Τηλεπικοινωνιακά Συστήματα

Διάλεξη 2η

Θωμάς Καμαλάκης

Χαροκόπειο Πανεπιστήμιο Αθηνών

Οκτώβριος 2020

Περιεχόμενα

Η έννοια του σήματος

- Στα συστήματα επικοινωνιών μεταδίδουμε αναλογικά και ψηφιακά *σήματα*.
- Ενα σήμα $v(t)$ είναι μία μεταβολή ενός μεγέθους v (π.χ. τάση) σε συνάρτηση με το χρόνο t
- Αναλογικό σήμα είναι ένα σήμα που μπορεί να πάρει *οποιαδήποτε τιμή* μέσα σε ένα διάστημα τιμών
- Ψηφιακό σήμα είναι ένα σήμα που μπορεί να πάρει *μόνο* κάποιες τιμές.
- Παράδειγμα αναλογικού σήματος, $x(t) = A\cos(2\pi f_0 t)$.
- Παράδειγμα ψηφιακού σήματος είναι μία bit-σειρά από 0 και 1.

Ο μετασχηματισμός Fourier

- Το φάσμα $X(f)$ ενός σήματος αναπαριστά το σήμα στο πεδίο των συχνοτήτων.

$$X(f) = \int_{-\infty}^{+\infty} x(t) \mathrm{e}^{-j2\pi f t} \mathrm{d}t \tag{1}$$

$$x(t) = \int_{-\infty}^{+\infty} X(f) \mathrm{e}^{j2\pi f t} \mathrm{d}f \tag{2}$$

- Από το σήμα $x(t)$ μπορούμε να υπολογίσουμε το φάσμα $X(f)$.
- Από το φάσμα $X(f)$ μπορούμε να υπολογίσουμε το σήμα $x(t)$.

Παράδειγμα

Έστω ένας παλμός $x(t)$ που είναι ίσος με 1 όταν $-T_1/2 \leq t \leq T_1/2$ και ίσος με μηδέν διαφορετικά.

$$X(f) = \int_{-\infty}^{+\infty} x(t)\mathrm{e}^{-j2\pi ft}\mathrm{d}t = \int_{-T_1/2}^{+T_1/2} \mathrm{e}^{-j2\pi ft}\mathrm{d}t =$$

$$= \frac{\mathrm{e}^{j\pi fT_1} - \mathrm{e}^{-j\pi fT_1}}{2\pi f} = \frac{\sin(\pi fT_1)}{j\pi f} = T_1\frac{\sin(\pi fT_1)}{\pi fT_1} = T_1\mathrm{sinc}(fT_1) \quad (3)$$

Παράδειγμα

Έστω ένας παλμός $x(t)$ που είναι ίσος με 1 όταν $-T_1/2 \leq t \leq T_1/2$ και ίσος με μηδέν διαφορετικά.

$$X(f) = \int_{-\infty}^{+\infty} x(t)\mathrm{e}^{-j2\pi ft}\mathrm{d}t = \int_{-T_1/2}^{+T_1/2} \mathrm{e}^{-j2\pi ft}\mathrm{d}t =$$

$$= \frac{\mathrm{e}^{j\pi fT_1} - \mathrm{e}^{-j\pi fT_1}}{2\pi f} = \frac{\sin(\pi fT_1)}{j\pi f} = T_1\frac{\sin(\pi fT_1)}{\pi fT_1} = T_1\mathrm{sinc}(fT_1) \quad (4)$$

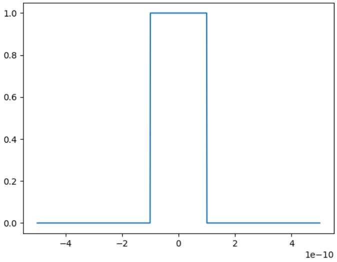
όπου

$$\mathrm{sinc}(x) = \begin{cases} 1 & , x = 0 \\ \frac{\sin(\pi x)}{\pi x} & , x \neq 0 \end{cases} \quad (5)$$

