【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。

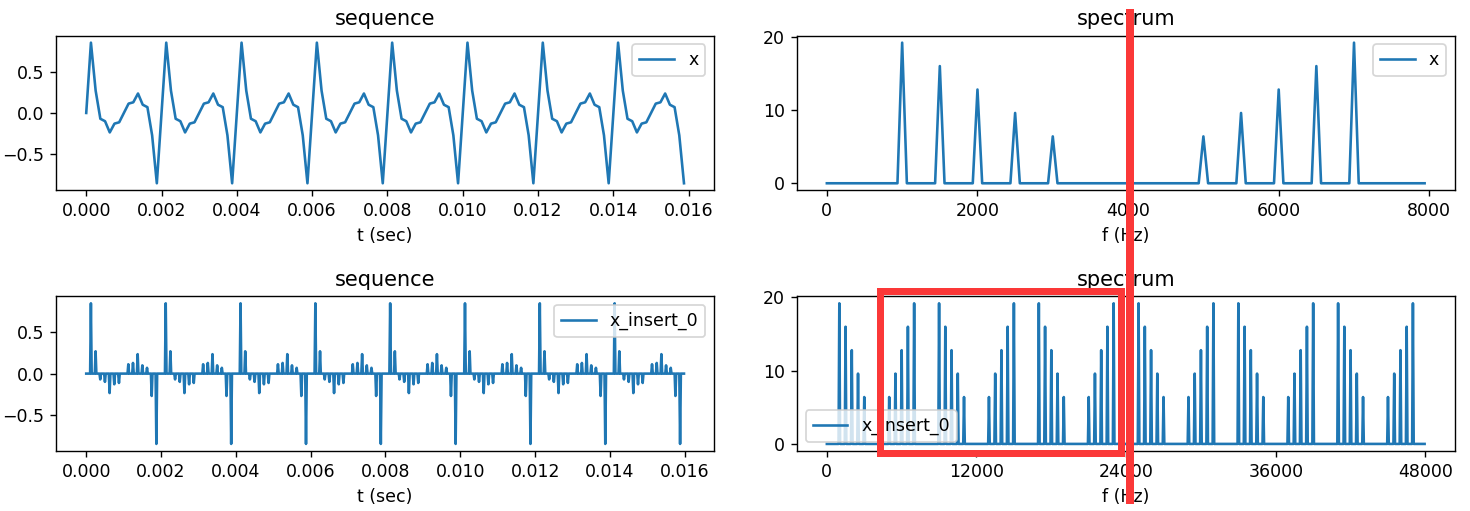
## 基于数字滤波的方法

### 基本原理及滤波器设计

* 基本原理

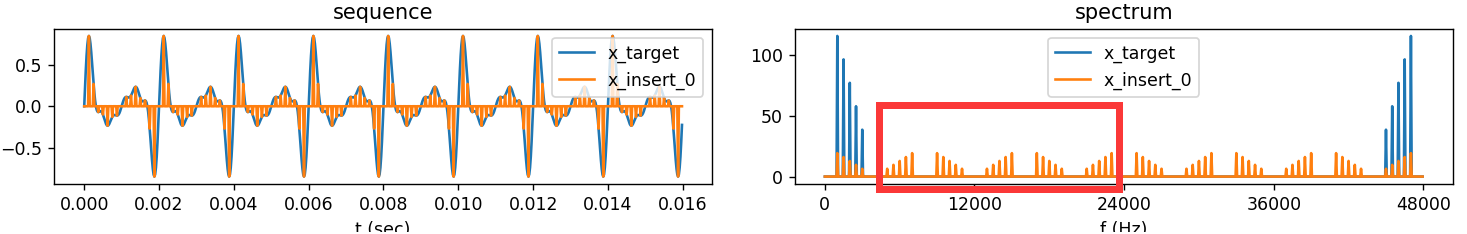
为了将低采样率的数字信号转化为高采样率的数字信号，可以在低采样率信号的相邻两个样点间插入个零值样点，使低采样率信号的采样间隔缩小倍，从而采样率提高倍，得到目标采样率。在本设计中，设为类似语音信号的低采样率的信号序列，其采样率为，欲将其采样率提高倍，即转化为采样率的高采样率信号，按照上述步骤，先在相邻两个样点间插入个零样点，得到中间信号。

利用Python生成、，并绘出两者的时域和频域图像对比如下：



其中左列为时域序列图像，右列为频谱图像；上下行分别为和。可以看出与相比，在频谱上多出了个镜像频谱（红框部分），同时幅度缩小倍。

利用Python生成目标高采样率(48KHz)信号，将中间信号和的时域频域图像画在同一图中进行对比如下：



其中蓝色波形为的图像，橙色波形为的图像。可以看出，与相比，在频域上同样多出的个镜像频谱（红框部分），同时幅值缩小倍。由此可知，增采样后的信号频谱中，只有与原信号相同的基带信号（频率位于内的频谱）是有用的，而多出的镜像频谱是多余的，因此为了滤除多余的这些镜像频谱，只提取基带信息，应让中间信号再通过一个低通滤波器，并将幅值增大倍，从而得到最终的高采样率的信号。

