哈尔滨理工大学

实　验　报　告

课程名称：语音信号处理

实验名称：短时时域分析

预习报告		过程及记录		结果分析		实验总结		总评	
教师签名：									

班　级　电信 21-1

学　号　2105040115

姓　名　刘云翔

指导教师　梁欣涛

2024 年 5 月 10 日

教务处 印制

实验名称	短时时域分析	时间	2024.5.10
		地点	E1205
同实验者	刘倚云，努尔森巴特·达吾提	班组	电信 21-115

一、 实验预习（准备）报告

1、实验目的

1、了解语音信号分帧与加窗的重要性和必要性。

2、掌握常用的窗函数和加窗分帧处理的原理。

3、能编程实现分帧函数，并恢复。

4、了解语音信号短时时域分析的原理。

5、掌握短时时域分析的一些参数计算方法。

6、根据原理能编程实现短时时域分析的参数计算。

2、实验相关原理及内容

语音信号的时域分析就是分析和提取语音信号的时域参数。语音信号本身就是时域信号，因而时域分析是最早使用，也是应用最广泛的一种分析方法，这种方法直接利用语音信号的时域波形。时域分析通常用于最基本的参数分析及应用，如语音的分割、预处理、分类等。语音信号的时域参数有短时能量、短时过零率、短时自相关函数和短时平均幅度差函数等。这些最基本的短时参数在各种语音信号数字处理技术中都有重要的应用。

1、语音分帧

贯穿于语音分析全过程的是“短时分析技术”。因为，语音信号从整体来看其特性及表征其本质特征的参数均是随时间而变化的，所以它是一个非平稳态过程，不能用处理平稳信号的数字信号处理技术对其进行分析处理。但是，由于不同的语音是由人的口腔肌肉运动构成声道某种形状而产生的响应，而这种口腔肌肉运动相对于语音频率来说是非常缓慢的，所以从另一方面看，虽然语音信号具有时变特性，但是在一个短时间范围内（一般认为在 10ms～30ms 的短时间内），其特性基本保持不变即相对稳定，因而可以将其看作是一个准稳态过程，即语音信号具有短时平稳性。所以任何语音信号的分析和处理必须建立在“短时” 的基础上，即进行“短时分析”，将语音信号分为一段一段来分析其特征参数，其中每一段称为一“帧”，帧长一般即取为 10~30ms。这样，对于整体的语音信号来讲，分析出的是由每一帧特征参数组成的特征参数时间序列。

分帧示意图如图 1-1 所示。

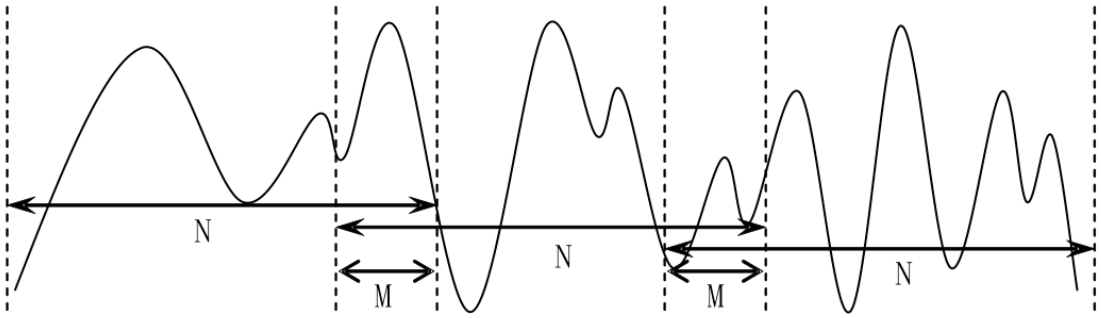


图 1-1 分帧示意图（N 为帧长，M 为帧移）

一般每秒的帧数约为 33~100 帧，视实际情况而定。分帧虽然可以采用连续分段的方法，但一般要采用如图 2-1 所示的交叠分段的方法，这是为了使帧与帧之间平滑过渡，保持其连续性。前一帧和后一帧的交叠部分称为帧移。帧移与帧长的比值一般取为 0～1/2。分帧是用可移动的有限长度窗口进行加权的方法来实现的，这就是用一定的窗函数。从而形成加窗，语音信号窗函数的选择（形状和

