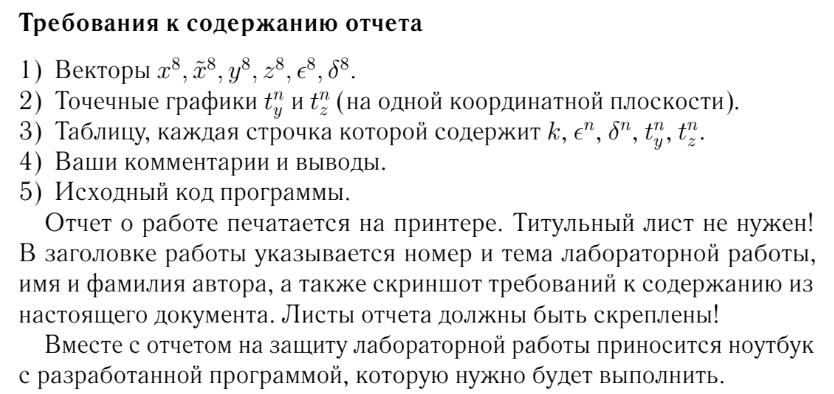
**Лабораторная работа №1, «Быстрое преобразование Фурье»**

Мусорского Павла, студента 3 курса 6б группы, 1 вариант



1. Постановка задачи

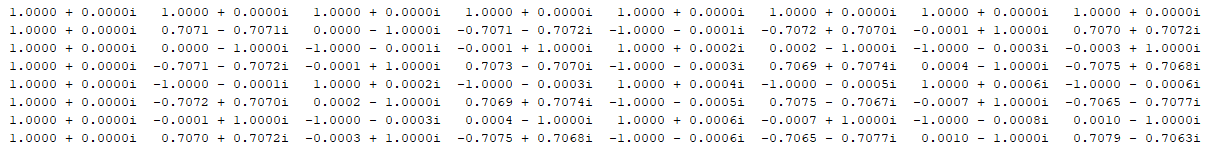
1. Сгенерировать в явном виде матрицу и ее факторизацию Кули-Тьюки. Проверить тождество = (1)
2. Выполнить основное задание с использованием рекурсивного алгоритма БПФ.

2. Ход выполнения

1. Матрицы определяются следующим образом:

= , где = exp (︂ −)︂.

В итоге генерирования в Matlab получил следующее представление для матрицы :

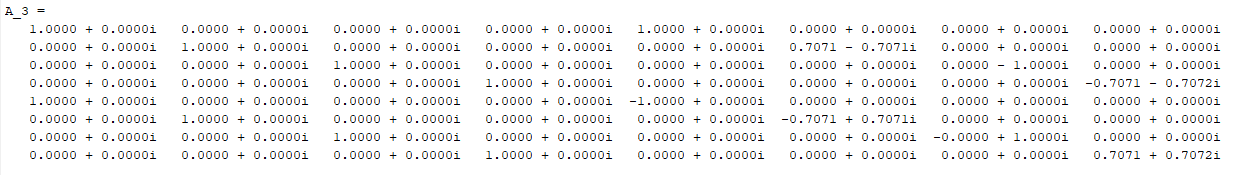


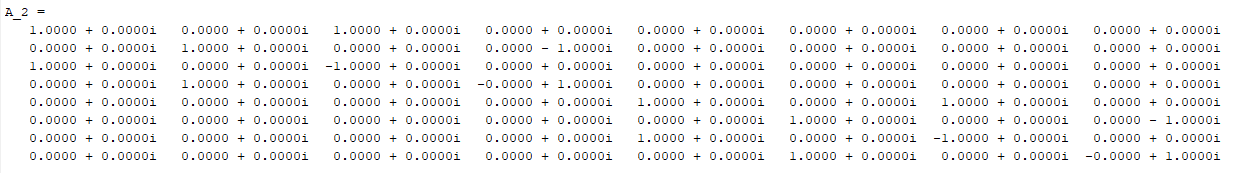
Матрицы в (1) определяются следующим образом:

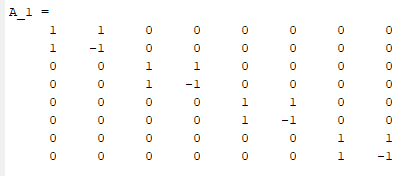
L=

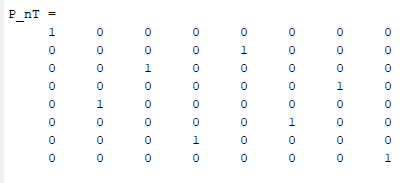
=diag {, … ,

В итоге получились матрицы следующего вида. Они приведены ниже одна под одной ввиду того, что в ином виде не уместились бы на экране.