综上，基于数字滤波的增采样方法可分为两步：

1. 根据增采样因子，在低采样率信号的相邻两个样点间插入个零值样点，产生一个高采样率的中间信号；
2. 将中间信号通过低通滤波器进行滤波，并将信号幅度扩大6倍，得到接近目标高采样率信号的信号。

其增采样方法的结构如下：



* 滤波器的技术指标及制定依据

由上述基本原理可知，所需滤波为低通滤波器，器需要处理的信号的采样频率为目标采样频率，因此滤波器采样频率同样设为；滤波器需要保留位于内的频率分量，因此滤波器通带理想截止频率设为，且实际低通滤波器在通带和阻带之间还有过渡带，取理想截止频率为过渡带中心，左右各预留的宽度，即滤波器过渡带范围为；最后通带最大衰减设为一较小值，保证通带频率成分得到较好保留，阻带最小衰减设为一较大值，保证阻带频率成分得到充分衰减。综上，滤波器技术指标列举如下：

滤波器类型：低通滤波器

采样频率：

理想截止频率：

通带截止频率：

阻带截止频率：

通带最大衰减：

阻带最小衰减：

* IIR和FIR滤波器设计代码介绍

1. IIR滤波器设计代码介绍
2. 根据制定的滤波器技术指标定义相关参数变量：

# 滤波器的技术指标

fs **=** fs\_up # 采样频率，等于增采样后的信号采样频率

f\_c **=** fs\_x **/** 2 # 理想截止频率，等于原信号采样频率的一半

f\_p **=** f\_c **-** 0.5e3 # 通带截止频率

f\_st **=** f\_c **+** 0.5e3 # 阻带截止频率

f\_tran **=** f\_st **-** f\_p # 过渡带

R\_p\_traget **=** 1 # 通带最大衰减，in dB

A\_s\_traget **=** 42 # 阻带最小衰减，in dB

1. 选择一种模拟滤波器类型，使用scipy库中设计IIR滤波器的函数signal.iirdesign()，传入必要的滤波器参数，计算满足指标的数字滤波器的系数 b 和 a，并根据b和a的长度计算滤波器的阶数：

# 选择其中一种模拟滤波器类型进行尝试

#ftype='butter'

#ftype='cheby1'

ftype**=**'cheby2'

#ftype='ellip'

# 计算满足指标的数字滤波器的系数 b 和 a

b**,** a **=** signal**.**iirdesign**(**

f\_p**,** f\_st**,**

R\_p\_traget**,** A\_s\_traget**,**

ftype**=**ftype**,**

fs**=**fs**)**

forder **=** max**(**len**(**b**),** len**(**a**))** **-** 1 # 滤波器阶数

IIR\_a **=** a

IIR\_b **=** b

1. 使用教师提供的滤波器分析函数analyze\_filter()对滤波器性能进行分析，该函数将输出滤波器在通带、过渡带、阻带中不同频率的幅度和相位响应结果：

# Analyze the filter.

bands**=** **[**

**(**'pass'**,** 0**,** f\_p**),**

**(**'tran'**,** f\_p**,** f\_st**),**

**(**'stop'**,** f\_st**,** fs**/**2**)**

**]**

R\_p**,** A\_s **=** analyze\_filter**(**bands**,** b**=**b**,** a**=**a**,** show\_plot**=True,** fs**=**fs**,** tick\_format**=**tick\_format\_to\_khz**,** amp\_in\_dB**=True)**

**print(**'type = %s, order = %d, R\_p = %f, A\_s = %f' **%** **(**ftype**,** forder**,** R\_p**,** A\_s**))**

**print(**'b ='**,** b**)**

**print(**'a ='**,** a**)**

1. FIR滤波器设计代码介绍
2. 根据制定的滤波器技术指标定义相关参数变量，同上步骤1)。
3. 选择一种窗函数类型，根据窗函数计算过渡带宽和窗口长度，若窗函数为kaiser窗，则实使用scipy库中的函数signal.iirdesign()，根据阻带最小衰减和过渡带宽度计算kaiser窗长度和β参数：

# 选择下面其中一种窗函数类型进行尝试

#window = 'hann'

#window = 'hamming'

#window = 'blackman'

window **=** 'kaiser'

