

Τηλεπικοινωνιακά Συστήματα

Διάλεξη 2η

Θωμάς Καμαλάκης

Χαροκόπειο Πανεπιστήμιο Αθηνών

Οκτώβριος 2020

Περιεχόμενα

- 1 Σήματα
- 2 Ολίγον Python
- 3 Συστήματα
- 4 Fast Fourier Transform
- 5 Python modules
- 6 Σήματα και Συστήματα
- 7 Ισχύς σημάτων

Η έννοια του σήματος

- Στα συστήματα επικοινωνιών μεταδίδουμε αναλογικά και ψηφιακά *σήματα*.
- Ένα 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.

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft} dt \quad (1)$$

$$x(t) = \int_{-\infty}^{+\infty} X(f)e^{j2\pi ft} df \quad (2)$$

- 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.

$$\begin{aligned} X(f) &= \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft} dt = \int_{-T_1/2}^{+T_1/2} e^{-j2\pi ft} dt = \\ &= \frac{e^{j\pi f T_1} - e^{-j\pi f T_1}}{2\pi f} = \frac{\sin(\pi f T_1)}{j\pi f} = T_1 \frac{\sin(\pi f T_1)}{\pi f T_1} = T_1 \text{sinc}(f T_1) \quad (3) \end{aligned}$$

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.

$$\begin{aligned} X(f) &= \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft} dt = \int_{-T_1/2}^{+T_1/2} e^{-j2\pi ft} dt = \\ &= \frac{e^{j\pi f T_1} - e^{-j\pi f T_1}}{2\pi f} = \frac{\sin(\pi f T_1)}{j\pi f} = T_1 \frac{\sin(\pi f T_1)}{\pi f T_1} = T_1 \text{sinc}(f T_1) \end{aligned} \quad (4)$$

όπου

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

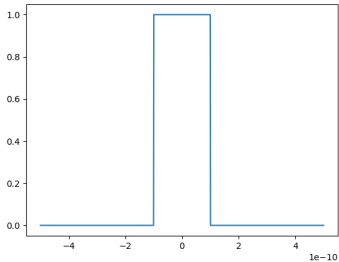
Ένα πρώτο παράδειγμα Python - plotsinc.py

```

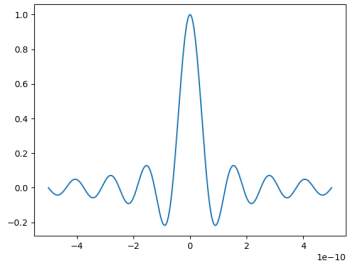
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Time axis
5 Npt = 1000                                # number of points in the time axis
6 T = 1e-9                                  # total signal duration [s]
7 T1 = 0.2e-9                               # pulse "on" duration [s]
8 t = np.linspace(-T/2, T/2, 1000)         # time axis
9
10 # Frequency axis
11 Npf = 1000                                # number of points in the frequency axis
12 Fmax = 40e9                               # maximum frequency
13 f = np.linspace(-Fmax, Fmax, 1000)       # frequency axis
14
15 # Time signal
16 x = np.zeros(t.size)
17 i = 0
18
19 for tm in t:
20     if np.abs(tm) <= T1/2.0:
21         x[i] = 1
22         i = i + 1
23
24 X = np.sinc( f * T1 )
25
26 plt.close('all')
27 plt.figure(1)
28 plt.plot(t, x)
29 plt.savefig
30 plt.figure(2)
31 plt.plot(t, X)

```

Ένα πρώτο παράδειγμα Python - plotsinc.py



(α') figure 1



(β') figure 2

Μερικές επεξηγήσεις

```
1 import numpy as np
2 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

```
1 # Time axis
2 Npt = 1000                                # number of points in the time axis
3 T = 1e-9                                   # total signal duration [s]
4 T1 = 0.2e-9                               # pulse "on" duration [s]
5 t = np.linspace(-T/2, T/2, 1000)          # time axis
6
7 # Frequency axis
8 Npf = 1000                                # number of points in the frequency axis
9 Fmax = 40e9                               # maximum frequency
10 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

```
1 # Time signal
2 x = np.zeros(t.size)
3 i = 0
4
5 for tm in t:
6     if np.abs(tm) <= T1/2.0:
7         x[i] = 1
8         i = i + 1
9
10 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

