南开大学
Nankai University

南　　开　　大　　学

网络空间安全学院

信息隐藏技术实验报告

实验 2：语音信号的常用处理方法上机实验

姓名：2113203 付政烨

年级：2021 级

专业：信息安全、法学

指导教师：李朝晖

2024 年 3 月 25 日

目录

一、 实验内容

- 学习慕课：2.2 语音信号处理基础
- FFT
- DWT
- DCT

在 matlab 中调试完成课堂上的例题，练习使用常用的语音信号处理方法。
要求：编程实现，提交实验报告。

二、 原始语音信号

在实验开始前，首先是对个人录制的语音信号进行采集与可视化表达。该语音信号的持续时间为 11 秒。如图 1 所展示，通过声波的形式可见，该信号包含了时间序列上的波动特性，是对声音物理特性的直观表达。在进行任何形式的信号处理或分析之前，对原始语音数据的这种展现是至关重要的，它为后续的信号处理提供了基础数据，并有助于初步了解语音信号的特征，例如频率范围、强度变化等。此外，通过对原始语音信号的可视化分析，可以对信号中可能存在的噪声、异常值或者是特定的语音模式有一个初步的认识，为进一步的信号处理技术的应用奠定基础。

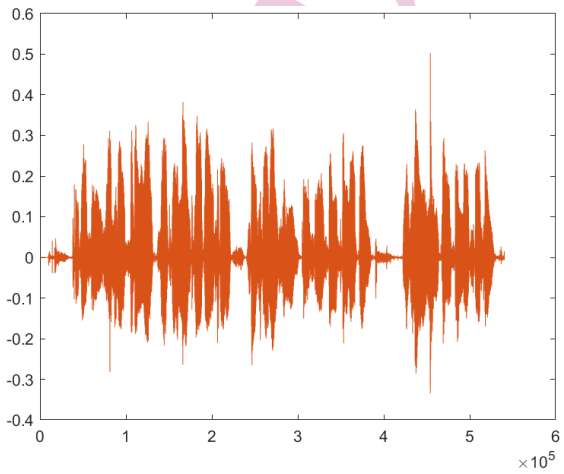

图 1

```
1 wav_name = 'fu_voice.wav';
2 wav = audioread(wav_name);
3 plot(wav);
```

三、 语音信号的傅氏变换

(一) FFT Matlab 介绍

```
1 %FFT Discrete Fourier transform.
2 % FFT(X) is the discrete Fourier transform (DFT) of vector X. For
```

```
% matrices, the FFT operation is applied to each column. For N-D
% arrays, the FFT operation operates on the first non-singleton
% dimension.
%
% FFT(X,N) is the N-point FFT, padded with zeros if X has less
% than N points and truncated if it has more.
%
% FFT(X,[],DIM) or FFT(X,N,DIM) applies the FFT operation across the
% dimension DIM.
%
% For length N input vector x, the DFT is a length N vector X,
% with elements
% N
% X(k) = sum x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
% n=1
% The inverse DFT (computed by IFFT) is given by
% N
% x(n) = (1/N) sum X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
% k=1
%
% See also FFT2, FFTN, FFTSHIFT, FFTW, IFFT, IFFT2, IFFTN.
% Copyright 1984-2005 The MathWorks, Inc.
% Built-in function
```

The 'i' in the 'Nth root of unity' 是虚数单位

调用：

1. Y = fft（y）；
2. Y = fft（y，N）；

式中，y 是序列，Y 是序列的快速傅里叶变换。y 可以是一向量或矩阵，若 y 为向量，则 Y 是 y 的 FFT，并且与 y 具有相同的长度。若 y 为一矩阵，则 Y 是对矩阵的每一列向量进行 FFT。

说明：

1. 函数 fft 返回值的数据结构具有对称性根据采样定理，fft 能分辨的最高频率为采样频率的一半（即 Nyquist 频率），函数 fft 返回值是以 Nyqusit 频率为轴对称的，Y 的前一半与后一半是复数共轭关系。
2. 幅值作 FFT 分析时，幅值大小与输入点数有关，要得到真实的幅值大小，只要将变换后的结果乘以 2 除以 N 即可（但此时零频—直流分量—的幅值为实际值的 2 倍）。对此的解释是：Y 除以 N 得到双边谱，再乘以 2 得到单边谱（零频在双边谱中本没有被一分为二，而转化为单边谱过程中所有幅值均乘以 2，所以零频被放大了）。
3. 基频若分析数据时长为 T，则分析结果的基频就是 f0=1/T，分析结果的频率序列为 [0:N-1]*f0
4. 执行 N 点 FFT 在调用格式 2 中，函数执行 N 点 FFT。若 y 为向量且长度小于 N，则函数将 y 补零至长度 N，若向量 y 的长度大于 N，则函数截断 y 使之长度为 N。

注意：

使用 N 点 FFT 时，若 N 大于向量 y 的长度，将给频谱分析结果带来变化，应该特别注意。傅立叶原理表明：任何连续测量的时序或信号，都可以表示为不同频率的余弦（或正弦）波信号的无限叠加。FFT 是离散傅立叶变换的快速算法，可以将一个信号变换到频域。