# 根据选择的窗函数，得到相应过渡带宽（凯泽窗除外）和窗口长度N

**if** **(**window **==** 'hann'**):**

window\_tran **=** 6.2 **\*** **(**fs**/**2**)**

N **=** int**(**np**.**ceil**(**window\_tran **/** f\_tran**))**

**elif** **(**window **==** 'hamming'**):**

window\_tran **=** 6.6 **\*** **(**fs**/**2**)**

N **=** int**(**np**.**ceil**(**window\_tran **/** f\_tran**))**

**elif** **(**window **==** 'blackman'**):**

window\_tran **=** 11 **\*** **(**fs**/**2**)**

N **=** int**(**np**.**ceil**(**window\_tran **/** f\_tran**))**

**elif** **(**window **==** 'kaiser'**):**

N**,** kaiser\_beta **=** signal**.**kaiserord**(**A\_s\_traget**,** **(**f\_st **-** f\_p**)/(**0.5**\***fs**))** # 计算凯泽窗口的长度N和β参数

1. 调用scipy库中设计FIR滤波器的函数signal.firwin()，传入必要的滤波器参数，计算满足指标的数字滤波器的系数 b 和 a：

# 计算满足指标的数字滤波器的系数 b 和 a

b **=** signal**.**firwin**(**N**,** f\_c**,** window**=(**window**,** kaiser\_beta**)** **if** **(**window **==** 'kaiser'**)** **else** window**,** fs**=**fs**)**

a **=** 1

FIR\_b **=** b

FIR\_a **=** a

1. 用教师提供的滤波器分析函数analyze\_filter()对滤波器性能进行分析：

# Analyze the filter.

bands**=** **[**

**(**'pass'**,** 0**,** f\_p**),**

**(**'tran'**,** f\_p**,** f\_st**),**

**(**'stop'**,** f\_st**,** fs**/**2**)**

**]**

**def** tick\_format\_to\_khz**(**value**,** tick\_number**):**

"""Show the value in kHz"""

**return** **(**'%.2f' **%** **(**value**/**1000**)).**rstrip**(**'0'**).**rstrip**(**'.'**)** **+** 'kHz'

R\_p**,** A\_s **=** analyze\_filter**(**bands**,** b**=**b**,** a**=**a**,** show\_plot**=True,** fs**=**fs**,** tick\_format**=**tick\_format\_to\_khz**,** amp\_in\_dB**=True)**

**print(**'type = %s, N = %d, R\_p = %f, A\_s = %f' **%** **(**window**,** N**,** R\_p**,** A\_s**))**

**print(**'b ='**,** b**)**

**print(**'a ='**,** a**)**

### 完整实现与验证

* 增采样函数 upsample\_filter() 代码介绍

1. 输入参数为低采样率信号x、增采样后信号的长度x\_len\_up、增采样因子up\_factor以及数字滤波器的系数b和a；
2. 先生成长度与增采样后的信号长度相同的0值数组，然后在0值数组中每隔 (增采样因子-1) 个数值插入一个原信号x的值，从而等效于在低采样率信号x的相邻两个样点间插入 (增采样因子-1) 个零值样点，得到插零序列x\_insert\_0；
3. 最后调用scipy库中的函数signal.lfilter ()对插零序列x\_insert\_0使用数字滤波器进行滤波，得到接近目标高采样率信号 x\_target 的信号 x\_filter：

**def** upsample\_filter**(**x**,** x\_len\_up**,** up\_factor**,** a**,** b**):**

'''

增采样处理函数，输入低采样率的信号 x，输出通过滤波方法得到的增采样信号 x\_filter。

输入参数：

x: array\_like

输入的低采样率信号x

x\_len\_up: int

增采样后信号的长度

up\_factor: int

增采样因子

a: array\_like

数字滤波器的系数a

b: array\_like

数字滤波器的系数b

返回：

x\_filter: array\_like

通过滤波方法得到的增采样信号 x\_filter

'''

# Insert zeros between samples.

x\_insert\_0 **=** np**.**zeros**(**x\_len\_up**)**

x\_insert\_0**[::**up\_factor**]** **=** x

# 插零后的信号序列输入数字滤波器进行滤波，并将滤波输出扩大6倍

x\_filter **=** signal**.**lfilter**(**b**=**b**,** a**=**a**,** x**=**x\_insert\_0**)** **\*** 6

**return** x\_filter

* 作图代码介绍

1. 输入参数为增采样后的信号x\_filter、目标高采样率信号x\_target以及IIR滤波器的模拟滤波器类型或FIR滤波器的窗函数类型ftype；
2. 由于数字滤波器的特性，x\_filter 相对于 x\_target 是有延迟，且需要经过一段时间才能达到稳定。因此，需要进行一定的处理才可得到有意义的比较。处理方法为根据数字滤波器所使用的窗函数类型或模拟滤波器类型截取x\_filter的“有效”部分并通过平移来尽量与x\_target对齐重叠。代码实现方法为利用切片截取x\_filter 和x\_target的相应“有效”部分并赋给新的数组变量x\_filter\_slice和x\_target\_slice；
3. 绘出x\_filter\_slice和x\_target\_slice的时域序列图像和频谱图像进行比较；

