε πια την απόκριση του φίλτρου σε διάφορα σήματα απλής συχνότητας.

<u>**Ας θεωρήσουμε το σήμα**</u> $x(t) = \cos(\Omega_0 t) = \dfrac{e^{j\Omega_0 t} + e^{-j\Omega_0 t}}{2}$. Τότε, σύμφωνα με τα παραπάνω, η έξοδος θα είναι $y(t) = \dfrac{1}{2}H(\Omega_0)e^{j\Omega_0 t} + \dfrac{1}{2}H(-\Omega_0)e^{-j\Omega_0 t}$. Αν $H(\Omega) = |H(\Omega)|e^{j\phi(\Omega)}$, και λάβουμε υπόψη το ότι οι συντελεστές του φίλτρου είναι πραγματικοί, τότε το μέτρο και η φάση της απόκρισης συχνότητας θα είναι άρτια και περιττή συνάρτηση, αντίστοιχα. Έτσι,

$$y(t) = |H(\Omega_0)|\left\{\frac{1}{2}e^{j[\phi(\Omega_0)+\Omega_0 t]} + \frac{1}{2}e^{-j[\phi(\Omega_0)+\Omega_0 t]}\right\}$$, δηλαδή

$$y(t) = |H(\Omega_0)|\cos[\Omega_0 t + \phi(\Omega_0)].$$

Ας δοκιμάσουμε με το σήμα $x(t) = \cos(0.8t)$ **:**

```
>> [m,i1]=min(abs(w-0.8)); % Βρίσκει ποιο στοιχείο του w είναι
                           % πιο κοντά στη συχνότητα 0.8 rad/s.

>> t=[0:0.01:200];

>> x1=cos(w(i1)*t); % Σήμα εισόδου

>> y1=abs(H(i1))*cos(w(i1)*t+angle(H(i1))); % Σήμα εξόδου

>> subplot(211)

>> plot(t,x1)

>> xlabel('t')

>> ylabel('x_{1}(t)')

>> subplot(212)
```

```
>> plot(t,y1)

>> xlabel('t')

>> ylabel('y_{1}(t)')
```

Σχεδιάστε το αποτέλεσμα που θα πάρετε

Τι παρατηρείτε ως προς την εξασθένιση του πλάτους του σήματος;

…………………………………………………………………..

Τι παρατηρείτε ως προς την αλλαγή στη συχνότητα του σήματος;

……………………………………………………………………

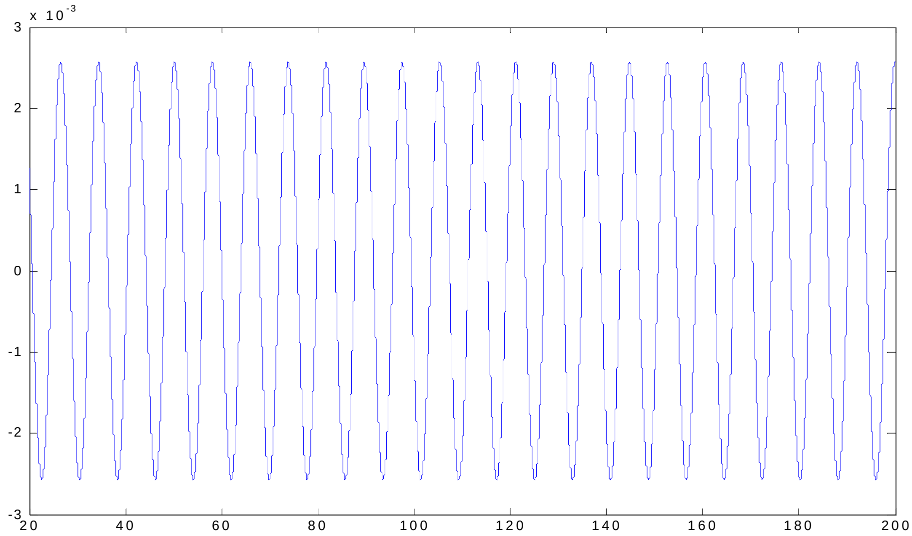
Ένας άλλος τρόπος να υπολογίσουμε την απόκριση του συστήματος, που εφαρμόζεται σε οποιοδήποτε σήμα εισόδου, όχι μόνο σε απλής συχνότητας, είναι μέσω της χρήσης της συνάρτησης **lsim**. Αυτή προσομοιώνει (simulation) ΓΧΑ σύστημα για δοσμένη είσοδο και χρονικό διάστημα. Στο παραπάνω παράδειγμα θα την εφαρμόζαμε ως εξής:

```
>> y1=lsim(num,den,x1,t);
```

```
>> plot(t,y1)
```

Το αποτέλεσμα είναι:

Παράξενο έτσι; Δεν μοιάζει καθόλου με το προηγούμενο αποτέλεσμα. Κι όμως. Ας δούμε το παραπάνω σήμα μετά τη στιγμή 20:



Στις αρχικές χρονικές στιγμές το σύστημα βρίσκεται σε μια **μεταβατική** (transient) **κατάσταση** και θα πρέπει να το δούμε όταν βρεθεί στη **μόνιμη κατάσταση** λειτουργίας του (steady state). Όντως, τα αποτελέσματα είναι τότε τα ίδια και στις δυο περιπτώσεις.

