ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. А.Н. Тихонова

Департамент электронной инженерии

Курс: Основы построения

инфокоммуникационных систем и сетей

Практическая работа №1

«Непрерывные и дискретные сигналы.

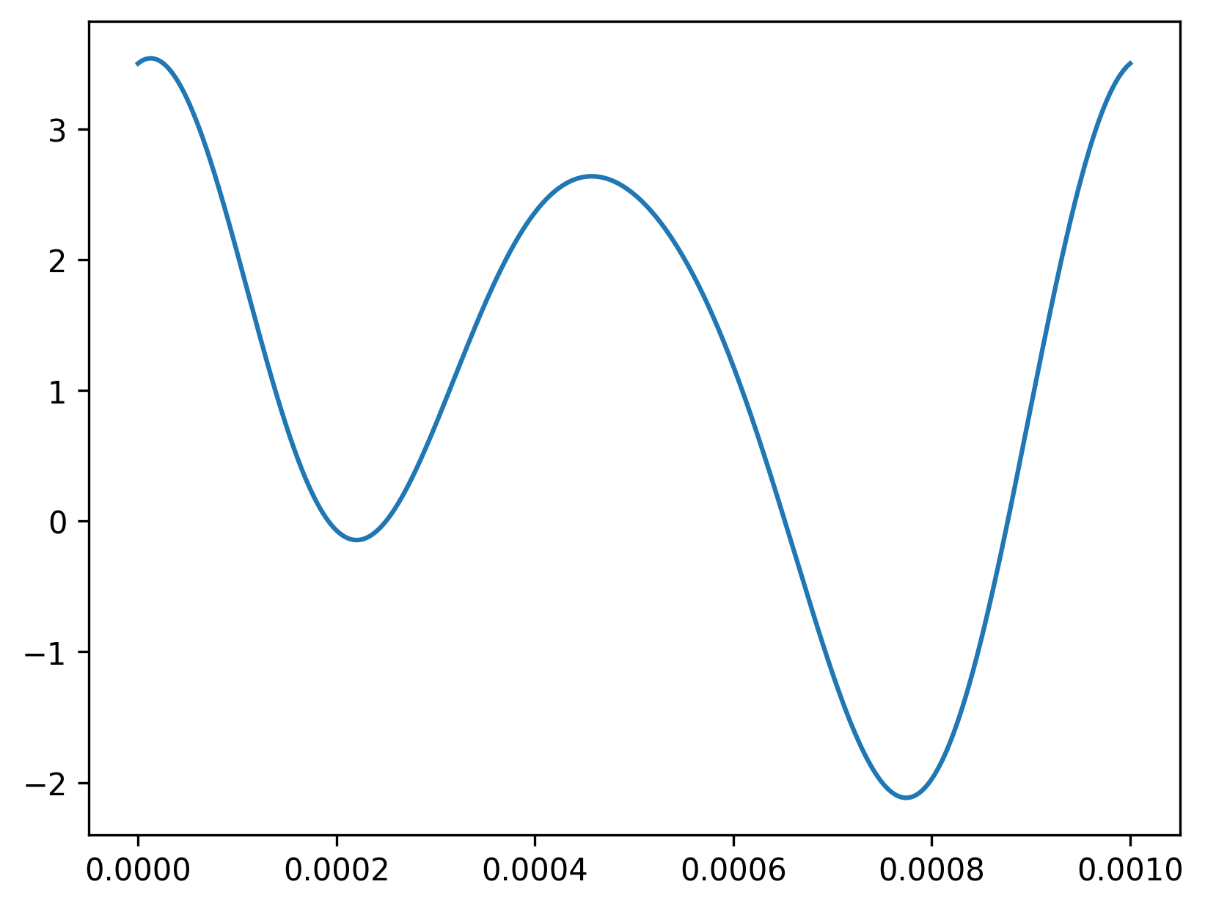
Графическое представление»

