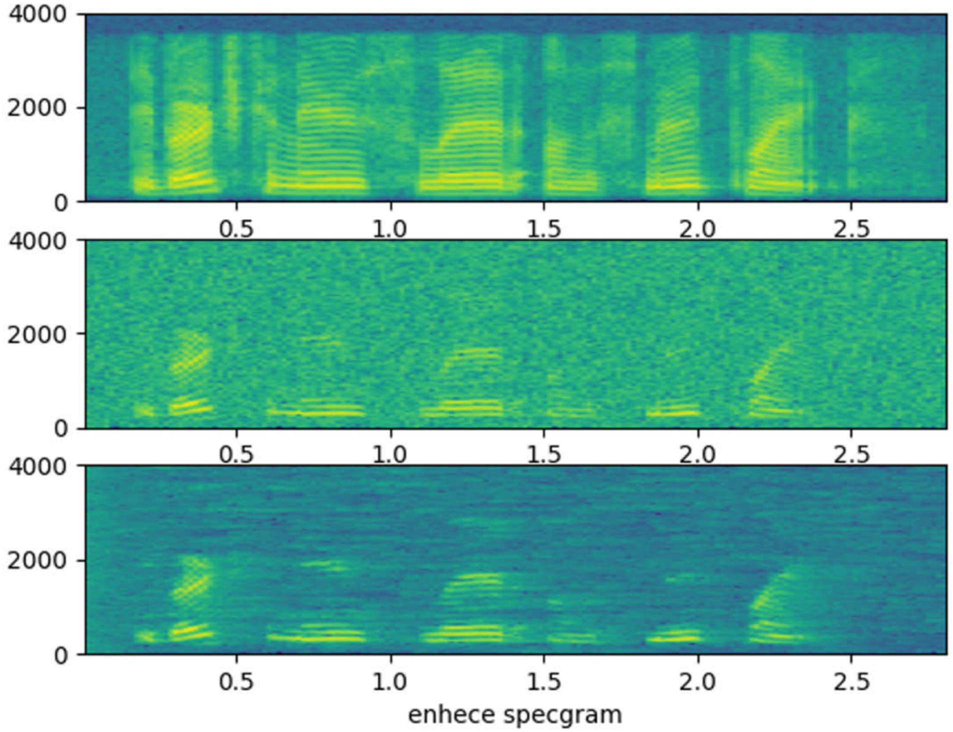
语音增强-最小均方误差估计

Speech Enhancement- MMSE



于泓
鲁东大学
信息与电气工程学院
2021.7.19

基于最小均方误差估计的语音增强

- 最小均方误差估计定义（**M**inimum **M**ean **S**quare **E**rror Estimation）

假设有两个随机变量 X , Y 他们之间均在联合分布，其中 Y **为观测信号**,

利用观测信号 Y 对 X 进行估计得到 $\hat{X}$,

令 $E\left(\left|X-\hat{X}\right|^2 |Y\right)$ 最小

$$\frac{\partial E\left(\left|X-\hat{X}\right|^2 |Y\right)}{\partial \hat{X}} = -2E\left((\hat{X}-X)\middle|Y\right) = -2\left(\hat{X}-E(X \mid Y)\right) = 0$$

$$\hat{X}=E(X \mid Y)$$

具体到语音去噪的任务

$\mathrm{X} \longrightarrow X_k \longrightarrow$ **clean** 第k个频点上的幅度值（**实**数）

$\hat{\mathrm{X}} \longrightarrow \hat{X}_k \longrightarrow$ **Enhance** 第k个频点上的幅度值（**实**数）

$\mathrm{Y} \longrightarrow Y(\omega_k) \longrightarrow$ **Noisy** 第k个频点上的值（**复**数）

$\mathrm{E}\left(\left|\mathrm{X}\text{-}\hat{\mathrm{X}}\right|^2 \middle| \mathrm{Y}\right) \longrightarrow \mathrm{E}\left(\left|X_k - \hat{X}_k\right|^2 \middle| Y(\omega_k)\right) \longrightarrow$ 关于幅度值的MMSE估计