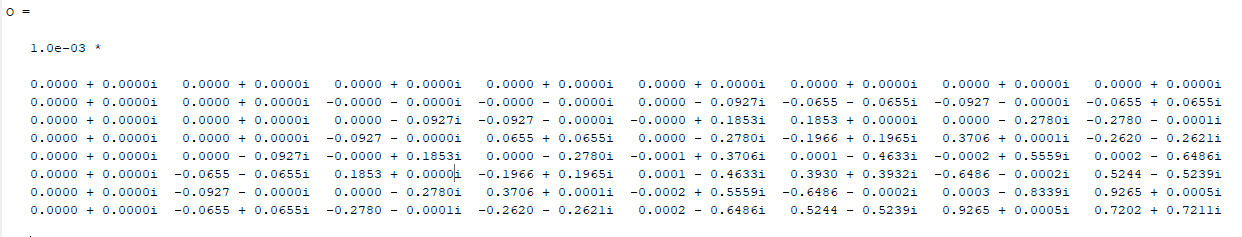








Дабы проверить справедливость тождества (1), отнял матрицы друг от друга, в итоге получив следующую матрицу-невязку O:



Считаю, что тождество можно считать справедливым, так как ненулевые элементы матрицы невязки достаточно малы.

2. Исполнив требования к содержанию отчета, получил следующие результаты.

1) Векторы

x^8=

56 20 64 99 25 67 99 4

x~^8=

56.0000 + 0.0000i 20.0000 + 0.0000i 64.0000 + 0.0000i

99.0000 + 0.0000i 25.0000 + 0.0000i 67.0000 - 0.0000i 99.0000 - 0.0000i 4.0000 - 0.0000i

y^8=

1.0e+02 \*

4.3400 + 0.0000i -0.6941 + 0.0106i -0.8200 + 0.1600i 1.3141 - 0.6894i 0.5400 + 0.0000i 1.3141 + 0.6894i -0.8200 - 0.1600i -0.6941 - 0.0106i

z^8=

1.0e+02 \*

4.3400 + 0.0000i -0.6941 + 0.0106i -0.8200 + 0.1600i 1.3141 - 0.6894i 0.5400 + 0.0000i 1.3141 + 0.6894i -0.8200 - 0.1600i -0.6941 - 0.0106i

