一、 实验环境

开发环境：MATLAB R2022a

所调函数：conv 卷积函数，fft 快速傅里叶变换，ifft 快速傅里叶逆变换

二、 实验目的

a) 编程部分：

通过多重循环的方法，手写一个傅里叶变换及傅里叶逆变换

b) 分析对比部分：

对比你使用的平台下的快速傅里叶变化的库函数；比较对于特定长度序列，他们的时间效率。可以多尝试不同长度进行分析

c) 验证卷积-傅里叶变化的关系

即对于时域的卷积等价于频域上的乘法；(请通过设计一个滤波的方法进行验证具体方法见频谱分析)

d) 频谱分析

验证对于有限长序列的频谱分析的方法画出输入序列的傅里叶频谱（谱线），并对经典的信号序列（例如特定频率正弦信号)进行频谱分析（参照频谱分析章节）

三、 实验内容

1. 手写傅里叶变换及傅里叶逆变换

a) dft

```
function [xk] = dft(x1)
N=length(x1);
xk=zeros(1,N);
for m=0:N-1
    sum=0.0;
    for k=0:N-1
        sum=sum+x1(k+1)*exp(-1j*2*pi*m*k/N);
    end
    xk(m+1)=sum;
end
xk
```

b) idft:

```
function [x] = idft(xk)
N=length(xk);
x=zeros(1,N);
for I=0:N-1
    sum=0.0;
    for k=0:N-1
        sum=sum+xk(k+1)*exp(1j*2*pi*I*k/N)/N;
    end
    x(I+1)=sum;
end
```

c) 测试运行:

测试数据: [1,1,1,1,1,1,1,1,1,1]。执行 dft 后结果 xk 如下，后将 xk 执行 idft，发现可以恢复回原序列:

```
>> dft([1,1,1,1,1,1,1,1,1,1])

xk =

  列 1 至 5

  10.0000 + 0.0000i  -0.0000 + 0.0000i  -0.0000 - 0.0000i  -0.0000 - 0.0000i   0.0000 - 0.0000i

  列 6 至 10

   0.0000 - 0.0000i   0.0000 - 0.0000i  -0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i
```

dft结果

```
ans =

  列 1 至 5

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 + 0.0000i

  列 6 至 10

   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
```

idft结果

通过 fft 验证发现 dft 结果正确:

```
>> fft([1,1,1,1,1,1,1,1,1,1])

ans =

    10     0     0     0     0     0     0     0     0     0
```

2. 分析对比

a) 通过 linspace 函数构造序列并使用 tic&toc 计时进行测试，测试代码如下：

dft.m | idft.m | run.m

```
test=linspace(1,1,100);
tic;
dft(test);
toc
tic;
fft(test);
toc
```

```
>> run
历时 0.002420 秒。
历时 0.000063 秒。
```

