Оглавление

1

Сглаживание сигнала

$$x[n] = s[n] + d[n]$$

s[n]=2*n*(0.9^n), n=[0,50] сигнал

d[n] = 0.8*(w[n]-0.5), w[n] — массив случайных величин с равномерным распределением на [0,1].

Задача: обработать сигнал x[n], чтобы получить приемлемую аппроксимацию сигнала s[n].

Решение:

$$y[n] = \frac{1}{3}(x[n-1] + x[n] + x[n+1])$$

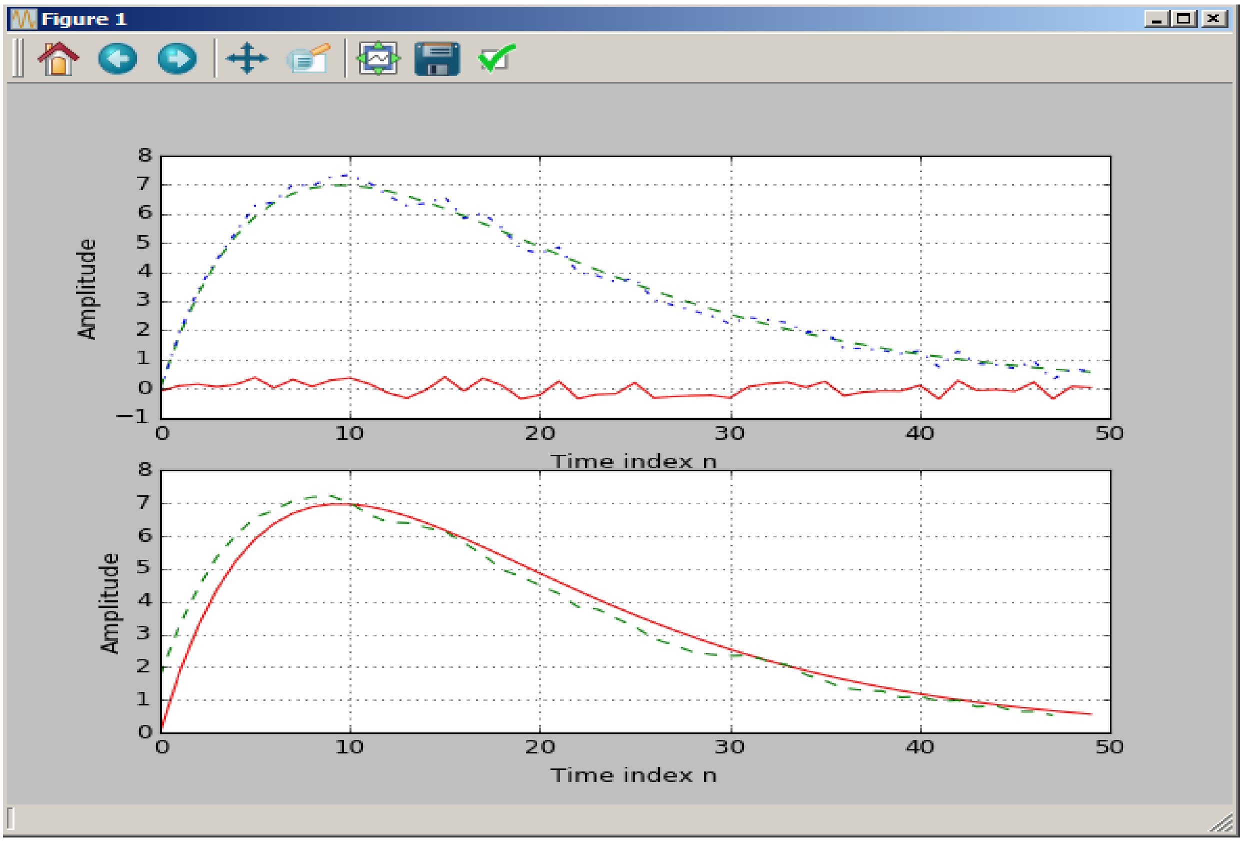


Рис. 1.

```python
n = 50
s = [2 * m * 0.9**m for m in range(n)]

w = np.random.uniform(0.0, 1.0, n)
d = [0.8 * (w[m] - 0.5) for m in range(n)]

x = [(s[m] + d[m]) for m in range(n)]
y = [((x[m-1] + x[m] + x[m+1]) / 3.0) for m in range(1, n - 1)]

fig, ax = plt.subplots(2, 1)

ax[0].plot(d, color='r', linestyle='-')
ax[0].plot(s, color='g', linestyle='--')
ax[0].plot(x, color='b', linestyle='-.')
ax[0].grid()
ax[0].set_xlabel('Time index n')
ax[0].set_ylabel('Amplitude')

ax[1].plot(s, color='r', linestyle='-')
ax[1].plot(y, color='g', linestyle='--')
ax[1].grid()
ax[1].set_xlabel('Time index n')
ax[1].set_ylabel('Amplitude')

plt.show()
```

2

Амплитудно-модулированный сигнал

Сложные сигналы можно получить, обрабатывая простые сигналы. Например, получение амплитудно-модулированного сигнала: высокочастотный сигнал

$$x_H[n] = \cos(\omega_H n)$$

Модулируется низкочастотным сигналом

$$x_L[n] \cos(\omega_L n)$$

По формуле:

$$y[n] = A(1 + m \cdot x_L[n])x_H[n] = A(1 + m \cdot \cos(\omega_L n))\cos(\omega_H n)$$

Для индекса модуляции m = 0.4, частот wH=2*pi* 0.1 и wL =2*pi* 0.01.

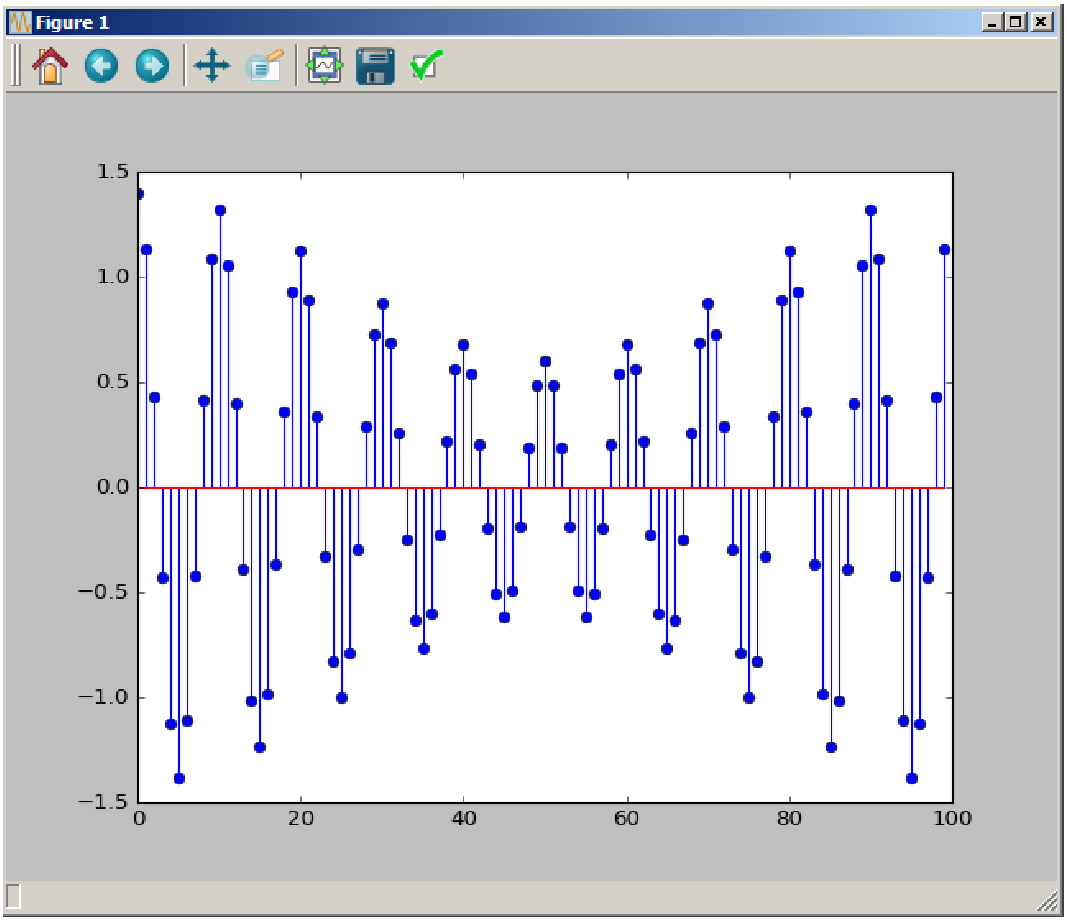


Рис. 2.

```
n = np.arange(100)
w_H = 2 * np.pi * 0.1
w_L = 2 * np.pi * 0.01

x_H = np.cos(w_H * n)
x_L = np.cos(w_L * n)

A = 1.0
m = 0.4

y = A * (1.0 + m * x_L) * x_H

plt.plot(y, '.')
plt.stem(n, y)
plt.show()
```

3

Нарисовать линейно-частотно модулированный сигнал (рис. 3)

y[n]=cos(a*n^2), a = pi/2/100

Какая минимальная и максимальная частота в этом сигнале?