**def** comp\_plots**(**x\_filter**,** x\_target**,** ftype**):**

'''

作图比较函数，比较增采样结果 x\_filter 与目标结果 x\_target

输入参数:

x\_filter: array\_like

增采样后的信号

x\_target: array\_like

目标高采样率信号

ftype: str

数字滤波器所使用的窗函数类型或模拟滤波器类型

'''

# 根据ftype截取x\_filter和x\_target的“有效”部分，并通过平移来尽量对齐重叠。

**if(**ftype **==** 'butter'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**18**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**elif(**ftype **==** 'cheby1'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**14**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**elif(**ftype **==** 'cheby2'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**2**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**elif(**ftype **==** 'ellip'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**4**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**else:**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**(**N**/**2**))::]**

x\_target\_slice **=** x\_target**[**0**:**len**(**x\_filter\_slice**)]**

# Plot the signals for compare.

fig**,** axs **=** plt**.**subplots**(**3**,** 2**,** figsize**=(**12**,**6**))**

fig**.**tight\_layout**(**h\_pad**=**4**)**

TITLES **=** **[**'sequence'**,** 'spectrum'**]**

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

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

plot\_signals**(**

**[**'x\_target'**,** 'x\_filter'**],**

**[**x\_target\_slice**,** x\_filter\_slice**],**

fs\_up**,** axs**[**2**,:],**

TITLES

**)**

* MSE计算代码介绍

1. 输入参数为增采样后的信号x\_filter、目标高采样率信号x\_target以及IIR滤波器的模拟滤波器类型或FIR滤波器的窗函数类型ftype；
2. 根据数字滤波器所使用的窗函数类型或模拟滤波器类型利用切片截取x\_filter 和x\_target的相应“有效”部分并赋给新的数组变量x\_filter\_slice和x\_target\_slice；
3. 计算“有效”部分x\_filter\_slice和x\_target\_slice的MSE，MSE等于两者对应元素差值的平方和 的平均数：

**def** cal\_MSE**(**x\_filter**,** x\_target**,** ftype**):**

'''

计算MSE函数，计算增采样后的信号与目标高采样率信号有效对齐部分的 MSE（均方差）

输入参数:

x\_filter: array\_like

增采样后的信号

x\_target: array\_like

目标高采样率信号

ftype: str

数字滤波器所使用的窗函数类型或模拟滤波器类型

'''

# 根据ftype截取x\_filter和x\_target的“有效”部分，并通过平移来尽量对齐重叠。

**if(**ftype **==** 'butter'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**18**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**elif(**ftype **==** 'cheby1'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**14**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**elif(**ftype **==** 'cheby2'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**2**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**elif(**ftype **==** 'ellip'**):**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**((**forder**+**1**)/**2**+**4**)+**fs**/**1e3**\***2**)::]**

x\_target\_slice **=** x\_target**[**int**(**fs**/**1e3**\***2**):**len**(**x\_filter\_slice**)+**int**(**fs**/**1e3**\***2**)]**

**else:**

x\_filter\_slice **=** x\_filter**[**int**(**np**.**floor**(**N**/**2**))::]**

x\_target\_slice **=** x\_target**[**0**:**len**(**x\_filter\_slice**)]**

# 计算有效对齐部分的 MSE

MSE **=** sum**((**x\_target\_slice**-**x\_filter\_slice**)\*\***2**)** **/** len**(**x\_filter\_slice**)**

