### 信号与系统实验报告

名 称： 快速傅里叶变换算法探究及应用

学 院： 计算机科学与工程学院、软件学院、人工智能学院

专 业： 计算机科学与技术

学 号： 09019106

姓 名： 牟倪

日期： 2021 年 6 月 2 日

1. 实验目的
2. 加深对快速傅里叶变换的理解。
3. 熟悉并掌握按时间抽取FFT算法的程序编制。
4. 了解应用FFT进行信号分析中可能出现的问题，如混淆、泄露等，以便在实际应用中正确应用FFT。
5. 实验任务
6. 完成实验内容全部题目，分析解决调试代码过程中出现的问题。
7. 认真完成本次实验小结，思考快速傅里叶变换的原理和算法及其应用。
8. 主要设备、软件平台
9. 硬件：计算机
10. 软件：Matlab
11. 实验内容
12. 参照“按时间抽取法FFT-基2”算法结构，编写相应的FFT程序*myFFT*()。

我的思路：

按时间抽取的基2FFT算法可以借助递归的思想进行实现。主要步骤如下：

1. 将x(n)按奇数位和偶数位分为两组：x1(r)=x(2r), x2(r)=x(2r+1)；
2. 对偶数序列x1和奇数序列x2进行FFT，此处可以递归调用；
3. 计算x(n)的FFT：

其中k满足 。

递归的边界条件是什么呢？当输入序列x(n)的长度N=2，即输入序列只含两点时，可以直接计算FFT序列。公式如下：

这就是递归的边界条件。

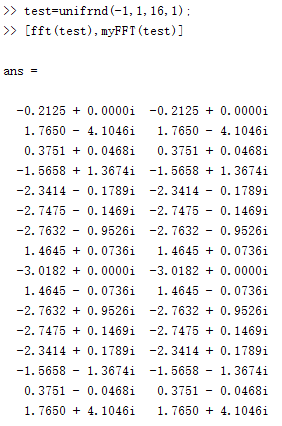
根据该递归逻辑编写程序，可以得到有限长序列x(n)的FFT序列X(n)。

代码：

1. function [X]=myFFT(x)
2. % 输入为N点有限长序列x（列向量），输出为其离散傅里叶变换X（列向量），要求N为2的幂
3. % 使用按时间抽取的FFT算法
4. N=length(x);X=zeros(N,1);
5. if N==2 % 递归边界条件
6. X(1)=x(1)+x(2);X(2)=x(1)-x(2);return;
7. end
8. x1=zeros(N/2,1);x2=x1;
9. for r=0:N/2-1
10. x1(r+1)=x(2\*r+1); % 偶数位
11. x2(r+1)=x(2\*r+1+1); % 奇数位
12. end
13. X1=myFFT(x1);X2=myFFT(x2); % 递归调用！
14. W=exp(-1i\*2\*pi/N);
15. for k=0:N/2-1
16. X(k+1)=X1(k+1)+W^k\*X2(k+1); % 计算X
17. X(k+N/2+1)=X1(k+1)-W^k\*X2(k+1);
18. end
19. end

正确性验证：

将myFFT()与Matlab内置函数fft()进行比较。如下图所示：



可以看出，myFFT()的输出与fft()完全一致，myFFT()的实现是正确的。

1. 用所编写的myFFT()分析信号



* + 1. 信号频率，采样点数，采样间隔
    2. 信号频率，采样点数，采样间隔
    3. 信号频率，采样点数，采样间隔
    4. 信号频率，采样点数，采样间隔
    5. 将信号④后补全32个0，完成64点FFT

要求：

记录各种情况下的X(k)值，绘制频谱图并对结果分析讨论，说明参数的变化对信号频谱产生的影响；频谱只需绘制幅度频谱，归一化处理；

程序需提供人机交互模式(控制台/图形窗口均可)；提供是否补零输入选项；提供参数输入功能；

打印myFFT()源程序，标注相关代码注释。

第一部分——信号分析：

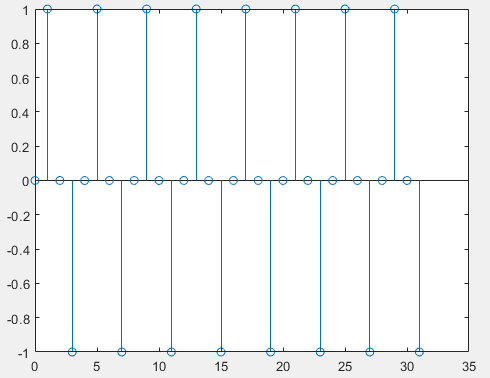
编写函数mySampling()得到采样的序列。函数的输入包括信号频率f、采样点数N和采样间隔T。代码如下：