1. 有些信号在时域上是很难看出什么特征的，但是如果变换到频域之后，就很容易看出特征（频率，幅值，初相位）；
2. FFT 可以将一个信号的频谱提取出来，进行频谱分析，为后续滤波准备；
3. 通过对一个系统的输入信号和输出信号进行快速傅里叶变换后，两者进行对比，对系统可以有一个初步认识。

(二) FFT 处理示例

图二展现的是一个音频信号通过快速傅立叶变换（FFT）后的结果。快速傅立叶变换是一种算法，用于计算一个信号的离散傅立叶变换（DFT）及其逆变换，这在信号处理领域有着广泛的应用。其主要用途在于分析一个信号中各个不同频率成分的幅度或强度。

在该图中，横轴（x 轴）代表频率，单位为赫兹（Hz），而纵轴（y 轴）则表示对应于每个频率成分的幅度或强度。图中的尖峰表示存在一个在特定频率下具有显著强度的频率成分，即此频率成分在整个信号中占据主导地位。由图所示，该音频信号经 FFT 处理后表明其能量主要集中在 20kHz 至 40kHz 的频率范围内。

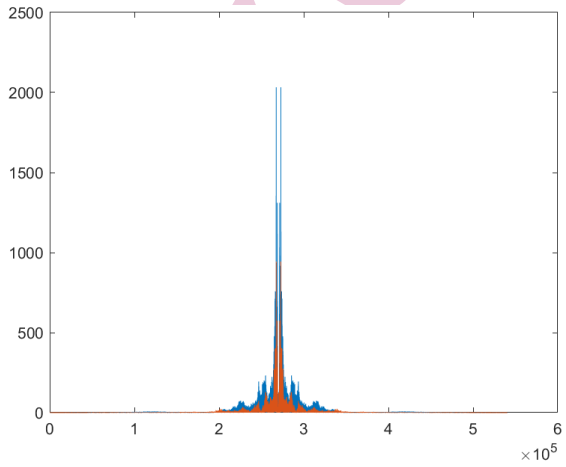

图 2

```
wav_name = 'fu_voice.wav';
wav = audioread(wav_name);
f_wav = fft(wav);
plot(abs(fftshift(f_wav)));
```

四、 语言信号的小波变换

(一) DWT Matlab 介绍

```
function [a,d] = dwt(x,varargin)
% a:信号的近似 d:信号的分解

%DWT Single-level discrete 1-D wavelet transform.
% DWT performs a single-level 1-D wavelet decomposition 信号的
% with respect to either a particular wavelet ('wname',
% see WFILTERS for more information) or particular wavelet filters
% (Lo_D and Hi_D) that you specify.
%
% [CA,CD] = DWT(X,'wname') computes the approximation
% coefficients vector CA and detail coefficients vector CD,
% obtained by a wavelet decomposition of the vector X.
% 'wname' is a character vector containing the wavelet name.
%
% [CA,CD] = DWT(X,Lo_D,Hi_D) computes the wavelet decomposition
% as above given these filters as input:
% Lo_D is the decomposition low-pass filter.
% Hi_D is the decomposition high-pass filter.
% Lo_D and Hi_D must be the same length.
%
% Let LX = length(X) and LF = the length of filters; then
% length(CA) = length(CD) = LA where LA = CEIL(LX/2),
% if the DWT extension mode is set to periodization.
% LA = FLOOR((LX+LF-1)/2) for the other extension modes.
% For the different signal extension modes, see DWTMODE.
%
% [CA,CD] = DWT(...,'mode',MODE) computes the wavelet
% decomposition with the extension mode MODE you specify.
% MODE is a character vector containing the extension mode.
%
% Example:
% x = 1:8;
% [ca,cd] = dwt(x,'db1','mode','sym')
%
% See also DWTMODE, IDWT, WAVEDEC, WAVEINFO.

% M. Misiti, Y. Misiti, G. Oppenheim, J.M. Poggi 12-Mar-96.
% Last Revision: 06-Feb-2011.
% Copyright 1995-2015 The MathWorks, Inc.
```

小波变换时 20 世纪 80 年代中后期逐渐发展起来的一种数学分析方法，他一出现就受到数学界和工程界的广泛重视。1984 年法国科学家 J.Molet 在分析地震波的局部特性时，首先用小波变换对信号进行分析，并提出小波这一术语。

小波，小的波形，小是指其具有衰减性，波是指其具有波动性，即小波的振幅具有振幅正负相间的震荡形式。小波理论采用多分辨率思想，非均匀的划分时频空间，它使信号仍能在一组正交基上进行分解，为非平稳信号的分析提供了新途径。

小波就是在函数空间的一个满足条件的函数或者信号。小波分析能够对函数和信号进行任意指定点处的任意精细结构的分析，同时，这也决定了小波分析在对非平稳信号进行时频分析时，

具有对时频同时局部化的能力。

连续小波的时频窗时时频平面上一个可变的矩形，他的时频窗的面积与小波的母函数有关，这一点决定了小波变换在信号的时频分析中的特殊作用。

小波分析特点：

小波变换的时频关系受到不确定性原理的制约。还有恒 Q 性质，Q 为母小波的品质因数。Q= 带宽/中心频率。恒 Q 性质是小波变换的一个重要性质，也是小波变换区别于其他类型的变换，且被广泛应用的一个重要原因。当用较小的 a 对信号做高频分析时，实际上使用高频小波对信号做细致观察；而用较大的 a 对信号做低频分析时，实际上使用低频小波对信号做概貌观察。小波分析师傅里叶分析的发展和拓展，区别是:

