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

专 业：人工智能

学 号：58119304

姓 名：朱启鹏

日期：2021 年 6 月 1 日

一、实验目的

1. 加深对快速傅里叶变换的理解。
2. 熟悉并掌握按时间抽取 FFT 算法的程序编制。
3. 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。

二、实验任务

1. 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。
2. 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。

三、主要设备、软件平台

1. 硬件：计算机
2. 软件：Matlab

四、实验内容

1. 参照“按时间抽取法 FFT-基 2”算法结构，编写相应的 FFT 程序 *myFFT*()。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。

代码实现如下：

```
1. import numpy as np
2. import math
3. import argparse
4. import matplotlib.pyplot as plt
5. #实现复数指数函数
6. def cexp(c):
7. assert isinstance(c,complex)
8. return math.exp(abs(c))*c.real/abs(c)+(math.exp(abs(c))*c.imag/abs(c))*1
j if c!=0 else 1
9. #实现阶跃函数
10. def u(x):
11. return 1 if x >= 0 else 0
12. #目标函数
13. def target(n,f,T,N):
14. return math.sin(2*math.pi*f*n*T)*(u(n)-u(n-N))
```

```
#实现 FFT 变换所需初始分组
def get_init(samples):
if len(samples)==2:
return samples
else:
ls = [ samples[i] for i in range(0,int(len(samples)/2))]
rs = [samples[i] for i in range(int(len(samples)/2),len(samples))]
ls = get_init(ls)
rs = get_init(rs)
result = []
for i in range(int(len(samples)/2)):
result.append(ls[i])
result.append(rs[i])
return result
#快速傅里叶变换
def MyFFT(samples):
assert math.log(len(samples),2)==int(math.log(len(samples),2))
buffer =[ get_init(samples),[None for i in range(len(samples))]]
v = 0
group_size = 1
for round in range(int(math.log(len(samples),2))):
group_size*=2
for group_f in range (0,len(samples),group_size):
for _ in range(int(group_size/2)):
z = cexp(-
1j*(2*_*math.pi)/group_size)*buffer[v][group_f+_+int(group_size/2)]
buffer[(v+1)%2][group_f+_]=buffer[v][group_f+_]+z
buffer[(v+1)%2][group_f+_+int(group_size/2)]=buffer[v][group
_f+_]-z

v=(v+1)%2

return buffer[v]

def main(args):

sample = [target(i,args.f,args.T,args.N) for i in range(args.N)]
a = input("Full fill zeros?[y/N(default)]")
if a == 'y' or a=='Y' :
for _ in range(args.N):
sample.append(0)

result = MyFFT(sample)
```

```
result = [abs(_) for _ in result]
fig,ax = plt.subplots()
ax.scatter(np.arange(0,int(len(result)/2),1),result[0:int(len(result)/2)]
)
fig.savefig('FFT result with parameter f={} N={} T={} {}.jpg'.format(arg
s.f,args.N,args.T,('filling zeros' if (a == 'y' or a=='Y') else ''))
plt.show()


return None

if __name__ == '__main__':

argparser=argparse.ArgumentParser(description='Display FFT result of the
origin function in terminal')

argparser.add_argument('--f',type=float,help='frequency')
argparser.add_argument('--N',type=int,help='sampling count')
argparser.add_argument('--
T',type=float,help='interval of sampling points')

args=argparser.parse_args()

main(args)
```

调用示例

2. 用所编写的 myFFT()分析信号

$$x(n)=\sin(2\pi f\,n)[u(n)-u(n-N\bullet T)],\quad -\infty<n<\infty$$

① 信号频率 $f=50Hz$，采样点数 $N=32$，采样间隔 $T=0.005s$
② 信号频率 $f=50Hz$，采样点数 $N=64$，采样间隔 $T=0.005s$
③ 信号频率 $f=100Hz$，采样点数 $N=32$，采样间隔 $T=0.005s$
④ 信号频率 $f=1000Hz$，采样点数 $N=32$，采样间隔 $T=0.012s$
⑤ 将信号④后补全 32 个 0，完成 64 点 FFT

