### 实验一 快速傅里叶变换验证与实现

##### 一、实验目的

1、在理论学习的基础上，通过本实验加深对离散傅里叶变换的理解。

2、熟悉并掌握按时间抽取法编写快速傅里叶变换（FFT）算法的程序。

3、了解应用FFT进行信号频谱分析过程中可能出现的问题，例如频谱混淆、泄漏、栅栏效应等，以便在实际中正确使用FFT算法进行信号处理。

##### 二、实验内容

①用MATLAB自带的FFT函数，分别在以下情况对sin2πFt绘制时域、频域结果图，讨论所得的结果是否存在频谱混淆、泄漏、栅栏效应等问题：

a)信号频率F＝50Hz，采样点数N=32，采样间隔T=0.000625s；

b)信号频率F＝50Hz，采样点数N=32，采样间隔T=0.005s；

c)信号频率F＝50Hz，采样点数N=32，采样间隔T=0.0046875s；

d)信号频率F＝50Hz，采样点数N=32，采样间隔T=0.004s；

e)信号频率F＝50Hz，采样点数N=64，采样间隔T=0.000625s；

f)信号频率F＝250Hz，采样点数N=32，采样间隔T=0.005s；

g)将c)中信号后补32个0，做64个样点的FFT，并与直接采样64个点做FFT的结果进行对比。

②仔细理解教材第四章中的图4.22，参照流程图，自行编写出基-2按时间抽取FFT的MATLAB程序代码。并用实验内容①中正弦信号a)~g)的各种情况验证证明自编程序的有效性。

③（选做者加分）自行编写出基-2按频率抽取FFT的MATLAB程序代码。并用实验内容①中正弦信号验证证明自编程序的有效性。

##### 三、实验思考题

1、在实验内容①的a)、b)、c)和d)情况中，正弦信号的初始相位对频谱图中的幅度特性是否有影响？为什么？

2、信号补零后做FFT是否可以提高信号频谱的分辨率？为什么？

##### 四、程序编写

###### 第一部分：（注：实际上compare\_N64的功能与新添加的图表8相同，但是考虑到严谨性故新添加了图表8，同时保留了compare\_N64的代码，其余部分同理）

function plot\_time\_freq(F, T, N, zero\_padding, compare\_N64)

% F: 信号频率

% T: 采样间隔

% N: 采样点数

% zero\_padding: 补零至2\*N个样点的开关

% compare\_N64: 开关，用于将补零和直接采样64个点进行比较

t = (0:N-1) \* T; % 生成时域采样点

x = sin(2\*pi\*F\*t); % 生成正弦信号

% 绘制时域图

figure;

subplot(2,2,1);

plot((0:N-1)\*T, x);

title(['时域信号 (F=', num2str(F), ' Hz, T=', num2str(T), ' s, N=', num2str(N), ')']);

xlabel('时间 (s)');

ylabel('幅值');

% 补零选项

if zero\_padding

x = [x, zeros(1, N)]; % 补零至2\*N样点

N = 2 \* N; % 更新样点数

end

% 计算 FFT

X = fft(x); % 计算 FFT

subplot(2,2,2);

stem((0:N-1)\*T, x);

title('采样后的离散信号');

xlabel('时间 (s)');

ylabel('幅值');

% 绘制频域图

subplot(2,2,3);

stem(0:N-1, abs(X)/N); % 归一化幅值

title('频域信号');

xlabel('k');

ylabel('幅值');

% 如果 compare\_N64 为真，进行与直接采样64个点的对比

if compare\_N64

% 直接采样64个点 (不补零)

N64 = 64;

t64 = (0:N64-1) \* T; % 生成时域采样点

x64 = sin(2\*pi\*F\*t64); % 生成正弦信号

X64 = fft(x64); % 计算64点FFT

% 绘制补零和直接采样的频域对比

subplot(2,2,4);

hold on;

stem(0:N64-1, abs(X64)/N64, 'DisplayName', '直接采样64点'); % 直接采样64点的频域

title('频域信号对比');

xlabel('k');

ylabel('幅值');

legend;

hold off;

end

end

% a) F=50Hz, N=32, T=0.000625s

plot\_time\_freq(50, 0.000625, 32, 0, 0);