长度），对于短时分析参数的特性影响很大。为此应选择合适的窗口，使其短时参数更好地

反映语音信号的特性变化。选择的依据有两类：

1） 窗口的形状：一个好的窗函数的标准是：在时域因为是语音波形乘以窗函数，所以要减小时间窗两端的坡度，使窗口边缘两端不引起急剧变化而平滑过渡到零，这样可以使截取出的语音波形缓慢降为零，减小语音帧的截断效应；在频域要有较宽的 3dB 带宽以及较小的边带最大值。

2） 窗口的长度：如果长度很大，则它等效于很窄的低通滤波器，语音信号通过时，反映波形细节的高频部分被阻碍，短时能量随时间变化很小，不能真实的反映语音信号的幅度变化； 反之，长度太小时，滤波器的通带变宽，短时能量随时间有急剧的变化，不能得到平滑的能量函数。通常认为在一个语音帧内应包含 1~7 个基音周期。然而不同人的基音周期变化很大，从女性和儿童的 2ms 到老年男子的 14ms（即基音频率的变化范围为 500Hz~70Hz），所以 N 的选择比较困难。通常在 10kHz 取样频率下，N 折中选择为 100~200 点为宜（即 10~20ms 持续时间）。常见的窗函数如前一章所列。

2、短时能量

设第 n 帧语音信号的短时能量用 En 表示，则其计算公式如下：

$$E_n = \sum_{m=0}^{N-1} x_n^2(m)$$

是一个度量语音信号幅度值变化的函数，但它有一个缺陷，即它对高电平非常敏感（因为它计算时用的是信号的平方）。

3、实验方法及步骤设计

1. 根据语音分帧的思想，编写分帧函数。函数定义如下：

函数格式：frameout=enframe(x,win,inc)

输入参数：x 是语音信号；win 是帧长或窗函数，若为窗函数，帧长便取窗函数长；inc 是帧移。

输出参数：frameout 是分帧后的数组，长度为帧长和帧数的乘积。

根据分帧后的语音，绘制连续四帧语音信号（不用窗函数）

2. 编程实现矩形窗、汉明窗和汉宁窗。

4、实验设备仪器选择及材料

PC 一台 Matlab 2021a 软件

5、实验思考题解答

1、编程实现将分帧加窗后的语音信号恢复成原始分帧前的信号。

代码如下：

```
% 加载音频文件
[audio, fs] = audioread(' C3_1_y.wav ');
% 将音频信号预处理为单通道（单声道）
if size(audio, 2) > 1
    audio = mean(audio, 2); % 将立体声转换为单声道
end
```

```
% 定义窗口和帧移
N = 256; % 窗口长度
M = 128; % 帧移
% 创建窗函数
rect_win = rectwin(N); % 矩形窗
hamming_win = hamming(N); % 汉明窗
hanning_win = hann(N); % 汉宁窗
% 分帧和加窗
frames_hamming = enframe(audio, hamming_win, M);
% 计算短时能量
energy_hamming = short_time_energy(frames_hamming);
% 计算短时自相关
autocorr_hamming = short_time_autocorr(frames_hamming);
% 绘制窗函数时域波形
figure;
subplot(3,1,1);
plot(rect_win);
title('矩形窗');
subplot(3,1,2);
plot(hamming_win);
title('汉明窗');
subplot(3,1,3);
```

