南開大學

# 信息隐藏技术课程实验报告

# 语音信号的常用处理方法


学院：网络空间安全学院
专业：信息安全
学号：2113997
姓名：齐明杰
班级：信安2班

# 目录

# 1　实验目的

实验2：语音信号的常用处理方法上机实验。

内容：
1、学习慕课：2.2语音信号处理基础
2、FFT
3、DWT
4、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。
提交方法：qq群作业。
时间：2024-3-26

# 2　实验原理

## 2.1　FFT（快速傅里叶变换）

FFT是一种算法，用于计算离散傅里叶变换（DFT）及其逆变换。DFT将一个信号从时域转换到频域，揭示了该信号的频率成分。FFT算法大大减少了计算量，使得数字信号处理在实时系统中成为可能。

### 2.1.1　离散傅里叶变换（DFT）

为了理解FFT，首先需要掌握DFT的基础概念。DFT将一个离散时间信号转换为一组离散频率成分。这个转换揭示了原始信号的频谱，即信号的频率成分及其幅度。

DFT定义为：

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N}kn}$$

其中，$X[k]$是变换后的频域信号，$x[n]$是原始时域信号，$N$是信号的长度，$k$是频率索引，$e^{-\frac{i2\pi}{N}kn}$是复指数形式的旋转因子，表示旋转和周期性。

### 2.1.2　FFT的原理和优化

FFT利用了DFT计算中的对称性和周期性，通过将DFT分解为更小的DFTs来减少计算量。最常用的FFT算法是Cooley-Tukey算法，它通过递归地将DFT分解为更小的DFT块来实现效率的提升。

- **分治策略**：如果信号长度(N)可以分解为两个较小的因数，例如$N = N_1 \times N_2$，那么DFT可以分解为$N_1$个长度为$N_2$的DFT和$N_2$个长度为$N_1$的DFT的组合。这种分解可以递归进行，

直到分解到可以直接计算的程度。
- **蝴蝶运算**：在FFT算法中，"蝴蝶"运算是一种特殊的运算步骤，用于合并分解后的小DFTs。这种运算得名于数据流图中的形状，类似于蝴蝶的翅膀。蝴蝶运算有效地实现了DFT的合并步骤，进一步提高了计算效率。

### 2.1.3 FFT的应用

FFT在数字信号处理领域有广泛的应用，特别是在语音信号处理中：

- **频谱分析**：通过FFT可以分析语音信号的频谱，识别出不同的音素和频率成分。
- **滤波**：在频域对信号进行滤波，可以去除噪声或强调某些频率成分。
- **信号压缩**：通过频域分析，可以识别出信号中的重要成分，并对其进行有效压缩。
- **特征提取**：在语音识别和音乐信息检索中，FFT可以用于提取信号的特征，如基频、谐波等。

## 2.2 DWT（离散小波变换）

离散小波变换（DWT）是一种用于时间-频率分析和信号处理的数学工具。与传统的傅里叶变换相比，小波变换提供了对信号的多尺度（或多分辨率）分析能力，使其在处理具有突变或非平稳特征的信号时特别有效。这一特性使DWT成为图像处理、信号压缩、去噪等领域的重要工具。

### 2.2.1 DWT的基本原理

DWT通过将信号与一组小波函数进行卷积，来获得信号的时间-频率表示。这些小波函数是通过对母小波（一个固定的函数）进行缩放（改变频率）和平移（改变时间位置）得到的。与傅里叶变换使用正弦和余弦函数作为基函数不同，小波变换采用的是具有有限长度的小波基函数，这使得小波变换能够同时提供时间和频率信息。

### 2.2.2 多尺度分析

DWT的核心在于它的多尺度分析能力。通过逐步缩放母小波，DWT可以在不同的频率尺度上分析信号。在每个尺度上，信号都被分解为一系列小波系数，这些系数捕获了信号在对应尺度的特征。这种分解通常以金字塔形式进行，高层次的分解捕获信号的粗略特征（低频信息），而底层次的分解捕获细节特征（高频信息）。

