### 信号与系统实验报告

名 称： FIR数字滤波器的设计及应用

学 院： 计算机科学与工程学院

专 业： 计算机科学与技术

学 号： 09022107

姓 名： 梁耀欣

日期： 2024年 5 月 29 日

一、实验目的

1. 进一步理解FIR滤波器的线性相位特性，熟悉4种线性相位FIR滤波器的幅频特性、相频特性。

2. 掌握用窗函数法设计FIR数字滤波器的原理及方法，了解各种窗函数对滤波器性能的影响。

3. 掌握用频率采样法设计FIR数字滤波器的原理及方法，了解不同采样点数和不同的过渡带点数对滤波器性能的影响。

4. 结合学过的信号分析的基本概念、基本理论及方法，将数字滤波器设计基本原理、基本方法进行综合运用，以解决实际信号分析与处理问题。

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FIR数字滤波器的设计方法及其应用。

　　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. 选择合适的窗函数设计一个线性相位FIR低通滤波器，通带截止频率，阻带截止频率，阻带最小衰减。要求分别绘制理想单位脉冲序列响应曲线、窗函数曲线、实际单位脉冲序列响应曲线、FIR低通滤波器的幅频响应曲线和相频响应曲线。
2. 利用频率采样法设计线性相位低通滤波器

1）要求3dB截止频率,采样点数分别取和，选用的情况。分别显示理想幅频特性的采样序列和实际滤波器的幅频响应，观察采样点数对滤波器特性的影响。

2）要求3dB截止频率,采样点数取，选用的情况。分别插入一个过渡点和两个过度点时滤波器的幅频响应，观察增加过度点对滤波器特性的影响。

1. 利用计算机的声音编辑工具录制一段语音信号，生成.wav文件。对语音信号的时域波形和频谱进行观察和分析，设计FIR数字滤波器，并用设计的滤波器对语音信号进行滤波，比较滤波前后语音信号的波形及频谱。

1）低通滤波器性能指标: ，，，；

2）高通滤波器性能指标: ，，，。

五、实验小结

**1.线性相位FIR低通滤波器**

实验思路：首先，确定通带和阻带的截止频率，分别为ω\_p=0.2πrad和ω\_s=0.4πrad。这两个频率之间的差值，即过渡带宽度，将用于计算滤波器的长度。根据过渡带宽度和滤波器的设计要求，估算出滤波器的阶数和长度。接下来，选择合适的窗函数，这里选择了Hamming窗，因为它在时域和频域之间提供了良好的折中。

计算理想低通滤波器的单位冲击响应是设计滤波器的关键一步。理想低通滤波器的截止频率设置为（ω\_p+ω\_s）/2，以确保通带和阻带之间的平衡。通过sinc函数计算理想的单位冲击响应，然后应用Hamming窗函数对其进行加权，以得到实际滤波器的冲击响应。

为了评估滤波器的性能，需要计算其频率响应，包括幅频响应和相频响应。使用MATLAB的freqz函数可以方便地得到频率响应的数据。绘制理想单位冲击响应、Hamming窗函数、实际单位冲击响应以及滤波器的幅频响应和相频响应图，以直观地观察滤波器的特性和性能。

整个实验过程通过MATLAB编程实现，最终生成多个图像，每个图像分别展示不同的响应曲线，从而全面地展示滤波器的设计过程和效果。

实验代码：

function Hamming\_Filter()

Wp = 0.2 \* pi; % 通带截止频率

Ws = 0.4 \* pi; % 阻带截止频率

trans\_bandwidth = Ws - Wp; % 过渡带宽

filter\_length = ceil(6.6 \* pi / trans\_bandwidth) + 1; % 滤波器长度

n = 0:1:filter\_length - 1;

Wc = (Wp + Ws) / 2; % 理想低通滤波器的截止频率

ideal\_response = ideal\_lp(Wc, filter\_length); % 理想滤波器的单位冲击响应

hamming\_window = (hamming(filter\_length))'; % Hamming窗

actual\_response = ideal\_response .\* hamming\_window; % 实际Hamming窗响应

[db, mag, phase, w] = freqz\_m(actual\_response, [1]); % 计算实际滤波器的幅度响应

delta\_w = 2 \* pi / 1000;