e^8=

1.0480e-14

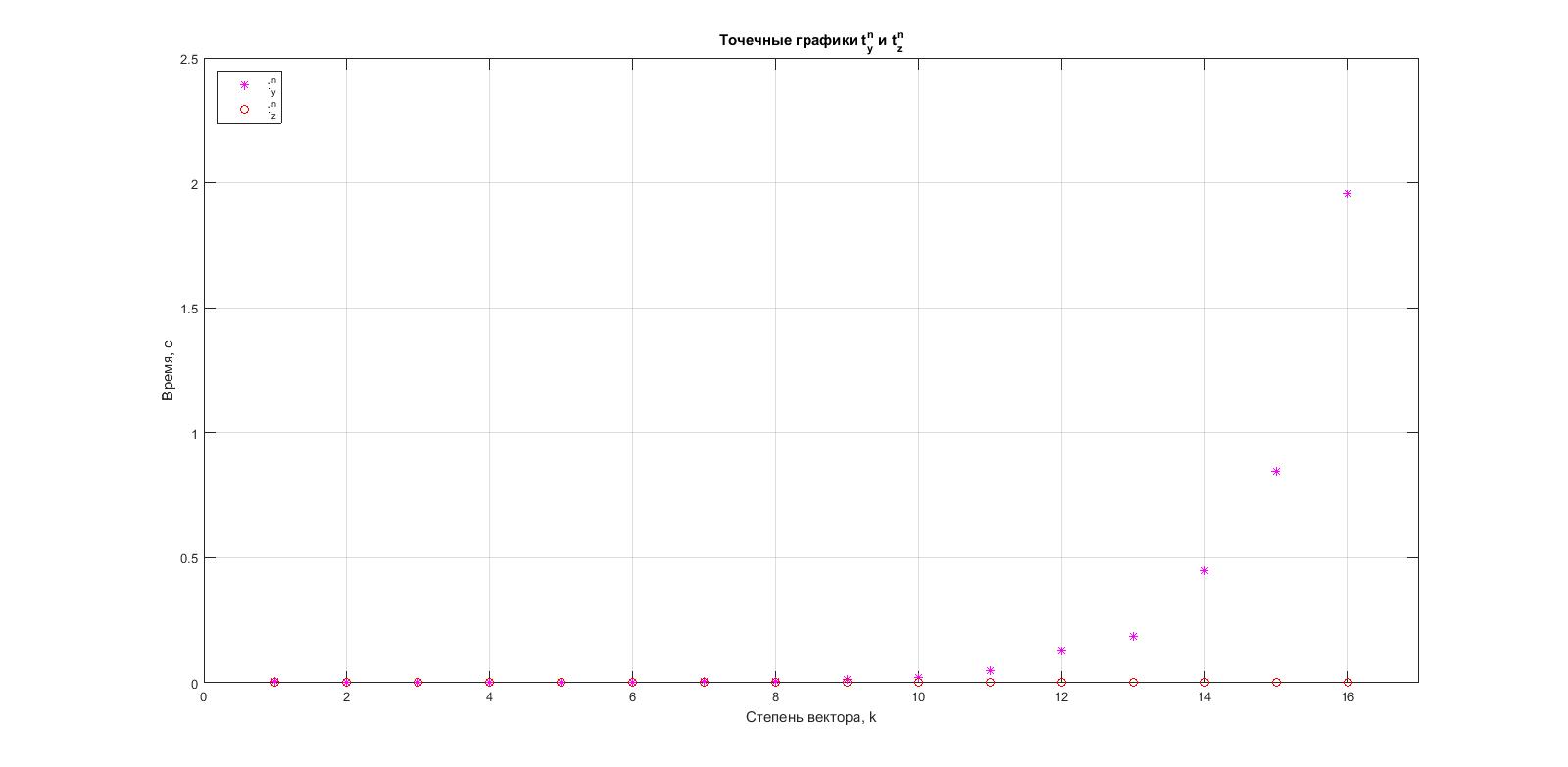
b^8=

4.6048e-14

Соответствующие векторы покомпонентно совпадают по крайней мере в пределах стандартной области вывода Matlab, а нормы невязок имеют весьма большой порядок малости (-14). На мой взгляд, перечисленное дает возможность считать эти векторы равными. Таким образом, можно заключить, что алгоритм работает адекватно.

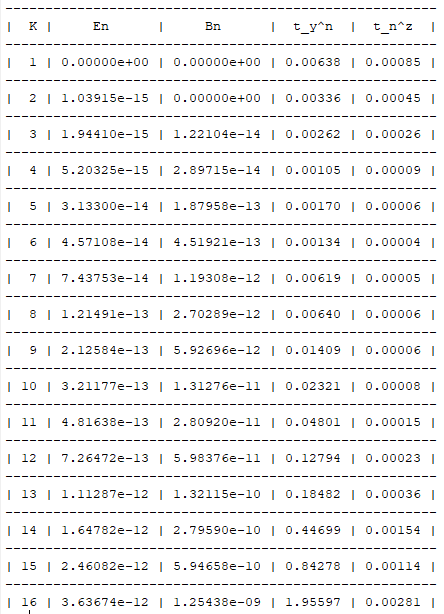
2) Точечные графики

Можно сделать очевидный вывод, что встроенная в Matlab функция для вычисления БПФ реализована гораздо оптимальнее моей, так как из графиков видно, что с увеличением количества компонент в векторе время проведения операций FFT и truefft всё более отличается.