注意与维纳滤波的区别 $\mathrm{E}\left[\left|X(\omega_k)\text{-}\hat{X}(\omega_k)\right|^2\right]$

应当是所有频点
根据独立假设只取第k个频点

$$\hat{X}_k = \mathrm{E}(X_k \left| Y(\omega_k)\right.) = \int X_k P(X_k \mid Y(\omega_k)) dX_k$$

条件概率密度函数

傅里叶变换数据的概率密度函数PDF的假设

实部/虚部 0均值

中心极限定理

$$Y(\omega_k)=\sum_{n=0}^{N-1} y(n)e^{-j\omega_k n}=y(0)+a_1y(1)+a_2y(2)+\cdots+a_{N-1}y(N-1)$$

实部/满足0均值高斯分布

y(n) 0均值

0均值复高斯分布

$$Y(\omega_k)=\mathrm{Re}(Y(\omega_k))+j\,\mathrm{Im}(Y(\omega_k))$$

$$P(Y(\omega_k))=\frac{1}{\pi\lambda_Y(k)}\exp\left(-\frac{Y(\omega_k)}{\lambda_Y(k)}\right)$$

实部 虚部 → 高斯分布 均值 0 方差 $\delta^2=\frac{\lambda_Y(k)}{2}$

$$Y(\omega_k)=Y_k e^{j\theta_k}$$

$$\lambda_Y(k)=\mathrm{E}\left[\left(Y(\omega_k)-m\right)^*\left(Y(\omega_k)-m\right)\right]=\mathrm{E}\left[\left|Y(\omega_k)\right|^2\right]$$

模值：瑞丽分布 $P(Y_k)=\frac{Y_k}{\frac{1}{2}\lambda_Y(k)}\exp\left(-\frac{Y_k}{\lambda_Y(k)}\right)$

相位：-pi~ pi 均匀分布 $P(\theta_k)=\frac{1}{2\pi}$

在语音增强任务中，根据幅度最小均方误差估计的原则可以得到:

$$\hat{X}_k = \mathrm{E}(X_k \left| Y(\omega_k)\right.) = \int x_k P(x_k \mid Y(\omega_k)) dx_k$$

$$= \frac{\int_0^{\infty} x_k P(Y(\omega_k) \mid x_k) P(x_k) dx_k}{\int_0^{\infty} P(Y(\omega_k) \mid x_k) P(x_k) dx_k}$$

其中 $P(Y(\omega_k) \mid x_k) P(x_k) = \int\limits_0^{2\pi} P(Y(\omega_k) \mid x_k, \theta_x) P(x_k, \theta_x) d\theta_x$

$$\hat{X}_k = \frac{\int_0^{\infty}\int_0^{2\pi} x_k P(Y(\omega_k) \mid x_k, \theta_x) P(x_k, \theta_x) d\theta_x dx_k}{\int_0^{\infty}\int_0^{2\pi} P(Y(\omega_k) \mid x_k, \theta_x) P(x_k, \theta_x) d\theta_x dx_k}$$

因为 $Y(\omega_k) = X(\omega_k) + D(\omega_k)$

$$p(Y(\omega_k) \mid x_k, \theta_x) = \frac{1}{\pi \lambda_d(k)} \exp\left\{ -\frac{1}{\lambda_d(k)} \left| Y(\omega_k) - X(\omega_k) \right|^2 \right\}$$

$$p(x_k, \theta_x) = \frac{x_k}{\pi \lambda_x(k)} \exp\left\{ -\frac{x_k^2}{\lambda_x(k)} \right\}$$

x_k与θ_x相互独立

带入公式

$$\hat{X}_k = \frac{\int_0^\infty \int_0^{2\pi} x_k P(Y(\omega_k) \mid x_k, \theta_x) P(x_k, \theta_x) d\theta_x dx_k}{\int_0^\infty \int_0^{2\pi} P(Y(\omega_k) \mid x_k, \theta_x) P(x_k, \theta_x) d\theta_x dx_k}$$

