**安徽大学人工智能学院**

**《数字信号处理》**

**实验案例设计报告**

**课程名称：** 数字图像处理实验

**专 业：** 机器人工程

**班 级：** 3班

**学 号：** WA2224013

**姓 名：** 郭义月

**任课老师： 谭春雨**

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | 实验4 | 实验次序 | 4 |
| 实验地点 | 笃行南楼A104 | 实验日期 | 11.6 |
| 实验内容：  **例8.1**  已知序列，变成实现DIT-FFT，计算X(k)  **实验目的：**  通过编程实现离散傅里叶变换（DIT-FFT），计算给定序列 x(n) 的频域表示 X(k)。通过对比手动实现的FFT结果和MATLAB内置FFT函数的结果，验证手动实现的正确性和有效性，加深对FFT算法的理解。  **实验原理：**  DIT-FFT算法的运算过程都很有规律，它有三个显著特点  (1)原位计算  对于的FFT共进行M级运算，每级由个蝶形运算组成。在同一级中,每个蝶形的输人数据只对本蝶形有用.且输出节点与输人节点在同-水平线上.这就意味着每计算完一个蝶形后.所得数据可立即存人原输人数据所占用的数组元素(存储单元)，这种原位(址)计算的方法可节省大量内存。  (2)蝶形运算  实现FFT运算的核心是蝶形运算，找出蝶形运算的规律是编程的基础。蝶形运算是分级进行的，每级的蝶形运算可以按蝶形因子的指数大小排序进行。如果指数大小一样则可从上往下依次进行蝶形运算。对点的FFT共有M级运算,用L表示从左到右的运算级数(I=1.2...M)。第L级共有个不同指数的蝶形因子.用R表示这些不同指数蝶形因子从上到下的顺序(R=1,2....B-1)。第R个蝶形因子的指数为P=。  首先读人数据，根据数据长度确定运算级数M,运算总点数,不足做补零处理。然后对读人数据进行数据倒序操作。数据倒序后从第1级开始逐级进行,共进行M级运算。在进行第L级运算时.先算出该级不同旋转因子的个数 (也是该级中各个蝶形运算两输人数据的间距)，再从R=1开始按序计算，直到R=B-1结束。每个R对应的旋转因子指数P= .旋转因子指数相同的蝶形从上往下依次逐个运算  (3)倒序  为了保证运算输出的X(k)按顺序排列，要求序列x(n)倒序输人，即在运算前要先对输入的序列进行位序颠倒。  根据以上所说的计算流程，可编写按时间抽选的离散傅里叶变换快速算法函数  **实验代码：**  clc;clear;close all;  xn = [1,2,3,4,5,6,7,8];  ditfft(xn)  function Xk = ditfft(xn)  M = nextpow2(length(xn));  N = 2^M;  for m = 0:N/2-1  WN(m+1) = exp(-j\*2\*pi/N)^m;  end  A = [xn,zeros(1,N-length(xn))];  disp(A);  J = 0;  for I= 0:N-1  if I<J  T = A(I+1);  A(I+1) = A(J+1);  A(J+1) = T;  end  K = N/2;  while J >= K  J = J - K;  K = K/2;  end  J = J + K;  end  disp('倒序后各个存储单元的数据：');  disp(A);  for L = 1:M  disp('运算级次：');  disp(L);  B = 2^(L-1);  for R = 0:B-1  P = 2^(M-L)\*R;  for K = R:2^L:N-2  T = A(K+1)+A(K+B+1)\*WN(P+1);  A(K+B+1) = A(K+1) - A(K+B+1) \* WN(P+1);  A(K+1) = T;  end  end  disp('本级运算后各存储单元的数据：');  disp(A);  end  disp('输出各存储单元的数据：');  Xk = A;  end  **实验结果：**    调用matlab提供的fft快速计算函数进行计算，比较两者的结果，可以看出，两者相等，证明了函数ditfft程序的正确。    **实验内容：例8.2**  已知序列的频谱X(k)为，试求序列  **实验目的：**  通过编程实现逆离散傅里叶变换（IDFT），计算给定频谱 X(k) 对应的时间序列 x(n)。通过对比手动实现的 IDFT 结果和 MATLAB 内置 ifft 函数的结果，验证手动实现的正确性和有效性，加深对 IDFT 算法的理解。  **实验原理：**  逆离散傅里叶变换（IDFT）用于将频域信号转换回时间域信号。利用傅里叶变换的共轭对称性，可以通过对频谱 X(k) 取共轭后进行快速傅里叶变换（FFT），再对结果取共轭并归一化，得到对应的时间序列 x(n)。这样，通过使用 FFT 和共轭运算，可以高效地实现IDFT。