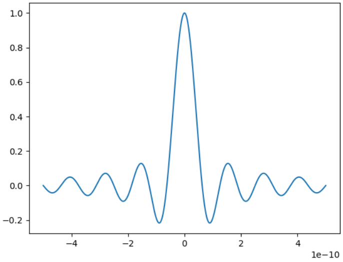
Ένα πρώτο παράδειγμα Python - plotsinc.py

```python
import numpy as np
import matplotlib.pyplot as plt

# Time axis
Npt = 1000                          # number of points in the time axis
T = 1e-9                            # total signal duration [s]
T1 = 0.2e-9                         # pulse "on" duration [s]
t = np.linspace(-T/2, T/2, 1000)    # time axis

# Frequency axis
Npf = 1000                          # number of points in the frequency axis
Fmax = 40e9                         # maximum frequency
f = np.linspace(-Fmax, Fmax, 1000)  # frequency axis

# Time signal
x = np.zeros(t.size)
i = 0

for tm in t:
    if np.abs(tm) <= T1/2.0:
        x[i] = 1
    i = i + 1

X = np.sinc( f * T1 )

plt.close('all')
plt.figure(1)
plt.plot(t, x)
plt.savefig
plt.figure(2)
plt.plot(t, X)
```

Ένα πρώτο παράδειγμα Python - plotsinc.py



(α′) figure 1



(β′) figure 2

Μερικές επεξηγήσεις

```python
import numpy as np
import matplotlib.pyplot as plt
```

- Στην αρχή κάνουμε import τις βιβλιοθήκες numpy και matplotlib.
- Χρειαζόμαστε την numpy για να χειριστούμε τα σήματα.
- Περιέχει μερικές βασικές συναρτήσεις (π.χ. η sinc κτλ)
- Η matplotlib χρησιμοποιείται για τις γραφικές παραστάσεις.

Μερικές επεξηγήσεις

```
# Time axis
Npt = 1000                      # number of points in the time axis
T = 1e-9                        # total signal duration [s]
T1 = 0.2e-9                     # pulse "on" duration [s]
t = np.linspace(-T/2, T/2, 1000)    # time axis

# Frequency axis
Npf = 1000                      # number of points in the frequency axis
Fmax = 40e9                     # maximum frequency
f = np.linspace(-Fmax, Fmax, 1000)  # frequency axis
```

- Ορίζουμε ορισμένες παραμέτρους:
  - Npt είναι ο αριθμός των σημείων που έχει ο άξονας του χρόνου.
  - T είναι η διάρκεια του άξονα.
  - T1 είναι η διάρκεια που ο παλμός είναι ίσος με 1.
  - t είναι ο άξονας του χρόνου.
  - Npf είναι ο αριθμός των σημείων στον άξονα των συχνοτήτων.
  - Fmax είναι η μέγιστη συχνότητα που θα έχουμε στον άξονα.
  - f είναι ο άξονας των συχνοτήτων.

Μερικές επεξηγήσεις

```python
# Time signal
x = np.zeros(t.size)
i = 0

for tm in t:
    if np.abs(tm) <= T1/2.0:
        x[i] = 1
    i = i + 1

X = np.sinc( f * T1 )
```

- Υπολογίζουμε το σήμα στο πεδίο του χρόνου:
- φτιάχνουμε ένα πίνακα με όλο μηδενικά, x = np.zeros(t.size).
- t.size είναι το μέγεθος του πίνακα t.
- μηδενίζουμε έναν counter i
- στην ουσία για κάθε τιμή tm στον άξονα αυτόν κοιτάμε να δούμε:
    - αν είναι εντός του -T1/2 και T1/2 τότε θέτουμε ίσο με 1.
    - αν όχι δεν κάνουμε τίποτα και παραμένει ίσο με μηδέν.
- στο πεδίο των συχνοτήτων εφόσον υπάρχει η sinc τα πράγματα είναι πιο εύκολα...

