极小信噪比下的动态滤波研究

赵文亮　自 64
2016011452
(zhaowl16@mails.tsinghua.edu.cn)

2018 年 6 月 30 日

摘要

本文从噪声强度远远大于信号强度这一类工程背景出发，提出了极小信噪比下的动态滤波的概念，进而经过复杂的理论分析和严格的数学推导，得出了极小信噪比下的动态滤波器的实现原理和设计方法。最后，本文通过计算机模拟信号和噪声，并将推导出的动态滤波方法加以应用，取得了非常好的滤波效果。为了直观展示滤波的过程和效果，本文基于 Shiny 编写了可视化 App[1]。

关键词：极小信噪比；动态滤波；DTFT

1　引言

滤波是信号处理中的重要问题之一。在一些应用背景下，噪声的幅值远远大于信号的幅值，我们将这样的情况称为“极小信噪比”（Low Signal-Noise Ratio，LSNR）。极小信噪比下的静态滤波问题从频域解决较为容易，然而涉及到动态滤波时，传统的滤波方法会遇到困难。本文旨在提出一个能够将微小的信号从强大的噪声中有效提取出来的动态滤波方法。

本文首先通过理论分析，提出一个行之有效的滤波方法，再根据噪声的频域特征对算法进行改进和推广。最后，本文将通过计算机模拟来产生信号和噪声，并使用所提出的的算法进行滤波，将滤波所得信号与原始信号进行比对，从而验证算法的正确性和优越性。

2　理想滤波器的局限性

2.1　理想滤波器的设计

假设信号序列为 $x[t]$，滤波器为一个线性时不变系统 $\mathscr{L}$，其单位脉冲响应（Impulse Response Function, IRF）为 $h[t], t = 0, \pm1, \pm2, \cdots$ 。则输入信号 $x[t]$ 通过 $\mathscr{L}$ 的输出为：

$$y[t] = \sum_{k=-\infty}^{\infty} h[k]x[t-k] \tag{2.1}$$

使用离散时间傅里叶变换（Discrete-time Fourier Transform, DTFT）分析 $x[t]$ 和 $H[t]$ 的频谱，可以得

[1] 网址为https://john-williams.shinyapps.io/LSNR/，详见第 6 节

到两者在频域的表达式：

$$X(e^{\mathrm{j}\omega}) = \sum_{t=-\infty}^{\infty} x[t]e^{-\mathrm{j}t\omega} \tag{2.2}$$

$$H(e^{\mathrm{j}\omega}) = \sum_{t=-\infty}^{\infty} h[t]e^{-\mathrm{j}t\omega} \tag{2.3}$$

由 DTFT 频谱的周期性，本文所述的 ω 均取 $[-\pi, \pi]$ 的主值区间。由逆变换可得：

$$x[t] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{\mathrm{j}\omega})e^{\mathrm{j}t\omega}\,\mathrm{d}\omega \tag{2.4}$$

$$h[t] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{\mathrm{j}\omega})e^{\mathrm{j}t\omega}\,\mathrm{d}\omega \tag{2.5}$$

进一步可以得到：

$$y[t] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{\mathrm{j}\omega})H(e^{\mathrm{j}\omega})e^{jt\omega}\,\mathrm{d}\omega \tag{2.6}$$

由式 (2.6) 可以看出，如果信号和噪声在频域上无重叠，则可以通过设计 $\mathscr{L}$ 的频率响应（Frequency Response Function, FRF）来去除噪声。设信号对应的频率区间为 I，则令

$$H(e^{\mathrm{j}\omega}) = \begin{cases} 1, & \omega \in I \\ 0, & \omega \notin I \end{cases} \tag{2.7}$$

即可通过频域将噪声完全去除。例如，若信号的频率全部位于 $[0, \alpha)$ 中而噪声的频率全部位于 $(\alpha, \pi]$，称 α 为截止频率。则一种可能的 FRF 如图 1 所示。显然可知该滤波器可以完全将噪声滤除。

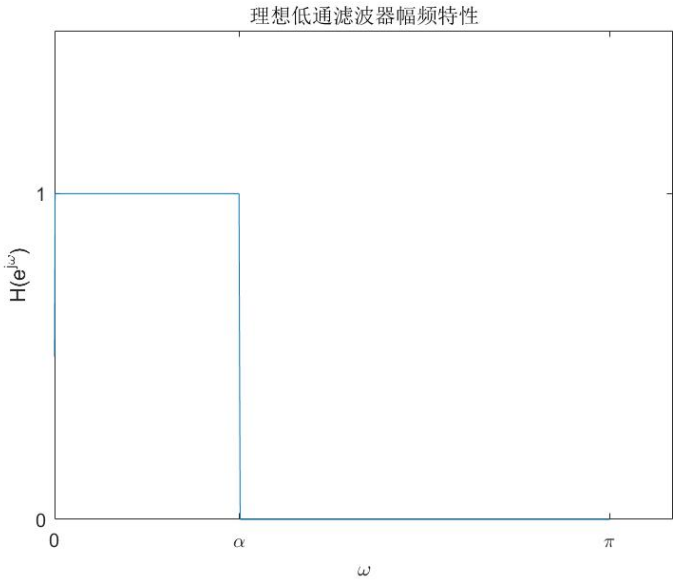


图 1: 理想滤波器幅频特性

2.2 理想滤波器的局限性

上述讨论的 FRF 在实际的滤波应用中由于实时性较差而存在局限性。在实际应用中，往往需要进行动态滤波，于是无法得到信号完整的频域表达，而只能在时域上进行滤波，对应的 IRF 只能是有限项，则 FRF 只

能是

$$H_N(e^{\mathrm{j}\omega}) = \sum_{k=-N}^{N} h[k]e^{-\mathrm{j}k\omega} \tag{2.8}$$

当 $N \to \infty$ 时，上式才与理想的 FRF 相同。所以，实际的滤波性能依赖于 N 的选取，N 越大滤波性能越好。但是如果 $h[k]$ 的收敛速度较慢，则一味增加 N 的取值并不一定有实质性的效果。

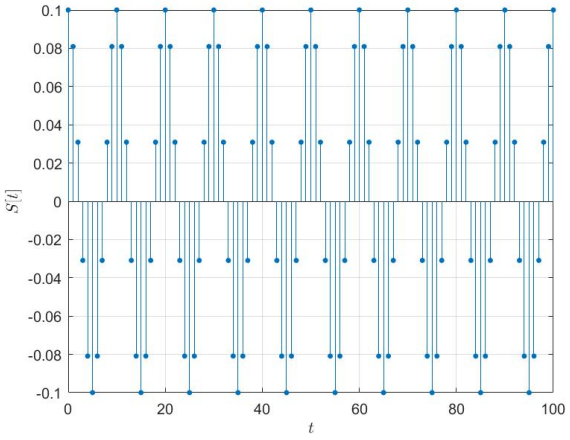
例如，假设信号为

$$S[t] = 0.1\cos\left(\frac{\pi}{5}t\right)$$

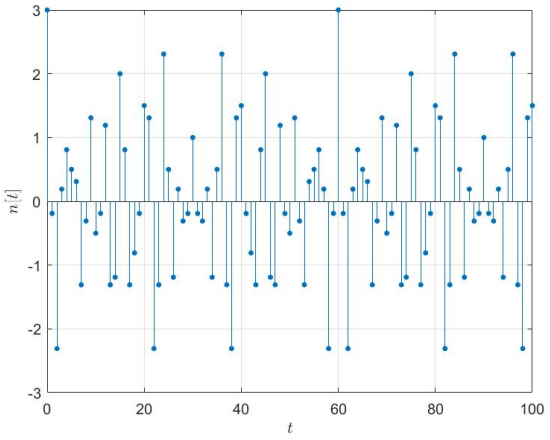
噪声为

$$n[t] = \cos\left(\frac{2\pi}{3}t\right) + \cos\left(\frac{\pi}{2}t\right) + \cos\left(\frac{2\pi}{5}t\right)$$