```
1 x = np.zeros(t.size)
2
3 for i, tm in enumerate(t):
4     if np.abs(tm) <= T1/2.0:
5         x[i] = 1
```

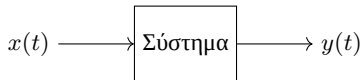
- Αντί να ορίζουμε έναν counter αφήνουμε την Python να το κάνει για εμάς:
- η enumerate επιστρέφει όλα τα στοιχεία τους πίνακες και τους δείκτες τους

Python enumerate - enum.py

```
1 import numpy as np
2
3 l = [2,3,4,5]          # this is a Python list
4 print('Value of l:' , l)
5
6 x = np.array(l)        # cast the list to an array
7 print('Value of x:' , x)
8
9 e = enumerate(x)       # enumerate the numpy array x
10 el = list(e)           # this is now a list
11 print('Value of el:', el)
12
13 for i, v in enumerate(x):
14     print('index = ', i, ' value = ', v)
```

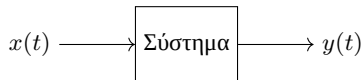
```
Value of l: [2, 3, 4, 5]
Value of x: [2 3 4 5]
Value of el: [(0, 2), (1, 3), (2, 4), (3, 5)]
index = 0  value = 2
index = 1  value = 3
index = 2  value = 4
index = 3  value = 5
```

Συστήματα



- 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



- 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 $x(t)$.

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

Κρουστική απόκριση

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

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

- 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:

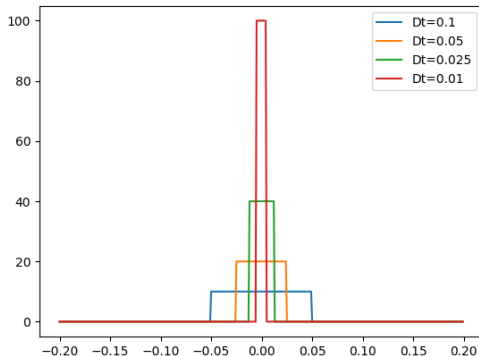
$$y(t) = \int_{-\infty}^{+\infty} h(t - \tau) \delta(\tau) d\tau = \frac{1}{\Delta t} \int_0^{\Delta t} h(t - \tau) d\tau \cong \frac{1}{\Delta t} \Delta t h(t) = h(t) \quad (8)$$

- Παραπάνω θεωρούμε ότι η $h(t)$ δεν μεταβάλλεται 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

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def delta(t, Dt):
5     x = np.zeros(t.size)
6
7     for i, tm in enumerate(t):
8         if -Dt/2.0 <= tm <= Dt/2.0:
9             x[i] = 1.0/Dt
10
11     return x
12
13 Dts = np.array([0.1, 0.05, 0.025, 0.01])
14 t = np.arange(-0.2, 0.2, 0.001)
15
16 plt.close('all')
17 plt.figure(1)
18
19 for Dt in Dts:
20     x = delta(t,Dt)
21     label = 'Dt=' + str(Dt)
22     plt.plot(t, x, label = label)
23
24 plt.legend()
```

delta.py



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

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft} dt \quad (9)$$

- με τον διακριτό μετασχηματισμό Fourier,

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi kn}{N}} \quad (10)$$

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 $e^{-j2\pi f_k t_n}$ και $e^{-j\frac{2\pi kn}{N}}$ βλέπουμε ότι:

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

- Επομένως θα έχουμε:

$$\Delta f \Delta t = \frac{1}{N} \quad (12)$$

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 συχνотική 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 ταιριάζουν 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 συχνотική 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 ταιριάζουν πλήρως οι συχνότητες.
- Ευτυχώς η numpy το έχει ήδη αυτό έτοιμο για εμάς.