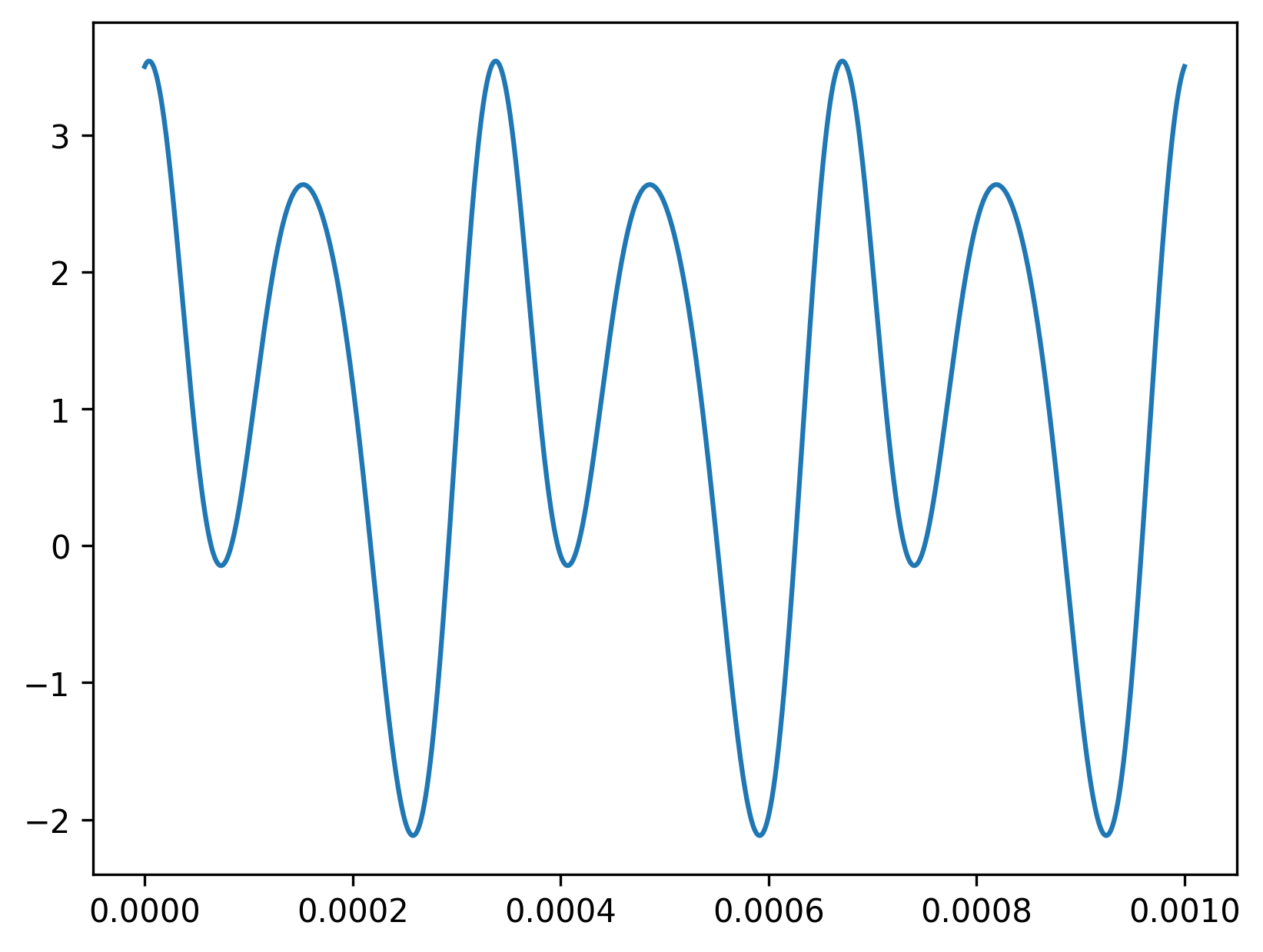
Ефремов Виктор Васильевич

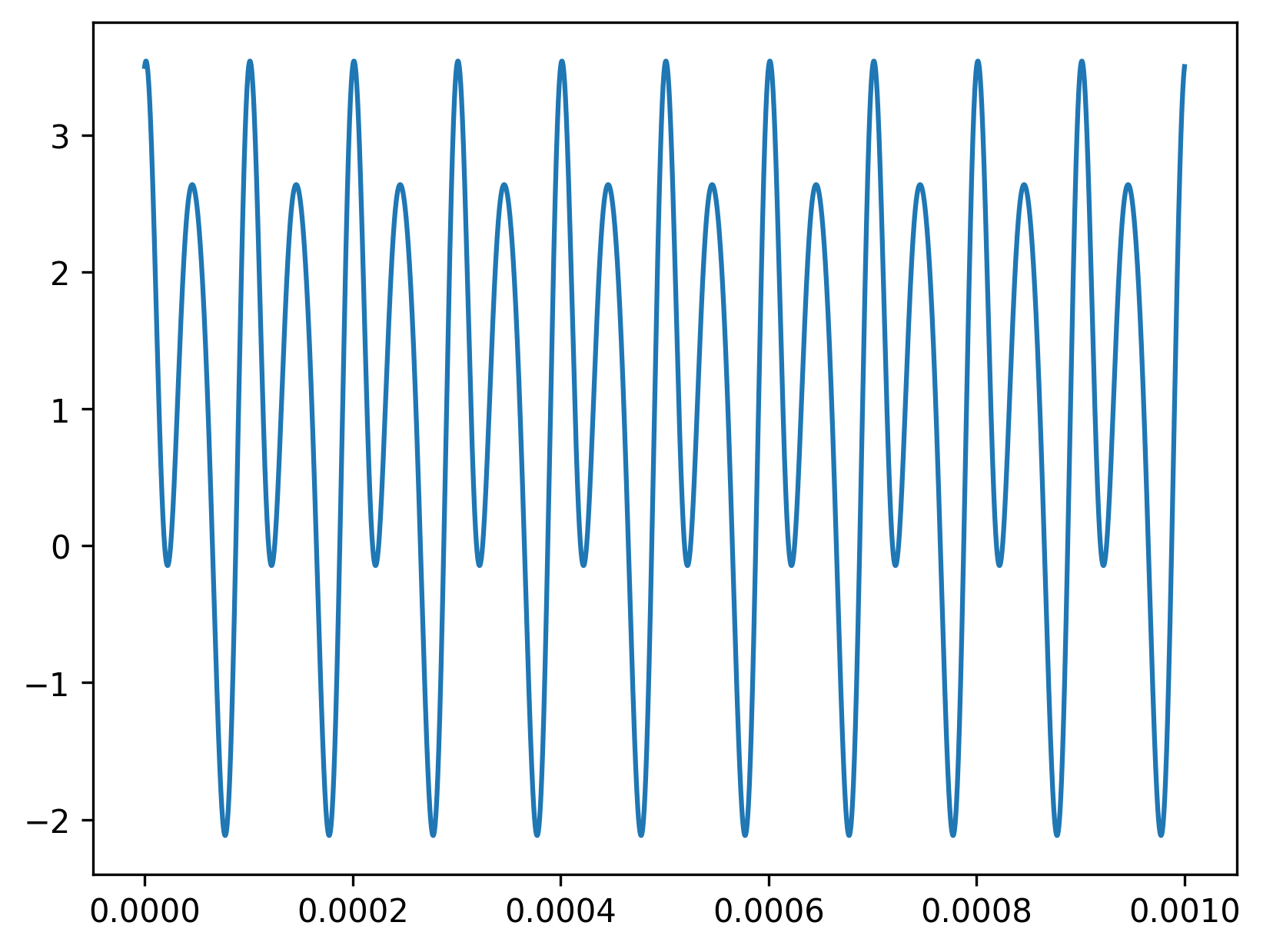
БИТ-203

Москва

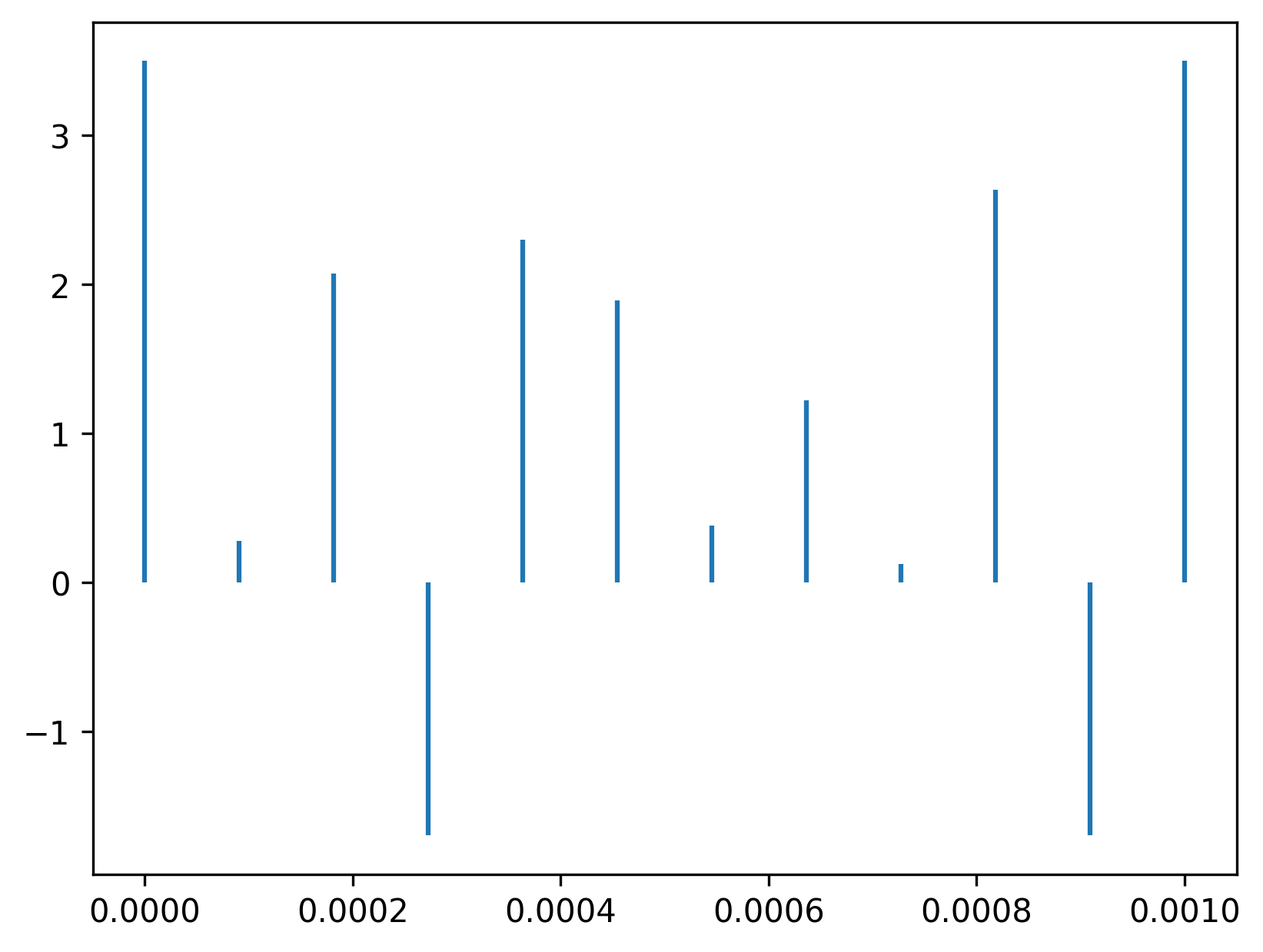
2022