Ap = -(min(db(1:1:Wp / delta\_w + 1))); % 实际通带波动

As = -round(max(db(Ws / delta\_w + 1:1:501))); % 实际阻带衰减

% 绘制理想单位脉冲响应

figure;

stem(n, ideal\_response, 'filled', 'MarkerFaceColor', 'b')

title('理想单位脉冲响应')

xlabel('n')

ylabel('h\_d[n]')

% 绘制Hamming窗函数

figure;

stem(n, hamming\_window, 'filled', 'MarkerFaceColor', 'b')

title('Hamming窗函数')

xlabel('n')

ylabel('w[n]')

% 绘制实际单位脉冲响应

figure;

stem(n, actual\_response, 'filled', 'MarkerFaceColor', 'b')

title('实际单位脉冲响应')

xlabel('n')

ylabel('h[n]')

% 绘制FIR低通滤波器的幅频响应

[h, w] = freqz(actual\_response, 1);

db = 20 \* log10(abs(h)) / max(abs(h)); % 归一化频率

figure;

plot(w / pi, db)

title('FIR低通滤波器的幅频响应')

xlabel('归一化频率 (\times\pi rad/sample)')

ylabel('幅值 (dB)')

axis([0, 1, -100, 10]);

% 绘制FIR低通滤波器的相频响应

phase = angle(h);

figure;

plot(w / pi, unwrap(phase));

title('FIR低通滤波器的相频响应')

xlabel('归一化频率 (\times\pi rad/sample)')

ylabel('相位 (弧度)')

end

function hd = ideal\_lp(Wc, N)

% 理想滤波器的单位冲击响应

alpha = (N - 1) / 2;

n = [0:1:(N - 1)];

m = n - alpha + eps; % 加上最小数以避免除零

hd = sin(Wc \* m) ./ (pi \* m);

end

function [db, mag, phase, grd, w] = freqz\_m(b, a)

% 计算实际滤波器的幅度响应

[H, w] = freqz(b, a, 1000, 'whole');

H = (H(1:1:501))';

w = (w(1:1:501))';

mag = abs(H);

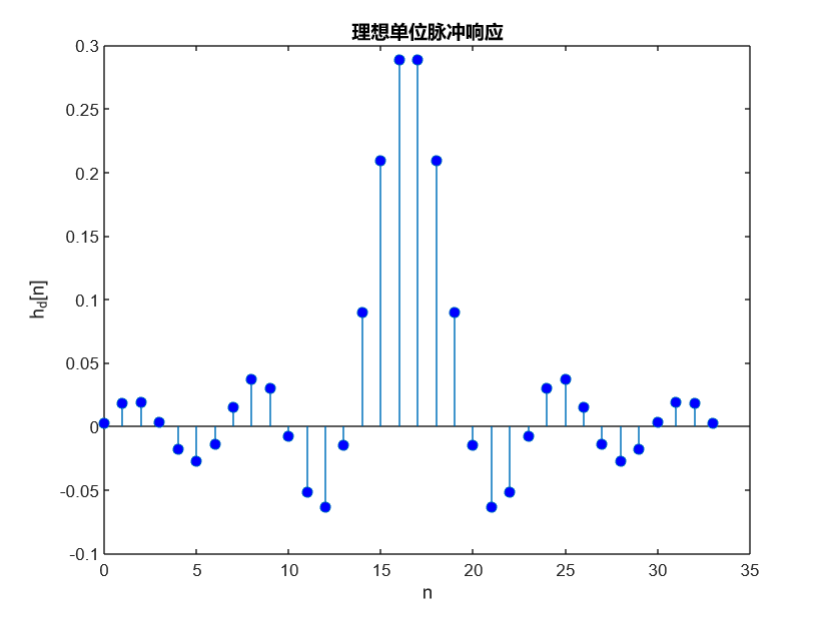
db = 20 \* log10((mag + eps) / max(mag));