### 2.2.3 小波和尺度函数

DWT使用两类函数：小波函数和尺度函数。尺度函数与小波函数类似，但用于捕获信号的平滑部分（即低频分量）。每个级别的DWT分解都会产生两个系列的系数：一系列近似系数（通过尺度函数计算）表示信号的低频部分，一系列细节系数（通过小波函数计算）表示信号的高频细节。

1. **近似系数（$A_k$）**：$A_k = \sum_n x[n] \cdot \phi_{k,n}$ 其中，$x[n]$是原始信号，$\phi_{k,n}$是尺度函数（用于捕获信号的低频信息），k和n分别代表分解的层次和时间索引。
2. **细节系数（$D_k$）**：$D_k = \sum_n x[n] \cdot \psi_{k,n}$ 这里，$\psi_{k,n}$是小波函数（用于捕获信号的高频细节），其他符号与近似系数的表达式相同。

### 2.2.4 DWT的应用

1. **信号去噪**：DWT可以识别和分离信号中的噪声成分，通过修改或去除某些小波系数来减少噪声。
2. **图像压缩**：JPEG 2000等图像压缩标准使用DWT来去除图像中的冗余信息，有效减小文件大小。
3. **特征提取**：在模式识别和图像分析中，DWT被用于提取有用的特征，这些特征对于分类、识别等任务至关重要。
4. **信号分析**：DWT提供的多尺度视角使其成为分析非平稳信号（如金融时间序列、生物信号等）的强大工具。

## 2.3 DCT（离散余弦变换）

离散余弦变换（DCT）是一种将信号或图像从时域（或空间域）转换到频域的技术，它是一种线性、可逆变换，广泛应用于信号和图像压缩领域。DCT特别适用于处理具有高度相关性数据的场合，因为它能够将这些数据转换为频率分量，其中大部分能量集中在变换结果的一小部分系数中。这使得DCT在图像和视频压缩（如JPEG和MPEG标准）中非常有效。

### 2.3.1 DCT的基本概念

DCT的基本思想是将信号分解成一系列余弦波的和。这些余弦波的频率逐渐增加，允许信号的不同部分被不同频率的余弦波表示。在图像压缩中，这意味着图像的平滑区域（低频成分）和细节区域（高频成分）可以分别处理。

### 2.3.2 DCT的数学表达

二维DCT（应用于图像处理）的数学表达式为：

$$F(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cos\left[\frac{\pi(2m+1)u}{2M}\right] \cos\left[\frac{\pi(2n+1)v}{2N}\right]$$

其中，$f(m,n)$是输入图像的像素值，$F(u,v)$是DCT变换后的系数，M和N分别是图像的行数和列数，$C(u)$和$C(v)$是归一化系数，u和v表示频率变量。

### 2.3.3 DCT的类型

存在多种类型的DCT，但最常用的是DCT-II，它通常简称为“DCT”。DCT的其他变体（如DCT-I、DCT-III、DCT-IV）在特定应用中有其用途，但在图像和视频压缩中，DCT-II是最重要的。

### 2.3.4 DCT的应用

1. **图像压缩**：DCT是JPEG图像压缩标准的核心技术。它通过将图像转换为频率域，并仅保留重要的频率成分（通常是低频成分），来实现压缩。
2. **视频压缩**：在MPEG等视频压缩标准中，DCT用于压缩每一帧图像。通过去除时间上的冗余和空间上的冗余，DCT有效减少了视频数据的大小。
3. **音频信号处理**：虽然DWT在音频处理中更常用，但DCT也可以用于音频信号的压缩和编码，特别是在需要去除音频文件中的冗余数据时。