可知在长度为 100 的情况下，fft 比所写 dft 快 38.4 倍。

b) 多次改变序列长度，数据如下：

i. 序列长 1000

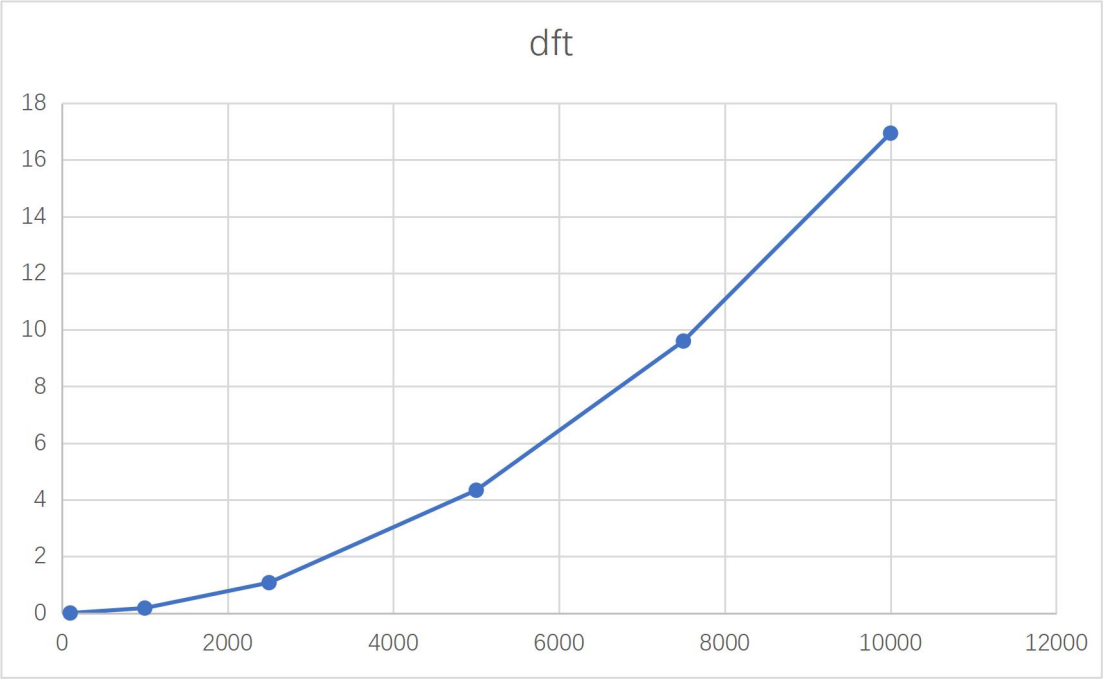
```
>> run
历时 0.176678 秒。
历时 0.001498 秒。
```

ii. 序列长 10000

```
>> run
历时 16.943531 秒。
历时 0.002738 秒。
```

iii. 多次改变序列长度进行统计，得到以下结果

序列长度	Dft 所用时间/s	Fft 所用时间/s	时间比
100	0.002420	0.000063	38.4127
1000	0.176678	0.001498	117.9426
2500	1.074187	0.001347	797.4662
5000	4.337479	0.001772	2447.787
7500	9.603992	0.001861	5160.662
10000	16.943531	0.002738	6188.287
50000	433.586940	0.005831	74358.93

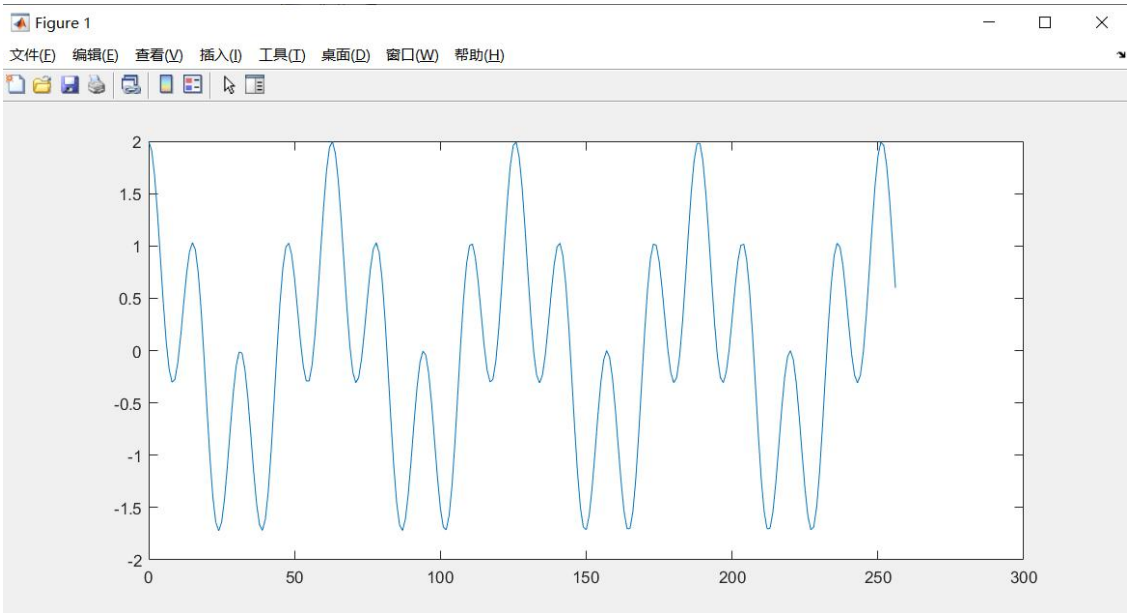


Dft 通过二重循环进行累加，所用时间呈 2 次幂上升，而 fft 上升幅度较小。

3. 通过滤波验证卷积、傅里叶变化

a) 选取课上 ppt 中的滤波例子进行编程验证，输入信号为$\cos 0.1t + \cos 0.4t$，设置采样频率 f_s=256 进行采样形成待操作的初始序列：

