lab01

March 5, 2017

1 No1.

1.1

, c , .

1.2

MATLAB Simulink 3, . 150–170 (.). python (Jupyter Notebook). LaTeX.

1.3

python 3. , : - scipy.signal - - mathplotlib - - numpy - Jupyter Notebook.

1.3.1 1

python , :

$$s_1(t) = \cos(2\pi f_0 t + \phi) \tag{1}$$

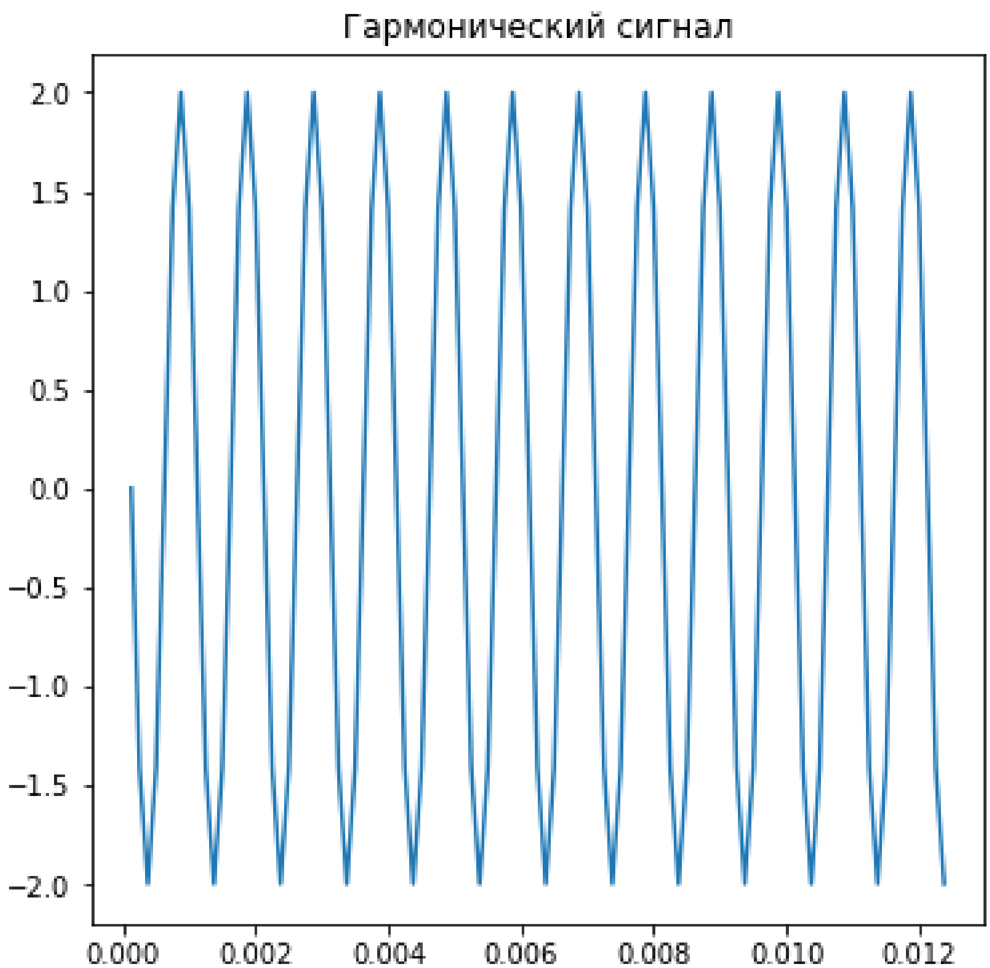
MATLAB:

```
In [82]: % autocall 1

         def elem_multiple(a, b):
                 if a.shape != b.shape:
                         raise IOError("Shapes of matrices are not the same")
                         return
                 else:
                         res = np.zeros(a.shape)
                         for i in range(a.shape[0]):
                                 for j in range(a.shape[1]):
                                         res[(i,j)] = a[(i,j)] * b[(i,j)]
                         # print(res.shape)
                         return res
```

```
Automatic calling is: Smart
```