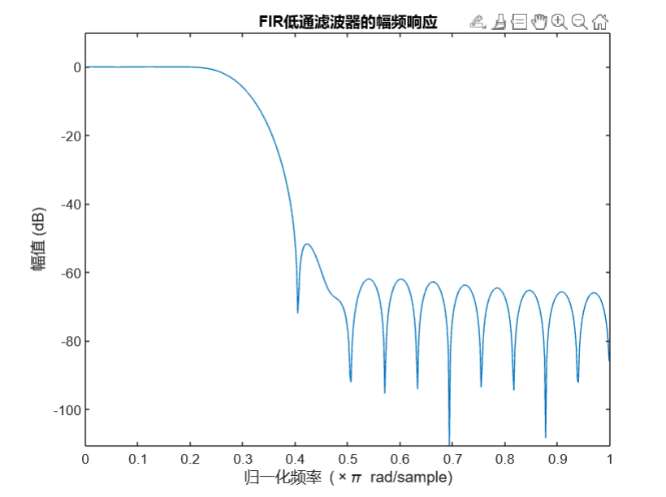
phase = angle(H);

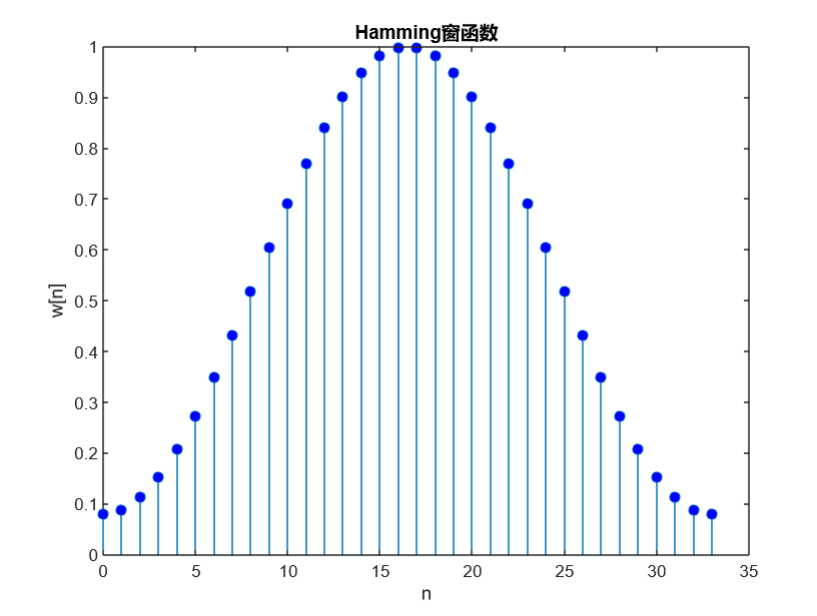
grd = grpdelay(b, a, w);