计算可得

Gamma函数

合流超几何函

$$\hat{X}_k = \sqrt{\lambda_k}\ \Gamma(1.5)\ \Phi(-0.5, 1; -v_k)$$

$$\lambda_k = \frac{\lambda_x(k)\lambda_d(k)}{\lambda_x(k) + \lambda_d(k)} = \frac{\lambda_x(k)}{1 + \xi_k}$$

$$v_k = \frac{\xi_k}{1 + \xi_k} \gamma_k$$

$$\gamma_k = \frac{Y_k^2}{\lambda_d(k)}$$ 后验信噪比

$$\xi_k = \frac{\lambda_x(k)}{\lambda_d(k)}$$ 先验信噪比

进一步化简可得

$$\hat{X}_k = \underbrace{\frac{\sqrt{\pi}}{2}\frac{\sqrt{v_k}}{\gamma_k}\exp\left(-\frac{v_k}{2}\right)\left[(1+v_k)I_o\left(\frac{v_k}{2}\right)+v_k I_1\left(\frac{v_k}{2}\right)\right]}_{\text{关于先验/后验信噪比的增益函数}}Y_k$$

贝塞尔函数

ξ_k γ_k

其中 ξ_k 对噪声抑制起主要的作用，
并且MMSE的方法对 ξ_k 的波动比较敏感

因此需要对 ξ_k 需要进行准确且较为平滑的估计

判决导引法（Decision-Directed）

$$\xi_k(m)=\frac{E\{X_k^2(m)\}}{\lambda_d(k,m)}$$

第m帧的先验信噪比

$$\xi_k(m)=\frac{E\left\{Y_k^2(m)-D_k^2(m)\right\}}{\lambda_d(k,m)}$$

$$=\frac{E\left\{Y_k^2(m)\right\}}{\lambda_d(k,m)}-\frac{E\left\{D_k^2(m)\right\}}{\lambda_d(k,m)}$$

$$=E\{\gamma_k(m)\}-1$$

从 γ_k 进行推导

两者结合 采用递推估计的方法

$$\hat{\xi}_k(m)=a\frac{\hat{X}_k^2(m-1)}{\lambda_d(k,m-1)}+(1-a)\max[\gamma_k(m)-1,0]$$

$$0<a<1$$

$$\hat{\xi}_k(0)=a+(1-a)\max[\gamma_k(0)-1,0]$$

初始值

引入限制参数 $\xi_{\min}$

$$\hat{\xi}_k(m)=\max\left[a\frac{\hat{X}_k^2(m-1)}{\lambda_d(k,m-1)}+(1-a)\max[\gamma_k(m)-1,0],\xi_{\min}\right]$$

基于VAD的噪声估计

先验知识：能量比较小的语音帧，通常是噪声帧

流程：(1) 设定一个SNR阈值θ

(2) 计算语音前M帧的平均能量作为噪声能量 E_n $$E_n = \sum_k \lambda_d(k)$$

(3) for t = 1:N 对每一帧进行遍历

计算每帧的能量 E_s 并计算信噪比 SNR = E_s/E_n $$E_Y = \sum_k \lambda_Y(k)$$

如果 SNR< θ

$$\lambda_d(k) = \mu * \lambda_d(k) + (1-\mu)\lambda_Y(k)$$

```
def enh_mmse(noisy,noise,para):
    n_fft = para["n_fft"]
    hop_length = para["hop_length"]
    win_length = para["win_length"]

    S_noisy = librosa.stft(noisy,n_fft=n_fft, hop_length=hop_length, win_length=win_length)
    S_noise = librosa.stft(noise,n_fft=n_fft, hop_length=hop_length, win_length=win_length)

    phase_nosiy = np.angle(S_noisy)
    mag_noisy = np.abs(S_noisy)

    D,T = np.shape(mag_noisy)


    mag_nosie = np.mean(np.abs(S_noise),axis=1)
    power_noise = mag_nosie**2

    mag_enhance = np.zeros([D,T])
    aa = para["a_DD"]
```

```python
for i in range(T):

    # 获取每一帧的 能量谱和幅度谱
    mag_frame = mag_noisy[:,i]
    power_frame = mag_frame**2

    # 获取用来进行 VAD 计算的 信噪比
    SNR_VAD = 10 * np.log10(np.sum(power_frame)/np.sum(power_noise))

    # 计算后验信噪比
    gamma = np.minimum(power_frame / power_noise , para["max_gamma"])

    # 计算先验信噪比
    if i == 0:
        ksi = aa + (1 - aa) * np.maximum(gamma - 1 , 0)
    else:
        ksi = aa * power_enhance_frame / power_noise + (1 - aa) * np.maximum(gamma - 1 , 0)
        # 对 ksi 的最小值进行限制
        ksi = np.maximum(para["ksi_min"] , ksi)
```