matplotlib $s_1(t)$

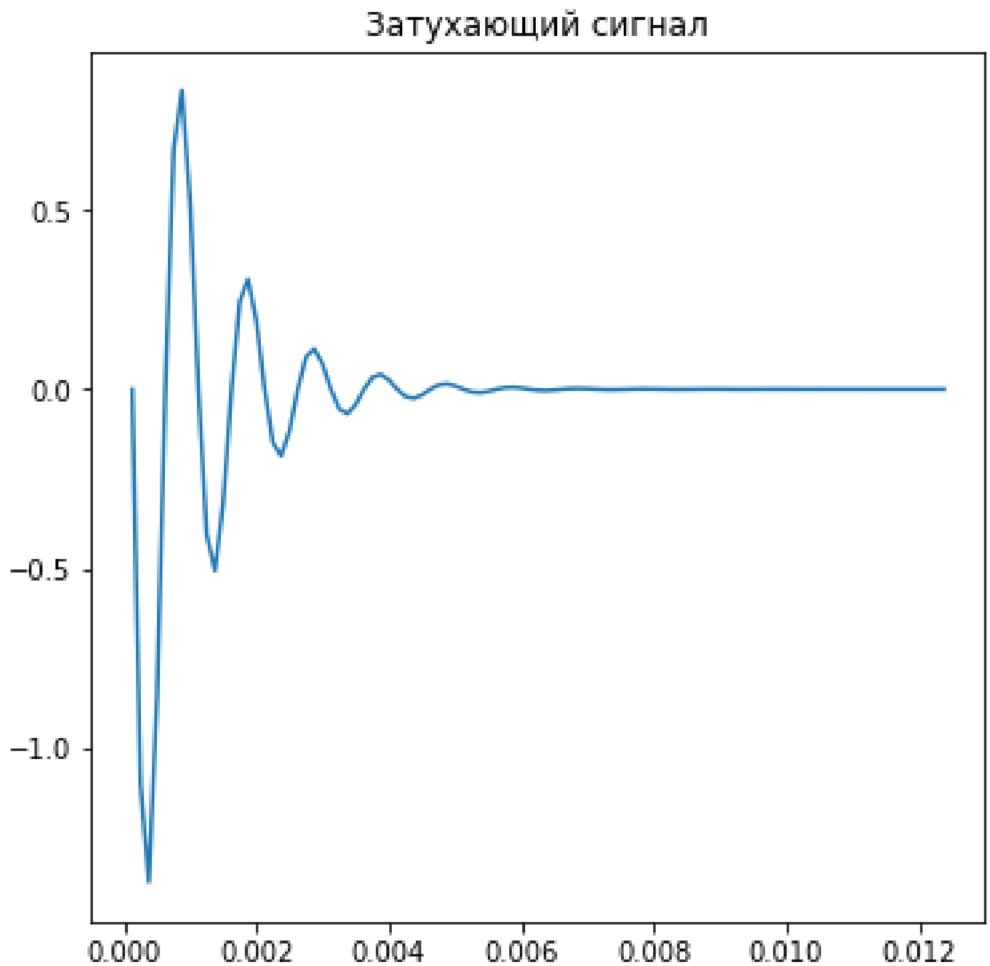
In [83]:
```
import numpy as np
import math
import matplotlib.pyplot as plt

%matplotlib inline

Fs = 8000 # discrete frequency
t = np.matrix(np.arange(0.0, 1.0, (1 / Fs))).transpose()
A = 2 # amplitude
f0 = 1000
phi =  np.pi /4
s1 = A * np.cos(2 * np.pi *  f0 * t + phi) # harmonic signal
fig = plt.figure(figsize=(6,6))
plt.title(' ')
plt.plot(t[1:100],s1[1:100])
fig.savefig('picturesNote/001harmonic.png', dpi=200)
plt.show()
```



