【注意】本文档中所有蓝色文字，是教师对内容的要求或补充说明。学生在提交报告前，应将它们全部删除。

**目录**

[1. 简介](#_Toc4991)

[2. 增采样方法设计](#_Toc4788)

[2.1. 基于数字滤波的方法](#_Toc8778)

[2.2. 基于时域插值的方法](#_Toc27223)

[2.3. 效果比较与分析](#_Toc4716)

[3. 实际语音信号增采样](#_Toc11115)

[3.1. 程序实现与封装](#_Toc14056)

[3.2. 主观测试与分析](#_Toc12915)

[3.3. 客观测试与分析【可选】](#_Toc16761)

[4. 设计改进【可选】](#_Toc15460)

[5. 总结](#_Toc3294)

[5.1. 设计完成情况](#_Toc27698)

[5.2. 经验与收获](#_Toc24226)

[6. 附录——相关技术简介](#_Toc3578)

**要求**：定稿后更新目录，以保证目录的正确

# 简介

数字信号的增采样，是指将低采样率的离散时间序列，通过一定的处理手段，转化成高采样率的离散时间序列。增采样广泛应用于信号混合、多采样率信号处理等领域。

本课程设计运用Python语言，设计并实现多种对语音信号的增采样方法，并比较不同方法的效果的客观和主观差异，以及分析形成差异的原因。

以上段落文字可以直接复制使用。

# 增采样方法设计

**要求**

* 按照 workbench.ipynb 的指引，完成增采样方法初步设计。

## 基于数字滤波的方法

### 基本原理及滤波器设计

**要求**

* 描述方法的基本原理。
* 制定滤波器的技术指标，以及制定依据。
* 介绍一个IIR和一个FIR滤波器的设计代码。

### 完整实现与验证

**要求**

* 介绍完整实现该方法的增采样函数 upsample\_filter() 的代码
* 介绍作图的代码（比较增采样结果 x\_filter 与目标结果 x\_target）
* 介绍计算MSE的代码

### 最优方案

**要求**

* 尝试不同的滤波器类型和参数，找到最优方案，即 MSE 最小的滤波器及其参数。

## 基于时域插值的方法

### 基本原理

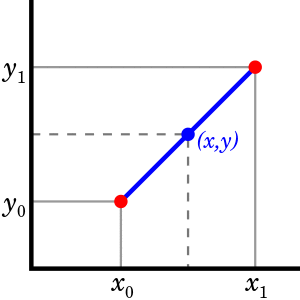
* “最近邻域”插值法的基本原理

最简单的插值算法，直接通过最近的时域采样点的数值生成增采样的数值。算法的优点是计算量小、简单、速度快，缺点是时域图像会产生锯齿，增采样效果一般。

* “线性”插值法的基本原理

在数学中，线性插值是一种使用线性多项式在一组已知数据点离散范围内构造新数据点的曲线拟合方法。

如果两个已知点由坐标和给出，则线性插值是这些点之间的直线。对于间隔中的值*，*沿直线的值从方程中给出



* 实现插值法的代码

f\_interp = interpolate.interp1d(x\_sub, x, kind=kind)

实现插值法的代码如上，所使用的函数为 SciPy 函数库中的 interpolate.interp1d 函数，即为一维插值函数。其中的参数为

|  |  |
| --- | --- |
| 参数 | 作用 |
| x\_sub | 序列的下标 |
| x | 需要增采样的序列 |
| kind | 增采样方式 |

该段代码实现了声明一个函数 f\_interp，当需要进行增采样时，可以像这样调用

x\_interp = f\_interp(x\_len\_up)

其中，x\_len\_up 为增采样后的序列的长度

### 完整实现与验证

* 完整实现该方法的增采样函数 upsample\_interp() 的代码。

def **upsample\_interp**(x, up\_factor, kind='zero'):

    """

    输入低采样率的信号，输出通过指定插值方法得到的增采样信号

    Args:

        x: array, 待插值信号

        up\_factor: int, 上采样因子

        kind: str, 默认是 'zero'

    Returns:

        x\_interp: array, 增采样后序列

    """

    x\_len = **len**(x)

    x\_sub = np.linspace(0, x\_len, x\_len)

    x\_len\_up = np.linspace(0, x\_len, 6 \* x\_len)

    f\_interp = interpolate.interp1d(x\_sub, x, kind=kind)

    x\_interp = f\_interp(x\_len\_up)

    return x\_interp

对于其中的参数 kind，有如下候选值可供选择：

|  |  |
| --- | --- |
| 候选值 | 作用 |
| 'zero', 'nearest' | 阶梯插值，相当于零阶B样条曲线 |
| 'slinear', 'linear' | 线性插值，用一条直线连接所有的取样点, 相当于一阶B样条曲线 |
| 'quadratic', 'cubic' | 二阶和三阶B样条曲线，更高阶的曲线可以直接使用整数值指定 |

* 作图代码（比较增采样结果 x\_interp 与目标结果 x\_target）。

def **gen\_comp\_plots**(x\_interp, x\_target, fs\_up):

