МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 1

Дискретизация, эффект "маскировки" частот и восстановление дискретного сигнала

Выполнил студент 585 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Губченко В.М.

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Уланов П.Н.

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

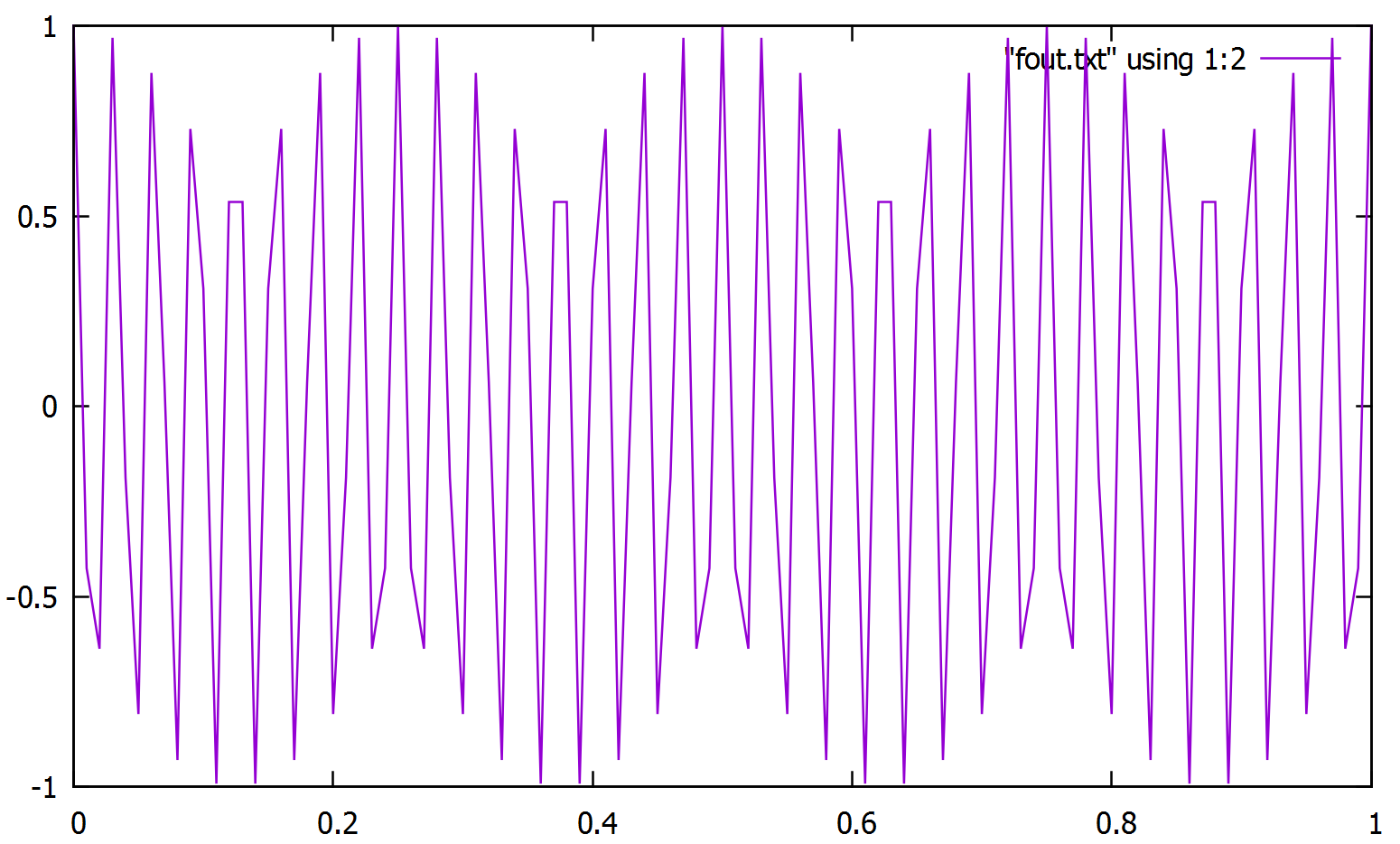
Барнаул 2019

# Задание 1

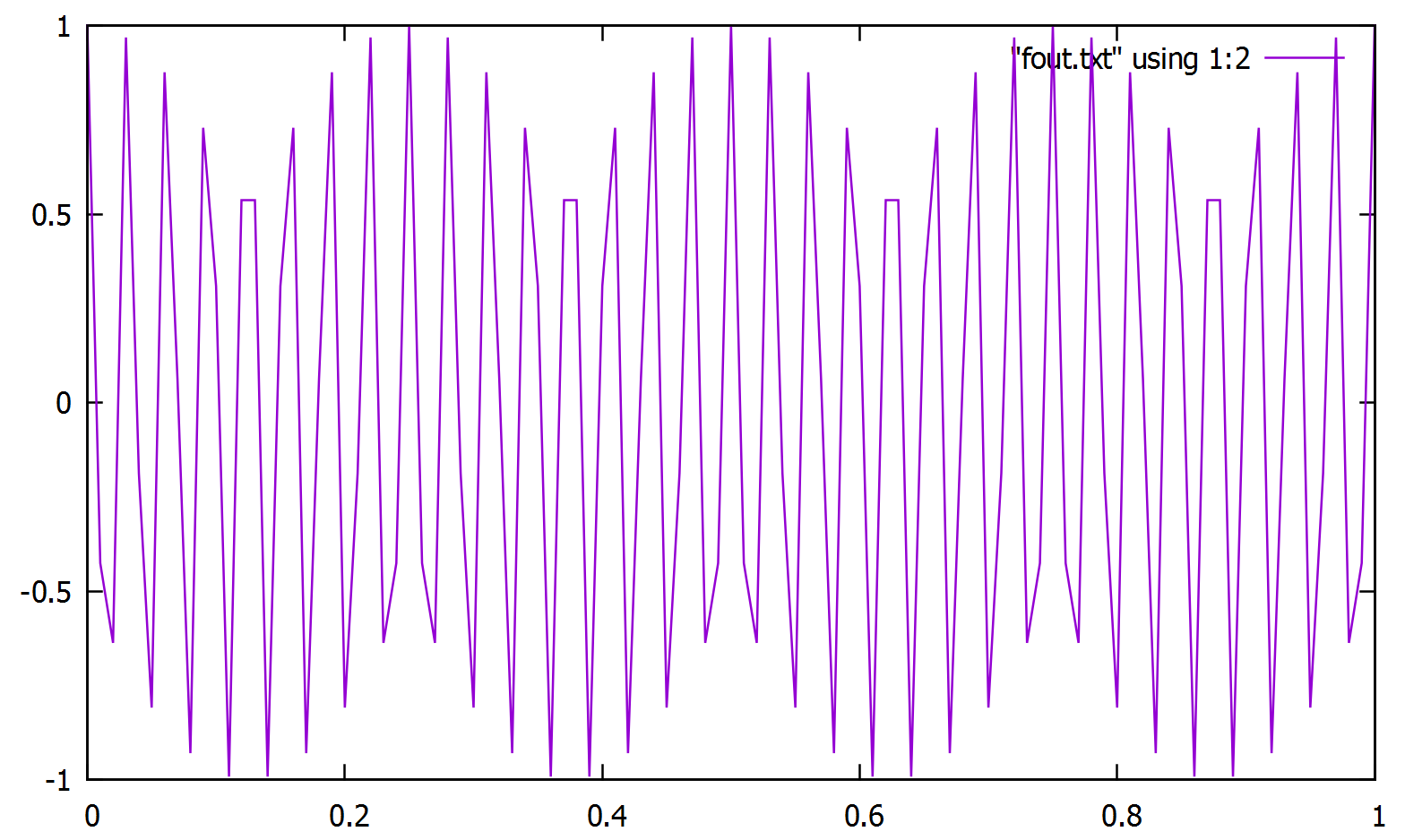
Создайте на основе непрерывного сигнала с заданной частотой дискретный сигнал с заданным интервалом дискретизации T. Для этого фактически нужно просто вычислить значения сигнала в определенные моменты времени (T, 2T, 3T,…). Определите частоту сигнала, который «маскируется» под данный сигнал при таком интервале дискретизации. Постройте графики обоих сигналов и сравните их

## Решение

Исходный сигнал *x*(*t*)=cos(2π*ft*). Частота поступающего сигнала f = 32. Частота дискретизации = 100, период дискретизации T=0.01с. частота маскирующегося сигнала .



