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

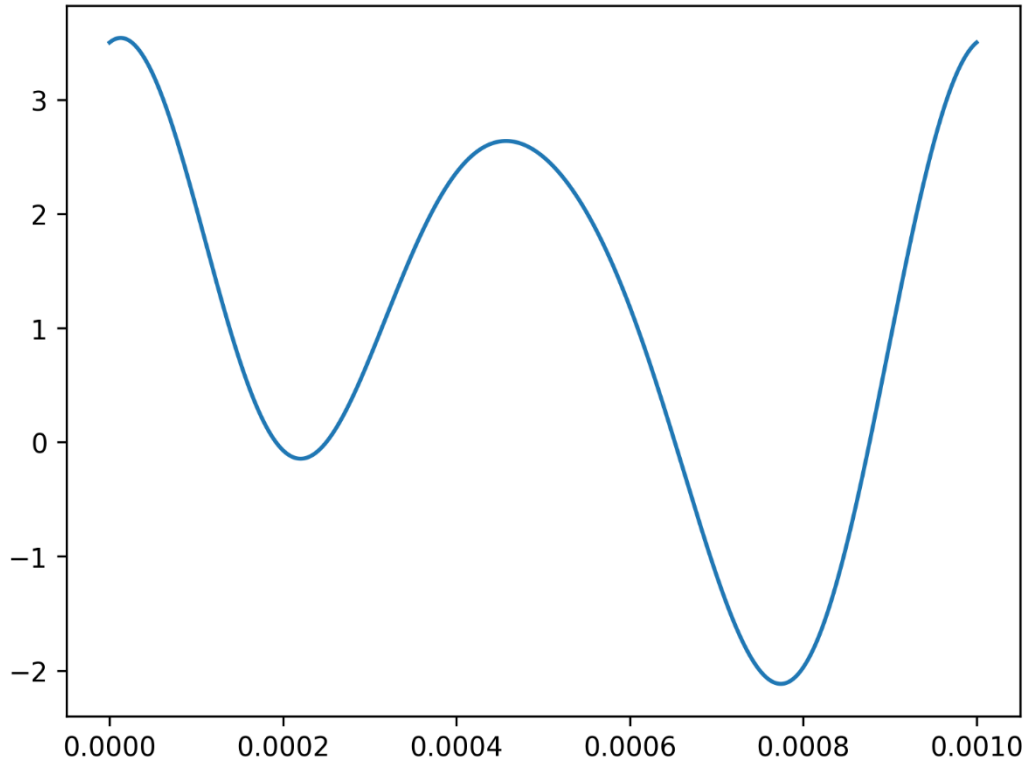
Практическая работа №1

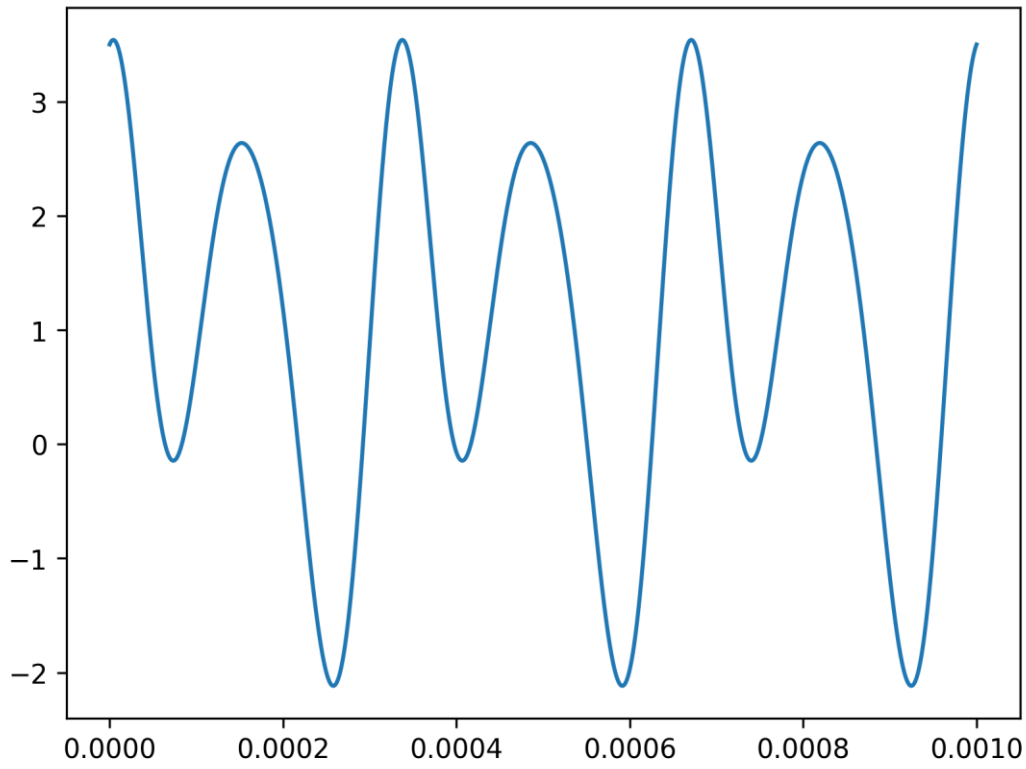
«Непрерывные и дискретные сигналы.
Графическое представление»