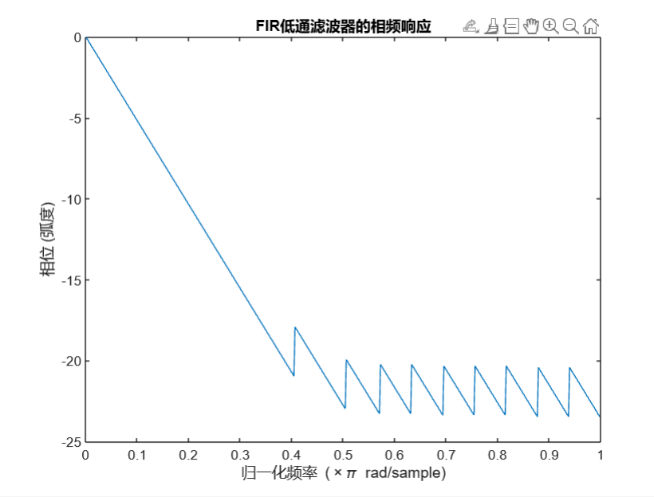
end

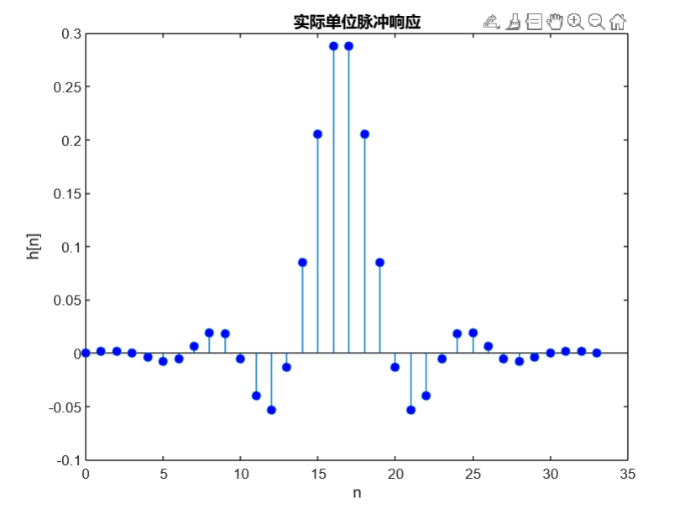
实验结果：

产生的图像如下：









**2.利用频率采样法设计线性相位低通滤波器**

（1）

实验思路：

设计线性相位低通滤波器的 MATLAB 代码，使用频率采样法，以实现给定的3dB截止频率和不同采样点数的要求。首先，我们定义截止频率和采样点数。然后，根据采样点数，计算频域采样序列，其中采样点在截止频率范围内的部分被置为1，其余部分为0。接着，通过逆离散傅立叶变换得到实际滤波器的单位响应。最后，计算滤波器的频率响应，并绘制理想幅频特性的采样序列和实际滤波器的幅频响应图。整个过程包括频域采样序列的计算、逆离散傅立叶变换、频率响应的计算以及绘图。

实验代码：

%% N为33

N1 = 33;

n1 = 0:N1-1;

wc = pi/2;

M1 = fix(wc / (2\*pi/N1));

M2 = N1 - 2\*M1 - 1;

A1 = [ones(1, M1+1), zeros(1, M2), ones(1, M1)];

theta1 = pi\*(N1-1)/N1\*(0:N1-1); % Correction here

Hk1 = A1 .\* exp(1i \* theta1);

h1 = real(ifft(Hk1));

[H1, w1] = freqz(h1, 1);

K1 = (N1-1) / 2;

wal = (0:K1-1) / K1;

subplot(2,2,1);

plot(wal, A1(1:K1), '-', w1/pi, abs(H1));

xlabel('频率(单位\pi)'); ylabel('H(e^{j\omega})');

subplot(2,2,3);

stem(n1, h1, '.'); title('实际滤波器的单位响应(N=33)');

xlabel('n'); ylabel('h1(n)');

%% N为63

N1 = 63;

n1 = 0:N1-1;

wc = pi/2;

M1 = fix(wc / (2\*pi/N1));

M2 = N1 - 2\*M1 - 1;

A1 = [ones(1, M1+1), zeros(1, M2), ones(1, M1)];

theta1 = pi\*(N1-1)/N1\*(0:N1-1); % Correction here

Hk1 = A1 .\* exp(1i \* theta1);

h1 = real(ifft(Hk1));

[H1, w1] = freqz(h1, 1);

K1 = (N1-1) / 2;

wal = (0:K1-1) / K1;

subplot(2,2,2);

plot(wal, A1(1:K1), '-', w1/pi, abs(H1));

