信息隐藏技术第一次实验报告

专业：信息安全 姓名：魏伯繁 学号：2011395

第一部分：使用FFT处理音频

FFT即傅里叶变换，常用的使用傅里叶变换进行音频处理的方式包括快速傅里叶变换和短时傅里叶变换它们有一些相似之处，但也有一些显著的区别。

傅里叶变换是一种将时域信号转换为频域信号的方法，它可以将一个时域信号分解成若干个正弦和余弦函数的叠加。傅里叶变换的输入是一个无限长的连续时间信号，输出是该信号在频域上的频谱，可以得到该信号的频率分量、幅度和相位信息。傅里叶变换通常适用于稳态信号的频谱分析，如正弦波、方波等。

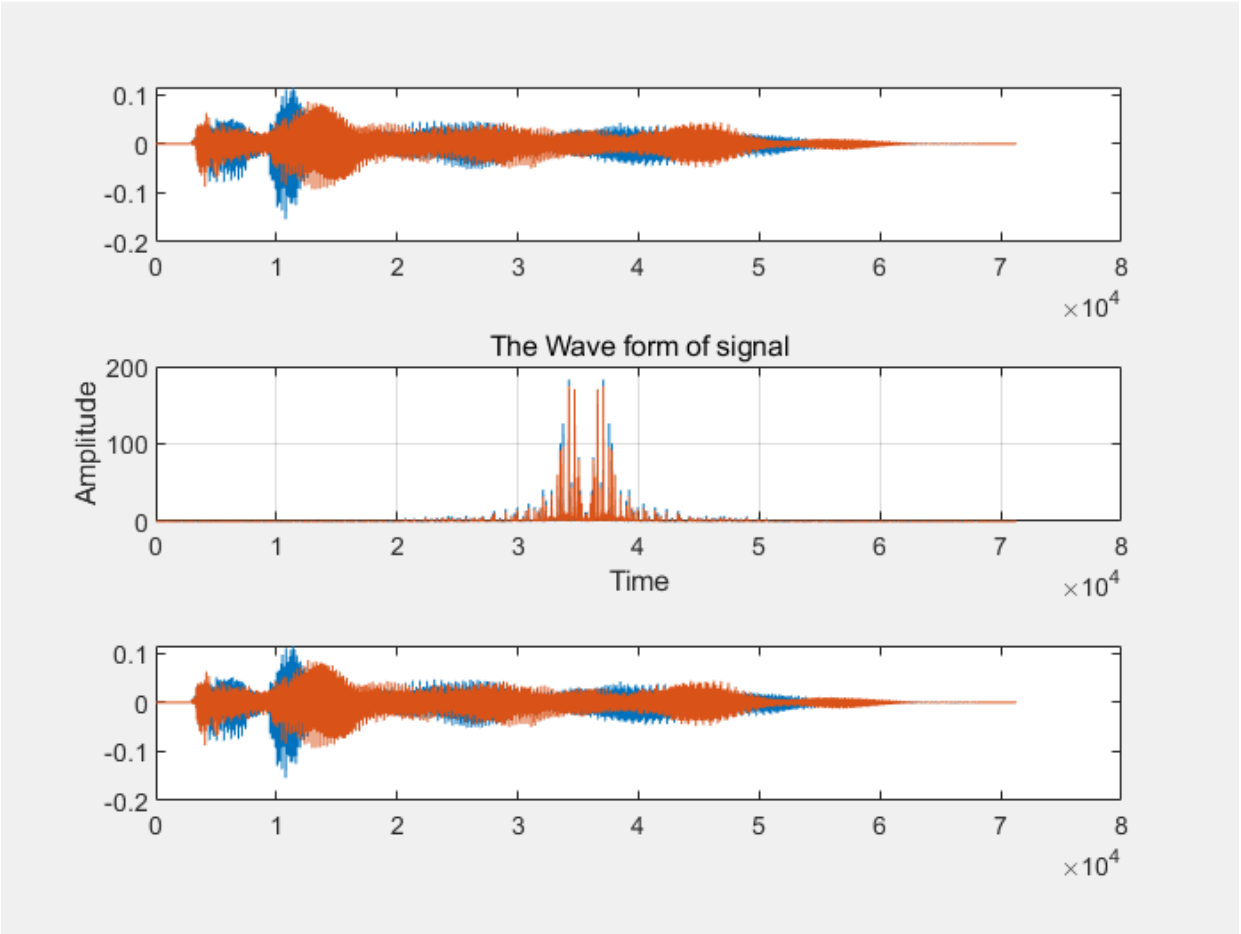
短时傅里叶变换是一种将时域信号分段处理的方法，它可以在每个时间窗口内计算傅里叶变换，然后将结果拼接起来得到整个信号的频谱。短时傅里叶变换的输入是一个分段的时域信号，输出是该信号在时间和频率上的频谱，可以得到信号在不同时间和频率上的变化情况。短时傅里叶变换通常适用于非稳态信号的频谱分析，如音频信号、语音信号等。