Pythonic κώδικας - plotsincen.py

```
x = np.zeros(t.size)

for i, tm in enumerate(t):
    if np.abs(tm) <= T1/2.0:
        x[i] = 1
```

- Αντί να ορίζουμε έναν counter αφήνουμε την Python να το κάνει για εμάς:
- η enumerate επιστρέφει όλα τα στοιχεία τους πίνακες και τους δείκτες τους

Python enumerate - enum.py

```python
import numpy as np

l = [2,3,4,5]         # this is a Python list
print('Value of l:' , l)

x = np.array(l)        # cast the list to an array
print('Value of x:' , x)

e = enumerate(x)       # enumerate the numpy array x
el = list(e)           # this is now a list
print('Value of el:', el)

for i, v in enumerate(x):
    print('index = ',i,' value = ',v)
```

```
Value of l: [2, 3, 4, 5]
Value of x: [2 3 4 5]
Value of el: [(0, 2), (1, 3), (2, 4), (3, 5)]
index =  0  value =  2
index =  1  value =  3
index =  2  value =  4
index =  3  value =  5
```

Συστήματα

$$x(t) \longrightarrow \boxed{\text{Σύστημα}} \longrightarrow y(t)$$

- Οτιδήποτε επεξεργάζεται ένα σήμα το ονομάζουμε *σύστημα*.
- Στην πιο απλή περίπτωση ένα σύστημα έχει μία είσοδο την $x(t)$ και μία έξοδο την $y(t)$.
- Η πρώτη υπόθεση που θα κάνουμε είναι ότι το σύστημα είναι *γραμμικό*.
  - δηλαδή αν στην είσοδο $x_1(t)$ η έξοδος είναι $y_1(t)$ και στην είσοδο του $x_2(t)$ η έξοδος είναι $y_2(t)$,
  - τότε η έξοδος που αντιστοιχεί στην είσοδο $c_1x_1(t) + c_2x_2(t)$ όπου c_1 και c_2 είναι σταθερές, είναι η $c_1y_1(t) + c_2y_2(t)$.
  - δηλαδή οι συντελεστές c_1 και c_2 μεταφέρονται από την είσοδο στην έξοδο.
  - το $c_1x_1(t) + c_2x_2(t)$ ονομάζεται γραμμικός συνδυασμός των $x_1(t)$ και $x_2(t)$.

Συστήματα

$$x(t) \longrightarrow \boxed{\text{Σύστημα}} \longrightarrow y(t)$$

- Μία άλλη κατηγορία συστήματος είναι τα *χρονικά αναλλοίωτα* συστήματα.
- στην πράξη ένα χρονικά αναλλοίωτο σύστημα είναι ένα σύστημα στο οποίο η απόκριση δεν εξαρτάται από την χρονική στιγμή που «εισάγουμε» το σήμα εισόδου.
- έτσι αν η έξοδος στο $x(t)$ είναι $y(t)$, η έξοδος στο $x(t - t_0)$ είναι $y(t - t_0)$.

Συστήματα στην πράξη

- Στην πράξη όλα τα συστήματα *δεν* είναι γραμμικά και *δεν* είναι χρονικά αναλλοίωτα.
- Για παράδειγμα, ας θεωρήσουμε τον ενισχυτή του στερεοφωνικού σας.
- Αν τερματίσετε την ένταση ακούτε παραμόρφωση. Επομένως η απόκριση στο $cx(t)$ δεν είναι πάντα το $cy(t)$.
- Ο ενισχυτής σας δεν είναι γραμμικός!
- Επίσης ο ενισχυτής σας μπορεί να «ζεσταθεί» και η απόκριση του να μην είναι η ίδια με αυτή όταν τον ανάψατε.
- Επομένως ο ενισχυτής σας δεν είναι χρονικά αναλλοίωτος!