cos (2 * PI * f * n) = cos ((0.5*100) * PI * n^2)

f = (0.5 / 100) / 2 * n

f_min(0) = 0

f_max(100) = 0.25

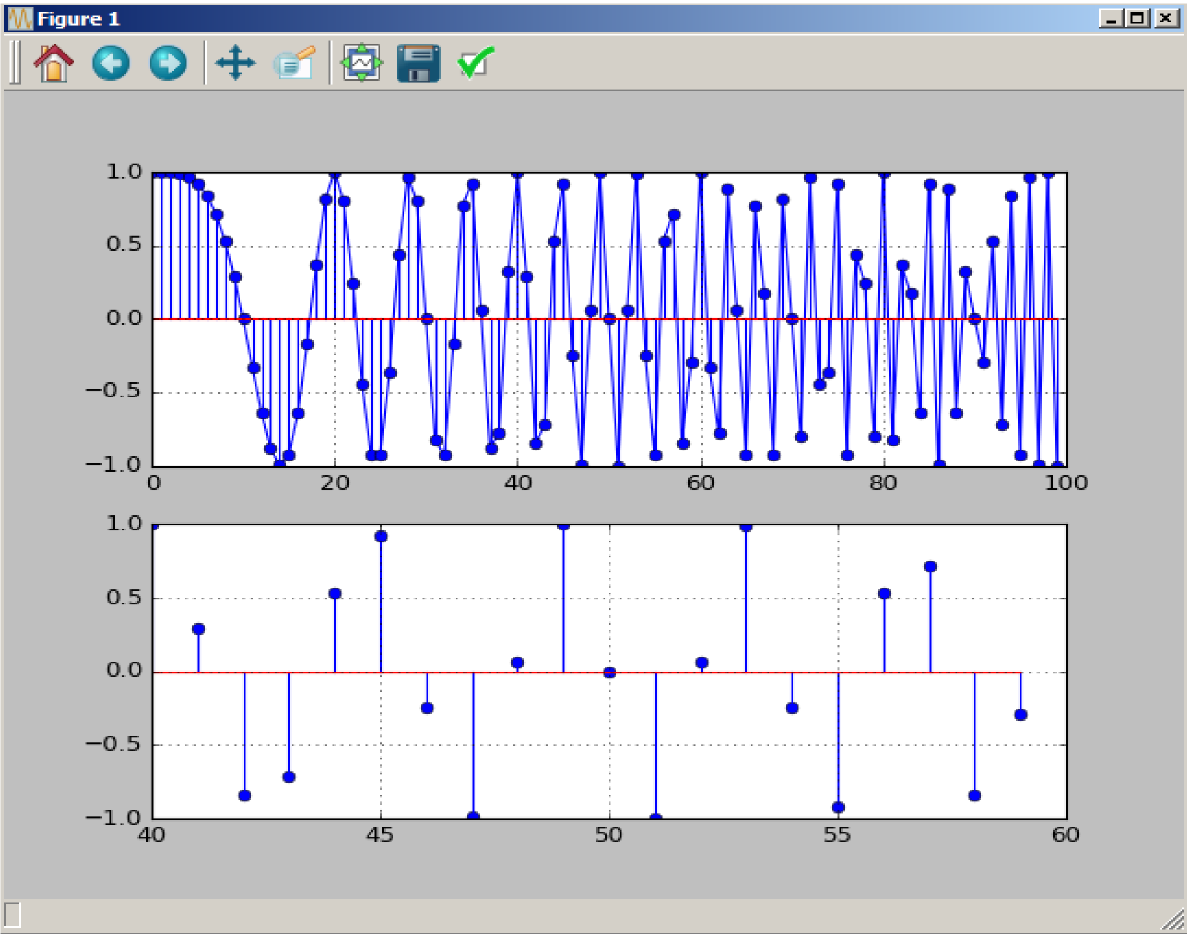
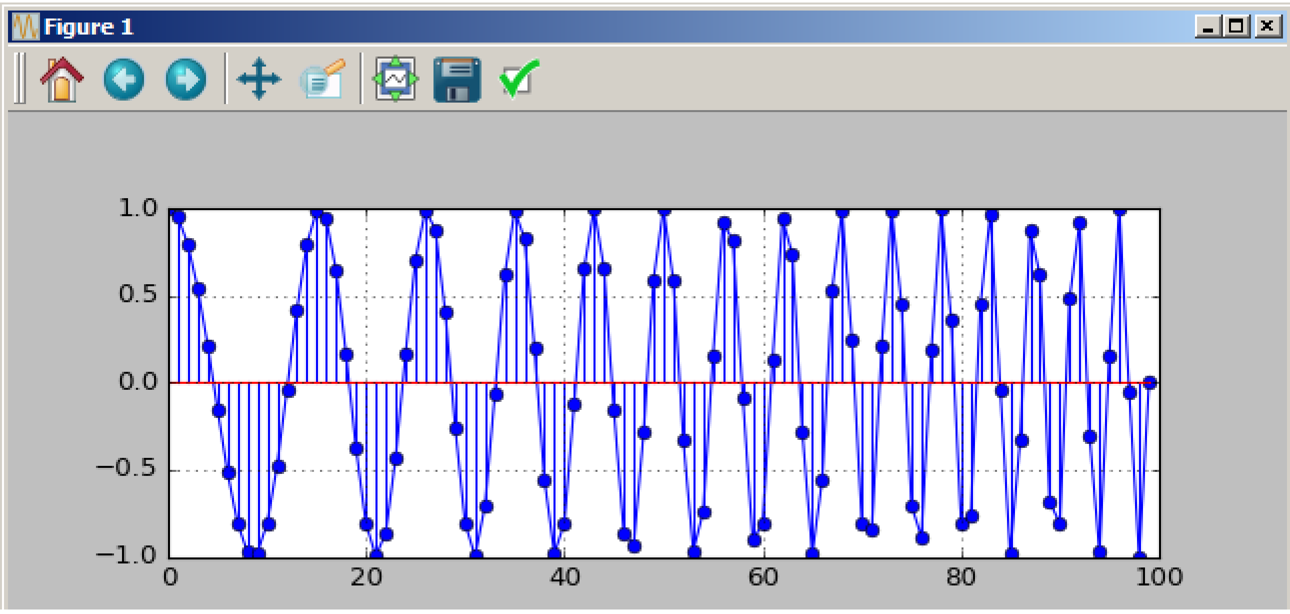


Рис. 3

```
n = np.arange(100)
a = np.pi / 2.0 / 100.0
y = np.cos(a * n**2)

fig, ax = plt.subplots(2, 1)

ax[0].plot(y, '-')
ax[0].stem(n, y)
ax[0].grid()

ax[1].stem(n[40:60], y[40:60])
ax[1].grid()

plt.show()
```

Постройте рисунок с минимальной частотой 0.1 и максимальной частотой 0.3.



4

Система, описываемая уравнением

$$y[n] = \frac{1}{3}(x[n-1] + x[n] + x[n+1])$$

не является детерминированной.

Детерминированная система

$$y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$$

В общем виде

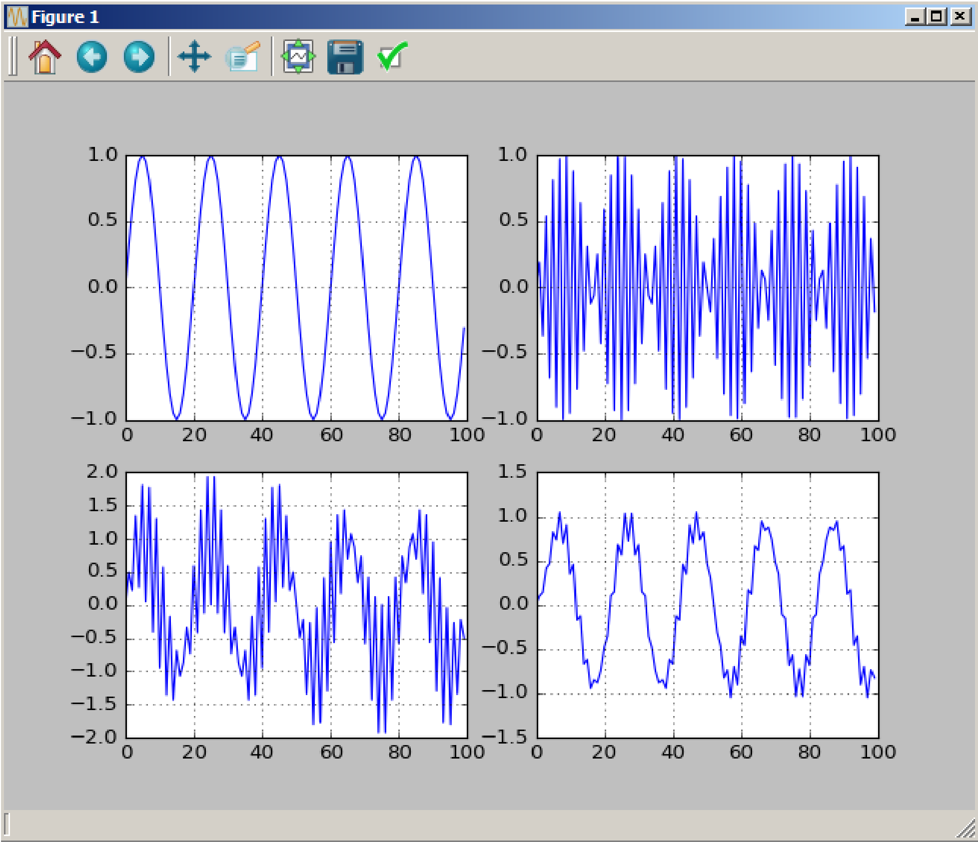
$$y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$$

(M точечный сглаживающий КИХ фильтр — фильтр скользящего среднего).