% b) F=50Hz, N=32, T=0.005s

plot\_time\_freq(50, 0.005, 32, 0, 0);

% c) F=50Hz, N=32, T=0.0046875s

plot\_time\_freq(50, 0.0046875, 32, 0, 0);

% d) F=50Hz, N=32, T=0.004s

plot\_time\_freq(50, 0.004, 32, 0, 0);

% e) F=50Hz, N=64, T=0.000625s

plot\_time\_freq(50, 0.000625, 64, 0, 0);

% f) F=250Hz, N=32, T=0.005s

plot\_time\_freq(250, 0.005, 32, 0, 0);

% g) F=50Hz, N=32, T=0.0046875s, 信号补零至64个样点，并与直接采样64个点对比

plot\_time\_freq(50, 0.0046875, 32, 0, 0);

plot\_time\_freq(50,0.0046875,64,0,0);

###### 第二部分：

% 基-2按时间抽取的FFT算法，流程图版

function X = FFT\_time\_extract\_flowchart(x)

N = length(x);

X = x; % 初始化FFT输出为输入信号

% 步骤1: 位反转排序

X = bitrevorder(X);

% 步骤2: 循环处理FFT的每个阶段 (M = log2(N))

M = log2(N);

for m = 1:M

LE = 2^m; % 步骤3: LE = 2^M (分组的大小)

LE1 = LE / 2; % LE1是LE的一半

W = exp(-2i\*pi\*(0:LE1-1)/LE); % 步骤4: 旋转因子

% 步骤5: 处理每个分组

for j = 0:LE:N-1

for i = 1:LE1

ip = i + j; % I + LE1 -> IP

temp = X(ip+LE1) \* W(i); % A(IP)\*U -> T

X(ip+LE1) = X(ip) - temp; % A(I) + T -> A(IP)

X(ip) = X(ip) + temp; % A(IP) - T -> A(I)

end

end

end

end

% 比较自编FFT与MATLAB内置FFT结果

function compare\_fft(F, T, N, m, zero\_padding)

% 生成信号，带有初始相位m

t = (0:N-1) \* T;

x = sin(2\*pi\*F\*t + m);

% 绘制时域图

figure;

subplot(2,2,1);

plot((0:N-1)\*T, x);

title(['时域信号 (F=', num2str(F), ' Hz, T=', num2str(T), ' s, N=', num2str(N), ', 初始相位=', num2str(m), ')']);

xlabel('时间 (s)');

ylabel('幅值');

% 补零选项

if zero\_padding

x = [x, zeros(1, N)]; % 补零至2\*N样点

N = 2 \* N; % 更新样点数

end

% 自编FFT

X\_custom = FFT\_time\_extract\_flowchart(x); % 修改后的FFT函数

% MATLAB内置FFT

X\_builtin = fft(x);

subplot(2,2,2);

stem((0:N-1)\*T, x);

title('采样后的离散信号');

xlabel('时间 (s)');

ylabel('幅值');

% 绘制频域比较图

subplot(2,2,3);

stem(0:N-1, abs(X\_custom)/N);

title('自编 FFT 频域信号');

xlabel('k');

ylabel('幅值');

subplot(2,2,4);

stem(0:N-1, abs(X\_builtin)/N);

title('MATLAB 内置 FFT 频域信号');

xlabel('k');

ylabel('幅值');

% 比较结果

diff = max(abs(X\_custom - X\_builtin));

disp(['最大差异: ', num2str(diff)]);

end

% 主函数，添加参数输入选择

function main\_fft\_time()

disp('请选择参数输入方式：');

disp('1. 使用默认参数');

disp('2. 自定义输入参数');

choice = input('输入您的选择 (1 或 2): ');

if choice == 1

% 使用默认参数

run\_default\_tests\_time();

elseif choice == 2

% 自定义参数输入

F = input('信号频率F/hz= ');

N = input('采样点数N= ');

T = input('采样时间间隔T/s= ');

m = input('初始相位= ');

flag = input('是否补零（1为yes/0为no）= ');

compare\_fft(F, T, N, m, flag);

else

disp('无效的选择');

end

end

% 默认参数测试

function run\_default\_tests\_time()