1. function [x]=mySampling(f,N,T)
2. x=zeros(N,1);
3. for n=0:N-1
4. x(n+1)=sin(2\*pi\*f\*n\*T);
5. end
6. end

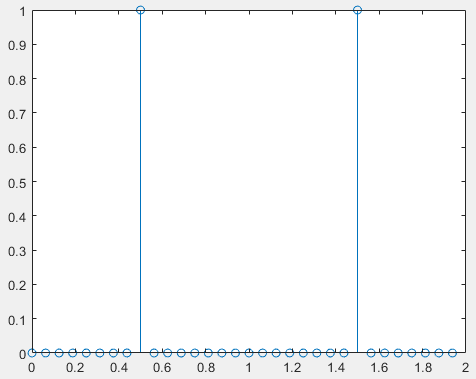
下面对采样得到的序列进行FFT：

序列①：

采样得到的x(n)如图所示：



FFT结果X(k)（归一化幅度频谱）如图所示：



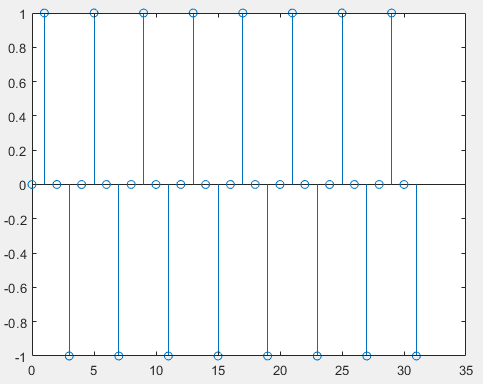
横轴的值为。

结果讨论：

采样间隔时，采样频率。信号的角频率。，满足Nyquist采样定理。采样点个数N=32，即对信号的频谱进行32等分的采样。同时，NT=0.16s，1/f=0.02s，窗口函数正好卡在周期整数倍，因此没有出现频谱泄露。

根据公式，原信号的频谱应在x>0出有一正向脉冲，在x<0的对称处有一与x>0脉冲大小相同的负向脉冲。考虑到FFT结果的圆周对称特性，上图所示的结果（归一化后）是合理的。