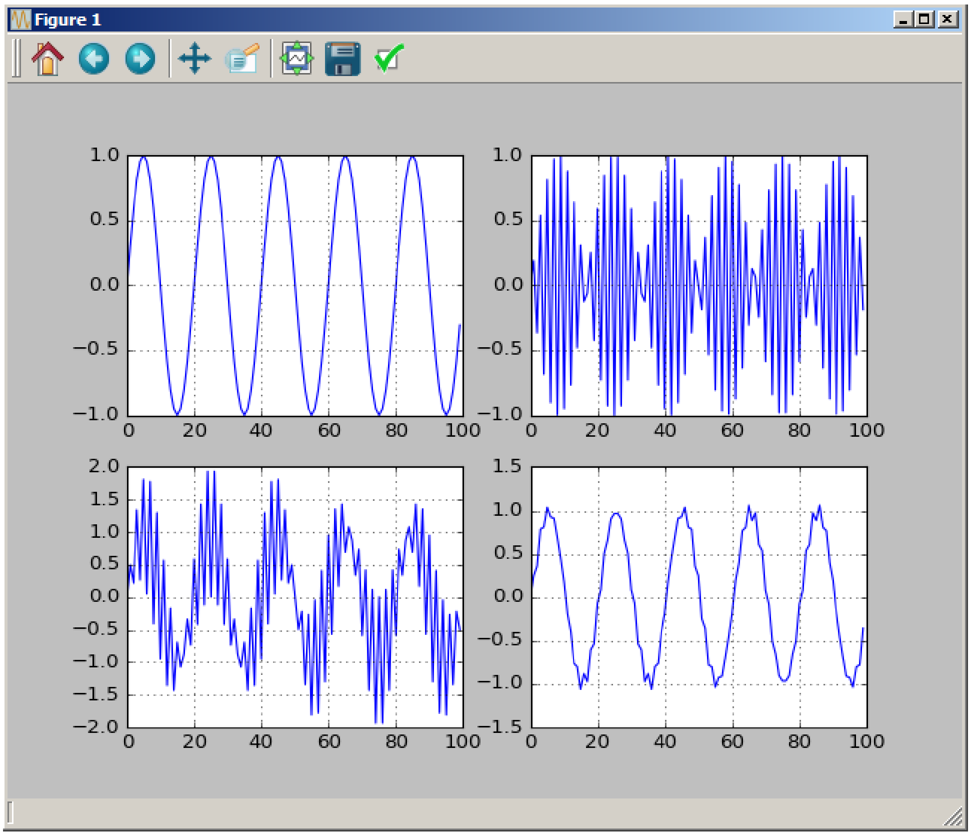
Задача. Даны два сигнала. Первый (s1) с низкой частотой f1=0.05, второй (s2) с высокой частотой f2= 0.47. x=s1+s2. Получить сигнал y (M=5). Построить графики на рис. 4.

Уравнение y[n] можно реализовать так:

```
from scipy.signal import lfilter
m=5
num=np.ones(m)
y=lfilter(num,1,x)/m
```

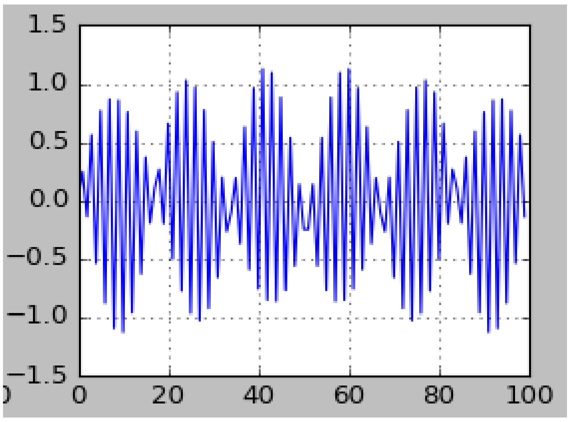


Постройте рисунок для M=2.



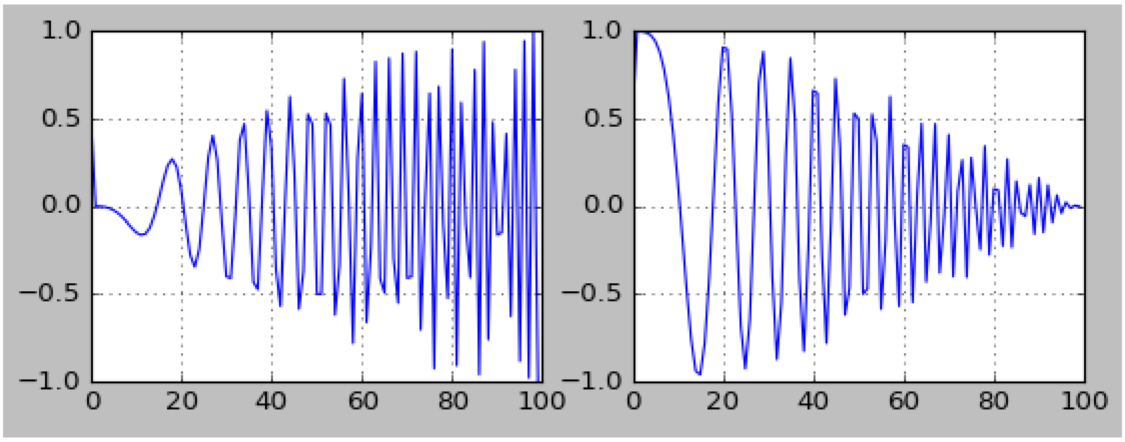
Какой сигнал подавляется?

Постройте график для y[n]=0.5(x[n] - x[n - 1])



Какой сигнал подавляется?

Обработайте двумя этими способами сигнал из задания 3 (линейно-частотно модулированный).




```
n = np.arange(100)
s1 = np.sin(2 * np.pi * 0.05 * n)
s2 = np.sin(2 * np.pi * 0.47 * n)
x = s1 + s2

m = 2
num = np.ones(m)
y = lfilter(num, 1, x) / m

y2 = np.zeros(100)
y2[0] = x[0] / 2.0
for n in np.arange(1,100):
    y2[n] = (x[n]-x[n-1]) / 2.0
```

```

a = np.pi / 2 / 100
M = 2
n2 = np.arange(100)
x2 = np.cos(a * n2**2)

y3 = lfilter(np.ones(M), 1, x2) / M

y4 = np.zeros(100)
y4[0] = x2[0]/2
for n in np.arange(1,100):
    y4[n] = (x2[n]-x2[n-1]) / 2



fig, ax = plt.subplots(2, 2)

ax[0][0].plot(s1)
ax[0][0].grid()

ax[0][1].plot(s2)
ax[0][1].grid()

ax[1][1].plot(y3)
ax[1][1].grid()

ax[1][0].plot(y4)
ax[1][0].grid()

plt.show()
```

5

Важный подкласс линейный стационарных систем описывается уравнением [Оппенгейм, стр. 54]

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]$$

Рассмотрим систему

$$y[n]-0.4\,y[n-1]+0.75\,y[n-2] = 2.2403\,x[n]+2.4908\,x[n-1]+2.2403\,x[n-2]$$

Проверим свойство линейности этой системы. Согласно [Оппенгейм, стр. 39]: (lfilter (coeff_x, coeff_y , sequence_x))

Класс *линейных систем* определяется по принципу суперпозиции. Если $y_1[n]$ и $y_2[n]$ — отклики системы на сигналы $x_1[n]$ и $x_2[n]$, то систему называют *линейной* тогда и только тогда, когда

$$T\{x_1[n]+x_2[n]\} = T\{x_1[n]\} + T\{x_2[n]\} \quad \text{и} \quad T\{ax[n]\} = aT\{x[n]\}, \tag{2.26}$$

где a — произвольная константа. Первое из свойств называют *аддитивностью*, а второе — *однородностью*. Оба свойства можно записать одной формулой по принципу суперпозиции:

$$T\{ax_1[n]+bx_2[n]\} = aT\{x_1[n]\} + bT\{x_2[n]\}, \tag{2.27}$$

где a и b — произвольные константы. Последнее соотношение легко может быть переписано для нескольких сигналов, а именно

$$\text{если} \quad x[n] = \sum_k a_k x_k[n], \quad \text{то} \quad y[n] = \sum_k a_k y_k[n],$$

где $y_k[n]$ — реакция системы на поданный сигнал $x_k[n]$.

Сформировать сигнал x1[n] = cos(2*pi*0.1*n);

Сформировать сигнал x2[n] = cos(2*pi*0.4 *n);

Сформировать сигнал x[n] = a*x1[n]+b*x2[n], a=2, b=-3;

Рассчитать сигнал y1[n] на выходе системы (вход x1[n]);

Рассчитать сигнал y2[n] на выходе системы (вход x2[n]);

Рассчитать сигнал y[n] на выходе системы (вход x[n]);

Найти разность y[n]-(a*y1[n]+b*y2[n]).