Πως περιγράφονται τα συστήματα;

- Ο πρώτος τρόπος είναι στο πεδίο του χρόνου.
- Ένα γραμμικό, χρονικά αναλλοίωτο σύστημα χαρακτηρίζεται από αυτό που λέμε «κρουστική απόκριση» $h(t)$.
- Η κρουστική απόκριση καθορίζεται από την φύση του συστήματος.
- Αν την ξέρουμε μπορούμε να υπολογίσουμε (επί της αρχής) την έξοδο $y(t)$ σε κάθε είσοδο $x(t)$.

$$y(t) = \int_{-\infty}^{+\infty} h(t-\tau)x(\tau)\mathrm{d}\tau \tag{6}$$

Κρουστική απόκριση

- Ας θεωρήσουμε έναν πολύ στενό παλμό $\delta(t)$ που διαρκεί διάστημα Δt και έχει πλάτος Δt^{-1}.

$$\delta(t) = \begin{cases} \frac{1}{\Delta t} & , 0 \leq t \leq \Delta t \\ 0 & , \text{διαφορετικά} \end{cases} \tag{7}$$

- αν το σκεφτείτε, είναι ένας «κεραυνός»: κτυπάει ακαριαία και με πολύ μεγάλη ένταση (εάν $\Delta t \rightarrow 0$).
- Η έξοδος του συστήματος σε αυτή την είσοδο θα είναι:

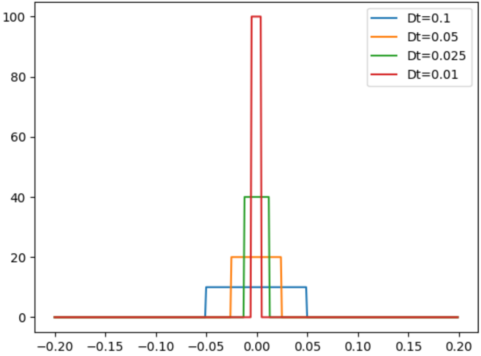
$$y(t) = \int_{-\infty}^{+\infty} h(t-\tau)\delta(\tau)\mathrm{d}\tau = \frac{1}{\Delta t}\int_{0}^{\Delta t} h(t-\tau)\mathrm{d}\tau \cong \frac{1}{\Delta t}\Delta t h(t) = h(t) \tag{8}$$

- Παραπάνω θεωρούμε ότι η $h(t)$ δεν μετβάλλεται σημαντικά όταν το Δt είναι πολύ μικρό.
- Άρα η κρουστική απόκριση είναι η έξοδος του συστήματος όταν βάζουμε έναν πολύ σύντομο παλμό $\delta(t)$ με πολύ μεγάλο πλάτος.

delta.py

```python
import numpy as np
import matplotlib.pyplot as plt

def delta(t, Dt):
    x = np.zeros(t.size)

    for i, tm in enumerate(t):
        if -Dt/2.0 <= tm <= Dt/2.0:
            x[i] = 1.0/Dt

    return x

Dts = np.array([0.1, 0.05, 0.025, 0.01])
t = np.arange(-0.2, 0.2, 0.001)

plt.close('all')
plt.figure(1)

for Dt in Dts:
    x = delta(t,Dt)
    label = 'Dt=' + str(Dt)
    plt.plot(t, x, label = label)

plt.legend()
```

delta.py



Dt=0.1
Dt=0.05
Dt=0.025
Dt=0.01
100
80
60
40
20
0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20

Συστήματα στο πεδίο των συχνοτήτων

- Η περιγραφή στο πεδίο των συχνοτήτων είναι πιο εύκολη.
- Υπολογίζουμε το φάσμα $X(f)$ του $x(t)$.
- Υπολογίζουμε το φάσμα $H(f)$ του $h(t)$.
- Υπολογίζουμε το φάσμα $Y(f) = H(f)X(f)$.
- Υπολογίζουμε το σήμα $y(t)$ από το φάσμα $Y(f)$.

Υπολογίζοντας τα φάσματα χωρίς ολοκληρώματα

- Πολλές φορές δεν μπορούμε να υπολογίσουμε τα φάσματα
- Μπορούμε να δοκιμάσουμε αριθμητική ολοκλήρωση.
- Σε αυτό μας βοηθάει ο γρήγορος μετασχηματισμός Fourier - Fast Fourier Transform - FFT.
- Στην ουσία προσεγγίζουμε το

$$X(f) = \int_{-\infty}^{+\infty} x(t)\mathrm{e}^{-j2\pi f t}\mathrm{d}t \tag{9}$$