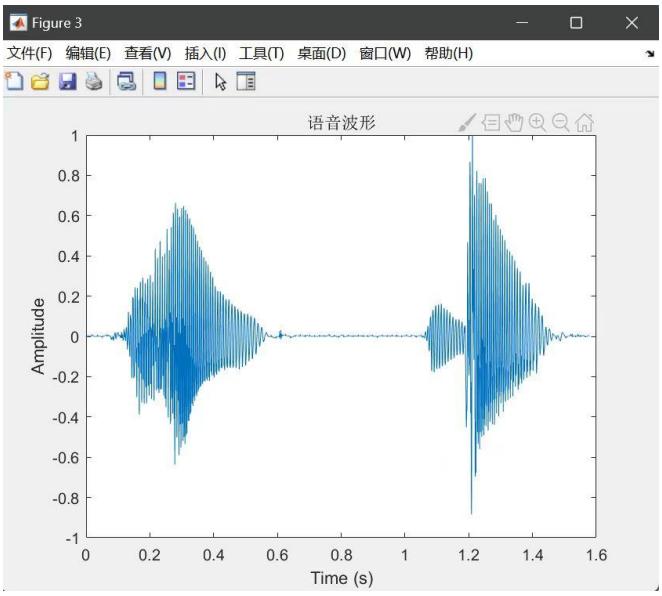
```
plot(hanning_win);
title('汉宁窗');
% 绘制连续四帧语音信号
figure;
for i = 1:4
    subplot(4,1,i);
    plot(frames_hamming(7+i-1, :));
    title(['第 ', num2str(7+i-1), ' 帧']);
    xlabel('样点');
    ylabel('幅度');
end
% 绘制第 10 帧的短时自相关
figure;
plot(autocorr_hamming(:, 10)); % 可以选择其他帧来展示
title('第 10 帧的短时自相关');
xlabel('滞后');
ylabel('自相关值');

% 定义函数 enframe
function frames = enframe(signal, window, hop)
    signal_length = length(signal);
    window_length = length(window);
```

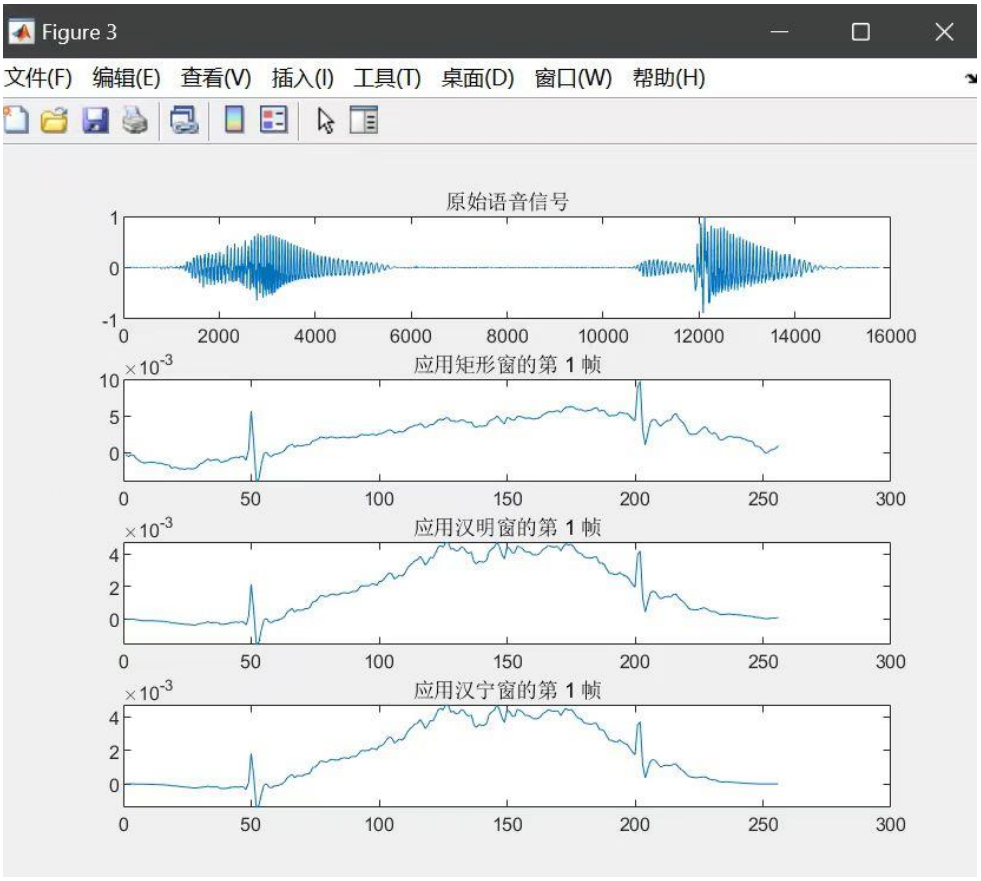
```
    num_frames = 1 + floor((signal_length - window_length) / hop);
    frames = zeros(num_frames, window_length);

    for i = 1:num_frames
        start = (i - 1) * hop + 1;
        frames(i, :) = signal(start:start + window_length - 1) .* window;
    end
end

% 定义函数 short_time_energy
function energy = short_time_energy(frames)
    energy = sum(frames.^2, 2); % 对每一帧计算能量
end
% 定义函数 short_time_autocorr
function autocorr = short_time_autocorr(frames)
    [numFrames, frameSize] = size(frames);
    autocorr = zeros(frameSize * 2 - 1, numFrames); % 调整大小以容纳完整的自相关输出
    for i = 1:numFrames
        autocorr(:, i) = xcorr(frames(i, :), 'biased');
    end
end
```