fftshift - fft - fftshift

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # function to create the square pulse
5 def square(t , T):
6
7     f = np.logical_and( t <= T / 2.0 , t >= -T
8         /2.0 )
9     return f.astype('float')
10
11 # time axis
12 T = 4;
13 Tmax = 30.0
14 N = 1024
15 n = np.arange(-N / 2.0, N / 2.0, 1)
16 Dt = 2 * Tmax / N
17 t = n * Dt
18
19 # frequency axis
20 Df = 1.0 / ( N * Dt)
21 f = n * Df
22
23 # signal in the time domain
24 x=square(t,T)
25
26 # signal in the frequency domain
27 X=Dt*np.fft.fftshift(np.fft.fft(np.fft.fftshift(x
28     )))
29
30 # Analytical expression for the spectrum
31 X2= T * np.sinc( f * T )
32
33 # Plot results
34 plt.figure(1)
35 plt.xlabel('t [msec]')
36 plt.ylabel('x(t)')
37 plt.plot(t,x)
38
39 plt.figure(2)
40 plt.plot(t,np.real(X2),t,X,'o')
41 plt.xlim(-3,3)
42 plt.legend(['analytical','numerical'])
43 plt.xlabel('f [kHz]')
44 plt.ylabel('X(f)')

```

Μερικές επεξηγήσεις

```

1 # function to create the square pulse
2 def square(t , T):
3
4     f = np.logical_and( t <= T / 2.0 , t >= -T/2.0 )
5     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
- $t \leq T/2.0$ είναι ίσο με True στις θέσεις του t όπου είναι μικρότερες του $T/2.0$
- $t \geq -T/2.0$ είναι ίσο με True στις θέσεις του t όπου είναι μεγαλύτερες του $-T/2.0$
- όταν κάνουμε logical and στην ουσία παίρνουμε True όταν $-T/2 \leq t \leq T/2$.
- 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

```
1 T = 4;
2 Tmax = 30.0
3 N = 1024
4 n = np.arange(-N / 2.0, N / 2.0, 1)
5 Dt = 2 * Tmax / N
6 t = n * Dt
7
8 # frequency axis
9 Df = 1.0 / ( N * Dt)
10 f = n * Df
```

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

Μερικές επεξηγήσεις

```
1 # signal in the frequency domain
2 X=Dt*np.fft.fftshift(np.fft.fft(np.fft.fftshift(x)))
3
4 # Analytical expression for the spectrum
5 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.
- $X = Dt * \text{np.fftshift}(\text{np.fft}(\text{np.fftshift}(x)))$
- Πολλαπλασιάζουμε με Dt επειδή στο ολοκλήρωμα του μετασχηματισμού Fourier υπάρχει και εκεί ένα dt.

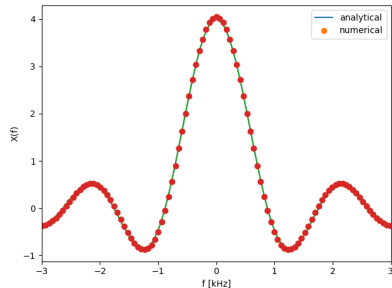
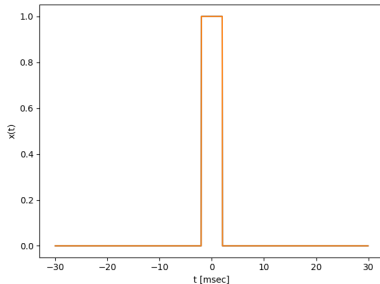
$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft} dt \quad (13)$$

Annotations

```

1 # Plot results
2 plt.figure(1)
3 plt.xlabel('t [msec]')
4 plt.ylabel('x(t)')
5 plt.plot(t,x)
6
7 plt.figure(2)
8 plt.plot(t,np.real(X2),t,X,'o')
9 plt.xlim(-3,3)
10 plt.legend(['analytical','numerical'])
11 plt.xlabel('f [kHz]')
12 plt.ylabel('X(f)')

```



Το πρώτο μας Python module

```

1 import numpy as np
2
3 # function to create the square pulse
4 def square(t, T):
5     f = np.logical_and( t <= T / 2.0 , t >= -T/2.0 )
6     return f.astype('float')
7
8 # create time axis
9 def time_axis(Tmin, Tmax, N):
10     n = np.arange(-N / 2.0, N / 2.0, 1)
11     Dt = (Tmax - Tmin) / N
12     return n * Dt
13
14 # create frequency axis
15 def frequency_axis(t):
16     N = t.size
17     Dt = t[1] - t[0]
18     n = np.arange(-N / 2.0, N / 2.0, 1)
19     Df = 1.0 / ( N * Dt)
20     return n * Df
21
22 # Calculate spectrum of x using FFT
23 def spectrum(t, x):
24     Dt = t[1] - t[0]
25     return Dt*np.fft.fftshift(np.fft.fft(np.fft.fftshift(x)))
26
27 # Inverse fourier transform
28 def inv_spectrum(f, X):
29     Df = f[1] - f[0]
30     N = f.size
31     return N * Df * np.fft.fftshift(np.fft.ifft(np.fft.fftshift(X)))
32
33 # Square filter

```



Διαβάζοντας το module