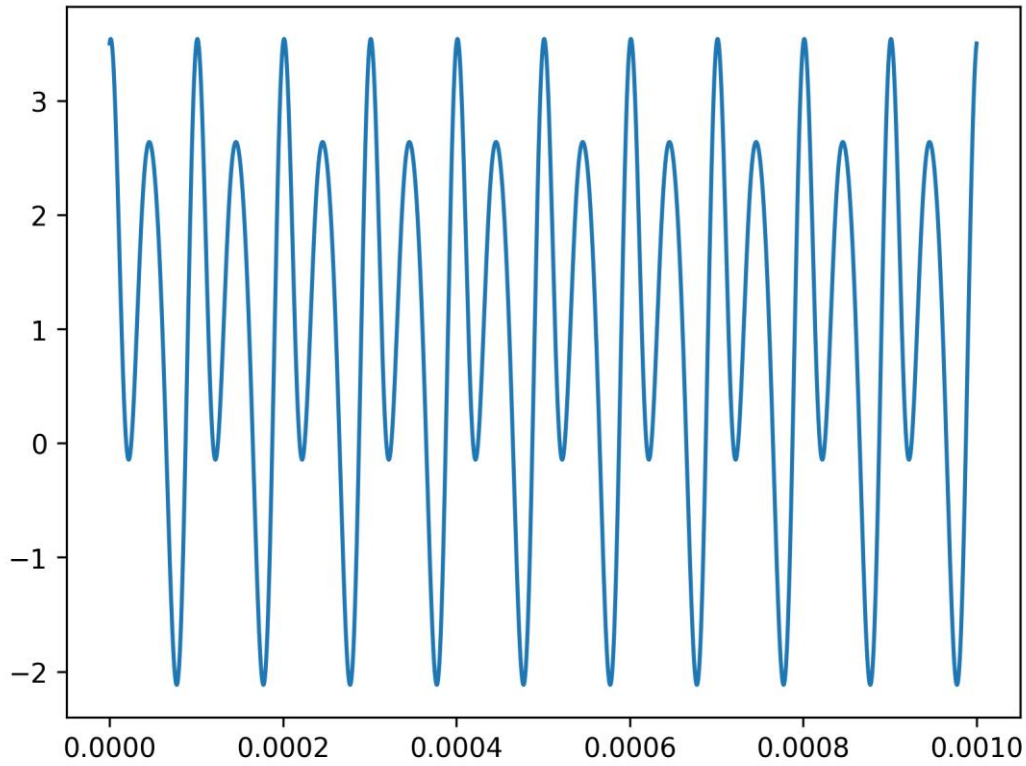
Ефремов Виктор Васильевич
БИТ-203

Москва
2022

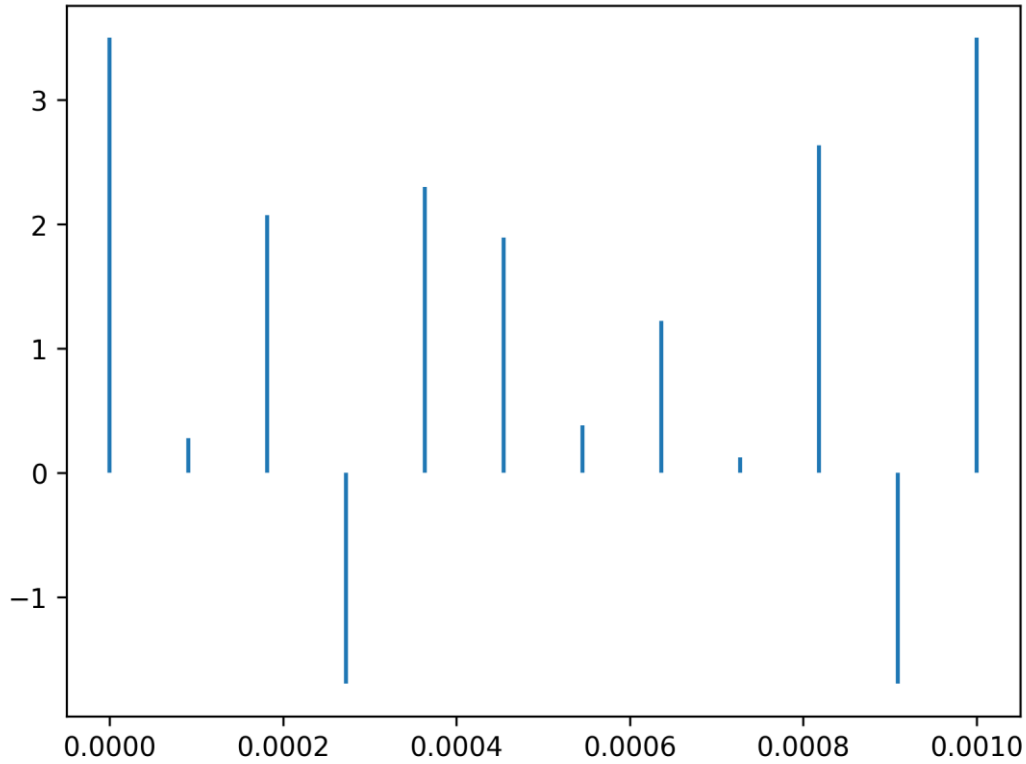
Я код переписал заново. Просто потому что. Весь код в конце.
Построим графики для частот в 1кГц, 3кГц, 10кГц от 0 до 10мс.
Они отличаются только количеством периодов.

