南开大学

Nankai University

南　　开　　大　　学

网　络　空　间　安　全　学　院

信息隐藏技术实验二

语音信号的常用处理方法

穆禹宸　2012026

年级：2020 级

专业：信息安全、法学双学位班

指导教师：李朝晖

2023 年 3 月 7 日

目录

一、 实验要求

分别使用下面三个方法，进行语音信号处理:

- FFT
- DWT
- DCT

二、 原始语音信号观察和展示

在开始整个实验之前，需要先将我的语音信号进行一个读取和展示。如下所示，是我所自己录音的一段长达 11s 的语音信号：如图1所示

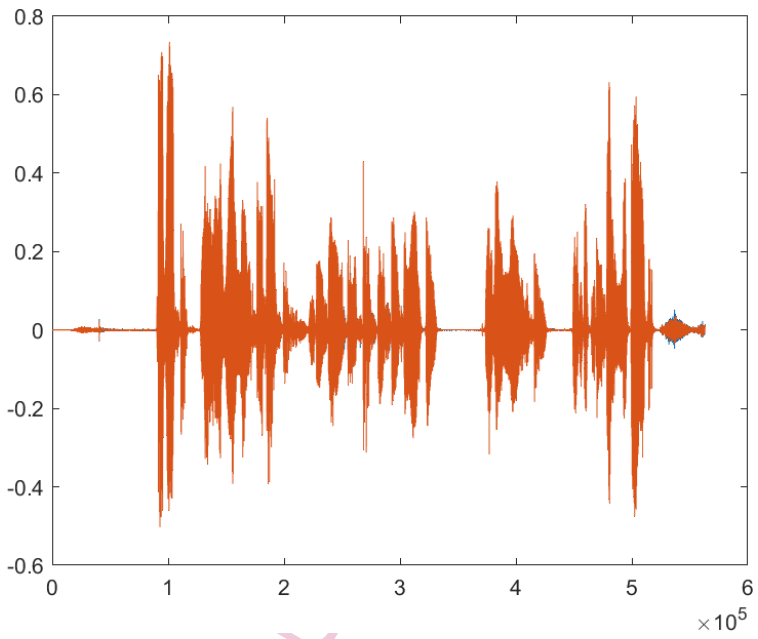

图 1: Caption

我们所使用的代码，如下图所示:

展示代码

```
% Clear Memory and Command window
clc;
clear all;
close all;
[x, fs] = audioread('myvoice.wav');
plot(x);
```

三、 快速傅里叶变换

(一) FFT MATLAB 介绍

matlab 介绍

```
%FFT Discrete Fourier transform.
% 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） 实验代码以及结果展示

首先，我们需要如下代码：

展示代码