因此，傅里叶变换和短时傅里叶变换的主要区别在于：

1. 输入信号的类型：傅里叶变换的输入是一个无限长的连续时间信号，而短时傅里叶变换的输入是一个分段的时域信号。
2. 处理方式：傅里叶变换将整个信号进行频域分析，短时傅里叶变换将信号分段处理，每个时间窗口内进行频域分析。
3. 应用场景：傅里叶变换适用于稳态信号的频谱分析，短时傅里叶变换适用于非稳态信号的频谱分析。

```
[x,fs]=audioread('tada.wav'); %读取名为“tada.wav”的音频文件，将音频信号存储在变量x中，并将采样率存储在变量fs中。

fx=fft(x);%对音频信号进行FFT（快速傅里叶变换），将结果存储在变量fx中。

x0=ifft(fx);%对FFT结果进行逆变换，将结果存储在变量x0中。由于FFT是一种线性变换，因此对FFT结果进行逆变换将返回原始音频信号。

figure('name','快速傅里叶变换');%创建一个名为“快速傅里叶变换”的图形窗口，用于显示音频信号波形和频谱图。

subplot(3,1,1);plot(x);%将图形窗口分成3行1列，选择第1个子图，绘制原始音频信号波形。这里使用plot函数绘制信号的波形。

% fftshift函数的作用是将一维或多维离散傅里叶变换的频域结果进行移位，使得频域零频分量（直流分量）位于频谱中心。
subplot(3,1,2);plot(abs(fftshift(fx)));  %选择第2个子图，绘制FFT结果的幅度谱。使用fftshift函数将零频位置移到频谱的中心。

%对第2个子图添加X轴标签“Time”，Y轴标签“Amplitude”，标题“Wave form of signal”，并打开网格线。
xlabel('Time');ylabel('Amplitude');title('The Wave form of signal');grid on;

%选择第3个子图，绘制逆变换的结果（即原始音频信号）。
```

```
subplot(3,1,3);plot(x0);
```



在本次实验中，我们选取了db4作为基函数，db4是一种小波基函数，是Daubechies小波家族中的一员。在Matlab中，db4小波函数可以通过调用wfilters函数来获取。db4小波函数是一种紧支小波，也称为紧框小波。它的名称db4表示该小波函数是Daubechies小波家族中第四种基函数，也是最常用的一种。

db4小波函数具有许多优良的性质，如正交性、紧支性、对称性等，因此在信号处理和图像处理中广泛应用。它可以用于小波变换、小波包变换、小波分析和合成等方面。在小波变换中，db4小波函数通常用于低通滤波器和高通滤波器的设计，以便对信号进行多尺度分解和重构。在图像处理中，db4小波函数也常用于去噪、压缩、边缘检测等方面。