**print(**'type = %s, MSE = %f' **%** **(**ftype**,** MSE**))**

**return** **(**MSE**)**

### 最优方案

**要求**

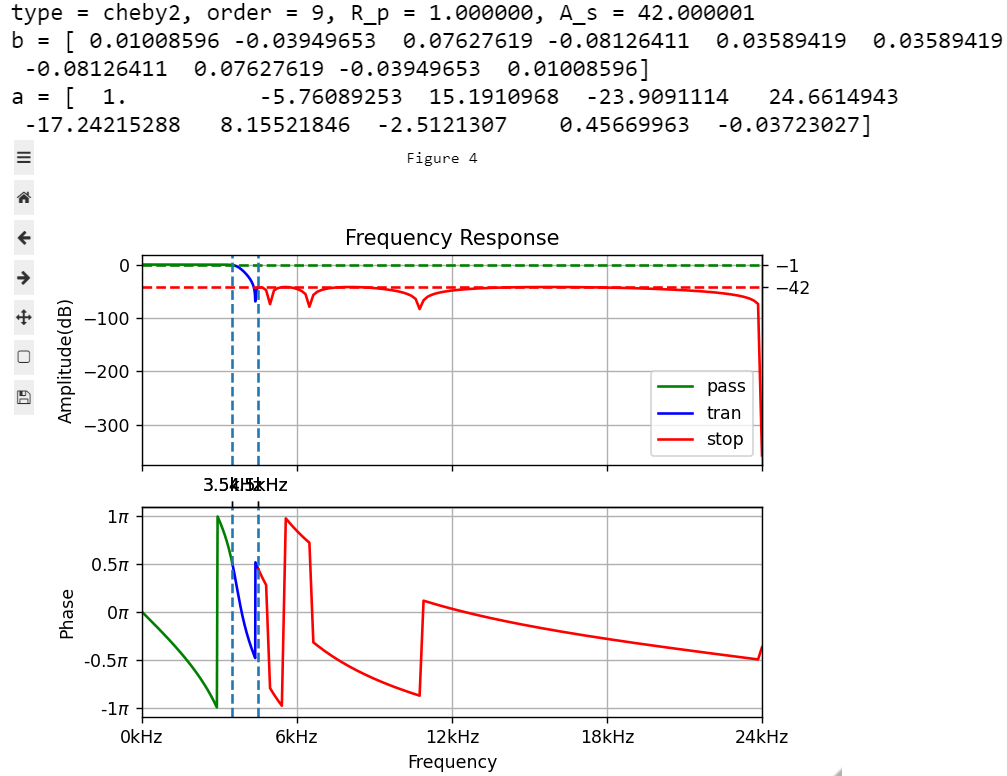
* 尝试不同的滤波器类型和参数，找到最优方案，即 MSE 最小的滤波器及其参数。

1. 在上述制定的滤波器技术指标的情况下，对IIR滤波器分别采用butter、cheby1、cheby2和ellip模拟滤波器进行设计，分别求得四种情况下的MSE和滤波器阶数如下：

|  |  |  |
| --- | --- | --- |
| **Type** | **MSE** | **Order** |
| butter | 0.00535 | 21 |
| cheby1 | 0.005715 | 9 |
| cheby2 | 0.002195 | 9 |
| ellip | 0.003524 | 5 |

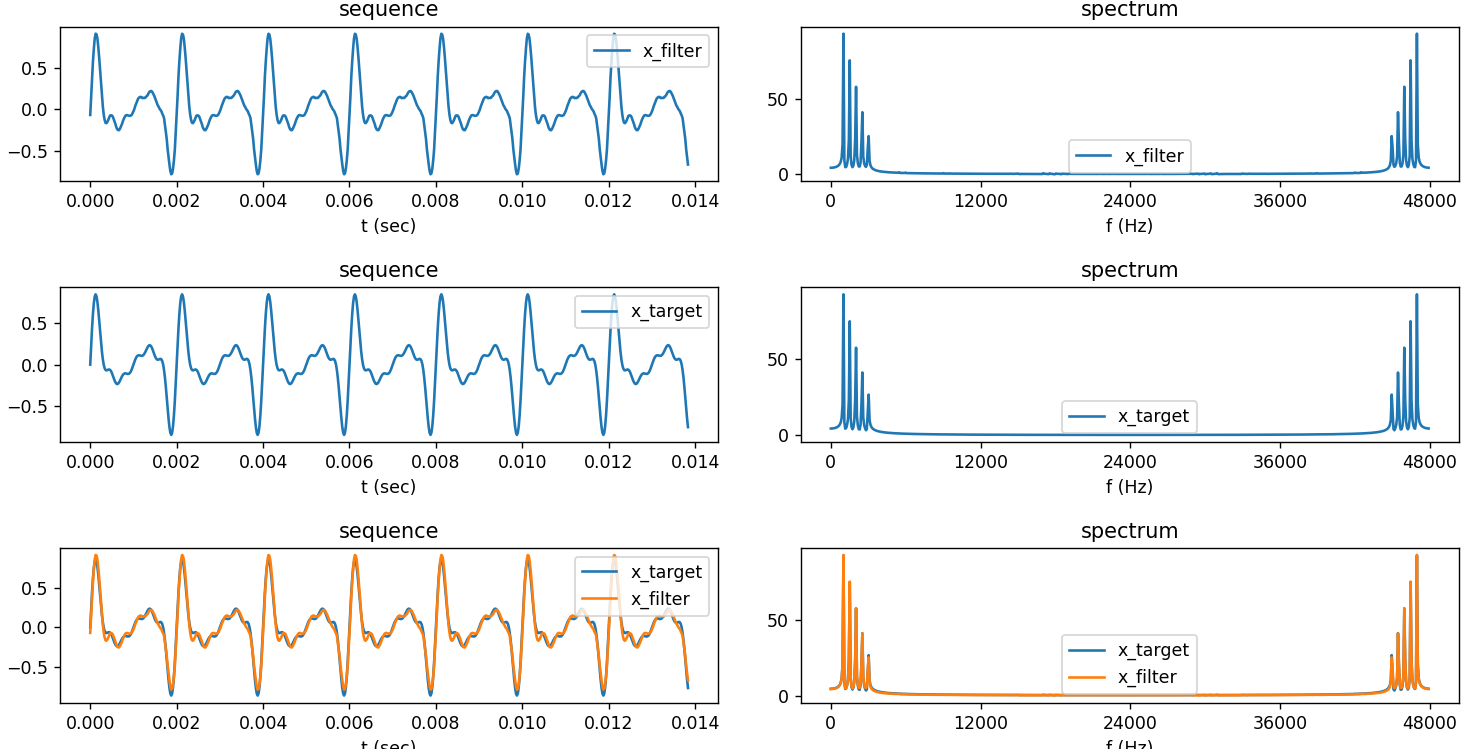
由表可知用cheby2模拟滤波器设计的IIR滤波器是MSE最小且阶数也较少的方案，可以认为是最优方案。