```
%%
[x,f]=audioread ('myvoice.wav');
fx=fft(x);
plot(abs(fftshift(fx)));
```

如图2所示

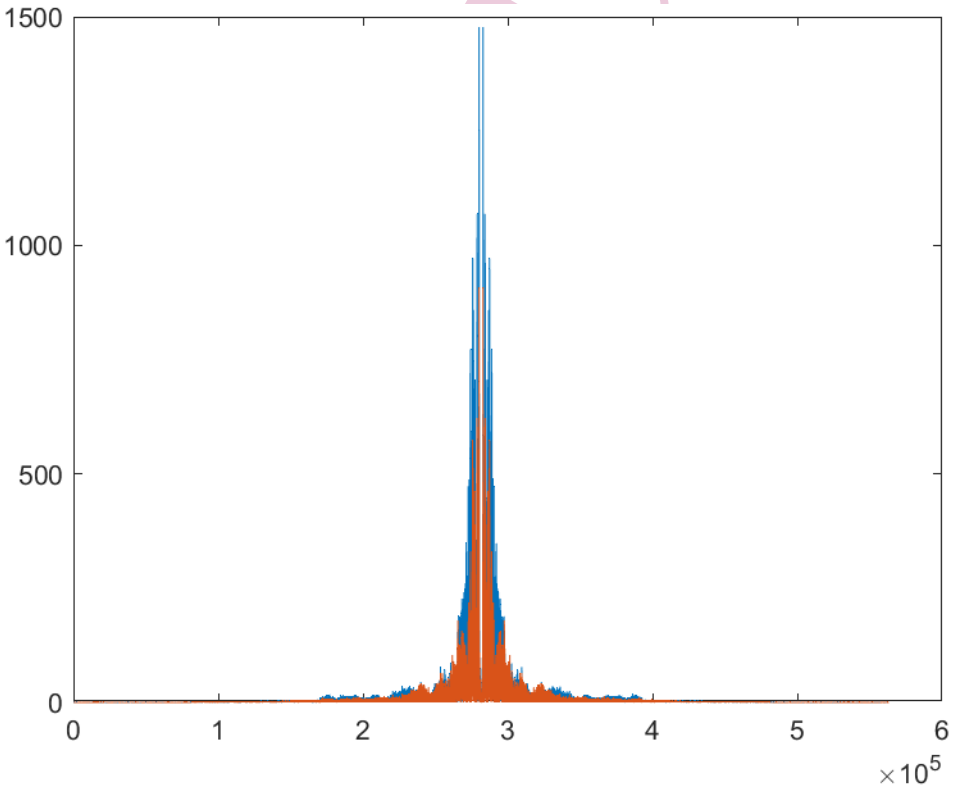

图 2: 快速傅里叶变换

可以看到，进行处理后，语音信号的能量大部分集中在 20kHz 至 40kHz 之间。

四、 离散小波变换

(一) DWT MATLAB 介绍

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) 实验代码以及结果展示

对于离散小波分解，我们分别做了三种处理:

1. 一级小波分解 (dwt)
2. 一级小波分解 (wavedec)
3. 三级小波分解 (wavedec)

1. 一级小波分解 (dwt):

这里小波基采用 Daubechies-4 小波，然后我们分别给出原始语音信号、一级分解的细节分量、一级分解的近似分量，和最后一级分解重构的结果。

代码如下所示:

一级小波分解 (dwt)

```
%%
% Clear Memory and Command window
clc;
clear all;