- με τον διακριτό μετασχηματισμό Fourier,

$$X_k = \sum_{n=0}^{N-1} x_n \mathrm{e}^{-j\frac{2\pi k n}{N}} \tag{10}$$

Υπολογίζοντας τα φάσματα χωρίς ολοκληρώματα

- Θέτουμε $x_n = x(n\Delta t)$ και θεωρούμε το σήμα $x(t)$ σε διακριτές στιγμές $t_n = n\Delta t$.
- επίσης θεωρούμε ότι θέλουμε να υπολογίσουμε το φάσμα σε συγκεκριμένες συχνότητες $f_k = k\Delta f$.
- Ταιριάζοντας τα δύο εκθετικά $\mathrm{e}^{-j2\pi f_k t_n}$ και $\mathrm{e}^{-j\frac{2\pi kn}{N}}$ βλέπουμε ότι:

$$f_k t_n = k\Delta f \times n\Delta t = \frac{kn}{N} \tag{11}$$

- Επομένως θα έχουμε:

$$\Delta f \Delta t = \frac{1}{N} \tag{12}$$

Υπολογίζοντας τα φάσματα χωρίς ολοκληρώματα

- Επομένως στην περίπτωση όπου θέλουμε να προσεγγίσουμε τον μετασχηματισμό Fourier με τον DFT (και τον FFT) θα πρέπει να θυμόμαστε ότι:
    - Αν Δt είναι η χρονική απόσταση μεταξύ των διαδοχικών χρονικών στιγμών που θεωρούμε στον άξονα του χρόνου
    - και Δf είναι η συχνοτική απόσταση μεταξύ των διαδοχικών συχνοτήτων που θεωρούμε στον άξονα των συχνοτήτων,
    - θα πρέπει $\Delta f \Delta t = \frac{1}{N}$ όπου N το πλήθος των σημείων των πινάκων f και t.
- Μία πιο ενδελεχής μαθηματική ανάλυση δείχνει ότι θα πρέπει να κάνουμε μία μετάθεση των δειγμάτων στο πεδίο των συχνοτήτων για να ταιριάξουν πλήρως οι συχνότητες.
- Ευτυχώς η numpy το έχει ήδη αυτό έτοιμο για εμάς.

Φάσματα με τον FFT

- Επομένως στην περίπτωση όπου θέλουμε να προσεγγίσουμε τον μετασχηματισμό Fourier με τον DFT (και τον FFT) θα πρέπει να θυμόμαστε ότι:
    - Αν Δt είναι η χρονική απόσταση μεταξύ των διαδοχικών χρονικών στιγμών που θεωρούμε στον άξονα του χρόνου
    - και Δf είναι η συχνοτική απόσταση μεταξύ των διαδοχικών συχνοτήτων που θεωρούμε στον άξονα των συχνοτήτων,
    - θα πρέπει $\Delta f \Delta t = \frac{1}{N}$ όπου N το πλήθος των σημείων των πινάκων f και t.
- Μία πιο ενδελεχής μαθηματική ανάλυση δείχνει ότι θα πρέπει να κάνουμε μία μετάθεση των δειγμάτων στο πεδίο των συχνοτήτων για να ταιριάξουν πλήρως οι συχνότητες.
- Ευτυχώς η numpy το έχει ήδη αυτό έτοιμο για εμάς.

fftshift - fft - fftshift

```
import numpy as np
import matplotlib.pyplot as plt

# function to create the square pulse
def square(t , T):

    f = np.logical_and( t <= T / 2.0 , t >= -T
      /2.0 )
    return f.astype('float')

# time axis
T = 4;
Tmax = 30.0
N = 1024
n = np.arange(-N / 2.0, N / 2.0, 1)
Dt = 2 * Tmax / N
t = n * Dt

# frequency axis
Df = 1.0 / ( N * Dt)
f = n * Df

# signal in the time domain
x=square(t,T)

# signal in the frequency domain
X=Dt*np.fft.fftshift(np.fft.fft(np.fft.fftshift(x
       )))

# Analytical expression for the spectrum
X2= T * np.sinc( f * T )

# Plot results
plt.figure(1)
plt.xlabel('t [msec]')
plt.ylabel('x(t)')
plt.plot(t,x)

plt.figure(2)
plt.plot(t,np.real(X2),t,X,'o')
plt.xlim(-3,3)
plt.legend(['analytical','numerical'])
plt.xlabel('f [kHz]')
plt.ylabel('X(f)')
```