第二部分 使用DWT处理语音信号

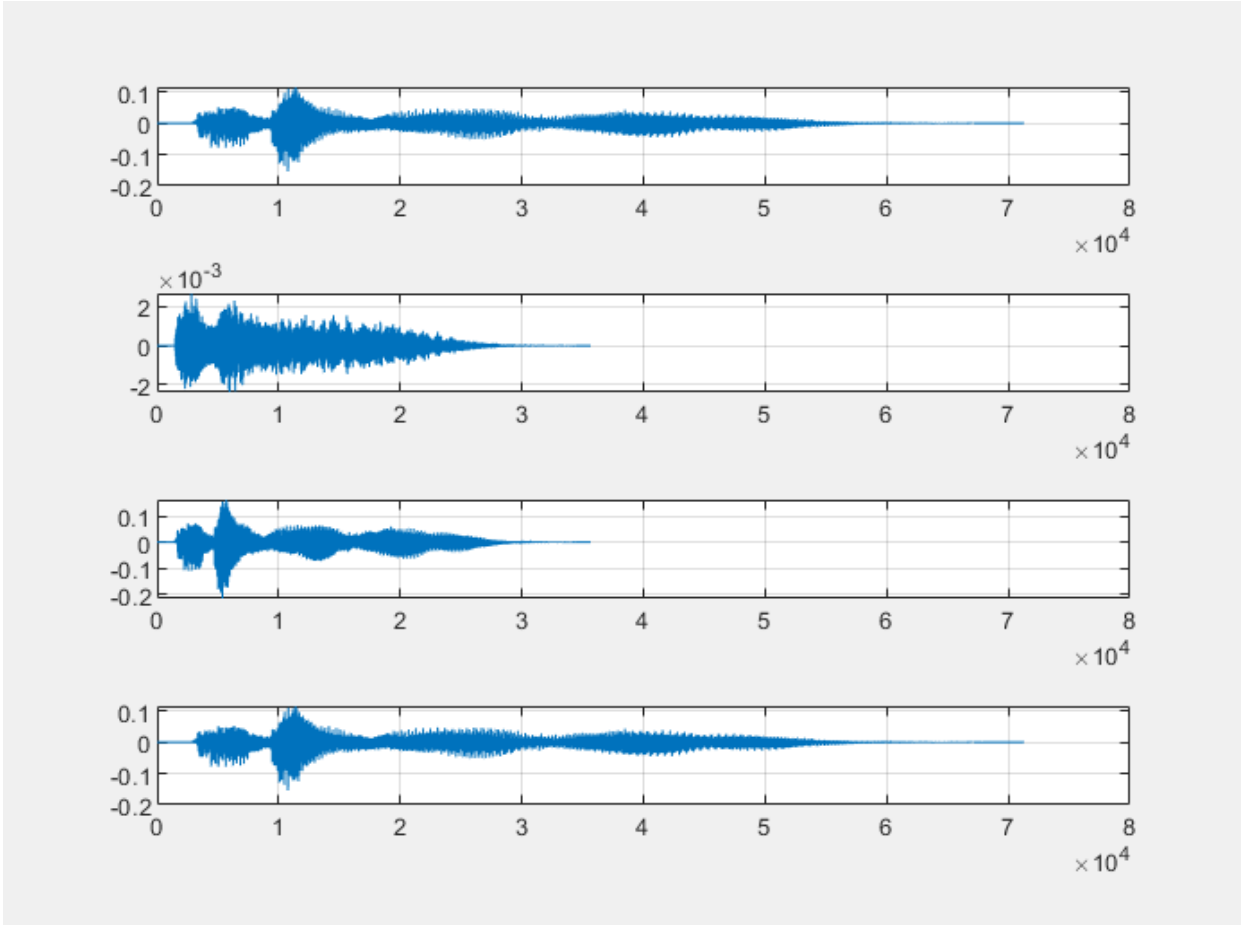
2.1一级小波分解（DWT）

DWT是离散小波变换（Discrete Wavelet Transform）的缩写，是一种用于信号处理和数据压缩的数学方法。它可以将任意信号分解成多个尺度和频率的小波成分，从而实现信号的多分辨率分析和处理。

DWT是一种基于滤波器组的信号分解方法，它利用低通滤波器和高通滤波器将信号分解为不同频率的子带。然后，每个子带可以再次进行分解，直到达到所需的分辨率或精度。这种多尺度分解的特性使得DWT适用于处理非平稳和非平滑的信号，例如语音信号、图像信号等。

在语音信号处理中，DWT通常用于信号降噪、特征提取、压缩和识别等方面。例如，可以利用DWT将语音信号分解为多个子带，然后对不同子带的信号进行降噪和特征提取。此外，可以利用DWT对语音信号进行压缩，从而实现语音数据的存储和传输。在语音识别中，DWT也可以用于提取语音信号的特征，例如基音周期、共振峰等。

```
[a,fs]=audioread('tada.wav'); %将返回的音频数据存储在变量"a"中，采样率存储在变量"fs"中

% 使用了"db4"小波作为小波分析的基函数。
[ca1,cd1]=dwt(a(:,1),'db4');%对"a"中的第一个声道进行了小波一级分解，并将返回的低频成分存储在变量"ca1"中，高频成分存储在变量"cd1"中

a0=idwt(ca1,cd1,'db4',length(a(:,1)));%对"ca1"和"cd1"进行小波一级重构，并将重构后的信号存储在变量"a0"中。

figure('name','小波一级分解与重构');%指定图像的名称

% 将原始信号、低频成分、高频成分和重构后的信号绘制在同一个窗口中，并使用"linkaxes"函数将四个子图的X轴链接在一起。
% grid on是一种命令，用于在当前图形窗口中显示网格线。该命令可以使图像更易于读取和分析，特别是在绘制曲线时，可以更清晰地看出曲线的波动情况。
ax(1)=subplot(4,1,1);plot(a(:,1));grid on; % subplot函数用于在同一图像窗口中显示多个子图
ax(2)=subplot(4,1,2);plot(cd1);grid on;
ax(3)=subplot(4,1,3);plot(ca1);grid on;
ax(4)=subplot(4,1,4);plot(a0);grid on;

% linkaxes函数可以将多个图形坐标轴绑定在一起，以便它们在相同的数据范围内自动放大和缩小。这可以帮助用户更方便地比较多个数据集之间的差异和趋势，同时还能保持它们的比例一致。
linkaxes(ax,'x');
```