    """

    比较增采样结果 x\_interp 与目标结果 x\_target

    Args:

        x\_interp: array, 增采样后的信号

        x\_target: array, 目标信号

        fs\_up: int, 增采样后的频率

    """

    fig, axs = plt.subplots(3, 2, figsize=(13,7))

    fig.tight\_layout(h\_pad=3)

    TITLES = ['sequence', 'spectrum']

    plot\_signals(['x\_interp'], [x\_interp], fs\_up, axs[0,:], TITLES)

    plot\_signals(['x\_target'], [x\_target], fs\_up, axs[1,:], TITLES)

    plot\_signals(

        ['x\_target', 'x\_interp'],

        [x\_target, x\_interp],

        fs\_up, axs[2,:],

        TITLES

    )

* 计算MSE的代码。

def **gen\_MSE**(records\_real, records\_predict):

    """

    获得均方差

    Args:

        records\_real: array, 真实值

        records\_predict: array, 目标值

    Returns:

        float, 均方差

    """

    if **len**(records\_real) == **len**(records\_predict):

        return **sum**(**map**(lambda x, y: (x - y)\*\*2, records\_real, records\_predict)) / x\_len\_up

    else:

        return None

### 最优方案

本次实验设计了一个函数用于获取最优方案，代码如下：

def **best\_insterp**():

    MSEs = {}

    kinds = ('zero', 'nearest', 'slinear', 'linear', 'quadratic', 'cubic')

    for kind in kinds:

        x\_interp = upsample\_interp(x, up\_factor, kind=kind)

        MSEs[kind] = gen\_MSE(x\_interp, x\_target)

    for key, value in MSEs.items():

**print**("\nMethod: " + key)

**print**("MSE:    " + str(np.round(value, 4)))

    fig, ax = plt.subplots(figsize=(5, 4))

    i = 0

    for value in MSEs.values():

        ax.scatter(i, value)

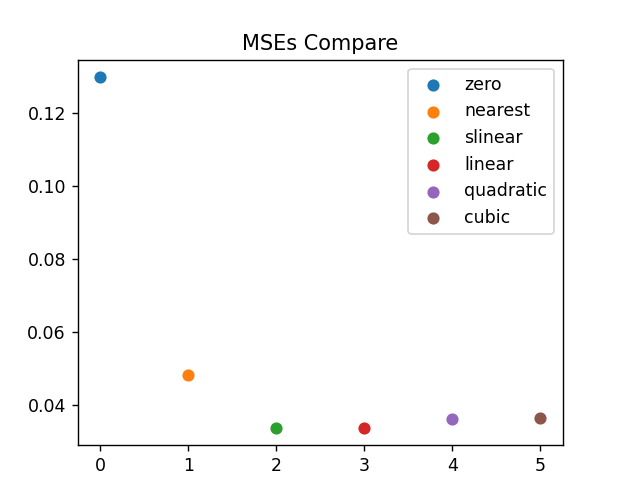
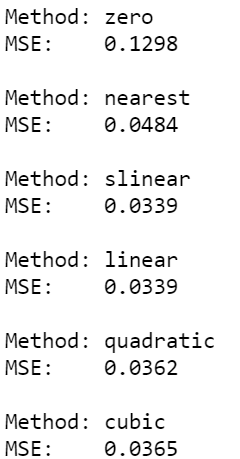
        i += 1

    ax.legend(MSEs.keys())

    ax.set\_title('MSEs Compare')

