数字信号处理实验

基于DFT的线性卷积计算

北京航空航天大学
电子信息工程学院
雷 鹏，王 俊
peng.lei@buaa.edu.cn

主要内容

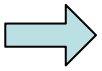
- 1. MATLAB基础函数
- 2. 基于时域计算的线性卷积实现
- 3. 基于DFT的线性卷积实现
- 4. 实验内容及要求

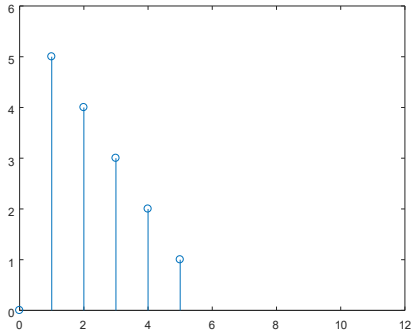
1. MATLAB基础函数

线性卷积函数

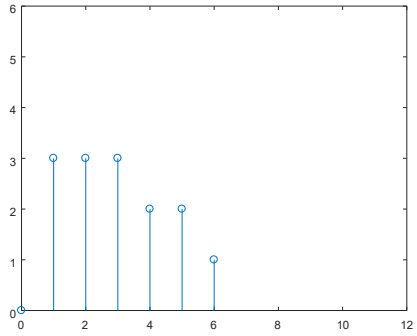
- c = conv(a,b)
  - a　：卷积序列，长度为M
  - b　：卷积序列，长度为N
  - c　：输出序列，长度为M+N-1

```
x = [0 5 4 3 2 1]; xn = 0:5;
h = [0 3 3 3 2 2 1]; hn = 0:6;
figure(1); stem(xn,x); axis([0 12 0 6]);
figure(2); stem(hn,h); axis([0 12 0 6]);
y = conv(x,h); yn = 0:11;
figure(3); stem(yn,y); xlim([0 12]);
```

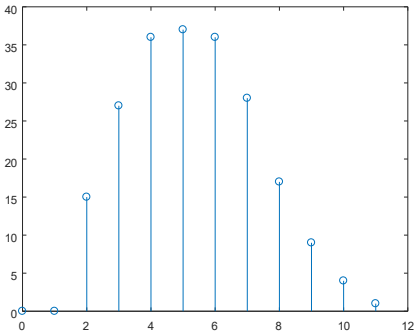




x



h



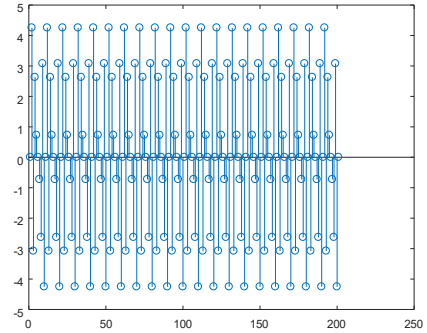
y

1. MATLAB基础函数

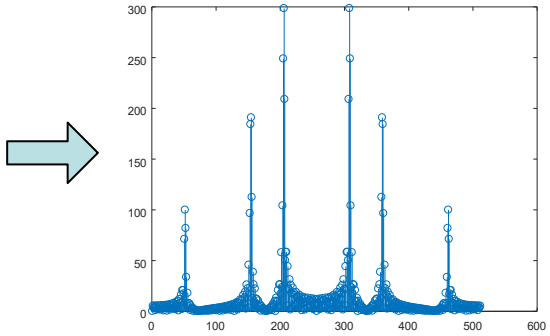
DFT的快速实现函数

- y = fft(sig,n)
  - y : n点DFT计算结果
  - sig : 输入序列
  - n : DFT运算的点数

```
t = 0:0.01:2;
x = sin(2*pi*10*t)+2*sin(2*pi*30*t)+...
3*sin(2*pi*40*t);
n = 512;
y = fft(x,n);
figure(1); stem(x);
figure(2); stem(abs(y));
```