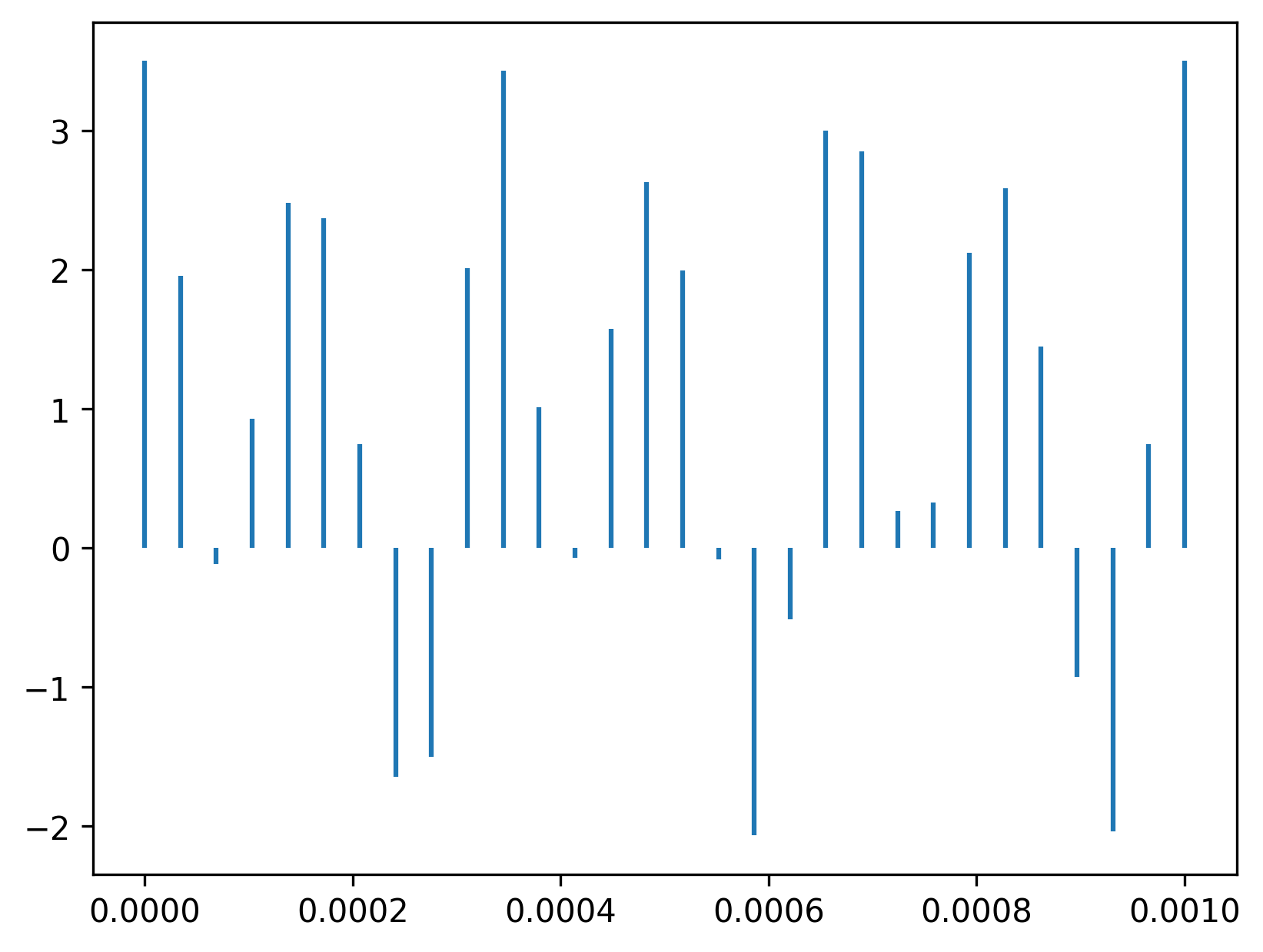
Я код переписал заново. Просто потому что. Весь код в конце.  
Построим графики для частот в 1кГц, 3кГц, 10кГц от 0 до 10мс.  
Они отличаются только количеством периодов.

