一个连续的声音信号经过麦克风采样

会变成一个离散的时间信号!

$x(t) \xrightarrow{\text{sampling}} x[n]$

注意 n 只能为整数，对应到matlab里面一个数组.

举例：　$x(t) = \sin 10\pi t$　　$f_c = 10Hz$.

$0 \le t \le 5s.$

$x[n] = \sin \pi n$　(n为整数，$n = 1, 2, \cdots, 50$)

$x[k]$ 的含义是第k个采样值，

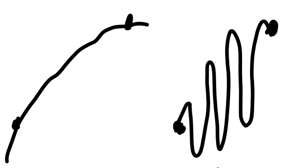
$x[k] = x(\frac{k}{f_c})$

由此我们得到了一个序列（采样完的信号）$x[n]$

离散的周期信号可以展开成傅里叶级数的形式。

即 $x[n]=\sum_{k=1}^{N} a_k e^{jk(\frac{2\pi}{N})n} = \sum_{k=2}^{N+1} a_k e^{jk(\frac{2\pi}{N})n} = \sum_{k=r+1}^{N+r} a_k e^{jk(\frac{2\pi}{N})n}$

这是因为 $e^{jk(\frac{2\pi}{N})(n+lN)} = e^{jk(\frac{2\pi}{N})n}\cdot 1$



这两个函数在两点处的值没有差别!

$=\sum_{n=1}^{N} 1 = N$

首先证明: $\sum_{n=1}^{N} e^{jk(2\pi/N)n} = \begin{cases} N, & k=0,\pm N,\pm 2N,\cdots \\ 0, & 其它. \end{cases}$

$=\sum_{n=1}^{N} e^{jk(2\pi/N)n} = e^{jk(\frac{2\pi}{N})}\frac{1-1}{1-e^{jk(2\pi/N)}}$

$=0.$

$\sum_{n=1}^{N} x[n] e^{jr(2\pi/N)n} = \sum_{n=1}^{N} \sum_{k=1}^{N} a_k e^{j(k-r)2\pi/N\cdot n} = \sum_{k=1}^{N} a_k \sum_{n=1}^{N} e^{j(k-r)\frac{2\pi n}{N}}$

$= a_r N$ $\qquad a_r = \frac{1}{N}\sum_{n=1}^{N} x[n] e^{-jr(2\pi/N)n}$

因此 $x[n] = \sum_{k=1}^{N} a_k e^{jk(2\pi/N)n}$

$$a_k = \frac{1}{N}\sum_{n=1}^{N} x[n] e^{-jk(2\pi/N)n}.$$

fft{x[n]} 会得到一个与x[n]长度相等的数组。f[k]

$$f[k] = N a_k = \sum_{n=1}^{N} x[n] e^{-jk(2\pi/N)n}$$

如果x[n]是一个实信号

$$a_{-k} = \frac{1}{N}\sum_{n=1}^{N} x[n] e^{jk(2\pi/N)n}$$

$$a_{+k} = \frac{1}{N}\sum_{n=1}^{N} x[n] e^{-jk(\frac{2\pi}{N})n} \Rightarrow a_{+k} = a_{-k}^{*}$$

事实上我们只需要考虑 f[k] 的 1~N/2

因为这个数组后半部分实际上是冗余的信息

实际上对于 $1 \le k \le (N/2+1)$

$\frac{f[k]}{N}$ 的物理意义是 $\frac{(k-1)}{N} \times f_s$ 的频率成分的强度.

※ 在我看来，这种方法是前人总结出来的 减少误差的一种方法

现在我们进入短时傅里叶变换

为什么傅里叶变换不行 非要用短时傅里叶变换？

原因是一个声音信号在不同的时刻变化非常的快

直接做傅里叶变换误差会很大。

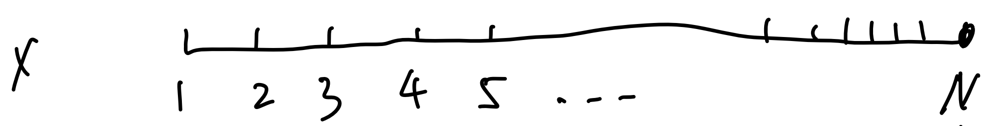
我们把这个信号进行切分。

对每一个部分分别做傅里叶变换

具体来说 是3步：

① 分帧　　② 加窗　　③ fft

分帧： 创建一个 ($2\lfloor \frac{N}{512} \rfloor + 1$)行，512列的二维数组 y



将x分成若干帧

我们对这个信号进行一下切分！每帧包含512个采样值，同时每帧之间有一半是重叠的，

具体来说：

第一帧：将x[1]~x[512]存在2维数组y的第一行中

第二帧：(一半重叠)将x[257]~x[769]

