Оглавление

[1 2](#_Toc465286674)

[2 4](#_Toc465286675)

[3 6](#_Toc465286676)

[4 8](#_Toc465286677)

[5 10](#_Toc465286678)

[6 13](#_Toc465286679)

[7 16](#_Toc465286680)

[8 19](#_Toc465286681)

[9 22](#_Toc465286682)

[10 23](#_Toc465286683)

# 1

Сглаживание сигнала



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:



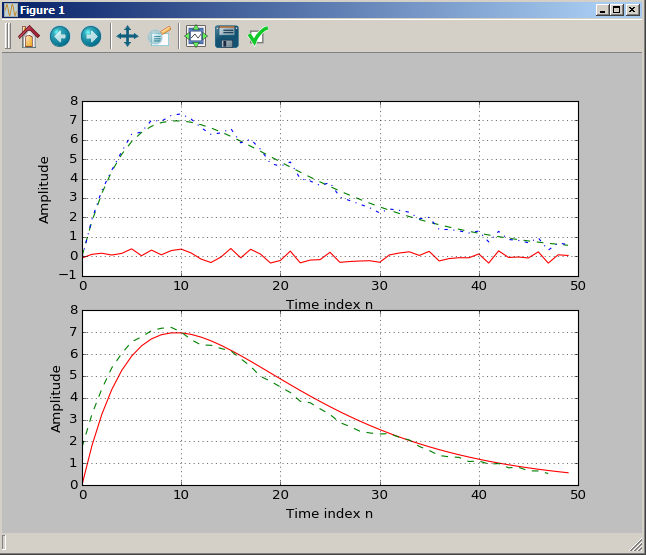


Рис. 1.

1. n = 50
2. s = [2 \* m \* 0.9\*\*m **for** m **in** range(n)]
3. w = np.random.uniform(0.0, 1.0, n)
4. d = [0.8 \* (w[m] - 0.5) **for** m **in** range(n)]
6. x = [(s[m] + d[m]) **for** m **in** range(n)]
7. y = [((x[m-1] + x[m] + x[m+1]) / 3.0) **for** m **in** range(1, n - 1)]
9. fig, ax = plt.subplots(2, 1)
11. ax[0].plot(d, color='r', linestyle='-')
12. ax[0].plot(s, color='g', linestyle='--')
13. ax[0].plot(x, color='b', linestyle='-.')
14. ax[0].grid()
15. ax[0].set\_xlabel('Time index n')
16. ax[0].set\_ylabel('Amplitude')
17. ax[1].plot(s, color='r', linestyle='-')
18. ax[1].plot(y, color='g', linestyle='--')
19. ax[1].grid()
20. ax[1].set\_xlabel('Time index n')
21. ax[1].set\_ylabel('Amplitude')
22. plt.show()

# 2

Амплитудно-модулированный сигнал

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



Модулируется низкочастотным сигналом



По формуле:

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

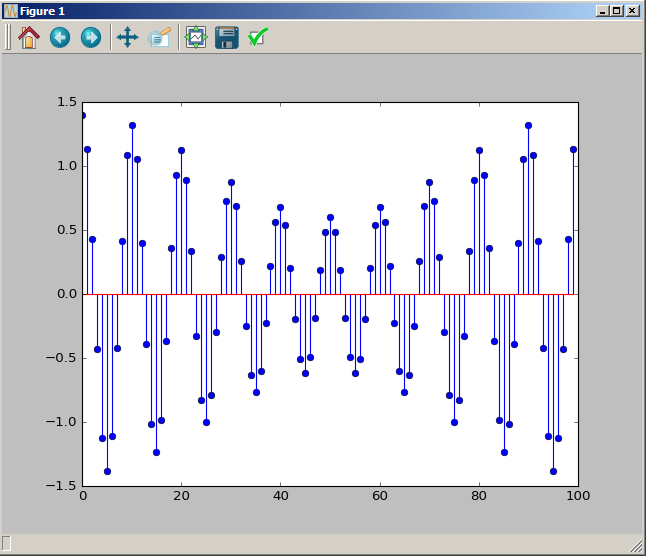


Рис. 2.