时域序列



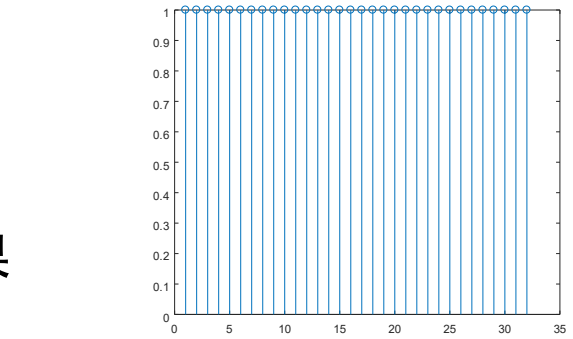
512点的DFT幅度

1. MATLAB基础函数

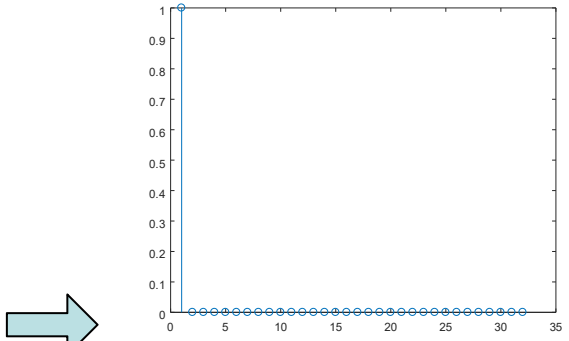
IDFT函数

- y = ifft(x)
  - y : n点IDFT计算结果
  - x : 输入序列

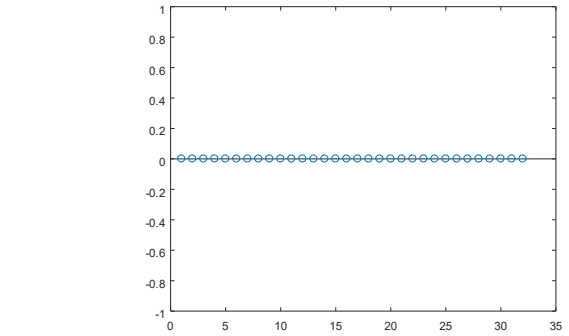
```
x = ones(32,1);
y = ifft(x);
figure(1); stem(x);
figure(2); stem(real(y));
figure(3); stem(imag(y));
```



频域序列



IDFT结果实部

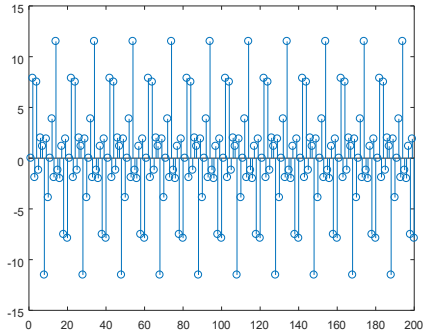


IDFT结果虚部

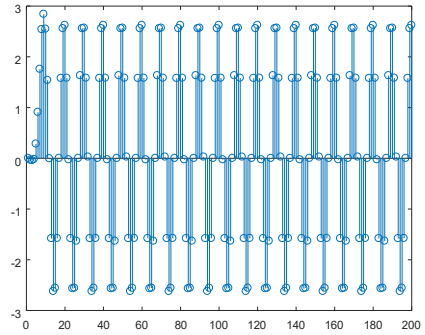
2. 基于时域计算的线性卷积实现

- 时域线性卷积实现滤波处理
  - 低通滤波

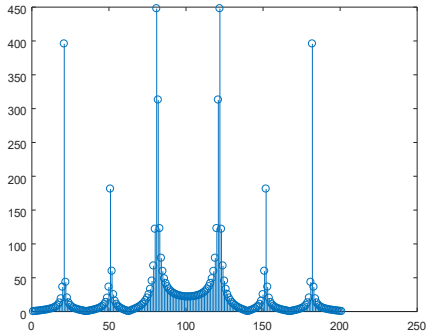
```
t = 0:0.0005:0.1;
x = 4*sin(2*pi*200*t)+2*sin(2*pi*500*t)+...
6*sin(2*pi*800*t);
h = [-0.0043, -0.0065, 3.0912e-18, 0.0415, ...
0.1254 , 0.2161, 0.2558, 0.2161, 0.1254, ...
0.0415, 3.0912e-18, -0.0065, -0.0043];
y = conv(x,h);
figure(1); stem(x); xlim([0 200]);
figure(2); stem(y); xlim([0 200]);
figure(3); stem(abs(fft(x)));
figure(4); stem(abs(fft(y)));
```