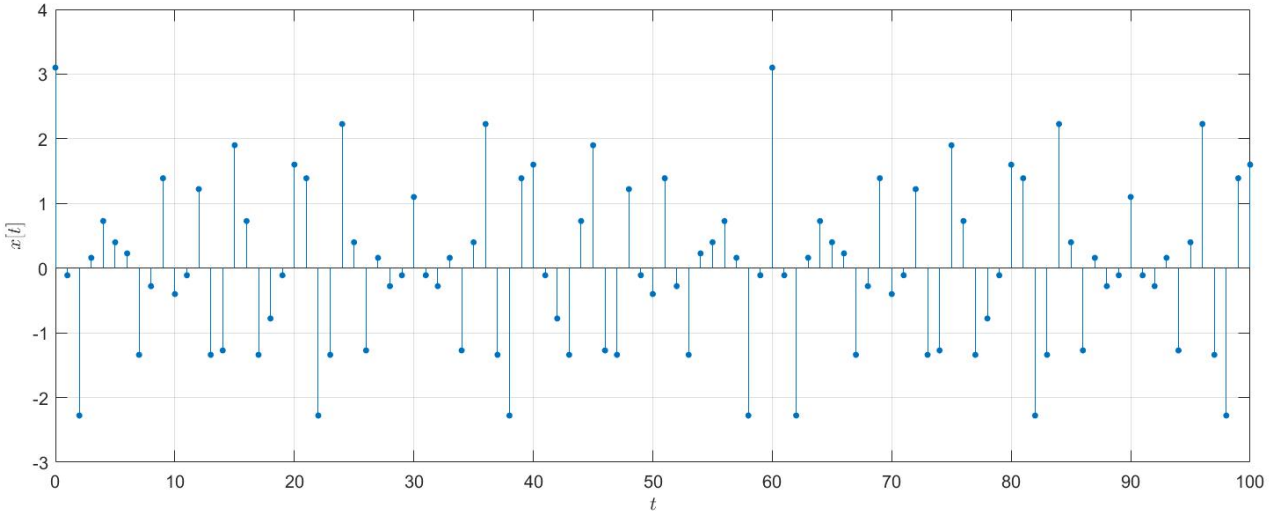
则滤波器的输入信号为二者之和，$x[t] = S[t] + n[t]$。信号、噪声以及系统输入的波形如图 2 所示。



(a) 信号 $S[t]$



(b) 噪声 $n[t]$



(c) 滤波器输入 $x[t]$

图 2: 信号、噪声与输入信号波形图

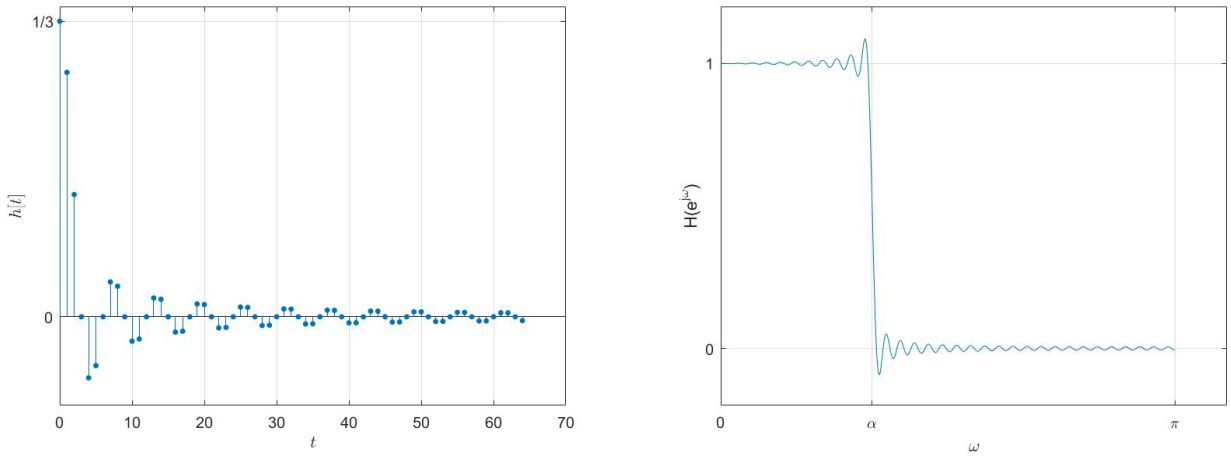
显然信号和噪声在频域上是分离的，取截止频率 $\alpha = \dfrac{\pi}{3}$，取 FRF

$$H(e^{\mathrm{j}\omega}) = \begin{cases} 1, & |\omega| \le \dfrac{\pi}{3} \\ 0, & \dfrac{\pi}{3} < |\omega| \le \pi \end{cases} \tag{2.9}$$

则对应的 IRF 为

$$h[t] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{\mathrm{j}\omega})e^{\mathrm{j}t\omega}\,\mathrm{d}\omega = \frac{1}{2\pi}\int_{-\frac{\pi}{3}}^{\frac{\pi}{3}} e^{\mathrm{j}t\omega}\,\mathrm{d}\omega = \frac{1}{\pi}\int_{0}^{\frac{\pi}{3}} \cos(\omega t)\,\mathrm{d}\omega = \frac{\sin\left(\frac{\pi}{3}t\right)}{\pi t} \quad (t \neq 0) \tag{2.10}$$

特别地，$h[0] = \dfrac{1}{\pi}\displaystyle\int_0^{\frac{\pi}{3}} \mathrm{d}\omega = \frac{1}{3}$。实际中 $h[t]$ 不能取到无穷多项，设 $-N \le t \le N$，画出截尾的 $h[t]$ 的波形如图 3a 所示，其中 $N = 64$。由于 $h[t]$ 具有对称性，此处仅仅画出了 $t \ge 0$ 的 $h[t]$ 值。



(a) IRF (b) FRF

图 3: 截尾的 IRF（$N = 64$）和对应的 FRF

根据截尾的 IRF，可以计算得到对应的 FRF

$$H_N(e^{\mathrm{j}\omega}) = \sum_{t=-64}^{64} h[t]e^{-\mathrm{j}t\omega} = h_0 + 2\sum_{t=1}^{64} \frac{\sin\left(\frac{\pi t}{3}\right)}{\pi t}\cos t\omega \tag{2.11}$$

如图 3b 所示。从中可见截尾的 IRF 在一定程度上可以实现最初的设想，但是对应的 FRF 与理想的 FRF 还是有一定的差距，尤其是在截止频率附近出现了过冲（Gibbs 现象）。将这组 IRF 应用于滤波中，得到 $y[t]$ 的图像，如图 4 所示，计算误差 $\Delta[t] = y[t] - S[t]$，如图 5 所示。

可见相比于输入，$y[t]$ 更加接近于原始信号。但是通过计算误差可知，滤波结果的误差的幅值约为 0.02，约为原始信号的 20%。这一现象当信噪比很小的时候更为明显，特别是噪声强度比信号强度大上万倍的情况。我们扩大噪声的强度到信号强度的 10000 倍，令

$$n[t] = 1000 \times \left(\cos\left(\frac{2\pi}{3}t\right) + \cos\left(\frac{\pi}{2}t\right) + \cos\left(\frac{2\pi}{5}t\right)\right)$$

同时增加滤波项数，令 $N = 512$，此时得到的误差如图 6 所示。可见误差的幅值已经远远大于信号的幅值。所以当对滤波精度要求较高时，这种方法并不能很好的实现目标。增加的取值虽然可以减少尾部泄漏，但是效果并不明显。所以针对这个问题，我们需要提出另一种行之有效的滤波方法。