1. 傅里叶变换用到的基本函数具有唯一性，小波分析用到的函数具有不唯一性，同样一个问题用不同的小波函数进行分析，有事结果相差甚远。
2. 在频域中，傅里叶变换具有较好的局部化能力，特别是对于那些频率成分比较简单的确定信号，傅里叶变换可以很容易的把信号表示成各种频率成分叠加和的形式；但在时域中，傅里叶变换没有局部化能力，无法从信号的傅里叶变换中看出原信号在任一时间点附近的形态。
3. 若用信号通过滤波器来解释，小波变换与短时傅里叶变换的不同之处在于，对短时傅 5 / 10 里叶变换来说，带通滤波器的带宽与中心频率无关；相反，小波变换带通滤波器的带宽则正比于中心频率，即小波变换对应的滤波器有一个恒定的相对带宽。

(二) DWT 处理示例

1. 一级小波分解 (DWT)

小波基采用 Daubechies-4 小波，(1) 是原始语音信号，(2) 是一级分解的细节分量，(3) 是一级分解的近似分量，分解后数据长度缩减一半。(4) 是一级分解重构的结果。

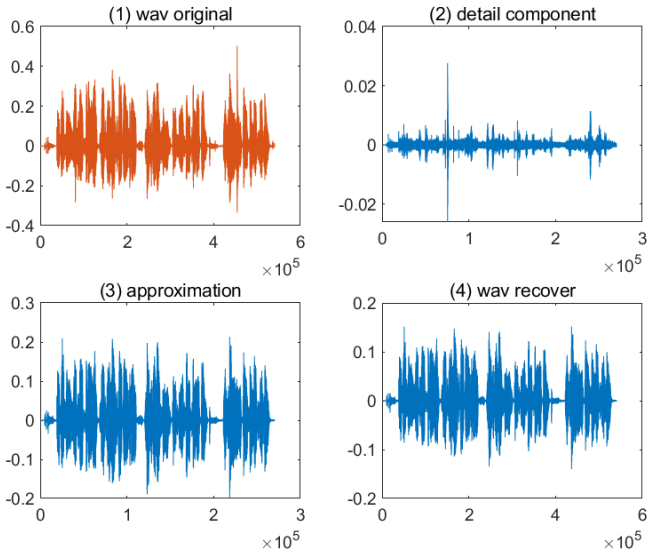

图 3

上述代码实现了对音频文件的小波变换（Discrete Wavelet Transform, DWT）处理与可视化展示。首先，通过 audioread 函数读取音频文件，获取其数据及采样率，并将音频信号转换为

向量格式。接着，利用 Daubechies-4 小波基，对音频信号进行小波变换，提取出不同级别的近似和细节系数。通过逆小波变换重构音频信号，并在一个图形窗口中绘制原始音频信号、各级细节系数、近似系数和重构音频信号的波形图，从而实现音频信号的处理和分析的可视化展示。

```
wav_name = 'fu_voice.wav';
[wav,fs] = audioread(wav_name); % 获得语音信号
len = length(wav); % 语音信号大小
wav_vec = zeros(len,1); % 预分配内存
for i = 1 : len % 语音信号转向量
    wav_vec(i) = wav(i);
end
[ca1,cd1] = dwt(wav_vec,'db4'); % 小波基选用 Daubechies-4 小波
wav0 = idwt(ca1,cd1,'db4',len); % 逆 dwt
figure
subplot(2,2,1),plot(wav);
subplot(2,2,2),plot(cd1); % 细节分量
subplot(2,2,3),plot(ca1); % 近似分量
subplot(2,2,4),plot(wav0);
axes_handle = get(gcf, 'children');
axes(axes_handle(4)); title('(1) wav original');
axes(axes_handle(3)); title('(2) detail component');
axes(axes_handle(2)); title('(3) approximation');
axes(axes_handle(1)); title('(4) wav recover');
saveas(gcf, 'waveforms2.png');
```

2. 一级小波分解 (WAVEDEC)