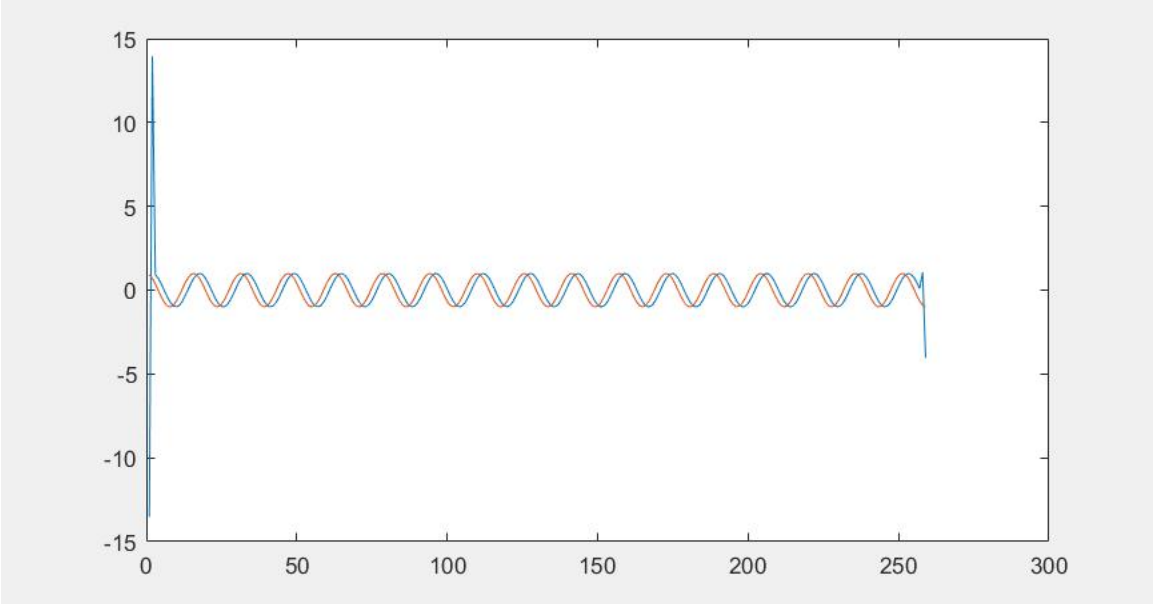
```
t=0:1:256; %采样步长
y = cos(0.1*t)+cos(0.4*t);%时域，原信号
plot(t,y);%原有信号
```



b) 时域卷积部分（希望过滤 cos0.4t，留下 cos0.1t）

初始化序列 h 为课上算出的卷积核，h=[-6.76195,13.456335,-6.76195]，将原序列 y 与该序列 h 进行卷积，得到 result1。result1 应该过滤掉了 cos0.4t，只剩 cos0.1t，故另外设置 y1=cos0.1t，将卷积结果和 y1 画在同一张图中，代码及结果如下（橘色线为 y1=cos0.1t，蓝色线为卷积结果 result1）：

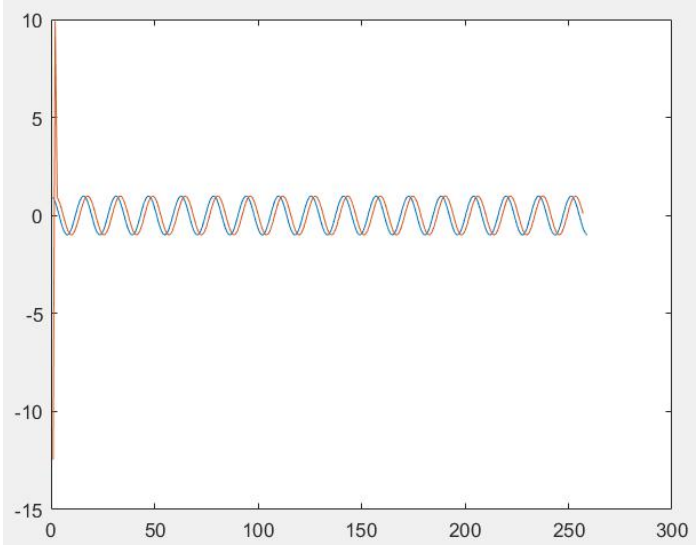
```
h=[-6.76195,13.456335,-6.76195];
result1=conv(y,h);%滤波结果！
```