## 2.4 DCT的优势

DCT的主要优势在于其能够有效地集中信号能量，使得大部分信息集中在少数几个系数中。这种“能量紧凑”的特性使得DCT非常适合于信号和图像的压缩，因为在压缩过程中可以优先保留这些包含大部分信息的系数，而舍弃其他较不重要的系数，从而达到降低数据量、减少存储空间需求的目的，同时尽量保持原始数据的质量。

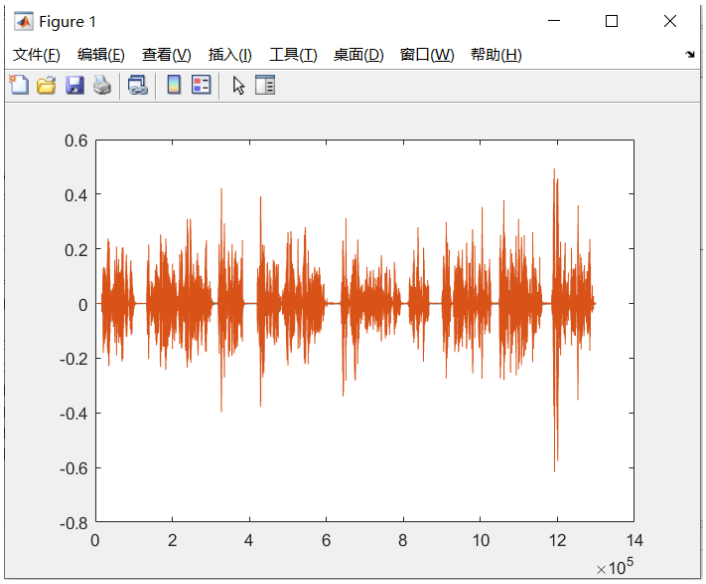
# 3 实验过程

## 3.1 音频准备

实验需要用到**语音素材**，因此我录制了一个长约27秒的**音频**(*voice.wav*)，在matlab中进行试读取，编写如下代码：

```
%%
% 初始化：清除内存，关闭所有图形窗口，并清空命令窗口
clc;
clear variables; % 使用clear variables代替clear all以清除工作空间变量
close all;
% 读取音频文件
[signal, ~] = audioread('voice.wav');
% 绘制时域信号
figure; % 绘图指令，确保图形在新窗口中打开
plot(signal);
title('Time Domain Signal');
```

运行这个**代码节**，可以得到音频的波形图：




## 3.2 FFT

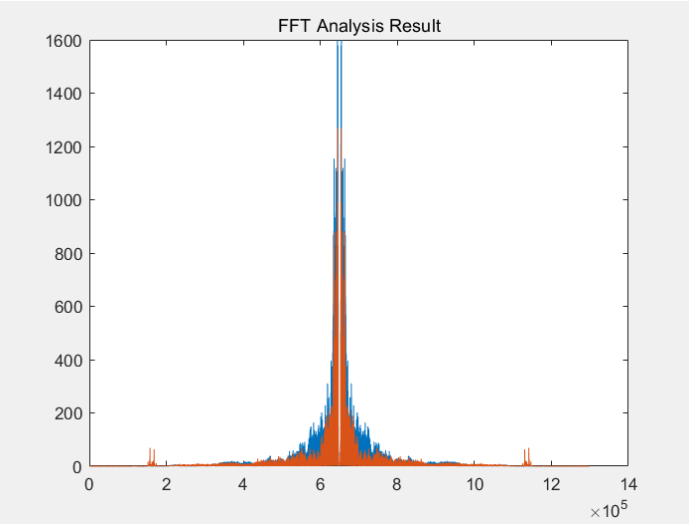
FFT利用了DFT的对称性和周期性，通过分治策略将原始DFT分解为较小的DFTs，从而减少了计算量。

编写如下代码：

```
%%
clc;
clear variables;
close all;
% FFT分析
[signal, freq] = audioread('voice.wav');
signalFFT = fft(signal);
plot(abs(fftshift(signalFFT)));
title('FFT Analysis Result');
```