小波基采用 Daubechies-4 小波，(1) 是原始语音信号，(2) 是一级分解的细节分量，(3) 是一级分解的近似分量，(4) 是一级分解重构的结果。

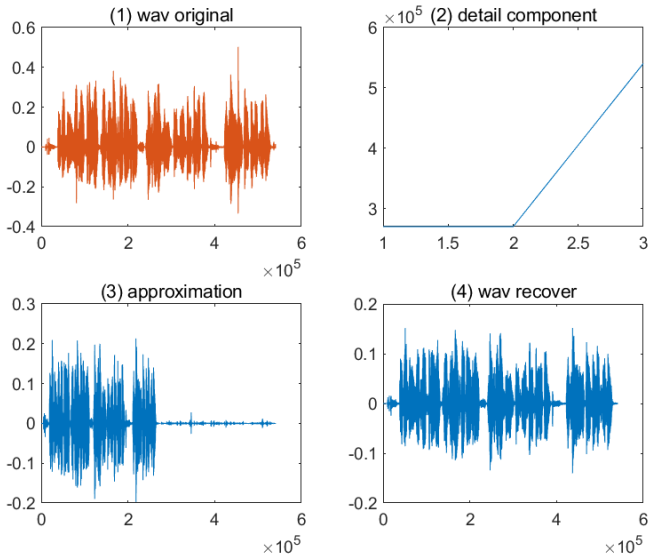

图 4

代码首先通过 audioread 函数读取一个名为 dida.wav 的语音文件，并将其内容存储在变量

中。随后，代码通过预分配内存和循环将语音信号转换为向量形式，尽管这一步骤实质上是多余的，因为原始数据已经是向量格式。接下来，采用 Daubechies-4 小波对语音信号执行一级小波分解，得到近似分量和细节分量。基于这些分量，再通过小波重构函数恢复语音信号，以便与原始信号进行比较。

```
wav_name = 'fu_voice.wav';
[wav,fs] = audioread(wav_name); % 获得语音信号
len = length(wav); % 语音信号大小
wav_vec = zeros(len,1); % 预分配内存
for i = 1 : len % 语音信号转向量
    wav_vec(i) = wav(i);
end
[ca1,cd1] = wavedec(wav_vec,1,'db4'); % 小波基选用 Daubechies-4 小波
wav0 = waverec(ca1,cd1,'db4'); % 逆 wavedec
figure
subplot(2,2,1),plot(wav);
subplot(2,2,2),plot(cd1); % 细节分量
subplot(2,2,3),plot(ca1); % 近似分量
subplot(2,2,4),plot(wav0);
axes_handle = get(gcf, 'children');
axes(axes_handle(4)); title('(1) wav original');
axes(axes_handle(3)); title('(2) detail component');
axes(axes_handle(2)); title('(3) approximation');
axes(axes_handle(1)); title('(4) wav recover');
saveas(gcf, 'waveforms2.png');
```

3. 三级小波分解 (WAVEDEC)

小波基采用 Daubechies-4 小波，(1) 是原始语音信号，(2) 是三级分解的细节分量，(3) 是一级分解的近似分量，(4) 是二级分解的近似分量，(5) 是三级分解的近似分量，(4) 是三级分解重构的结果。

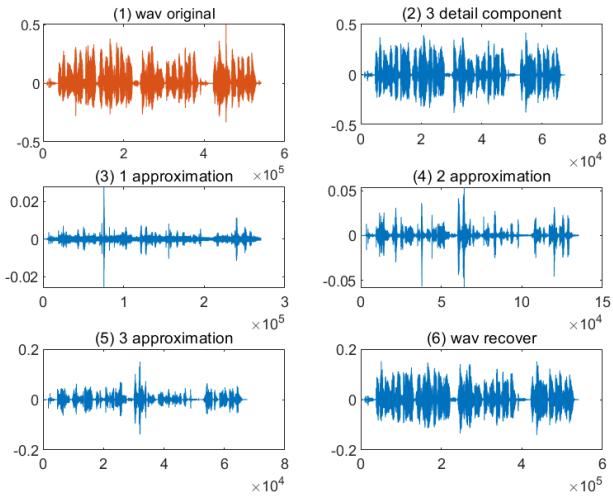

图 5

代码的主要是对指定的 WAV 格式的语音文件（在本例中为"dida.wav"）进行小波分解和重构处理。首先，代码通过 audioread 函数读取语音文件，获取语音信号及其采样频率。随后，它将这个一维语音信号转换为向量形式，并对该向量执行三级小波分解，使用的小波基是 Daubechies-4（db4）。分解过程产生三个不同级别的细节分量（cd1、cd2、cd3）和一个三级近似分量（ca3）。之后，代码利用这些分量执行小波重构，得到重构后的语音信号。为了可视化这一过程和结果，代码生成了一个图形界面，其中包含六个子图。这些子图分别展示了原始的语音信号、三级近似分量、三个不同级别的细节分量，以及最终通过小波重构得到的语音信号。

```
wav_name = 'fu_voice.wav';
[wav,fs] = audioread(wav_name); % 获得语音信号
len = length(wav); % 语音信号大小
wav_vec = zeros(len,1); % 预分配内存
for i = 1 : len % 语音信号转向量
    wav_vec(i) = wav(i);
end
[c,l] = wavedec(wav_vec,3,'db4'); % 小波基选用 Daubechies-4 小波
ca3 = appcoef(c,l,'db4',3); % 三级分解近似分量
cd3 = detcoef(c,l,3); % 三级分解细节分量
cd2 = detcoef(c,l,2); % 二级分解细节分量
cd1 = detcoef(c,l,1); % 一级分解细节分量
wav0 = waverec(c,l,'db4'); % 逆 dwt
figure
subplot(3,2,1),plot(wav);
subplot(3,2,2),plot(ca3); % 三级分解细节分量
subplot(3,2,3),plot(cd1); % 一级分解近似分量
subplot(3,2,4),plot(cd2); % 二级分解近似分量
subplot(3,2,5),plot(cd3); % 三级分解近似分量
subplot(3,2,6),plot(wav0); % 三级重构
axes_handle = get(gcf, 'children');
axes(axes_handle(6)); title('(1) wav original');
axes(axes_handle(5)); title('(2) 3 detail component');
axes(axes_handle(4)); title('(3) 1 approximation');
axes(axes_handle(3)); title('(4) 2 approximation');
axes(axes_handle(2)); title('(5) 3 approximation');
axes(axes_handle(1)); title('(6) wav recover');
saveas(gcf, 'waveforms3.png');
```