2.2一级小波分解（WAVEDEC）

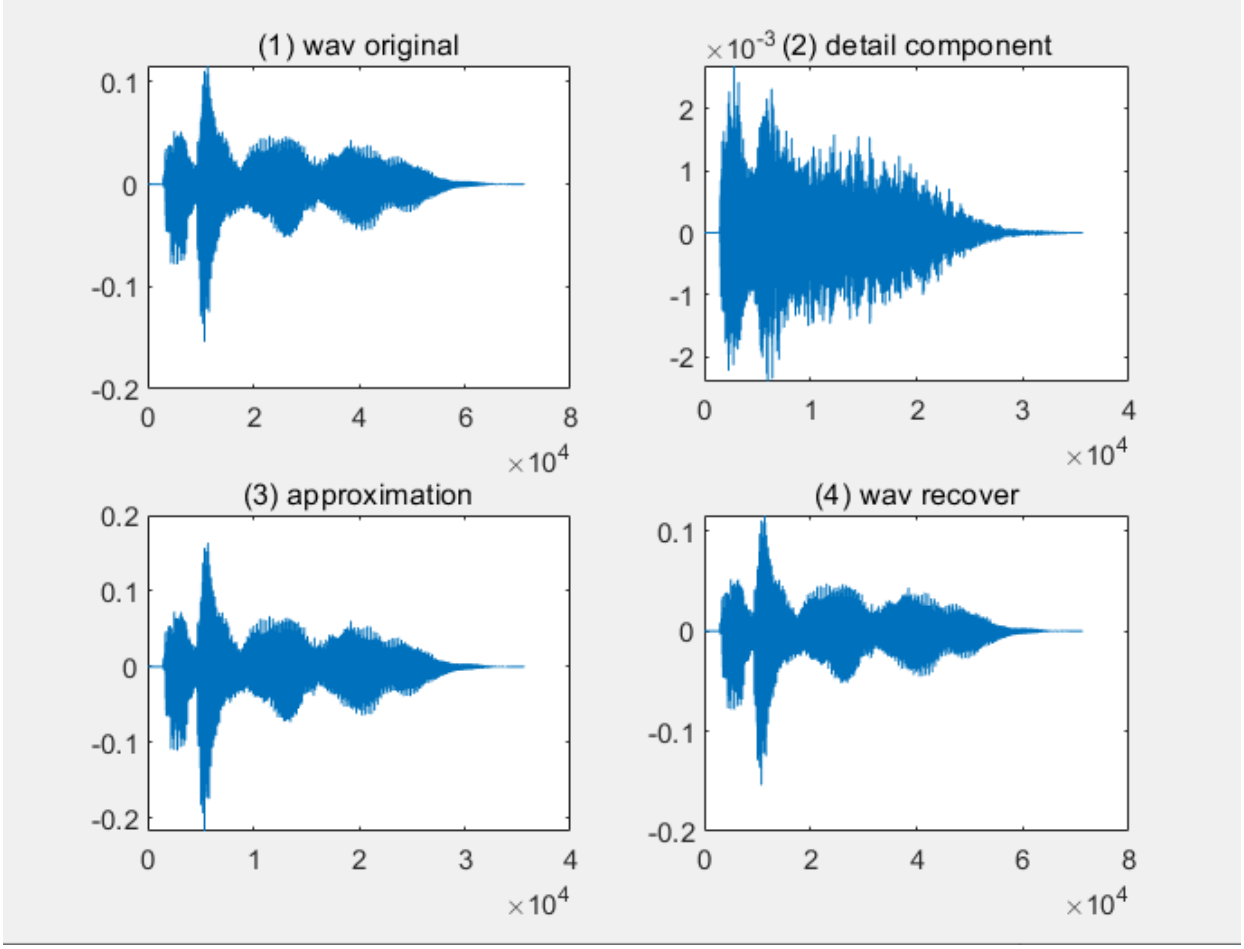
这段Matlab代码的功能是对给定的音频文件“tada.wav”进行一级小波分解，提取其细节分量和近似分量，并对分解后的信号进行一级小波重构，最终生成一个包括原始音频、细节分量、近似分量以及重构后音频的4个子图的画布。

其中，函数“audioread()”用于读取音频文件，返回读取的音频信号“a”以及采样率“fs”。函数“dwt()”使用db4小波对音频信号进行一级小波分解，返回分解后的近似分量“ca1”和细节分量“cd1”。函数“idwt()”使用db4小波对分解后的近似分量和细节分量进行一级小波重构，返回重构后的音频信号“a0”。最后，通过“subplot()”和“plot()”函数绘制出原始音频、细节分量、近似分量以及重构后音频的4个子图，并使用 “title()”函数为每个子图添加标题。

```
[a,fs]=audioread('tada.wav');
[ca1,cd1]=dwt(a(:,1),'db4');%使用 db4 小波对 a 的第一列进行一级离散小波变换（DWT），返回分解后的近似系数（ca1）和细节系数（cd1）
a0=idwt(ca1,cd1,'db4',length(a(:,1)));%使用 db4 小波对近似系数和细节系数进行一级离散小波逆变换（IDWT），返回重构后的信号（a0）

figure
% 将当前的图形窗口分成2x2个网格，并分别绘制原始信号、细节分量、近似分量和重构后的信号的波形图。
subplot(2,2,1);          plot(a(:,1));
subplot(2,2,2);          plot(cd1);
subplot(2,2,3);          plot(ca1);
subplot(2,2,4);          plot(a0);

% 当前绘图的坐标轴切换到相应的子图中，并设置相应的标题。
```

```
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')
```