1. n = np.arange(100)
2. w\_H = 2 \* np.pi \* 0.1
3. w\_L = 2 \* np.pi \* 0.01
5. x\_H = np.cos(w\_H \* n)
6. x\_L = np.cos(w\_L \* n)
8. A = 1.0
9. m = 0.4
11. y = A \* (1.0 + m \* x\_L) \* x\_H
13. plt.plot(y, '.')
14. plt.stem(n, y)
15. 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

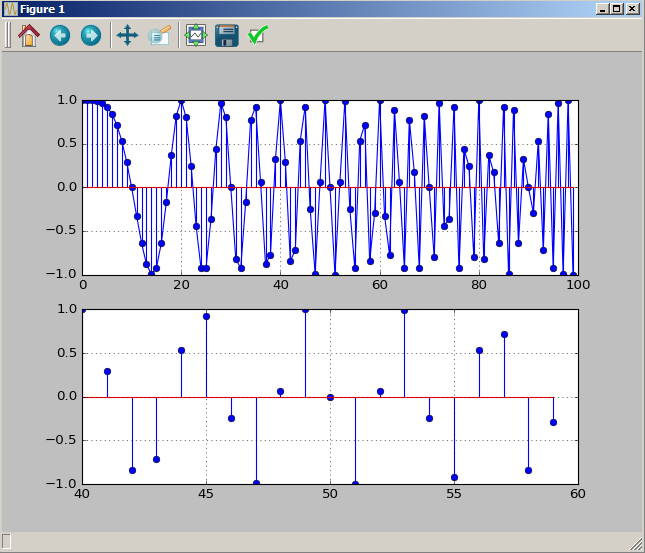
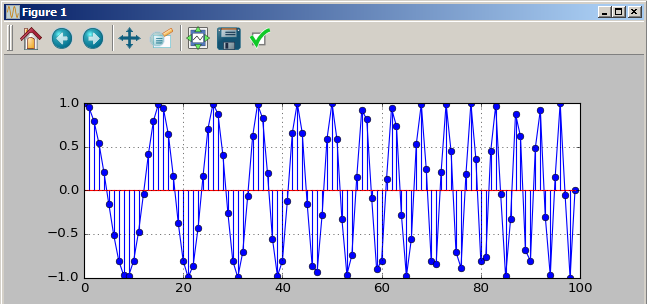


Рис. 3

1. n = np.arange(100)
2. a = np.pi / 2.0 / 100.0
3. y = np.cos(a \* n\*\*2)
5. fig, ax = plt.subplots(2, 1)
7. ax[0].plot(y, '-')
8. ax[0].stem(n, y)
9. ax[0].grid()
11. ax[1].stem(n[40:60], y[40:60])
12. ax[1].grid()
14. plt.show()

Постройте рисунок с минимальной частотой 0.1 и максимальной частотой 0.3.



# 4

Система, описываемая уравнением

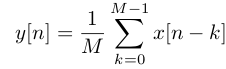


не является детерминированной.

Детерминированная система



В общем виде



(М 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.

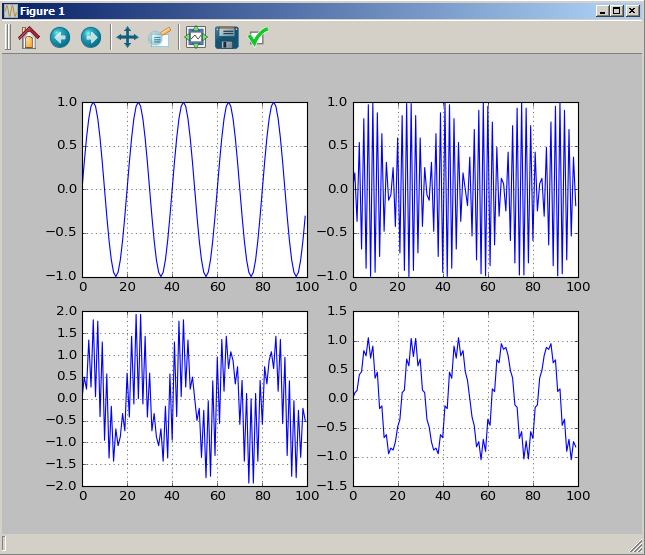
Уравнение y[n] можно реализовать так:

from scipy.signal import lfilter

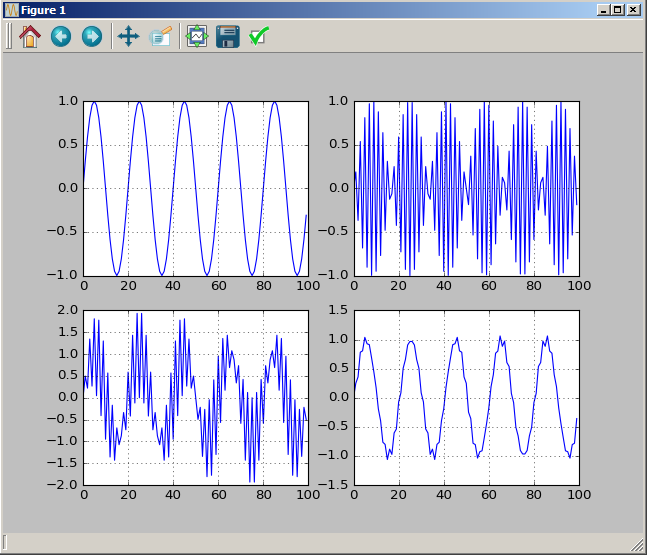
m=5

num=np.ones(m)

y=lfilter(num,1,x)/m

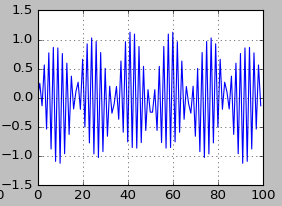


Постройте рисунок для M=2.



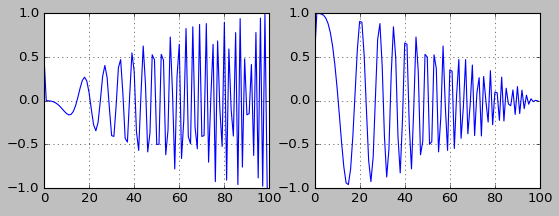
Какой сигнал подавляется?

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).

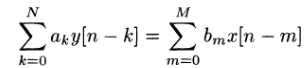


1. n = np.arange(100)
2. s1 = np.sin(2 \* np.pi \* 0.05 \* n)
3. s2 = np.sin(2 \* np.pi \* 0.47 \* n)
4. x = s1 + s2
6. m = 2
7. num = np.ones(m)
8. y = lfilter(num, 1, x) / m
10. y2 = np.zeros(100)
11. y2[0] = x[0] / 2.0
12. **for** n **in** np.arange(1,100):
13. y2[n] = (x[n]-x[n-1]) / 2.0
15. a = np.pi / 2 / 100
16. M = 2
17. n2 = np.arange(100)
18. x2 = np.cos(a \* n2\*\*2)
20. y3 = lfilter(np.ones(M), 1, x2) / M
22. y4 = np.zeros(100)
23. y4[0] = x2[0]/2
24. **for** n **in** np.arange(1,100):
25. y4[n] = (x2[n]-x2[n-1]) / 2


29. fig, ax = plt.subplots(2, 2)
31. ax[0][0].plot(s1)
32. ax[0][0].grid()
34. ax[0][1].plot(s2)
35. ax[0][1].grid()
37. ax[1][1].plot(y3)
38. ax[1][1].grid()
40. ax[1][0].plot(y4)
41. ax[1][0].grid()
43. plt.show()

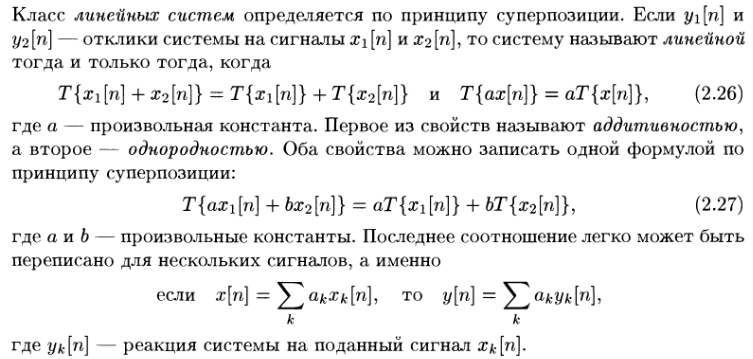
# 5

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



Рассмотрим систему

Проверим свойство линейности этой системы. Согласно [Оппенгейм, стр. 39]: ( lfilter ( coeff\_x, coeff\_y , sequence\_x) )



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]).