该段代码实现了比较六种插值方法的MSE，以选出最优方案，即MSE最小的方案。

比较过程如下：

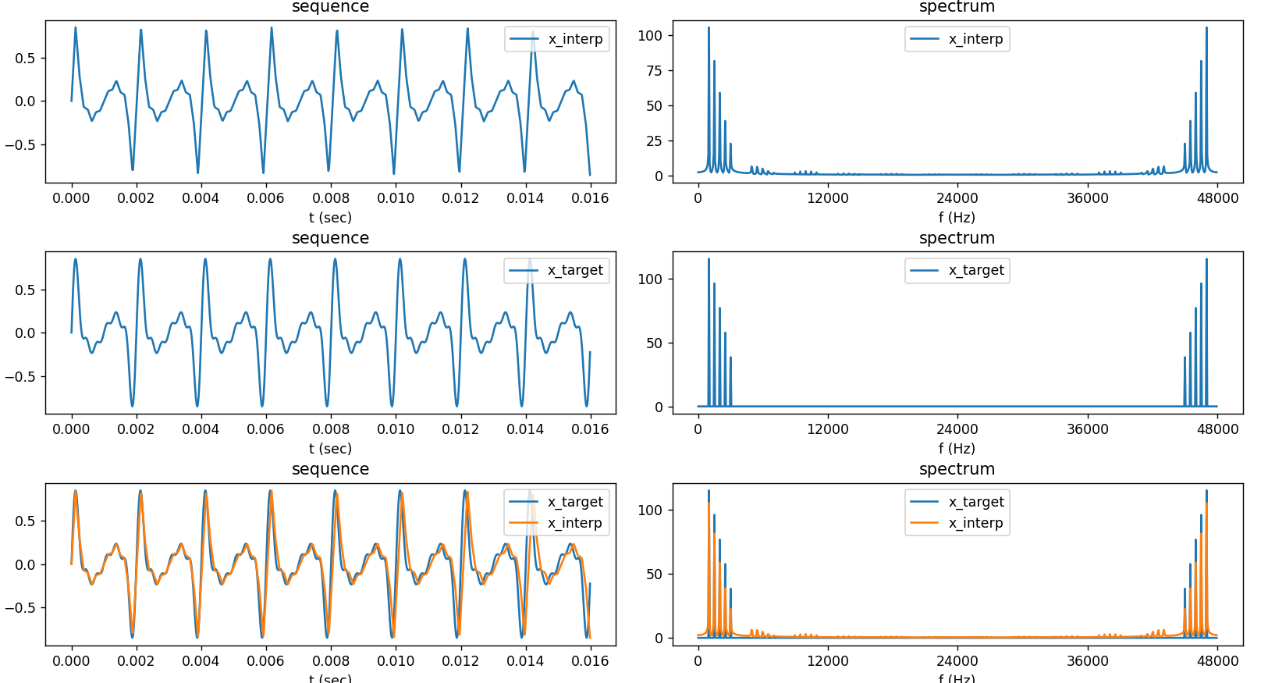


从以上数据可以看出，两个线性插值的均方差在所有其他插值方法中是最优的，两者在本次实验中使用并无区别。以下使用 linear 完成实验。

x\_interp = upsample\_interp(x,up\_factor, kind='linear')

gen\_comp\_plots(x\_interp, x\_target, fs\_up)

对函数进行以上调用，得到增采样后的信号以及与目标信号的可视化比较过程，图示如下：



可见，使用线性插值的方法得到的增采样信号与目标信号的差距已经不大，是比较优秀的插值方法。

## 效果比较与分析

**要求**

* 比较基于数字滤波、基于时域插值这两种方法中各自的最优方案。
  + 作图比较两种方法的增采样结果 x\_filter、x\_interp 与目标结果 x\_target。
  + 比较两种方法的增采样结果的MSE。
* 分析两种方法差异产生的原因。

# 实际语音信号增采样

## 程序实现与封装

**要求**

* 编写可在命令行独立运行的Python程序，实现对音频文件的增采样。
  + 命令行参数包括：增采样方案、输入音频文件名、输出音频文件名。
  + 增采样方案提供4种选择：IIR滤波、FIR滤波、最近邻域插值、线性插值
  + 输入音频文件为wav格式，采样率固定8KHz。
  + 输出音频文件为wav格式，采样率固定48KHz。
* 【提示】注意保存输出音频文件的时候，确定数据类型是否正确。

## 主观测试与分析

**要求**

* 一位组员作为测试组织者，其他组员作为测试参与者
  + 组织者将4个不同方案的增采样结果文件改名，令到参与者无法从文件名中了解该文件是从哪个方法增采样得到。
  + 参与者使用音频播放软件，带耳机或在安静的环境中听音，对比每个增采样结果文件和目标高采样率音频文件。分别对4个文件进行打分，与目标高采样率音频文件越接近的越高分。
  + 组织者收集所有参与者的评分结果，统计得出每种增采样方案的平均得分。
* 分析得分情况和预期是否相符，如果有显著差异，分析原因。

## 客观测试与分析【可选】

**要求**

* 编写可在命令行独立运行的Python程序，计算增采样结果与目标结果之间的MSE。
  + 命令行参数包括：增采样音频文件名1、目标音频文件名2、文件1起始样本、文件2起始样本、比较样本数。
  + 因为增采样音频文件相对于目标音频文件可能存在不同程度的延迟，需要通过指定“起始样本”、“比较样本数”来截断和对齐，实现有意义的比较。
* 计算以上4种增采样方案结果的MSE，分析是否与主观测试结果吻合。

# 设计改进【可选】

**要求**

* 提出对方法设计、实现、测试等方面的改进设想。
* 分析说明改进的目的、原理和可行性依据。
* 实施对系统的改进，并通过仿真验证改进的效果。

# 总结

## 设计完成情况

**要求**

* 列出已完成的内容
* 列出未完成的内容，及导致未能完成的原因

## 经验与收获

**要求**：在本次设计过程中，所得到的经验与收获

# 附录——相关技术简介

**要求：**简要介绍本设计中所使用的

* 计算机语言
  + 官方名称、版本号、官网URL
* 主要函数包/模组
  + 只需包含非标准函数包/模组等（不是计算机语言自带的）
  + 官方名称、版本号、官网URL
  + 在本设计中的用途
* 工具软件
  + 编程开发、报告撰写、作图、其他辅助工具等
  + 官方名称、版本号、官网URL
  + 在本设计中的用途