上面的代码展示了如何使用 db4 小波对给定的声音信号进行一级分解和重构，并绘制相应的波形图。



2.3三级小波分解（WAVEDEC）

小波分解是一种常见的信号处理方法，也常用于语音信号分析。小波分解将语音信号分解为多个尺度的子带，每个子带中包含了不同频率的信息。小波分解过程可以分为以下步骤：

1. 选择小波基函数

小波基函数是一组正交基函数，不同的小波基函数可以用于分解不同类型的信号。在语音信号处理中，通常使用Daubechies小波基函数，如db4。

2. 将语音信号进行小波变换

将原始语音信号进行小波变换，得到多个分解系数。在MATLAB中，可以使用wavedec函数进行小波分解。

3. 分解系数的分组

将分解系数分成多个组，每个组代表一个尺度的子带。分组的方法通常是将最后一层低频分解系数作为近似系数，其他高频系数作为细节系数。

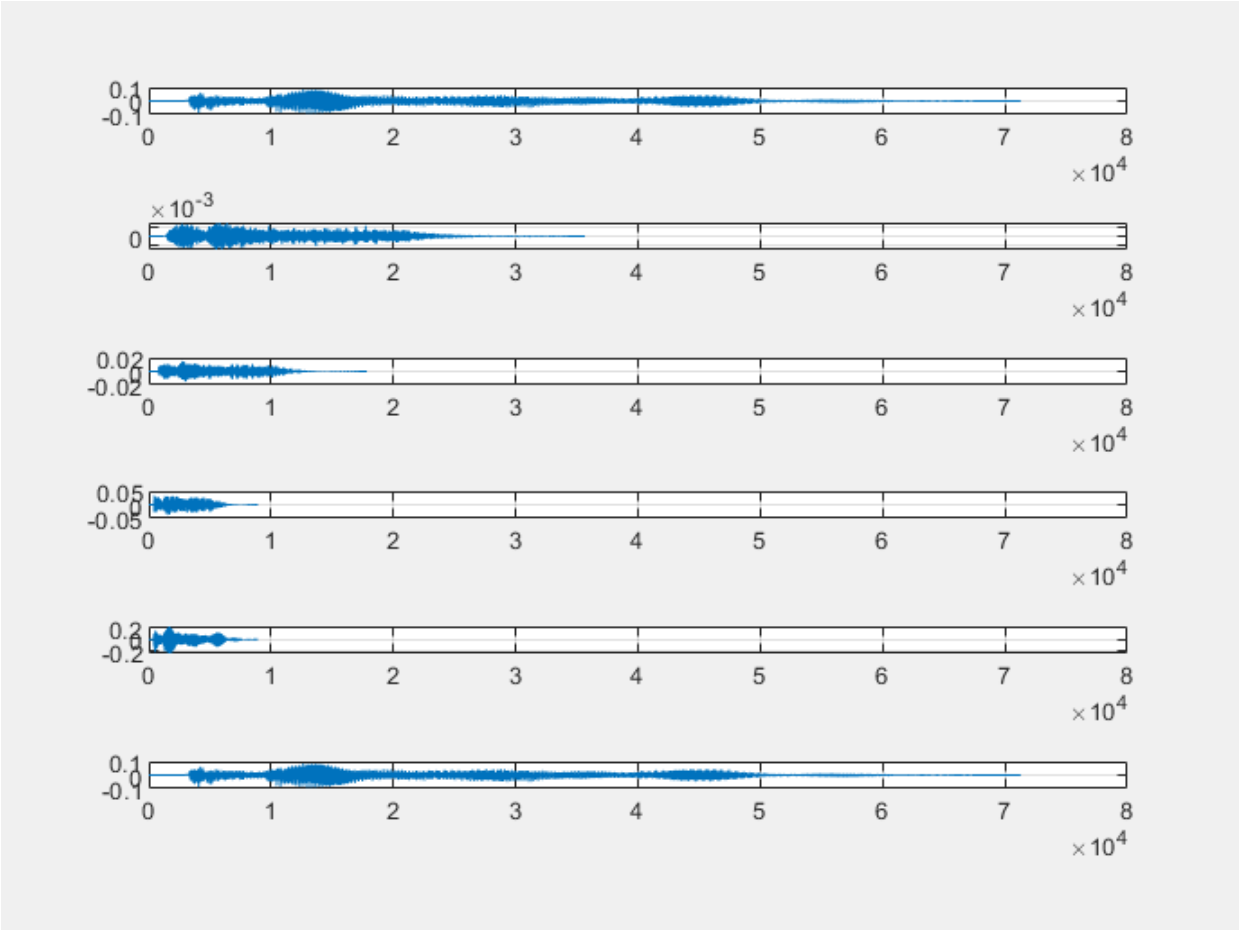
4. 重构语音信号

将分组后的分解系数用小波基函数进行重构，得到多个子带信号。在MATLAB中，可以使用waverec函数进行小波重构。

5. 分析子带信号

分析每个子带信号，可以提取语音信号的不同频率成分，得到一些有用的特征信息。例如，可以用高频子带来检测语音信号中的共振峰位置，或用低频子带来检测语音信号的基频。

```
[a,fs]=audioread('tada.wav'); %读取音频文件 'tada.wav'，将音频数据存储在 a 中，采样率存储在 fs 中。
[c,l]=wavedec(a(:,2),3,'db4');%对a的第二列进行小波分解，使用 db4 小波基函数，分解3级，返回低频和高频系数。低频系数保存在c的第一部分，高频系数保存在c的其他部分。最后为含有各个尺度下的长度。