Μερικές επεξηγήσεις

```python
# function to create the square pulse
def square(t , T):

    f = np.logical_and( t <= T / 2.0 , t >= -T/2.0 )
    return f.astype('float')
```

- Εδώ φτιάχνουμε έναν τετραγωνικό παλμό με τα περίφημα Python one-liners.
- t είναι ο άξονας του χρόνου, T είναι η διάρκεια του παλμού
- t <= T/2.0 είναι ίσο με True στις θέσεις του t όπου είναι μικρότερες του T/2.0
- t >= -T/2.0 είναι ίσο με True στις θέσεις του t όπου είναι μεγαλύτερες του -T/2.0
- όταν κάνουμε logical and στην ουσία παίρνουμε True όταν -T/2 <= t <= T/2.
- στη συνέχεια μετατρέπουμε τα True/False σε 1/0 με την astype('float')

```
thkam@kirkhome:/mnt/storage/thkam$ python3 -i
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> a = np.array([True, False, True])
>>> print(a)
[ True False  True]
>>> print( a.astype('float') )
[1. 0. 1.]
```

Μερικές επεξηγήσεις

```
T = 4;
Tmax = 30.0
N = 1024
n = np.arange(-N / 2.0, N / 2.0, 1)
Dt = 2 * Tmax / N
t = n * Dt

# frequency axis
Df = 1.0 / ( N * Dt)
f = n * Df
```

- Στην συνέχεια ορίζουμε τον άξονα του χρόνου και τον άξονα των συχνοτήτων.
- ο άξονας του χρόνου αρχίζει από το -Tmax και φτάνει στο +Tmax.
- επομένως $\Delta t = 2T_{\text{max}}/N$ όπου N το πλήθος των σημείων.
- στη συνέχεια υπολογίζουμε το Δf σύμφωνα με την σχέση $\Delta f \Delta t = 1/N$.

Μερικές επεξηγήσεις

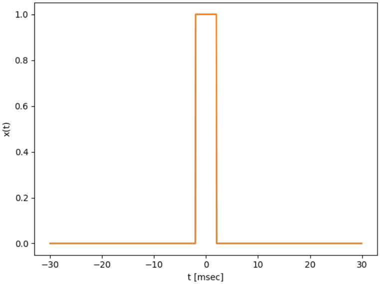
```
# signal in the frequency domain
X=Dt*np.fft.fftshift(np.fft.fft(np.fft.fftshift(x)))

# Analytical expression for the spectrum
X2= T * np.sinc( f * T )
```

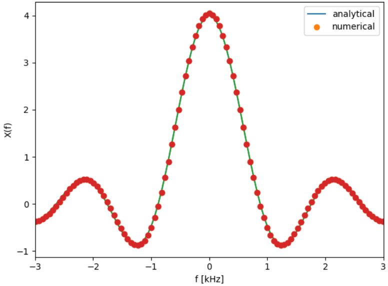
- Εδώ είναι το «ζουμί».
- κάνουμε αυτό που είπαμε με τον fft και τις μεταθέσεις των δειγμάτων ώστε να υπολογιστεί σωστά το φάσμα.
- X = Dt * np.fftshift(np.fft(np.fftshift(x)))
- Πολλαπλασιάζουμε με Dt επειδή στο ολοκλήρωμα του μετασχηματισμού Fourier υπάρχει και εκεί ένα dt.

$$X(f) = \int_{-\infty}^{+\infty} x(t)\mathrm{e}^{-j2\pi f t}\mathrm{d}t \tag{13}$$

Annotations

```
# Plot results
plt.figure(1)
plt.xlabel('t [msec]')
plt.ylabel('x(t)')
plt.plot(t,x)

plt.figure(2)
plt.plot(t,np.real(X2),t,X,'o')
plt.xlim(-3,3)
plt.legend(['analytical','numerical'])
plt.xlabel('f [kHz]')
plt.ylabel('X(f)')
```