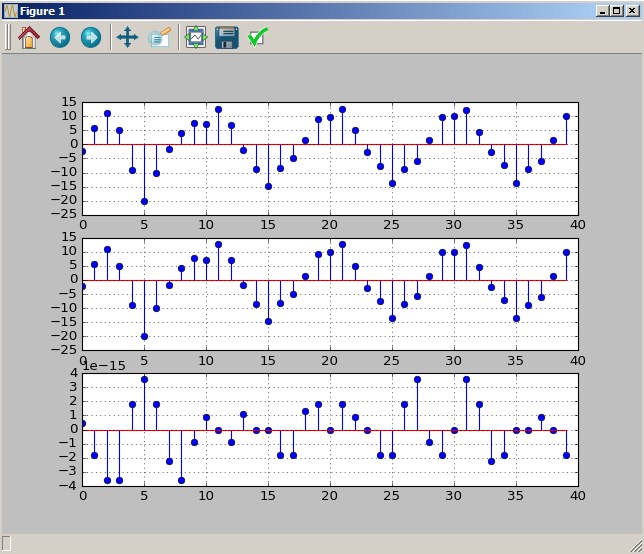


Рис. 5.

1. n = np.arange(40)
2. x1 = np.cos(2 \* np.pi \* 0.1 \* n)
3. x2 = np.cos(2 \* np.pi \* 0.4 \* n)
5. a =  2
6. b = -3
7. x = a \* x1 + b \* x2
9. coeff\_x = [2.2403, 2.4908, 2.2403]
10. coeff\_y = [1, -0.4, 0.75]
12. y1 = lfilter(coeff\_x, coeff\_y, x1)
13. y2 = lfilter(coeff\_x, coeff\_y, x2)
14. y  = lfilter(coeff\_x, coeff\_y, x)
16. diff = y - (a \* y1 + b \* y2)
18. fig, ax = plt.subplots(3, 1)
20. ax[0].stem(y)
21. ax[0].grid()
23. ax[1].stem(a \* y1 + b \* y2)
24. ax[1].grid()
26. ax[2].stem(diff)
27. ax[2].grid()
29. plt.show()

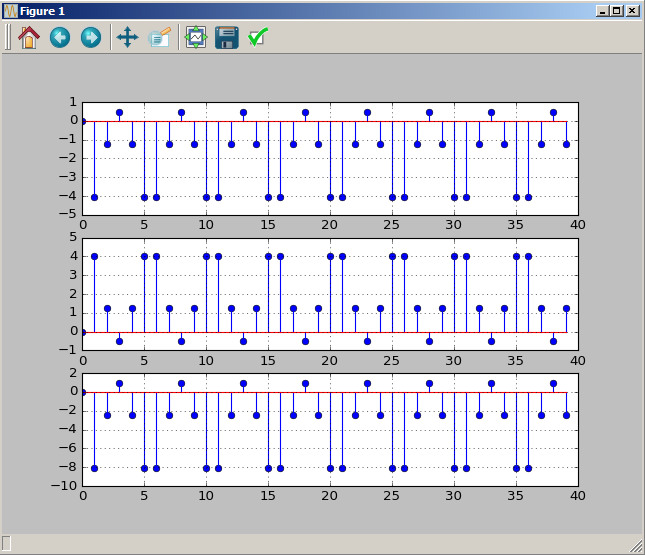
Система линейная? – Система не линейная.

Проверьте для различных частот и значений а, b.

Проверьте аналогичным способом линейность системы



Постройте графики.



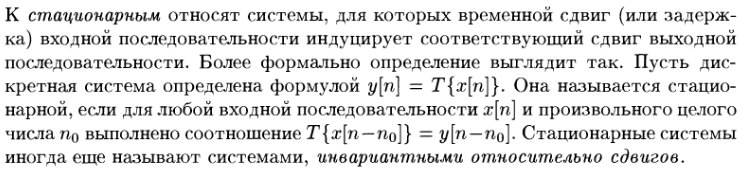
Система не линейная.