1. **FFT变换**：首先使用`fft(signal)`对音频信号进行快速傅里叶变换，这一步骤将信号从时域转换到频域，得到信号的频谱表示。
2. **频谱中心化**：`fftshift(signalFFT)`将FFT的结果进行中心化处理，使得频谱的低频部分位于中心，高频部分分布在两侧，这样做更符合一般的频谱显示习惯。
3. **绘制频谱图**：通过`plot(abs(...))`绘制FFT变换后信号的幅度谱。使用`abs`是因为FFT的结果是复数，我们需要其模长来表示幅度信息。

运行结果:




## 3.3 DWT

DWT的基本思想是将信号分解为一系列的近似和细节，这些近似和细节由小波函数（也称为母小波）通过不同的缩放和平移得到。这个过程可以递归进行，以得到多个层次的分解，从而在不同的尺度上分析信号。

针对离散小波的多层次分解，我在这里完成了三种处理。分别如下：

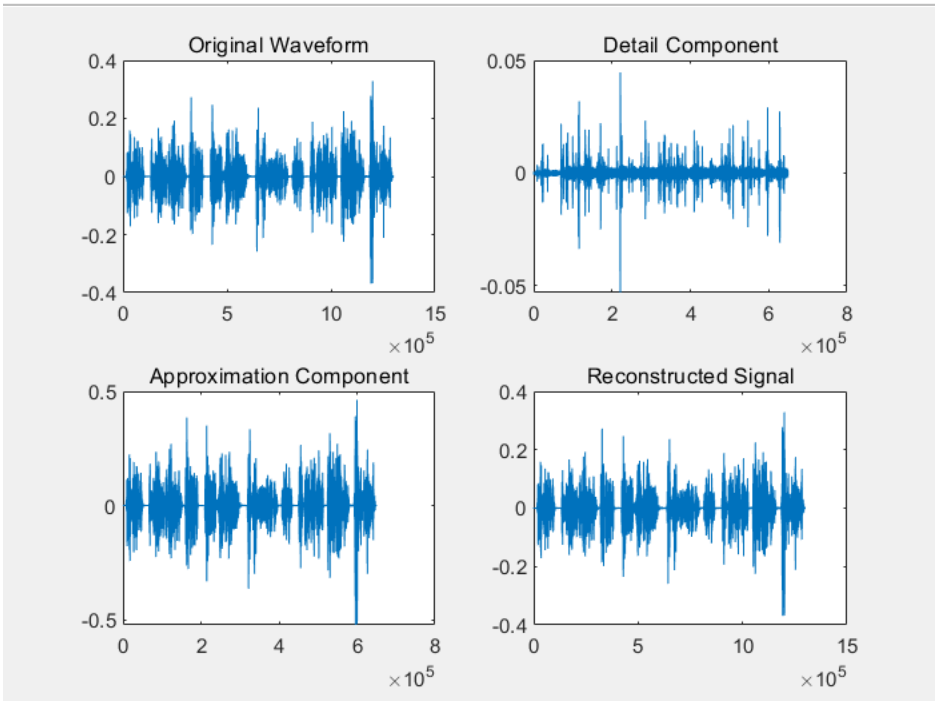
### 3.3.1 一级小波分解($dwt$)

编写如下代码:

```m
%%
clc;
clear variables;
close all;
% DWT分析
[origSignal, origSampleRate] = audioread("voice.wav");
[approxComponents, detailComponents] = dwt(origSignal(:,1), 'db4');
reconstructedSignal = idwt(approxComponents, detailComponents, 'db4', length(origSignal(:,1)));
% 绘制DWT分析结果
subplot(2, 2, 1); plot(origSignal(:, 1)); title('Original Waveform');
subplot(2, 2, 2); plot(detailComponents); title('Detail Component');
subplot(2, 2, 3); plot(approxComponents); title('Approximation Component');
subplot(2, 2, 4); plot(reconstructedSignal); title('Reconstructed Signal');
```