% a) F=50Hz, N=32, T=0.000625s, 初始相位=0

compare\_fft(50, 0.000625, 32, 0, 0);

% b) F=50Hz, N=32, T=0.005s, 初始相位=0

compare\_fft(50, 0.005, 32, 0, 0);

% c) F=50Hz, N=32, T=0.0046875s, 初始相位=0

compare\_fft(50, 0.0046875, 32, 0, 0);

% d) F=50Hz, N=32, T=0.004s, 初始相位=0

compare\_fft(50, 0.004, 32, 0, 0);

% e) F=50Hz, N=64, T=0.000625s, 初始相位=0

compare\_fft(50, 0.000625, 64, 0, 0);

% f) F=250Hz, N=32, T=0.005s, 初始相位=0

compare\_fft(250, 0.005, 32, 0, 0);

% g) F=50Hz, N=32, T=0.0046875s, 初始相位=0, 补零

compare\_fft(50, 0.0046875, 32, 0, 1);

compare\_fft(50, 0.0046875, 64, 0, 0);

end

% 运行按时间抽取FFT测试

main\_fft\_time();

###### 第三部分：

% 基-2按频率抽取的FFT算法

function X = FFT\_freq\_extract(x)

N = length(x);

X = bit\_reverse\_order(x); % 二进制反转顺序排列

stages = log2(N); % FFT 需要的阶段数

% 基-2按频率抽取计算过程

for s = 1:stages%蝶形运算

m = 2^s;

W\_m = exp(-2i\*pi\*(0:m/2-1)/m); % 旋转因子

for k = 0:(N/m - 1)%阶段处理

for j = 1:m/2%合并操作

t = W\_m(j) \* X(k\*m + j + m/2);

u = X(k\*m + j);

X(k\*m + j) = u + t;

X(k\*m + j + m/2) = u - t;

end

end

end

end

% 二进制反转顺序排列

function X = bit\_reverse\_order(x)

N = length(x);

X = zeros(1, N);

num\_bits = log2(N);

for n = 0:N-1

reversed\_n = bit\_reverse(n, num\_bits);

X(reversed\_n + 1) = x(n + 1); % MATLAB 索引从 1 开始

end

end

% 二进制反转

function r = bit\_reverse(n, num\_bits)

r = 0;

for i = 0:num\_bits-1

r = bitor(r, bitshift(bitand(n, 1), num\_bits - 1 - i));

n = bitshift(n, -1);

end

end

% 比较按频率抽取 FFT 和 MATLAB 内置 FFT 结果

function compare\_fft\_freq\_extract(F, T, N, m, zero\_padding)

% 生成信号，带有初始相位m

t = (0:N-1) \* T;

x = sin(2\*pi\*F\*t + m);

% 补零选项

if zero\_padding

x = [x, zeros(1, N)]; % 补零至2\*N样点

N = 2 \* N; % 更新样点数

end

% 自编按频率抽取的 FFT

X\_custom\_freq = FFT\_freq\_extract(x);

% MATLAB 内置 FFT

X\_builtin = fft(x);

% 绘制时域图

figure;

subplot(2,2,1);

plot((0:N-1)\*T, x);

title(['时域信号 (F=', num2str(F), ' Hz, T=', num2str(T), ' s, N=', num2str(N), ', 初始相位=', num2str(m), ')']);

xlabel('时间 (s)');

ylabel('幅值');

subplot(2,2,2);

stem((0:N-1)\*T, x);

title('采样后的离散信号');

xlabel('时间 (s)');

ylabel('幅值');

% 绘制频域比较图

subplot(2,2,3);

stem(0:N-1, abs(X\_custom\_freq)/N); % 将横坐标修改为采样点索引

title('自编 按频率抽取 FFT 频域信号');

xlabel('采样点');

ylabel('幅值');

subplot(2,2,4);

stem(0:N-1, abs(X\_builtin)/N);

title('MATLAB 内置 FFT 频域信号');

xlabel('k');

ylabel('幅值');