结果展示：

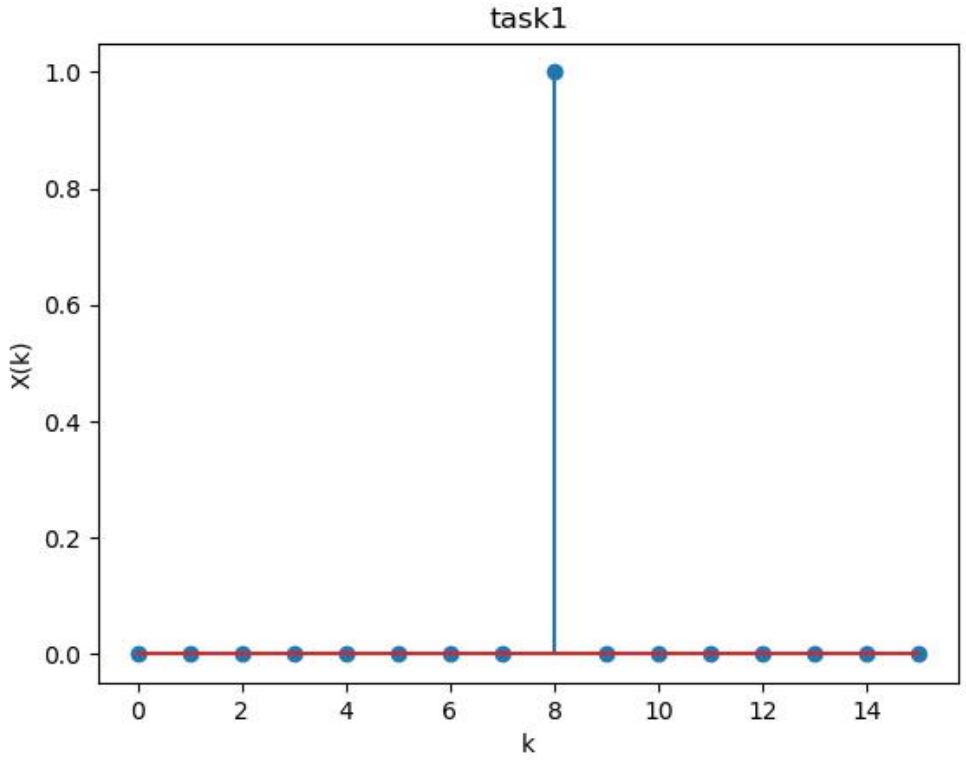