1. **一级小波分解**：利用`dwt`函数，对音频信号执行一级小波分解。此处选用的是'Daubechies 4'（`'db4'`）小波。分解结果包括近似分量`approxComponents`和细节分量`detailComponents`。
2. **重构信号**：通过`idwt`函数，使用近似分量和细节分量重构信号。这验证了小波分解的逆过程能够还原原始信号。
3. **结果展示**：展示了原始信号、细节分量、近似分量以及重构信号的波形，直观地理解小波分解和重构的效果。

运行结果：



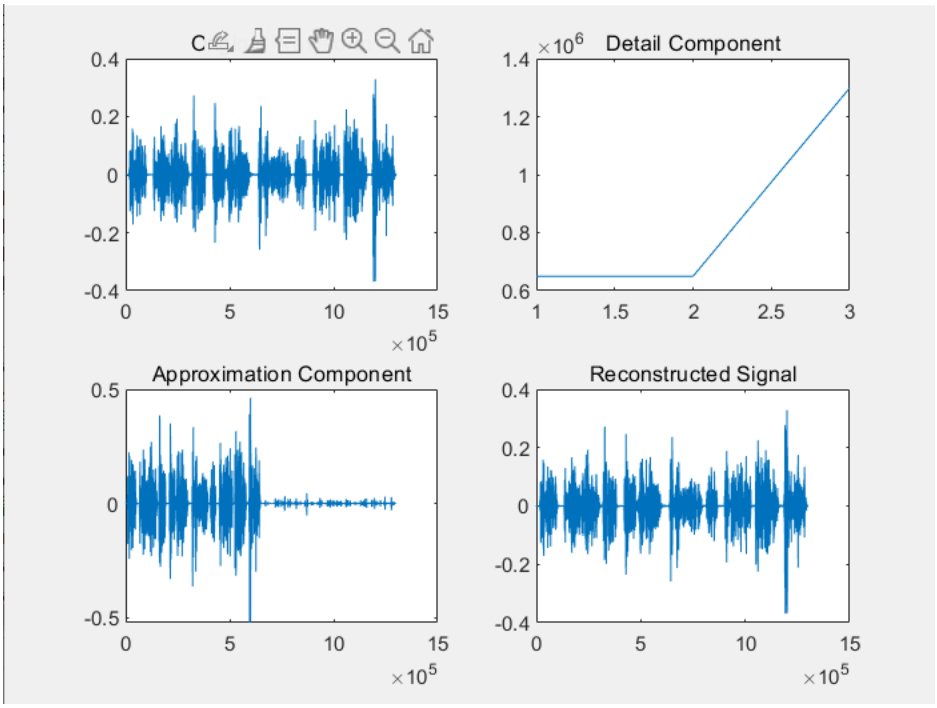
### 3.3.2 一级小波分解(*wavedec*)

编写如下代码:

```
%%
clc;
clear variables;
close all;
% 使用wavedec和waverec进行DWT分解和重构
[signalDWT, signalRate] = audioread("voice.wav");
[coefficients, levels] = wavedec(signalDWT(:,1), 1, 'db4');
reconstructedDWT = waverec(coefficients, levels, 'db4');
% 绘图展示分解和重构结果
subplot(2, 2, 1); plot(signalDWT(:, 1)); title('Original Waveform');
subplot(2, 2, 2); plot(levels); title('Detail Component');
subplot(2, 2, 3); plot(coefficients); title('Approximation Component');
subplot(2, 2, 4); plot(reconstructedDWT); title('Reconstructed Signal');
```