1. n = np.arange(40)
2. x1 = np.cos(2 \* np.pi \* 0.1 \* n)
3. x2 = np.cos(2 \* np.pi \* 0.4 \* n)
5. a =  2
6. b = -3
7. x = a \* x1 + b \* x2
9. y1 = np.zeros(40)
10. y2 = np.zeros(40)
11. y  = np.zeros(40)
13. **for** k **in** np.arange(1, 40):
14. y1[k] = x1[k] \* x1[k-1]
15. y2[k] = x2[k] \* x2[k-1]
16. y[k]  = x[k] \* x[k-1]
18. diff = y - (a \* y1 + b \* y2)
20. fig, ax = plt.subplots(3, 1)
22. ax[0].stem(y)
23. ax[0].grid()
25. ax[1].stem(a \* y1 + b \* y2)
26. ax[1].grid()
28. ax[2].stem(diff)
29. ax[2].grid()
31. plt.show()

# 6

Проверим, что система

стационарна. Согласно [Оппенгейм, стр. 40]

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.

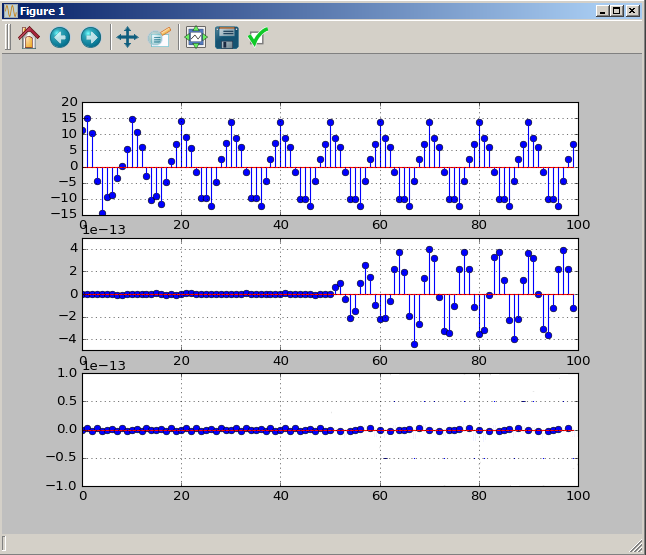
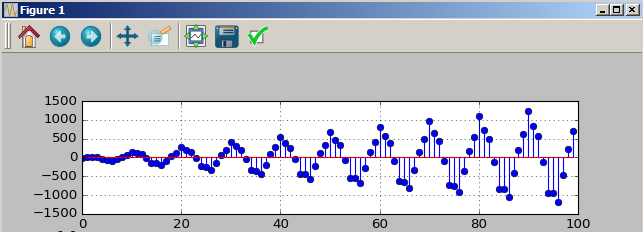


Рис. 6.

Система стационарна.



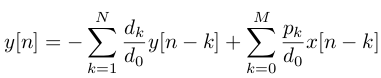


Система не является стационарной.

1. n = np.arange(100)
2. x1 = np.cos(2 \* np.pi \* 0.1 \* n)
3. x2 = np.cos(2 \* np.pi \* 0.4 \* n)
5. x1\_ = np.cos(2 \* np.pi \* 0.1 \* (n-10))
6. x2\_ = np.cos(2 \* np.pi \* 0.4 \* (n-10))
8. a = 2
9. b = 3
10. x  = a \* x1  + b \* x2
11. x\_ = a \* x1\_ + b \* x2\_
13. coeff\_x = [2.2403, 2.4908, 2.2403]
14. coeff\_y = [1, -0.4, 0.75]
16. y = lfilter(coeff\_x, coeff\_y, x)
17. yd = lfilter(coeff\_x, coeff\_y, x\_)
19. diff\_y = y - yd
20. diff\_x = x - x\_
22. x\_n1\_1 = np.cos(2 \* np.pi \* 0.1 \* (n-1))
23. x\_n1\_2 = np.cos(2 \* np.pi \* 0.4 \* (n-1))
24. x\_n1   = a \* x\_n1\_1 + b \* x\_n1\_2
25. x\_n    = n \* x
27. y\_n = lfilter(coeff\_x, coeff\_y, x\_n + x\_n1)
29. fig, ax = plt.subplots(3, 1)
31. ax[0].stem(y\_n)
32. ax[0].grid()
34. ax[1].stem(diff\_y)
35. ax[1].grid()
37. ax[2].stem(diff\_x)
38. ax[2].grid()
40. plt.show()

# 7

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



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,'-.')