五、 语音信号的 DCT 变换

(一) DCT Matlab 介绍

```
function b=dct(a,varargin)
%DCT Discrete cosine transform.
% Y = DCT(X) returns the discrete cosine transform of vector X.
% If X is a matrix, the DCT operation is applied to each
% column. For N-D arrays, DCT operates on the first non-singleton
% dimension. This transform can be inverted using IDCT.
%
```

```
% Y = DCT(X,N) pads or truncates the vector X to length N
% before transforming.
%
% Y = DCT(X,[],DIM) or Y = DCT(X,N,DIM) applies the DCT operation along
% dimension DIM.
%
% Y = DCT(...,'Type',K) specifies the type of discrete cosine transform
% to compute. K can be one of 1, 2, 3, or 4, to represent the DCT-I,
% DCT-II, DCT-III, and DCT-IV transforms, respectively. The default
% value for K is 2 (the DCT-II transform).
%
% % Example:
% % Find how many DCT coefficients represent 99% of the energy
% % in a sequence.
%
% x = (1:100) + 50*cos((1:100)*2*pi/40); % Input Signal
% X = dct(x); % Discrete cosine transform
% [XX,ind] = sort(abs(X)); ind = fliplr(ind);
% num_coeff = 1;
% while (norm([X(ind(1:num_coeff)) zeros(1,100-num_coeff)])/norm(X)<.99)
% num_coeff = num_coeff + 1;
% end;
% num_coeff
%
% See also FFT, IFFT, IDCT.

% Author(s): C. Thompson, 2-12-93
% S. Eddins, 10-26-94, revised
% Copyright 1988-2016 The MathWorks, Inc.

% References:
% 1) A. K. Jain, "Fundamentals of Digital Image
% Processing", pp. 150-153.
% 2) Wallace, "The JPEG Still Picture Compression Standard",
% Communications of the ACM, April 1991.
```

离散余弦变换（DCT）是一种重要的信号处理工具，经常用于语音信号处理中。在语音信号的分析和编码过程中，DCT 可以有效地将信号从时域转换到频域。以下是使用离散余弦变换处理语音信号的基本概念和步骤：

DCT 的作用：

- 数据压缩：DCT 有助于去除信号中的冗余信息，这对于数据压缩非常有用。在语音编码中，这可以帮助降低存储和传输的数据量。
- 频域分析：通过将语音信号从时域转换到频域，DCT 有助于分析信号的频率成分，这对于信号的分析和处理非常重要。

使用 DCT 处理语音信号的步骤：

- 信号切片：首先，将连续的语音信号分割成短时间帧。这是因为语音信号是非平稳的，而 DCT 适合处理近似平稳的信号。

- 窗函数：对每个信号帧应用窗函数，如汉宁窗或汉明窗，以减少帧的边缘效应。
- 应用 DCT：对每个窗口化的帧应用离散余弦变换。这将每个帧转换为一系列余弦基函数的系数，这些系数表示原始信号在频域中的能量分布。
- 能量打包：在许多情况下，语音信号的大部分能量都集中在 DCT 系数的较低部分。可以根据需要选择前 N 个系数，以代表原始信号的主要信息，实现数据压缩。
- 反变换：如果需要，可以通过逆离散余弦变换（IDCT）重新获得时域信号，用于重构或其他处理。

(二) DCT 处理示例

本部分是语音信号文件进行离散余弦变换（DCT）处理，并进行逆变换以恢复原始信号。首先，使用 audioread 函数读取“dida.wav”文件，获取语音信号及其采样率。通过计算信号的长度并初始化一个零向量 wav_vec，准备存储转换后的语音信号。通过一个循环，将读取的语音信号逐个元素地赋值到 wav_vec 中，将语音信号转换成向量形式。使用 dct 函数对转换后的向量进行离散余弦变换，然后使用 idct 函数对变换结果进行逆变换，以尝试恢复原始的语音信号。最后，利用 plot 函数和 subplot 功能，在一个图形窗口中绘制三个子图：原始的语音信号波形、进行 DCT 变换后的信号波形、以及通过逆 DCT 变换恢复的语音信号波形。