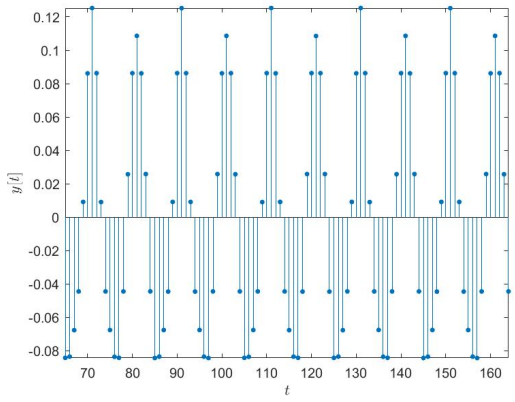


图 4: $N = 64$ 时滤波结果

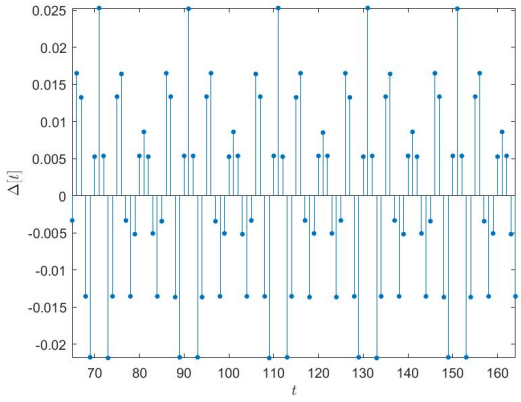


图 5: $N = 64$ 时滤波误差

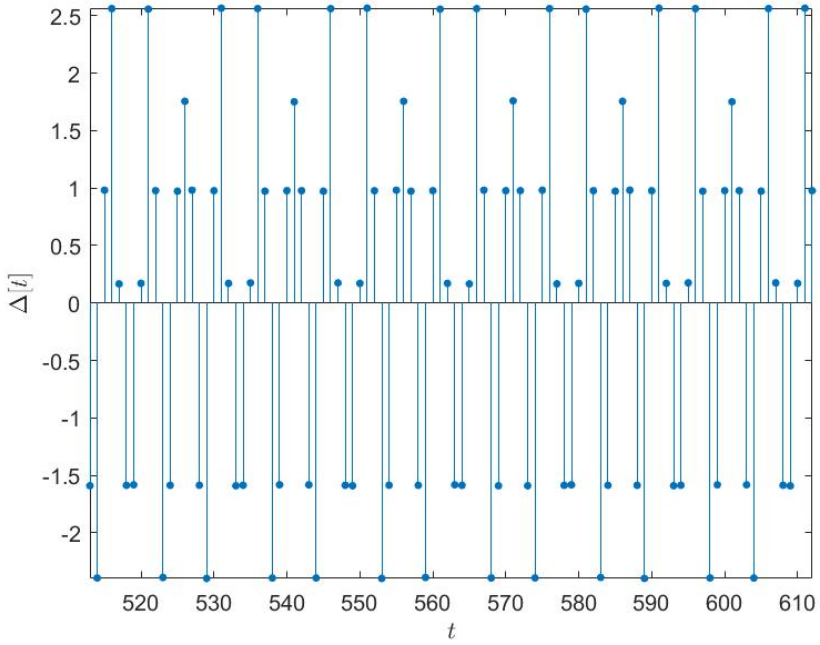


图 6: $N = 512$ 时滤波误差

3 LSNR 动态滤波器

3.1 滤波器限制条件

针对截断的理想滤波器的 IRF 在实际滤波中存在的问题，我们提出 LSNR 动态滤波器的设计。首先仍然假设信号与噪声在频域上分离，并设截止频率为 α ，即噪声集中在 $[\alpha, \pi]$ 上。LSNR 动态滤波器应该满足以下几个条件：

1. 滤波系数 $h[t], t = 0, \pm1, \pm2, \cdots \pm N$，其中 $h[t]$ 为实数且 $h[t] = h[-t]$。

2. 对应的 FRF 为

$$H(e^{\mathrm{j}\omega}) = \sum_{t=-N}^{N} h[t]e^{\mathrm{j}t\omega} = \sum_{t=-N}^{N} h[t](\cos(t\omega) + \mathrm{j}\sin(t\omega)) = \sum_{t=-N}^{N} h[t]\cos(t\omega) \tag{3.1}$$

且

$$H(0) = 1 \tag{3.2}$$

3. 对于误差 δ 和截止频率 $\alpha > 0$，我们的目标是最优的 FRF（Optimum FRF, OFRF），记做 $H^*(e^{\mathrm{j}\omega})$，满足

$$\max_{\alpha\le\omega\le\pi} |H^*(e^{\mathrm{j}\omega})| \le \inf_{\{h[t]\}} [\max_{\alpha\le\omega\le\pi} |H(e^{\mathrm{j}\omega})|] \le \delta \tag{3.3}$$

需要指出的是，若单纯考虑条件 3，则将 IRF 系数缩小一定的倍数后总可以满足条件，但这种情况将噪声滤掉的同时也将信号滤掉。故需要限制信号频域的 FRF，即增加 $H(0) = 1$ 的条件来保证信号得到保留。而 $H(0)$ 的具体取值不影响滤波效果，此处取 $H(0) = 1$ 是为了简化后续的推导过程。另外，我们不能要求 $\delta = 0$，否则会有

$$0 = \delta \ge \max_{\alpha\le\omega\le\pi} |H^*(e^{\mathrm{j}\omega})| \ge |H^*(e^{\mathrm{j}\omega})|_{\alpha\le\omega\le\pi} \ge 0$$

则 $H^*(e^{\mathrm{j}\omega})$ 在 $\alpha \le \omega \le \pi$ 恒为零，注意到 $H^*(e^{\mathrm{j}\omega})$ 为 $\cos\omega$ 的多项式，在无穷个点处为零则可以推出 $H^*(e^{\mathrm{j}\omega}) \equiv 0$。即滤掉噪声的同时也滤掉了信号。

3.2 滤波器系数求解

解决问题的关键在确定 $H(0) = 1$ 的条件下考虑 $\max_{\alpha\le\omega\le\pi} |H^*(e^{\mathrm{j}\omega})|$ 的最小值，直接考虑这个问题并不容易，所以我们采用一个等价的方法，即先假定 $\max_{\alpha\le\omega\le\pi} |H^*(e^{\mathrm{j}\omega})|$ 固定，不妨设 $\max_{\alpha\le\omega\le\pi} |H^*(e^{\mathrm{j}\omega})| = 1$，考虑 $H(0)$ 的最大值。为此，令

$$\mathscr{H} = \{H(e^{\mathrm{j}\omega}) | H(e^{\mathrm{j}\omega}) = \sum_{t=-N}^{N} h[t]\cos t\omega, h[t] = h[-t], h[t] \in \mathbb{R}, |H(e^{\mathrm{j}\omega})| \le 1, \forall\omega \in [\alpha, \pi]\} \tag{3.4}$$

下面证明一个定理。

定理 1. *若 $\hat{H}(e^{\mathrm{j}\omega}) \in \mathscr{H}$ 满足 $|\hat{H}(0)| = C \ge |H(0)|, \forall H \in \mathscr{H}$，则只需做如下调整：令 $\tilde{h}[t] = \frac{1}{C}\hat{h}[t]$，则 $\{\tilde{h}[t]\}$ 是在 $\delta = \frac{1}{C}$ 下的最优 IRF（Optimum IRF, OIRF）。*

证明 1. 令

$$\tilde{H}(e^{\mathrm{j}\omega}) = \sum_{t=-N}^{N} \tilde{h}[t]\cos t\omega, \quad 0 \ge \omega \ge \pi$$

则

$$|\tilde{H}(0)| = |\sum_{t=-N}^{N} \frac{\hat{h}[t]}{C}| = \frac{1}{C}|\hat{H}(0)| = 1$$

且

$$\max_{\alpha\le\omega\le\pi} |\tilde{H}(e^{\mathrm{j}\omega})| = \frac{1}{C}\max_{\alpha\le\omega\le\pi} |\hat{H}(e^{\mathrm{j}\omega})| \le \frac{1}{C}$$

$\forall H_1$ 满足条件 *1* 和 *2*，不妨设

$$\max_{\alpha\le\omega\le\pi} |H_1(e^{\mathrm{j}\omega})| = M$$

令 $H_2(e^{\mathrm{j}\omega})=\dfrac{1}{M}H_1(e^{\mathrm{j}\omega})$，则有 $\max\limits_{\alpha\le\omega\le\pi}|H_2(e^{\mathrm{j}\omega})|=1$，即 $H_2\in\mathscr{H}$。于是 $1=|H_1(0)|=M|H_2(0)|\le MC$，即 $\max\limits_{\alpha\le\omega\le\pi}|H_1(e^{\mathrm{j}\omega})|=M\ge\dfrac{1}{C}$。而 $\tilde{H}$ 也满足条件 *1* 和 *2*, 故 $\max\limits_{\alpha\le\omega\le\pi}|\tilde{H}(e^{\mathrm{j}\omega})|\ge\dfrac{1}{C}$。另一方面，$\max\limits_{\alpha\le\omega\le\pi}|\tilde{H}(e^{\mathrm{j}\omega})|\le\dfrac{1}{C}$，故 $\max\limits_{\alpha\le\omega\le\pi}|\tilde{H}(e^{\mathrm{j}\omega})|=\dfrac{1}{C}$。则 $\tilde{H}$ 为 $\delta=\dfrac{1}{C}$ 下的 *OFRF*，即 $\{\tilde{h}[t]\}$ 是 $\delta=\dfrac{1}{C}$ 下的 *OIRF*。 □

有了定理 1，求解 OFRF 的问题转化成了求一个 $\hat{H}\in\mathscr{H}$ 使得 $|\hat{H}(0)|$ 最大。由 H 的定义可知 $\forall H\in\mathscr{H}$，H 是关于 $\cos t\omega, t=0,\pm1,\pm2,\cdots,\pm N$ 的函数。由于 $\cos t\omega$ 可以展开成 $\cos\omega$ 的 $|t|$ 阶多项式，H 也是 $\cos\omega$ 的多项式。在此基础上，可以建立 H 与 N 阶多项式 P 的一一对应。定义

$$\begin{aligned}\mathscr{P}&=\{P_N(\cos\omega)|\text{关于 }\cos\omega\text{ 的 }N\text{ 阶实系数多项式，且 }|P_N(\cos\omega)|\le1,0<\alpha\le\omega\le\pi\}\\&=\{P_N(x)|\text{关于 }x\text{ 的 }N\text{ 阶实系数多项式，且 }|P_N(x)|\le1,-1\le x\le a=\cos\alpha\}\end{aligned}\tag{3.5}$$