% 比较结果

diff = max(abs(X\_custom\_freq - X\_builtin));

disp(['按频率抽取 FFT 与 MATLAB FFT 最大差异: ', num2str(diff)]);

end

% 主函数，添加参数输入选择

function main\_fft\_freq()

disp('请选择参数输入方式：');

disp('1. 使用默认参数');

disp('2. 自定义输入参数');

choice = input('输入您的选择 (1 或 2): ');

if choice == 1

% 使用默认参数

run\_default\_tests\_time();

elseif choice == 2

% 自定义参数输入

F = input('信号频率F/hz= ');

N = input('采样点数N= ');

T = input('采样时间间隔T/s= ');

m = input('初始相位= ');

flag = input('是否补零（1为yes/0为no）= ');

compare\_fft\_freq\_extract(F, T, N, m, flag);

else

disp('无效的选择');

end

end

% 默认参数测试

function run\_default\_tests\_time()

% a) F=50Hz, N=32, T=0.000625s, 初始相位=0

compare\_fft\_freq\_extract(50, 0.000625, 32, 0, 0);

% b) F=50Hz, N=32, T=0.005s, 初始相位=0

compare\_fft\_freq\_extract(50, 0.005, 32, 0, 0);

% c) F=50Hz, N=32, T=0.0046875s, 初始相位=0

compare\_fft\_freq\_extract(50, 0.0046875, 32, 0, 0);

% d) F=50Hz, N=32, T=0.004s, 初始相位=0

compare\_fft\_freq\_extract(50, 0.004, 32, 0, 0);

% e) F=50Hz, N=64, T=0.000625s, 初始相位=0

compare\_fft\_freq\_extract(50, 0.000625, 64, 0, 0);

% f) F=250Hz, N=32, T=0.005s, 初始相位=0

compare\_fft\_freq\_extract(250, 0.005, 32, 0, 0);

% g) F=50Hz, N=32, T=0.0046875s, 初始相位=0, 补零

compare\_fft\_freq\_extract(50, 0.0046875, 32, 0, 1);

end

% 运行按时间抽取FFT测试

main\_fft\_freq();

##### 五、实验结果

###### 第一部分：(注：第八张图表为直接采样64个点，下同)

图表, 直方图

描述已自动生成图表, 图示

描述已自动生成图表

描述已自动生成图表, 图示

描述已自动生成图表, 直方图

描述已自动生成图表, 图示

描述已自动生成图表

描述已自动生成图表

描述已自动生成

###### 第二部分：

图形用户界面, 图表, 直方图

描述已自动生成图表, 图示

描述已自动生成图表

描述已自动生成图表, 图示

描述已自动生成图表, 直方图

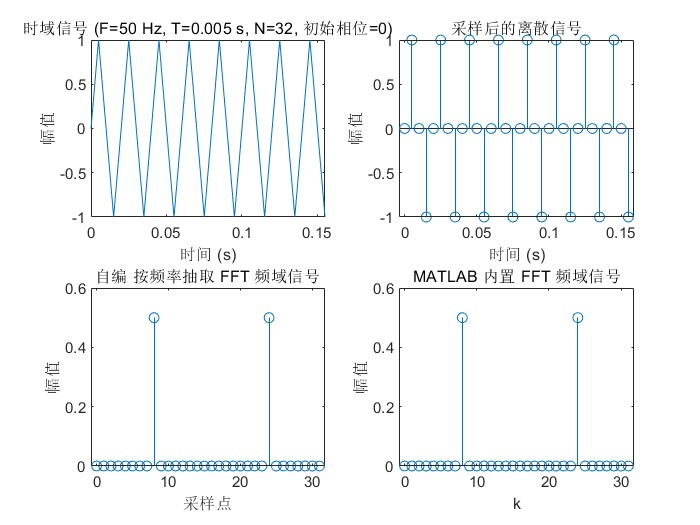
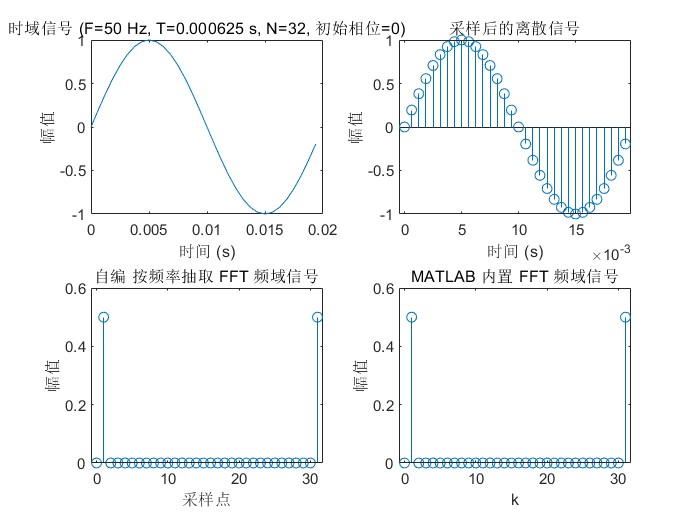
描述已自动生成图表, 图示