close all;
[a,fs]=audioread("myvoice.wav");
[ca1,cd1]=dwt(a(:,1),'db4');
a0=idwt(ca1,cd1,'db4',length(a(:,1)));
%绘图
subplot(2,2,1); plot(a(:,1)); %原始波形
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 ( a0 ) ; %一级分解的重构结果
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') ;
```

然后展示结果:

如图3所示

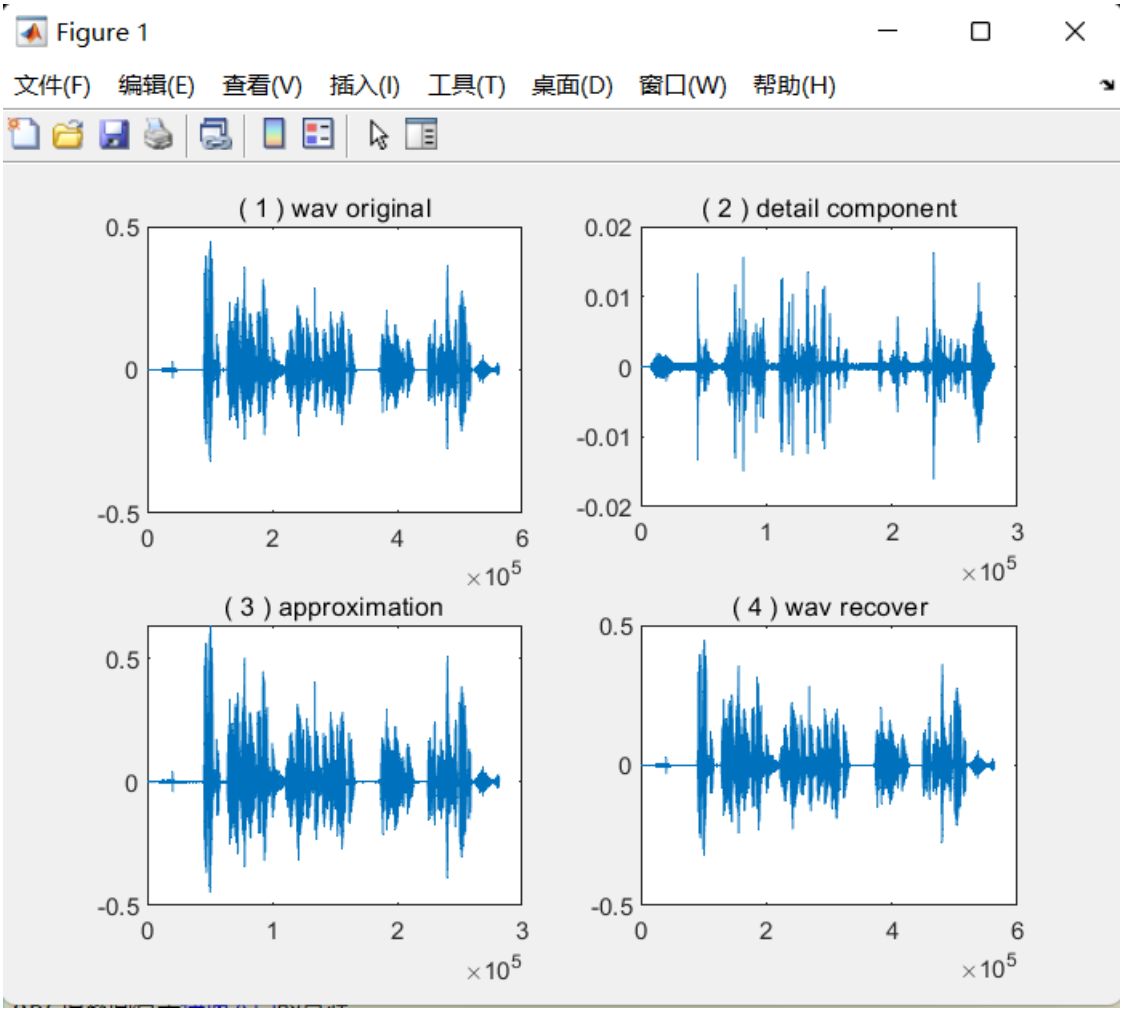

图 3: 一级小波分解 (dwt)

(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。

2. 一级小波分解 (wavedec):

这里小波基也采用 Daubechies-4 小波，然后我们分别给出原始语音信号、一级分解的细节分量、一级分解的近似分量，和最后一级分解重构的结果。

一级小波分解 (wavedec)