至此，问题转化为寻求 $\hat{P}_N(x)$ 使得 $|\hat{P}_N(1)|\ge|P_N(1)|,\forall P_N(x)\in\mathscr{P}$。定理 2 构造出了满足条件的最优解。

定理 2. 对 $0<\alpha<\pi$，正偶数 N，令 $\hat{P}_N(x)=\cos\left(N\cos^{-1}\dfrac{2x-a+1}{a+1}\right)$，则 $\hat{P}_N(x)$ 满足 $|\hat{P}_N(x)\le1|,-1\le x\le a=\cos\alpha$ 和 $|P_N(1)|\ge|P_N(1)|,\forall P_N(x)\in\mathscr{P}$。

证明 2. 由切比雪夫多项式 $T_N(y)=\cos(N\cos^{-1}y)$ 可知，$\hat{P}_N(x)$ 是关于 x 的多项式，且显然有 $|\hat{P}_N(x)|\le1,-1\le x\le a=\cos\alpha$。下证 $|P_N(1)|\ge|P_N(1)|,\forall P_N(x)\in\mathscr{P}$。

考虑到 $[-1,a]$ 的区间宽度为 $a+1$，故为了代入切比雪夫多项式需进行变换 $y=\dfrac{2x-a+1}{a+1}$，这样就有 $-1\le y\le1$。从这个思路出发，令

$$x_m=\frac{1}{2}\left[(a+1)\cos\frac{m\pi}{N}+a-1\right],m=0,1,2,\cdots,N$$

或

$$\cos\frac{m\pi}{N}=\frac{2x_m-a+1}{a+1}$$

则 $-1\le x_m\le a,m=0,1,2,\cdots,N$，且 $m<n$ 时，$x_m>x_n$。由 *Lagrange* 插值多项式可知：

$$P_N(x)=\sum_{m=0}^{N}P_N(x_m)\frac{\prod_{n\ne m}(x-x_n)}{\prod_{n\ne m}(x_m-x_n)}$$

故

$$|P_N(1)|=\left|\sum_{m=0}^{N}P_N(x_m)\frac{\prod_{n\ne m}(1-x_n)}{\prod_{n\ne m}(x_m-x_n)}\right|\le\sum_{m=0}^{N}|P_N(x_m)|\left|\frac{\prod_{n\ne m}(1-x_n)}{\prod_{n\ne m}(x_m-x_n)}\right|$$

易知 $\prod_{n\ne m}(1-x_n)\ge0$，而

$$\prod_{n\ne m}(x_m-x_n)=\prod_{n<m}(x_m-x_n)\prod_{n>m}(x_m-x_n)$$

由 x_m 单调性可知，第二项非负，第一项为负。故

$$\left|\prod_{n\ne m}(x_m-x_n)\right|=(-1)^m\prod_{n\ne m}(x_m-x_n)$$

则

$$|P_N(1)|\le\sum_{m=0}^{N}|P_N(x_m)|\left|\frac{\prod_{n\ne m}(1-x_n)}{\prod_{n\ne m}(x_m-x_n)}\right|=\sum_{m=0}^{N}|P_N(x_m)|(-1)^m\frac{\prod_{n\ne m}(1-x_n)}{\prod_{n\ne m}(x_m-x_n)}$$

又因为 $|P_N|(x)\le1,-1\le x\le a=\cos\alpha,-1\le x_m\le a,m=0,1,2,\cdots,N$，有

$$|P_N(1)| \le \sum_{m=0}^{N}(-1)^m \frac{\prod_{n\neq m}(1-x_n)}{\prod_{n\neq m}(x_m-x_n)}$$

注意到 $|\hat{P}_N(x_m)| = \cos(N\cos^{-1}(\cos\frac{m\pi}{N})) = (-1)^m$ 和 $\hat{P}_N(1) = \sum_{m=0}^{N}\hat{P}_N(x_m)\frac{\prod_{n\neq m}(1-x_n)}{\prod_{n\neq m}(x_m-x_n)}$，可得

$$|P_N(1)| \le \sum_{m=0}^{N}(-1)^m \frac{\prod_{n\neq m}(1-x_n)}{\prod_{n\neq m}(x_m-x_n)} = \sum_{m=0}^{N}\hat{P}_N(x_m)\frac{\prod_{n\neq m}(1-x_n)}{\prod_{n\neq m}(x_m-x_n)} = \hat{P}_N(1)$$

□

根据以上的结论有

$$\hat{H}(e^{\mathrm{j}\omega}) = \cos\left(N\cos^{-1}\frac{2\cos\omega-\cos\alpha+1}{\cos\alpha+1}\right), \quad \alpha\le\omega\le\pi \tag{3.6}$$

且 $|\hat{H}(e^{\mathrm{j}\omega})|\le 1$，$\hat{H}(0)$ 具有最大值。然而上述表达式只在 $\alpha\le\omega\le\pi$ 上有定义，对于 $0\le\omega<\alpha$，由欧拉公式可得

$$\begin{aligned}\cos(N\cos^{-1}y) &= \frac{e^{\mathrm{j}N\cos^{-1}y}+e^{-\mathrm{j}N\cos^{-1}y}}{2} = \frac{(e^{\mathrm{j}\cos^{-1}y})^N+(e^{-\mathrm{j}\cos^{-1}y})^N}{2}\\ &= \frac{1}{2}\left[(y+\sqrt{y^2-1})^N+(y-\sqrt{y^2-1})^N\right]\end{aligned} \tag{3.7}$$

故

$$\hat{P}_N(x) = \frac{1}{2}\left\{\left(\frac{2x-a+1}{a+1}+\sqrt{\left(\frac{2x-a+1}{a+1}\right)^2-1}\right)^N+\left(\frac{2x-a+1}{a+1}-\sqrt{\left(\frac{2x-a+1}{a+1}\right)^2-1}\right)^N\right\} \tag{3.8}$$

且

$$\hat{P}_N(1) = \frac{1}{2}\left\{\left(\frac{3-a}{a+1}+\sqrt{\left(\frac{3-a}{a+1}\right)^2-1}\right)^N+\left(\frac{3-a}{a+1}-\sqrt{\left(\frac{3-a}{a+1}\right)^2-1}\right)^N\right\} \tag{3.9}$$

设 $\delta = \frac{1}{\hat{P}_N(1)}$，则 OFRF 为

$$H^*(e^{\mathrm{j}\omega}) = \begin{cases}\frac{\delta}{2}\left\{\left(\Phi(\omega)+\sqrt{\Phi^2(\omega)-1}\right)^N+\left(\Phi(\omega)-\sqrt{\Phi^2(\omega)-1}\right)^N\right\}, & 0\le\omega<\alpha\\ \delta\cos(N\cos^{-1}\Phi(\omega)), & \alpha\le\omega\le\pi\end{cases} \tag{3.10}$$

4 LSNR 动态滤波器的应用

4.1 应用方法

在第 3.2 节中已经得到了 OFRF 的解，在实际应用过程中的一个很重要的问题是如何选取滤波项数 N。注意到在式 (3.9) 中，有

$$\left(\frac{3-a}{a+1}+\sqrt{\left(\frac{3-a}{a+1}\right)^2-1}\right)\left(\frac{3-a}{a+1}-\sqrt{\left(\frac{3-a}{a+1}\right)^2-1}\right) = 1$$

而且第一项大于 1，所以第二项小于 1. 当 N 比较大的时候，有

$$\hat{P}_N(1) \approx \frac{1}{2}\left(\frac{3-a}{a+1}+\sqrt{\left(\frac{3-a}{a+1}\right)^2-1}\right)^N \tag{4.1}$$

对于给定的 δ，只需求解

$$\frac{1}{2}\left(\frac{3-a}{a+1}+\sqrt{\left(\frac{3-a}{a+1}\right)^2-1}\right)^N \geq \frac{1}{\delta} \tag{4.2}$$

即可得到 N 的估计值。由于滤波在时域上进行，而且滤波项数有限，故这种滤波方法可以实现动态滤波，延迟至多为 N。而由式 (4.1) 可知，OFRF 很快就会收敛到满足误差要求，所以 N 一般不会很大。