2、编程比较不同的窗函数对短时时域参数估计的影响。



代码如下：

```
% 加载音频文件

[audio, fs] = audioread('C2_6_y.wav');
```

```
% 将音频信号预处理为单通道（单声道）
if size(audio, 2) > 1
    audio = mean(audio, 2); % 将立体声转换为单声道
end
% 定义窗口和帧移
N = 256; % 窗口长度
M = 128; % 帧移
% 创建窗函数
rect_win = rectwin(N); % 矩形窗
hamming_win = hamming(N); % 汉明窗
hanning_win = hann(N); % 汉宁窗
% 分帧和加窗
frames_rect = enframe(audio, rect_win, M);
frames_hamming = enframe(audio, hamming_win, M);
frames_hanning = enframe(audio, hanning_win, M);
% 计算短时能量
energy_hamming = short_time_energy(frames_hamming);
% 计算短时自相关
autocorr_hamming = short_time_autocorr(frames_hamming);
% 绘制原始信号和加窗后的信号对比
figure;
subplot(4,1,1);
```

```
plot(audio);
title('原始语音信号');
subplot(4,1,2);
plot(frames_rect(1, :));
title('应用矩形窗的第 1 帧');
subplot(4,1,3);
plot(frames_hamming(1, :));
title('应用汉明窗的第 1 帧');
subplot(4,1,4);
plot(frames_hanning(1, :));
title('应用汉宁窗的第 1 帧');
% 绘制窗函数时域波形
figure;
subplot(3,1,1);
plot(rect_win);
title('矩形窗');
subplot(3,1,2);
plot(hamming_win);
title('汉明窗');
subplot(3,1,3);
plot(hanning_win);
title('汉宁窗');
```

```
% 绘制短时自相关
figure;
plot(autocorr_hamming(:, 10)); % 可以选择其他帧来展示
title('第 10 帧的短时自相关');
xlabel('滞后');
ylabel('自相关值');
% 定义和使用函数
function frames = enframe(signal, window, hop)
    signal_length = length(signal);
    window_length = length(window);
    num_frames = 1 + floor((signal_length - window_length) / hop);
    frames = zeros(num_frames, window_length);
    for i = 1:num_frames
        start = (i - 1) * hop + 1;
        frames(i, :) = signal(start:start + window_length - 1) .* window;
    end
end

function energy = short_time_energy(frames)
    energy = sum(frames.^2, 2); % 对每一帧计算能量
end
```

```
function autocorr = short_time_autocorr(frames)
    [numFrames, frameSize] = size(frames);
    autocorr = zeros(frameSize * 2 - 1, numFrames); % 调整大小以容纳完整的自相关输出
    for i = 1:numFrames
        autocorr(:, i) = xcorr(frames(i, :), 'biased');
    end
end
```