最终，通过对比手动计算的 IDFT 结果和 MATLAB 内置的 ifft 函数的结果，可以验证算法的正确性和效率  **实验代码：**  方法一：利用，IFFT运算与FFT运算共用一个子程序来实现，可以调用ditfft或fft函数命令来实现，matlab程序为：  clc;clear;close all;  Xk = [36,-4+9.6569i,-4+4i,-4+1.6569i,-4,-4-1.6569i,-4-4i,-4-9.6569i];  N = length(Xk);  Xk1 = conj(Xk);  xn1 = ditfft(Xk1);  xn1 = conj(xn1)/N;  xn1 = real(xn1)  xn2 = fft(Xk1);  xn2 = conj(xn2)/N;  xn2 = abs(xn2)  function Xk = ditfft(xn)  M = nextpow2(length(xn));  N = 2^M;  for m = 0:N/2-1  WN(m+1) = exp(-j\*2\*pi/N)^m;  end  A = [xn,zeros(1,N-length(xn))];  disp(A);  J = 0;  for I= 0:N-1  if I<J  T = A(I+1);  A(I+1) = A(J+1);  A(J+1) = T;  end  K = N/2;  while J >= K  J = J - K;  K = K/2;  end  J = J + K;  end  disp('倒序后各个存储单元的数据：');  disp(A);  for L = 1:M  disp('运算级次：');  disp(L);  B = 2^(L-1);  for R = 0:B-1  P = 2^(M-L)\*R;  for K = R:2^L:N-2  T = A(K+1)+A(K+B+1)\*WN(P+1);  A(K+B+1) = A(K+1) - A(K+B+1) \* WN(P+1);  A(K+1) = T;  end  end  disp('本级运算后各存储单元的数据：');  disp(A);  end  disp('输出各存储单元的数据：');  Xk = A;  end  **实验结果：**    方法二：直接调用matlab提供的快速傅里叶逆变换算法函数命令ifft实现    **实验内容：8-3**  对序列进行离散傅里叶变换（DFT）时，试比较直接采用DFT与采用快速变换FFT计算的时间差异  **实验目的：**  通过对比直接计算离散傅里叶变换（DFT）和快速傅里叶变换（FFT）的执行时间，评估两种方法在计算效率上的差异。通过实验结果验证 FFT 在计算大规模 DFT 时的高效性，从而加深对 FFT 算法优越性的理解。  **实验原理：**  在实验中，将对不同长度的随机序列分别使用直接计算 DFT（DFTfor 和 DFTmat）和快速算法 FFT（DIT-FFT）进行变换，并记录每种方法的执行时间。通过绘制执行时间与序列长度的关系图，可以清晰地对比两种方法的时间差异，验证 FFT 的高效性  **实验代码：**  clc;clear;close all;  Nmax = 256;  ditfft\_time = zeros(1,Nmax);  for n = 1:Nmax  x = rand(1,n);  t = clock;  ditfft(x);  ditfft\_time(n) = etime(clock,t);  end  k = 1:Nmax;  subplot(3,1,1);plot(k,ditfft\_time,'--');  ylabel('t/s');title('DIT-FFT执行时间');  DFTfor\_time = zeros(1,Nmax);  for n = 1:Nmax  x = rand(1,n);  t = clock;  DFTfor(x);  DFTfor\_time(n) = etime(clock,t);  end  k = 1:Nmax;  subplot(3,1,2);plot(k,DFTfor\_time,'--');  ylabel('t/s');title('DFTfor执行时间');  DFTmat\_time = zeros(1,Nmax);  for n = 1:Nmax  x = rand(1,n);  t = clock;  DFTmat(x);  DFTmat\_time(n) = etime(clock,t);  end  k = 1:Nmax;  subplot(3,1,3);plot(k,DFTmat\_time,'--');  ylabel('t/s');title('DFTmat执行时间');xlabel('N');  function Xk = ditfft(xn)  M = nextpow2(length(xn));  N = 2^M;  for m = 0:N/2-1  WN(m+1) = exp(-j\*2\*pi/N)^m;  end  A = [xn,zeros(1,N-length(xn))];  disp(A);  J = 0;  for I= 0:N-1  if I<J  T = A(I+1);  A(I+1) = A(J+1);  A(J+1) = T;  end  K = N/2;  while J >= K  J = J - K;  K = K/2;  end  J = J + K;  end  disp('倒序后各个存储单元的数据：');  disp(A);  for L = 1:M  disp('运算级次：');  disp(L);  B = 2^(L-1);  for R = 0:B-1  P = 2^(M-L)\*R;  for K = R:2^L:N-2  T = A(K+1)+A(K+B+1)\*WN(P+1);  A(K+B+1) = A(K+1) - A(K+B+1) \* WN(P+1);  A(K+1) = T;  end  end  disp('本级运算后各存储单元的数据：');  disp(A);  end  disp('输出各存储单元的数据：');  Xk = A;  end  function Xk = DFTmat(xn)  N = length(xn);  n = 0:N-1;k = n;nk = n'\*k;  WN = exp(-j\*2\*pi/N);  Wnk = WN.