4.2 应用实例

下面将得到了滤波方法应用于第 2.2 中的信号和噪声：

$$\begin{aligned}
S[t] &= 0.1\cos\left(\frac{\pi}{5}\right)\\
n[t] &= 1000\times\left(\cos\left(\frac{2\pi}{3}t\right)+\cos\left(\frac{\pi}{2}t\right)+\cos\left(\frac{2\pi}{5}t\right)\right)\\
x[t] &= S[t]+n[t]
\end{aligned}$$

首先确定滤波项数，仍取截止频率 $\alpha=\frac{\pi}{3}$ ，则 $a=\cos\alpha=\frac{1}{2}$，则由式 4.1 可得

$$\hat{P}_N(1) \approx \frac{1}{2}\left(\frac{5}{3}+\frac{4}{3}\right)^N = \frac{3^N}{2}$$

取 $\delta=10^{-7}$，由 $\frac{3^N}{2}\geq\frac{1}{\delta}$ 解得 $N\geq\frac{8\ln 2+7\ln 5}{\ln 3}\approx 15.3$，故取 $N=16$，则有

$$H^*(e^{\mathrm{j}\omega}) = \begin{cases} \dfrac{10^{-7}}{2}\left\{\left(\Phi(\omega)+\sqrt{\Phi^2(\omega)-1}\right)^{16}+\left(\Phi(\omega)-\sqrt{\Phi^2(\omega)-1}\right)^{16}\right\}, & 0\leq\omega<\dfrac{\pi}{3}\\ 10^{-7}\times\cos(16\cos^{-1}\Phi(\omega)), & \dfrac{\pi}{3}\leq\omega\leq\pi \end{cases} \tag{4.3}$$

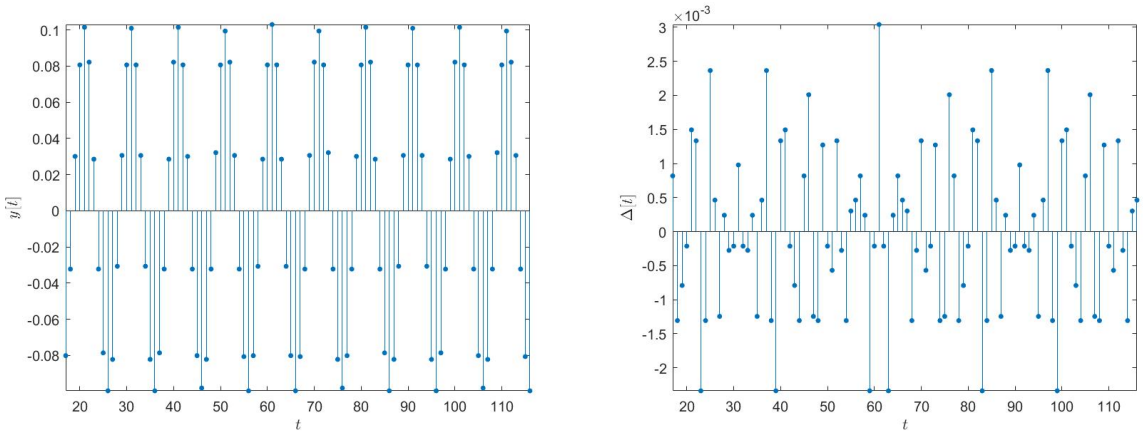
其中 $\Phi(\omega)=\frac{4\cos\omega+1}{3}$。进而可以通过式 (2.5) 求得 $h[t]$，再通过式 (2.1) 即可计算出滤波结果 $y[t]$。由于信号的频率为 $\frac{\pi}{5}$，滤波结束后我们需要将结果除以 $H^*(\frac{\pi}{5})$ 进行补偿。

得到 $y[t]$ 如图 7a 所示，滤波误差如图 7b 所示。可见误差的量级已经达到了 10^{-3}。

进一步地，我们取 $\delta=10^{-13}$，得到 $N=30$，得到滤波结果如图 8 所示。从中可见误差的数量级仅为 10^{-8}，远远小于信号幅值的数量级，滤波效果非常好。

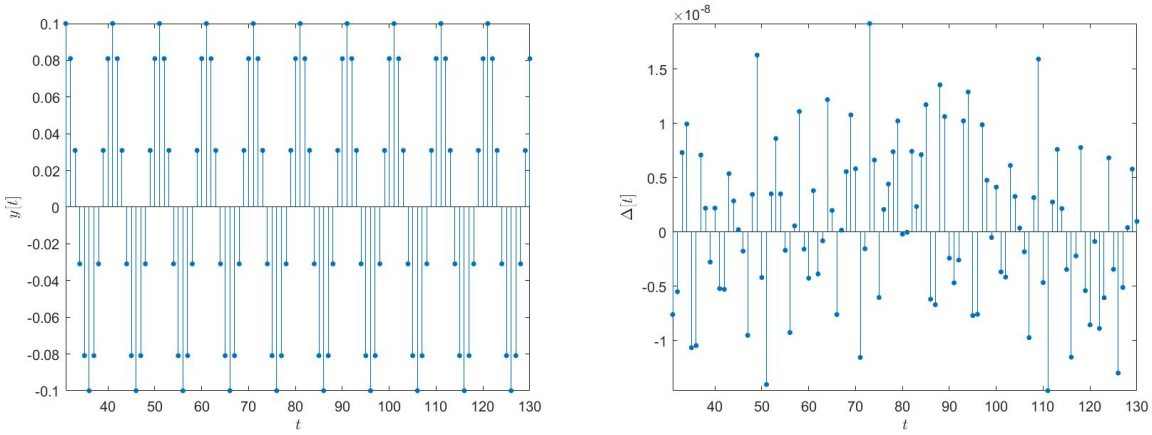
5 LSNR 动态滤波器的推广

上文所述的 LSNR 动态滤波器假定信号和噪声的频域完全分离且独立存在于两个区间。而事实上，信号和噪声的频域交错排列的情况也会出现。下面我们指出，对于这种情况，LSNR 动态滤波方法稍加推广后仍然适用。



(a) $y[t]$ (b) $\Delta[t]$

图 7: $N = 16, \delta = 10^{-7}$ 滤波结果



(a) $y[t]$ (b) $\Delta[t]$

图 8: $N = 30, \delta = 10^{-13}$ 滤波结果

5.1 噪声—信号—噪声类型的推广

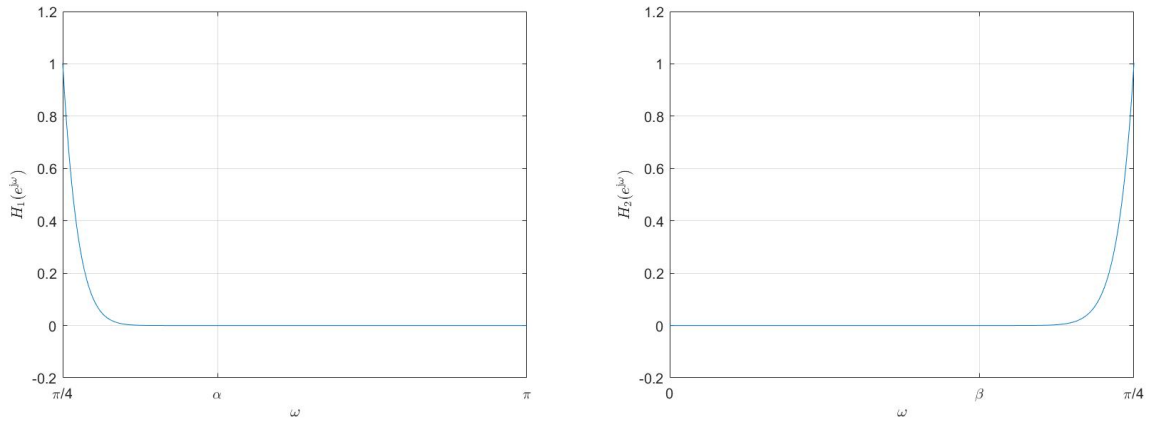
现假定在频域上存在噪声—信号—噪声的交替排列，且信号的频域集中在一个区间。例如：

$$\begin{aligned} S[t] &= 0.1\cos\left(\frac{\pi}{4}t\right) \\ n[t] &= 1000 \times \left(\cos\left(\frac{\pi}{12}t\right) + \cos\left(\frac{2\pi}{3}t\right)\right) \end{aligned} \tag{5.1}$$