```
1 import matplotlib.pyplot as plt
2 import commlib as cl
3
4 # time axis
5 T = 4;
6 Tmax = 30.0
7 N = 1024
8
9 t = cl.time_axis( -Tmax, +Tmax, N)
10 f = cl.frequency_axis(t)
11 x = cl.square(t, T)
12 X = cl.spectrum(t, x)
13
14 # Analytical expression for the spectrum
15 X2= T * np.sinc( f * T )
16
17 plt.close('all')
18
19 # Plot results
20 plt.figure(1)
21 plt.xlabel('t [msec]')
22 plt.ylabel('x(t)')
23 plt.plot(t,x)
24 plt.figure(2)
25 plt.plot(t,np.real(X2),t,X', 'o')
26 plt.xlim(-3,3)
27 plt.legend(['analytical','numerical'])
28 plt.xlabel('f [kHz]')
29 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:

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

- 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`).

```
1 # Inverse fourier transform
2 def inv_spectrum(f, X):
3     Df = f[1] - f[0]
4     N = f.size
5     return N * Df * np.fft.fftshift(np.fft.ifft(np.fft.fftshift(X)))
6
7 # Square filter
8 def square_filter(f, Fmax):
9     return square(f, Fmax)
10
11 # System action
12 def system_action(t, x, Hcallable):
13     f = frequency_axis(t)
14     Hf = Hcallable(f)
15     X = spectrum(t, x)
16     Y = X * Hf
17     return inv_spectrum(f, Y)
```