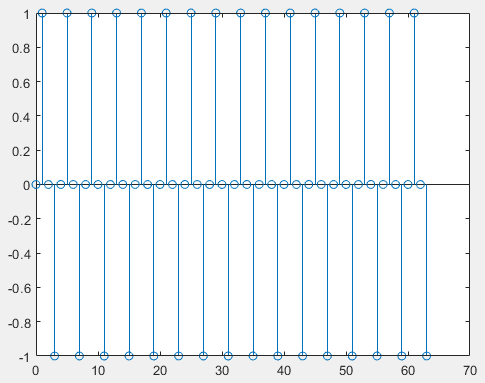
根据公式，由FFT结果还原原序列，如下图所示：



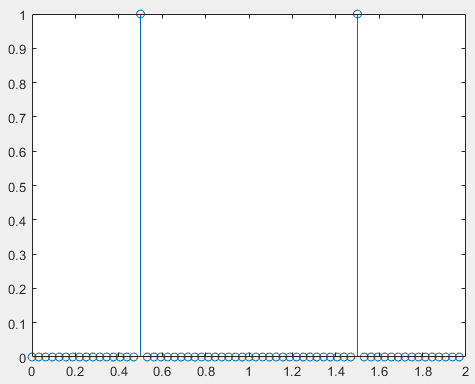
与原序列完全一致。这进一步验证了FFT程序编写的正确性。

序列②：

采样得到的x(n)如图所示：



FFT结果X(k) （归一化幅度频谱）如图所示：



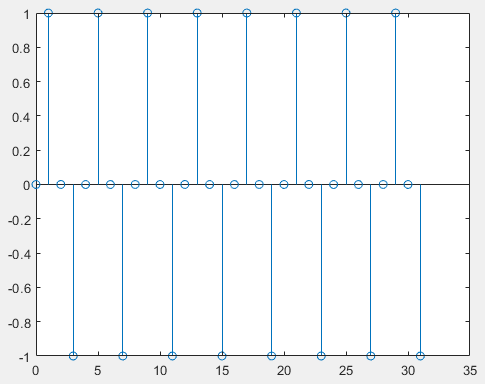
横轴的值为。

结果讨论：

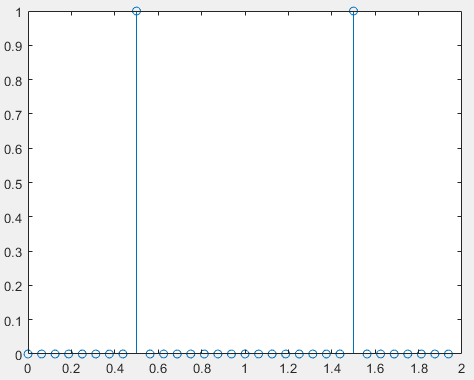
采样间隔时，采样频率。信号的角频率。，满足Nyquist采样定理。采样点个数N=64，即对信号的频谱进行64等分的采样。同时，NT=0.32s，1/f=0.02s，窗口函数正好卡在周期整数倍，因此没有出现频谱泄露。

序列③：

采样得到的x(n)如图所示：



FFT结果X(k) （归一化幅度频谱）如图所示：



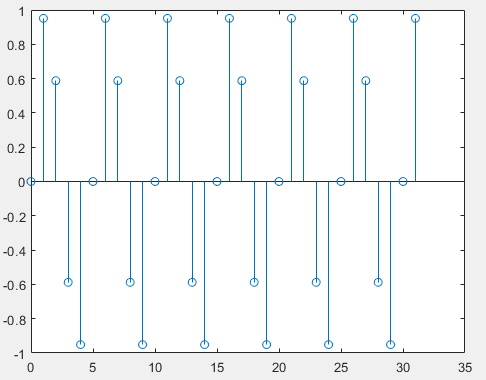
横轴的值为。

结果讨论：

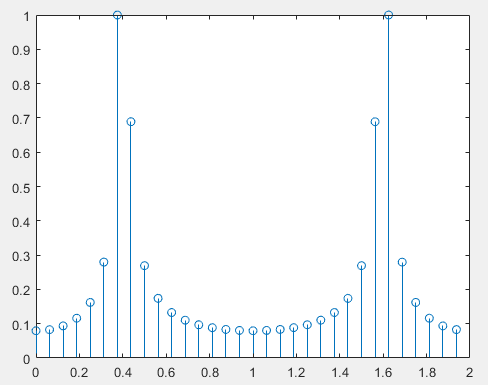
采样间隔时，采样频率。信号的角频率。，满足Nyquist采样定理。采样点个数N=32，即对信号的频谱进行32等分的采样。同时，NT=0.08s，1/f=0.01s，窗口函数正好卡在周期整数倍，因此没有出现频谱泄露。

序列④：

采样得到的x(n)如图所示：



FFT结果X(k) （归一化幅度频谱）如图所示：

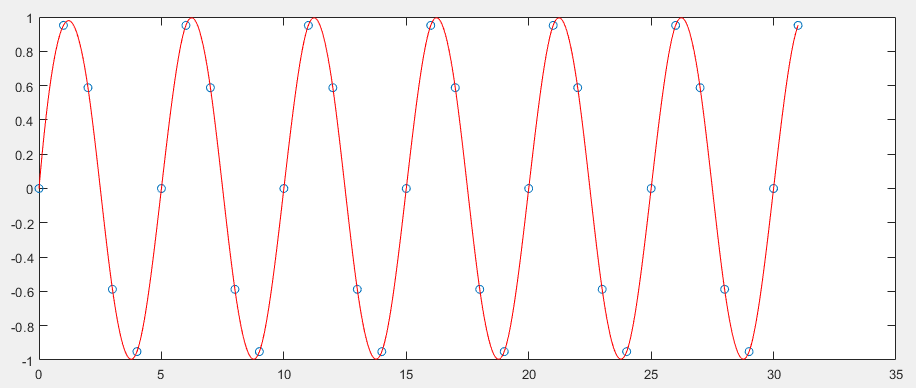


横轴的值为。

结果讨论：

采样间隔时，采样频率。信号的角频率。，不满足Nyquist采样定理，出现欠采样的情况。当对X(k)进行IFFT时，会出现频谱混叠的情况，因此无法根据FFT频谱还原x(n)序列。

