Лабораторная работа №1
По курсу «Цифровая обработка сигналов»
Изучение спектров сигналов

Работу выполнил:
Студент группы Б14-501
Попцов П. А.
Работу проверил:
Ктитров С. В.

1. Задание 1. Вычислить спектр непрерывного сигнала.

Исходный сигнал:	x1(t) = 10sin(2*pi*5*t) + 5sin(2*pi*10*t)
Время начала сигнала:	t0 = 0.01
Время конца сигнала:	t1 = 4.99
Период дискретизации:	dt1 = 0.01

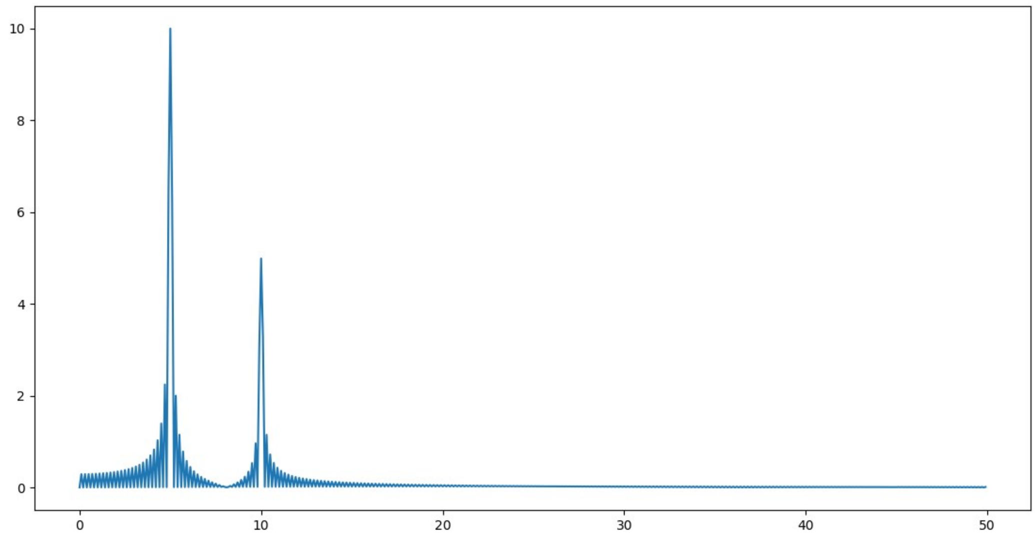

Рисунок 1. Спектр непрерывного сигнала x1(t).

Спектр получен с помощью реализованной функции *Sc*, которая вычисляет спектр непрерывного сигнала по определению с помощью метода интегрирования по частям.
На рисунке 1 приведён амплитудный спектр сигнала x1(t), по оси абсцисс отложены частоты в герцах, по оси ординат амплитуды, соответствующие своим частотам.

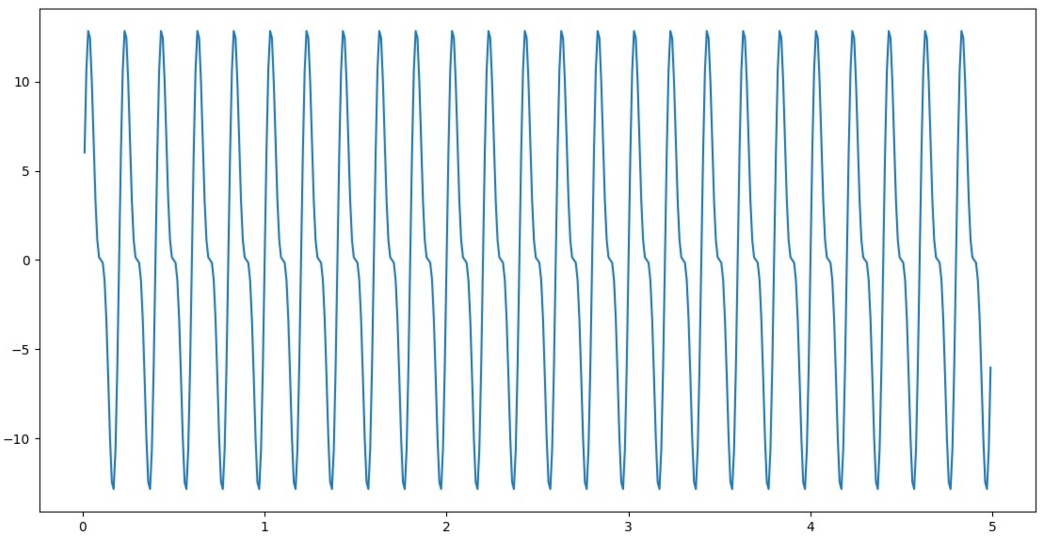

Рисунок 2: Дискретные отсчёты сигнала x1(t) для периода дискретизации dt1 = 0.01.

2. Задание 2. Дискретизовать сигнал.

Исходный сигнал: x1(t) = 10sin(2*pi*5*t) + 5sin(2*pi*10*t)

Время начала сигнала: t0 = 0.01

Время конца сигнала: t1 = 4.99

Период дискретизации: dt1 = 0.01

На рисунке 2 приведён график отсётов сигнала x1(t), полученный с помощью реализованной функции *SigcToSigd* и выполненный для 0.01 <= t <= 4.99 с периодом дискретизации dt1 = 0.01. По оси абсцисс отложено время t в секнудах, по оси ординат величина сигнала x1(t).

3. Задание 3. Вычислить спектр дискретного сигнала по определению.

Исходный сигнал: x1(t) = 10sin(2*pi*5*t) + 5sin(2*pi*10*t)

Время начала сигнала: t0 = 0.01

Время конца сигнала: t1 = 4.99

Период дискретизации: dt1 = 0.01

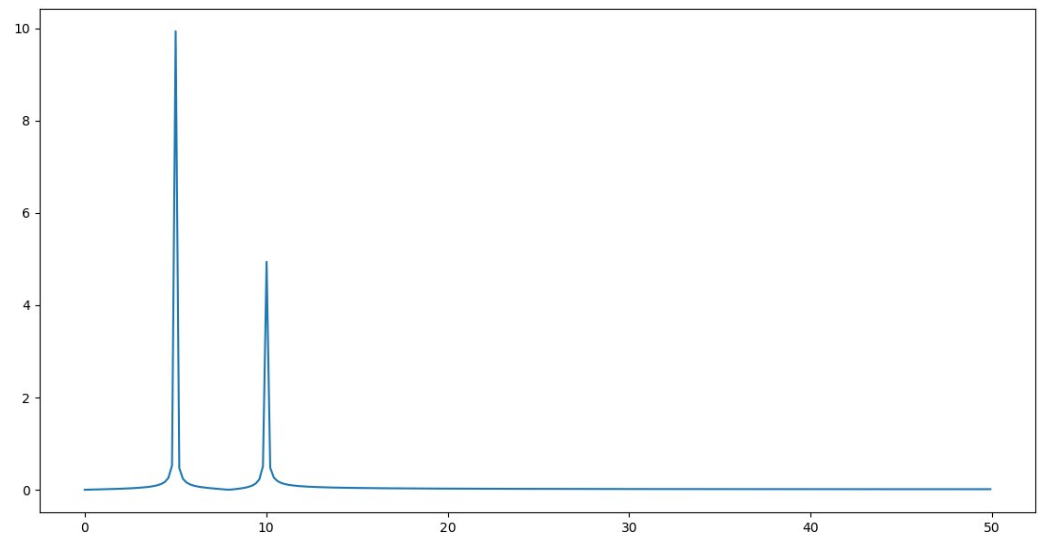

Рисунок 3: Спектр дискретного сигнала x1(t) вычисленный по определению спектра дискретного сигнала для периода дискретизации dt1 = 0.01.