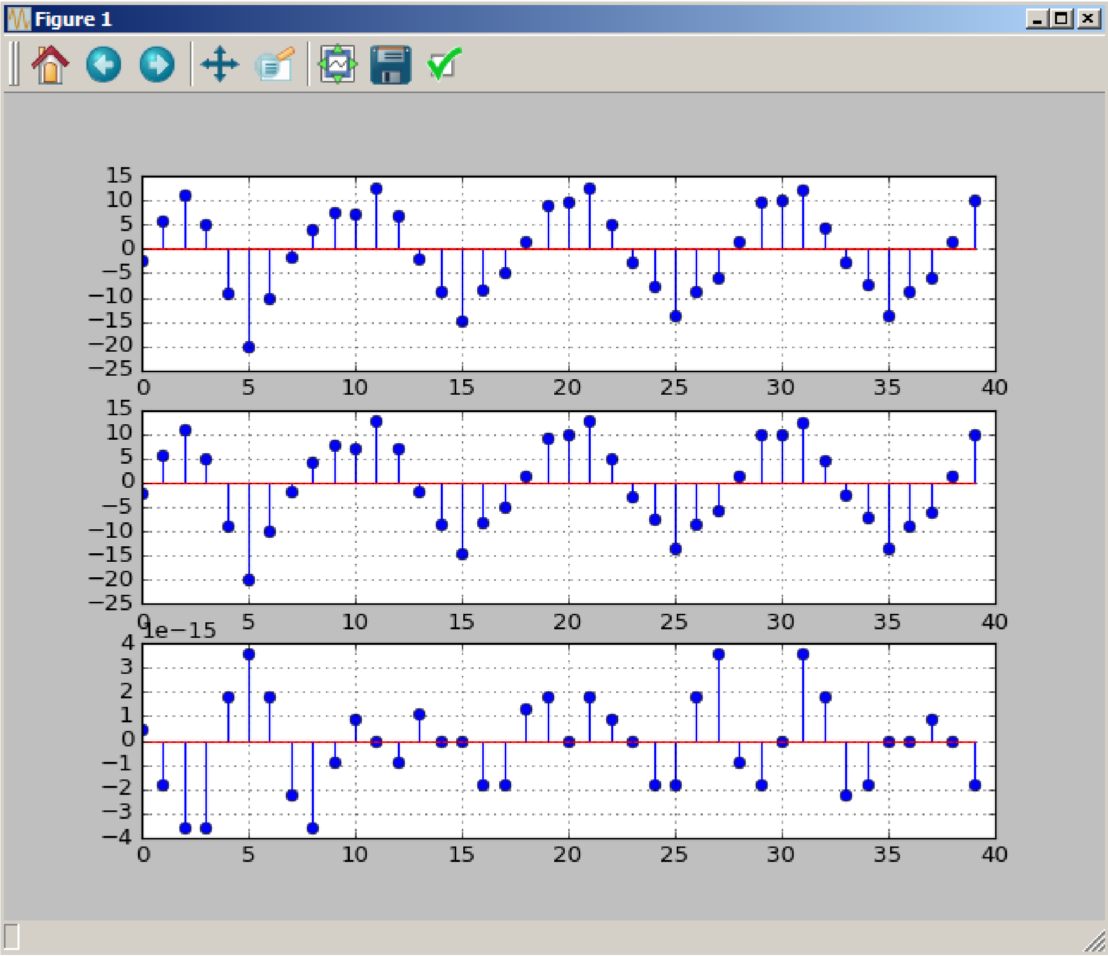


Рис. 5.

```
n = np.arange(40)
x1 = np.cos(2 * np.pi * 0.1 * n)
x2 = np.cos(2 * np.pi * 0.4 * n)

a =  2
b = -3
x = a * x1 + b * x2

coeff_x = [2.2403, 2.4908, 2.2403]
coeff_y = [1, -0.4, 0.75]

y1 = lfilter(coeff_x, coeff_y, x1)
y2 = lfilter(coeff_x, coeff_y, x2)
y  = lfilter(coeff_x, coeff_y, x)

diff = y - (a * y1 + b * y2)

fig, ax = plt.subplots(3, 1)

ax[0].stem(y)
ax[0].grid()
```

```
ax[1].stem(a * y1 + b * y2)
ax[1].grid()

ax[2].stem(diff)
ax[2].grid()

plt.show()
```

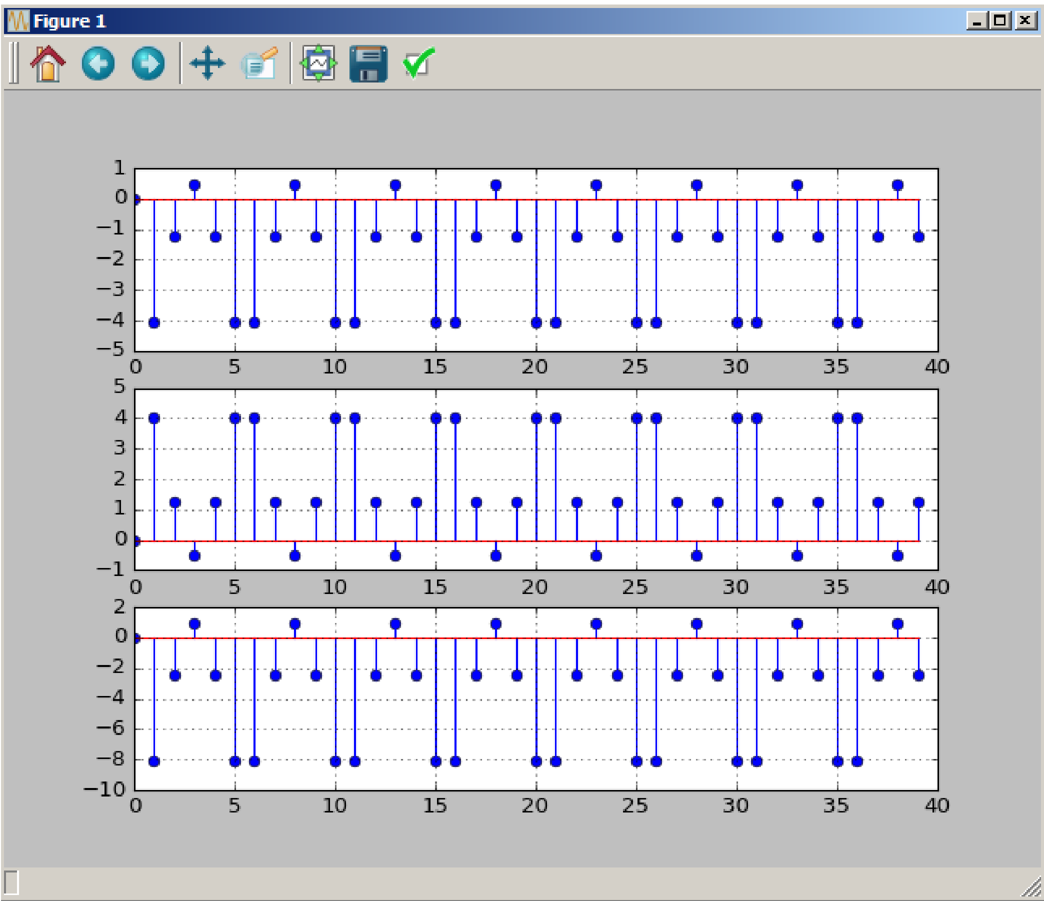
Система линейная? – Система не линейная.

Проверьте для различных частот и значений a, b.

Проверьте аналогичным способом линейность системы

$$y[n] = x[n]\,x[n-1]$$

Постройте графики.



Система не линейная.

```python
n = np.arange(40)
x1 = np.cos(2 * np.pi * 0.1 * n)
x2 = np.cos(2 * np.pi * 0.4 * n)

a =  2
b = -3
x = a * x1 + b * x2

y1 = np.zeros(40)
y2 = np.zeros(40)
y  = np.zeros(40)

for k in np.arange(1, 40):
    y1[k] = x1[k] * x1[k-1]
    y2[k] = x2[k] * x2[k-1]
    y[k]  = x[k] * x[k-1]

diff = y - (a * y1 + b * y2)

fig, ax = plt.subplots(3, 1)

ax[0].stem(y)
ax[0].grid()

ax[1].stem(a * y1 + b * y2)
ax[1].grid()

ax[2].stem(diff)
ax[2].grid()

plt.show()
```

6

Проверим, что система

$$y[n]-0.4\,y[n-1]+0.75\,y[n-2] = 2.2403\,x[n]+2.4908\,x[n-1]+2.2403\,x[n-2]$$

стационарна. Согласно [Оппенгейм, стр. 40]

К *стационарным* относят системы, для которых временной сдвиг (или задержка) входной последовательности индуцирует соответствующий сдвиг выходной последовательности. Более формально определение выглядит так. Пусть дискретная система определена формулой $y[n] = T\{x[n]\}$. Она называется стационарной, если для любой входной последовательности $x[n]$ и произвольного целого числа n_0 выполнено соотношение $T\{x[n-n_0]\} = y[n-n_0]$. Стационарные системы иногда еще называют системами, *инвариантными относительно сдвигов*.

Сформировать сигнал x[n] = a* cos(2*pi*0.1*n)+b*cos(2*pi*0.4 *n)], a=2, b=-3;

Сформировать сигнал x[n-10];

Рассчитать отклик системы y[n] на вход x[n];

Рассчитать отклик системы yd[n] на вход x[n-10];

Показать, что yd и y[n-10] совпадают.