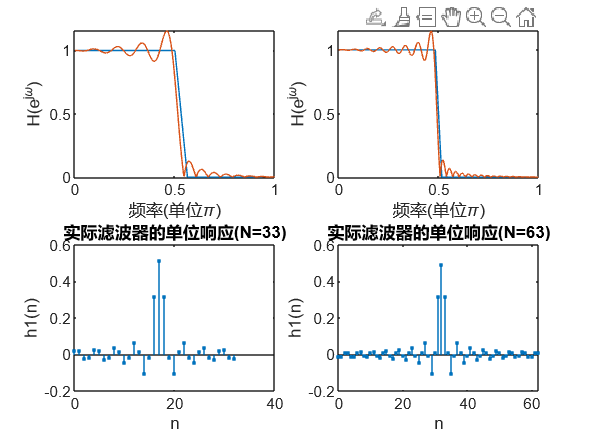
xlabel('频率(单位\pi)'); ylabel('H(e^{j\omega})');

subplot(2,2,4);

stem(n1, h1, '.'); title('实际滤波器的单位响应(N=63)');

xlabel('n'); ylabel('h1(n)');

实验结果：



（2）

实验思路：

首先，定义一些参数，包括滤波器的长度 N、频率截止点 wc，以及三个过渡点的值 T1、T2 和 T3。根据这些参数计算两个滤波器的频域响应。对于每个滤波器，定义了频域采样序列 A 和相位角度 theta，然后利用这些序列和角度计算了滤波器的频域响应 Hk。然后，利用逆离散傅立叶变换将频域响应转换为时域单位响应 h，并使用 freqz 函数计算了频率响应 H 和对应的频率向量 w。最后，将频率响应转换为以 dB 为单位的幅度响应 dbH，并绘制两个滤波器的幅频响应图。左图显示了绝对幅度，右图显示了以 dB 为单位的幅度。使用 subplot 函数将两个图放置在同一窗口中，以便进行比较。

实验代码：

%% 添加过渡点

N = 33;

n = 0:N-1;

wc = pi/2;

T1 = 0.3;

T2 = 0.6;

T3 = 0.1;

M1 = fix(wc/(2\*pi/N));

M2 = N - 2\*M1 - 1;

% 第一个滤波器

A1 = [ones(1, M1), T1, zeros(1, M2), T1, ones(1, M1-1)];

theta1 = -pi\*(0:N-1)\*(N-1)/N;

Hk1 = A1 .\* exp(1i \* theta1);

h1 = real(ifft(Hk1));

[H1, w1] = freqz(h1, 1);

% 第二个滤波器

A2 = [ones(1, M1), T2, T3, zeros(1, M2-2), T3, T2, ones(1, M1-1)];

theta2 = -pi\*(0:N-1)\*(N-1)/N;

Hk2 = A2 .\* exp(1i \* theta2);

h2 = real(ifft(Hk2));

[H2, w2] = freqz(h2, 1);

% 转换为dB单位

dbH1 = 20\*log10(abs(H1)/max(abs(H1)));

dbH2 = 20\*log10(abs(H2)/max(abs(H2)));

% 绘图

subplot(1,2,1);

plot(w1/pi, abs(H1), 'b', w2/pi, abs(H2), 'r');

grid on;

xlabel('频率(单位\pi)'); ylabel('H(e^{j\omega})');

title('幅频响应');

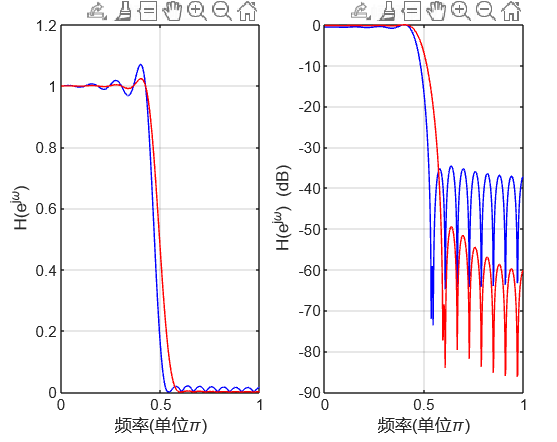
subplot(1,2,2);

plot(w1/pi, dbH1, 'b', w2/pi, dbH2, 'r');

grid on;

xlabel('频率(单位\pi)'); ylabel('H(e^{j\omega}) (dB)');

title('幅频响应 (dB)');