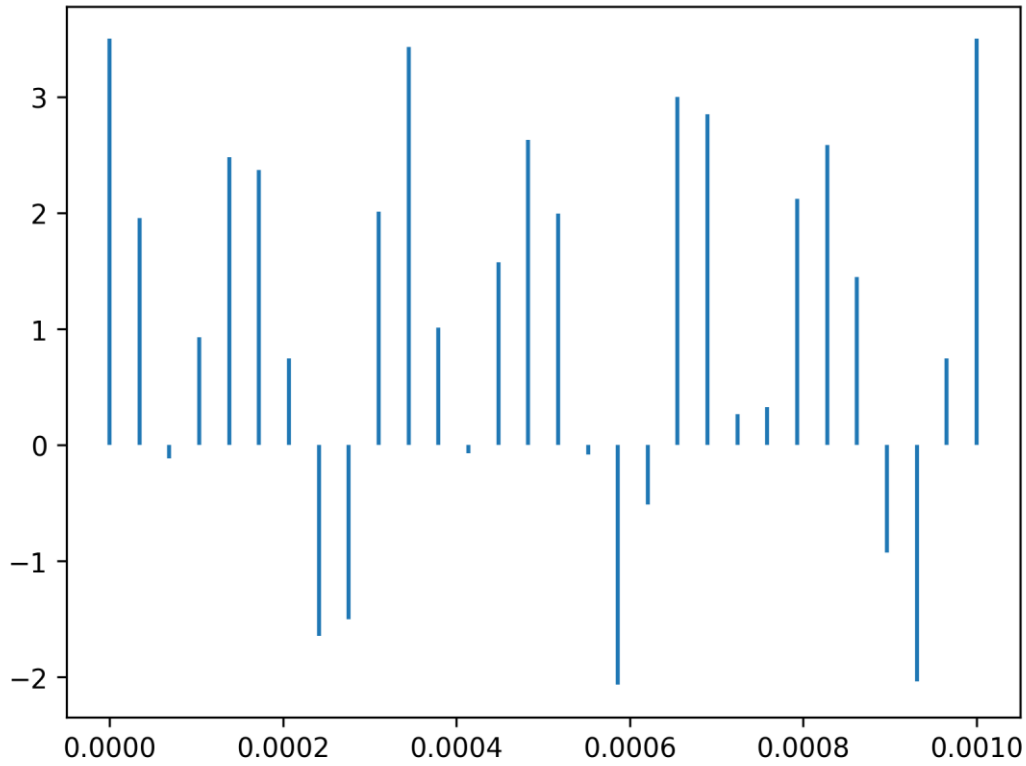


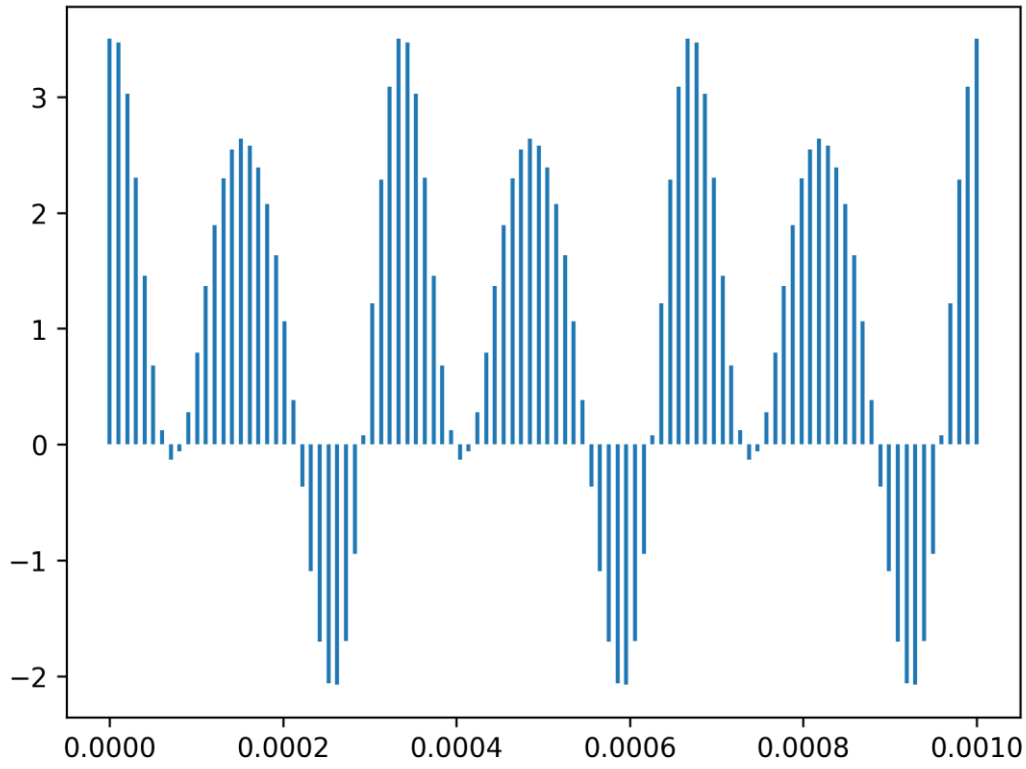




Построим для частоты в 3кГц дискретные по времени графики с 12, 30 и 100 отсчетами.