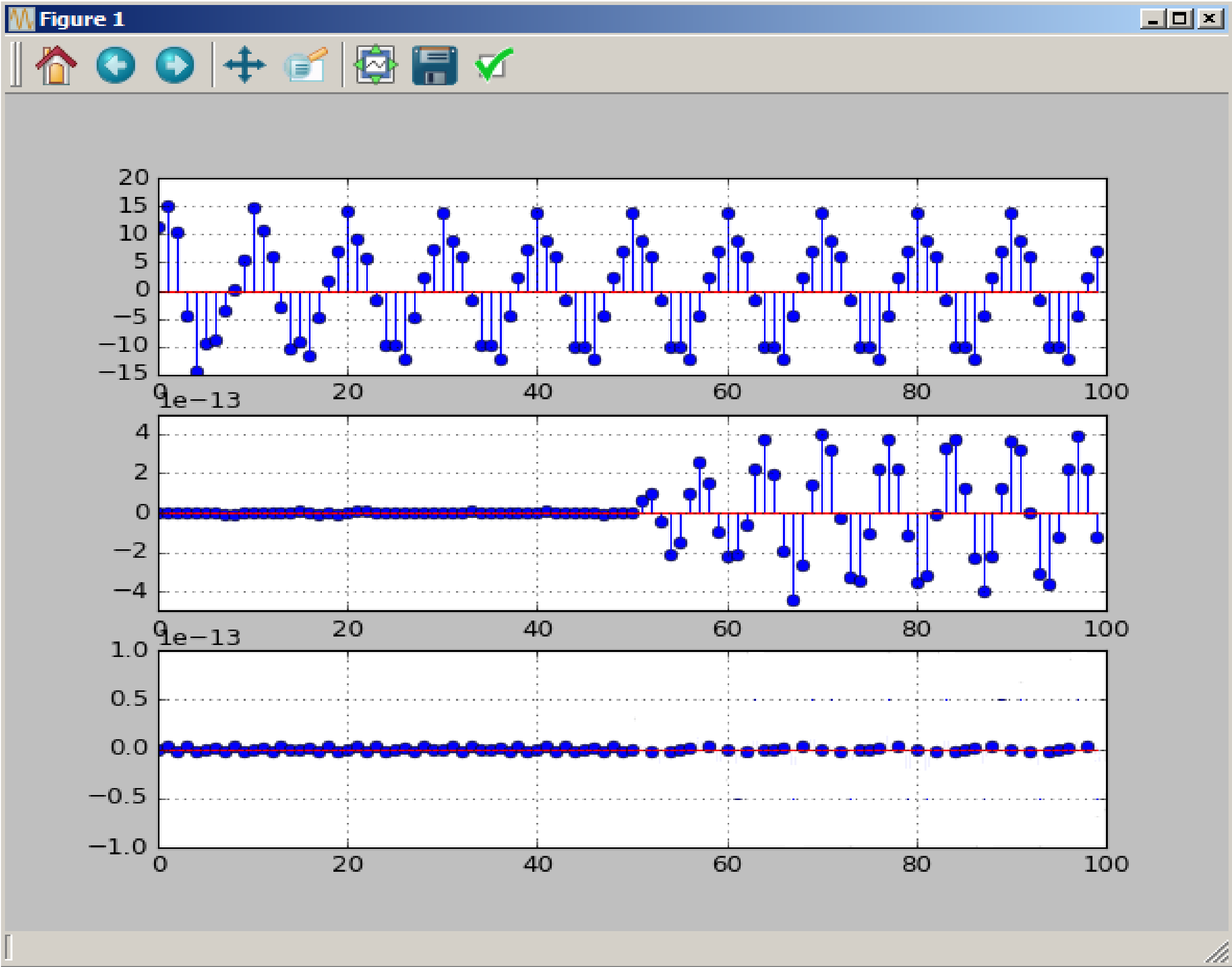
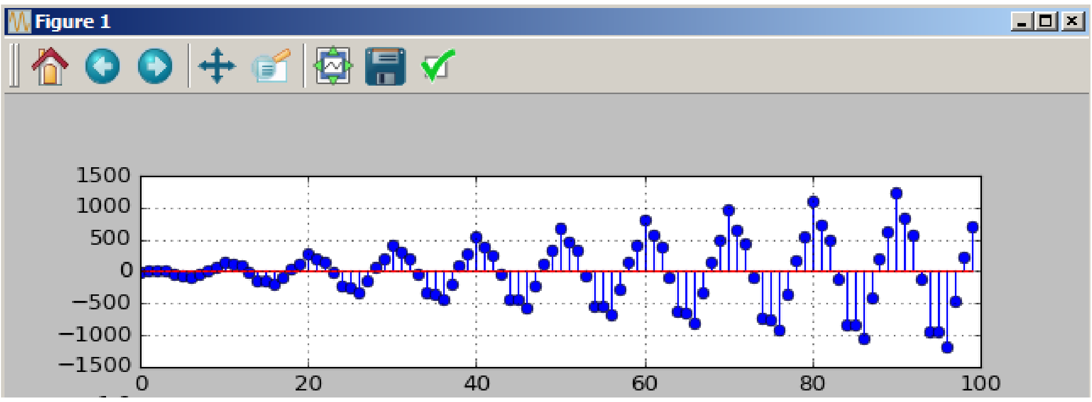


Рис. 6.

Система стационарна.

$$y[n] = n\,x[n] + x[n-1]$$



Система не является стационарной.

```
n = np.arange(100)
x1 = np.cos(2 * np.pi * 0.1 * n)
x2 = np.cos(2 * np.pi * 0.4 * n)

x1_ = np.cos(2 * np.pi * 0.1 * (n-10))
x2_ = np.cos(2 * np.pi * 0.4 * (n-10))

a = 2
b = 3
x  = a * x1  + b * x2
x_ = a * x1_ + b * x2_

coeff_x = [2.2403, 2.4908, 2.2403]
coeff_y = [1, -0.4, 0.75]

y = lfilter(coeff_x, coeff_y, x)
yd = lfilter(coeff_x, coeff_y, x_)

diff_y = y - yd
diff_x = x - x_

x_n1_1 = np.cos(2 * np.pi * 0.1 * (n-1))
x_n1_2 = np.cos(2 * np.pi * 0.4 * (n-1))
x_n1   = a * x_n1_1 + b * x_n1_2
x_n    = n * x

y_n = lfilter(coeff_x, coeff_y, x_n + x_n1)

fig, ax = plt.subplots(3, 1)

ax[0].stem(y_n)
ax[0].grid()

ax[1].stem(diff_y)
ax[1].grid()

ax[2].stem(diff_x)
ax[2].grid()

plt.show()
```

7

Линейные стационарные системы [Оппенгейм, раздел 2.4, стр. 42]

$$y[n] = -\sum_{k=1}^{N} \frac{d_k}{d_0} y[n-k] + \sum_{k=0}^{M} \frac{p_k}{d_0} x[n-k]$$

$$y[n] - 0.4\,y[n-1] + 0.75\,y[n-2] = 2.2403\,x[n] + 2.4908\,x[n-1] + 2.2403\,x[n-2]$$

Построить импульсную характеристику системы

Для этого можно поступить так:

```
N = 40
x = np.zeros(N)
x[0] = 1
h = lfilter(num, den, x)
nn = np.arange(N)
ax[0].stem(nn,h,'-.')
```