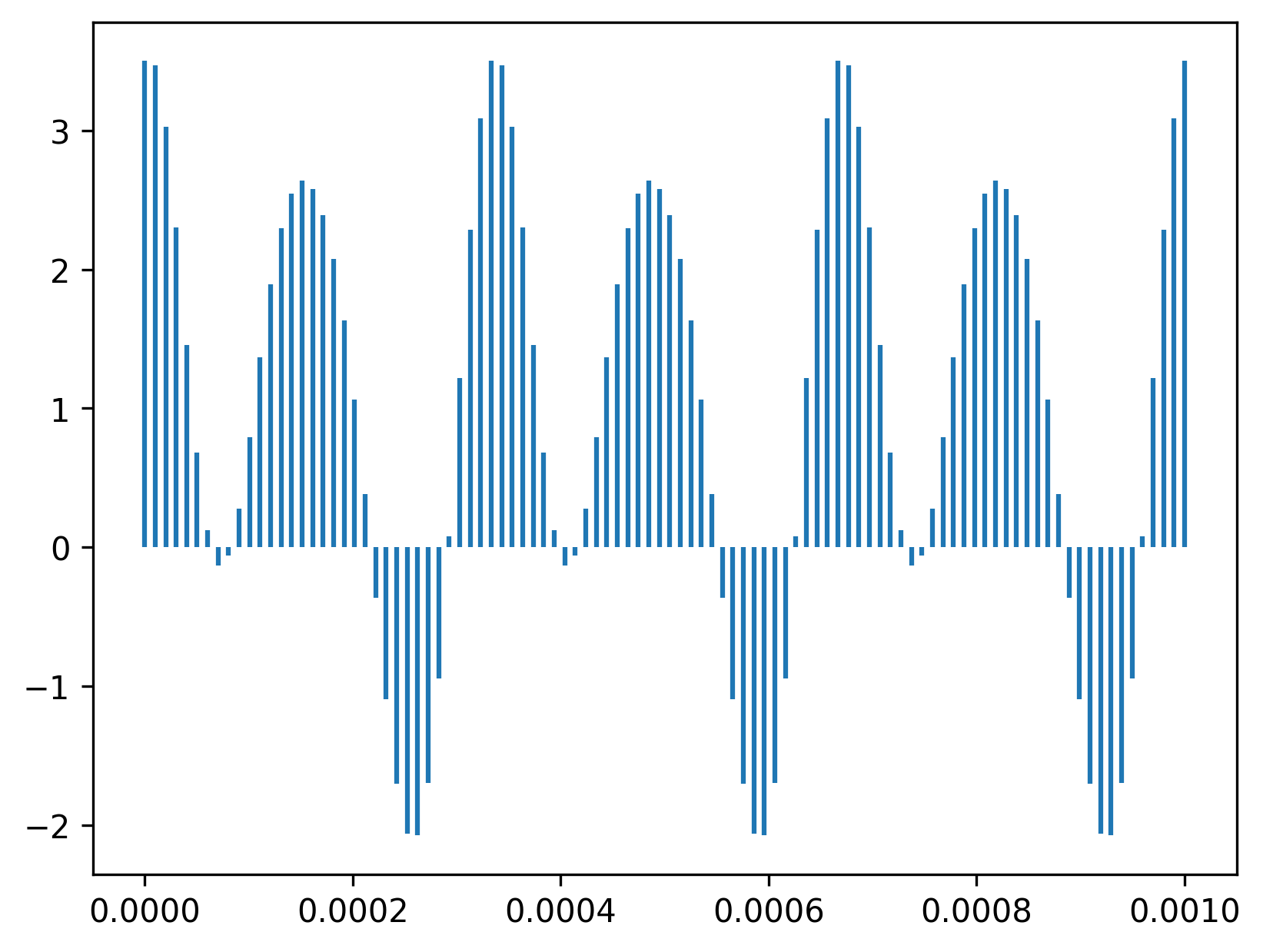




Построим для частоты в 3кГц дискретные по времени графики с 12, 30 и 100 отсчетами.