### Рис. 1 График с частотой сигнала f=32



**Рис. 2 График с частотой сигнала f=68**

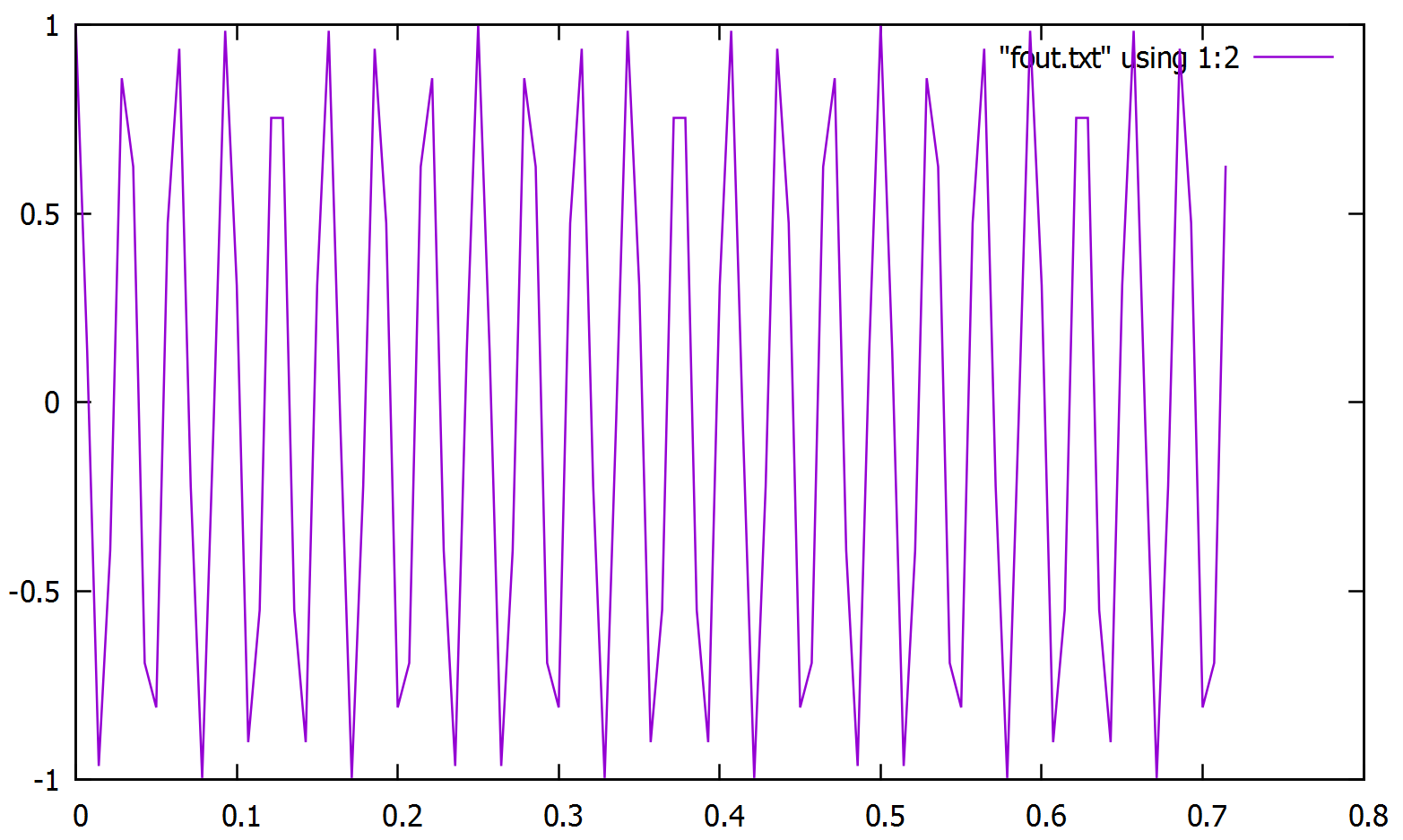
Графики сигналов одинаковы.