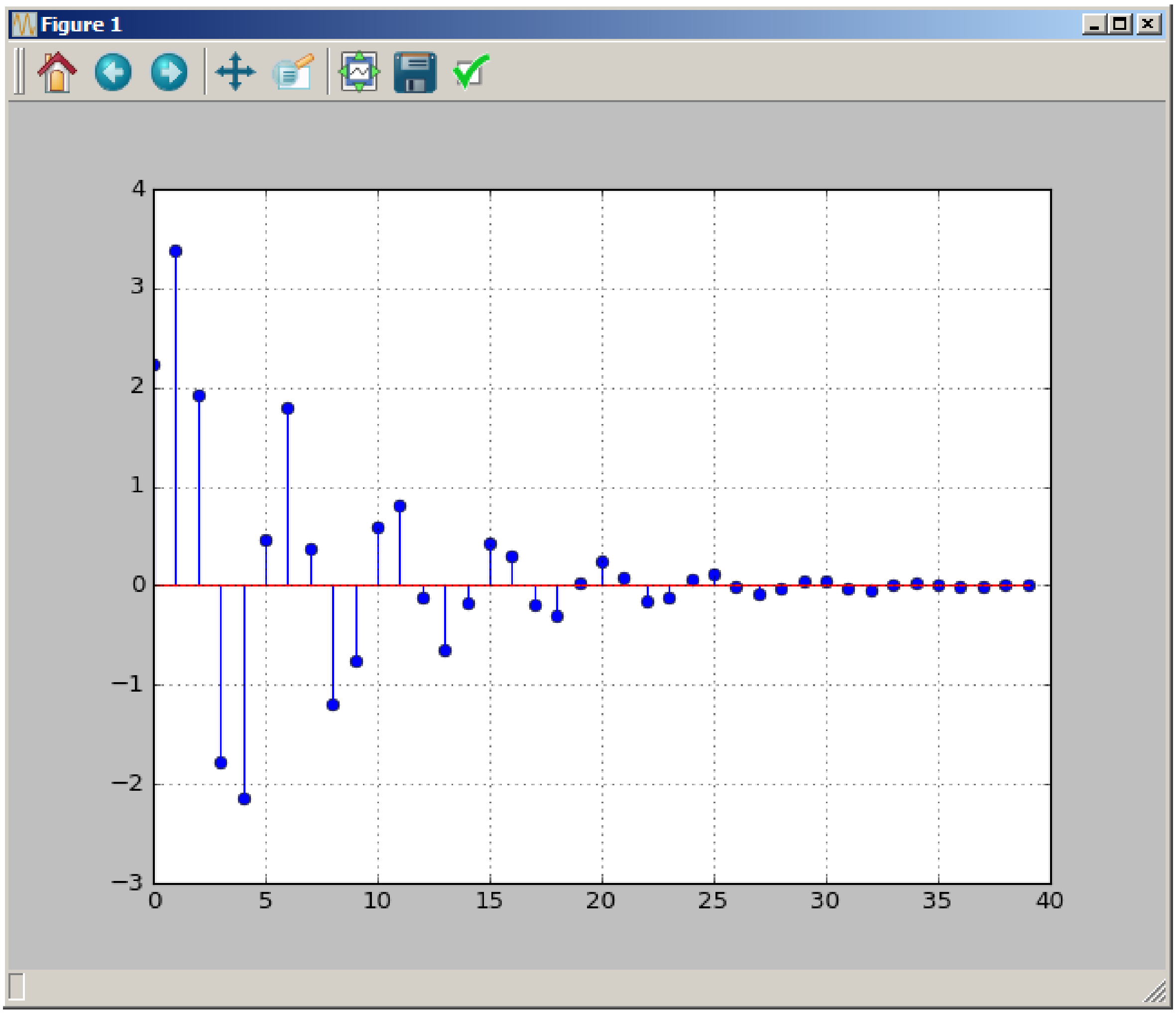
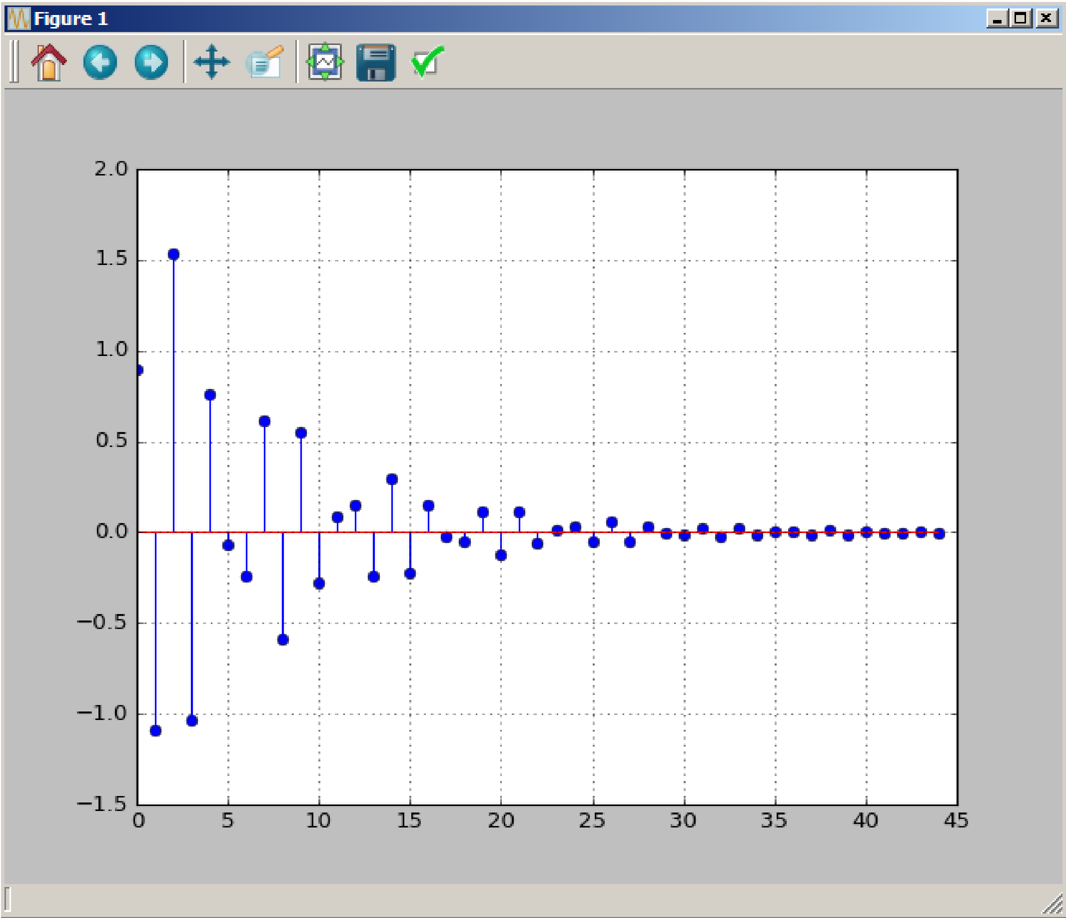


Рис. 7.

Постройте график первых 45 отсчетов импульсной характеристики системы:

$$y[n] + 0.71\,y[n-1] - 0.46\,y[n-2] - 0.62\,y[n-3]$$
$$= 0.9\,x[n] - 0.45\,x[n-1] + 0.35\,x[n-2] + 0.002\,x[n-3]$$




```
N = 40
x = np.zeros(N)
x[0] = 1
coeff_x = [2.2403, 2.4908, 2.2403]
coeff_y = [1, -0.4, 0.75]

h = lfilter(coeff_x, coeff_y, x)
nn = np.arange(N)

N2 = 45
coeff_x1 = [0.9, -0.45, 0.35, 0.002]
coeff_y1 = [1,   0.71, -0.46, -0.62]
x = np.zeros(N2)
x[0] = 1
h = lfilter(coeff_x1, coeff_y1, x)
nn = np.arange(N2)

plt.stem(nn, h)
plt.grid()

plt.show()
```

8

Показать, что выход системы 4-го порядка

$$y[n] + 1.6\,y[n-1] + 2.28\,y[n-2] + 1.325\,y[n-3] + 0.68\,y[n-4]$$
$$= 0.06\,x[n] - 0.19\,x[n-1] + 0.27\,x[n-2] - 0.26\,x[n-3] + 0.12\,x[n-4]$$

совпадает с выходом каскада из двух последовательных систем 2-го порядка:

первая система в каскаде

$$y_1[n] + 0.9\,y_1[n-1] + 0.8\,y_1[n-2] = 0.3\,x[n] - 0.2\,x[n-1] + 0.4\,x[n-2]$$

вторая система в каскаде

$$y_2[n] + 0.7\,y_2[n-1] + 0.85\,y_2[n-2] = 0.2\,y_1[n] - 0.5\,y_1[n-1] + 0.3\,y_1[n-2]$$

Для этого надо построить импульсную характеристику системы 4-го порядки и сравнить ее с импульсной характеристикой каскада из двух последовательных систем 2-го порядка.