3) Таблица с элементами векторов k,

Как уже было замечено в первом пункте, нормы невязок имеют весьма большой порядок малости, от -9 до -14, что позволяет считать использование рекурсивного алгоритма БПФ правомерным, хоть он и работает медленнее встроенной функции Matlab.



3. Листинг программ

***Факторизация Кули-Тьюки***

function Cooley\_Tukey\_Factorization ()

clc;clear all;

n=8;

el=w(n);

F=zeros(n);

for i=1:n

for j=1:n

F(i,j)=(el^(i-1))^(j-1);

end

end

disp('Matrix F\_8:');

disp(F);

t=log2(n);

A=eye(n);

B=eye(n);

for q=1:t

L=2^q;

sub=eye(L);

even=sub(1:2:end,:);

odd=sub(2:2:end,:);

PTL=[even;odd]; %PTL

IL2=eye(L/2);%IL2

r=w(L);

lya=zeros(1,L/2);

for i=1:L/2

lya(i)=r^(i-1);

end

lyambda=diag(lya); %lyambda

BL=[IL2,lyambda;IL2,(-1)\*lyambda;]; %BL

Aq=zeros(n);

RTL=zeros(n);

p=length(BL);

g=n/p;

for k=1:g

for i=1:p

for j=1:p

Aq((k-1)\*p+i,(k-1)\*p+j)=BL(i,j);

RTL((k-1)\*p+i,(k-1)\*p+j)=PTL(i,j);

end

end

end

A=Aq\*A;

fprintf('Matrix A\_%.f: \n',q);

disp(Aq);

B=B\*RTL; %RTL

end

disp('Matrix P\_8T:');

disp(B);

disp('Matrix O=F\_8-A\_3\*A\_2\*A\_1\*P\_8T:');

O=F-A\*B;

disp(O)

end

function w = w(n)

pi=3.1415;

i=sqrt(-1);

w=exp(-2\*pi\*i/n);

end

***Рекурсивный алгоритм БПФ***

function Recursive\_FFT()

clc; clear all;

T1=[]; T2=[]; K=[];

B=[]; E=[];

for k=1:16

K=[K,k];

n=2^k;

x = randi(100,n,1);

tic;

y=FFT(x);

t1=toc;

T1 = [T1, t1];

tic;

z=truefft(x);

t2=toc;

T2 = [T2, t2];

x\_=IFFT(y);

En=norm(x-x\_.');

E=[E,En];

Bn = norm(y-z.');

B = [B, Bn];

if k==3

format bank;

% форматированный вывод, дабы не выводилось 50.0000 + 0.0000i

disp('x^8:'); disp(x.');

disp('x~8:'); disp(x\_);

disp('y^8:'); disp(y);

disp('z^8:'); disp(z.');

format;

% восстановление формата вывода по умолчанию для вывода невязок

disp('e^8:'); disp(En);

disp('b^8:'); disp(Bn);

end

end

ty = double(T1);

tz=double(T2);

figure('Color','w');

plot( K,ty, 'M\*', K, tz, 'ro')

grid on

zoom on

xlabel('Степень вектора, k');

axis([0 17 0 2.5]);

ylabel('Время, c');

title('Точечные графики t\_y^n и t\_z^n');

legend({'t\_y^n','t\_z^n'},'Location','northwest','Orientation','vertical')

zoom on

disp('------------------------------------------------------');

disp('| K | En | Bn | t\_y^n | t\_n^z |');

for i=1:16

fprintf('------------------------------------------------------\n');

formatSpec = '|%3.f | %0.5e | %0.5e | %0.5f | %0.5f |\n';

fprintf(formatSpec,K(i),E(i),B(i),ty(i),tz(i));

end

end

function y = sub(x,p)

N = length(x);

if N == 1

y = x;

else

M = N / 2;

W = exp(p\*2i\*pi\*(0:M-1)/N);

odd = sub(x(1:2:N),p);

even = sub(x(2:2:N),p);

l = odd(1:M) + W.\*even(1:M);

r = odd(1:M) - W.\*even(1:M);

y = [l,r];

end

end

function y = FFT(x)

y=sub(x,-1);

end

function zn=truefft(x)

zn=fft(x);

end

function y = IFFT(x)

N = length(x);

y=1/N\*sub(x,1);

end