则可以令截止频率 $\alpha = \dfrac{\pi}{2}, \beta = \dfrac{\pi}{6}$。这样，信号的频域集中在 β, α 上，而噪声的频域分别位于 $(0, \beta)$ 和 (α, π) 上。首先，我们把信号和位于 $(0, \beta)$ 上的噪声看成整体，即使用 LSNR 动态滤波的方法将 (α, π) 上的噪声滤掉。这一过程和前文所述方法完全一样，记 OFRF 为 H_1，令 $\delta = 10^{-13}$，取 $N = 20$，作出 OFRF 图像如图 9a 所示。将相应的 IRF 应用于滤波即可滤掉 (α, π) 上的噪声。

接下来我们要滤掉 $(0, \beta)$ 上的噪声，为此，令

$$\alpha_1 = 2 \times \frac{\pi}{4} - \beta$$

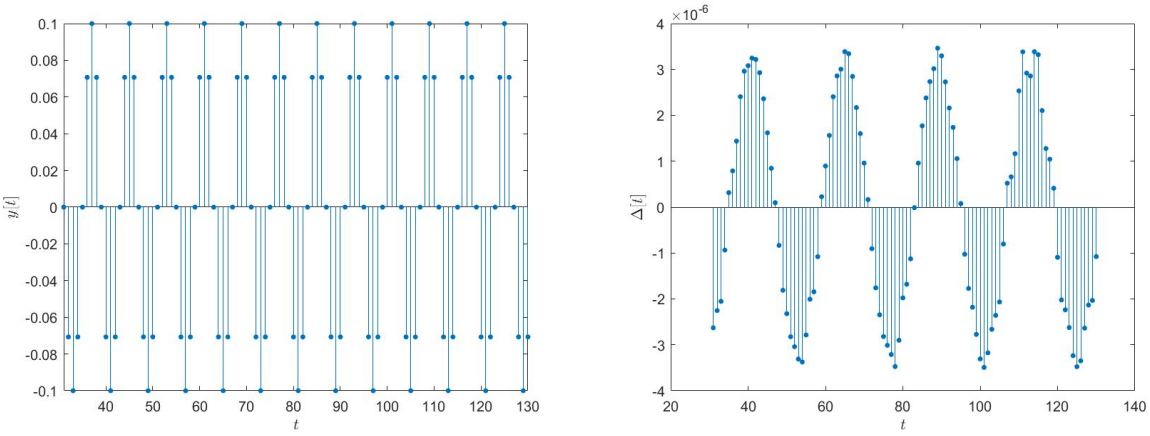


(a) $H_1(e^{\mathrm{j}\omega})$ (b) $H_2(e^{\mathrm{j}\omega})$

图 9: 噪声—信号—噪声推广下的 OFRF

即将 $(0,\beta)$ 关于 $\omega=\dfrac{\pi}{4}$ 取对称，那么对称后的噪声位于 $(\alpha_1, 2\times\dfrac{\pi}{4}-0)\subset(\alpha_1,\pi)$。于是以 α_1 为截止频率应用 LSNR 动态滤波，令 $\delta=10^{-13}$，取 $N=30$，得到 OFRF 记做 $\bar{H}_2$。再将 $\bar{H}_2$ 关于 $\omega=\dfrac{\pi}{4}$ 对称回来得到 H_2，如图 9b 所示。

我们只要先后使用 H_1 和 H_2 对应的 OIRF $h_1[t]$ 和 $h_2[t]$ 进行滤波，就可以将两个区间的噪声滤掉，结果如图 10 所示。可见误差仅为 10^{-6} 量级。



(a) $y[t]$ (b) $\Delta[t]$

图 10: 噪声—信号—噪声推广下的滤波结果

5.2 多信号多噪声的推广

现假定信号和噪声都处于多个频域上，不妨设截止频率

$$0\le\alpha_1<\alpha_2<\cdots<\alpha_n\le\pi \tag{5.2}$$

下面给出这种情况下的滤波方法：

1. 任取一个信号作为目标信号，设该信号位于区间 (α_k,α_{k+1}) 上。

2. 若 $\alpha_k = 0$，则直接应用第 3.2 节的方法即可将该信号提取；若 $\alpha_k \neq 0$，则将 $[0, \alpha_k)$ 与 $(\alpha_{k+1}, \pi]$ 上所有的输入视为噪声，利用第 5.1 节的推广方法，即可将该信号提取出来。

3. 对于每一个频段的待提取信号重复上述过程。

4. 将每次提取出的信号叠加即可。

6 LSNR 动态滤波的交互 App

为了将滤波的过程更好地呈现出来，我使用 R 语言基于 Shiny 编写了可交互的 App[2]，用户可以改变信号和噪声的频率、截止频率、信噪比、预期误差、滤波项数等参数，并可以查看输入波形、滤波器参数、输出波形及实际误差等。App 的界面截图如图 11 所示。

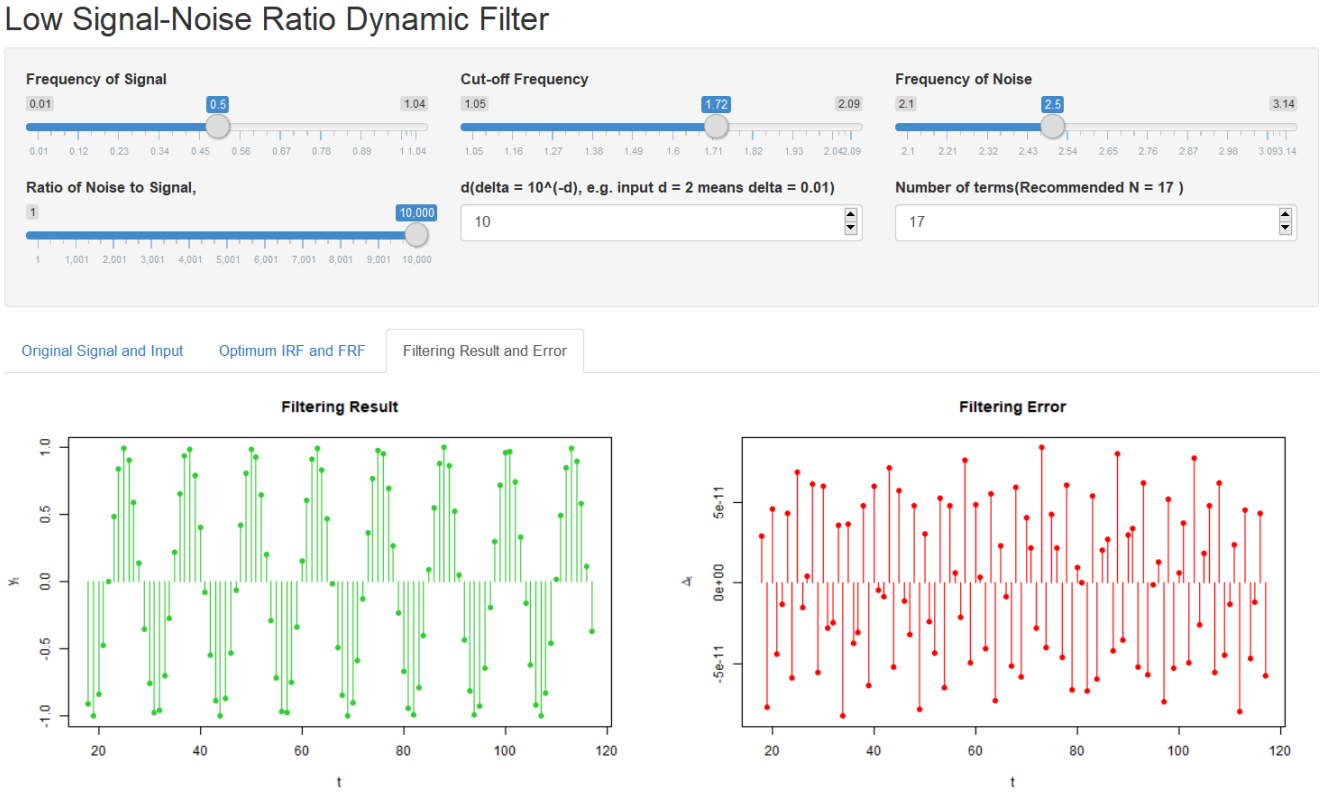


图 11: LSNR 动态滤波可视化 App 界面

7 结语

本文对极小信噪比下的动态滤波问题进行研究，得到了最优的滤波系数的求解方法，并对信号与噪声分布在频域多个频带的情形做出了推广。本文使用计算机对算法进行了充分的验证，得到了非常好的滤波效果，在实际应用中采用这种滤波器可以提升效果并节省时间。但是，所有的推导建立在信号和噪声的频率范围已知的条件下，如何从随机频率的强大噪声中提取有效信号仍然是一个十分困难的问题。

[2]点击https://john-williams.shinyapps.io/LSNR/即可访问

参考文献

[1] 谢衷洁, 滤波及其应用. 湖南教育出版社, 1998.