二、 实验过程及记录

语言信号分帧：

代码如下：

```
clc
clear all
close all
[x,Fs]=audioread('C3_1_y.wav');            % 读入数据文件
wlen=200; inc=100;                  % 给出帧长和帧移
N=length(x);                              % 信号长度
time=(0:N-1)/Fs;                          % 计算出信号的时间刻度
signal=enframe(x,wlen,inc)';        % 分帧
i=input('请输入起始帧号(i):');
tlabel=i;
subplot 411; plot((tlabel-1)*inc+1:(tlabel-1)*inc+wlen,signal(:,tlabel),'b'); axis tight%
```

画出时间波形

```
xlim([(i-1)*inc+1 (i+2)*inc+wlen])
ylim([-0.1,0.1])
title(['(a)当前波形帧号：', num2str(i)]);
ylabel('幅值'); xlabel('帧长');
tlabel=i+1;
subplot 412; plot((tlabel-1)*inc+1:(tlabel-1)*inc+wlen,signal(:,tlabel),'b'); axis tight%
```

画出时间波形

```
xlim([(i-1)*inc+1 (i+2)*inc+wlen])
ylim([-0.1,0.1])
title(['(b)当前波形帧号：', num2str(i+1)]);
ylabel('幅值'); xlabel('帧长');
tlabel=i+2;
subplot 413; plot((tlabel-1)*inc+1:(tlabel-1)*inc+wlen,signal(:,tlabel),'b'); axis tight%
```

画出时间波形

```
xlim([(i-1)*inc+1 (i+2)*inc+wlen])
ylim([-0.1,0.1])
title(['(c)当前波形帧号：', num2str(i+2)]);
ylabel('幅值'); xlabel('帧长');
tlabel=i+3;
subplot 414; plot((tlabel-1)*inc+1:(tlabel-1)*inc+wlen,signal(:,tlabel),'b'); axis tight%
```

画出时间波形

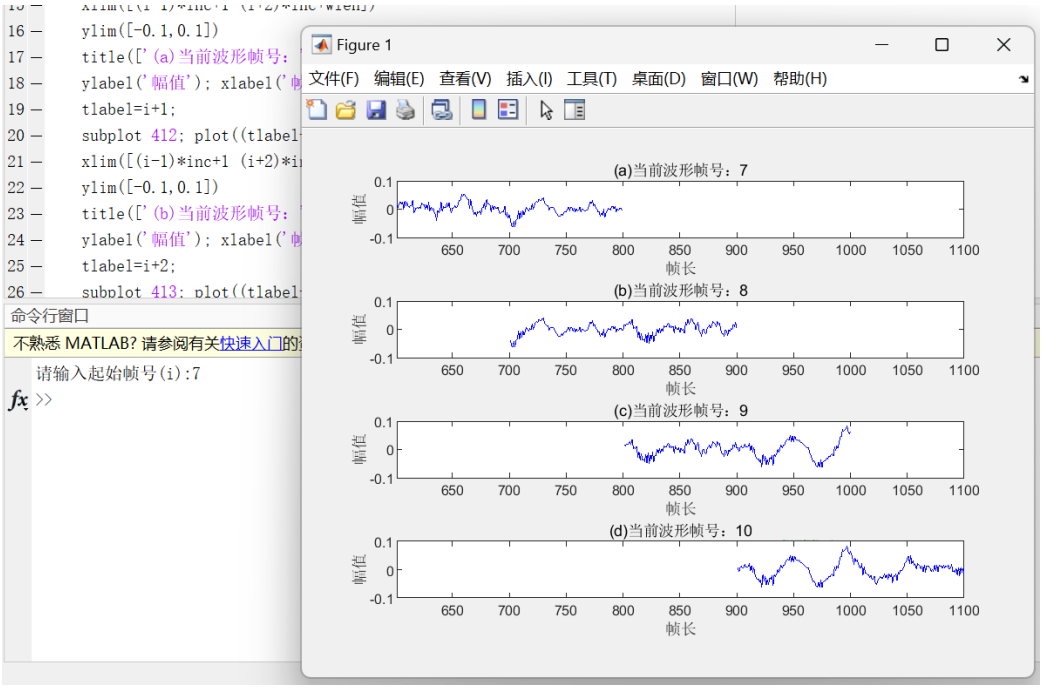
xlim([(i-1)*inc+1 (i+2)*inc+wlen])