. , :

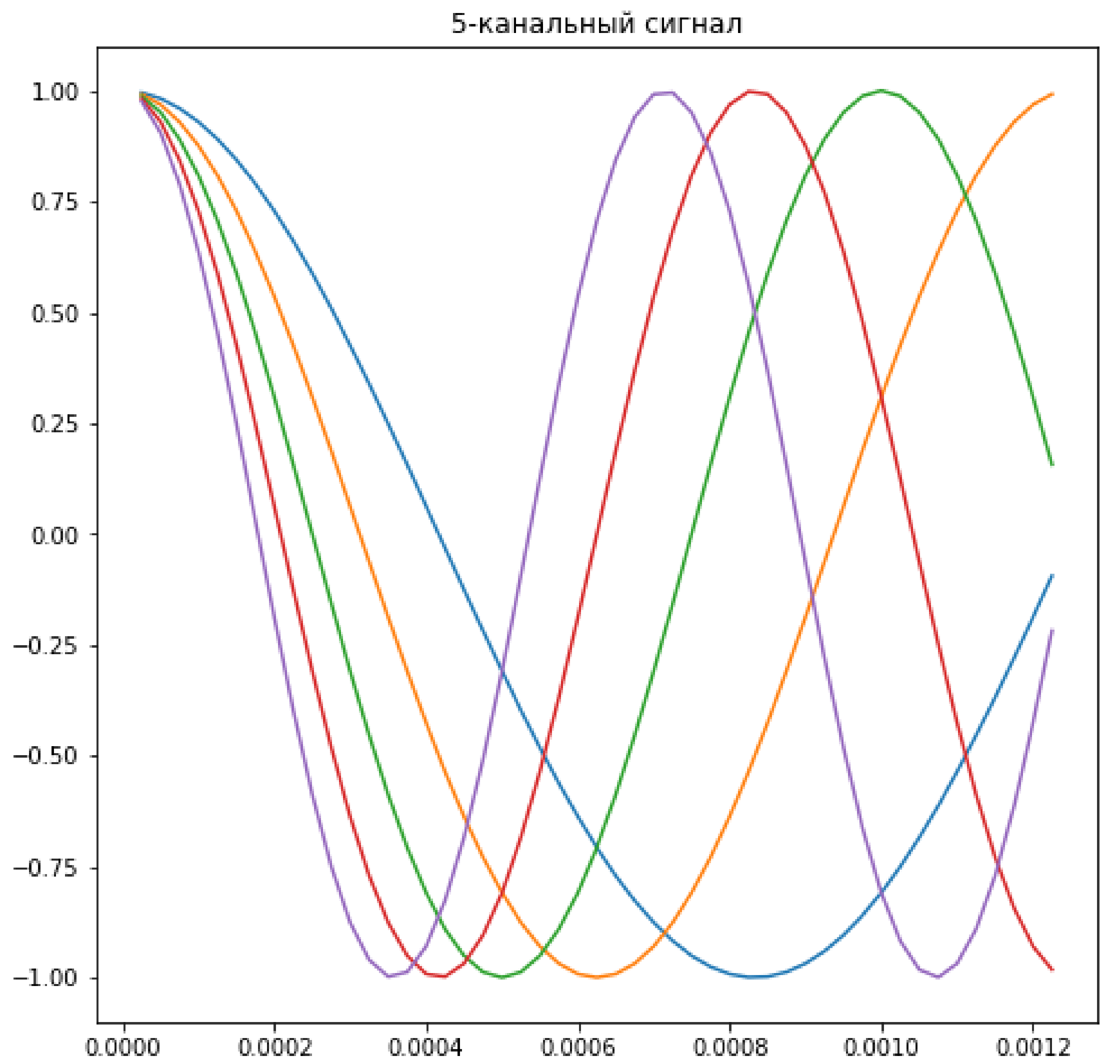
$$s_2(t) = \cos(2\pi f_0 t + \phi) e^{-\alpha t} \tag{2}$$

