**实验五 离散傅立叶变换及谱分析**

1. **实验目的**
2. 熟悉离散傅里叶级数变换变换原理，并掌握离散傅里叶级数正变换和反变换的MATLAB程序实现方法；
3. 熟悉离散傅里叶变换原理，并掌握离散傅里叶正变换和反变换的MATLAB程序实现方法；
4. 熟悉快速离散傅里叶变换原理，并掌握快速离散傅里叶正变换和反变换的MATLAB程序实现方法；
5. 通过对比实验，直观体会DFT和FFT的运行时间差异；
6. 通过实验对比原信号与先进行FFT后再进行IFFT的重构信号，分析误差及其原因，以便正确应用FFT。
7. **实验原理与方法**
8. **离散傅里叶级数正变换和反变换**

**原理：**离散周期信号的离散傅里叶级数（DFS）的频谱是周期性的，因为时域的离散对应于频率的周期。离散周期序列的DFS离散谱是离散时间序列傅里叶变换（DTFT）主值序列连续谱的离散抽样，即在（0，2π）的频域区间上取N个点。

**时域：**离散整型时间变量的周期函数；

**频域：**离散整型频率变量的周期函数。

离散傅里叶级数正变换和反变换数学表达式如下：

**正变换：** （5-1）

**反变换：** （5-2）

其中，和都是周期为N的周期序列，。

**方法：**MATLAB中没有提供离散傅里叶级数的正变换和反变换函数，需要自己编译。MATLAB参考程序如下：

* **离散傅里叶级数正变换DFS代码**

function [Xk] = dfs(xn,N)

% 计算离散傅里叶级数(DFS)系数

n = [0:1:N-1]; % n的行向量

k = [0:1:N-1]; % k的行向量

WN = exp(-j\*2\*pi/N); % Wn 因子

nk = n'\*k; % 产生一个含nk值的N 乘 N维矩阵

WNnk = WN .^ nk; % DFS 矩阵

Xk = xn \* WNnk; % DFS 系数的行向量

* **离散傅里叶级数反变换IDFS代码**

function [xn] = idfs(Xk,N)

% 计算逆离散傅里叶级数(IDFS)

n = [0:1:N-1]; % n的行向量

k = [0:1:N-1]; % k 的行向量

WN = exp(-j\*2\*pi/N); % Wn 因子

nk = n'\*k; % 产生一个含nk值的N 乘 N维矩阵

WNnk = WN .^ (-nk); % IDFS 矩阵

xn = (Xk \* WNnk)/N; % IDFS 值的行向量

1. **离散傅里叶正变换和反变换**

**原理：**离散傅里叶变换（DFT），是傅里叶变换在时域和频域上都呈现离散的形式，将时域信号的采样变换为在离散时间傅里叶变换（DTFT）频域的采样。在形式上，变换两端（时域和频域上）的序列是有限长的，而实际上这两组序列都应当被认为是离散周期信号的主值序列。即使对有限长的离散信号作DFT，也应当将其看作经过周期延拓成为周期信号再作变换。

**时域：**离散整型时间变量的非周期函数；

**频域：**离散整型频率变量的非周期函数。

离散傅里叶正变换和反变换数学表达式如下：

**正变换：** （5-3）

**反变换：** （5-4）

其中，。

**方法：**一般在实际应用中会用FFT来实现信号的离散傅里叶变换，因此MATLAB中并未提供原始的离散傅里叶正变换和反变换函数，需要自己编译。MATLAB参考程序如下：

* **离散傅里叶正变换DFT代码**

function [Xk] = dft(xn,N)

% 计算离散傅里叶叶变换

n = [0:1:N-1]; % n的行向量

k = [0:1:N-1]; % k的行向量

WN = exp(-j\*2\*pi/N); % Wn 因子

nk = n'\*k; % 产生一个含nk值的N 乘 N维矩阵

WNnk = WN .^ nk; % DFT 矩阵

Xk = xn \* WNnk; % DFT 系数的行向量

* **离散傅里叶反变换IDFT代码**

function [xn] = idft(Xk,N)