ca3=appcoef(c,l,'db4',3);%从c和l中提取出第三级近似系数，保存在 ca3 中
cd3=detcoef(c,l,3);%从c和l中提取出第三级细节系数，保存在 cd3 中
cd2=detcoef(c,l,2);%从c和l中提取出第二级细节系数，保存在 cd2 中
cd1=detcoef(c,l,1);%从c和l中提取出第一级细节系数，保存在 cd1 中
a0=waverec(c,l,'db4');% 使用 db4 小波基函数，将 c 和 l 中的系数进行小波重构，得到原始信号的近似结果 a0。
figure('name','小波三级分解与重构');
% 和前面的subplot一样，绘制结果图像并展示
ax(1)=subplot(6,1,1);plot(a(:,2));grid on;
ax(2)=subplot(6,1,2);plot(cd1);grid on;
ax(3)=subplot(6,1,3);plot(cd2);grid on;
ax(4)=subplot(6,1,4);plot(cd3);grid on;
ax(5)=subplot(6,1,5);plot(ca3);grid on;
ax(6)=subplot(6,1,6);plot(a0);grid on;
linkaxes(ax,'x');
```



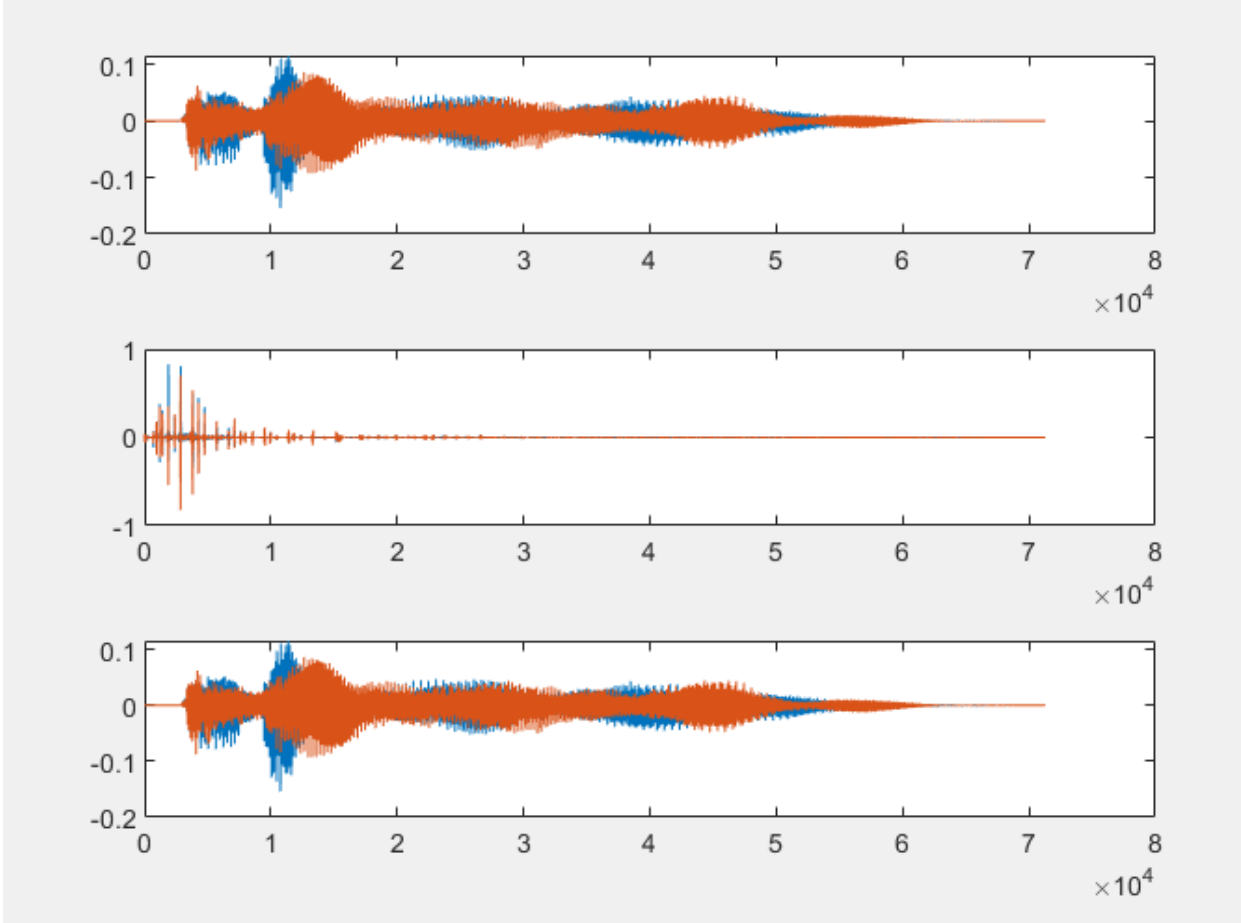
第三部分 使用dct处理语音信号

DCT（Discrete Cosine Transform）即离散余弦变换，是一种把时域信号转换到频域的技术。DCT通过计算一个信号的离散余弦系数，可以将信号转换成一组具有不同频率和幅度的余弦函数。它的主要作用是将一个时间信号转换为一个相关的频率域信号，方便分析和处理。