На рисунке 3 приведён амплитудный спектр дискретного сигнала x1(t), по оси абсцисс отложены частоты в герцах, по оси ординат амплитуды, соответствующие своим частотам. Спектр дискретного сигнала x1(t) получен с помощью реализованной функции *Sd.*

Невооруженным глазом видно сходство непрерывного спектра на рисунке 1 и дискретного спектра на рисунке 3.

4. Задание 4. Вычислить спектр дискретного сигнала через спектр непрерывного. Сравнить результат с заданием 3.

Исходный сигнал: x1(t) = 10sin(2*pi*5*t) + 5sin(2*pi*10*t)

Время начала сигнала: t0 = 0.01

Время конца сигнала: t1 = 4.99

Период дискретизации: dt1 = 0.01

На рисунке 4 представлено два амплитудных спектра. Верхний амплитудный спектра представляет собой дискретный амплитудный спектр, вычисленный по определению. Нижний — спектр, полученный из непрерывного спектра. На обоих графиках по оси абсцисс отложена частота в герцах, по оси ординат амплитуда сигнала. Из графиков видно, что амплитудный спектр дискретного сигнала, полученный двумя разными способами — совпадает.

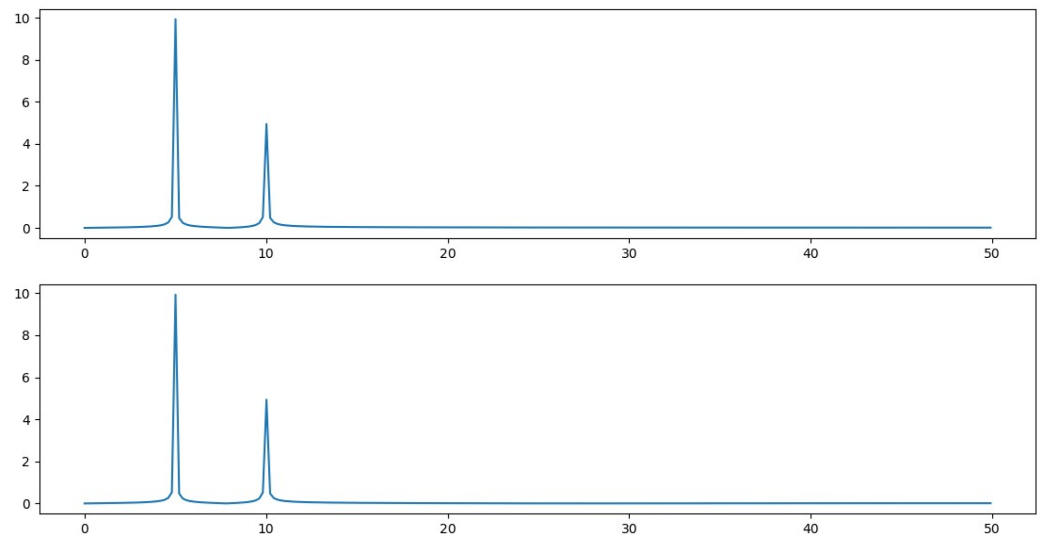

Рисунок 4: Амплитудный спектр дискретного сигнала x1(t), полученный по определению спектра дискретного сигнала (сверху) и амплитудный спектр дискретного сигнала, полученного из спектра непрерывного сигнала x1(t) (снизу). Значения получены для периода дискретизации dt1 = 0.01.

5. Задание 5. Выполнить задания 2-4 для другого шага дискретизации.

Исходный сигнал: x1(t) = 10sin(2*pi*5*t) + 5sin(2*pi*10*t)
Время начала сигнала: t0 = 0.01
Время конца сигнала: t1 = 4.99
Период дискретизации: dt2 = 0.005 = dt1/2

На рисунках 5-7 приведены результаты выполнения заданий 2-4 соответственно для тех же данных, приведенных на рисунках 2-4, за исключением периода (частоты) дискретизации, которая была уменьшена в два раза: с 0.01 секунд до 0.005 секунд соответственно. Как видно, полученные графики на рисунках 5-7 не имеют принципиальных отличий от ранее полученных графиков на рисунках 2-4, так же новые графики не получились более детализированными, по сравнению с графиками для периода дискретизации dt1 = 0.01, хотя этот эффект является очевидным и естественным следствием уменьшения периода дискретизации (увеличения частоты дискретизации). Такой результат можно объяснить обратившись к виду сигнала x1(t), он содержит две гармоники с частотами 5 и 10 герц соответственно, но выбранный период дискретизации dt1 = 0.01 уже в пять раз меньше минимально возможного периода дискретизации из теоремы Котельникова (T = 1/(2*f) => T1 = 1/20 = 0.05, T2 = 1/10 = 0.1), а второй выбранный период дискретизации dt2 = 0.005 уже в десять больше, чем величина T1 = 0.05, и в двадцать раз больше, чем величина T2 = 0.1.

6. Задание 6. Задать спектр удовлетворяющий теореме Котельникова1, получить сигнал соответствующий данному спектру.

Исходный спектр: S(w) = rect(w/(2*a)), a = pi/2, где:

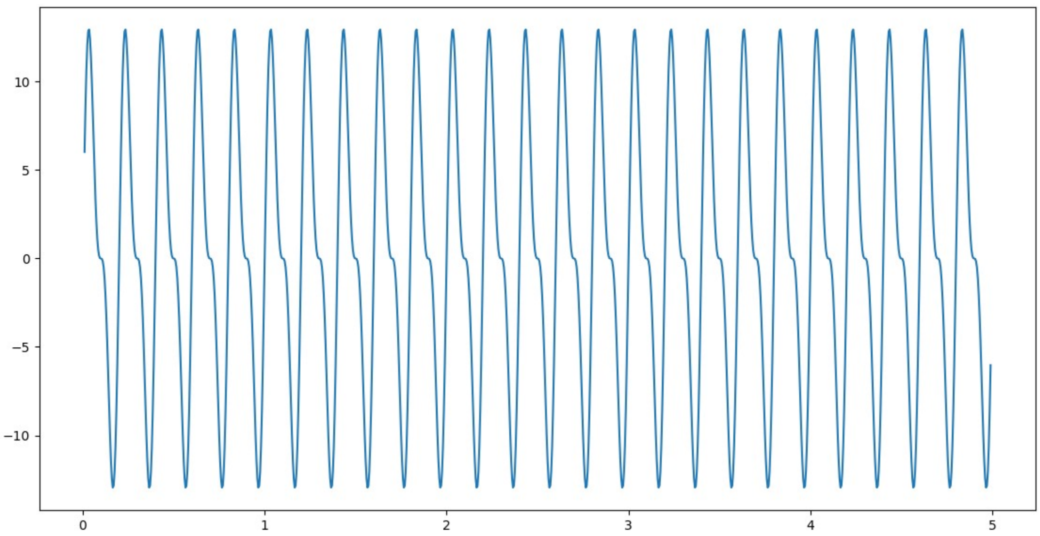

Рисунок 5: Дискретные отсчёты сигнала x1(t) для периода дискретизации dt2 = 0.005.