% 计算逆离散傅里叶变换

n = [0:1:N-1]; % n的行向量

k = [0:1:N-1]; % k的行向量

WN = exp(-j\*2\*pi/N); % Wn 因子

nk = n'\*k; % 产生一个含nk值的N 乘 N维矩阵

WNnk = WN .^ (-nk); % IDFT 矩阵

xn = (Xk \* WNnk)/N; % IDFT 的行向量

1. **快速傅里叶正变换和反变换**

**原理：** 快速傅里叶正变换（FFT）的基本思想是把原始的N点序列，依次分解成一系列的短序列。充分利用DFT计算式中指数因子 所具有的对称性质和周期性质，进而求出这些短序列相应的DFT并进行适当组合，达到删除重复计算，减少乘法运算和简化结构的目的。

**方法：**计算离散傅里叶变换的快速方法，有按时间抽取的FFT算法和按频率抽取的FFT算法。前者是将时域信号序列按偶奇分排，后者是将频域信号序列按偶奇分排。它们都借助于的两个特点：一是周期性；二是对称性。

MATLAB提供了快速傅里叶正变换函数fft和快速傅里叶反变换函数ifft，其语句格式为

• Y = fft(X) %对X做离散傅里叶正变换，输出为Y。

• Y = fft(X,n) %若X点的个数少于n，则在后面加0，凑到n；若X的点的个数多于n，则删除多余的数。

• X = ifft(Y) %对Y做离散傅里叶反变换，输出为X。

• X = ifft(Y,n) %若X点的个数少于n，则在后面加0，凑到n；若X的点的个数多于n，则删除多余的数。

1. **实验内容**
2. **离散傅里叶级数正变换和反变换**

%离散傅里叶级数正变换和反变换。

**代码：**

xn=[1,2,3,4];

N=4;

xk=dfs(xn,N) % 离散傅里叶级数正变换

xn1=idfs(xk,N) %离散傅里叶级数反变换

**结果：**

xk = 10.0000 + 0.0000i -2.0000 + 2.0000i -2.0000 - 0.0000i -2.0000 - 2.0000i

xn1 = 1.0000 - 0.0000i 2.0000 - 0.0000i 3.0000 - 0.0000i 4.0000 + 0.0000i

1. **离散傅里叶正变换和反变换**

%离散傅里叶变换正变换和反变换。

**代码：**

xn=[4,3,2,1];

N=4;

xk=dft(xn,N) % 离散傅里叶正变换

xn1=idft(xk,N) %离散傅里叶反变换

**结果：**

xk = 10.0000 + 0.0000i 2.0000 - 2.0000i 2.0000 - 0.0000i 2.0000 + 2.0000i

xn1 = 4.0000 - 0.0000i 3.0000 + 0.0000i 2.0000 + 0.0000i 1.0000 + 0.0000i

1. **快速傅里叶正变换和反变换**

%快离散傅里叶变换正变换和反变换。

**代码：**

xn=[3,4,1,2];

N=4;

xk=fft(xn,N) % 离散傅里叶正变换

xn1=ifft(xk,N) %离散傅里叶反变换

**结果：**

xk = 10.0000 + 0.0000i 2.0000 - 2.0000i -2.0000 + 0.0000i 2.0000 + 2.0000i

xn1 = 3 4 1 2

1. **DFT和FFT的运行时间对比**

% DFT和FFT的运行时间对比。

**代码：**

N=16384; % 2^14 序列长度

M=500; % 非零值长度

x=[1:M,zeros(1,N-M)]; %非零值的其它部分补零

t=cputime; %记录时间

y1=dft(x,N); %离散傅里叶变换

Time\_dft=cputime-t %离散傅里叶变换时间，不同的计算机上运行时间会不一样

t1=cputime;

y2=fft(x,N);