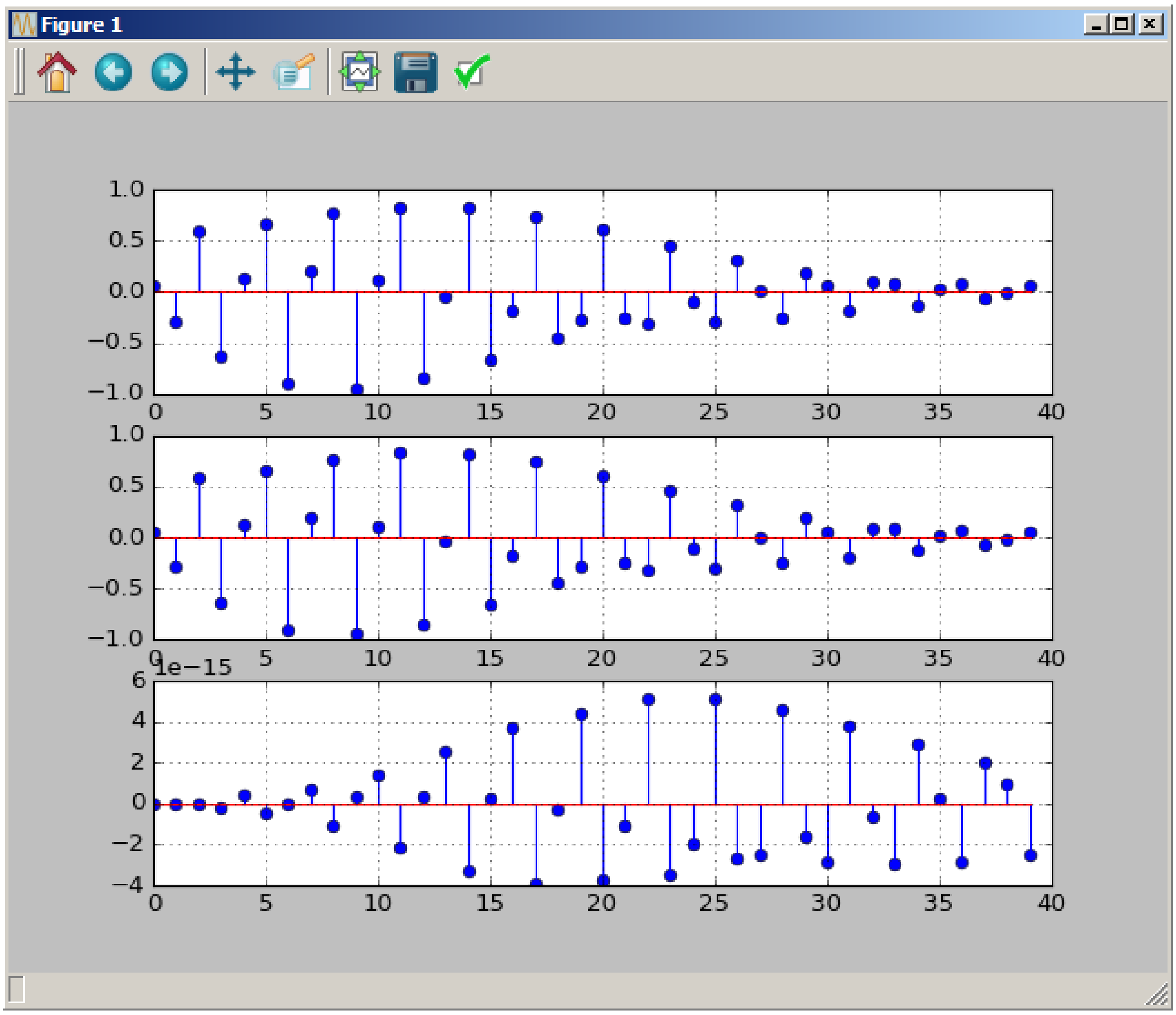


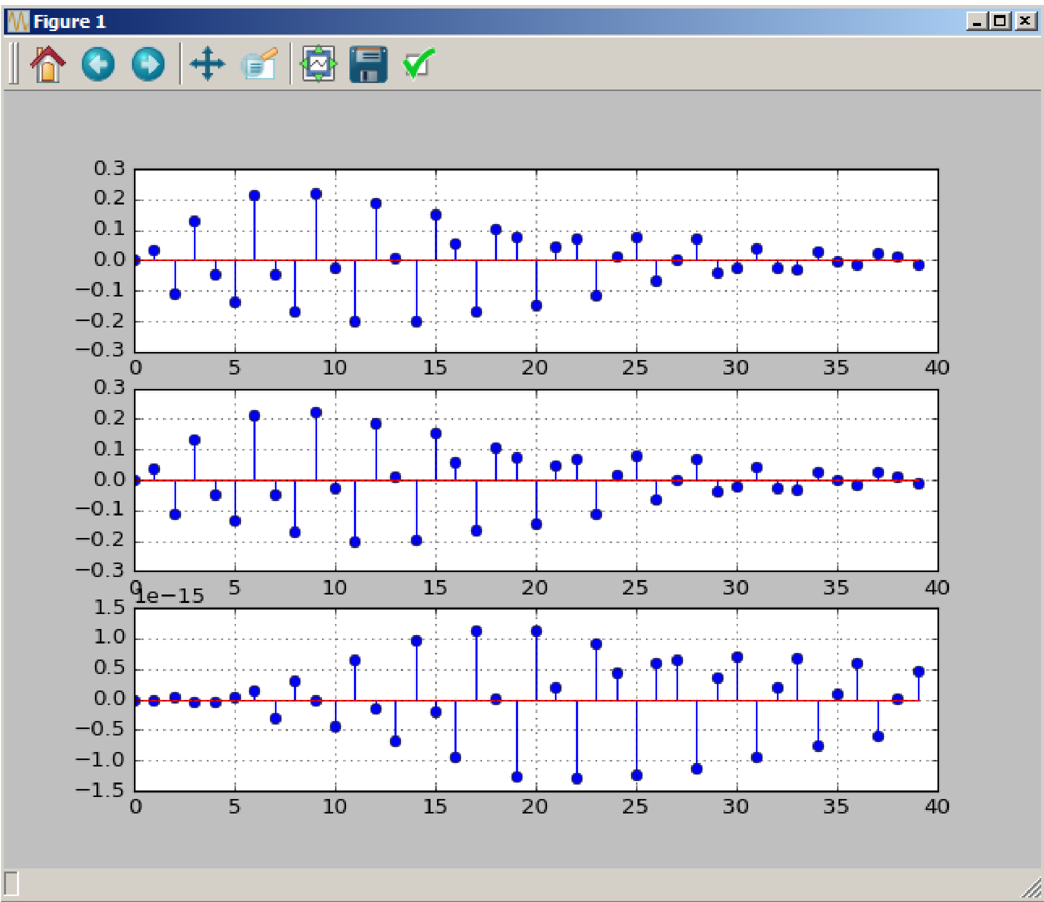
Рис. 8.

```
N = 40
x = np.zeros(N)
x[0] = 1

coeff_x_4 = [0.06, -0.19, 0.27, -0.26, 0.12]
coeff_y_4 = [1, 1.6, 2.28, 1.325, 0.68]

coeff_x1_2 = [0.3, -0.2, 0.4]
coeff_y1_2 = [1, 0.9, 0.8]

coeff_x2_2 = [0.2, -0.5, 0.3]
coeff_y2_2 = [1, 0.7, 0.85]

h_4 = lfilter(coeff_x_4, coeff_y_4, x)
nn_4 = np.arange(N)

h1_2 = lfilter(coeff_x1_2, coeff_y1_2, x)
nn1_2 = np.arange(N)

h2_2 = lfilter(coeff_x2_2, coeff_y2_2, h1_2)
nn2_2 = np.arange(N)
```

```
diff = h_4 - h2_2

fig, ax = plt.subplots(3, 1)

ax[0].stem(nn_4, h_4)
ax[0].grid()

ax[1].stem(nn1_2, h2_2)
ax[1].grid()

ax[2].stem(diff)
ax[2].grid()

plt.show()
```

Постройте графики, если на вход подается не единичный импульс, а синусоидальная последовательность.

Будут ли изменения, если в каскаде изменить порядок следования систем? Постройте график.



9

Рассчитать отклик системы с импульсной характеристикой

h = [3 2 1 -2 1 0 -4 0 3]

на вход

x = [1 -2 3 -4 3 2 1]

с помощью функции lfilter и функции вычисления свертки np.convolve ([Оппенгейм, стр. 43]).

Получить графики.

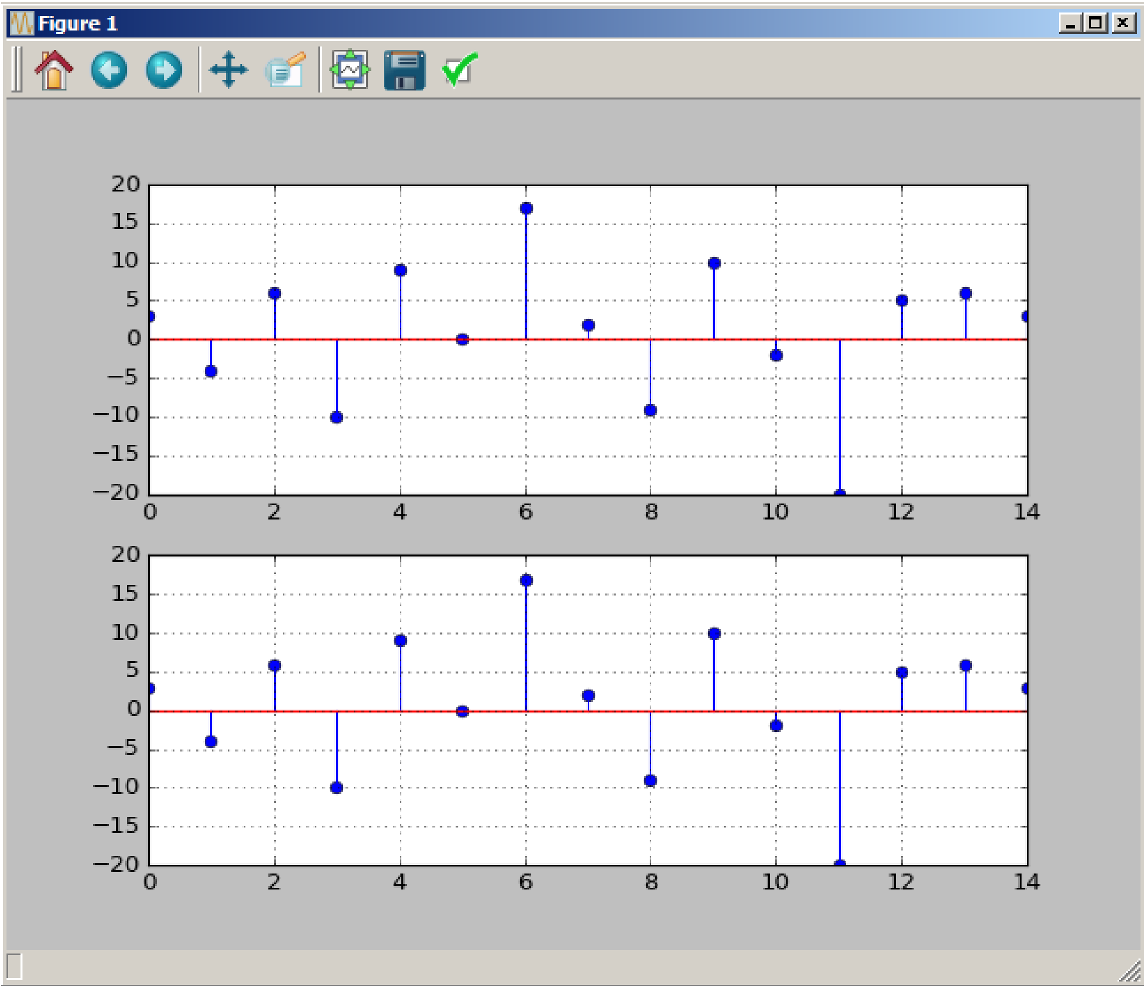


Рис. 9.

```
h = [3, 2, 1, -2, 1, 0, -4, 0, 3]
x = [1, -2, 3, -4, 3, 2, 1]

y_conv = convolve(x, h)
y_filter = lfilter(h, 1, np.concatenate((x,np.zeros(8)), axis=0))

fig, ax = plt.subplots(2, 1)

```

```
ax[0].stem(y_conv)
ax[0].grid()

ax[1].stem(y_filter)
ax[1].grid()

plt.show()
```

10

Даны две системы. Первая

$$y[n] = 0.5\,x[n] + 0.27\,x[n-1] + 0.77\,x[n-2]$$

Вторая

$$y[n] = 0.45\,x[n] + 0.5\,x[n-1] + 0.45\,x[n-2] + 0.53\,y[n-1] - 0.46\,y[n-2]$$

Найти отклик этих систем на входное воздействие:

$$x[n] = \cos\left(\frac{20\pi n}{256}\right) + \cos\left(\frac{200\pi n}{256}\right)$$