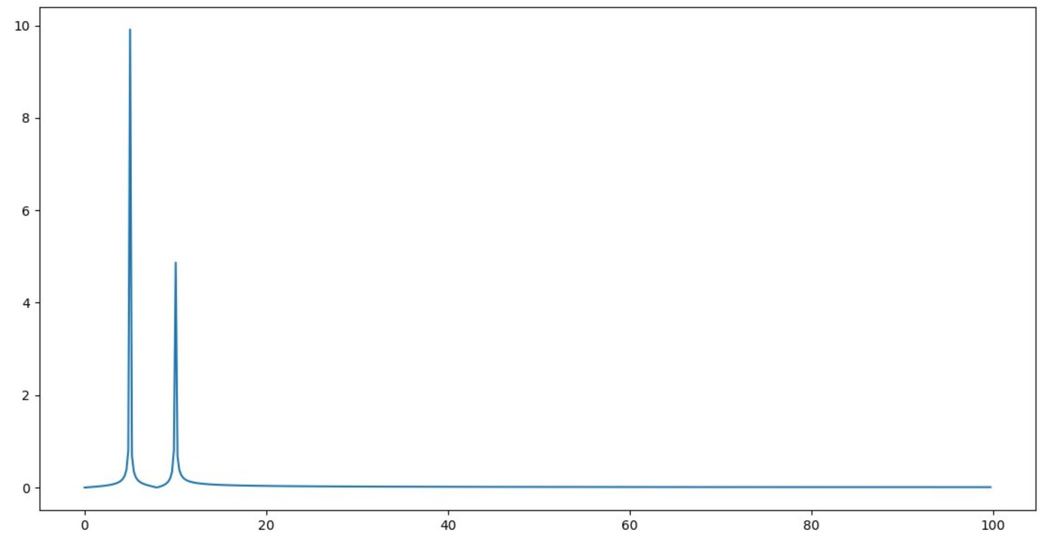

Рисунок 6: Спектр дискретного сигнала x1(t) вычисленный по определению спектра дискретного сигнала для периода дискретизации dt2 = 0.005.

$$rect(x) = \begin{cases} 0; & |x| > 0.5 \\ 0.5; & |x| = 0.5 \\ 1; & |x| < 0.5 \end{cases}$$

Начальная частота(w, рад/с): w0 = -pi
Конечная частота(w, рад/с): w1 = pi
Период дискретизации по циклической частоте: dw = pi/500

На рисунке 8 представлен график заданного спектра S(w), при это по оси абсцисс отложена циклическая частота w = 2*pi*f рад/с, а по оси ординат величина спектра S(w). Заданный спектр является действительной функцией действительной переменной.

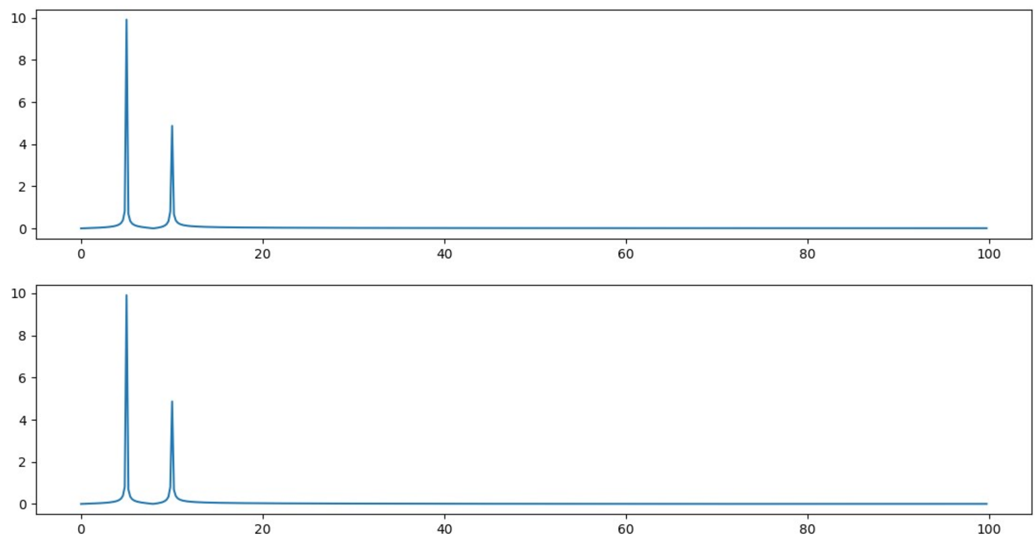

Рисунок 7: Амплитудный спектр дискретного сигнала x1(t), полученный по определению спектра дискретного сигнала (сверху) и амплитудный спектр дискретного сигнала, полученного из спектра непрерывного сигнала x1(t) (снизу). Значения получены для периода дискретизации dt2 = 0.005.

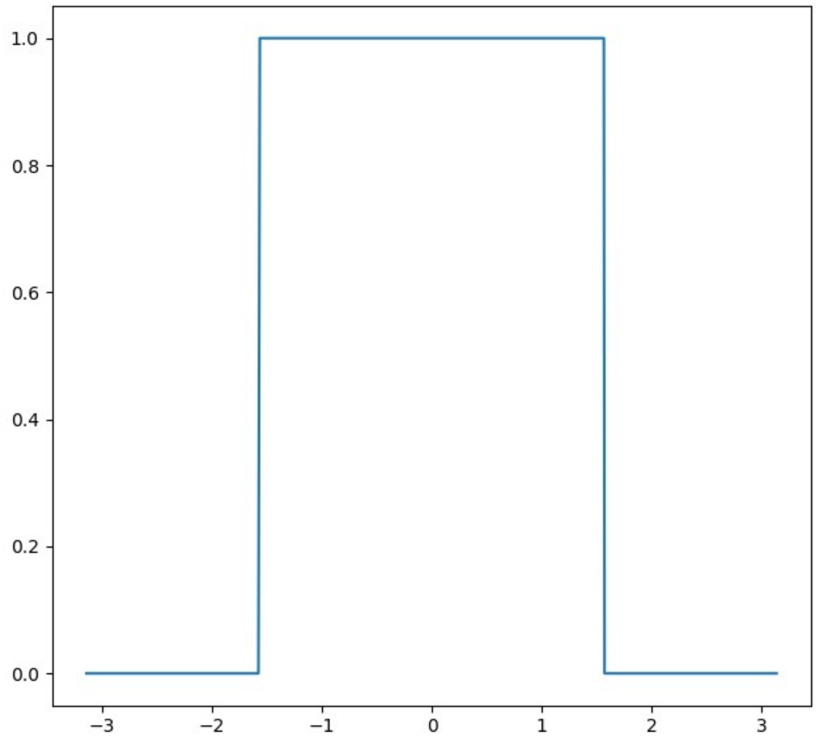

Рисунок 8: Заданный спектр S(w).

Для заданного спектра S(w) исходный сигнал y(t) можно найти аналитически, выполнив

обратное преобразование Фурье для функции S(w). В результате получим искомый сигнал y(t), которому соответствует спектр S(w):
y(t) = a/pi*sinc(a*t), где a = pi/2 и sinc(x) = sin(pi*x)/(pi*x). Следовательно:
y(t) = 0.5*sinc(0.5*pi*t).