ylim([-0.1,0.1])

title(['(d)当前波形帧号：', num2str(i+3)]);

ylabel('幅值'); xlabel('帧长');

实验结果：



实现矩形窗、汉明窗和汉宁窗：

代码如下：

clc

clear all

close all

N=32;nn=0:(N-1);

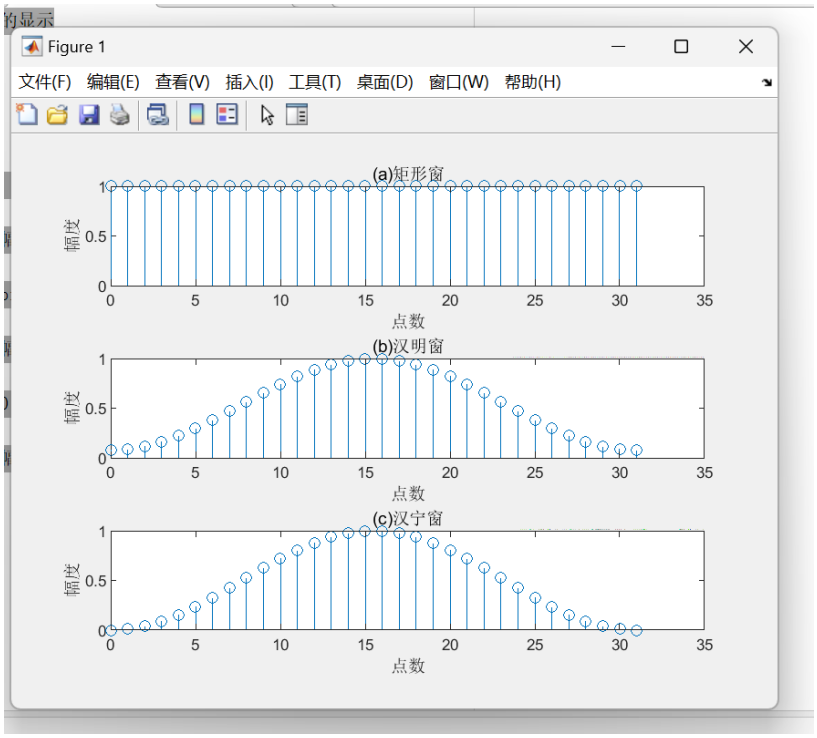
subplot(311);

w = ones(N,1);　　%矩形窗实现

stem(nn,w)

```
xlabel('点数');ylabel('幅度');title('(a)矩形窗')
subplot(312);
  w = 0.54 - 0.46*cos(2*pi*(0:N-1)'/(N-1));          %汉明窗实现
stem(nn,w)
xlabel('点数');ylabel('幅度');title('(b)汉明窗')
subplot(313)
w = 0.5*(1 - cos(2*pi*(0:N-1)'/(N-1)));          %汉宁窗实现
stem(nn,w)
xlabel('点数');ylabel('幅度');title('(c)汉宁窗')
```