c) 频域乘法部分

首先将两个序列 y、h 通过 fft 转换到频域，分别对应为 Y 和 H，计算 Y 与 H 的乘积，并将乘积做傅里叶逆变换得到 result3，代码及结果如下（蓝色线为 y1=cos0.1x，橘色线为频域乘积结果 result3）：

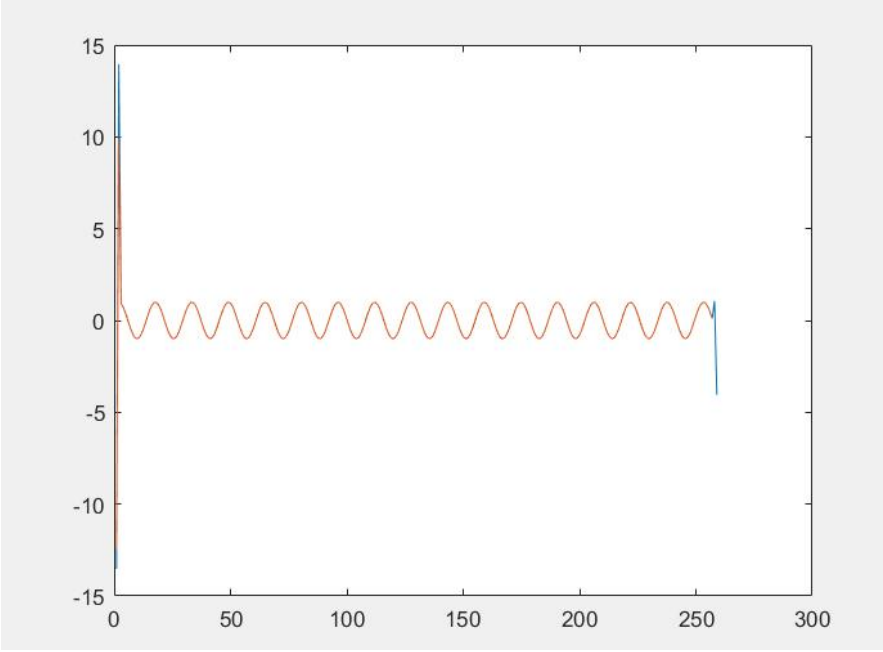
```
for i=4:257
    h(i)=0;
end
Y = fft(y);
H = fft(h);
result2=Y.*H;
result3=ifft(result2);
hold on;
plot(1:257,result3);
hold off;
```



注意：由于 y 与 h 的序列长度不同，傅里叶结果不能直接对位相乘，所以将 y 和 h 看作无限长序列，其他值即为 0，所以将 h 进行补零操作，使两个序列等长，后进行乘法操作。