# Задание 2

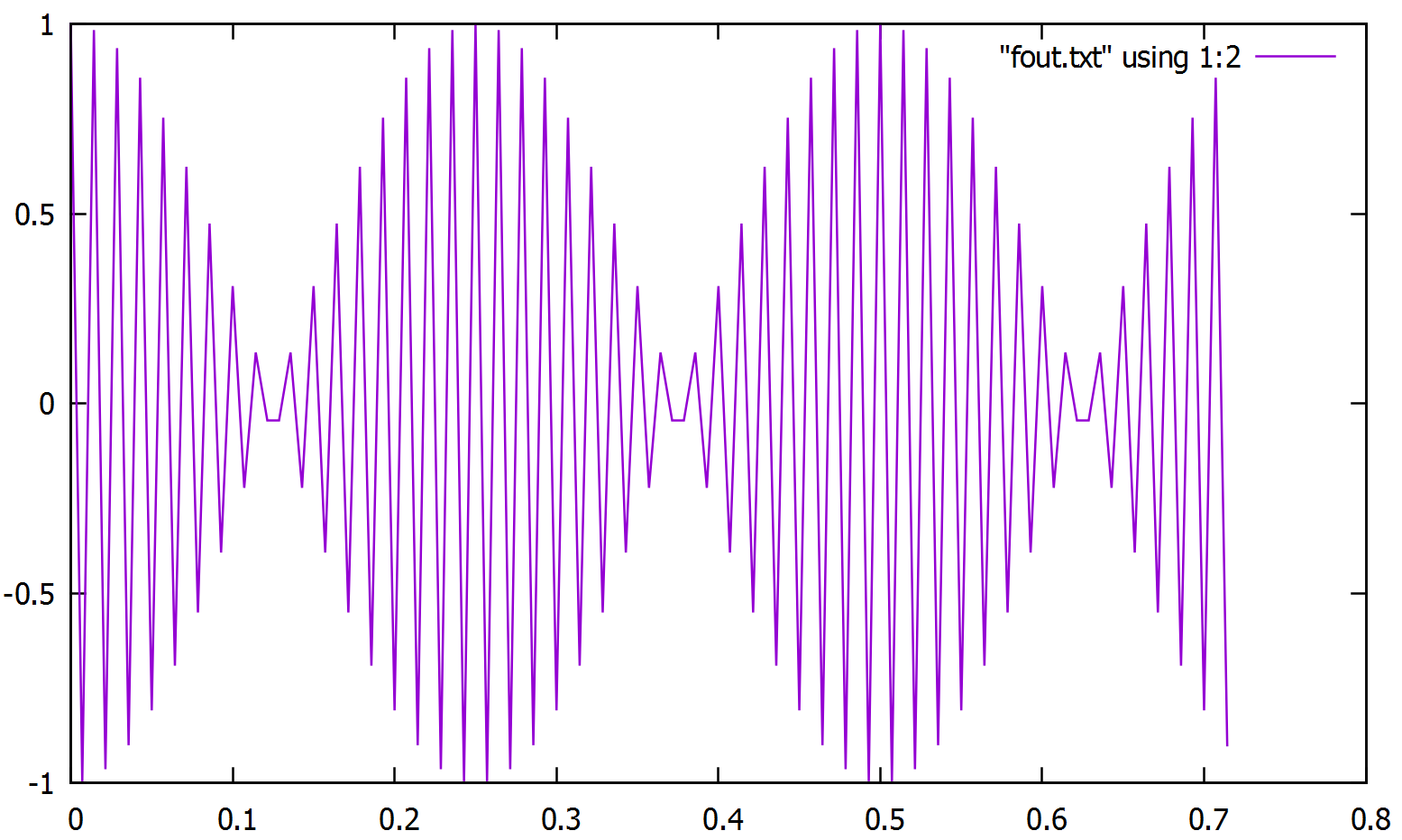
Уменьшите интервал дискретизации так, чтобы половина частоты дискретизации была больше, чем частоты обоих сигналов. Постройте графики сигналов с новым интервалом дискретизации.