In [84]:
```
import numpy as np
import matplotlib.pyplot as plt


Fs = 8000 # discrete frequency
t = np.matrix(np.arange(0.0, 1.0, (1 / Fs))).transpose()
f0 = 1000

phi =  np.pi /4
alpha = 1000

s2 = elem_multiple(s1, np.exp(-alpha * t))

fig = plt.figure(figsize=(6,6))
plt.title(' ')
plt.plot(t[1:100],s2[1:100])
fig.savefig('picturesNote/002harmonic.png', dpi=100)
plt.show()
```



( ), 5. : $s_3(t) = \cos(2\pi f_0 t),\ f_0 = (600, 800, 1000, 1200, 1400)$

In [85]:
```
import numpy as np
import matplotlib.pyplot as plt


Fs = 40000 # discrete frequency
t = np.matrix(np.arange(0.0, 1.0, (1 / Fs))).transpose()
f = np.matrix([600, 800, 1000, 1200, 1400])

s3 = np.matrix(np.cos(2*np.pi * t * f)) # 5 channels signal

fig = plt.figure(figsize=(8, 8))
plt.title('5- ')
plt.plot(t[1:50], s3[1:50])
fig.savefig('picturesNote/003_5chnls.png', dpi=200)
plt.show()
```



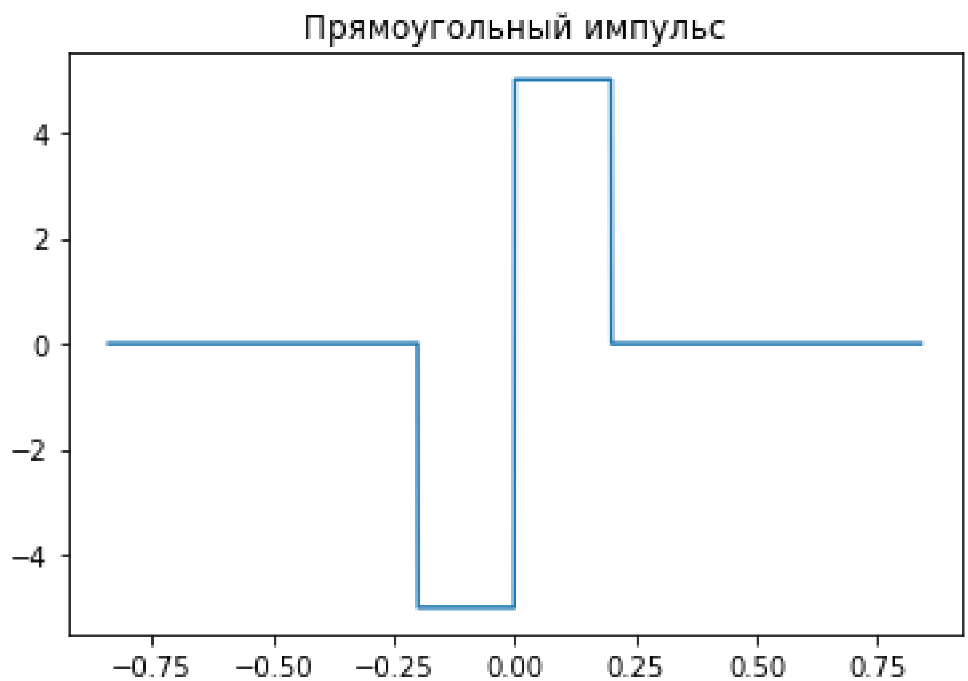
- .

1.3.2 2

python scipy.signal MATLAB , .

```
In [86]: import numpy as np
         import matplotlib.pyplot as plt

         # rectangular impulse
         def rect_impls_1(t, width):
                 sig = np.zeros(len(t))
                 for i in sig:
                         if -width/2 <= i < width / 2:
                                 i = 1
```

```
        for i in range(len(sig)):
                if -width/2 <= t[i] < width / 2:
                        sig[i] = 1
        return sig

Fs = 1000
t = np.matrix(np.arange(-0.84, 0.84, (1 / Fs))).transpose()
width = 0.2
A = 5
s = -A *rect_impls_1(t + width / 2, width) + A * rect_impls_1(t - width / 2 ,width)
plt.title(' ')
plt.plot(t,s)
plt.gcf().savefig('picturesNote/004rectImp.png', dpi=100)
plt.show()
```



.