^nk;  Xk = xn\*Wnk;  end  function X=DFTfor(xn)  N = length(xn);  X = zeros(1,N);  for k = 0:N-1  for n = 0:N-1  X(k+1) = X(k+1) + xn(n+1) \*exp(-j\*2\*pi\*n\*k/N);  end  end  end  **实验结果：**    **实验内容：例9-1**  已知序列，，试利用快速卷积法计算这两个序列的卷积和  **实验目的：**  利用快速卷积法计算给定序列 x(n) 和 h(n) 的卷积和 y(n)，并通过绘图展示输入序列和卷积结果。通过实验验证快速卷积法在计算序列卷积中的有效性和准确性，加深对快速卷积法原理的理解。  **实验原理：**  序列x(n)的长度N=15，序列h(n)的长度N=20，线性卷积的长度N=15+20-1=34，因此，可用34点圆周卷积替代线性卷积，在求，以及时，可以用FFT实现，需要先对序列x(n)和h(n)补零，使之长度达到34  **实验代码：**  clc;clear;close all;  nx = 0:14;xn = sin(0.4\*nx);  nh = 0:19;hn = 0.9.^nh;  N1 = length(xn);N2 = length(hn);  N = N1 + N2 - 1;  xn = [xn,zeros(1,N-N1)];  hn = [hn,zeros(1,N-N2)];  yn = fftconv(xn,hn,N);  nn = 0:N-1;  subplot(3,1,1);stem(nn,xn);title('序列x(n)');  subplot(3,1,2);stem(nn,hn);title('序列h(n)');  subplot(3,1,3);stem(nn,yn);title('序列y(n)');  function y = fftconv(x1,x2,N)  Xk1 = fft(x1,N);  Xk2 = fft(x2,N);  Yk = Xk1.\*Xk2;  y = ifft(Yk);  end  **实验结果：**    **实验内容：9-2**  令是一个L点在[0,1]之间均匀分布的随机数，是一个L点均值为0，方差为1的高斯随机序列，试比较直接计算线性卷积所需要的时间和利用FFT计算所需的时间。  **实验目的：**  比较直接计算线性卷积所需的时间和利用快速傅里叶变换（FFT）计算  所需的时间，通过实验验证 FFT 在大规模卷积计算中的高效性。  **实验原理：**  计算线性卷积，可调用matlab提供的conv函数来实现，利用FFT计算（简称快速卷积），根据圆周卷积提到线性卷积的条件是，通过对，做N点傅里叶变换（用FFT实现），即，然后对利用离散傅里叶逆变换（用IFFT实现）求得，即，因此，可调用MATLAB提供的fft和ifft函数实现。  **实验代码：**  clc;clear;close all;  K = 1024;  conv\_time = zeros(1,K);fft\_time = zeros(1,K);  for L = 1:K  tc = 0;tf = 0;  N = 2\*L-1;  nu = ceil(log10(N)/log10(2));N1 = 2^nu;  for i = 1:100  x1 = rand(1,L);x2 = randn(1,L);  t0 = clock;y1 = conv(x1,x2);  t1 = etime(clock,t0);tc = tc + t1;  t0 = clock;  Y2 = fft(x1,N1).\*fft(x2,N1);  y2 = ifft(Y2,N1);  t2 = etime(clock,t0);tf = tf + t2;  end  conv\_time(L) = tc/100;  fft\_time(L) = tf/100;  end  n = 1:K;  plot(n,conv\_time(n),'k--');  hold on;  plot(n,fft\_time(n),'b--');  hold off;xlabel('N');ylabel('t/s')  **实验结果：**  从图中可以看出，随着L的增大，FFT实现快速卷积的耗时远小于线性卷积的耗时，随着L的增大，线性卷积所需时间近似按指数增长，而FFT实现快速卷积的耗时基本上呈线性增长（注意：由于，快速卷积耗时在L的某一范围内基本上是不变的）    **实验过程中遇到的问题**  FFT算法具有蝶形结构，运算量小，采用原位计算，输入或输出序列的倒位序三个特点，相关matlab函数有:  X=fft(x,N)：采用fft算法计算序列向量x的N点DFT变换，当N缺省时，fft函数自动按x的长度计算DFT  x=ifft(X,N)：采用fft算法计算序列向量X的N点IDFT  clock：按年，月，日，时，分，秒格式返回当前时间 | | | |