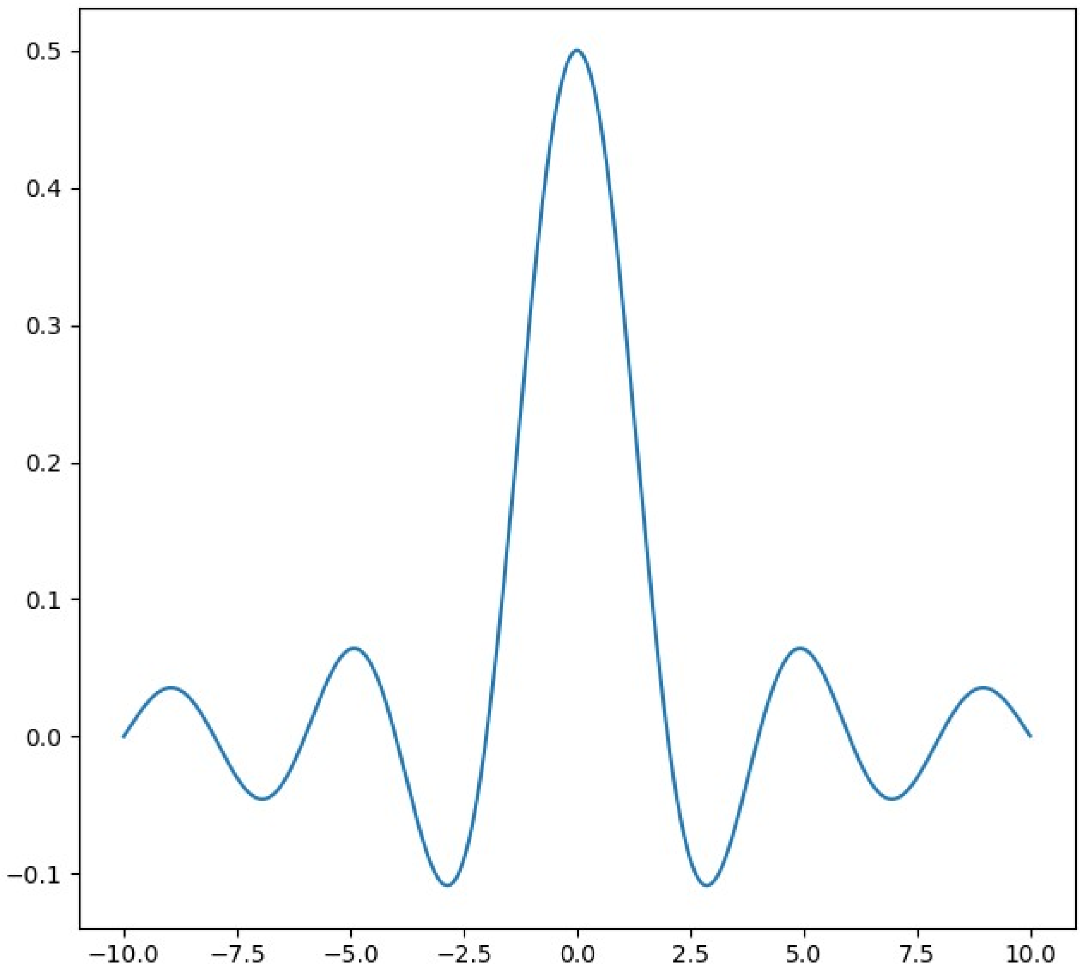

Рисунок 9: График сигнала y(t) на отрезке времени -10 <= t <= 10.

На рисунке 9 приведен график сигнала y(t) на отрезке времени от -10 секунд до 10 секунд включительно.

7. Задание 7. Дискретизовать сигнал, выбрав такты удовлетворяющими и неудовлетворяющими теореме Котельникова.

Исходный сигнал: x2(t) = 10.0sin(2*pi*1*t) + 10sin(2*pi*2*t) + … + 10sin(2*pi*50*t)
Первый эксперимент:
Начальное время: t1 = 0.1
Конечное время: t2 = 9.9
Период дискретизации: dt = 0.1
Второй эксперимент:
Начальное время: t1' = 0.01

Конечное время: t2' = 9.99

Период дискретизации: dt' = 0.01

По условию теоремы Котельникова: для восстановления сигнала период дискретизации T должен быть меньше или равен чем 1/(2*50) = 0.01. Следовательно, в первом эксперименте исходные данные соответствуют случаю, когда условия теоремы Котельникова не удовлетворены, а во втором эксперименте данные соответствуют случаю, когда условия теоремы Котельникова удовлетворены. Для обоих экспериментов непрерывный сигнал x2(t) дискретизуется с помощью функции *SigcToSigd*.

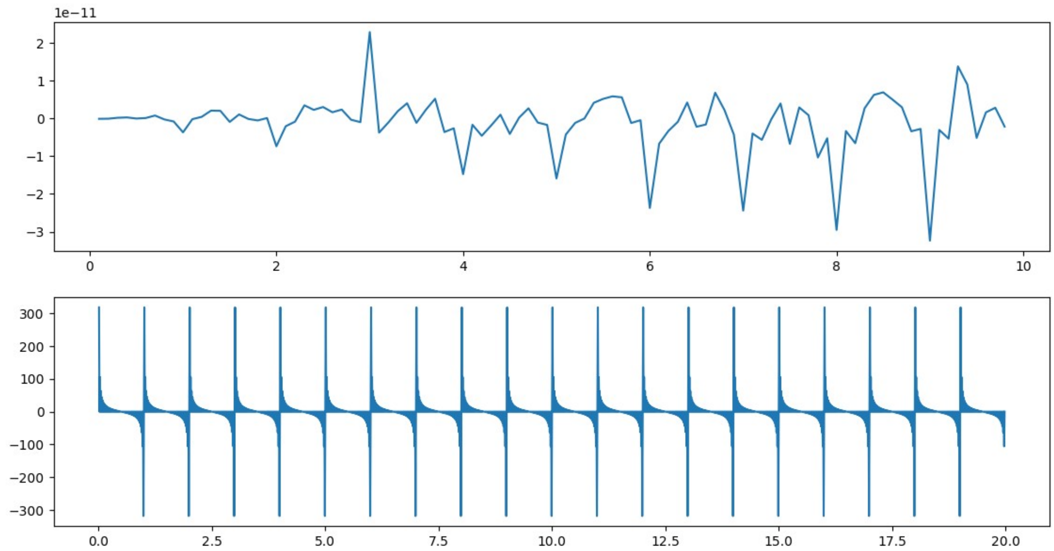

Рисунок 10: Дискретизованный непрерывный сигнал x2(t) для данных первого эксперимента (сверху) и для данных второго эксперимента (снизу).

На рисунке 10 изображены два набора отсчётов непрерывного сигнала x2(t), которые соответствуют первому и второму эксперименту соответственно. По оси абсцисс на обоих подграфиках отложено время, а по оси ординат величина сигнала x2(t).

8. Задание 8. Восстановить непрерывный сигнал по отсчётам полученным в задании 6.

При помощи разработанной функции *TK*, отсчёты полученные в пункте 6 используются для восстановления непрерывного сигнала x2(t).