Ένα παράδειγμα συστήματος

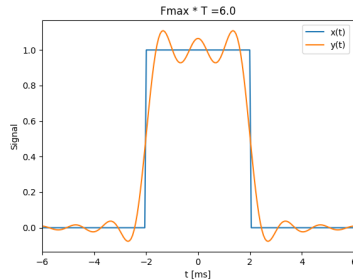
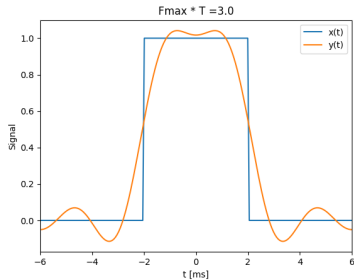
- Έτσι γράφουμε κώδικα που μπορούμε να επαναχρησιμοποιήσουμε.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import commlib as cl
4
5 T = 4;
6 Tmax = 30.0
7 N = 1024
8 Fmax = 20.0/T
9
10 t = cl.time_axis(-Tmax, +Tmax, N)      # time axis
11 x = cl.square(t,T)                    # input signal
12 Hc = lambda f : cl.square_filter(f, Fmax) # filter callable
13
14 y = cl.system_action(t, x, Hc)
15
16 plt.close('all')
17 plt.figure(1)
18 plt.title('Fmax * T =' + str(Fmax*T))
19 plt.xlabel('t [ms]')
20 plt.ylabel('Signal')
21 plt.plot(t, np.real(x), label='x(t)')
22 plt.plot(t, np.real(y), label='y(t)')
23 plt.legend()
24 plt.xlim(-6,6)
```

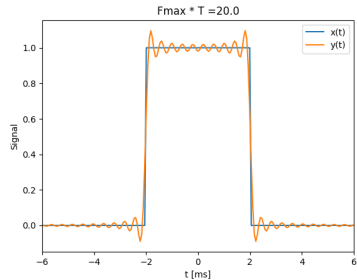
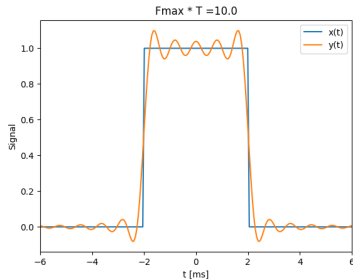
Python callables

- 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



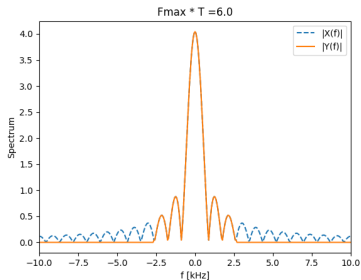
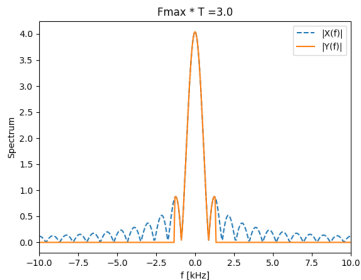
Αποτελέσματα



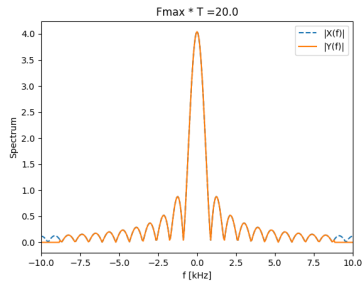
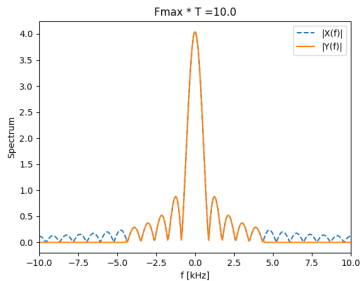
Ερμηνεία με τα φάσματα

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import commlib as cl
4
5 T = 4;
6 Tmax = 30.0
7 N = 1024
8 Fmax = 20.0/T
9
10 t = cl.time_axis(-Tmax, +Tmax, N)      # time axis
11 x = cl.square(t,T)                    # input signal
12 Hc = lambda f : cl.square_filter(f, Fmax) # filter callable
13
14 y = cl.system_action(t, x, Hc)
15
16 f = cl.frequency_axis(t)
17 X = cl.spectrum(t,x)
18 Y = cl.spectrum(t,y)
19
20 plt.close('all')
21 plt.figure(1)
22 plt.title('Fmax * T = ' + str(Fmax*T))
23 plt.xlabel('f [kHz]')
24 plt.ylabel('Spectrum')
25 plt.plot(t, np.abs(X), '—', label='|X(f)|')
26 plt.plot(t, np.abs(Y), label='|Y(f)|')
27 plt.xlim([-10, 10])
28 plt.legend()
```

Ερμηνεία με τα φάσματα



Ερμηνεία με τα φάσματα



Τι είναι η ισχύς και η ενέργεια;

- Στις επικοινωνίες μας ενδιαφέρει πόση ισχύ εκπέμπουμε
 - για λόγους οικονομικούς.
 - για λόγους ασφάλειας.
- η *ενέργεια* E ενός σήματος $x(t)$ ορίζεται ως:

$$E = \int_{-\infty}^{+\infty} |x(t)|^2 dt \quad (15)$$

- η *ενέργεια* $E(t_a, t_b)$ ενός σήματος $x(t)$ στο χρονικό διάστημα $[t_a, t_b]$ ορίζεται ως:

$$E(t_a, t_b) = \int_{t_a}^{t_b} |x(t)|^2 dt \quad (16)$$

- η *στιγμιαία ισχύς* $p(t)$ του σήματος ορίζεται ως:

$$p(t) = |x(t)|^2 \quad (17)$$

- η *μέση ισχύς* P του σήματος ορίζεται ως:

$$P = \lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_{-\frac{\tau}{2}}^{+\frac{\tau}{2}} |x(t)|^2 dt \quad (18)$$

Ένα παράδειγμα

$$x(t) = A \cos(2\pi f_0 t) \quad (19)$$

Ας υπολογίσουμε την ενέργεια στο διάστημα $[0, T]$ όπου $T = \frac{1}{f_0}$ είναι η περίοδος του σήματος.

$$\begin{aligned} E(0, T) &= \int_0^T |x(t)|^2 dt = A^2 \int_0^T \cos^2(2\pi f_0 t) dt = \\ &= \frac{A^2}{2} \int_0^T (1 + \cos(4\pi f_0 t)) dt = \frac{A^2 T}{2} \quad (20) \end{aligned}$$