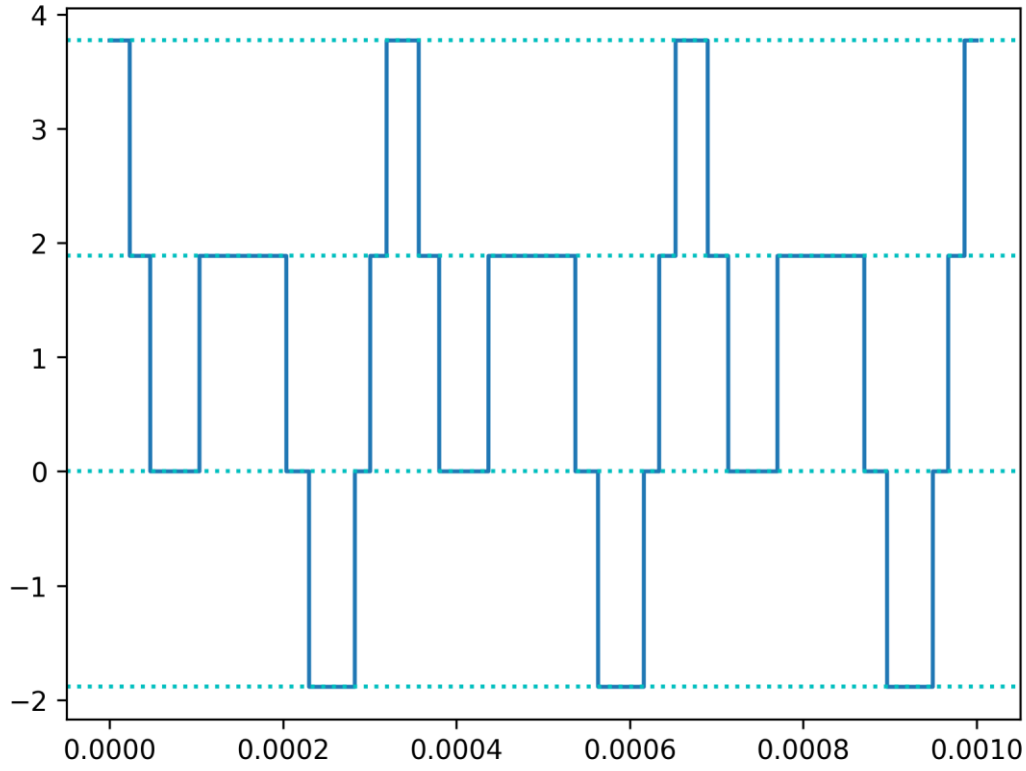


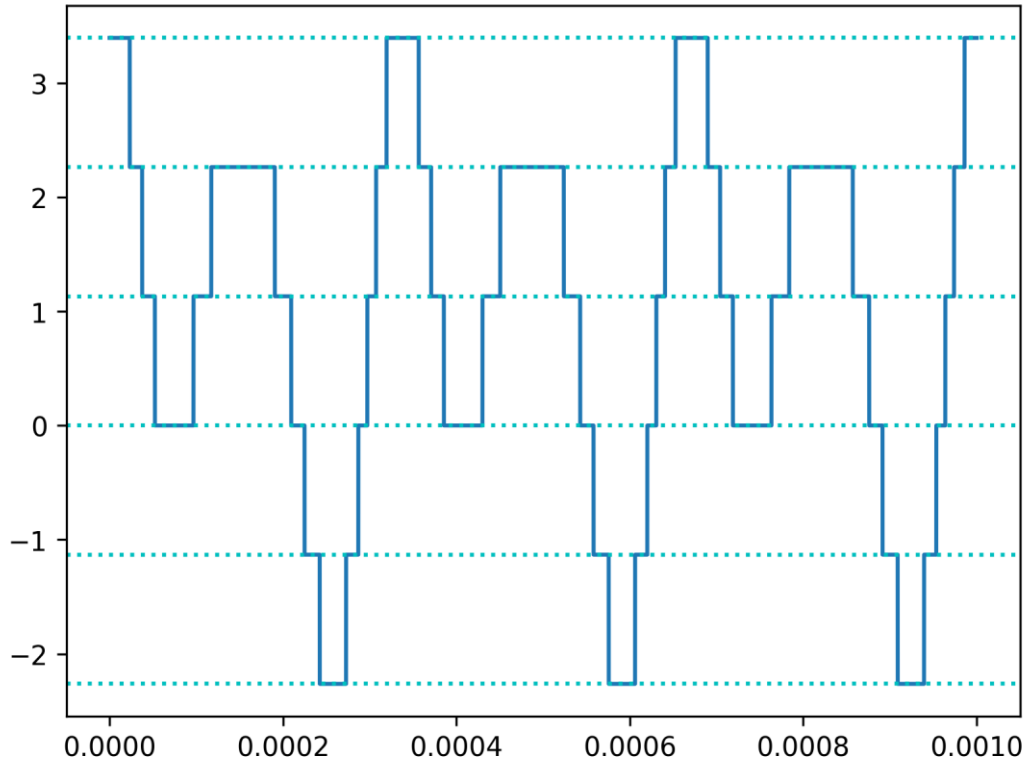




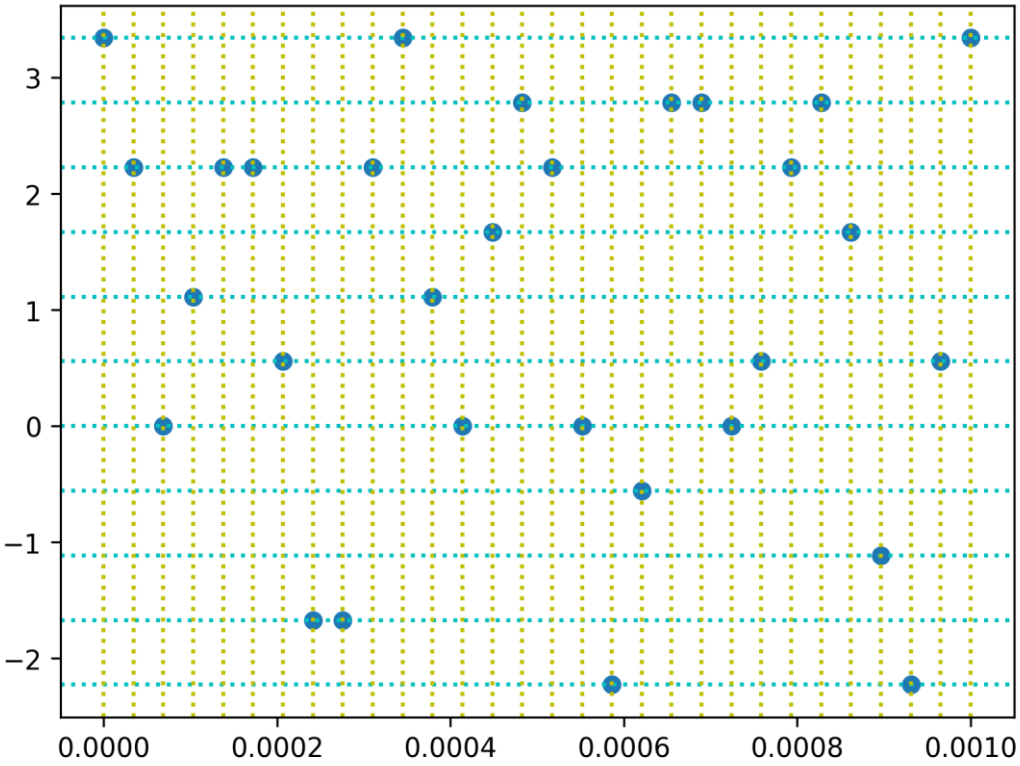
При 12 отсчетах исходный сигнал еле угадывается. При 30 лучше, но восстановить все еще затруднительно. При 100 заметны и мелкие детали.