, scipy.signal. :

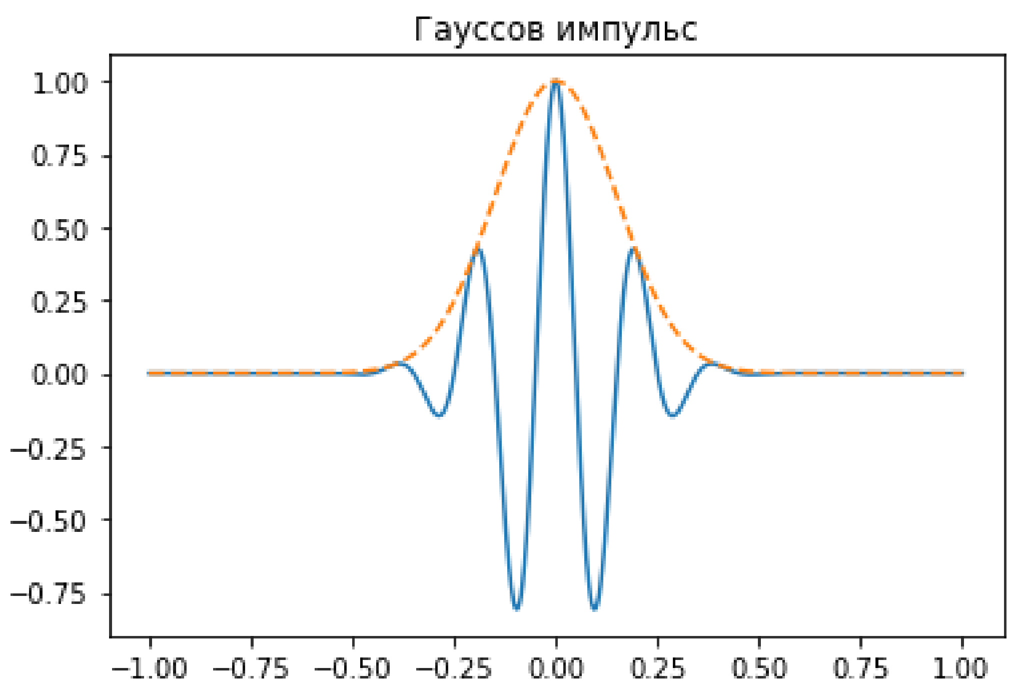
$$y(t) = \cos(2\pi f_c t)e^{-\alpha t^2} \tag{3}$$