事实上，欠采样的序列本身即无法代表原信号的特性。按照欠采样序列对原信号进行推测，可能会得到如下图所示的结果：

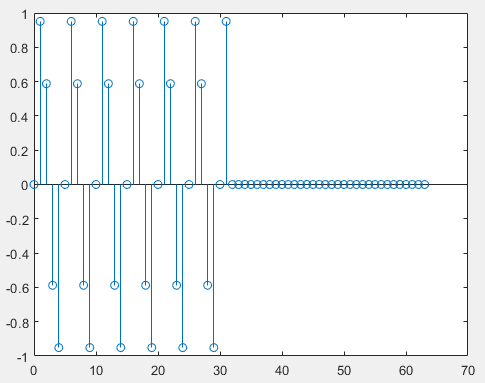


频率比原信号低很多。这显然与原信号不一致。

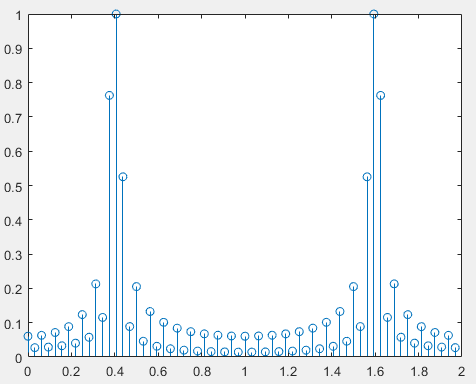
同时，NT=0.0384s，1/f=0.001s，1/f不能整除NT，窗口函数没有卡在周期整数倍。因此，除欠采样外，信号也出现了频谱泄露情况。

序列⑤：

采样得到的x(n)如图所示：



FFT结果X(k) （归一化幅度频谱）如图所示：



结果讨论：

采样间隔时，采样频率。信号的角频率。，不满足Nyquist采样定理，出现欠采样的情况。当对X(k)进行IFFT时，会出现频谱混叠的情况。对序列进行补0拓展没有改变欠采样的实质，因此仍无法根据FFT频谱还原x(n)序列。

同时，对序列进行补0拓展也没有改变频谱泄露的实质，频谱泄露的情况仍然存在。

第二部分——控制台人机交互模式：

编写函数myFFTinteract()。输入为原序列x。该函数可以接受长度N不为2的幂的序列，并按照交互内容将其补全。在函数的执行过程中，若发现序列长度N不为2的幂，则询问是否补零，得到指示是否补0的变量zerofill。若不补零，输入补全序列的数值fillvalue。对序列进行补全，然后进行FFT。输出为FFT结果X。

程序代码如下：

1. function X=myFFTinteract(x)
2. x0=x;N=length(x);
3. while(bitand(N,1)==0&&N>1)
4. N=N/2;
5. end
6. if N~=1
7. zerofill=input("输入序列长度不是2的幂，请输入是(1)否(0)补零：");
8. if zerofill==0
9. fillvalue=input("请输入补全的数值：");
10. N=length(x);
11. x0=[x0;ones(2^ceil(log(N)/log(2))-N,1).\*fillvalue];
12. end
13. end
14. X=myFFT(x0);
15. end

正确性验证：

如图所示：

|  |  |
| --- | --- |
| E:\temp\1622627294(1).png | E:\temp\1622627330(1).png |

myFFTinteract()可以判断序列长度是否为2的幂，并按照交互内容将序列长度补为2的幂。程序功能完整，实现正确。

**myFFT()代码已经附在前文。**

1. 实验小结
2. Matlab中用1i表示虚数单位；
3. 递归是很有力的工具，基于递归编写的代码思路清晰、易于调试；
4. Matlab函数支持递归调用；
5. Matlab支持复数的加减乘除运算；
6. Matlab中对复数取模使用abs()，取实部使用real()，取虚部使用imag()，取共轭使用conj()；
7. Matlab中stem函数支持stem(X,Y)形式；
8. Matlab中提供支持位运算的函数，如bitand()等。
9. Matlab中，可以使用input()函数读控制台输入，得到符合Matlab数据格式的数值、矩阵或字符串。