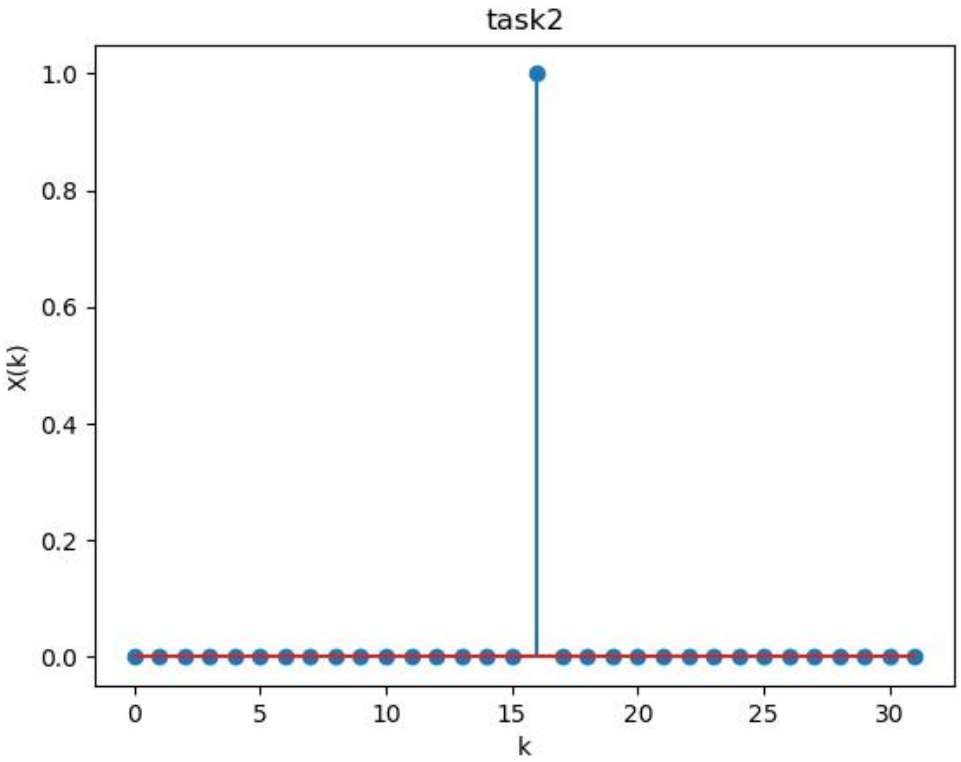
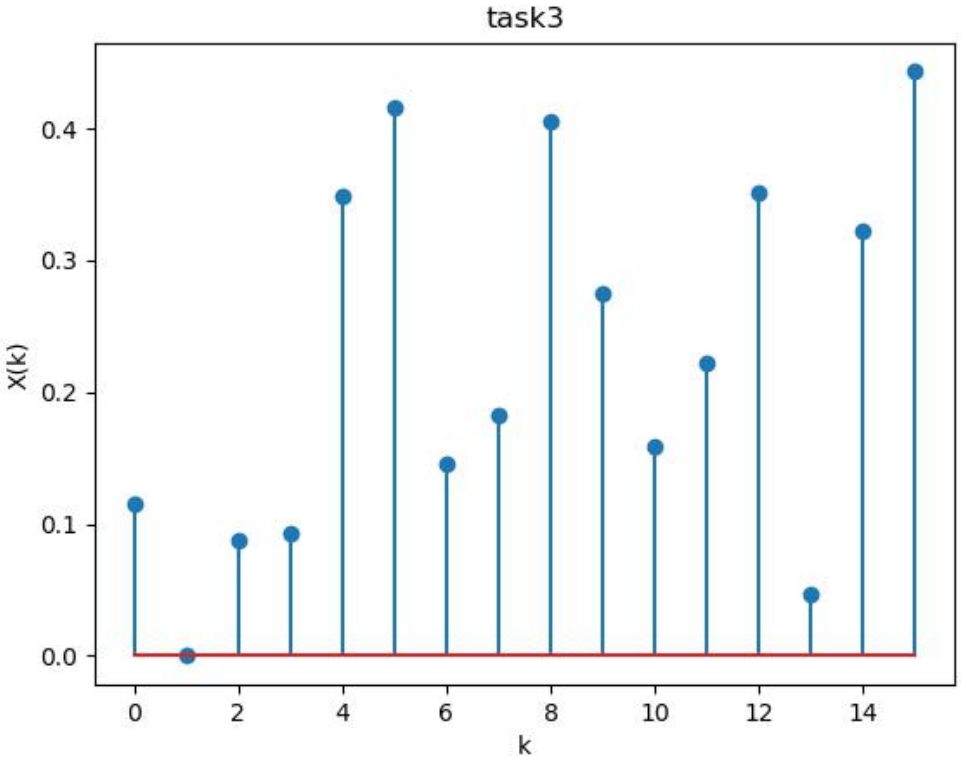