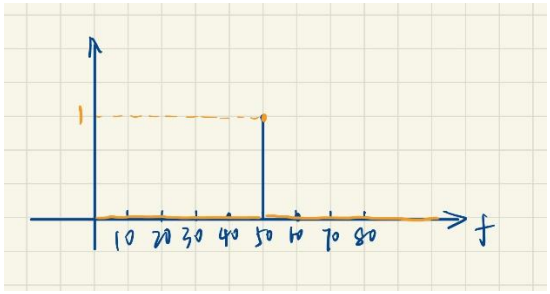
d) 验证时域的卷积等价于频域上的乘法

将 result1 和 result3 和理论结果 y1 画在同一张图里，发现两条线重合，成功验证：



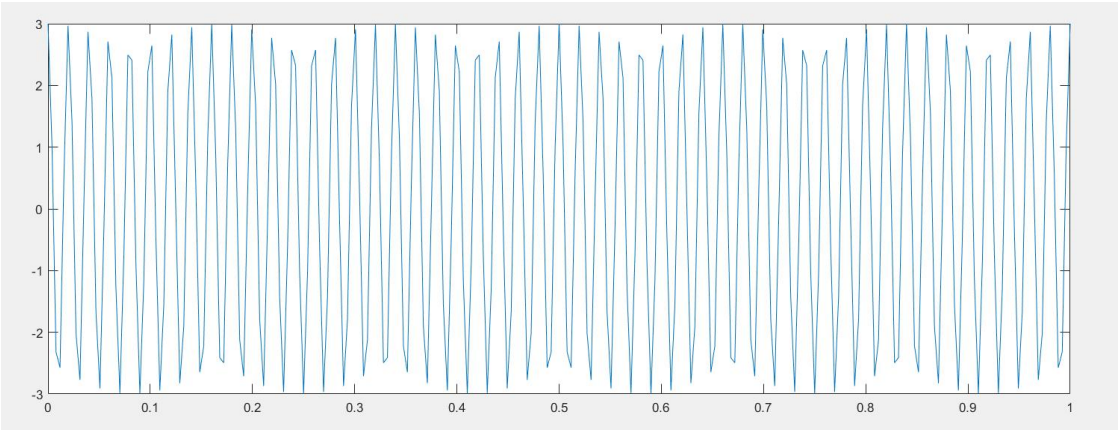
4. 对经典信号 cos10πt 进行频谱分析

a) 理想频域图（仅在频率为 50 时不为 0，其余频率连续均为 0）



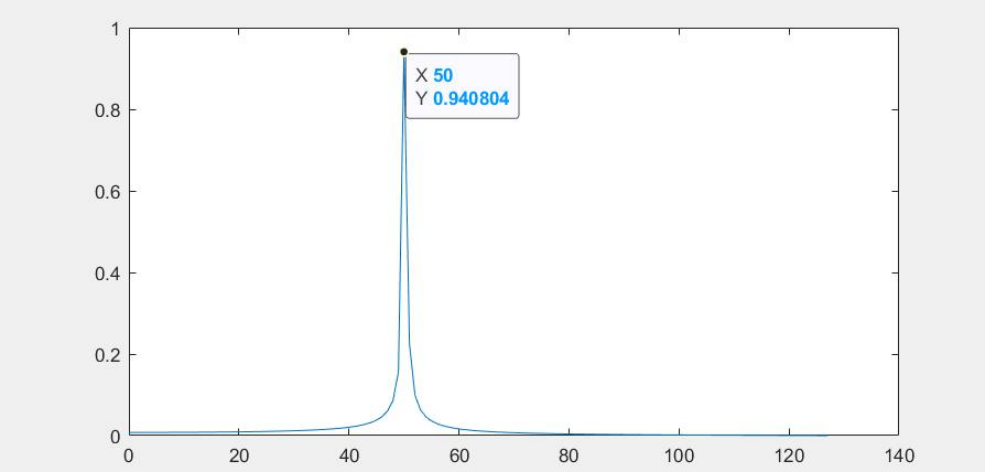
b) 验证过程

选取 256 个点，并以 f_s=256 的采样频率进行采样，构建有限长序列，得到信号图像如下：



通过 fft 实现傅里叶变换，根据资料所得频谱的幅值大小与选取的 N 有关，且变换后幅值的大小需要除以$\frac{N}{2}$后才是真实的幅值。

有效频率仅在 0~$\frac{fs}{2}$范围内，所以通过 plot(f(1:N/2),abs(Y(1:N/2)))只显示有效频率范围：



仅在频率为 50 时有峰值，与原信号 cos100πt 相符合。

代码如下：

```
t=0:1/256:1; %采样步长
y = cos(2*pi*50*t);
N=length(t); %样点个数
plot(t,y);%原有信号

fs=256;%采样频率
df=fs/(N-1);%分辨率
f=(0:N-1)*df;%每点的频率
Y = fft(y(1:N))/(N/2);%真实的幅值

figure(2)
plot(f(1:N/2),abs(Y(1:N/2)));
```