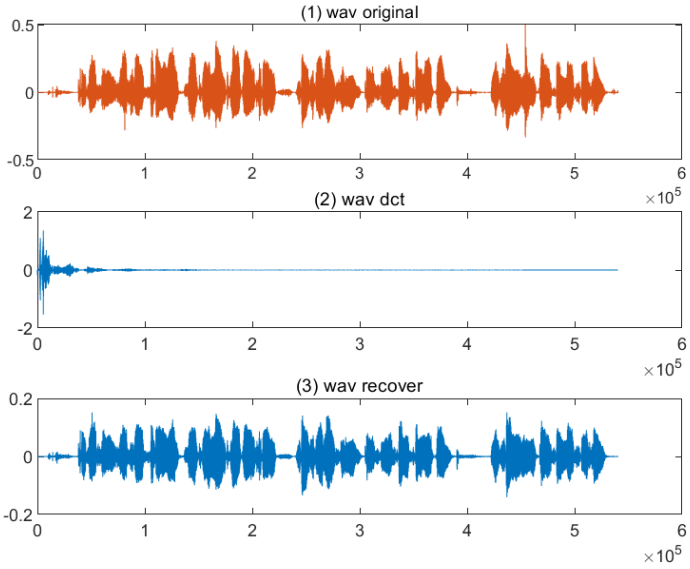

图 6

```
1 wav_name = 'fu_voice.wav';
2 [wav,fs] = audioread(wav_name); % 获得语音信号
3 len = length(wav); % 语音信号大小
4 wav_vec = zeros(len,1); % 预分配内存
5 for i = 1 : len % 语音信号转向量
6     wav_vec(i) = wav(i);
7 end
8 da = dct(wav_vec); % dct
```

```
wav0 = idct(da); % 逆 dct
figure
subplot(3,1,1),plot(wav);
subplot(3,1,2),plot(da); % dct
subplot(3,1,3),plot(wav0); % 重构
axes_handle = get(gcf, 'children');
axes(axes_handle(3)); title('(1) wav original');
axes(axes_handle(2)); title('(2) wav dct');
axes(axes_handle(1)); title('(3) wav recover');
saveas(gcf, 'waveforms4.png');
```

六、 扩展实验

(一) 线性预测编码 (LPC)

线性预测编码（LPC）是一种在信号处理，尤其是语音处理领域广泛使用的技术。它基于线性预测的概念，即将当前时刻的信号值作为前几个时刻信号值的线性组合来预测。LPC 广泛用于语音压缩和语音识别等领域，因为它能有效地表示语音信号的光谱特性。LPC 的基本思想是用过去的信号样本来预测当前的信号样本，并且用这种方式模拟人类声道的行为。具体来说，一个信号的当前样本可以看作是它之前的 P 个样本的线性组合，加上一个误差项。公式可以表示为:

$$s(n) = \sum_{k=1}^{P} a_k s(n-k) + e(n) \tag{1}$$

其中，$s(n)$ 是当前的信号样本，a_k 是预测系数（也就是我们要找的参数），P 是预测器的阶数（决定了将多少个过去的样本用于预测当前样本），而 $e(n)$ 是预测误差，也被称为激励信号。

LPC 分析过程

1. **信号预处理**：通常包括预加重步骤，用于平衡信号的频谱，使之在高频有更多的能量，这对于模拟人类声道的特性很重要。
2. **窗函数**：将信号分割成短时帧，通常每帧包含几十到几百个样本，并对每帧应用窗函数以减少帧边界的不连续。
3. **估计 LPC 参数**：通过最小化预测误差的能量来计算预测系数。这通常通过解线性方程组来实现，这些方程组由自相关函数或协方差函数的元素组成。
4. **计算误差（激励）信号**：使用找到的预测系数，从原始信号中去除预测部分，剩下的就是误差或激励信号。

```
wav_name = 'fu_voice.wav';
[wav, fs] = audioread(wav_name); % 获得语音信号
len = length(wav); % 语音信号的大小

% 线性预测编码分析
p = 12; % LPC的阶数，常用值为10至12，具体根据情况设定
[a, g] = lpc(wav, p); % LPC分析，a为预测系数，g为增益

% 使用LPC系数和增益来合成语音
```

```
est_wav = filter([0 -a(2:end)], 1, wav); % 根据LPC系数重建语音信号
est_wav = est_wav * sqrt(g); % 调整信号的能量，使其与原始信号匹配

% 可视化原始信号、LPC系数和重建的信号
figure
subplot(3,1,1), plot(wav), title('(1) 原始wav');
subplot(3,1,2), plot(a), title('(2) LPC系数');
subplot(3,1,3), plot(est_wav), title('(3) 重构wav');

% 保存可视化结果
saveas(gcf, 'lpc_waveforms.png');
```

上述 MATLAB 代码段实现了对一个语音信号的线性预测编码（LPC）分析和重构过程，并进行了可视化展示。代码首先从一个文件中读取语音信号，然后用设定阶数的 LPC 模型分析这个信号，提取出预测系数和增益。接着，利用这些 LPC 参数，结合原始信号，通过滤波器重建语音信号。最后，原始信号、LPC 系数、以及重建的信号被绘制在一张图上，分别展示，以便进行比较和分析。通过这一过程，用户可以直观地理解 LPC 模型如何对语音信号进行分析和重构，以及它在语音处理中的应用。

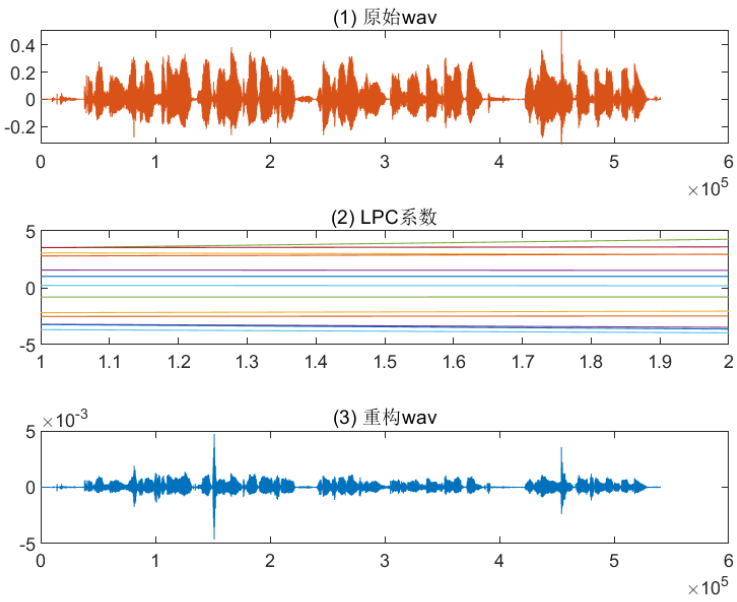

图 7

(二) 过零率（Zero Crossing Rate）

过零率（Zero Crossing Rate, ZCR）是音频和语音分析中常用的一个特征。它代表了信号从正变负或从负变正的频率。这个度量在检测语音信号中的有声和无声段落中被广泛使用。对于一个离散信号 $x[n]$，其过零率可以定义为：

$$ZCR = \frac{1}{2N} \sum_{n=1}^{N-1} |\text{sgn}(x[n]) - \text{sgn}(x[n-1])|, \tag{2}$$