使用cheby2模拟滤波器设计的IIR滤波器性能分析如下：



可以看到该滤波器满足技术指标要求。

使用作图函数将信号 x\_filter 与 x\_target 对齐画在同一个坐标系中进行比较，该滤波器方案增采样效果如下：



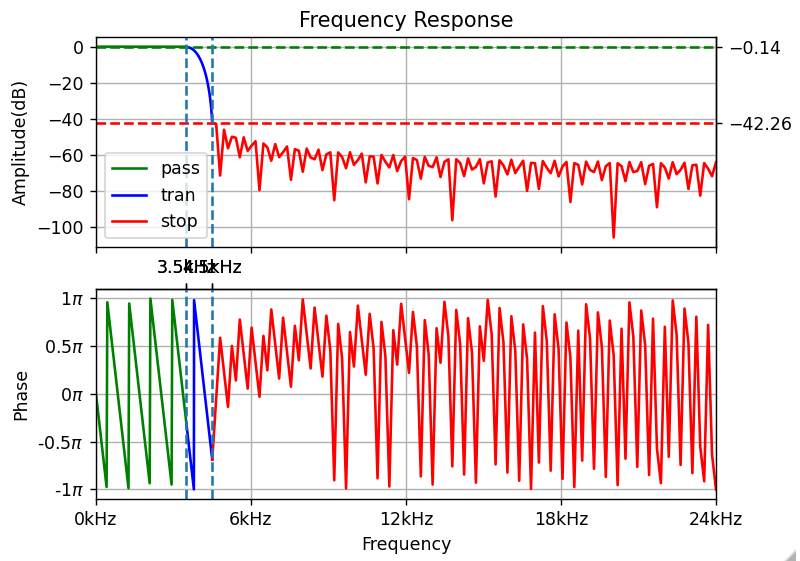
可以看到增采样后的信号x\_filter的时域和频域图像均与目标高采样率信号x\_target相接近，增采样效果良好。此外由于对x\_filter和x\_target进行了“有效”部分的截取，相当于在时域中加了窗函数，因此频谱跟原来未截断时相比有一定波动属正常现象。

1. 在上述制定的滤波器技术指标的情况下，对FIR滤波器分别采用hann、hamming、blackman、kaiser窗函数进行设计，分别求得四种情况下的MSE和窗口长度如下：

|  |  |  |
| --- | --- | --- |
| **Type** | **MSE** | **N** |
| hann | 0.000127 | 149 |
| hamming | 0.000131 | 159 |
| blackman | 0.001295 | 264 |
| kaiser | 0.000125 | 115 |