实验结果：



实现 FrameTimeC 函数：

代码如下：

```
function frameout=enframe(x,win,inc)
```

```
nx=length(x(:));                    % 取数据长度
nwin=length(win);                   % 取窗长
if (nwin == 1)                      % 判断窗长是否为 1，若为 1，即表示没有设窗函数
    len = win;                      % 是，帧长=win
else
    len = nwin;                     % 否，帧长=窗长
end
if (nargin < 3)                     % 如果只有两个参数，设帧 inc=帧长
    inc = len;
end
nf = fix((nx-len+inc)/inc); % 计算帧数
frameout=zeros(nf,len);             % 初始化
indf= inc*(0:(nf-1)).';         % 设置每帧在 x 中的位移量位置
inds = (1:len);                     % 每帧数据对应 1:len
frameout(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));     % 对数据分帧
if (nwin > 1)                       % 若参数中包括窗函数，把每帧乘以窗函数
    w = win(:)';                    % 把 win 转成行数据
    frameout = frameout .* w(ones(nf,1),:);   % 乘窗函数
end
```

实现短时能量、短时平均幅度和短时过零率：

代码如下：

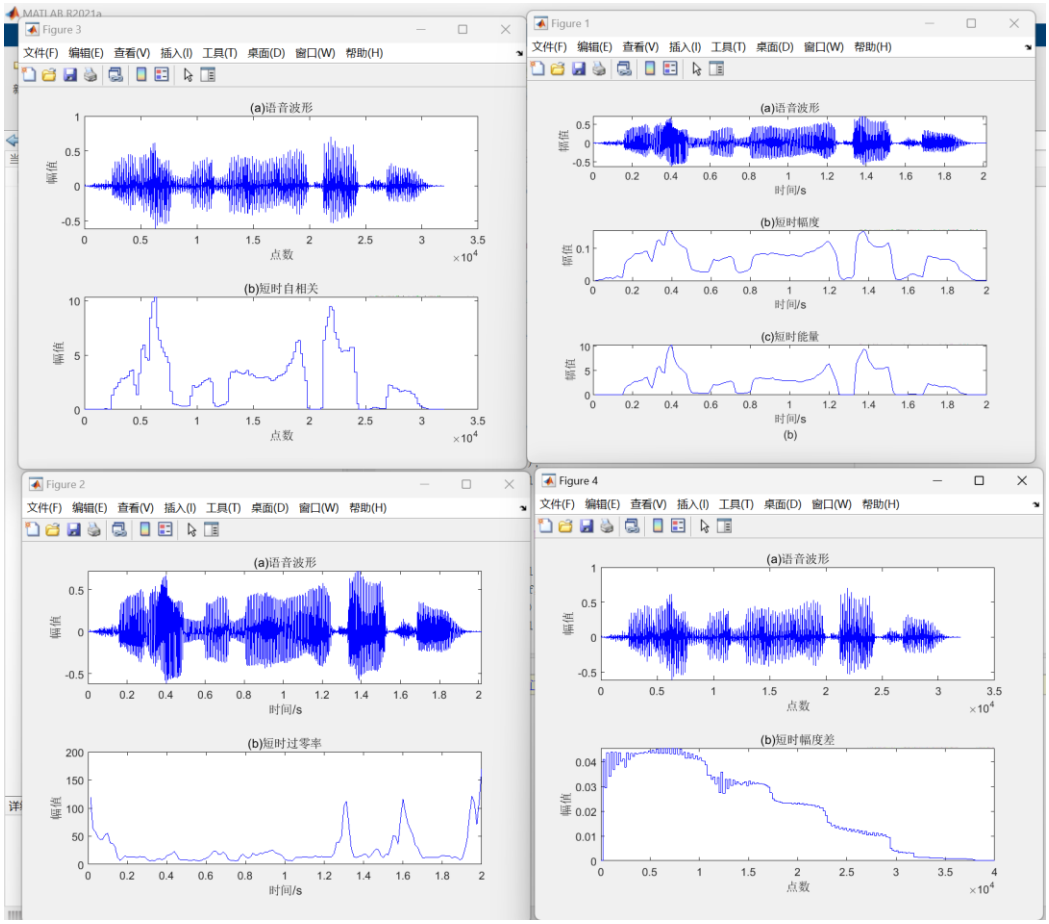
```
clear all; clc; close all;
```

```
[x,Fs]=audioread('C3_2_y.wav');            % 读入数据文件
wlen=200; inc=100;                  % 给出帧长和帧移
win=hanning(wlen);                  % 给出海宁窗
N=length(x);                        % 信号长度
time=(0:N-1)/Fs;                    % 计算出信号的时间刻度
En=STEn(x,win,inc);                 %短时能量
Mn=STMn(x,win,inc);                 %短时平均幅度
Zcr=STZcr(x,win,inc);               %短时过零率
%此处和上述 3 个参数不同，返回的不是向量而是矩阵，因为一帧信号得到的不是一个数值
X=enframe(x,win,inc)';        % 分帧
xn=X(:);
Ac=STAc(X);                         %计算短时自相关
Ac=Ac(:);
Amdf=STAmdf(X);                     %计算短时幅度差
Amdf=Amdf(:);
fn=length(En);                      % 求出帧数
figure(1)
subplot 311; plot(time,x,'b'); axis tight% 画出时间波形
title('(a)语音波形');
ylabel('幅值'); xlabel(['时间/s' 10 ]);
frameTime=FrameTimeC(fn,wlen,inc,Fs);     % 求出每帧对应的时间
```

```
subplot 312; plot(frameTime,Mn,'b')          %  画出短时幅度图
title('(b)短时幅度');
ylabel('幅值'); xlabel(['时间/s' 10 ]);
subplot 313; plot(frameTime,En,'b')          %  画出短时能量图
title('(c)短时能量');
ylabel('幅值'); xlabel(['时间/s' 10 '(b)']);
figure(2)
subplot 211; plot(time,x,'b'); axis tight%  画出时间波形
title('(a)语音波形');
ylabel('幅值'); xlabel(['时间/s' 10 ]);
subplot 212; plot(frameTime,Zcr,'b')         %  画出短时过零率图
title('(b)短时过零率');
ylabel('幅值'); xlabel(['时间/s' 10 ]);
figure(3)
subplot 211; plot(xn,'b'); %  画出时间波形
title('(a)语音波形');
ylabel('幅值'); xlabel(['点数' 10 ]);
subplot 212; plot(Ac,'b')                                %  画出短时自相关图
title('(b)短时自相关');
ylabel('幅值'); xlabel(['点数' 10 ]);
figure(4)
subplot 211; plot(xn,'b'); %  画出时间波形
```

```
title('(a)语音波形');
ylabel('幅值'); xlabel(['点数' 10 ]);
subplot 212; plot(Amdf,'b')                    %  画出短时幅度差
title('(b)短时幅度差');
ylabel('幅值'); xlabel(['点数' 10 ]);
```