描述已自动生成图形用户界面, 图表, 图示

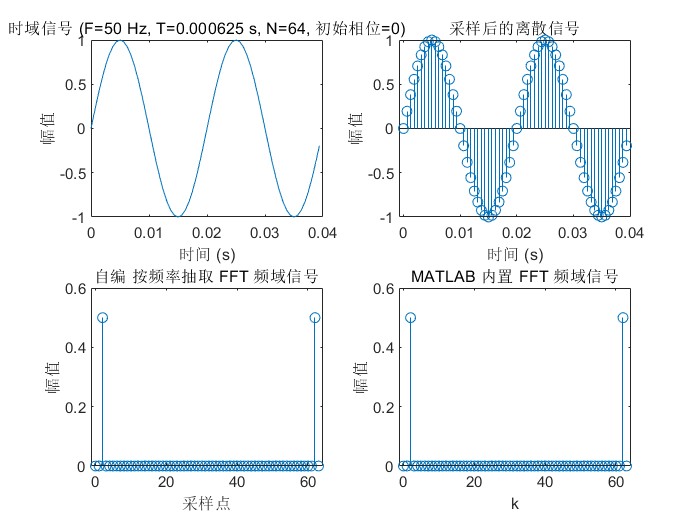
描述已自动生成图表

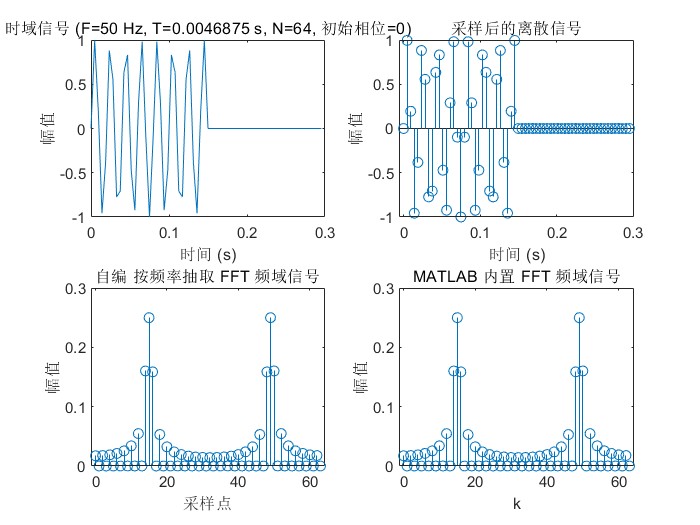
描述已自动生成

###### 第三部分：

图表

描述已自动生成图表, 图示

描述已自动生成图表, 图示

描述已自动生成

###### 结果分析（深入讨论，说明参数的变化对信号频谱产生哪些影响）：

**1. 信号频率F的影响**

信号频率F是信号的基本特性，它决定了信号的周期。如果信号频率过高，而采样频率不足，会导致频谱混叠。频率越高，采样频率必须相应增加，才能准确反映信号的频率成分。

* **高频信号和低采样频率**：如果信号频率较高，而采样频率​不足（即未满足采样定理，采样频率fs小于信号频率的两倍），就会在频谱中出现混叠现象，即频谱的能量映射到错误的频率位置。
* **信号频率接近 Nyquist 频率**：当信号频率接近 Nyquist 频率（fs/2）时，频谱会出现泄漏效应（leakage），使得频谱能量扩散。