图 1：f = 50, N = 32, T = 0.005



图二：f = 50, N = 64, T = 0.005



图三：f = 100, N = 32, T = 0.005

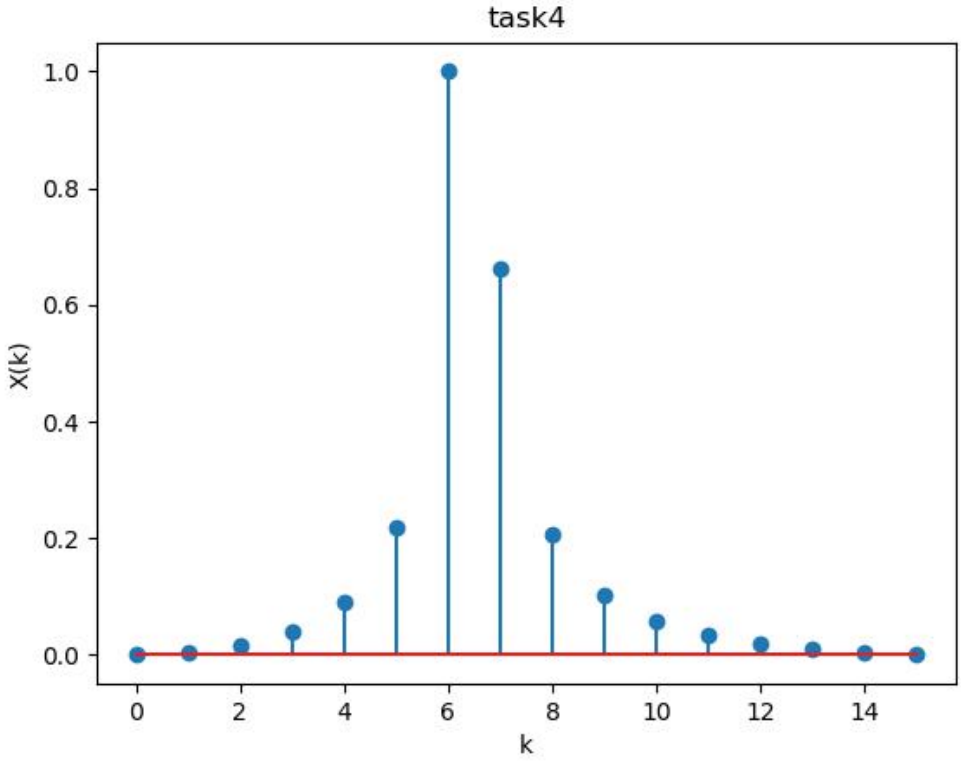


图 4：f = 100, N = 32, T = 0.005

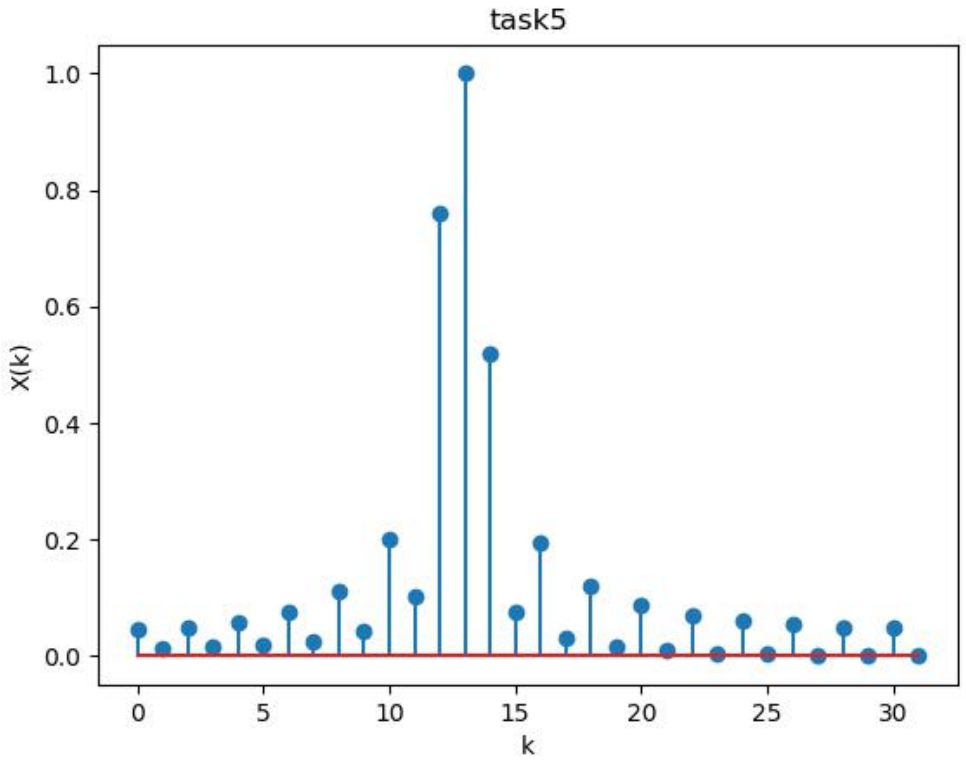


图 5：f=1000 N=32 T=0.005 补零至 64 条数据

实验分析：对于 f 的不同，其具体影响了快速傅里叶变换后频谱的形状；而 N 的不同影响了抽样的间隔，其影响了快速傅里叶变换后频谱的形状。所以任务一和任务二，在频率相同的情况下，形状都相同，而仅仅是抽样的密度不同。实验一三四，在 N 相同的情况下，由于频率不同，形成的频谱则是千差万别，但是它们抽样的密度确实相同的。任务四和任务五我们可以发现，补零的效果并不会对原本信号产生很大的影响，仍旧极大的保留了原有信号的特征，所以两者的形状几乎相同，任务五相对于任务四更多了一些低频的部分，这些部分可以用滤波器进行滤除。

五、实验小结

这次实验，通过使用 python 对快速傅里叶变换的实现，使我对 FFT 有了更深的理解。通过不同任务的编写，我对傅里叶变换中信号的采样频率与信号的周期对快速傅里叶变换的影响有了更深的理解。并且，我对信号与系统在工程上的运用，有了初步的认识，这对我之后的学习也有很大的好处。