Восстановленный сигнал демонстрируется на отрезке времени от 0.1 секунды до 19.9 секунд в первом эксперименте и на отрезке времени от 0.01 секунды до 19.99 секунд во втором эксперименте. Графики восстановленных сигналов x2(t) приведены на рисунках 11 и 12. Рисунки 11 и 12 разделены на две половины — верхнюю и нижнюю. В верхней половине приводится результат восстановления непрерывного сигнала по его отсчётам, а в нижней половине приводится истинный сигнал x2(t). Как видно из рисунков 11 и 12 — для данных не удовлетворяющим условиям теоремы Котельникова восстановление непрерывного сигнала по отсчётам невозможна, восстановленная картина сильно искажена по сравнению с истинным ходом сигнала x2(t). Для условий удовлетворяющих теореме Котельникова ситуация прямо противоположная. Непрерывный сигнал был успешно восстановлен по дискретным отсчётам.

9. Задание 9. Дискретизовать спектр, заданный в задании 6.

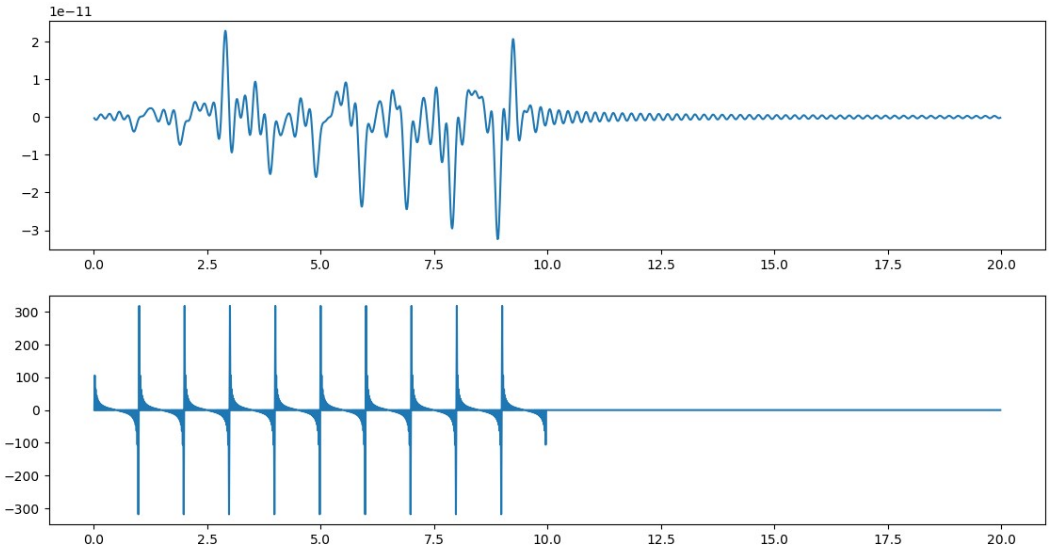

Рисунок 11: Восстановленный сигнал x2(t) для первого эксперимента. Сверху восстановленный по теореме Котельникова сигнал, снизу оригинальный непрерывный сигнал для проверки.

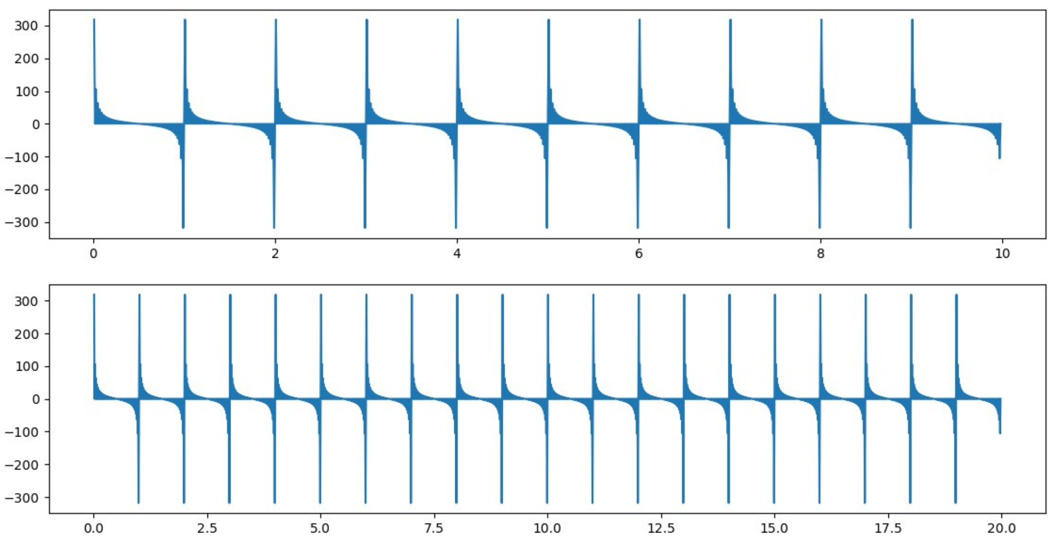

Рисунок 12: Восстановленный сигнал x2(t) для второго эксперимента. Сверху восстановленный по теореме Котельникова сигнал, снизу оригинальный непрерывный сигнал для проверки.

Спектр S(w), заданный в задании 6 был успешно дискретизован с использованием разработанной функции S*igcToSigd*, в ходе выполнения задания 6. Полученный график спектра S(w) изображен на рисунке 8.

10. Задание 10. Восстановить сигнал по дискретному спектру.

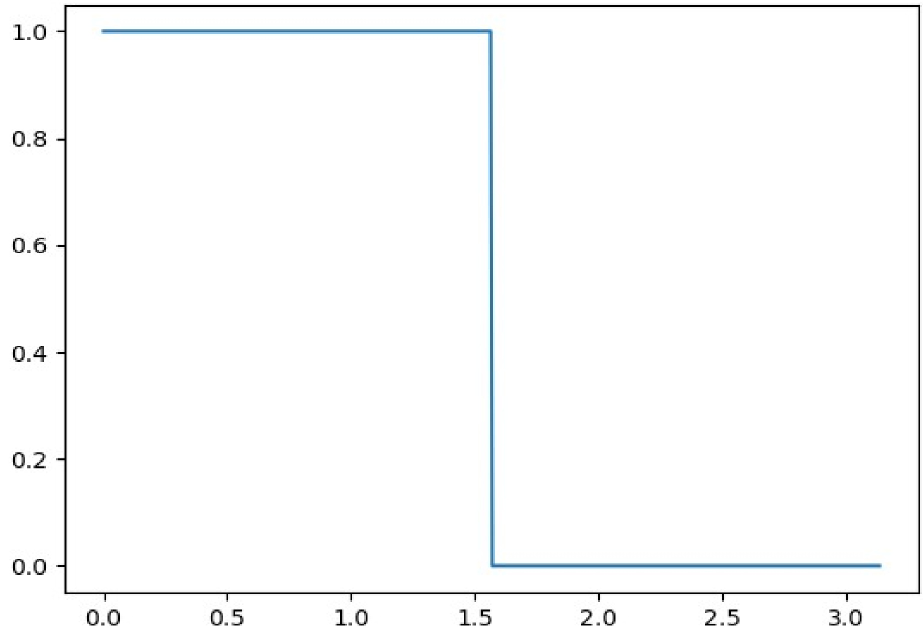

Рисунок 13. График спектра S(w) для w из правой полуплоскости.