```
%%
```

```
% Clear Memory and Command window
clc;
clear all;
close all;
[a,fs]=audioread("myvoice.wav");
[ca1,cd1]=dwt(a(:,1),'db4');
a0=idwt(ca1,cd1,'db4',length(a(:,1)));
%绘图
subplot(2,2,1); plot(a(:,1)); %原始波形
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(a0); %一级分解的重构结果
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');
```

这段代码与之前的 dwt 很像，区别在于我们使用了 wavedec 方式去进行小波分解。

如图4所示

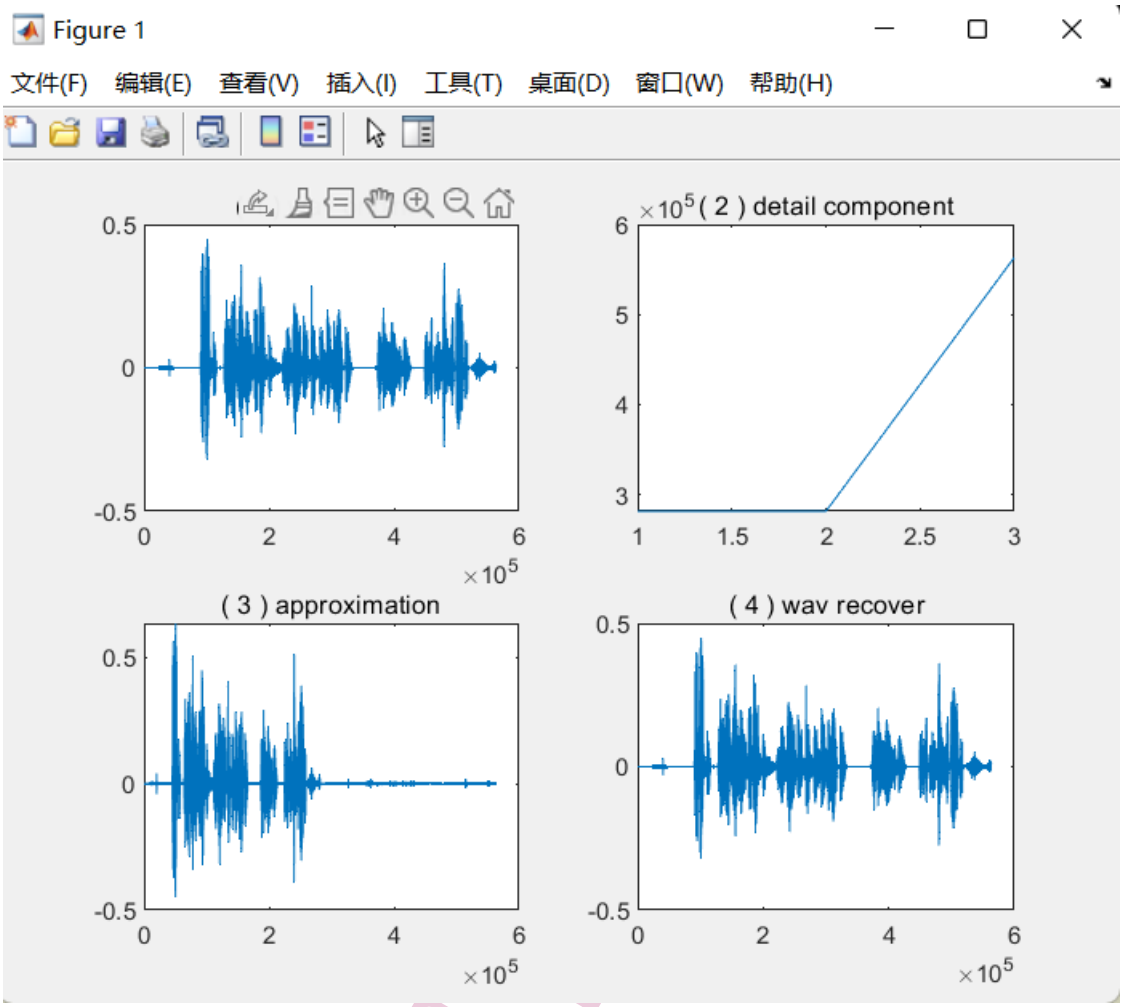

图 4: 三级小波分解 (wavedec)

(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。

3. 三级小波分解 (wavedec)：

三级小波分解比之前更为复杂，小波基仍然采用 Daubechies-4 小波，同时给出原始语音信号、三级分解的细节分量、一级分解的近似分量、二级分解的近似分量、三级分解的近似分量，最后是三级分解重构的结果。

三级小波分解 (wavedec)

```
%%
% Clear Memory and Command window
clc;
clear all;
close all;
[a,fs]=audioread('myvoice.wav') ;
[c,l]=wavedec(a(:,2),3 ,'db4') ;
ca3=appcoef(c,l,'db4',3) ;
cd3=detcoef(c,l,3) ;
```

```
cd2=detcoef(c,l,2) ;
cd1=detcoef(c,l,1) ;
a0=waverec(c,l,'db4') ;