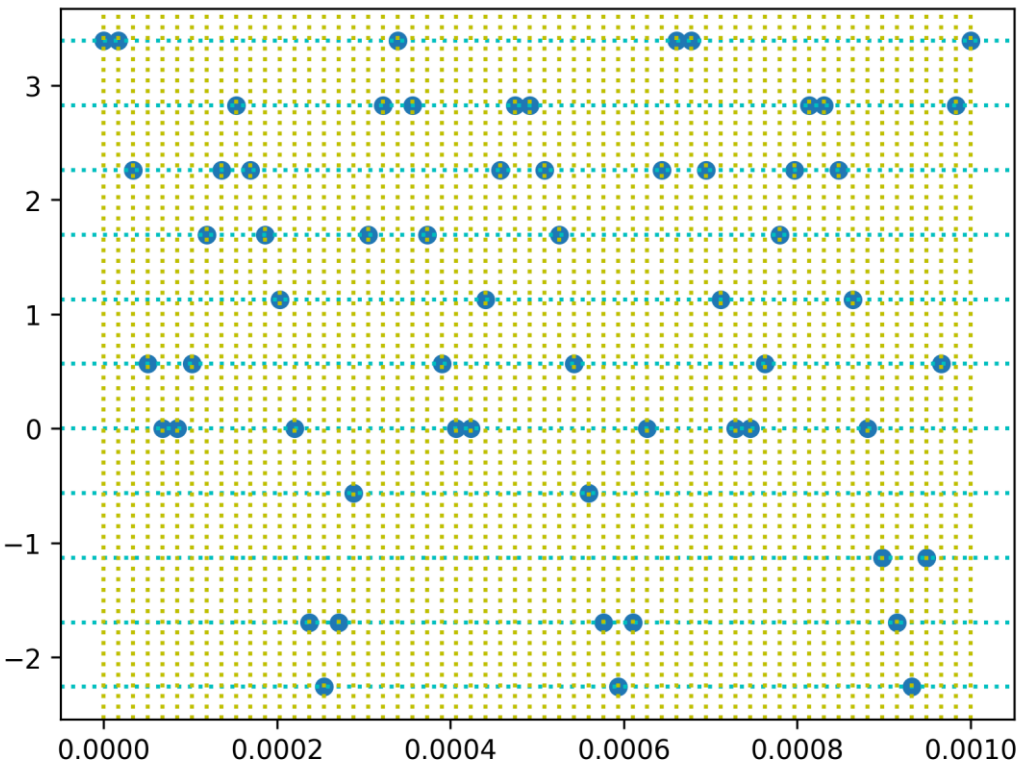
Для тех же 3кГц построим дискретные по амплитуде графики. С 4 и 6 уровнями сигнала.