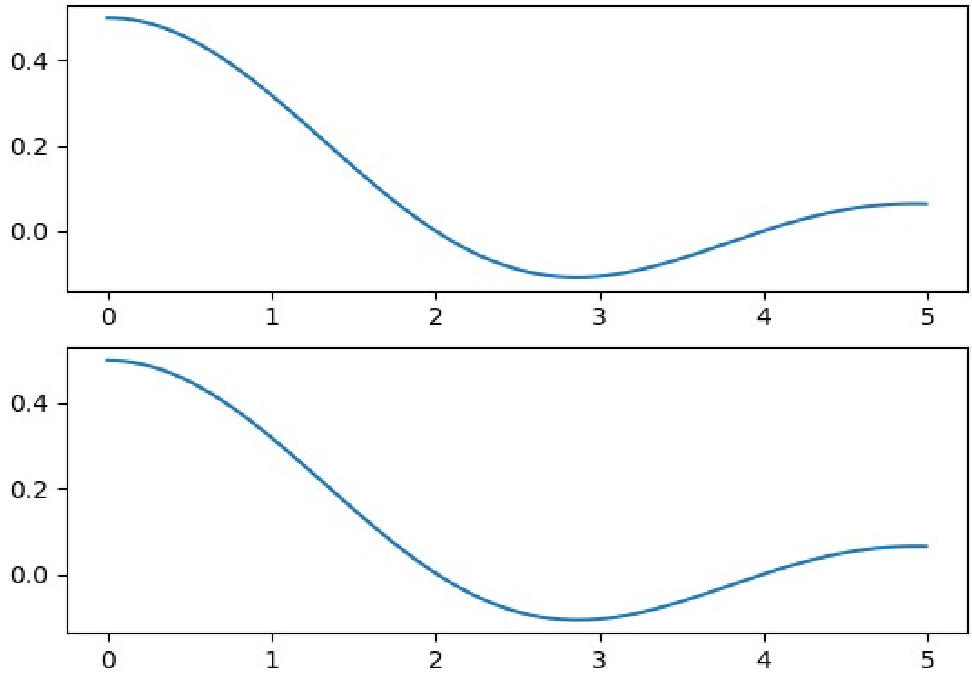

Рисунок 14. Восстановленный сигнал y(t) из дискретного спектра S(w) (снизу), сигнал y(t) восстановленный аналитическим способом из непрерывного спектра S(w) (сверху).

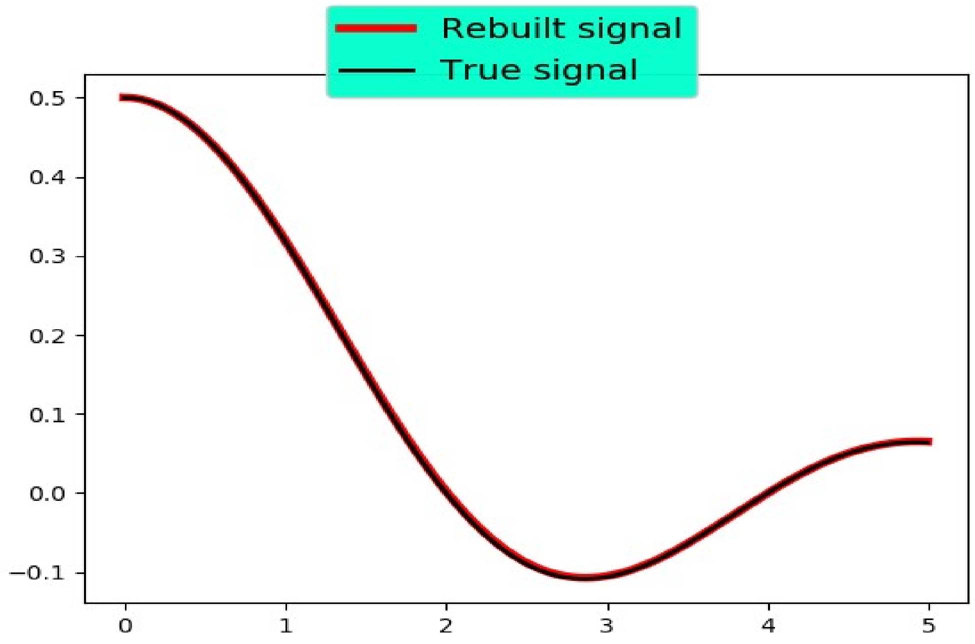

Рисунок 15. Наложенные сигналы: истинный сигнал y(t), полученный аналитически из непрерывного спектра S(w)(черный) и восстановленный сигнал y(t), полученный из дискретного спектра S(w)(красный).

С помощью программного кода, содержащегося в файле *lab1_10_task.py,* в данной работе восстанавливается сигнал y(t) по отсчётам дискретизованного непрерывного спектра S(w), удовлетворяющего условиям теореме Котельникова. Зная, что истинный сигнал y(t), полученный аналитически, является четной функцией, осуществим разложение искомого восстанавливаемого сигнала по конечному числу гармоник-косинусов, умноженных на амплитуды, которые соответственно являются дискретными отсётами непрерывного спектра S(w), нормированными на величину N, в точках w_k = k*dw, где k меняется от 0 для N — 1. На рисунках 14 и 15 представлены графики истинного и восстановленного сигнала y(t), по оси абсцисс отложено время t, по оси ординат величина сигнала y(t). По этим графикам наглядно видно, что восстановленный сигнал и истинный сигнал y(t) совпадают в пределах погрешности.

11. Исходный код программной реализации на языке python.