存在2维数组y的第二行中

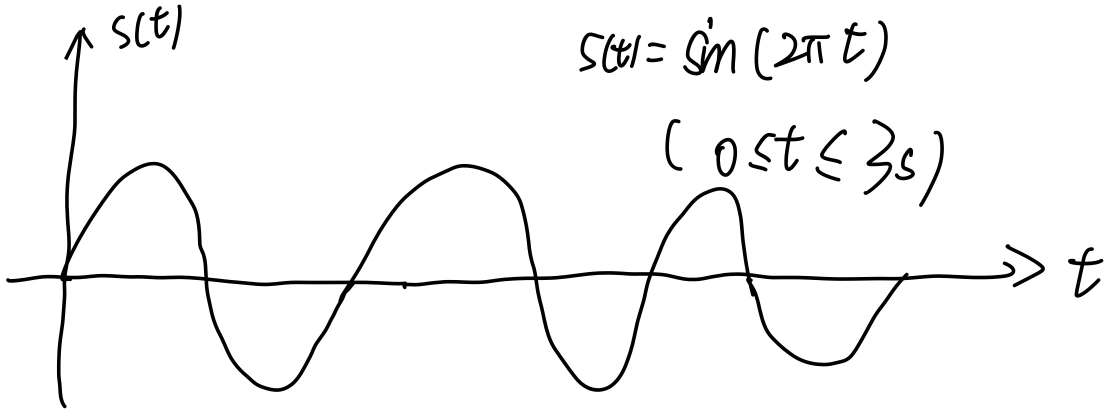
第三帧：x[513]~x[1024] 第三行。

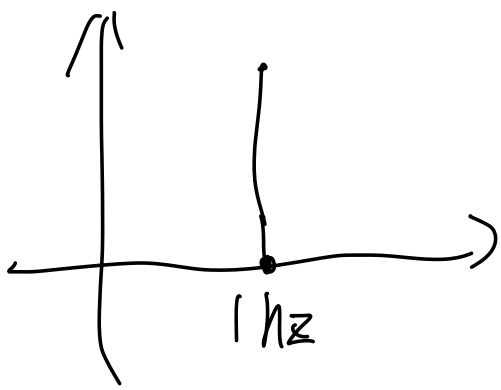
$$
y=\begin{bmatrix}
x[1] & x[2] & \cdots\cdots & x[512] \\
x[257] & x[258] & \cdots\cdots & x[768] \\
x[513] & x[514] & \cdots & x[1024] \\
x[769] & x[770] & \cdots & \\
\vdots & & &
\end{bmatrix}
$$