Time\_fft=cputime-t1 %快速离散傅里叶变换时间，不同的计算机上运行时间会不一样

**结果：**

Time\_dft = 416.6094

Time\_fft = 0.0313

1. **原信号与先进行FFT后再进行IFFT的重构信号比较**

**%**无噪声信号与先进行FFT后再进行IFFT的重构信号比较。

**代码：**

fs=100;

N=128;

n=0:N-1;

t=n/fs;

x=sin(2\*pi\*40\*t)+sin(2\*pi\*15\*t); %无噪声信号

subplot(2,2,1)

plot(t,x)

title('original signal ')

y=fft(x,N);

mag=abs(y);

f=(0:length(y)-1)'\*fs/length(y);

subplot(2,2,2)

plot(f,mag)

title('FFT to original signal')

xifft=ifft(y);

magx=real(xifft);

ti=[0:length(xifft)-1]/fs;

subplot(2,2,3)

plot(ti,magx);

title('signal from IFFT ')

yif=fft(xifft,N);

mag=abs(yif);

subplot(2,2,4)

plot(f,mag)

title('FFT to signal from IFFT')

error = xifft-x;%结果误差

figure;

stem(n,abs(error));%绘图，误差幅度

title('结果误差')

**结果：**





1. **学生作业**
2. 将无噪声信号与先进行FFT后再进行IFFT的重构信号比较实验中的信号添加随机噪声：x=sin(2\*pi\*40\*t)+sin(2\*pi\*15\*t)+ 5\*randn(1,N)，做对比实验，并绘制图形。

**运行代码:**

fs=100;

N=128;

n=0:N-1;

t=n/fs;

x=sin(2\*pi\*40\*t)+sin(2\*pi\*15\*t)+ 5\*randn(1,N);

subplot(2,2,1)

plot(t,x)

title('original signal ')

y=fft(x,N);

mag=abs(y);

f=(0:length(y)-1)'\*fs/length(y);

subplot(2,2,2)

plot(f,mag)

title('FFT to original signal')

xifft=ifft(y);

magx=real(xifft);

ti=[0:length(xifft)-1]/fs;

subplot(2,2,3)

plot(ti,magx);

title('signal from IFFT ')

yif=fft(xifft,N);

mag=abs(yif);

subplot(2,2,4)

plot(f,mag)

title('FFT to signal from IFFT')

error = xifft-x;%结果误差

figure;

stem(n,abs(error));%误差幅度

title('½á¹ûÎó²î')

**运行截图:**

****

****

1. 采用时间抽取的方法，通过2个1024点的FFT变换的组合实现2048点的FFT变换，并计算直接变换结果和组合变换结果之间的误差并绘制图形。
2. % 生成信号
3. N = 2048;
4. t = linspace(0, 1, N);
5. x = sin(2\*pi\*10\*t) + sin(2\*pi\*20\*t) + sin(2\*pi\*30\*t);
7. % 将信号分成两部分
8. x1 = x(1:2:end);
9. x2 = x(2:2:end);
11. % ·分别计算两个小的FFT变换
12. X1 = fft(x1);
13. X2 = fft(x2);
15. % 计算组合的FFT变换
16. X = zeros(1, N);
17. for k = 1:N/2
18. X(k) = X1(k) + exp(-1i\*2\*pi\*(k-1)/N)\*X2(k);
19. X(k+N/2) = X1(k) - exp(-1i\*2\*pi\*(k-1)/N)\*X2(k);
20. end
22. %计算直接变换的FFT
23. X\_direct = fft(x);
25. %绘制结果
26. subplot(2,1,1);
27. plot(abs(X));
28. title('FFT via Time Decimation');
30. subplot(2,1,2);
31. plot(abs(X\_direct));
32. title('Direct FFT');
34. % ¼ÆËãÎó²î²¢Êä³ö
35. MSE = mean(abs(X - X\_direct).^2);
36. disp(['MSE between FFT via Time Decimation and Direct FFT: ' num2str(MSE)]);

**运行截图:**



****