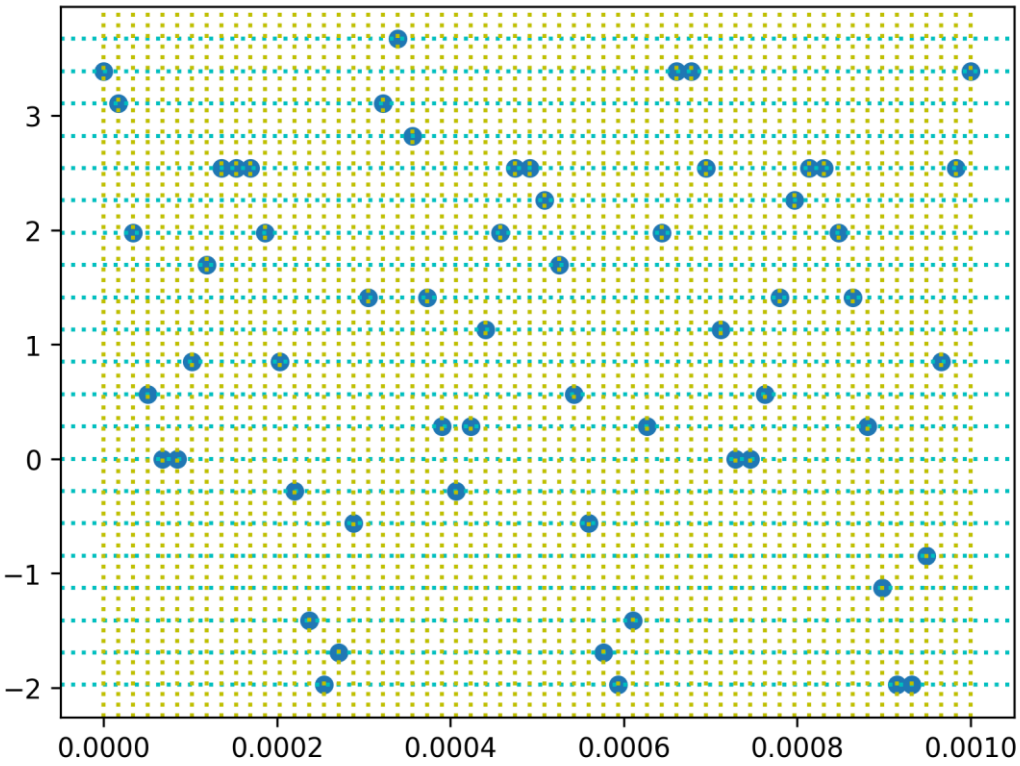


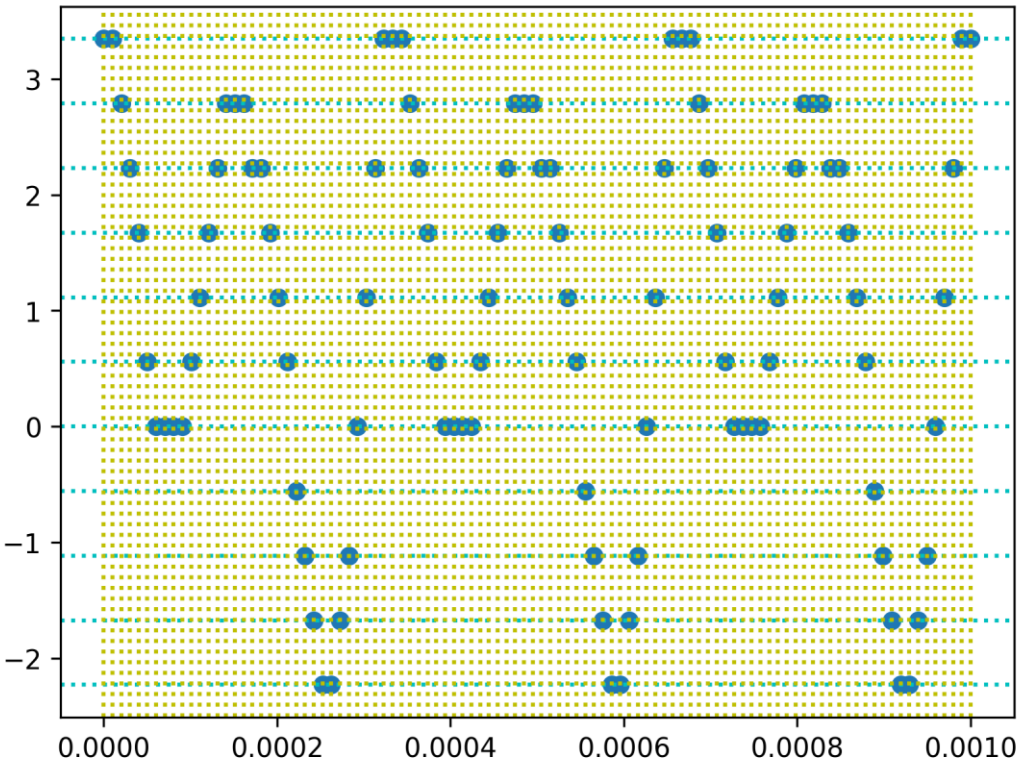


Построим так же соответствующий цифровой сигнал. При 30 уровнях по времени и 10 по амплитуде. 60 и 10, 60 и 20, 100 и 10.









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

```
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)
```