а) lab1_with_plots.py

```
import math
import cmath
import numpy as np
import os
import sys
from matplotlib import pyplot as plt
pi = math.pi
e = math.e
#Функция Хевисайда
def Theta(x):
	if "array" in str(type(x)) or "list" in str(type(x)):
		return np.array([ (1.0, 0.0)[value < 0] for value in x ])
	else:
		if x >= 0:
			return 1.0
		else:
			return 0.0
#тестовый сигнал, две гармоники и нормальный шум, чтобы не так скучно было
def sig(t):
	return 10*np.sin(2*pi*5*t) + 5*np.sin(2*pi*10*t) + np.random.normal(0.0, 1.0, (1, len(t))["array"
in str(type(t_a)) or "list" in str(type(t_a))])
def sig2(t):
	return 10*np.sin(2*pi*5*t) + 5*np.sin(2*pi*10*t)
#начальное время наблюдения сигнала
t_0 = 0.0
#конечное время наблюдения сигнала
t_f = 5.0
#период дискретизации
T = 0.01
t_a = np.arange(t_0, t_f, T)
N = len(t_a)
#прямое ДПФ
#на вход отсчеты сигнала
def FT(x):
	return [ sum([ x[n]*cmath.exp(-1j*(2*pi/len(x))*k*n) for n in range(0, len(x)) ]) for k in range(0,
len(x)) ]
#обратное ДПФ
#на вход отсчеты изображения Фурье сигнала
def FT1(x):
	return list(map(lambda t: t.conjugate()/len(x), FT(list(map(lambda t: t.conjugate(), x)))))
#вычислить дискретный спектр, ну тут надо нормировку делать, я так в уире делал вроде
#вроде все правильно, но объяснить не могу почему так
#вернет амплитудный спектр, фазовый спектр и частоты в герцах для оси x спектра
```

```
#на вход отсчеты сигнала
def Sd(x):
    X = FT(x)
    Ph = [ cmath.phase(z) for z in X ]
    A = [ abs(z) for z in X ]
    A = A[ 0 : round(len(A)/2) ]
    Ph = Ph[ 0 : round(len(Ph)/2) ]
    A = [ a/len(A) for a in A ]
    F = [ f/(T*len(x)) for f in np.arange(0, len(A), 1.0) ]
    return [A, Ph, F]
#посчитать спектр непрерывного сигнала
#ну тут надо численно проинтегрировать методом трапеций
#вроде результат сходится с дискретным спектром
#на вход: сигнал, начальное время наблюдения, конечное время наблюдения
#шаг сетки по времени для интрегрирования
#частоты для которых считем спектр
def Sc(sg, t1, t2, ts, f):
    t = np.arange(t1, t2, ts)
    N = len(t)
    w = [ 2*pi*fr for fr in f ]
    tmp = []
    for w0 in w:
        x = sg(t)*np.exp(-1j*w0*t)
        tmp.append(sum( [ (x[i] + x[i + 1])*0.5*(t[i + 1] - t[i]) for i in range(0, N - 1) ] ))
    return [ [ abs(z)/(ts*round(N/2)) for z in tmp ], [ cmath.phase(z) for z in tmp ] ]
def Sinc(x):
    epsilon = 1e-6
    if abs(x) < epsilon:
        return 1
    else:
        return math.sin(x)/x
#восстанавливаем сигнал по теореме котельникова
#x - отсчёты
#t1, t2 - начальный и конечнй моменты времени
#ts - шаг по времени
#TT - период дискретизации
def TK(x, t1, t2, ts, TT):
    tmp = []
    qwe = []
    t = np.arange(t1, t2, ts)
    NN = len(t)
    K = len(x)
    for ind, t0 in enumerate(list(t)):
        qwe = []
        qwe = [ x[k]*Sinc(pi/TT*(float(t0) - float(k)*TT)) for k in range(0, K) ]
        tmp.append(sum(qwe))
```

```
	return tmp
#получить дискретный спектр из непрерывного
#принимает:
#сам сигнал
#начальное, конечное время наблюдения и шаг дескритизации сигнала
#начальную и конечную частоты спектра
#период дескритизации дискретного спектра
#возвращает амлитудный и фазовый дискретный спектр сигнала
def SdFromSc(sg, t1, t2, ts, f1, f2, df):
	Ad = []
	Pd = []
	f = np.arange(f1, f2, df)
	tmp = Sc(sg, t1, t2, ts, f)
	return tmp
#Получить дискретный спектр из непрерывного
#Вход: непрерывный спектр
#Начальная, конечная частоты и шаг дискретизации по частоте
#Выход: дискретный спектр
def ScToSd(S_func, f0, f1, df):
	frs = np.arange(f0, f1, df)
	return [ S_func(i) for i in frs ]
#Получить дискретный сигнал из непрерывного
#Вход: непрерывный сигнал
#Начальное, конечное время и шаг дискретизации по времени
#Выход: дискретный сигнал
def SigcToSigd(sig_func, t0, t1, dt):
	ta = np.arange(t0, t1, dt)
#	return [ sig_func(i) for i in ta ]
	return sig_func(ta)
#Восстановить сигнал по дискретному спектру
#A, Ph - амплитудный и фазовый дискретные спектры
#f - частоты спектра
#t0, t1, ts - начальное, конечное время и шаг по времени
def SdToSig(A, Ph, f = [], t0 = 0, t1 = 0, ts = 0):
	Tmp = [ cmath.rect(v*len(A), Ph[i]) for i, v in enumerate(A) ]
	Tmp2 = [ cmath.rect(v*len(A), Ph[i]) for i, v in enumerate(A) ]
	Tmp2.reverse()
	Q = Tmp + np.conj(Tmp2).tolist()
	Q.pop()
	res = FT1(Q)
	return res
#Получить сигнал из непрерывного спектра
#Вход: функция спектра сигнала
#Начальная и конечная частота, шаг дескритизации по частоте и набор временных отсчётов
#Выход: сиганл соответствующий входному спектру
def SigFromSc(sfunc, f1, f2, df, t):
```

```
    fa = np.arange(f1, f2, df)
    N = len(fa)
    w = np.array([ 2*pi*fr for fr in fa ])
    w = 2*pi*fa
    TMP = []
#   print("N: ", N)
    for t0 in t:
#       print(t)
        x = [ x0/(2.0*pi) for x0 in sfunc(fa)*np.exp(1j*w*t0) ]
#       print(t0)
        TMP.append(sum( [ (x[i] + x[i + 1])*0.5*(w[i + 1] - w[i]) for i in range(0, N - 1) ] ))
    return TMP
delta_t = 0.1
t_begin = 0.1
t_end = 9.9
t_begin_restored = 0.01
t_end_restored = 19.99
delta_t_restored = 0.01
time_array = np.arange(t_begin, t_end, delta_t)
time_array_restored = np.arange(t_begin_restored, t_end_restored, delta_t_restored)
example_signal_1 = lambda t: np.sum([(1.0/i)*np.sin(2*pi*sqrt(i)*t) for i in range(1, 21)], axis = 0)
example_signal_1_d = SigcToSigd(example_signal_1, t_begin, t_end, delta_t)
example_signal_1_restored = TK(example_signal_1_d, t_begin_restored, t_end_restored,
delta_t_restored, delta_t)
example_signal_1_check_values = example_signal_1(np.arange(t_begin_restored, t_end_restored,
delta_t_restored))
example_signal_2 = lambda t: np.sum([(10.0)*np.sin(2*pi*i*t) for i in range(1, 51) ], axis = 0)
example_signal_2_d = SigcToSigd(example_signal_2, t_begin, t_end, delta_t)
example_signal_2_restored = TK(example_signal_2_d, t_begin_restored, t_end_restored,
delta_t_restored, delta_t)
example_signal_2_check_values = example_signal_2(np.arange(t_begin_restored, t_end_restored,
delta_t_restored))
delta_t = 0.01
t_begin = 0.01
t_end = 4.99
time_array = np.arange(t_begin, t_end, delta_t)
T = delta_t
example_signal_3 = lambda t: 10*np.sin(2*pi*5*t) + 5*np.sin(2*pi*10*t)
freq_array = np.arange(0.0, 50.0, 0.1)
example_signal_3_c_spectrum = Sc(example_signal_3, t_begin, t_end, delta_t, freq_array)
example_signal_3_d = SigcToSigd(example_signal_3, t_begin, t_end, delta_t)
example_signal_3_d_spectrum = Sd(example_signal_3_d)
example_signal_3_d_spectrum_from_c = Sc(example_signal_3, t_begin, t_end, delta_t,
example_signal_3_d_spectrum[-1])
delta_t = 0.005
T = delta_t
```

```
example_signal_3_d_ = SigcToSigd(example_signal_3, t_begin, t_end, delta_t)
example_signal_3_d_spectrum_ = Sd(example_signal_3_d_)
example_signal_3_d_spectrum_from_c_ = Sc(example_signal_3, t_begin, t_end, delta_t,
example_signal_3_d_spectrum_[-1])
f_begin = -pi
f_end = pi
delta_f = 2*pi/1000.0
freq_array = np.arange(f_begin, f_end, delta_f)
f_begin = -3/(2*pi)
f_end = 6/(2*pi)
delta_f = 1/(2*pi)
freq_array = np.arange(f_begin, f_end, delta_f)
def Delta(x):
    epsilon = 10e-9
    if "array" in str(type(x)) or "list" in str(type(x)):
        return np.array([ (0.0, 1.0)[abs(value) <= epsilon] for value in x ])
    else:
        if abs(x) <= epsilon:
            return 1.0
        else:
            return 0.0
example_spectrum_1 = lambda f: Theta(pi/2 - abs(f))*np.sin(f + pi/2)
example_spectrum_1 = lambda f: Delta(f - 1/(2*pi))
example_spectrum_1_values = example_spectrum_1(freq_array)
example_signal_4 = lambda t: sqrt(2.0/pi)*np.cos(pi*t/2.0)/(1 - t**2)
t_begin = 0.0
t_end = 5.0
delta_t = 0.01
T = delta_t
time_array = np.arange(t_begin, t_end, delta_t)
example_signal_from_spectrum_1 = SigFromSc(example_spectrum_1, f_begin, f_end, delta_f,
time_array)
########################################################################
def S(w):
    return np.exp(-1j*w)
def x(t):
    epsilon = 10e-6
    if abs(t - 1.0) <= epsilon:
        return (pi/2.0)**0.5
    else:
        return ((2.0/pi)**0.5)*(np.sin((t - 1)*pi/2.0))/(t - 1)
w_array = np.arange(-pi/2.0, pi/2.0, pi/200.0)
#####################################################################
plt.figure(0)
freq0 = np.arange(0.0, 50.0, 0.1)
plt.plot(freq0, example_signal_3_c_spectrum[0])
```

```
plt.figure(1)
time0 = np.arange(0.01, 4.99, 0.01)
plt.plot(time0, example_signal_3_d)
plt.figure(2)
plt.plot(example_signal_3_d_spectrum[-1], example_signal_3_d_spectrum[0])
plt.figure(3)
plt.subplot(2, 1, 1)
plt.plot(example_signal_3_d_spectrum[-1], example_signal_3_d_spectrum[0])
plt.subplot(2, 1, 2)
plt.plot(example_signal_3_d_spectrum[-1], example_signal_3_d_spectrum_from_c[0])
plt.figure(4)
time1 = np.arange(0.01, 4.99, 0.005)
plt.plot(time1, example_signal_3_d_)
plt.figure(6)
plt.plot(example_signal_3_d_spectrum_[-1], example_signal_3_d_spectrum_[0])
plt.figure(7)
plt.subplot(2, 1, 1)
plt.plot(example_signal_3_d_spectrum_[-1], example_signal_3_d_spectrum_[0])
plt.subplot(2, 1, 2)
plt.plot(example_signal_3_d_spectrum_[-1], example_signal_3_d_spectrum_from_c_[0])
plt.figure(8)
plt.plot(freq_array, example_spectrum_1_values)
plt.figure(9)
plt.plot(time_array, example_signal_from_spectrum_1)
time2 = np.arange(0.1, 9.9, 0.1)
time3 = np.arange(0.01, 19.99, 0.01)
plt.figure(10)
plt.subplot(2, 1, 1)
plt.plot(time2, example_signal_1_d)
plt.subplot(2, 1, 2)
plt.plot(time3, example_signal_1_check_values)
time2 = np.arange(0.1, 9.9, 0.1)
time3 = np.arange(0.01, 19.99, 0.01)
plt.figure(10)
plt.subplot(2, 1, 1)
plt.plot(time2, example_signal_2_d)
plt.subplot(2, 1, 2)
plt.plot(time3, example_signal_2_check_values)
dt = 0.01
tb = 0.01
te = 9.99
tbr = 0.01
ter = 19.99
dtr = 0.01
xd = SigcToSigd(example_signal_2, tb, te, dt)
xdr = TK(xd, tbr, ter, dtr, dt)
```

```
xdc = example_signal_2(np.arange(tbr, ter, dtr))
plt.figure(11)
plt.subplot(2, 1, 1)
plt.plot(np.arange(tb, te, dt), xd)
plt.subplot(2, 1, 2)
plt.plot(time3, xdc)
plt.figure(12)
plt.subplot(2, 1, 1)
plt.plot(time3, example_signal_2_restored)
plt.subplot(2, 1, 2)
plt.plot(time3, xdr)
```