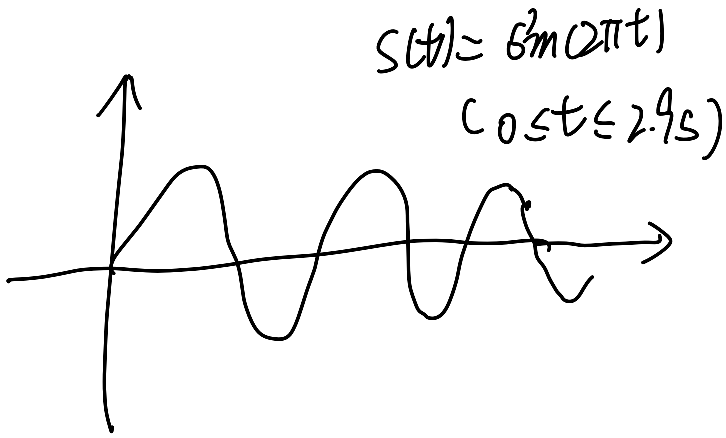
我们希望对每一帧单独进行傅里叶变换

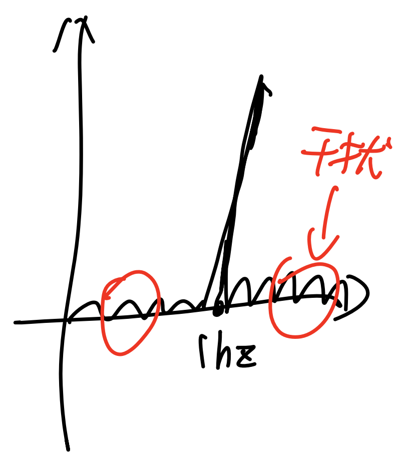
也就是对y的每一行单独进行傅里叶变换.

但是这里存在一个问题:









所以我们需要给 $s(t)\cdot w(t)$

减少我们切割信号对频谱造成的影响.

具体的实现方式 $w=\text{hann}(512)$

w是一个长度 512 的数组

$$y=\begin{bmatrix} x[1]w[1] & x[2]w[2] & \cdots & x[512]w[512] \\ x[257]w[1] & x[258]\cdot w[2] & \cdots & x[768]w[512] \\ x[513]w[1] & x[514]w[2] & \cdots & x[1024]w[512] \\ x[769]w[1] & x[770]w[2] & \cdots & \\ \vdots & & & \end{bmatrix}$$

③ 就是对 y 的每一行分别做傅里叶变换.

这个的实际含义就是把x[n]切分成了很多小段每一小段做一些实际处理后分别做傅里叶变换。

回到我们实际问题，我们对4个麦克风信号分别做STFT。

对于每个麦克风信号的每一帧你都会得到一个长度512的数组。

□ □ □ □ ⋯ □ ⋯ □ □

1 2 3 4 ⋯ k ⋯ 511 512

$$k \rightarrow \frac{(k-1)f_s}{512}$$ ← 对应的频率，

令 $f_0 = \frac{f_s}{512}$

回到music算法我们已经有了窄带信号的算法。

对于宽带我们只需要分解成窄带信号

Multiple Signal Classification algorithm (MUSIC algorithm)

Let $\boldsymbol{U}_n$ denote the $J\times(J-P)$ matrix containing the $J-P$ eigenvectors corresponding to all zero eigenvalues.

$$P_{music}(\theta)=\frac{1}{\sum_{i=1}^{J-P}\left|\underline{\mathbf{a}}^h(\theta)\underline{\boldsymbol{u}}_i\right|^2}=\frac{1}{\underline{\mathbf{a}}^h(\theta)\boldsymbol{U}_n\boldsymbol{U}_n^h\underline{\mathbf{a}}(\theta)}$$

The P largest peaks of $P_{music}(\theta)$ provide the source DOA, but the EVD of $\mathbf{A}\boldsymbol{R}_s\mathbf{A}^h$ is not available in practice .

$$\mathbf{A}\boldsymbol{R}_s\mathbf{A}^h\underline{\boldsymbol{u}}_i=\lambda\cdot\underline{\boldsymbol{u}}_i\Rightarrow\boldsymbol{R}_x\underline{\boldsymbol{u}}_i=(\mathbf{A}\boldsymbol{R}_s\mathbf{A}^h+\boldsymbol{R}_n)\underline{\boldsymbol{u}}_i=(\lambda+\sigma_n^2)\underline{\boldsymbol{u}}_i$$

So now $\underline{\boldsymbol{u}}_i$ is eigenvector of the smallest eigenvalues of $\boldsymbol{R}_x$ which means $\boldsymbol{U}_n$ contains the eigenvectors corresponding to the $J-P$ smallest eigenvalues of $\boldsymbol{R}_x$.

In practice, $\hat{\boldsymbol{R}}_x=\frac{1}{N}\sum_{k=1}^{N}\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^h[k]$.

对于每个 kf_0 $(k=0,1,2,\cdots,256)$

我们都可以算 $\hat{R}_x^{(kf_0)}$

第1个数据 →

这里的 $X[k]=$

第一帧 fft后 kf_0 第二帧 fft后 kf_0 第三帧 ...

2 → 第一帧 fft后 kf_0 第二帧 fft后 kf_0

3 →

4 →

得到R_x乘以下与窄带相同.