Το πρώτο μας Python module

```python
import numpy as np

# function to create the square pulse
def square(t , T):
    f = np.logical_and( t <= T / 2.0 , t >= -T/2.0 )
    return f.astype('float')

# create time axis
def time_axis(Tmin, Tmax, N):
    n = np.arange(-N / 2.0, N / 2.0, 1)
    Dt = (Tmax - Tmin) / N
    return n * Dt

# create frequency axis
def frequency_axis(t):
    N = t.size
    Dt = t[1] - t[0]
    n = np.arange(-N / 2.0, N / 2.0, 1)
    Df = 1.0 / ( N * Dt)
    return n * Df

# Calculate spectrum of x using FFT
def spectrum(t, x):
    Dt = t[1] - t[0]
    return Dt*np.fft.fftshift(np.fft.fft(np.fft.fftshift(x)))

# Inverse fourier transform
def inv_spectrum(f, X):
    Df = f[1] - f[0]
    N = f.size
    return N * Df * np.fft.fftshift(np.fft.ifft(np.fft.fftshift(X)))

# Square filter
```

Διαβάζοντας το module

```
import matplotlib.pyplot as plt
import commlib as cl

# time axis
T = 4;
Tmax = 30.0
N = 1024

t = cl.time_axis( -Tmax, +Tmax, N)
f = cl.frequency_axis(t)
x = cl.square(t, T)
X = cl.spectrum(t, x)

# Analytical expression for the spectrum
X2= T * np.sinc( f * T )

plt.close'('all)

# Plot results
plt.figure(1)
plt.xlabel'(t [msec'])
plt.ylabel'(x(t)')
plt.plot(t,x)
plt.figure(2)
plt.plot(t,np.real(X2),t,X','o)
plt.xlim-(3,3)
plt.legend'(['analytical','numerical])
plt.xlabel'(f [kHz'])
plt.ylabel'(X(f)')
```

Ένα παράδειγμα συστήματος

- Ας δείξουμε τι συμβαίνει όταν ένα σήμα περνάει από ένα σύστημα.
- Θα θεωρήσουμε έναν τετραγωνικό παλμό στην είσοδο ενός συστήματος.
- Το σύστημα κόβει απότομα συχνότητες οι οποίες είναι μεγαλύτερες από μία τιμή $F_{\max}$.
- δηλαδή:

$$H(f) = \begin{cases} 1 & , |f| \leq F_{\max} \\ 0 & , \text{διαφορετικά} \end{cases} \tag{14}$$

- Όταν για παράδειγμα ένα σήμα περνάει από ένα καλώδιο, «κόβονται» οι υψηλές συχνότητες.
- Με διαφορετικό τρόπο βέβαια - όχι τόσο απότομα.

Ένα παράδειγμα συστήματος

- Αλλάζουμε λίγο την commlib προσθέτοντας:
  - μία πολύ απλή συνάρτηση για το τετραγωνικό μας φίλτρο (square_filter).
  - μία συνάρτηση για τον αντίστροφο μετασχηματισμό Fourier (inv_spectrum).
  - μία συνάρτηση που υλοποιεί την δράση του συστήματος πάνω σε ένα σήμα (system_action).

```python
# Inverse fourier transform
def inv_spectrum(f, X):
    Df = f[1] - f[0]
    N = f.size
    return N * Df * np.fft.fftshift(np.fft.ifft(np.fft.fftshift(X)))

# Square filter
def square_filter(f , Fmax):
    return square(f, Fmax)

# System action
def system_action(t, x, Hcallable):
    f = frequency_axis(t)
    Hf = Hcallable(f)
    X = spectrum(t, x)
    Y = X * Hf
    return inv_spectrum(f, Y)
```