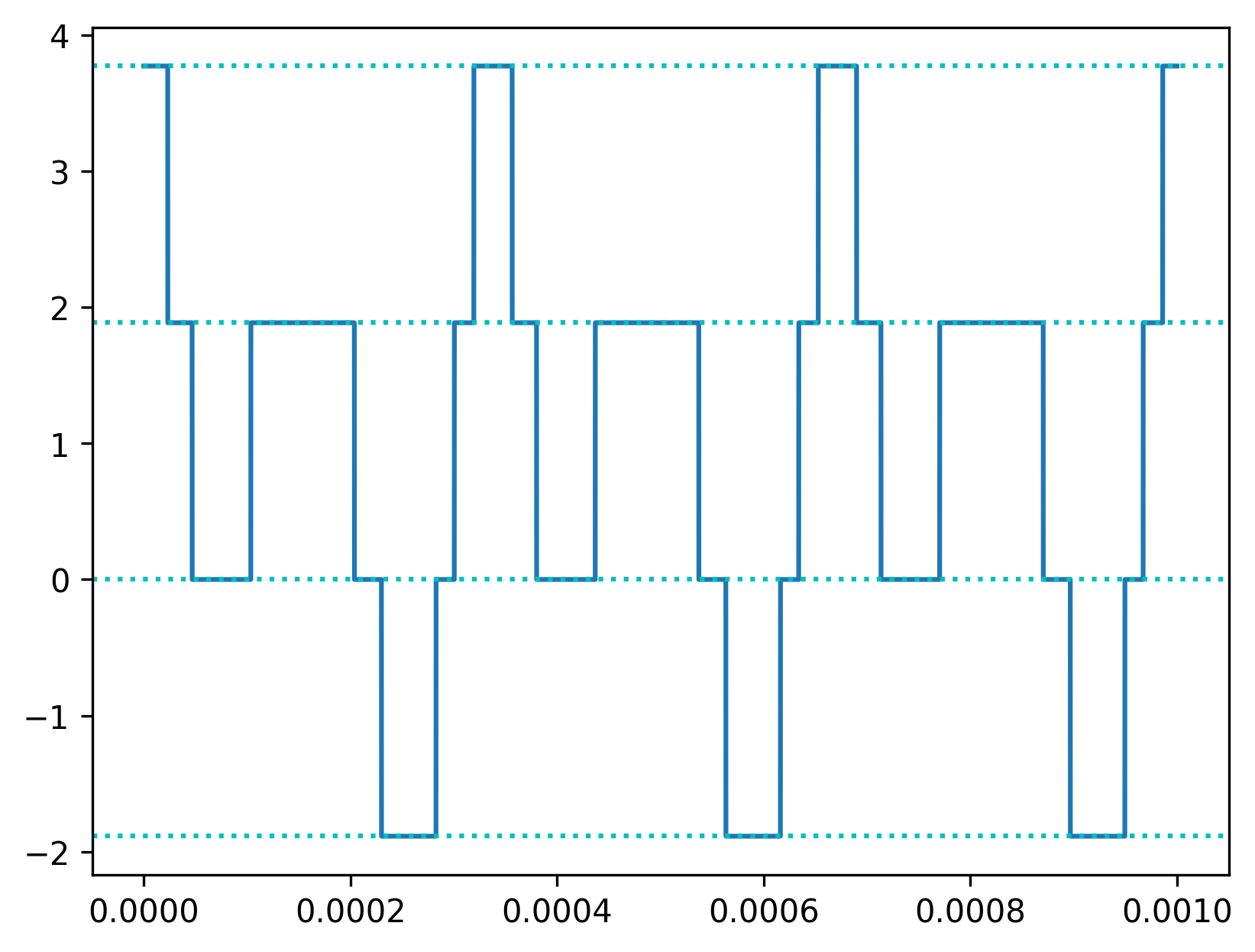


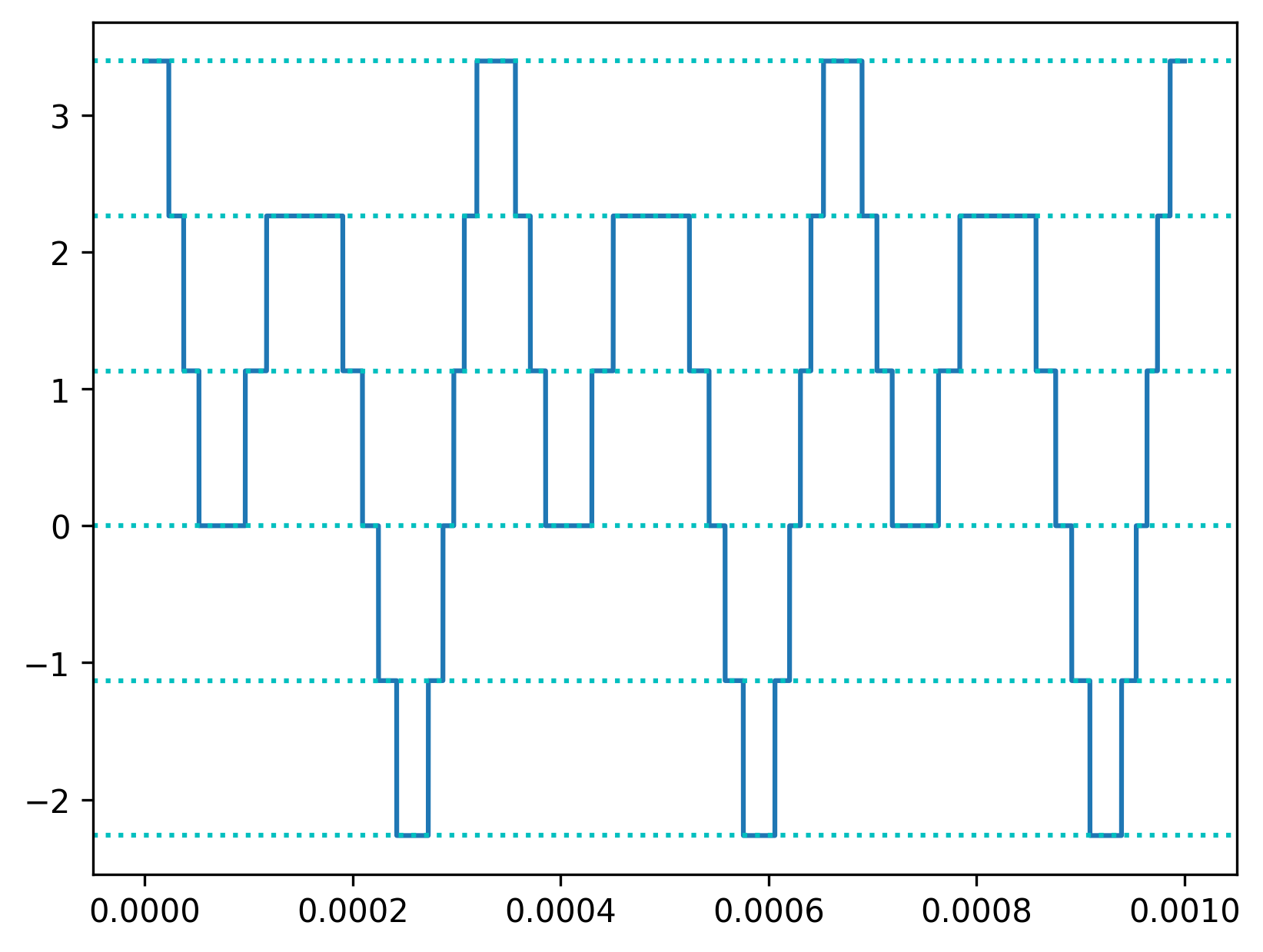




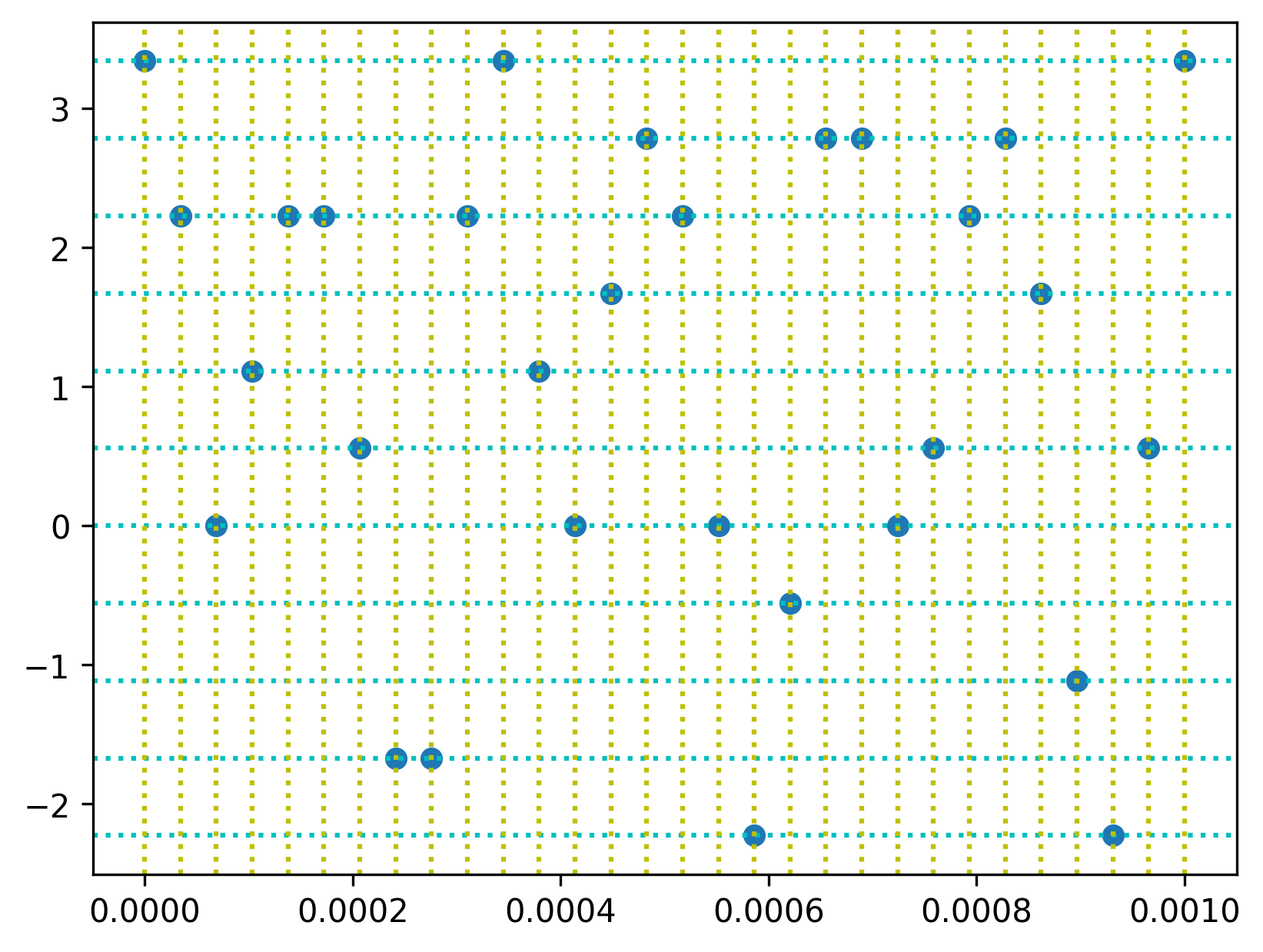
При 12 отсчетах исходный сигнал еле угадывается. При 30 лучше, но