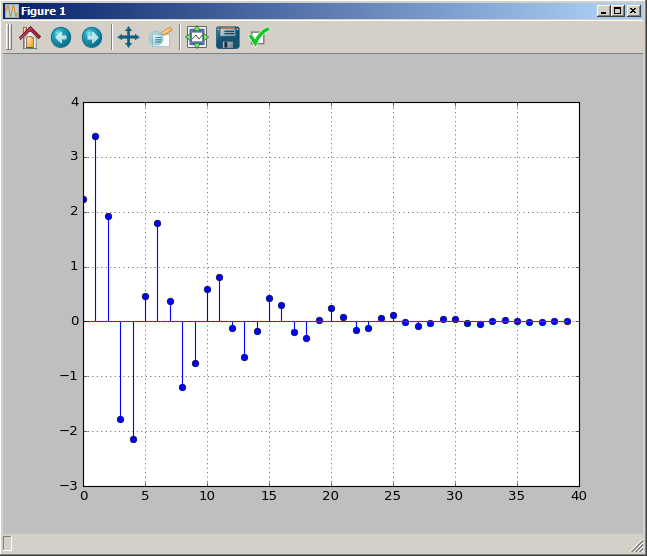
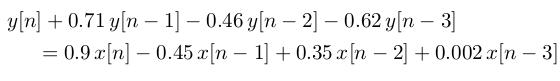
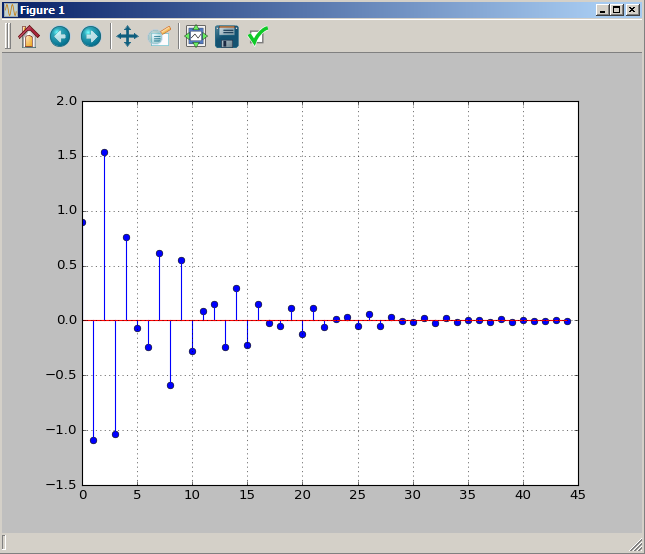


Рис. 7.

Постройте график первых 45 отсчетов импульсной характеристики системы:

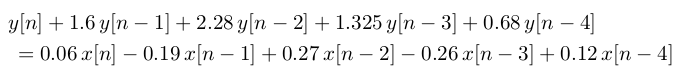




1. N = 40
2. x = np.zeros(N)
3. x[0] = 1
4. coeff\_x = [2.2403, 2.4908, 2.2403]
5. coeff\_y = [1, -0.4, 0.75]
7. h = lfilter(coeff\_x, coeff\_y, x)
8. nn = np.arange(N)
10. N2 = 45
11. coeff\_x1 = [0.9, -0.45, 0.35, 0.002]
12. coeff\_y1 = [1,   0.71, -0.46, -0.62]
13. x = np.zeros(N2)
14. x[0] = 1
15. h = lfilter(coeff\_x1, coeff\_y1, x)
16. nn = np.arange(N2)
18. plt.stem(nn, h)
19. plt.grid()
21. plt.show()