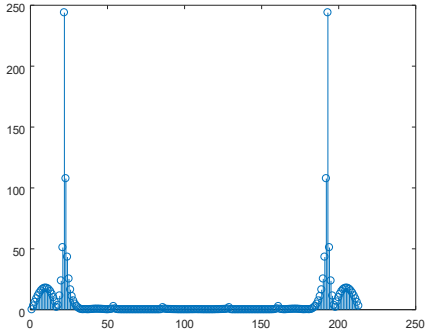
输入信号



输出信号



输入信号幅频

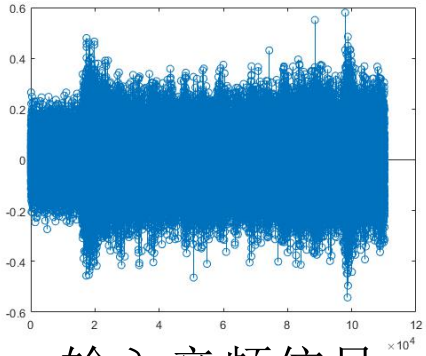


输出信号幅频

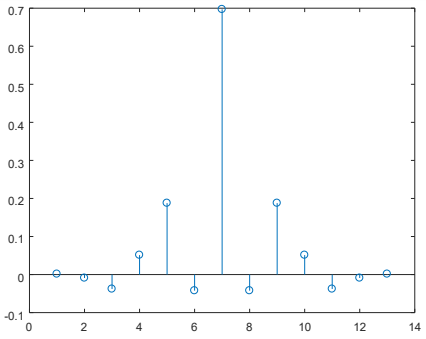
2. 基于时域计算的线性卷积实现

- 音频信号经过滤波器系统的时域实现

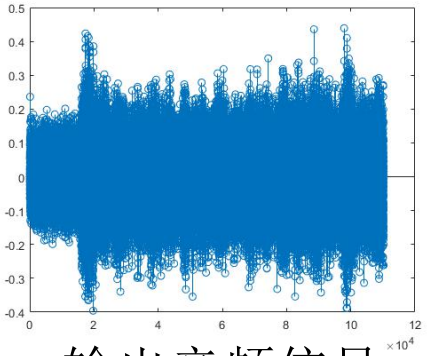
```
[sig, fs] = audioread('E:\Example_music_seg.wav');
h = [0.0015, -0.009, -0.0378, 0.0512, 0.1878, -0.0422, 0.6971, -0.0422, ...
0.1878, 0.0512, -0.0378, -0.009, 0.0015]; % 带阻滤波器
sig_fil = conv(sig,h);
player = audioplayer(sig_fil,fs); play(player);
figure(1); stem(sig);
figure(2); stem(h);
figure(3); stem(sig_fil);
```