восстановить все еще затруднительно. При 100 заметны и мелкие детали.

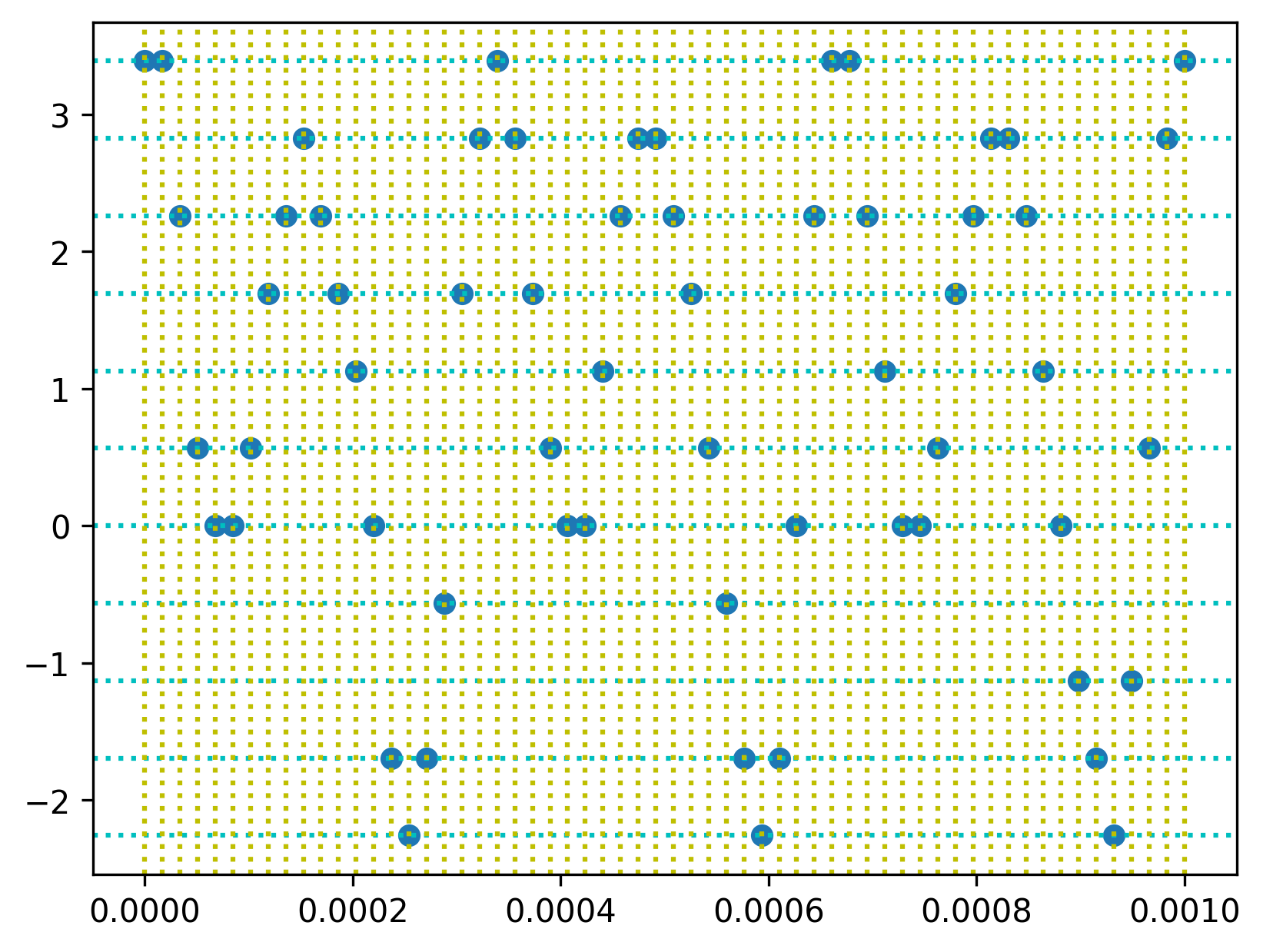
Для тех же 3кГц построим дискретные по амплитуде графики. С 4 и 6 уровнями сигнала.