- Αν το σήμα εισόδου είναι γραμμικός συνδυασμός σημάτων απλής συχνότητας, στις συχνότητες $\Omega_0, \Omega_1, \Omega_2, \ldots$, δηλαδή $x(t) = A_0 e^{j\Omega_0 t} + A_1 e^{j\Omega_1 t} + A_2 e^{j\Omega_0 t} + \cdots$, τότε η έξοδος του φίλτρου θα είναι ο ίδιος γραμμικός συνδυασμός των αντίστοιχων αποκρίσεων:

$$y(t) = H(\Omega_0)A_0 e^{j\Omega_0 t} + H(\Omega_1)A_1 e^{j\Omega_1 t} + H(\Omega_2)A_2 e^{j\Omega_0 t} + \cdots$$

Γενικά ένα σήμα μπορεί να γραφεί έτσι (ίσως με άπειρες συχνότητες) όπου τα πλάτη A_i συνδέονται με το MF του. Έτσι καταλήγουμε στη γνωστή μας θεμελιώδη σχέση

$$Y(\Omega) = H(\Omega)X(\Omega)$$

Για να τη χρησιμοποιήσουμε στην πράξη, θα πρέπει να ξέρουμε να υπολογίζουμε το MF ενός σήματος συνεχούς χρόνου.

Η ακόλουθη function υπολογίζει το MF με τον παραπάνω τρόπο, στο διάστημα συχνοτήτων $\left[-\frac{\pi}{T}, \frac{\pi}{T}\right]$. Η ιδιαιτερότητά της είναι ότι το σήμα δίνεται μέσω της συνάρτησης MATLAB που το υλοποιεί. Αυτή θα πρέπει να έχει συγκεκριμένο τρόπο κλήσης.

Δημιουργήστε ένα νέο **m-file** που θα το ονομάσετε **ctft.m** (δηλαδή Μετασχηματισμός Fourier συνεχούς χρόνου) και θα το αποθηκεύσετε στον κατάλογο work του Matlab .

```
function [X,w]=ctft(x)

% CTFT  Continuous-time Fourier transform

%

% Usage: [X,w]=ctft(x)

% x must be a string containing the call to the function

% computing the signal.

% Call syntax: [y,T,N,t1]=func(parameters).


x1=['[y,T,N,t1]=',x,';'];
```

```
eval(x1); % Υπολογισμός της έκφρασης στο string
x1

t2=t1+(N-1)*T;

t=t1:T:t2;

W=2*pi/(N*T);

w=[-(N/2)+1:(N/2)]*W;

X=T*exp(-j*w*t1).*fft(y);

subplot(221)

plot(t,y)

title('Signal')

subplot(223)

plot(w,abs(fftshift(X)))

title('Fourier transform magnitude')

subplot(224)

plot(w,angle(fftshift(X)))

title('Fourier transform phase')
```

Επίσης δημιουργήστε ένα **m-file** που θα το ονομάσετε **cosine.m** (δηλ. συνημίτονο) και θα το αποθηκεύσετε στον κατάλογο work του Matlab .

Για το σήμα $x(t) = \cos(\Omega_0 t)$ η συνάρτηση cosine θα είναι:

```
function [x,T,N,t1]=cosine(w0,t1,t2,N)

% COSINE    Computes cos(w0*t) for given w0

% Usage: [x,T,N,t1]=cosine(w0,t1,t2,N)

% Παρατηρείστε ότι πρέπει να ξαναεπιστρέφεται το
N και το t1
```

```
% ώστε να είναι διαθέσιμα στην ctft.

T=(t2-t1)/N;

t=t1:T:t1+(N-1)*T;

x=cos(w0*t);
```

Πληκτρολογήστε στο Matlab

```
>> [X,w]=ctft('cosine(5,0,15,300)');
```

Τα αποτελέσματα φαίνονται στο παρακάτω σχήμα.

Το Matlab δίνει τη δυνατότητα να βρούμε απευθείας το μετασχηματισμό Fourier μιας συνάρτησης με την εντολή **fourier ( ).**

Το Matlab επίσης δίνει τη δυνατότητα να υπολογίσουμε και τον αντίστροφο μετασχηματισμό Fourier μιας συνάρτησης με την εντολή **ifourier** ( ).

Σύνταξη εντολής fourier:

```
Σειρά fourier = fourier (συνάρτηση)
```

Σύνταξη εντολής ifourier:

αντίστροφος μετασχηματισμός fourier = ifourier (μετασχηματισμένη fourier)

<u>Παράδειγμα</u>

Να βρεθεί ο μετασχηματισμός Fourier της συνάρτησης $f(t)=e\text{-}x^2$ και στη συνέχεια ο αντίστροφος μετασχηματισμός Fourier.

Ο απαιτούμενος κώδικας είναι:

```
>> syms x

>> f=exp(-x^2);

>> fourier(f)

ans =

        ..........................................................................

>> ifourier(ans)

ans =

        .......................................................................
```

<u>Παράδειγμα</u>

Να βρεθεί ο μετασχηματισμός Fourier της συνάρτησης f(t)=exp(-|x|) και στη συνέχεια ο αντίστροφος μετασχηματισμός Fourier της συνάρτησης f(t).

Ο απαιτούμενος κώδικας είναι:

```
>> syms  x

>> f=exp(-abs(x));

>> fourier(f)

ans =

    ……………………………………………………………

>> fourier(f, u)

ans =

     ……………………………………………………………

>> syms  x

>> f=exp(-abs(x));

>> F=ifourier (f)

      F =

          …………………………………………………
```