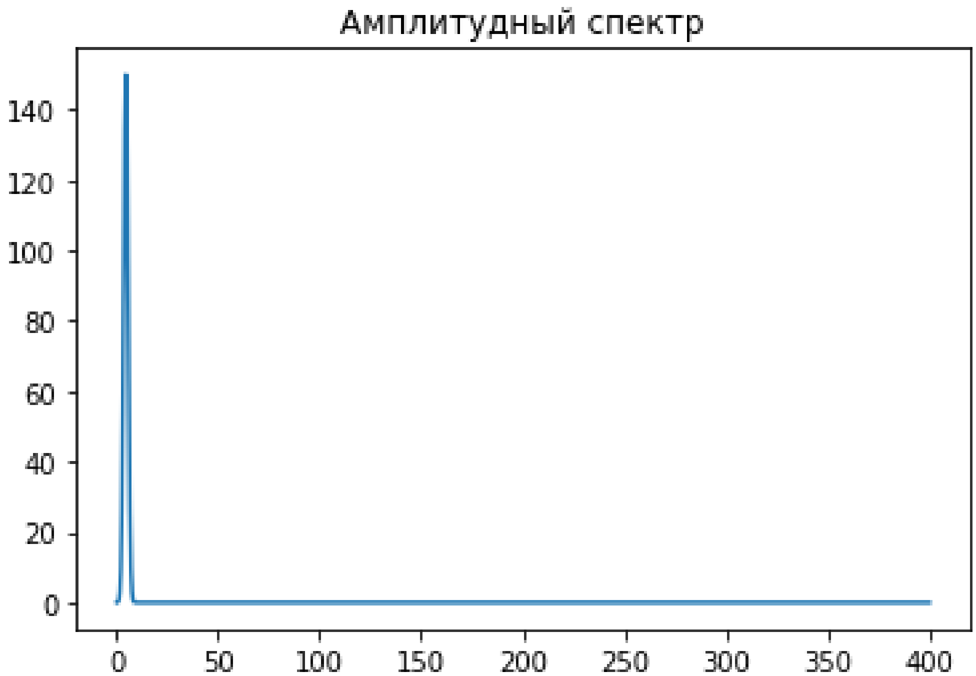
, .

```
In [87]: import numpy as np
         import matplotlib.pyplot as plt
         from scipy import signal
```

```
from scipy.fftpack import fft, fftfreq

Fs = 800
t = np.linspace(-1, 1, 2 * Fs)
i, e = signal.gausspulse(t, fc=5, retenv=True)
fig = plt.figure()
plt.plot(t, i, t, e, '--')
plt.title(' ')
fig.savefig('picturesNote/005_gaus.png', dpi=70)
plt.show()
N = len(t)
Fs = 800
T = 1.0 / Fs
x = t
y = i
yf = fft(y)
xf = fftfreq(N, 1.0 / Fs)
fig = plt.figure()
plt.title(' ')
plt.plot(xf[1:800], np.abs(yf[1:800]))
fig.savefig('picturesNote/006_spctrgaus.png', dpi=70)
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/scipy/signal/waveforms.py:236: FutureWarning: elementwise
  if t == 'cutoff':  # compute cut_off point
```