subplot (3 ,2 ,1) ; plot ( a ( : , 2 ) ) ;
subplot (3 ,2 ,2) ; plot ( ca3 ) ; % 三级分解近似分量
subplot (3 ,2 ,3) ; plot ( cd1 ) ; % 一级分解细节分量
subplot (3 ,2 ,4) ; plot ( cd2 ) ; % 二级分解细节分量
subplot (3 ,2 ,5) ; plot ( cd3 ) ; % 三级分解细节分量
subplot (3 ,2 ,6) ; plot ( a0 ) ; % 重构结果
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') ;
```

这段代码比前两段代码更为复杂，因为要展示更多级的分解细节分量。

如图5所示

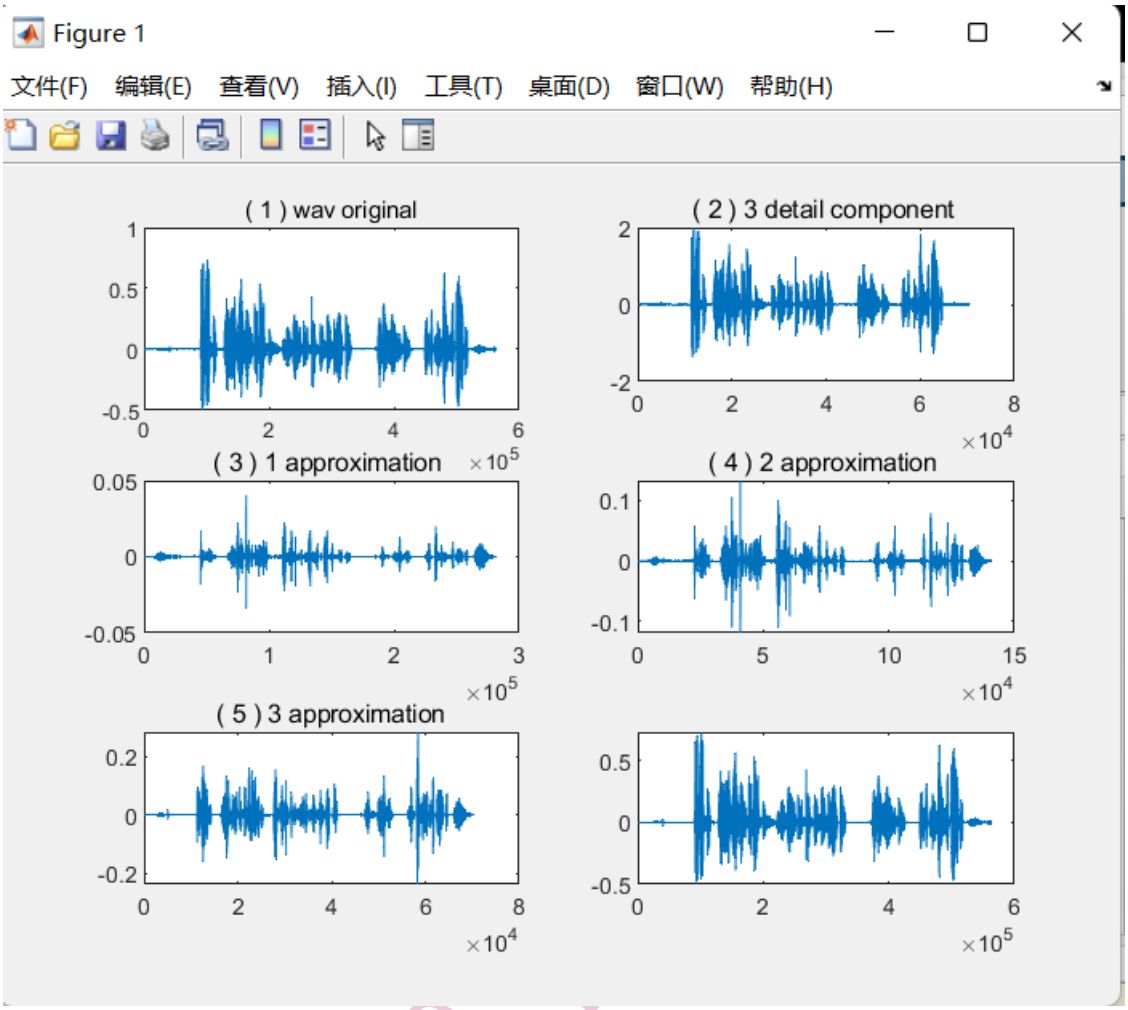

图 5: 三级小波分解 (wavedec)

(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。

五、 离散余弦变换

1. DCT Matlab 介绍

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.
```

DCT 全称为 Discrete Cosine Transform，即离散余弦变换。DCT 变换属于傅里叶变换的一种，常用于对信号和图像（包括图片和视频）进行有损数据压缩。

2. 实验代码以及结果展示

dct 变换

```
%%
% Clear Memory and Command window
clc;
clear all;
close all;
[a,fs]=audioread('myvoice.wav') ;
dct_a=dct(a(:,1)) ;
a0=idct(dct_a);
subplot(3,1,1);plot(a(:,1));%原始波形
subplot(3,1,2);plot(dct_a); %dct 处理后的波形
subplot(3,1,3);plot(a0); %重构得到的结果