**2. 采样点数N的影响**

采样点数N决定了 FFT 的计算点数，并影响频谱的分辨率。

* **采样点数较少**：当N较小时，频谱的分辨率较低，容易出现 *频谱泄漏* 现象。同时，FFT 频率分辨率Δf由Δf=1/NT决定，当N较小时，频谱的频率步长变大，无法精细地分辨不同的频率成分。
* **采样点数较多**：当N增加时，频谱的分辨率提高，频率步长变小，频谱更清晰，可以更好地观察细节。此外，采样点数增加也有助于减小频谱泄漏。

**3. 采样时间间隔T的影响**

采样时间间隔T影响采样频率fs，因此直接影响 FFT 频谱的范围和 Nyquist 频率。

* **采样频率高**（即T小）：高采样频率能够捕捉到更多的高频信息，避免混叠问题。同时也会使频谱的频率范围变大（从0到fs/2）。
* **采样频率低**（即T大）：如果采样频率过低，会导致频谱混叠现象。此外，较低的采样频率会减少可观察的频率范围，可能会导致信号的频率分量超出 Nyquist 频率。

**4. 初始相位m的影响**

初始相位m对频谱的幅度没有直接影响，但会影响频谱的相位部分。

在进行 FFT 变换时，FFT 的结果是复数，包含幅度和相位。初始相位 m*m* 只会影响频谱的相位特性，而不会影响频谱的幅度。如果我们只观察幅度谱，则不同的初始相位导致的频谱变化不大。但在信号重构或观察相位谱时，初始相位会有较大的影响。

**5. 是否补零（flag） 的影响**

补零的操作是对信号进行零填充以增加信号的长度。在 MATLAB 中，零填充后执行 FFT 时，采样点数 N*N* 增加，因此影响频谱的视觉分辨率。

* **补零的效果**：补零能够增加频谱的频率采样点，使得频率步长Δf变小（即频谱更平滑）。但补零并不会增加信号的实际频率信息。

##### 六、回答思考题

###### 1、在实验内容①的a)、b)、c)和d)情况中，正弦信号的初始相位对频谱图中的幅度特性是否有影响？为什么？

**理论上**，正弦信号的初始相位*φ***不会影响频谱图中的幅度特性**。因为 FFT 的幅度是频谱复数的模长，它反映的是信号的能量分布，初始相位只会影响频谱的相位部分而不是幅度。

但是，在**有限点数采样**的情况下（如实验中的 N = 32 点），初始相位可能会影响到**频谱泄漏**或**栅栏效应**的表现，从而间接影响幅度特性。具体而言，如果采样点的位置没有与信号的周期完美对齐，初始相位会导致信号在 FFT 分析中的能量扩散（即泄漏）到其它频率成分上，这可能使得幅度特性看起来有所变化。

###### 2、信号补零后做FFT是否可以提高信号频谱的分辨率？为什么？

补零的主要作用是增加 FFT 结果的频谱分辨率，即使得频率轴上的点变得更加密集。具体来说，FFT 的频谱分辨率与采样点数*N*和采样间隔*T*相关：增加N（通过补零的方式）会减小频率分辨率Δf，从而使得频谱中的频率轴更加细化，频率间隔变小，这样就可以更精细地看到信号的频率成分。**补零并不会改变信号的频谱内容**，也就是说补零并不会提高实际的频谱信息量，也不会让你“发现”新的频率成分。它仅仅是在频域中使得离散频谱的点数增加，使你能够以更精细的频率步长查看已有的频率成分。因此，**补零只是增加了频谱的平滑度，而不是增加了实际的频谱分辨率**。当我们对信号进行 FFT 分析时，FFT 结果是离散的，每个点表示一个特定频率下的能量大小。补零增加了 FFT 计算的频率采样点数，使得频谱图上出现更多的频率点，虽然这些点之间并没有新的频率信息，但图像会变得更加平滑和连续，视觉上更容易观察信号的频谱特性。