## Решение

Увеличим частоту дискретизации так, чтобы её половина была больше частот входных сигналов. Увеличиваем частоту дискретизации на 40, теперь . Таким образом, частота больше не будет маскирующейся.



### Рис. 3 График с частотой сигнала f=32 и частотой дискретизации 140



### Рис. 4 График с частотой сигнала f=68 и частотой дискретизации 140

Графики сигналов отличаются.

# Задание 3

Теорема Котельникова утверждает, что если в спектре сигнала не содержится частот больших, чем половина частоты дискретизации, то исходный сигнал может быть однозначно восстановлен по своим дискретным отчетам следующим образом:



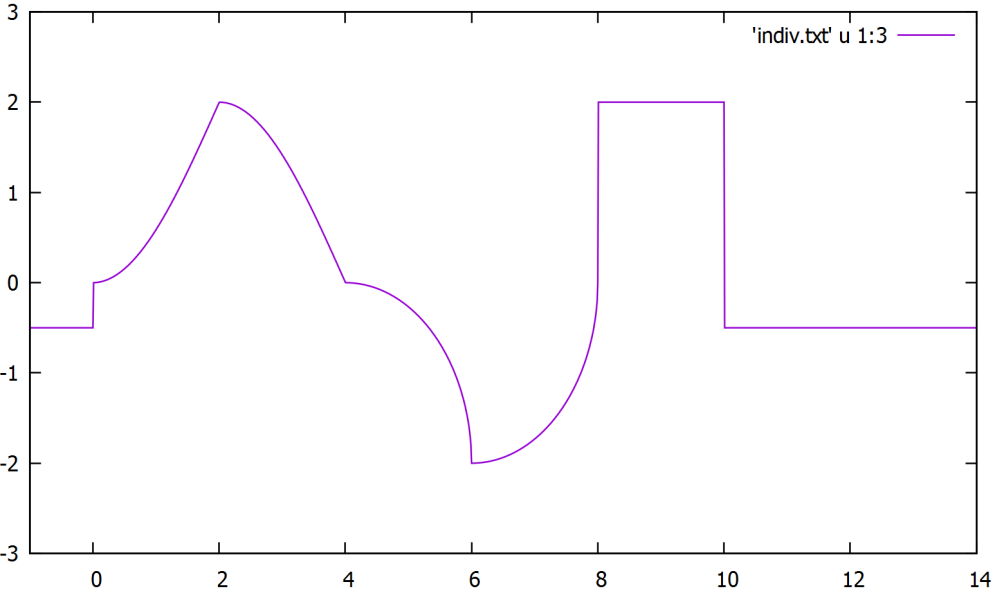
*T* – интервал (период) дискретизации,  - частота дискретизации.

Восстановите свой сигнал с помощью теоремы Котельникова, используя разное количество отсчетов (например, 10 и 50), постройте графики восстановленного сигнала.