```

```
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');
```

如图6所示

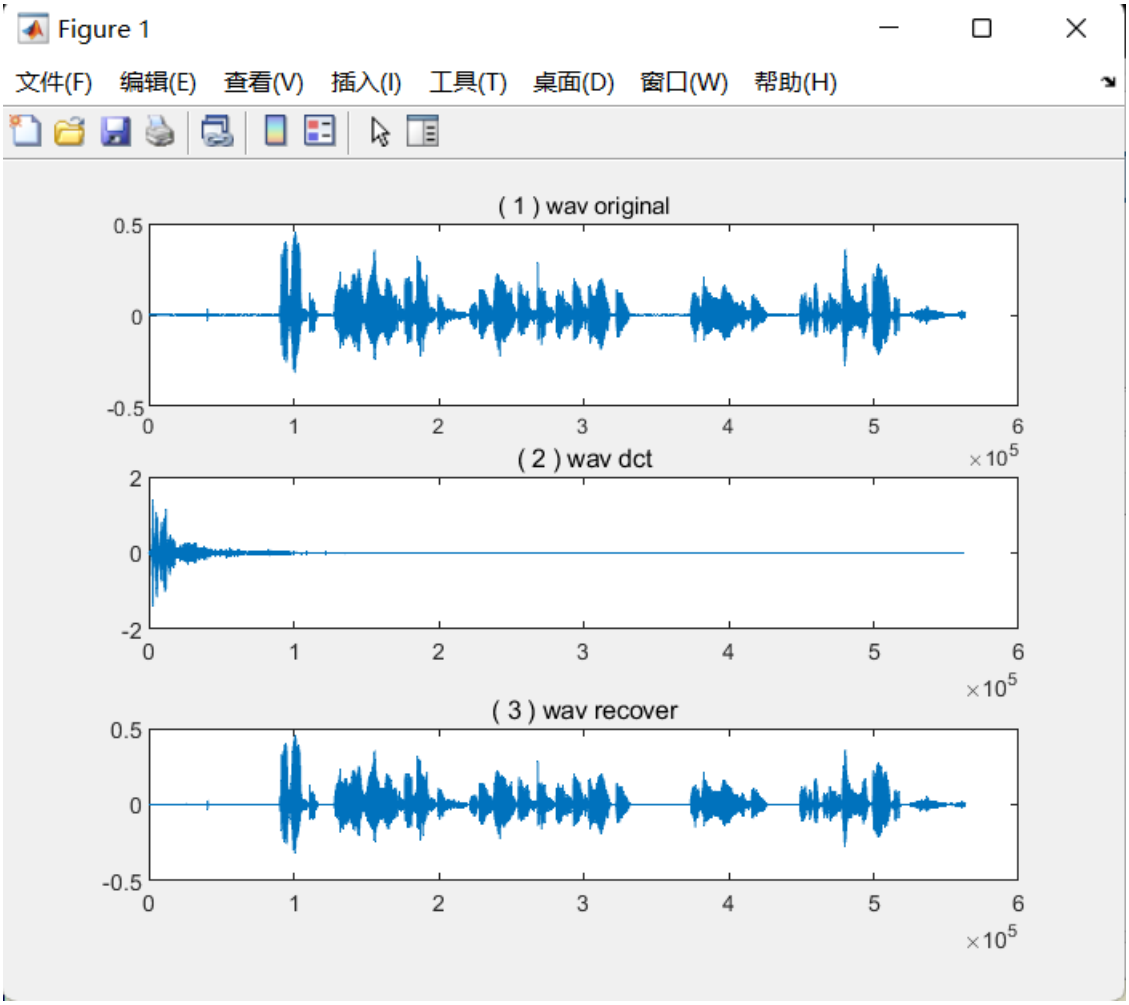

图 6: dct 变换

六、 整体展示

最后，我做了一个整体展示:

整体

```
% Clear Memory and Command window
clc;
clear all;
close all;
[x, fs] = audioread('myvoice.wav');
plot(x);
%%
```

```
fx = fft(x);
subplot(6,1,1);plot(abs(fftshift(fx)));
[a, fs] = audioread('myvoice.wav');
[ca1, cd1] = dwt(a(:,1), 'db4');
a0=idwt(ca1, cd1, 'db4', length(a(:,1)));
subplot(6,1,2);plot(a(:,1));
subplot(6,1,3);plot(cd1);
subplot(6,1,4);plot(ca1);
subplot(6,1,5);plot(a0);
[a, fs] = audioread('myvoice.wav');
res = dct(a);
subplot(6,1,6);plot(res);
```