其中 N 是样本总数，$\text{sgn}(\cdot)$ 是符号函数，用于判断信号的正负。

过零率的计算通常遵循以下步骤:

1. 对于给定的音频信号，将其分割成短时帧。
2. 对于每一帧，计算其过零次数。
3. 通过除以帧长来规范化这个计数，以获得过零率。

过零率在信号处理领域中有多种应用，包括：语音识别中的有声和无声段落的检测，音乐节奏和节拍检测以及声音质量评估等。

```
wav_name = 'fu_voice.wav'; % 语音文件名
[wav, fs] = audioread(wav_name); % 读取语音文件
len = length(wav); % 语音信号长度

% 计算过零率
zcr = zeros(len, 1); % 初始化过零率向量
for i = 2:len % 从第二个样本开始计算
    zcr(i) = zcr(i - 1) + 0.5 * abs(sign(wav(i)) - sign(wav(i - 1)));
end
zcr = zcr / fs; % 标准化过零率

% 可视化
figure;
subplot(3,1,1), plot(wav); % 原始语音信号
title('Original Wave');
xlabel('Time (samples)');
ylabel('Amplitude');

subplot(3,1,2), plot(zcr); % 过零率
title('Zero Crossing Rate');
xlabel('Time (samples)');
ylabel('Rate');

% 使用过零率进行简单语音活动检测
voice_activity = double(zcr > (mean(zcr) * 1.2)); % 使用过零率阈值进行判断
subplot(3,1,3), plot(voice_activity); % 可视化语音活动
title('Voice Activity Detection');
xlabel('Time (samples)');
ylabel('Activity (0=Silence, 1=Voice)');

% 保存可视化图像
saveas(gcf, 'zcr_voice_activity.png');
```

上述代码首先读取一个语音文件，并计算其长度。接着，它计算该语音信号的过零率（Zero Crossing Rate, ZCR），在语音信号中，过零率较高的区域往往对应于语音活动部分，而较低的区域可能表示静音或非语音部分。代码通过遍历语音信号的每个样本来计算过零率，并在完成后进行标准化处理。此外，代码还通过设置一个阈值，基于过零率来进行简单的语音活动检测，以区分语音和非语音部分。最后，代码将原始语音信号、计算得到的过零率以及语音活动检测的结果进行可视化，分别在三个子图中展示。

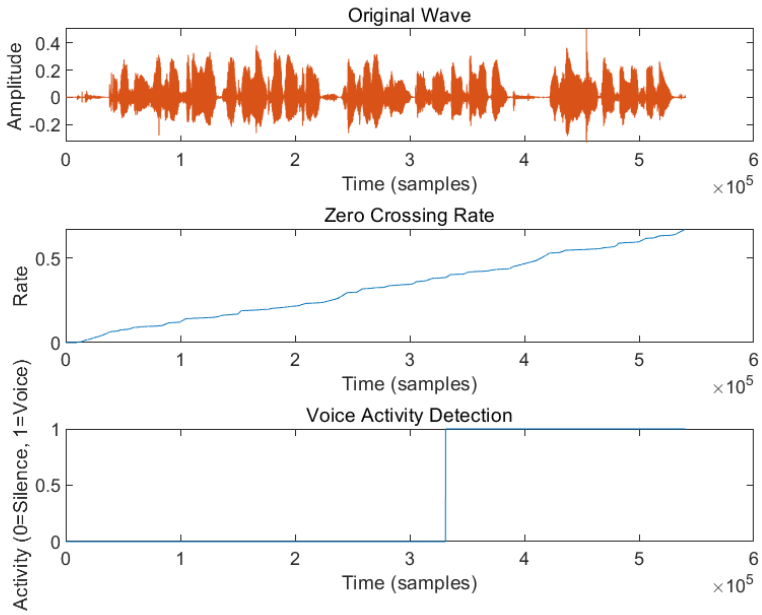

图 8

(三) 语音活动检测（VAD）

语音活动检测（VAD）用于判断音频信号中是否存在人声。它在多种应用中非常重要，例如在语音编码、语音识别和增强、以及通信系统中，帮助优化带宽使用和降低噪声。VAD 的基本原理是分析音频信号的特定特征，以区分语音和非语音（如静音、背景噪音等）段。这些特征包括：

- 能量水平：语音段通常比非语音段具有更高的能量。
- 频率内容：语音信号具有特定的频率范围，与静音或噪声不同。
- 零交叉率：这是信号正负变化的频率，语音段和非语音段在这个特征上表现不同。
- 谱特征：如梅尔频率倒谱系数（MFCC），这是在语音识别中广泛使用的一个特征。