**3.利用计算机的声音编辑工具录制一段语音信号**

实验思路：

准备一段.wav格式的语音信号文件，并使用MATLAB读取该文件以获取原始语音信号。对原始语音信号进行时域波形和频谱分析，分别设计低通和高通滤波器，确定其截止频率和通带/阻带衰减要求，并利用MATLAB的滤波器设计函数进行设计。将设计好的滤波器应用于原始语音信号上，得到滤波后的语音信号，并观察滤波效果。最后，比较滤波前后语音信号的时域波形和频谱图，分析滤波效果及其对语音信号的影响。

实验代码：

clc;

clear;

% 读取语音信号

[x, fs] = audioread('/MATLAB Drive/燕山路161号 5.wav', [1, 2^16]);

x = x(:, 1)'; % 仅使用左声道

FS = length(x);

% 计算频谱

X = myfft(x);

% 时间向量

t = (0:FS-1) / fs;

% 绘制原始语音信号时域波形图

figure(1);

subplot(4,1,1);

plot(t, x);

title('原始语音信号时域波形');

xlabel('时间');

ylabel('幅值');

grid on;

% 绘制原始语音信号频谱

yy = abs(X);

subplot(4,1,2);

plot(yy(1:length(yy)/2));

axis([0 length(yy)/2 0 2000]);

title('原始语音信号频谱');

xlabel('频率');

ylabel('幅度');

grid on;

% 低通滤波器设计和应用

Fp = 1000;

Fs = 1200;

Ft = 130000;

As = 100;

Ap = 1;

wp = 2 \* pi \* Fp / Ft;

ws = 2 \* pi \* Fs / Ft;

[n, wn] = ellipord(wp, ws, Ap, As);

hn = fir1(n-1, wn, hann(n));

y1 = filter(hn, 1, x);

% 绘制低通滤波后语音信号时域波形图和频谱

subplot(4,1,3);

plot(t, y1);

title('低通滤波后信号时域波形');

xlabel('时间');

ylabel('幅值');

grid on;

yy = abs(fft(y1));

subplot(4,1,4);

plot(yy(1:length(yy)/2));

axis([0 length(yy)/2 0 2000]);

title('低通滤波后信号频谱');

xlabel('频率');

ylabel('幅度');

grid on;

% 绘制高通滤波后语音信号时域波形图和频谱

figure(2);

subplot(4,1,1:2);

plot(t, x);

title('原始语音信号时域波形');

xlabel('时间');

ylabel('幅值');

grid on;

yy = abs(X);

subplot(4,1,3);

plot(yy(1:length(yy)/2));

axis([0 length(yy)/2 0 2000]);

title('原始语音信号频谱');

xlabel('频率');

ylabel('幅度');

grid on;

% 高通滤波器设计和应用

Fp = 4800;

Fs = 5000;

[n, wn] = ellipord(2\*pi\*Fp/Ft, 2\*pi\*Fs/Ft, Ap, As);

hn = fir1(n-1, wn, 'high', hann(n));

y2 = filter(hn, 1, x);

% 绘制高通滤波后语音信号时域波形图和频谱

subplot(4,1,4);

plot(t, y2);

title('高通滤波后信号时域波形');

xlabel('时间');

ylabel('幅值');

grid on;

yy = abs(fft(y2));

subplot(4,1,4);

plot(yy(1:length(yy)/2));

axis([0 length(yy)/2 0 2000]);

title('高通滤波后信号频谱');

xlabel('频率');

ylabel('幅度');

grid on;

% 自定义FFT函数

function ret = myfft(x)

sz = size(x,2);

ret = zeros(1,size(x,2));

if (size(x,2) == 2)

ret = [x(1)+x(2), x(1)-x(2)];

return

end

odd = myfft(x(1:2:size(x,2)));

even = myfft(x(2:2:size(x,2)));

for n = 1:sz/2

ret(n)=odd(n)+even(n)\*W(sz, n-1);

ret(n+sz/2)=odd(n)-even(n)\*W(sz, n-1);

end

return

end

function ret = W(N, n)

ret = exp(-1i \* 2 \* pi / N \* n);

end

实验结果：