如图7所示

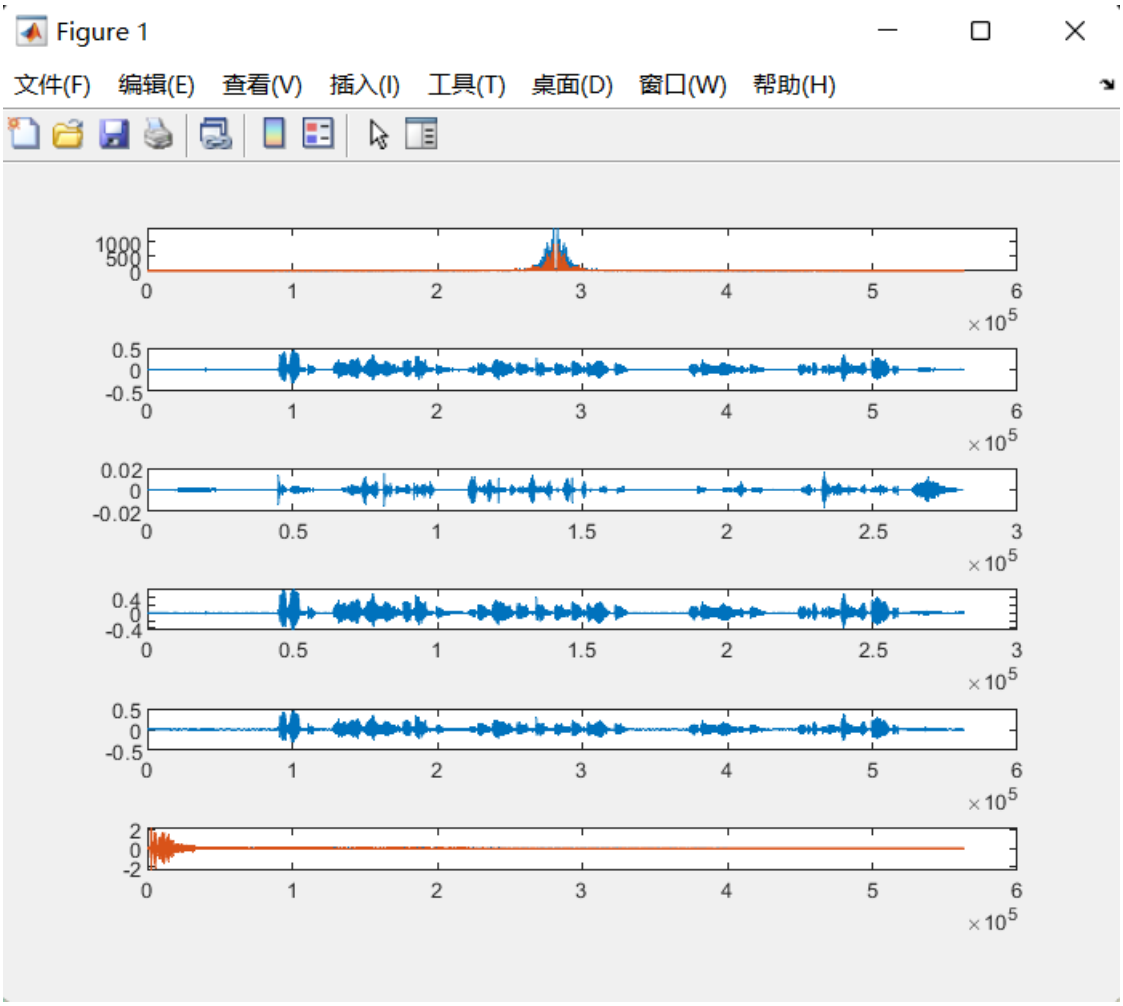

图 7: 整体展示

七、 实验心得体会

本次实验，我通过课上所学的方法，分别实现了 fft 、dwt 、dct 三种语音信号处理方式，并且进一步熟悉了所学知识，并初步掌握了 Matlab 对语音信号处理的方式。另外，通过上机实验，

由于本次实验进行较为顺利，我多次尝试了不同原始语音信号，做了更多的探索，在处理最初语音信号的基础上，也进一步探索了不同处理方式的影响和开销，尝试了不同参数和不同返回值。总而言之，本次实验，我收获很大！