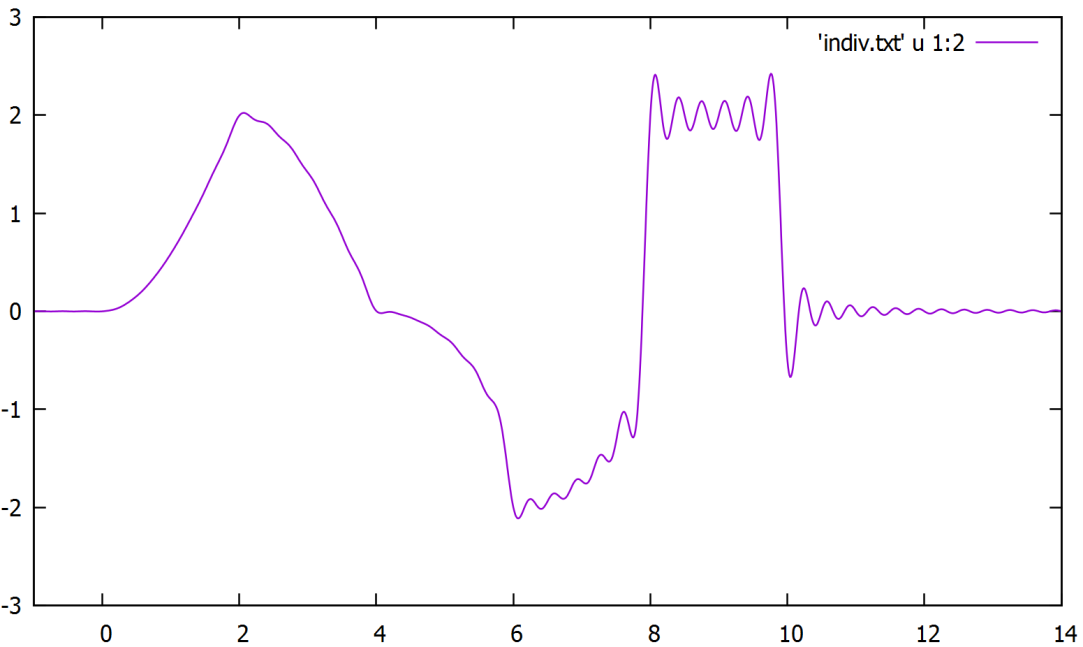
Обратите внимание, что прямое вычисление функции Котельникова  в некоторых точках (*t=nT*) приводит к делению на ноль. На самом деле, поскольку аргумент синуса тоже равен нулю, значение всей дроби следует положить равным 1 (поскольку ).

## Решение

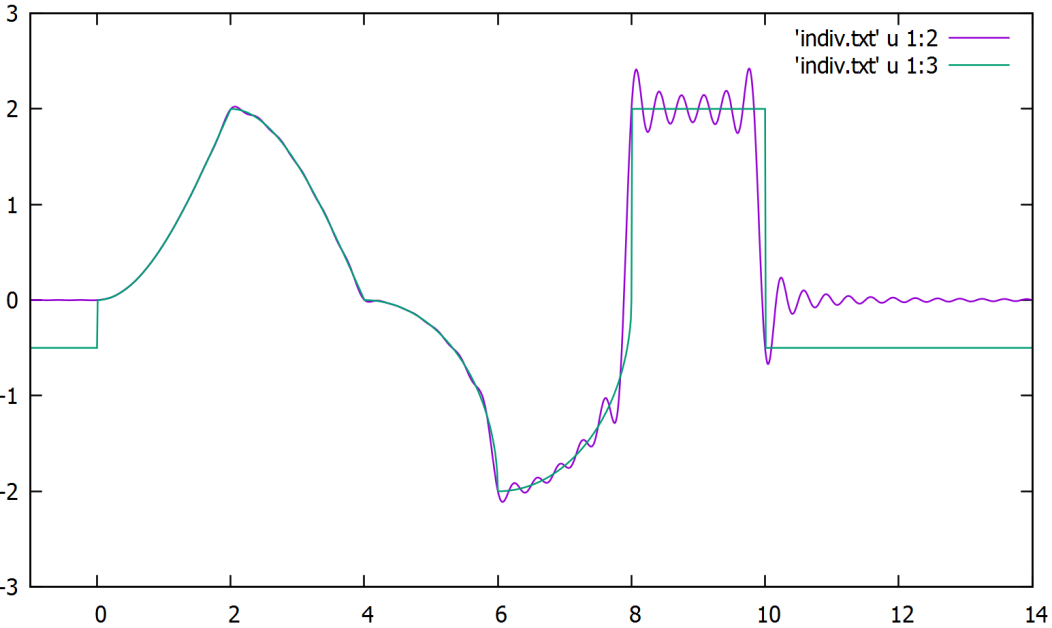
Сначала дискретизируем исходный сигнал (рис. 5). Затем восстанавливаем сигнал с помощью теоремы Котельникова (рис. 6). В некоторых точках знаменатель обращается в ноль. В этом случае по первому замечательному пределу значение всей дроби принимаем за 1. Строим графики восстановленного и идеального сигналов в одних осях (рис. 7).