在语音信号处理方面，DCT广泛应用于压缩编码中，如MP3等音频编码格式中采用了DCT技术进行信号的压缩编码，以便于音频文件的传输和存储。此外，DCT还可以用于信号去噪、语音识别、语音合成等领域，其中最常用的是信号压缩编码和去噪方面。DCT在去噪方面可以通过在频域进行滤波来去除噪声，同时保留原始信号的重要信息。

DCT常用于图像和音频信号的压缩和编码，以及信号的特征提取和分类。在语音信号处理中，DCT可以用于对声学参数进行编码，例如对语音信号的短时功率谱或倒谱系数进行DCT变换，可以得到一组具有更好区分性能的系数，这些系数可以用于语音信号的特征提取和识别。

```
[a,fs]=audioread('tada.wav');
da=dct(a);% 对输入信号a进行离散余弦变换（DCT），并将结果赋值给变量da。DCT是傅里叶变换的一种变种，它是一种线性变换，可以将一段时间域上的连续信号转换为在频域上不同频率的正弦函数和余弦函数。
a0=idct(da);% 对da进行离散余弦逆变换（IDCT），将结果赋值给变量a0。IDCT是DCT的逆变换，可以将离散余弦变换得到的频域信号重构回时间域。
figure('name','离散余弦变换');
% 创建一个新的窗口，并设置其名称为“离散余弦变换”。使用subplot函数将窗口分为3行1列，依次在每个子图中绘制输入信号a、DCT变换后的信号da和IDCT变换后的信号a0。
subplot(3,1,1);plot(a);
subplot(3,1,2);plot(da);
subplot(3,1,3);plot(a0);
```



第四部分：比较三种常用的语音信号处理方式

FFT、DWT和DCT都是常见的语音信号处理技术，它们在处理语音信号中具有不同的优点和应用场景。

快速傅里叶变换主要用于语音信号的频域分析，可以通过FFT将语音信号从时域转换为频域，得到语音信号的频谱信息，包括频率成分和振幅。FFT广泛应用于语音信号的音调分析、滤波和降噪等方面。

离散小波变换可以将语音信号分解成多个不同频率范围的子信号，每个子信号包含了不同的时间和频率信息。因此，DWT在处理语音信号时可以实现多分辨率分析，提取语音信号的局部时频特征。DWT常用于语音信号的降噪、压缩和特征提取等方面。

离散余弦变换主要用于语音信号的数据压缩和信号编码，可以将语音信号转换为一组系数，这些系数可以表示语音信号的能量分布情况，用于语音信号的压缩和编码。同时，DCT还可以实现频域滤波，去除语音信号中的高频噪声。DCT广泛应用于语音信号的编码、压缩和加密等方面。

不难看出，FFT、DWT和DCT在语音信号处理中有着各自独特的应用场景和优点，需要根据具体任务需求进行选择和使用。

第五部分 关键函数解析

1、FFS函数

在MATLAB中，FFT（Fast Fourier Transform）函数用于将信号从时间域转换为频率域。这个函数的输入是一个向量，表示一个离散时间的信号，并输出一个具有相同数量的点的向量，表示这个信号的离散频谱。该函数实现了快速离散傅里叶变换算法，它是一种快速计算傅里叶变换的算法。

其中X是输入的信号向量，Y是表示X的离散频谱的向量。如果X是一个矩阵，则FFT函数将逐列计算每个列的FFT。

```
Y = fft(X)
```

在FFT函数中，可以通过使用n作为第二个输入来指定点数。例如，如果X是一个长度为1024的向量，而您想要一个包含2048点的FFT，则可以通过以下方式实现：

```
Y = fft(X, 2048)
```

FFT函数输出的离散频谱是复数，其中实部表示信号的幅度，虚部表示信号的相位。如果您只想查看幅度，可以使用MATLAB中的abs函数取FFT的绝对值。

FFT函数是数字信号处理中非常有用的函数，可以在音频信号处理、图像处理、信号过滤、信号压缩等方面广泛应用。

2、IFFT函数

MATLAB中的ifft函数是快速傅里叶变换（FFT）的逆变换，用于将频域信号转换回时域信号。

```
x = ifft(X, n, dim)
```

其中，X是傅里叶变换后的频域信号，n是转换后的时域信号长度，dim是在哪个维度上执行逆变换。如果省略n，则n等于X中被变换的维度的长度。如果省略dim，则默认dim等于第一个非单一维度。

ifft的返回值是逆变换后的时域信号。具体来说，ifft函数采用逆蝶形算法，将输入信号X在频域进行倒序排列，然后对倒序排列后的信号执行FFT算法。最后，对输出的结果进行归一化操作，以得到与原始时域信号幅度相同的逆变换结果。