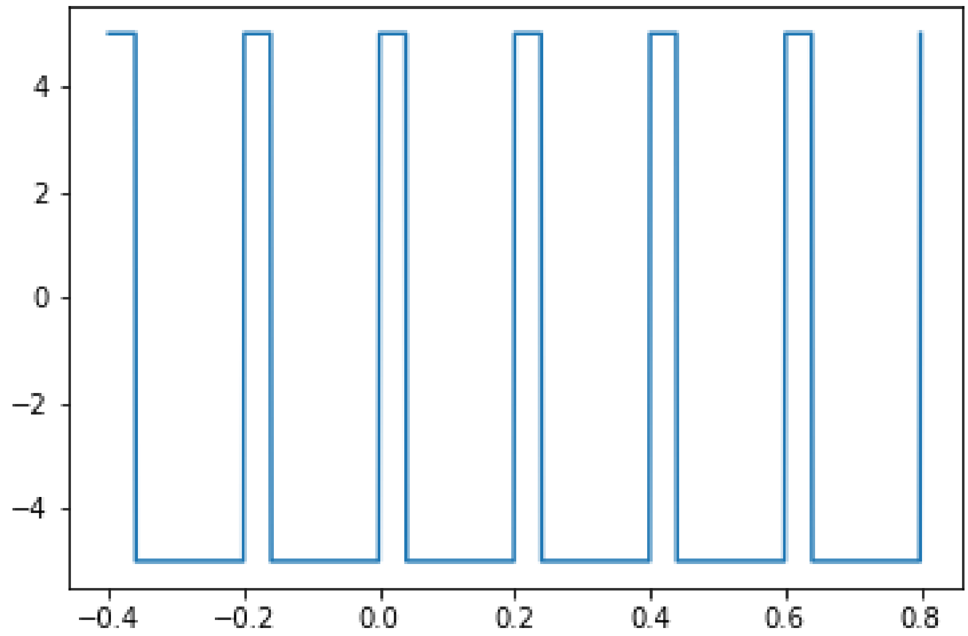




1.3.3 3

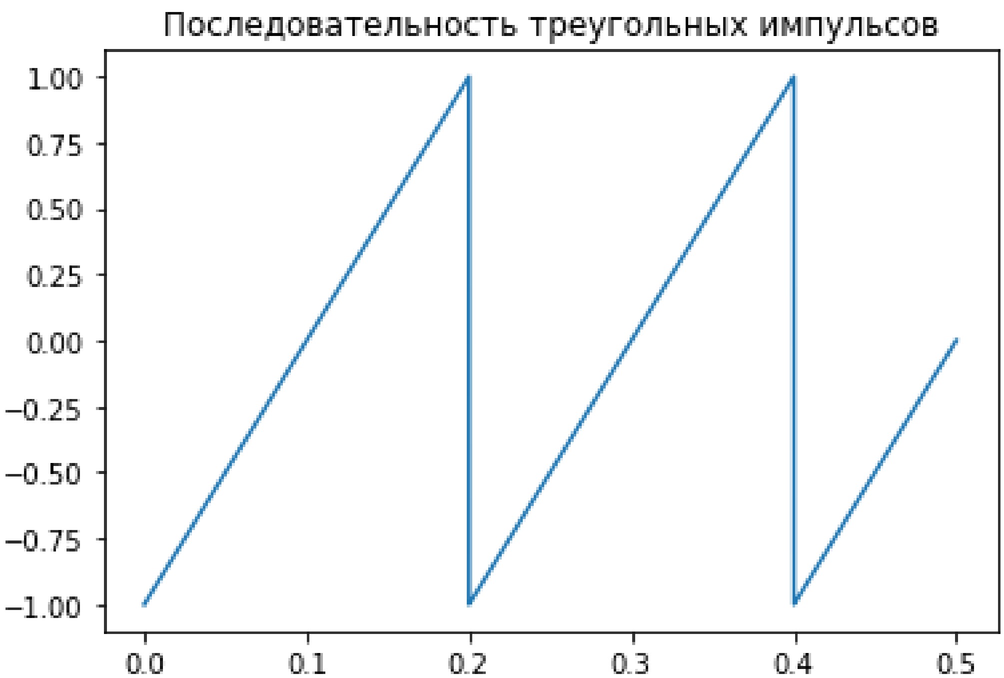
 - , . .

```
In [88]: import numpy as np
         import math
         import matplotlib.pyplot as plt
         from scipy import signal

         Fs = 1000 # discrete frequency
         t = np.matrix(np.arange(-0.4, 0.8, (1 / Fs))).transpose()
         A = 5
         fig = plt.figure()
         plt.plot(t, A *signal.square( 2 * np.pi * 5 * t, 0.2))
         fig.savefig('picturesNote/007rectImplses.png', dpi=100)
         plt.show()
```



C sawtooth scipy.signal .