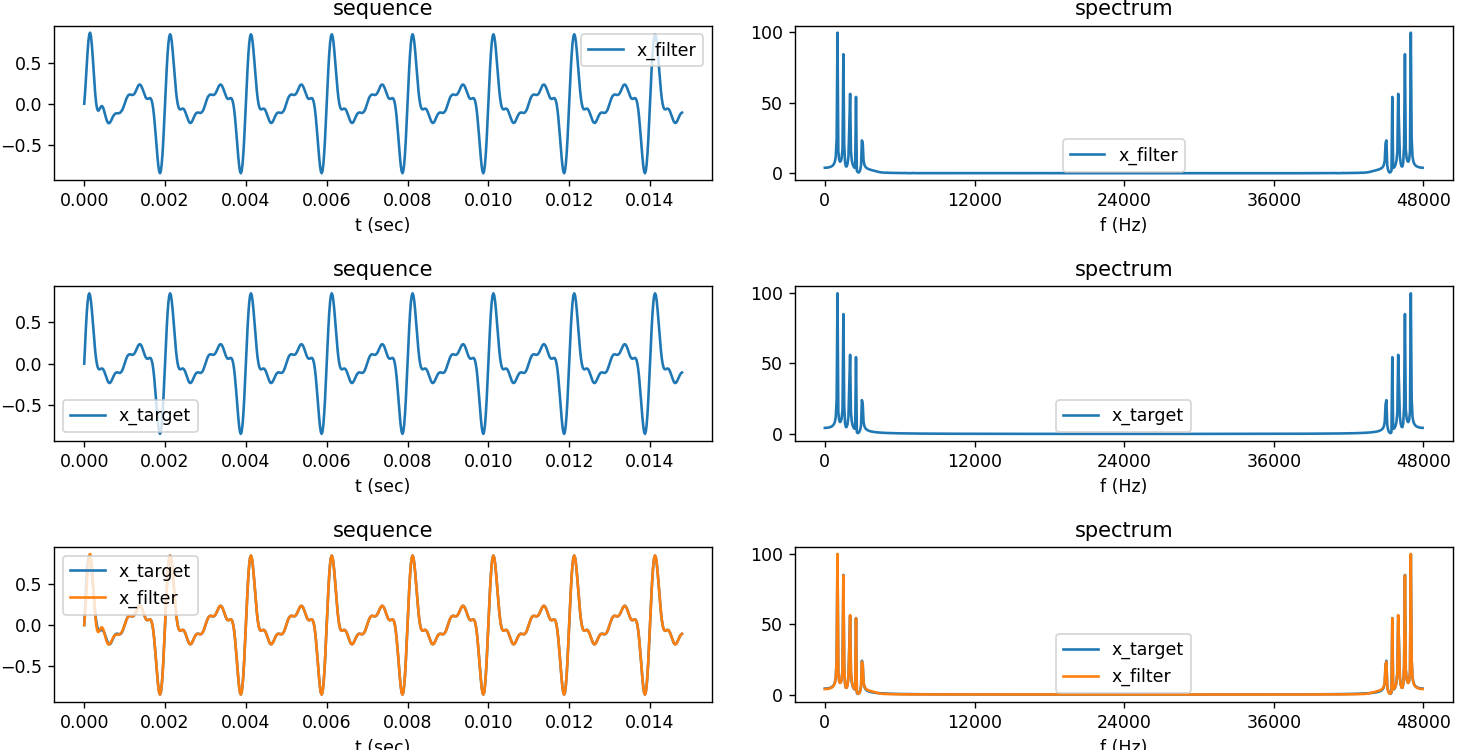
由表可知用kaiser窗设计的FIR滤波器是MSE和窗口长度N均最小的方案，可以认为是最优方案。

使用kaiser窗设计的FIR滤波器性能分析如下：



可以看到该滤波器满足技术指标要求。

使用作图函数将信号 x\_filter 与 x\_target 对齐画在同一个坐标系中进行比较，该滤波器方案增采样效果如下：



可以看到增采样后的信号x\_filter的时域和频域图像均与目标高采样率信号x\_target相接近，增采样效果良好。此外由于对x\_filter和x\_target进行了“有效”部分的截取，相当于在时域中加了窗函数，因此频谱跟原来未截断时相比有一定波动属正常现象。

## 基于时域插值的方法

### 基本原理

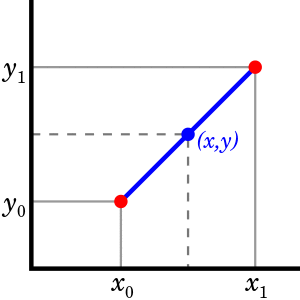
* “最近邻域”插值法的基本原理

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)

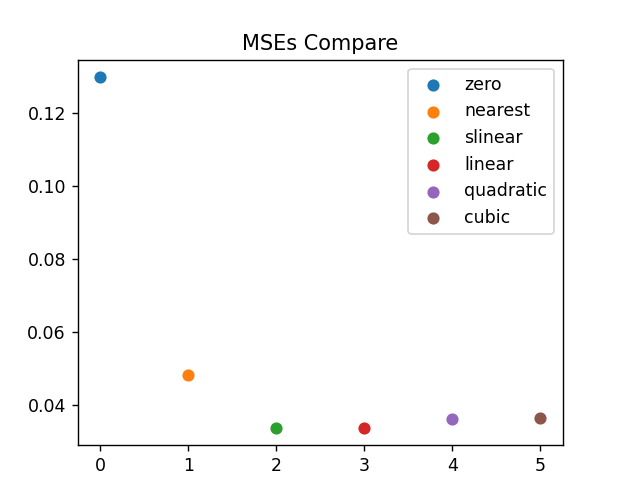
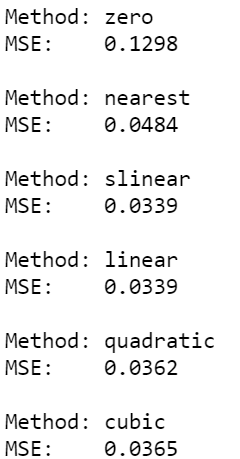
        i += 1