通常，在使用ifft函数时，需要先使用fft函数将信号转换为频域信号，然后再使用ifft函数将频域信号转换回时域信号。总之，ifft函数是MATLAB中用于将频域信号转换为时域信号的重要工具，广泛应用于信号处理、通信系统、音频处理等领域

3、DWT函数

```
[c,l] = dwt(x,Lo_D,Hi_D)
```

输入参数：

- x：待变换的向量或矩阵
- Lo_D：低通小波滤波器的分解系数
- Hi_D：高通小波滤波器的分解系数

输出参数：

- c：小波系数向量或矩阵
- l：向量或矩阵的分解结构

dwt函数的作用是将输入的向量或矩阵进行小波变换，返回小波系数和分解结构。Lo_D和Hi_D是小波函数的系数，通常通过wfilters函数生成。

4、IWT函数

```
x = idwt(c, l, Lo_R, Hi_R)
```

输入参数解释：

- c：小波系数向量或矩阵
- l：向量或矩阵的分解结构
- Lo_R：低通小波滤波器的重构系数
- Hi_R：高通小波滤波器的重构系数

输出参数解释：

- x：重构的向量或矩阵

idwt函数的作用是将小波系数和分解结构进行逆变换，返回重构后的向量或矩阵。Lo_R和Hi_R是小波函数的重构系数，通常通过wfilters函数生成。

需要注意的是，使用dwt和idwt函数进行小波变换和逆变换时，需要保证小波系数和分解结构的维度和类型一致。通常情况下，小波系数和分解结构是通过dwt函数返回的，然后再将它们传递给idwt函数进行逆变换。

在语音信号处理方面，小波变换被广泛应用于信号去噪、信号压缩、信号分析等方面。dwt和idwt函数是matlab中用于实现小波变换和逆变换的重要工具。

5、wavedec函数

```
[c,l] = wavedec(x,n,wname)
```

其中，x是待分解的一维信号，n是分解的级别，wname是选定的小波基函数。函数的输出c是一个向量，包含了所有的小波系数，l是一个包含了每个尺度下的小波系数数量的向量。

wavedec函数的工作过程如下：

1. 将原始信号x进行第一级小波分解，得到第一层近似系数和第一层细节系数。
2. 将第一层近似系数作为新的信号，重复第一步，直到达到指定的分解级别n。
3. 将每层的近似系数和细节系数按照从高到低的顺序拼接起来，得到一个包含所有小波系数的向量c。
4. 将每层的近似系数长度和细节系数长度存储在向量l中。

wavedec函数常用于小波压缩和小波滤波等领域，它能够将信号的高频和低频部分分离开来，有助于对信号进行分析和处理。

6、APPCOEF函数

```
C = appcoef(C,L,wname)
[C,L] = appcoef(___)
```

appcoef函数是小波分析函数集合之一。该函数用于计算小波变换的近似系数。

其中，C是一个包含小波分解系数的向量，L是长度为N的向量，表示各个小波层次的长度。wname是小波函数的名称。appcoef函数的作用是提取小波分解的近似系数，也就是低频分量。在小波分解中，低频分量包含信号的大部分能量信息，因此往往被用于信号的重构和分析。

例如，在语音信号处理中，可以使用appcoef函数提取低频分量进行音频信号的降噪和压缩。

7、DETCOEF函数

`detcoef`函数用于提取小波变换的细节系数。小波变换通过将信号分解为低频和高频成分来表示信号，其中低频成分是信号的近似部分，而高频成分是信号的细节部分。`detcoef`函数用于提取给定小波分解级别的细节系数，以便进一步处理或分析。

```
[c, l] = wavedec(x, n, wname);
d = detcoef(c, l, n);
```

其中，`x`是要进行小波变换的信号，`n`是小波分解的级别，`wname`是所选择的小波类型。`wavedec`函数用于进行小波分解，它返回小波系数向量`c`和长度向量`l`，表示每个分解级别中的系数数量。`detcoef`函数使用`c`和`l`向量以及所需级别`n`来计算细节系数向量`d`。

8、DCT与IDCT函数

```
y = dct(x)
x = idct(y)
```

其中，`x`是输入信号，`y`是DCT或IDCT变换后的输出。

DCT是一种用于数据压缩、信号处理等领域的数学变换技术。在MATLAB中，`dct`函数将输入信号`x`进行离散余弦变换，并返回DCT系数向量`y`。DCT变换的输出是一个向量，包含输入信号的频域信息，通常用于图像和音频压缩，以及信号处理等应用中。

IDCT函数是DCT变换的逆运算，将DCT系数向量`y`作为输入，并返回重构的原始信号`x`。因此，IDCT用于从DCT系数恢复原始信号，通常用于压缩后的数据解压缩、音频和图像处理等领域。

需要注意的是，DCT和IDCT函数在处理实际信号时通常会与其他信号处理函数一起使用，例如将DCT系数与小波系数结合使用进行信号压缩。