[2] 卓晴, 信号与系统. 2018.

[3] “Shiny - tutorial.” http://shiny.rstudio.com/tutorial/.

A 索引

插图

1 理想滤波器幅频特性 . . . 2
2 信号、噪声与输入信号波形图 . . . 3
3 截尾的 IRF（$N = 64$）和对应的 FRF . . . 4
4 $N = 64$ 时滤波结果 . . . 5
5 $N = 64$ 时滤波误差 . . . 5
6 $N = 512$ 时滤波误差 . . . 5
7 $N = 16, \delta = 10^{-7}$ 滤波结果 . . . 10
8 $N = 30, \delta = 10^{-13}$ 滤波结果 . . . 10
9 噪声—信号—噪声推广下的 OFRF . . . 11
10 噪声—信号—噪声推广下的滤波结果 . . . 11
11 LSNR 动态滤波可视化 App 界面 . . . 12

B 代码

B.1 MATLAB 仿真代码

B.1.1 主程序

```
%% global variables
global int
global lat
int = 'interpreter';
lat = 'latex';
R = 10000;% Ratio of noise to signal
n = 512;% number of terms of filter
H = @H1;% function to calculate the optimum frf
t = 0 : 20000;% time
%% ideal H
figure(1)
N = 1000;
alpha = pi / 3;
omega = linspace(0, pi, N);
H_ideal = heaviside(omega) - heaviside(omega - alpha);
plot(omega, H_ideal);
set(gca,'XTick',[0,pi/3,pi]);xticklabels={'0','\alpha','\pi'};
set(gca,'xticklabel',xticklabels);
set(gca,'yTick',[0, 1]);yticklabels={'0', '1'};
ylim([0 1.5]);
set(gca,'yticklabel',yticklabels);
xlabel('\omega');
ylabel('H(e^{j\omega})')
title('          ¨          ');
saveas(gcf, 'ideal_H.pdf')
%% signal
figure(2)
S_t = 0.1 *cos(2 * pi * t / 10);
% subplot(2, 2, 1);
stem(t, S_t, 'filled', 'MarkerSize', 3);
xlim([0, 100]);
xlabel('$t$', int, lat);
ylabel('$S[t]$', int, lat);
grid on;
figure()
n_t = R * 0.1 * (cos(2*pi*t/3) + cos(2*pi*t/4) + cos(2*pi*t/5));
% subplot(2, 2, 2);
stem(t, n_t, 'filled', 'MarkerSize', 3);
xlabel('$t$', int, lat);
ylabel('$n[t]$', int, lat);
xlim([0, 100]);
x_t = S_t + n_t;
grid on;
figure()
% subplot(2, 1, 2);
stem(t, x_t, 'filled', 'MarkerSize', 3);
xlabel('$t$', int, lat);
ylabel('$x[t]$', int, lat);
xlim([0, 100]);
grid on;
```

```
%% ht
k = -n : n;
h_t = IRFh(k);
grid on;
stem(k(n+1 : end), h_t(n+1 : end), 'filled', 'MarkerSize', 3);
xlabel('$t$', int, lat);
ylabel('$h[t]$', int, lat);
set(gca,'yTick',[0,1/3]);yticklabels={'0','1/3'};
set(gca,'yticklabel',yticklabels);
grid on;
%% H when n = 64
H_0 = H0(omega, 0);
plot(omega, H_0);
set(gca,'XTick',[0,pi/3,pi]);xticklabels={'0','\alpha','\pi'};
set(gca,'xticklabel',xticklabels);
set(gca,'yTick',[0, 1]);yticklabels={'0', '1'};
set(gca,'yticklabel',yticklabels);
xlabel('\omega');
ylabel('H(e^{j\omega})')
grid on;

%% using ideal filter
figure(2)
% subplot(3, 1, 1);
[t_plot, y_t] = filter1(x_t, h_t);
stem_plot(t_plot, y_t(t_plot));
axis tight;
xlabel('$t$', int, lat)
ylabel('$y[t]$', int, lat)
figure()
% subplot(3, 1, 2);
stem_plot(t_plot, S_t(t_plot));
axis tight;
figure()
% subplot(3, 1, 3)
delta_t = y_t - S_t;
stem_plot(t_plot, delta_t(t_plot));
xlabel('$t$', int, lat)
ylabel('$\Delta[t]$', int, lat)
axis tight;
%% using the optimized irf
% h_t = oirf(H, 16, 7, pi / 3, pi / 5, 1);
h_t = oirf(H, 30, 13, pi / 3, pi / 5, 1);
figure(3)
% subplot(3, 1, 1);
[t_plot, y_t] = filter1(x_t, h_t);
stem_plot(t_plot, y_t(t_plot));
xlabel('$t$', int, lat)
ylabel('$y[t]$', int, lat)
axis tight;
figure()
% subplot(3, 1, 2);
stem_plot(t_plot, S_t(t_plot));
axis tight;
figure()
% subplot(3, 1, 3)
delta_t = y_t - S_t;
stem_plot(t_plot, delta_t(t_plot));
```

```
xlabel('$t$', int, lat)
ylabel('$\Delta[t]$', int, lat)
axis tight;

%% multiple noise case
S_t = 0.1 * cos(2 * pi * t / 8);
n_t = R * 0.1 * (1 * cos(2 * pi * t/ 24) + 1 * cos(2 * pi * t/ 3));
x_t = S_t + n_t;

h1_t = oirf(H, 20, 13, pi  / 2, pi / 4, 1);
[t_1, y1_t] = filter1(x_t, h1_t);
S1_t = S_t(t_1);
figure(4)

subplot(3, 1, 1);
stem_plot(t_1, y1_t(t_1));
axis tight;
subplot(3, 1, 2);
stem_plot(t_1, S_t(t_1));
axis tight;
subplot(3, 1, 3)
delta_t = y1_t - S_t;
stem_plot(t_1, delta_t(t_1));
axis tight;


figure()
h2_t = oirf(H, 30, 13, pi / 6, pi / 4, 2);
pause
[t_2, y_t] = filter1(y1_t(t_1), h2_t);
% subplot(3, 1, 1);
stem_plot(t_2, y_t(t_2));
axis tight;
xlabel('$t$', int, lat)
ylabel('$y[t]$', int, lat)
% subplot(3, 1, 2);
figure()
stem_plot(t_2, S1_t(t_2));
axis tight;
% subplot(3, 1, 3)
figure()
delta_t = y_t - S1_t;
stem_plot(t_2, delta_t(t_2));
xlabel('$t$', int, lat)
ylabel('$\Delta[t]$', int, lat)
% axis tight;
```

B.1.2 OFRF 求解

```
function [s] =H1(omega, n, d, alpha, omega_s)
% calculate the frf of LPF
% n: number of terms
% d: delta = 10^(-d)
s = zeros(size(omega));
a = cos(alpha);
delta = 10 ^(-d);
```

```
for i = 1 : length(omega)
    if omega(i) <= alpha
        phi = (2 * cos(omega(i)) - a + 1) / (a + 1);
        s(i) = delta * cos(n * acos(phi));
    else
        phi = (2 * cos(omega(i)) -a + 1)/ (a + 1);
        s(i) = (delta/2) * ((phi + sqrt(phi^2 - 1))^n + (phi - sqrt(phi^2-1))^n);
    end
end
phi = (2 * cos(omega_s)- a + 1)/(a + 1);
adj1 = (10^(-d)/2) * ((phi + sqrt(phi^2 - 1))^n + (phi-sqrt(phi^2 - 1))^n);
s = s / adj1;
end
```