**Рис. 5 Исходный сигнал**



### Рис. 6 Восстановленный сигнал



**Рис. 7 Исходный и восстановленный сигналы**

Здесь мы видим, что в кусочных функциях графика, состоящих из синусов и окружностей, сигнал восстанавливается практически идеально, а резкие изменения графика порождают отклонения восстанавливающего сигнала.

## Вывод

В ходе выполнения лабораторной работы мы научились дискретизировать сигнал, находить маскирующийся сигнал и восстанавливать сигнал по теореме Котельникова.

Функции, реализующие построение графиков всех заданий, находятся в приложении А.

# Приложение А.

#include "stdafx.h"

#include <iostream>

#include <fstream>

#include <math.h>

#define PI 3.14159265359

double cosinus(double fs, double t)

{

return cos(2\*PI\*fs\*t);

}

double cosinus\_3(double t)

{

return cos(t);

}

using namespace std;

int first()//первое задание

{

ofstream fout("fout.txt");

double fs = 68

double t;

double result;

double b, a, fd, i;

a = 0;

b = 1;

fd = 100;

double step;

for(i = 0; i <=100; i++)

{

step = (b-a)/(fd);

t = a + i \* step;

result = cosinus(fs, t);

fout << t << " " << result << endl;

cout << t << " " << result << endl;

}

cout << "fs= " << fs <<endl;

return 0;

}

int second()//второе задание

{

ofstream fout("fout.txt");

double fs = 68

double t;

double result;

double b, a, fd, i;

a = 0;

b = 1;

fd = 140;

double step;

for(i = 0; i <=100; i++)

{

step = (b-a)/(fd);

t = a + i \* step;

result = cosinus(fs, t);

fout << t << " " << result << endl;

cout << t << " " << result << endl;

}

cout << "fs= " << fs <<endl;

return 0;

}

int third()//третье задание

{

ofstream fout("3.txt");

double N = 25, td, t, i, a = -5, b = 20, G;

td = (b - a) / N;

for (t = -5; t <= 40; t+=0.01)

{

G = 0;

for (i = 0; i <= N; i++)

{

if (((t / td) - i) == 0)

G += (cosinus\_3(i \* td));

else G += (cosinus\_3(i \* td)) \* ((sin(PI \* ((t / td) - i))) / (PI \*

((t / td) - i)));

}

cout << t << " " << G << " " << cosinus\_3(t) << endl; fout << t << " " << G << " " << cosinus\_3(t) << endl;

}

fout.close();

return 0;

}

double graf\_indiv(double x)

{

if(x<0)

{

return -0.5;

}else

if(x>=0 && x<=2)

{

return 2\*sin(x/1.278+11)+2;

} else

if(x>2 && x<=4)

{

return 2\*sin((x-2)/1.274+1.57);

} else

if(x>4 && x<=6)

{

return sqrt(4-pow(x-4,2))-2;

} else

if(x>6 && x<8)

{

return -sqrt(4-pow(x-6,2));

} else

if(x<10)

{

return 2;

} else

return -0.5;

}

int tri\_dop()//индивидуалка

{

ofstream fout("indiv.txt");

double N = 60, td, t, i, a = 0, b = 10, G;

td = (b - a) / N;

for (t = -5; t <= 15; t+=0.01)

{

G = 0;

for (i = 0; i <= N; i++)

{

if (((t / td) - i) == 0)

G += (graf\_indiv(i \* td));

else G += (graf\_indiv(i \* td)) \* ((sin(PI \* ((t / td) - i))) / (PI \*((t / td) - i)));

}

fout << t << " " << G << " " << graf\_indiv(t) << endl;

}

fout.close();

return 0;

}

int main()

{

int number;

setlocale(LC\_ALL, "Russian");

cout << "Выберите номер задания:";

cin >> number;

switch (number)

{

case 1:

first();

break;

case 2:

second();

break;

case 3:

third();

break;

case 4:

tri\_dop();

break;

}

return 0;

}