1. **小波分解**: `wavedec` 函数进行了更详细的小波分解，允许对信号执行多级分解。这里仍然是一级分解，使用 `'db4'` 小波。
2. **重构信号**: `waverec` 函数使用分解得到的系数和层级信息重构原始信号，验证小波分解的逆过程。
3. **结果展示**: 类似于 `dwt`，展示了原始信号、小波分解得到的系数（这里的绘制方法可能需要修正，以更准确展示近似和细节分量），以及重构信号的波形。

运行结果:



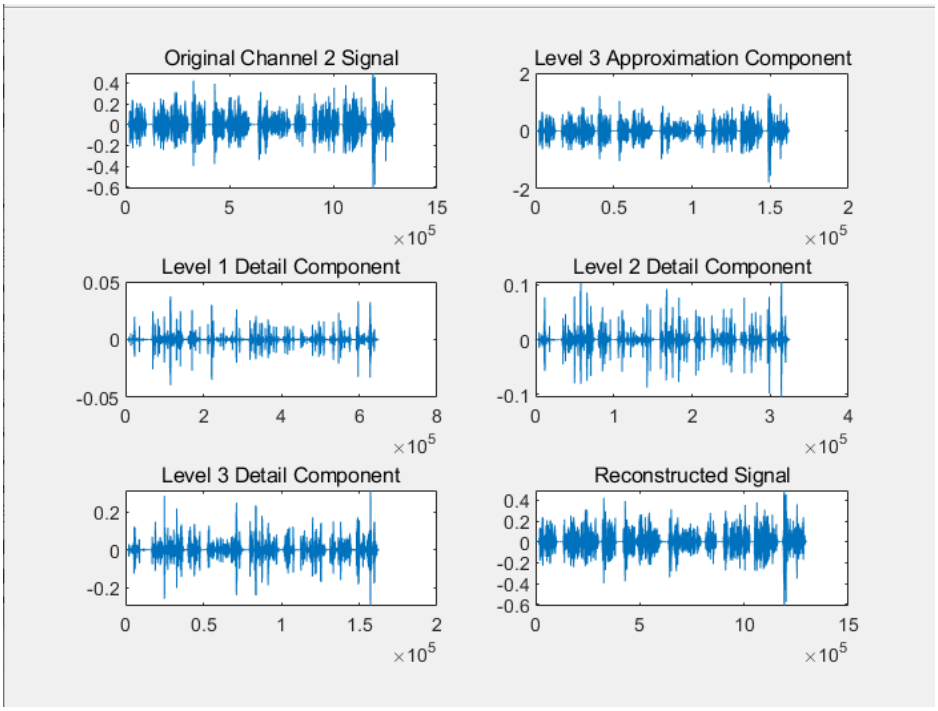
### 3.3.3 三级小波分解(*wavedec*)

编写如下代码:

```
%%
clc;
clear variables;
close all;
% 更深层次的DWT分解和重构
[deepDWTSignal, deepSampleRate] = audioread('voice.wav');
[deepCoefficients, deepLevels] = wavedec(deepDWTSignal(:,2), 3, 'db4');
% 提取各级分解的近似和细节成分
approx3 = appcoef(deepCoefficients, deepLevels, 'db4', 3);
detail3 = detcoef(deepCoefficients, deepLevels, 3);
detail2 = detcoef(deepCoefficients, deepLevels, 2);
detail1 = detcoef(deepCoefficients, deepLevels, 1);
reconstructedDeepDWT = waverec(deepCoefficients, deepLevels, 'db4');
% 绘图展示各级分解和重构结果
subplot(3, 2, 1); plot(deepDWTSignal(:, 2)); title('Original Channel 2
Signal');
subplot(3, 2, 2); plot(approx3); title('Level 3 Approximation Component');
subplot(3, 2, 3); plot(detail1); title('Level 1 Detail Component');
subplot(3, 2, 4); plot(detail2); title('Level 2 Detail Component');
subplot(3, 2, 5); plot(detail3); title('Level 3 Detail Component');
subplot(3, 2, 6); plot(reconstructedDeepDWT); title('Reconstructed
Signal');
```