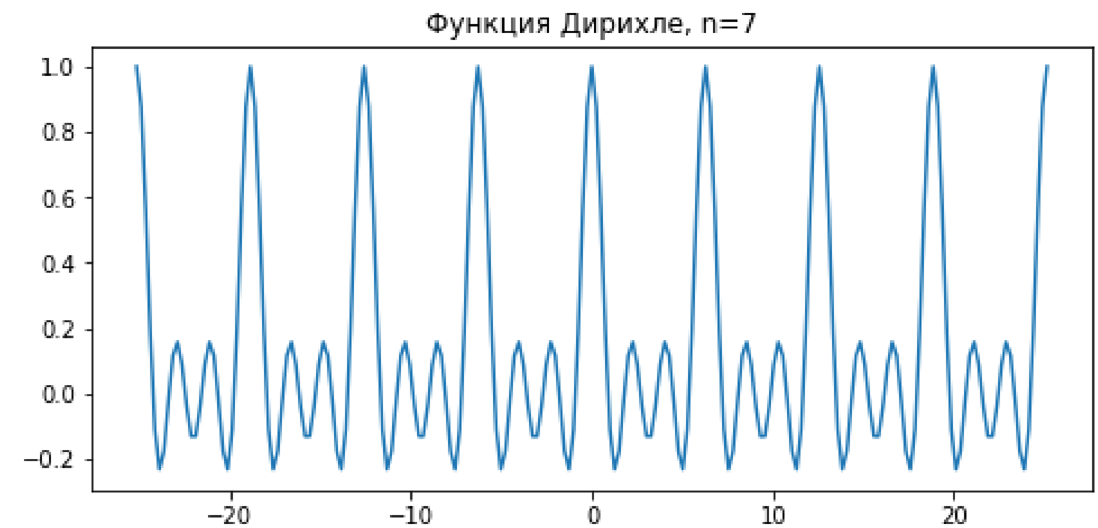
```
In [89]: import numpy as np
         import matplotlib.pyplot as plt
         from scipy import signal

         Fs = 8000 # discrete frequency
         t = np.linspace(0, 0.5, Fs)
         fig = plt.figure()
         plt.title('  ')
         plt.plot(t, signal.sawtooth(2 * np.pi * 5  * t))
         fig.savefig('picturesNote/008_sawtooth.png', dpi=100)
         plt.show()
```

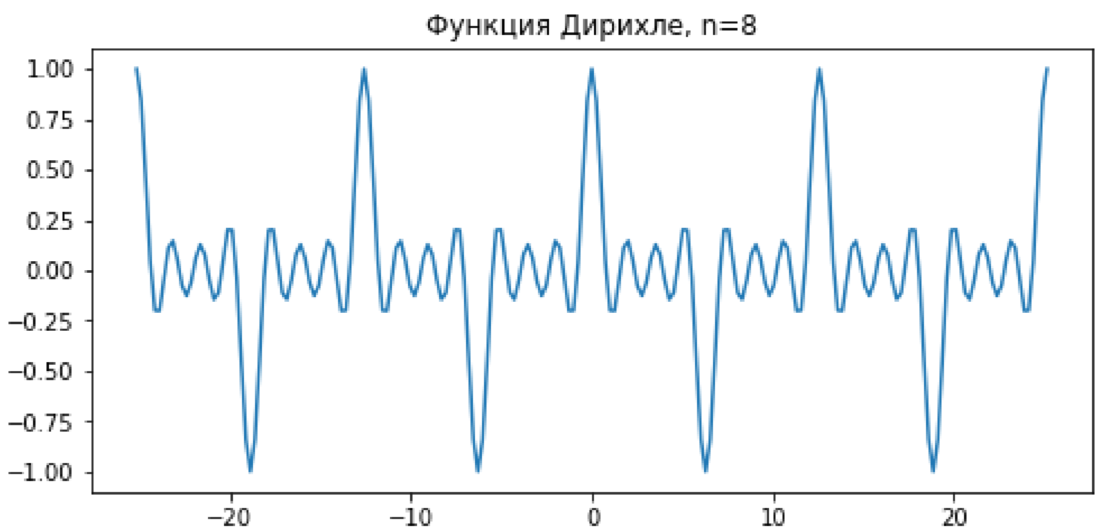


, :

$$diric_n(t) = \frac{\sin(nx/2)}{nsin(nx/2)}, n \in \mathbb{Z} \tag{4}$$

In [90]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from scipy import special


x = np.linspace(-8*np.pi, 8*np.pi, num=201)
fig = plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(x, special.diric(x, 7))
plt.title(' , n={}'.format(7))
plt.subplot(2, 1, 2)
plt.plot(x, special.diric(x, 8))
plt.title(' , n={}'.format(8))
fig.savefig('pictures/008_dirichle.png', dpi = 200)
plt.show()
```





1.4

python JupyterNotebook . : - - - -