    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：

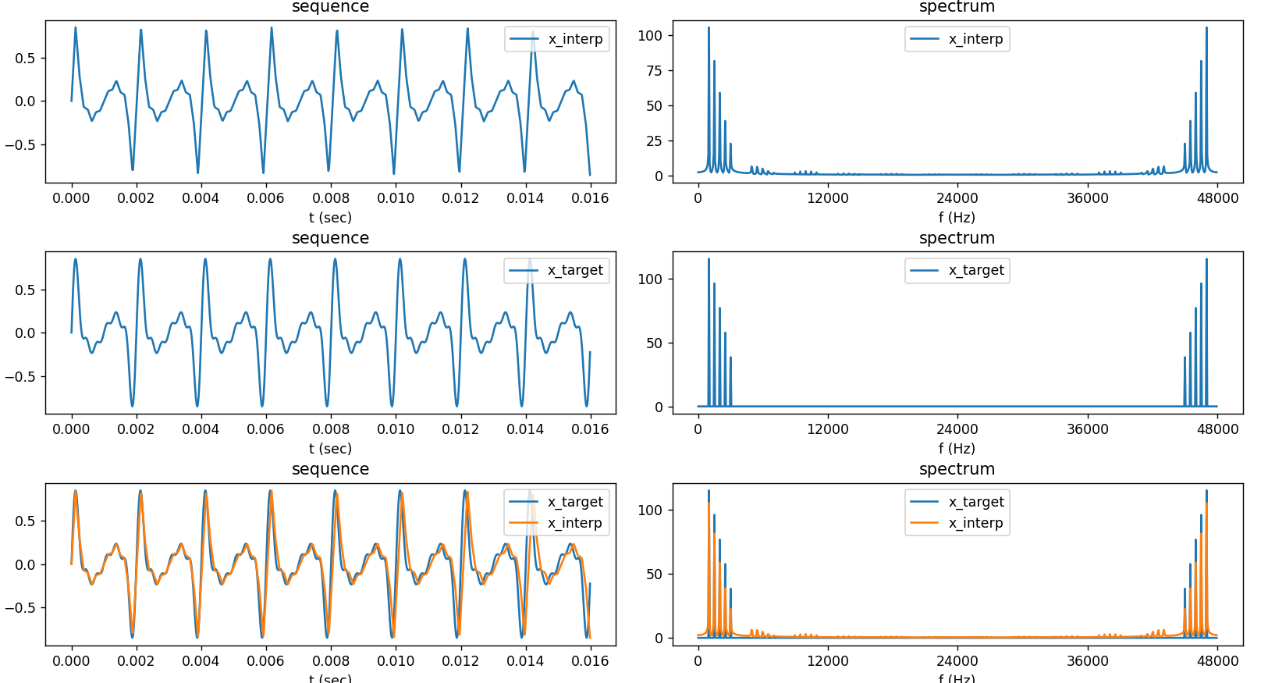


从以上数据可以看出，两个线性插值的均方差在所有其他插值方法中是最优的，两者在本次实验中使用并无区别。以下使用 linear 完成实验。

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

* 已完成的内容
* 基于数字滤波的增采样方法
* 基于时域插值的增采样方法
* 比较上述二者方法，选出最优方案
* 程序的实现与封装完成了主、客观测试及分析
* 未完成的内容
* 设计及改进实验

## 经验与收获

通过本次课程设计，首先学习到如何在python中对信号进行增采样的方法：一种为基于数字滤波的方法和基于时域插值的方法。在对上述设计方案进行验证时，采用了主客观结合的方式，进行验证，使方案更具合理性。

# 附录——相关技术简介

* 所用到的计算机语言
  + python
* 官方名称：

Python

* 版本号：

version 3.8.2

* 官网URL：

https://www.python.org

* 主要函数包/模组
  + scipy
* 官方名称：

Scipy

* 版本号：

version 1.5.4

* 官网URL：

<https://www.scipy.org>

* 用途：

模块里有fft、signal函数用于对信号进行频谱分析及处理。

* + ipython
* 官方名称：

Ipython

* 版本号：

version 7.19.0

* 官网URL：

https://ipython.org

* 用途：

用于使用python进行交互式计算，即生成.ipynb文件。

* 工具软件
  + Vs code
* 官方名称：

Visual Studio Code

* 版本号：

version 7.19.0

* 官网URL：

https://code.visualstudio.com/

* 用途：

用于使用编写python代码及调试。

* + WPS
* 官方名称：

WPS office

* 版本号：

version 11.1.0.10228

* 官网URL：

http://office.wps.cn

* 用途：

用于编写实验报告及整理相关实验数据和将相关数据绘制成表格。