实现方法

VAD 的实现可以根据简单的能量阈值到复杂的机器学习算法不等：

1. 基于能量的 VAD：通过比较音频帧的能量与预设阈值来判断是否为语音。
2. 基于频谱的 VAD：分析信号的频谱特性来区分语音和非语音。
3. 复合特征 VAD：结合多个声学特征，如能量、频率和零交叉率。
4. 基于模型的 VAD：使用机器学习模型（如神经网络）来识别语音段，这种方法通常需要大量的标记数据来训练模型。

VAD 的主要挑战包括噪音抑制、回声消除和在不同的环境条件下（如在街道、车内或室内）保持高性能。此外，不同人的语音差异、说话方式和口音也增加了 VAD 系统的复杂性。因此，设计高效且鲁棒的 VAD 系统是一个持续的研究课题，旨在提高语音通信和处理系统的总体性能和用户体验。

```
wav_name = 'fu_voice.wav';
[wav,fs] = audioread(wav_name); % 获得语音信号
len = length(wav); % 语音信号大小

% 简单的语音活动检测 (VAD)
frameLength = 0.03; % 以秒为单位的帧长度
frameSampleLength = floor(frameLength * fs);
numFrames = floor(len / frameSampleLength);
voiceSegments = zeros(len, 1); % 初始化 VAD 结果

for k = 1:numFrames
    frame = wav((k-1)*frameSampleLength+1:k*frameSampleLength);
    % 简单能量检测作为 VAD, 阈值需要根据您的信号进行调整
    if mean(frame.^2) > 0.01 % 假设的能量阈值
        voiceSegments((k-1)*frameSampleLength+1:k*frameSampleLength) = 1;
    end
end

% DCT 变换和逆变换
da = dct(wav); % DCT 变换
wav0 = idct(da); % 逆 DCT

% 绘制原始信号、DCT 和重构信号及 VAD 结果
figure;
subplot(4,1,1),plot(wav);
title('(1) Original wav');

subplot(4,1,2),plot(da);
title('(2) DCT of wav');

subplot(4,1,3),plot(wav0);
title('(3) Reconstructed wav');

subplot(4,1,4),plot(voiceSegments);
title('(4) Voice Activity Detection');

% 保存图像
saveas(gcf, 'waveforms_with_vad.png');
```

上述代码旨在处理和分析音频文件，即进行语音活动检测（VAD）和离散余弦变换（DCT）。首先，它通过 audioread 函数读取一个音频文件，然后计算其长度并初始化一个向量。随后，脚本执行 DCT 来分析音频信号的频谱特性，并使用逆 DCT 重建信号。在进行这些操作的同时，脚本应该执行语音活动检测。最后，脚本绘制并显示原始音频信号、DCT 变换后的信号和重构的音频信号的图表。

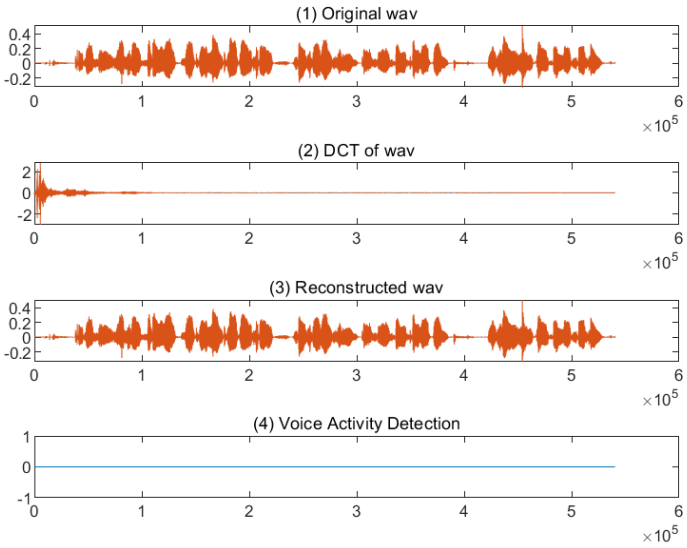

图 9

七、 实验心得体会

在本次研究中，我采用了在课堂上学习的方法，分别实施了快速傅里叶变换（FFT）、小波变换（DWT）、以及离散余弦变换（DCT）三种先进的语音信号处理技术，并拓展到了线性预测编码（LPC）、过零率（Zero Crossing Rate，ZCR）和语音活动检测（VAD）。这一系列实验不仅加深了我对这些理论知识的理解，而且通过使用 Matlab 作为主要的实验工具，我初步掌握了对语音信号进行处理的技术和方法。

通过对最初的语音信号进行基本处理及扩展实验，我不仅验证了所学理论的实际应用性，还进一步探究了不同处理技术对结果的影响。我了解到 LPC 能有效地表示语音信号的光谱特性，ZCR 在有声和无声段落检测中的重要性，以及 VAD 在语音通信中的应用价值。在这一探索过程中，我尝试了多种参数设置和不同的算法，以评估它们对最终语音信号处理质量和效率的影响。

本次实验极大地丰富了我的学术经验，并增进了我对语音信号处理领域的兴趣和理解。通过实际操作和结果分析，我深刻认识到了 FFT、DWT、DCT 以及 LPC、ZCR 和 VAD 在语音信号处理中的重要性和应用潜力。这一过程不仅强化了我对理论知识的掌握，还提高了我在实际应用中解决问题的能力。