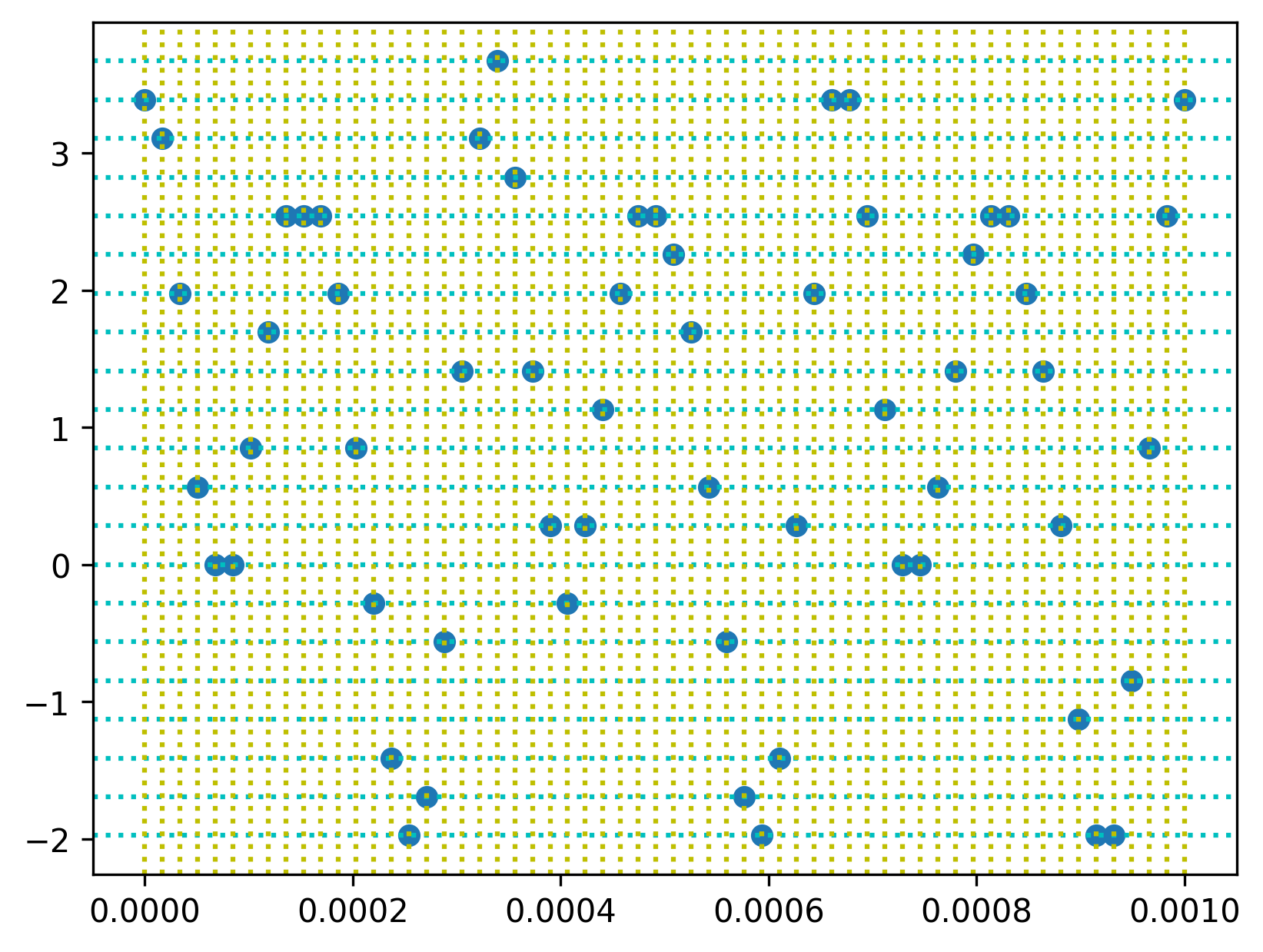


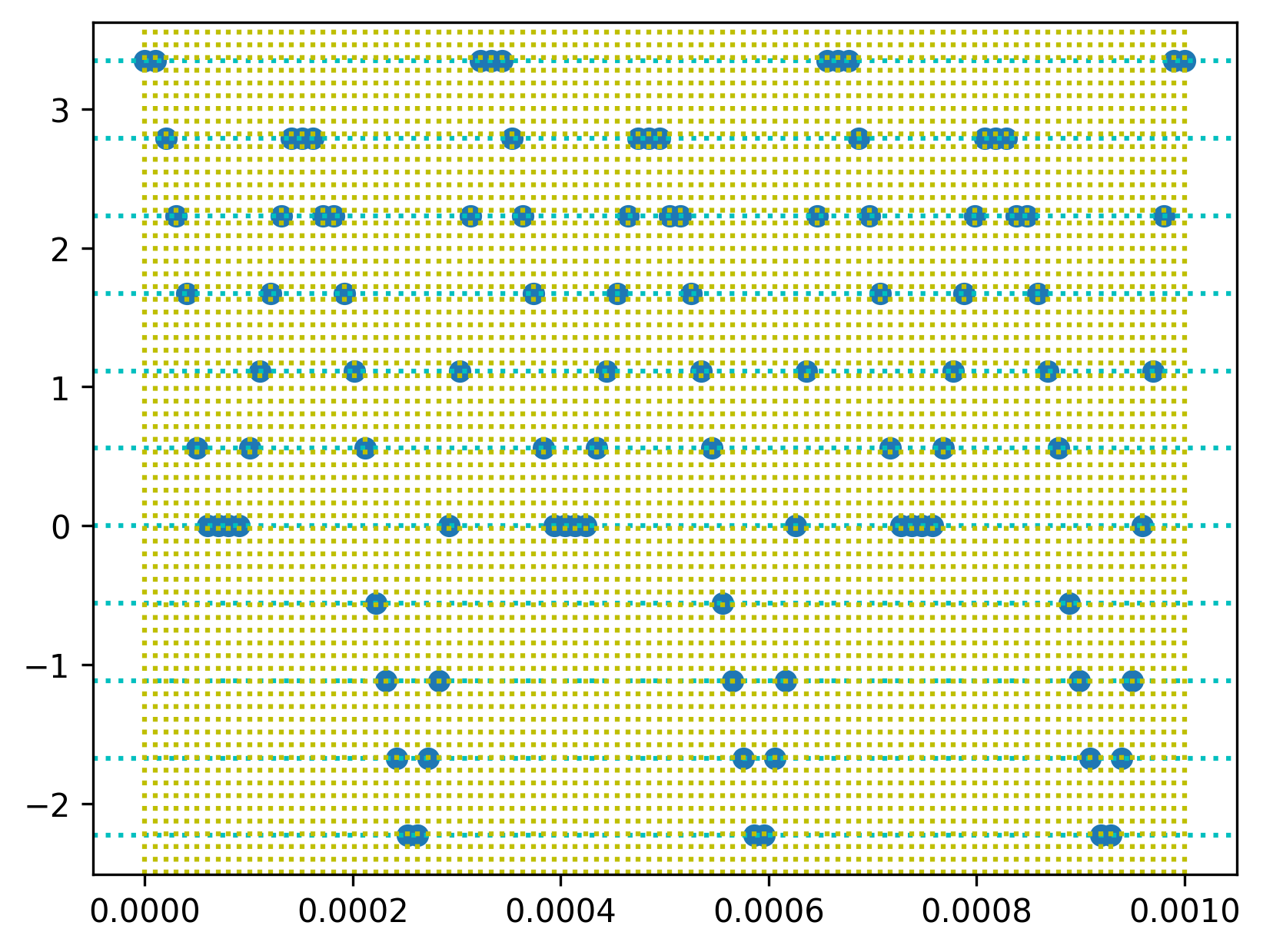


Построим так же соответствующий цифровой сигнал. При 30 уровнях по времени и 10 по амплитуде. 60 и 10, 60 и 20, 100 и 10.









Интересно, что при 30 отсчетах по времени исходный сигнал совершенно не виден, хотя при дискретном времени и непрерывной амплитуде он угадывается.

Так же в случае 60 отсчетов по времени разница между 10 и 20 уровнями амплитуды не заметна.

Код также лежит по адресу

[https://github.com/Wicirelllis/miem-docs/tree/master/ИТСС/7-8 ОПИСиС/prac\_1](https://github.com/Wicirelllis/miem-docs/tree/master/ИТСС/7-8%20ОПИСиС/prac_1)

import numpy as np  
import matplotlib.pyplot as plt  
  
SAMPLE\_MAX = 10\_000  
plt.figure(dpi=300)  
  
  
def signal(x):  
 return 1.0 + np.sin(2.0 \* np.pi \* x) + 2.0 \* np.cos(4.0 \* np.pi \* x) +\  
 0.5 \* np.cos(6.0 \* np.pi \* x)  
  
  
def cnt\_time\_cnt\_value\_plt(freq, time):  
 plt.clf()  
 x = np.linspace(0, time, SAMPLE\_MAX)  
 y = signal(freq \* x)  
 plt.plot(x, y)  
 plt.savefig(f"img/cnt\_cnt\_{freq}\_{time}\_x\_x.png", bbox\_inches='tight')  
  
  
def dcr\_time\_cnt\_value\_plt(freq, time, time\_samples):  
 plt.clf()  
 x = np.linspace(0, time, time\_samples)  
 y = signal(freq \* x)  
 for i, j in zip(x, y):  
 plt.vlines(i, ymin=0, ymax=j)  
 plt.savefig(f"img/dcr\_cnt\_{freq}\_{time}\_{time\_samples}\_x.png", bbox\_inches='tight')  
  
  
def cnt\_time\_dcr\_value\_plt(freq, time, value\_samples):  
 plt.clf()  
 x = np.linspace(0, time, SAMPLE\_MAX)  
 y = signal(freq \* x)  
 step = (np.max(y) - np.min(y)) / value\_samples  
 y\_dcr = np.around(y / step) \* step  
 plt.plot(x, y\_dcr)  
 # horizontal grid  
 upper = np.around(np.max(y) / step).astype(int)  
 lower = np.around(np.min(y) / step).astype(int)  
 for i in range(lower, upper + 1):  