B.2 OIRF 求解

```
function [h_t] = oirf(H, n, d, alpha, omega_s, type)
global int
global lat
% H: frf
% n: number of terms
% d: delta = 10^(-d)
% alpha: cutoff frequency
% omega_s: signal frequecy
% type = 1:lpf, type = 2:hpf
h_t = zeros(1, 2 * n + 1);
if type == 1
    H_1 = @(w)H(w, n, d, alpha, omega_s);
    omega = linspace(0, pi, 1000);
    Hejw = H_1(omega);
    plot(omega, Hejw);
    xlabel('$\omega$', int, lat);
    ylabel('$H_1(e^{\rm{j}\omega})$', int, lat);
    xlim([omega_s, pi]);
    grid on;
    set(gca,'XTick',[pi/4,pi/2, pi]);xticklabels={'\pi/4','\alpha', '\pi'};
    set(gca,'xticklabel',xticklabels);
    for k = -n : n
        f = @(w)H(w, n, d, alpha, omega_s) .* cos(k * w) * 2;
        h_t(k + n + 1) = 1 / (2 * pi) * integral(f, 0, pi);
    end
else
    if type == 2
        H_2 = @(w)H(2 * omega_s - w, n, d, 2 * omega_s - alpha, omega_s);
        omega = linspace(0, pi, 1000);
        Hejw = H_2(omega);
        plot(omega, Hejw);
        xlabel('$\omega$', int, lat);
        ylabel('$H_2(e^{\rm{j}\omega})$', int, lat);
        xlim([0, omega_s]);
        grid on;
        set(gca,'XTick',[0,pi/6, pi/4]);
        xticklabels={'0','\beta', '\pi/4'};
        set(gca,'xticklabel',xticklabels);
        for k = -n : n
            f = @(w)H(2 * omega_s - w, n, d, 2 * omega_s - alpha, omega_s) .* cos(k * w) * 2;
```

```
            h_t(k + n + 1) = 1 / (2 * pi) * integral(f, 0, pi);
        end
    end
end
end
```

B.2.1 作图

```
function []=stem_plot(x, y)
plot_range = 1 : 100;
stem(x(plot_range), y(plot_range), 'filled', 'MarkerSize', 3);
end
```

B.3 Shiny 可视化 App 代码

B.3.1 ui.R

```
#LSNR
library(shiny)
library(ggplot2)
dataset <- diamonds

fluidPage(

  titlePanel("Low Signal-Noise Ratio Dynamic Filter"),

  wellPanel(
    fluidRow(

      # signal, noise, cutoff frequency
      column(4,
      sliderInput('omega_s', 'Frequency of Signal', min=0.01, max=1.04,
                  value=0.5, step=0.01, round=-2)),
      column(4,
      sliderInput('cut_off', 'Cut-off Frequency', min=1.05, max=2.09, value = 1.57, step=0.01, round=-2)),
      column(4,
      sliderInput('omega_n', 'Frequency of Noise', min=2.10, max=3.14,
                  value=2.5 , step=0.01, round=-2))
    ),
    fluidRow(
      # amplitude
      column(4,
             sliderInput('nsr', 'Ratio of Noise to Signal,', min=1, max=10000, value=10000, step=1, round=0)
      ),
      # delta and N
      column(4,
      numericInput('d', 'd(delta = 10^(-d), e.g. input d = 2 means delta = 0.01)', 10, 1, 20)),


      column(4,
        uiOutput("terms")
      )
    )
  ),
```

```
    tabsetPanel(
      tabPanel("Original Signal and Input",
               fluidRow(
                 column(6, plotOutput('st')),
                 column(6, plotOutput('xt'))
               )
      ),
      tabPanel("Optimum IRF and FRF",
               fluidRow(
                 column(6, plotOutput('ht')),
                 column(6, plotOutput('Hw'))
               )
      ),
      tabPanel("Filtering Result and Error",
               fluidRow(
                 column(6, plotOutput('yt')),
                 column(6, plotOutput('Delta'))
               )
      )
    )
)
```

B.3.2 server.R

```
library(shiny)
source('functions.R')

function(input, output) {
  t = 1:2000
  dataset <- reactive({
    diamonds[sample(nrow(diamonds), input$sampleSize),]
  })

  St <- reactive({
    cos(input$omega_s * t)
  })
  nt <- reactive({
    input$nsr * cos(input$omega_n * t)
  })
  xt <- reactive({
    St() + nt()
  })

  ht <- reactive({
    oirf(input$N, input$d, input$cut_off, input$omega_s, input$omega_n)
  })

  result <- reactive({
    filter1(xt(), ht())
  })

  output$terms <- renderUI({
    N <- estimate_N(input$d, input$nsr, input$cut_off)
    "N" <- numericInput('N', paste('Number of terms(Recommended N =',
                                   estimate_N(input$d, input$nsr, input$cut_off),")"), N, 1, 100)
  })
```

```
  output$st <- renderPlot({
    stem_plot(t, St(), main="Signal",col="#32CD32", xlab="t", ylab=expression(S[t]))
  })

  output$xt <- renderPlot({
    stem_plot(t, xt(), main="Input", xlab="t", ylab=expression(x[t]), col="#00BFFF")
  })

  output$Hw <- renderPlot({
    ofrf(input$N, input$d, input$cut_off, input$omega_s, input$omega_n)
  })

  output$ht <- renderPlot({
    n <- (length(ht()) - 1) / 2
    t_h <- -n : n
    stem_plot(t_h, ht(), main="OIRF", xlab="t", ylab=expression(h[t]), col="#FF33AA")
  })

  output$yt <- renderPlot({
    yt <- result()$yt
    t_plot <- result()$t_plot
    stem_plot(t_plot, yt[t_plot], "Filtering Result", "t", expression(y[t]), col="#32CD32")
  })

  output$Delta <- renderPlot({
    delta_t = result()$yt - St()
    t_plot <- result()$t_plot
    stem_plot(t_plot, delta_t[t_plot], "Filtering Error", "t", expression(Delta[t]),col="#FF0000")
  })

}
```

B.3.3 functions.R

```
H <- function(omegas, n, d, alpha, omega_s){
  s <- rep(0, length(omegas))
  a <- cos(alpha)
  delta <- 10 ^(-d)
  i <- 1
  for (omega in omegas)
  {
    if (omega >= alpha)
    {
      phi <- (2 * cos(omega) - a + 1) / (a + 1)
      s[i] <- delta * cos(n * acos(phi))
    }
    else
    {
      phi <- (2 * cos(omega) -a + 1)/ (a + 1)
      s[i] <- (delta/2) * ((phi + sqrt(phi^2 - 1))^n + (phi - sqrt(phi^2-1))^n)
    }
    phi <- (2 * cos(omega_s)- a + 1)/(a + 1)
    adj1 <- (10^(-d)/2) * ((phi + sqrt(phi^2 - 1))^n + (phi-sqrt(phi^2 - 1))^n)
    s[i] <- s[i] / adj1;
    i <- i + 1
```

```
  }
  return(s)
}

test <- function(x){
  return(x)
}

oirf <- function(n, d, alpha, omega_s, omega_n){
  if (omega_s < omega_n)
  {
    H1 <- function(omega, n, d, alpha, omega_s){
      return(H(omega, n, d, alpha, omega_s))
    }
  }
  else
  {
    H1 <- function(omega, n, d, alpha, omega_s){
      return(H(2 * omega_s - omega, n, d, 2 * omega_s - alpha, omega_s))
    }
  }
  h <- rep(0, 2 * n + 1)
  for (k in -n : n)
  {
    f <- function(omega){
      return(H1(omega, n, d, alpha, omega_s) * cos(k * omega))
    }
    int <- integrate(f, 0, pi)
    h[k + n + 1] <- 1 / (2 * pi) * 2 * int[[1]];
  }
  return(h)
}

ofrf <- function(n, d, alpha, omega_s, omega_n){
  omega <- seq(omega_s, pi, 0.001)
  if (omega_s < omega_n)
  {
    H_omega <- H(omega, n, d, alpha, omega_s)
    return(plot(omega, H_omega, lty=1,col="#FFA500",type = "l",
                main="OFRF", xlab=expression(omega), ylab="H", xlim = c(omega_s, pi)))
  }
  else
  {
    H_omega <- H(2 * omega_s - omega, n, d, 2 * omega_s - alpha, omega_s)
    return(plot(omega, H_omega, lty=1, type = "l", xlim=c(0, omega_s)))
  }
}

filter1 <- function(xt, ht){
  n <- (length(ht) - 1) / 2
  yt <- convolve(xt, rev(ht), type = "open");
  yt <- yt[(n + 1) : (n + length(xt))]
  t_plot <- (n + 1) : (length(xt) - n)
  return(list(yt=yt, t_plot=t_plot))
}

stem_plot <- function(x, y, main, xlab, ylab, col="black"){
  plot_range <- 1 : 100
```

```
  plot(x[plot_range], y[plot_range], type='h', main=main, xlab=xlab, ylab=ylab, col=col)
  points(x[plot_range], y[plot_range], pch=19, col=col)
}

estimate_N <- function(d, nsr, alpha){
  a <- cos(alpha)
  b <- (3 - a) / (a + 1)
  c <- b + sqrt(b ^ 2 - 1)
  d <- 2 / 10 ^ (-d) * nsr
  return(ceiling(log(d) / log(c)))
}
```

B.4 其他代码

其他相关的代码见https://github.com/thu-jw/LSNR-Dynamic-Filter