1. **多级小波分解**：通过`wavedec`实现多级分解，这里进行了三级分解。每一级分解都提供了信号的不同层次的近似和细节信息。
2. **近似和细节成分提取**：利用`appcoef`和`detcoef`提取不同级别的近似和细节成分，展现信号的多层次特性。
3. **结果展示**：展示了原始信号、三个级别的细节分量以及近似分量，和通过这些成分重构的信号，展现了小波分解在多尺度分析中的能力。

运行结果:




## 3.4 DCT

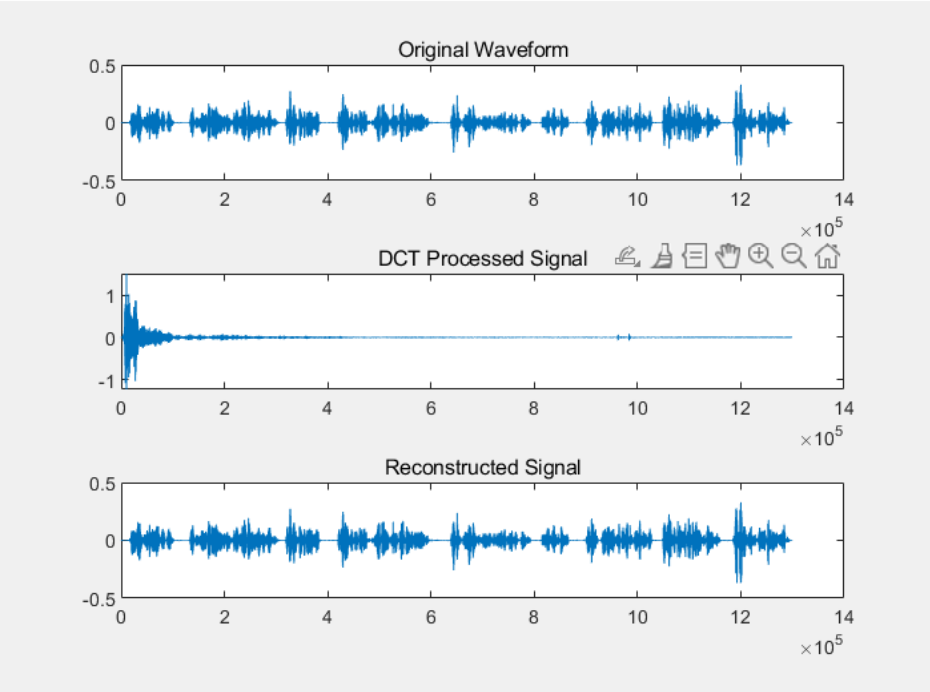
DCT通过分解信号为一系列余弦函数的和，这些余弦函数的频率成分表明了信号的频率特性。

编写如下代码：

```
%%
clc;
clear variables;
close all;
% DCT分析
[signalDCT, rateDCT] = audioread('voice.wav');
dctSignal = dct(signalDCT(:, 1));
reconstructedDCT = idct(dctSignal);
% 绘制DCT分析
subplot(3, 1, 1); plot(signalDCT(:, 1)); title('Original Waveform');
subplot(3, 1, 2); plot(dctSignal); title('DCT Processed Signal');
subplot(3, 1, 3); plot(reconstructedDCT); title('Reconstructed Signal');
```

1. **DCT变换**：通过`dct`函数对信号执行离散余弦变换，提取其频率成分。
2. **重构信号**：使用`idct`函数，利用DCT变换的结果重构原始信号。这验证了DCT变换的逆变换能力。
3. **结果展示**：展示了原始信号的波形、DCT变换后的信号以及重构后的信号，直观地理解DCT在信号分析和压缩中的应用。