实验结果：



三、 实验结果分析

通过短时时域分析，我们成功地将非平稳的语音信号划分为一系列短时平稳的帧，并通过不同的窗函数处理来减少截断效应。实验中，矩形窗、汉明窗和汉宁窗的实现展示了不同窗函数对信号平滑度的影响。短时能量、平均幅度、过零率等参数的计算，为我们提供了评估语音信号特性的重要指标。

实验结果表明，短时能量能够有效地反映出语音信号的幅度变化，而短时平均幅度和过零率则揭示了信号的局部特性。此外，短时自相关和幅度差函数的分析有助于进一步理解语音信号的周期性和动态变化。

四、实验总结（200-300 字）

在本次实验中，我们进行了短时时域分析的探索。我们了解到了语音信号加窗的重要性，掌握了矩形窗，汉明窗和汉宁窗等常用窗函数的原理及其对短时时域参数估计的影响。实验中，我们成功实现分帧函数，并通过 Matlab 编程计算了短时能量，短时过零率等关键参数。

实验结果显示，不同窗函数对信号处理效果有显著差异，其中汉宁窗因其平滑过渡特性，在减少截断效应方面表现较为出色，短时能量参数能够有效反映语音信号的幅度变化，而短时平均幅度和过零率则揭示了信号的局部特性。

总的来说，本次实验不仅加深了我们对语音信号处理的理解，而且通过实际操作提高了我们的实践能力。通过实验，我们学会了如何将理论知识应用于实际问题中，为今后的学习和研究打下了坚实基础。