Ένα παράδειγμα συστήματος

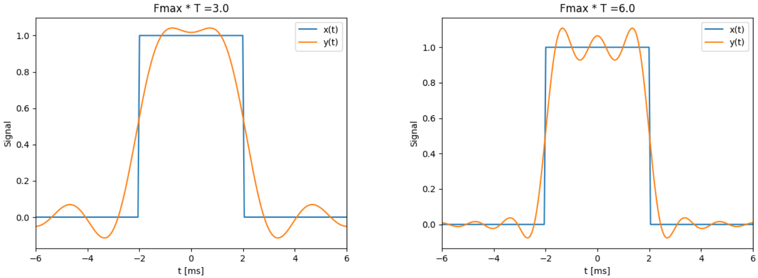
- Έτσι γράφουμε κώδικα που μπορούμε να επαναχρησιμοποιήσουμε.

```python
import numpy as np
import matplotlib.pyplot as plt
import commlib as cl

T = 4;
Tmax = 30.0
N = 1024
Fmax = 20.0/T

t = cl.time_axis(-Tmax, +Tmax, N)          # time axis
x = cl.square(t,T)                         # input signal
Hc = lambda f : cl.square_filter(f, Fmax) # filter callable

y = cl.system_action(t, x, Hc)

plt.close('all')
plt.figure(1)
plt.title('Fmax * T =' + str(Fmax*T))
plt.xlabel('t [ms]')
plt.ylabel('Signal')
plt.plot(t, np.real(x), label='x(t)')
plt.plot(t, np.real(y), label='y(t)')
plt.legend()
plt.xlim(-6,6)
```

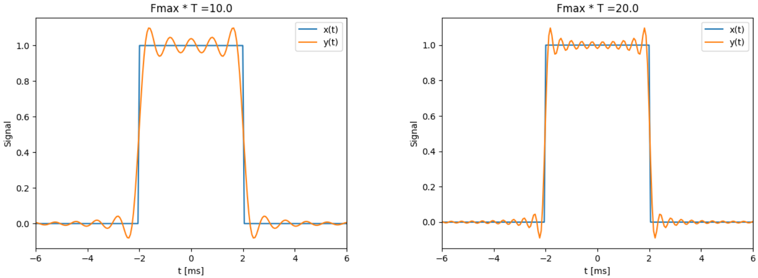
Python callables

- Προσέξτε ότι περνάμε ως όρισμα μία συνάρτηση στην system_action.
- μάλιστα αυτή η συνάρτηση ορίζεται μέσω ενός lambda.
- με το lambda ορίζουμε μία μικρή συνάρτηση που μπορεί να χρησιμοποιεί και παραμέτρους που έχουν οριστεί παραπάνω.
- Hc = lambda f : cl.square_filter(f, Fmax) σημαίνει φτιάξε μία συνάρτηση με μία είσοδο την f η οποία να υπολογίζεται από την cl.square_filter(f, Fmax)
- Με τον τρόπο αυτό περνάμε την κατάλληλη συνάρτηση μεταφοράς του φίλτρου στην system_action.

Αποτελέσματα



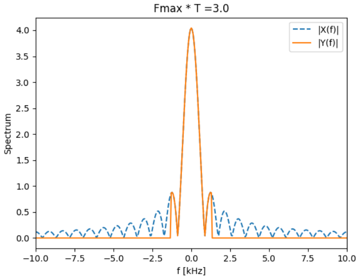
Αποτελέσματα



Ερμηνεία με τα φάσματα

```
import numpy as np
import matplotlib.pyplot as plt
import commlib as cl

T = 4;
Tmax = 30.0
N = 1024
Fmax = 20.0/T

t = cl.time_axis(-Tmax, +Tmax, N)          # time axis
x = cl.square(t,T)                         # input signal
Hc = lambda f : cl.square_filter(f, Fmax) # filter callable

y = cl.system_action(t, x, Hc)

f = cl.frequency_axis(t)
X = cl.spectrum(t,x)
Y = cl.spectrum(t,y)

plt.close('all')
plt.figure(1)
plt.title('Fmax * T =' + str(Fmax*T))
plt.xlabel('f [kHz]')
plt.ylabel('Spectrum')
plt.plot(t, np.abs(X), '--', label='|X(f)|')
plt.plot(t, np.abs(Y), label='|Y(f)|')
plt.xlim([-10, 10])
plt.legend()
```

Ερμηνεία με τα φάσματα



Ερμηνεία με τα φάσματα