 plt.axhline(i \* step, c='c', ls=':')  
 plt.savefig(f"img/cnt\_dcr\_{freq}\_{time}\_x\_{value\_samples}.png", bbox\_inches='tight')  
  
  
def dcr\_time\_dcr\_value\_plt(freq, time, time\_samples, value\_samples):  
 plt.clf()  
 x = np.linspace(0, time, time\_samples)  
 y = signal(freq \* x)  
 step = (np.max(y) - np.min(y)) / value\_samples  
 y\_dcr = np.around(y / step) \* step  
 plt.scatter(x, y\_dcr)  
 # horizontal grid  
 upper = np.around(np.max(y) / step).astype(int)  
 lower = np.around(np.min(y) / step).astype(int)  
 for i in range(lower, upper + 1):  
 plt.axhline(i \* step, c='c', ls=':')  
 # vertical grid  
 for i in x:  
 plt.axvline(i, c='y', ls=':')  
 plt.savefig(f"img/dcr\_dcr\_{freq}\_{time}\_{time\_samples}\_{value\_samples}.png", bbox\_inches='tight')  
  
  
# s = input("Введите частоту, время, количество уровней по времени и сигналу\n").split()  
# freq = int(s[0])  
# time = float(s[1])  
# time\_samples = int(s[2])  
# value\_samples = int(s[3])  
  
# cnt\_time\_cnt\_value\_plt(freq, time)  
# dcr\_time\_cnt\_value\_plt(freq, time, time\_samples)  
# cnt\_time\_dcr\_value\_plt(freq, time, value\_samples)  
# dcr\_time\_dcr\_value\_plt(freq, time, time\_samples, value\_samples)  
  
cnt\_time\_cnt\_value\_plt(1000, 0.001)  
cnt\_time\_cnt\_value\_plt(3000, 0.001)  
cnt\_time\_cnt\_value\_plt(10000, 0.001)  
  
dcr\_time\_cnt\_value\_plt(3000, 0.001, 12)  
dcr\_time\_cnt\_value\_plt(3000, 0.001, 30)  
dcr\_time\_cnt\_value\_plt(3000, 0.001, 100)  
  
cnt\_time\_dcr\_value\_plt(3000, 0.001, 3)  
cnt\_time\_dcr\_value\_plt(3000, 0.001, 5)  
cnt\_time\_dcr\_value\_plt(3000, 0.001, 10)  
  
dcr\_time\_dcr\_value\_plt(3000, 0.001, 30, 10)  
dcr\_time\_dcr\_value\_plt(3000, 0.001, 60, 10)  
dcr\_time\_dcr\_value\_plt(3000, 0.001, 100, 10)  
dcr\_time\_dcr\_value\_plt(3000, 0.001, 60, 20)