运行结果：



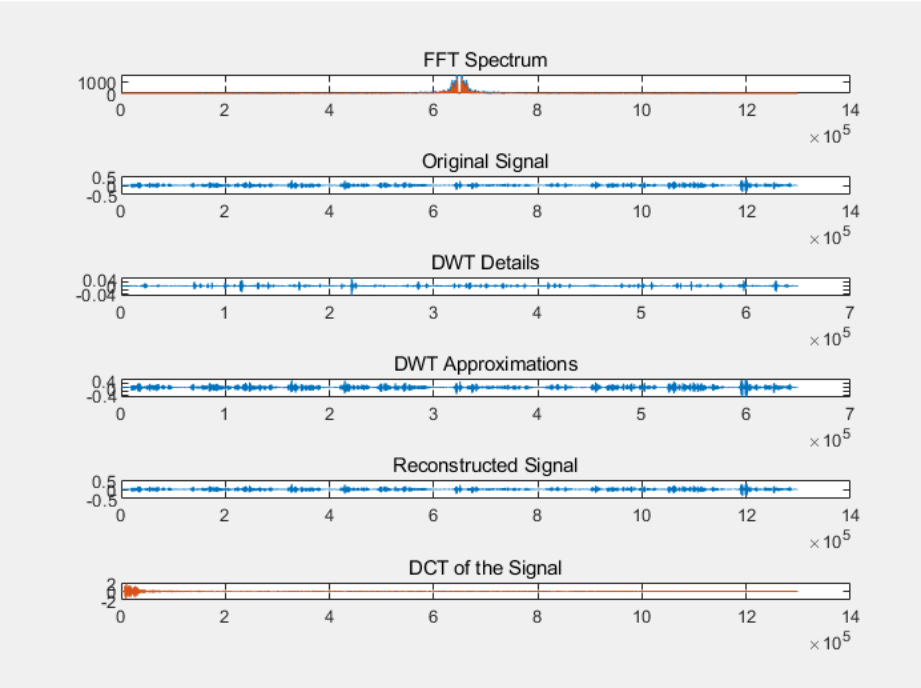

## 3.5 总结

集合上述各个代码，我们可以将其总结在一张图之中，编写如下代码：

```
%%
clc;
clear variables;
close all;
% 再次读取音频文件进行DWT分析
[signal, samplingRate] = audioread('voice.wav');
% FFT分析
signalFFT = fft(signal);
subplot(6,1,1);
plot(abs(fftshift(signalFFT)));
title('FFT Spectrum');
% DWT变换
[dwtApprox, dwtDetail] = dwt(signal(:,1), 'db4');
% IDWT重构信号
reconstructedSignal = idwt(dwtApprox, dwtDetail, 'db4',
length(signal(:,1)));
% 绘制DWT分析结果
subplot(6,1,2); plot(signal(:,1)); title('Original Signal');
subplot(6,1,3); plot(dwtDetail); title('DWT Details');
subplot(6,1,4); plot(dwtApprox); title('DWT Approximations');
subplot(6,1,5); plot(reconstructedSignal); title('Reconstructed Signal');

% DCT变换
dctResult = dct(signal);
subplot(6,1,6); plot(dctResult);
title('DCT of the Signal');
```

运行结果：




这样，我们就把上述的一些结果集中展示了出来，但是由于音频稍微过长，比例有一些不协调。

# 4 实验心得

通过完成这次实验，我深刻体会到了FFT、DWT和DCT在信号处理领域的强大应用。实验不仅让我掌握了这些变换的基本原理和数学模型，还通过实际编程练习加深了我的理解。特别是，我对于如何利用这些技术进行信号的分析、压缩和重构有了更为直观的认识。通过观察不同变换对同一信号的处理效果，我了解到每种变换在特定应用场景下的优势和局限性。这次实验不仅增强了我的实践能力，也激发了我对信号处理深入学习的兴趣，为我后续的学习和研究奠定了坚实的基础。