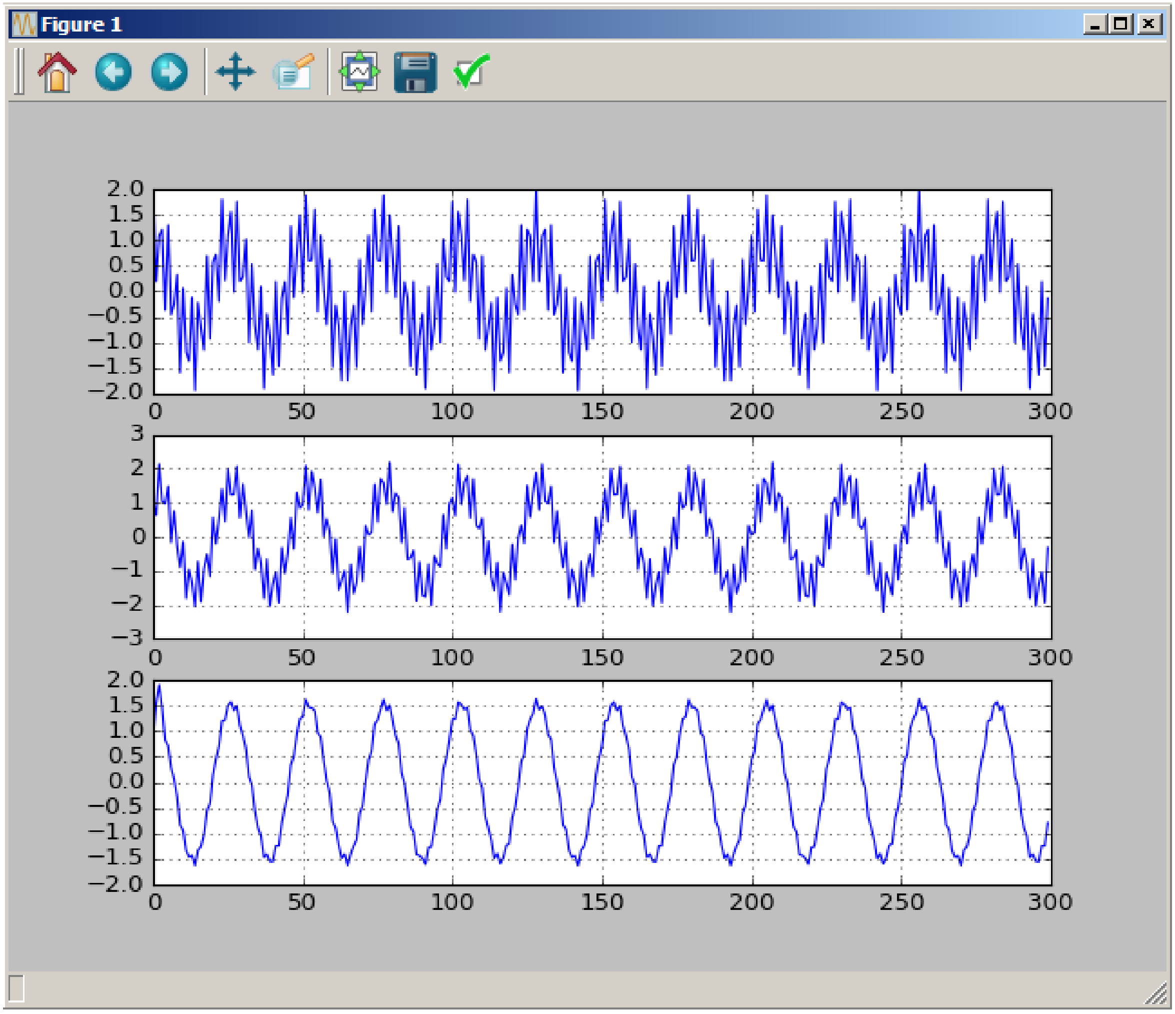


Рис. 10.

Обе системы — фильтры низких частот.

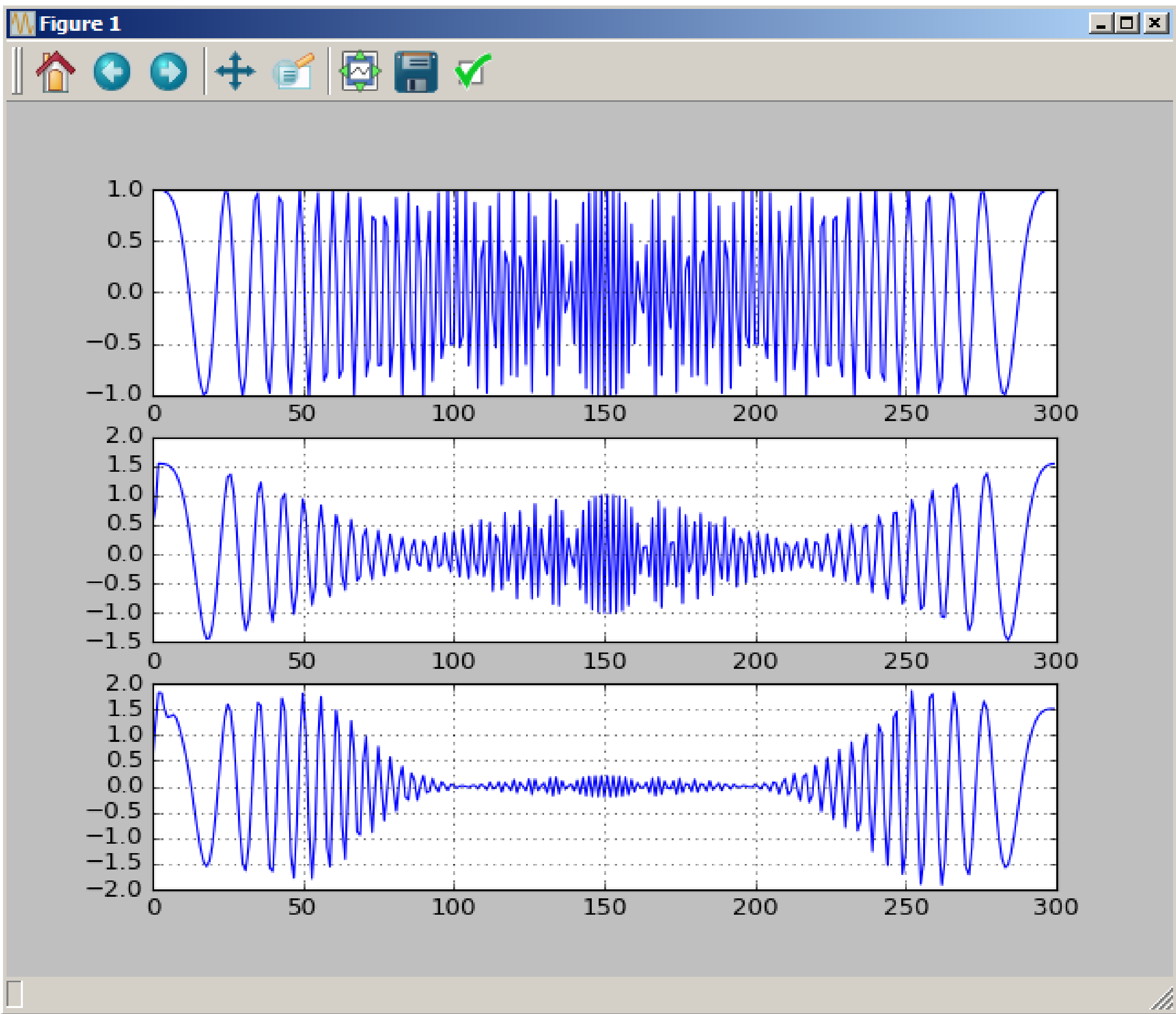
Какая система лучше подавляет высокие частоты? – Вторая система.

```
n = np.arange(300)
x1 = np.cos(20 * np.pi * n / 256)
x2 = np.cos(200 * np.pi * n / 256)
x = x1 + x2

coeff_x1 = [0.5, 0.27, 0.77]
coeff_y1 = [1]

coeff_x2 = [0.45, 0.5, 0.45]
coeff_y2 = [1, -0.53, 0.46]

y1 = lfilter(coeff_x1, coeff_y1, x)
y2 = lfilter(coeff_x2, coeff_y2, x)

```

```
fig, ax = plt.subplots(3, 1)

ax[0].plot(x)
ax[0].grid()

ax[1].plot(y1)
ax[1].grid()

ax[2].plot(y2)
ax[2].grid()

plt.show()
```

Найти отклик этих систем на линейно-частотно модулированную последовательность с мин. Частотой 0 и максимальной частотой 0.5.

Постройте графики.



```
n = np.arange(300)

x = np.cos(2 * np.pi * n**2 * 0.5 / 300)

coeff_x1 = [0.5, 0.27, 0.77]
coeff_y1 = [1]

coeff_x2 = [0.45, 0.5, 0.45]
coeff_y2 = [1, -0.53, 0.46]

y1 = lfilter(coeff_x1, coeff_y1, x)
y2 = lfilter(coeff_x2, coeff_y2, x)

fig, ax = plt.subplots(3, 1)

ax[0].plot(x)
ax[0].grid()

ax[1].plot(y1)
ax[1].grid()

ax[2].plot(y2)
ax[2].grid()

plt.show()
```