б) lab1_10_task.py

```
import numpy as np
from math import *
import cmath
import matplotlib.pyplot as plt
import scipy as sc
InfoMessage = "lab1_10_task.py"
def Sinc(x):
    return np.sinc(x/pi)
def Rect(x):
    epsilon = 10e-12
    if abs(x) > 0.5:
        return 0.0
    elif abs(abs(x) - 0.5) <= epsilon:
        return 0.5
    elif abs(x) < 0.5:
        return 1.0
def IntPart(x):
    return int(modf(x)[-1])
def Doc(x):
    print(x.__doc__)
print(InfoMessage)
a = pi/2.0
x_func = lambda t: a/pi*Sinc(a*t)
S_func = lambda w: Rect(w/(2.0*a))
wb = -pi
we = pi
dw = 2*pi/1000
wa = np.arange(wb, we, dw)
tb = 0.0
te = 5.0
```

```
dt = 0.01
ta = np.arange(tb, te, dt)
S_d = np.array([ S_func(w) for w in wa ])
x_d = np.array([ x_func(t) for t in ta ])
N = len(S_d)
S_dh = S_d[round(N/2) : ]
W_dh = wa[round(N/2) : ]
x_0 = []
for t in ta:
    x_0.append(np.sum([ Sn*np.cos(W_dh[n]*t)/(len(S_dh)) for n, Sn in enumerate(S_dh) ]))
x_0 = np.array(x_0)
plt.figure(1)
plt.plot(W_dh, S_dh)
plt.figure(2)
plt.subplot(2, 1, 1)
plt.plot(ta, x_d)
plt.subplot(2, 1, 2)
plt.plot(ta, x_0)
plt.figure(3)
p1 = plt.plot(ta, x_0, color = "red", linewidth = 4.0, label = "Rebuilt signal")
p2 = plt.plot(ta, x_d, color = "black", linewidth = 2.0, label = "True signal")
hndl = plt.gcf()
lg = hndl.legend(loc = "upper center", fontsize = "x-large")
lg.get_frame().set_facecolor('#00FFCC')
```

12. Заключение.

В рамках выполненной лабораторной работы, при помощи специально разработанного программного обеспечения, были рассмотрены свойства различного рода (в том числе дискретных и непрерывных) сигналов и их спектров.

Созданное программное обеспечение способно дискретизовать непрерывные сигналы, а также вычислять непрерывные и дискретные спектры сигналов, представляя при этом полученные данные сигналов и их спектров в наглядной графической форме.

Также в рамкой данной лабораторной работы в условиях практики на некоторых тестовых сигналах была изучена важнейшая теорема цифровой обработки сигналов — теореме Котельникова. В частности были показаны возможности по восстановлению непрерывных сигналов в зависимости от того, какой была выбрана их частота дискретизации.