# 8

Показать, что выход системы 4-го порядка

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

первая система в каскаде

вторая система в каскаде

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

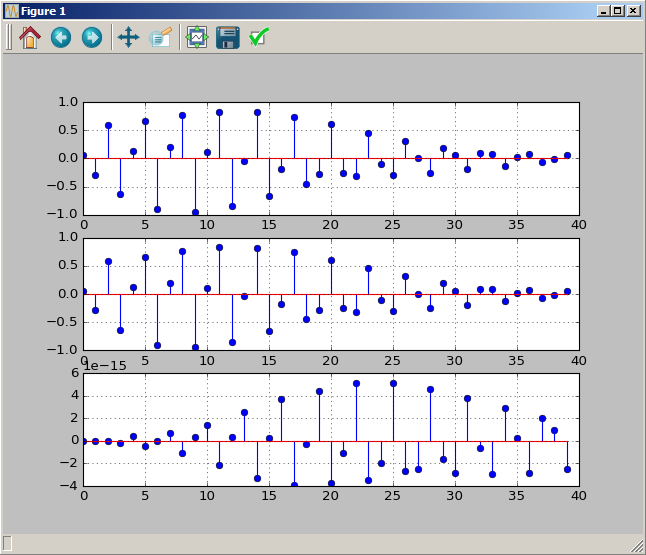


Рис. 8.

1. N = 40
2. x = np.zeros(N)
3. x[0] = 1
5. coeff\_x\_4 = [0.06, -0.19, 0.27, -0.26, 0.12]
6. coeff\_y\_4 = [1, 1.6, 2.28, 1.325, 0.68]
8. coeff\_x1\_2 = [0.3, -0.2, 0.4]
9. coeff\_y1\_2 = [1, 0.9, 0.8]
11. coeff\_x2\_2 = [0.2, -0.5, 0.3]
12. coeff\_y2\_2 = [1, 0.7, 0.85]
14. h\_4 = lfilter(coeff\_x\_4, coeff\_y\_4, x)
15. nn\_4 = np.arange(N)
17. h1\_2 = lfilter(coeff\_x1\_2, coeff\_y1\_2, x)
18. nn1\_2 = np.arange(N)
20. h2\_2 = lfilter(coeff\_x2\_2, coeff\_y2\_2, h1\_2)
21. nn2\_2 = np.arange(N)
23. diff = h\_4 - h2\_2
25. fig, ax = plt.subplots(3, 1)
27. ax[0].stem(nn\_4, h\_4)
28. ax[0].grid()
30. ax[1].stem(nn1\_2, h2\_2)
31. ax[1].grid()
33. ax[2].stem(diff)
34. ax[2].grid()
36. 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.

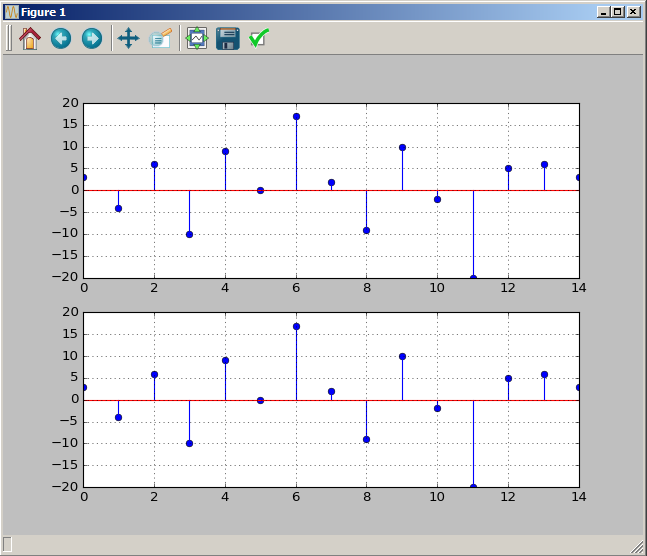


Рис. 9.

1. h = [3, 2, 1, -2, 1, 0, -4, 0, 3]
2. x = [1, -2, 3, -4, 3, 2, 1]
4. y\_conv = convolve(x, h)
5. y\_filter = lfilter(h, 1, np.concatenate((x,np.zeros(8)), axis=0))
7. fig, ax = plt.subplots(2, 1)
9. ax[0].stem(y\_conv)
10. ax[0].grid()
12. ax[1].stem(y\_filter)
13. ax[1].grid()
15. plt.show()

# 10

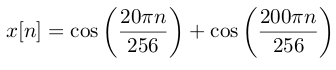
Даны две системы. Первая



Вторая



Найти отклик этих систем на входное воздействие:



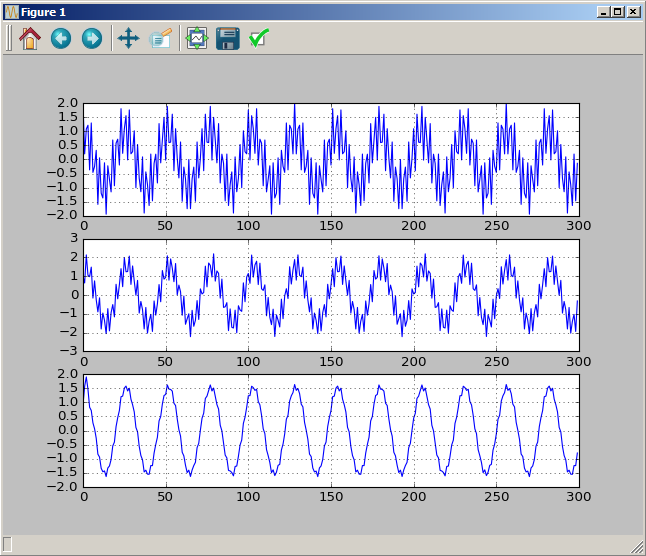


Рис. 10.

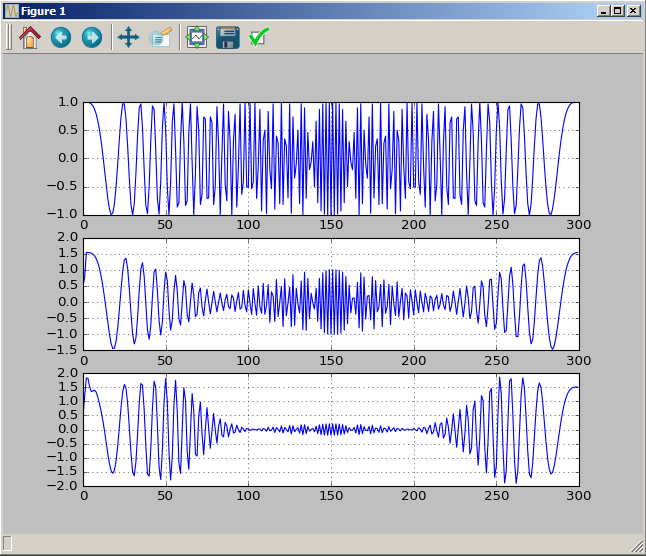
Обе системы — фильтры низких частот.

Какая система лучше подавляет высокие частоты? – Вторая система.

1. n = np.arange(300)
2. x1 = np.cos(20 \* np.pi \* n / 256)
3. x2 = np.cos(200 \* np.pi \* n / 256)
4. x = x1 + x2
6. coeff\_x1 = [0.5, 0.27, 0.77]
7. coeff\_y1 = [1]
9. coeff\_x2 = [0.45, 0.5, 0.45]
10. coeff\_y2 = [1, -0.53, 0.46]
12. y1 = lfilter(coeff\_x1, coeff\_y1, x)
13. y2 = lfilter(coeff\_x2, coeff\_y2, x)
15. fig, ax = plt.subplots(3, 1)
17. ax[0].plot(x)
18. ax[0].grid()
20. ax[1].plot(y1)
21. ax[1].grid()
23. ax[2].plot(y2)
24. ax[2].grid()
26. plt.show()

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

Постройте графики.



1. n = np.arange(300)
3. x = np.cos(2 \* np.pi \* n\*\*2 \* 0.5 / 300)
5. coeff\_x1 = [0.5, 0.27, 0.77]
6. coeff\_y1 = [1]
8. coeff\_x2 = [0.45, 0.5, 0.45]
9. coeff\_y2 = [1, -0.53, 0.46]
11. y1 = lfilter(coeff\_x1, coeff\_y1, x)
12. y2 = lfilter(coeff\_x2, coeff\_y2, x)
14. fig, ax = plt.subplots(3, 1)
16. ax[0].plot(x)
17. ax[0].grid()
19. ax[1].plot(y1)
20. ax[1].grid()
22. ax[2].plot(y2)
23. ax[2].grid()
25. plt.show()