输入音频信号



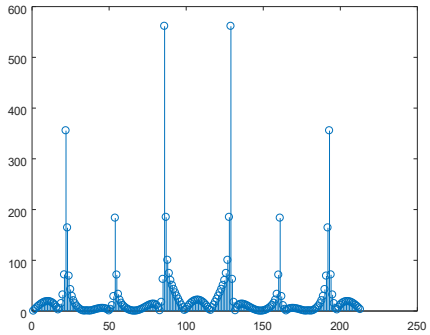
系统单位冲激响应



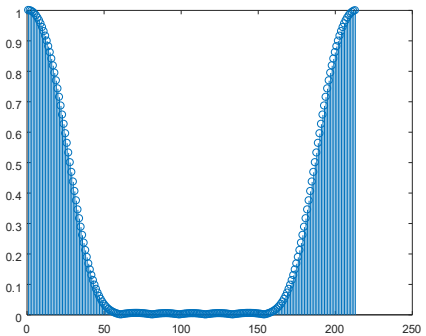
输出音频信号

3. 基于DFT的线性卷积实现

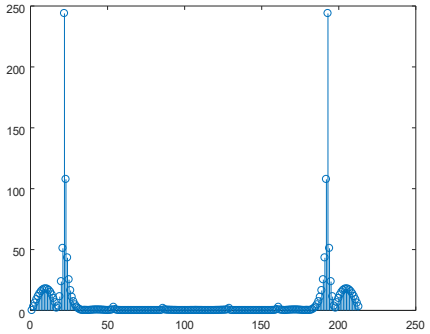
- DFT实现滤波处理
  - 低通滤波



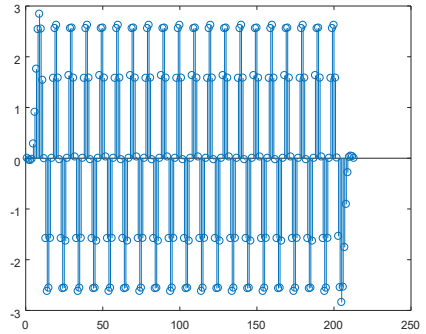
输入信号幅频



系统幅频响应



输出信号幅频响应



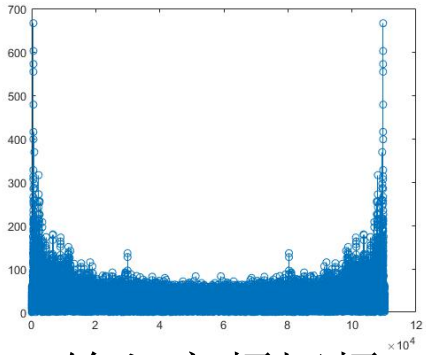
输出信号

```
t = 0:0.0005:0.1;
x = 4*sin(2*pi*200*t)+2*sin(2*pi*500*t)+...
6*sin(2*pi*800*t);
h = [-0.0043, -0.0065, 3.0912e-18, 0.0415, ...
0.1254 , 0.2161, 0.2558, 0.2161, 0.1254, ...
0.0415, 3.0912e-18, -0.0065, -0.0043];
N = length(x) + length(h) -1;
X_freq = fft(x,N);
H_freq = fft(h,N);
Y_freq = X_freq .* H_freq;
y = ifft(Y_freq);
figure(1); stem(abs(X_freq));
figure(2); stem(abs(H_freq));
figure(3); stem(abs(Y_freq));
figure(4); stem(y);
```

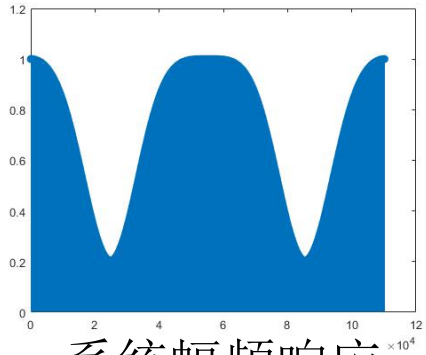
3. 基于DFT的线性卷积实现

- 音频信号经过滤波器系统的DFT实现

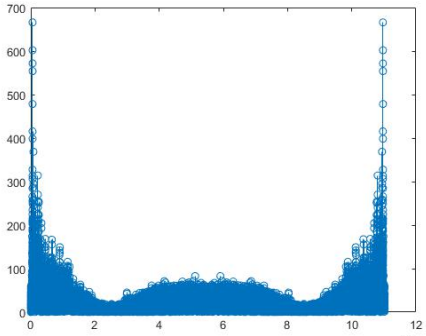
```
[sig, fs] = audioread('E:\Example_music_seg.wav');
h = [0.0015, -0.009, -0.0378, 0.0512, 0.1878, -0.0422, 0.6971, ...
-0.0422, 0.1878, 0.0512, -0.0378, -0.009, 0.0015]; % 带阻滤波器
N = length(sig) + length(h) -1;
Sig_freq = fft(sig,N);
H_freq = fft(h,N);
Y_freq = Sig_freq .* transpose(H_freq);
y = ifft(Y_freq);
player = audioplayer(y,fs);
play(player);
figure(1); stem(abs(Sig_freq));
figure(2); stem(abs(H_freq));
figure(3); stem(abs(Y_freq));
```



输入音频幅频



系统幅频响应



输出音频幅频

4. 实验内容及要求

- 实验一：基于差分方程的音频信号滤波处理
  - 给定离散时间系统的差分方程为 $y[n]=\sum_{m=0}^{15} b_m x[n-m]$，其中$b_m$=[-0.0024, -0.0042, 0.0095, 0.02, -0.038, -0.0696, 0.1374, 0.4472, 0.4472, 0.1374, -0.0696, -0.038, 0.02, 0.0095, -0.0042, -0.0024]；
  - 加载音频信号文件Test_music.wav，利用filter函数实现基于上述差分方程的音频信号滤波处理，给出输入、输出音频信号的时域图形，并使用play函数进行音频播放，分析输入、输出结果差异。

4. 实验内容及要求

- 实验二：基于时域线性卷积的音频信号滤波处理
  - 给定离散时间系统的单位冲激响应h[n]=[-0.0024, -0.0042, 0.0095, 0.02, -0.038, -0.0696, 0.1374, 0.4472, 0.4472, 0.1374, -0.0696, -0.038, 0.02, 0.0095, -0.0042, -0.0024]，画出h[n]的图形化显示结果；
  - 加载音频信号文件Test_music.wav，利用conv函数实现对输入音频的滤波处理，给出输入、输出音频信号的时域图形，并使用play函数进行音频播放，分析输入、输出结果差异。

4. 实验内容及要求

- 实验三：基于DFT的音频信号滤波处理
  - 对于实验一中的离散时间系统的单位冲激响应，使用freqz函数获得该系统的幅频响应和相频响应，并给出其图形化显示结果；
  - 加载音频信号文件Test_music.wav，利用DFT变换实现对输入音频的滤波处理，给出输入、输出音频信号的幅频及时域图形，使用play函数进行音频播放，并与实验一、实验二结果进行对比分析，讨论应如何选择DFT变换点数。

谢 谢