$$\hat{\xi}_k(m) = a\frac{\hat{X}_k^2(m-1)}{\lambda_d(k,m-1)} + (1-a)\max[\gamma_k(m)-1, 0]$$

$$\hat{\xi}_k(m) = \max\left[a\frac{\hat{X}_k^2(m-1)}{\lambda_d(k,m-1)} + (1-a)\max[\gamma_k(m)-1, 0], \xi_{\min}\right]$$

```python
    # 根据 VAD 更新 power_noise
    mu = para["mu_VAD"]
    if SNR_VAD < para["th_VAD"]:
        power_noise = mu * power_noise + (1 - mu) * power_frame

    H = para["fun_GAN"] (ksi,gamma)

    mag_enhance_frame = H * mag_frame
    mag_enhance[:,i] = mag_enhance_frame

    power_enhance_frame = mag_enhance_frame ** 2

S_enhec = mag_enhance*np.exp(1j*phase_nosiy)

enhance = librosa.istft(S_enhec, hop_length=hop_length, win_length=win_length)
return enhance
```

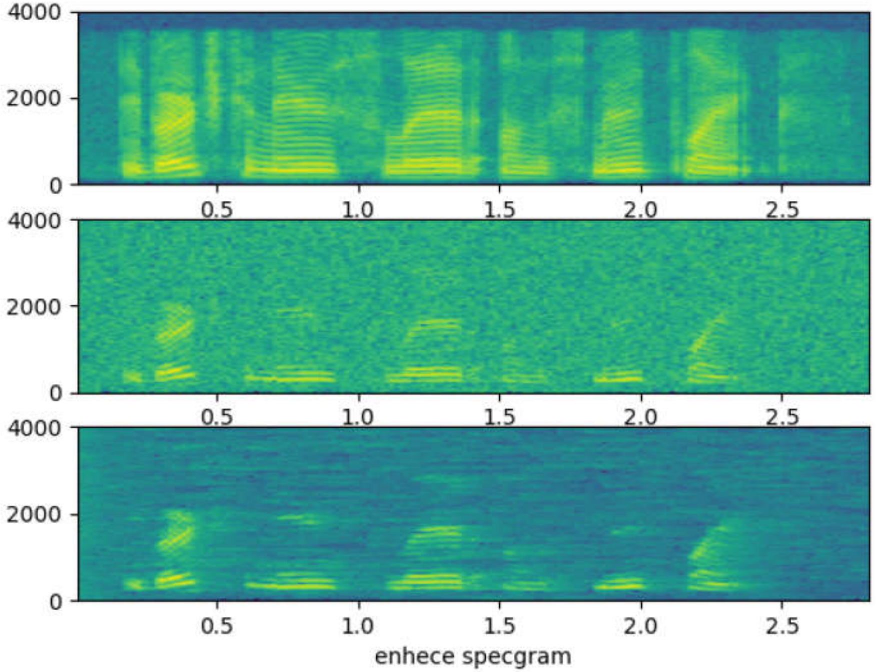
$$\lambda_d(k) = \mu * \lambda_d(k) + (1-\mu)\lambda_Y(k)$$

```python
def Gan_mmse(ksi,gamma):
    c = np.sqrt(np.pi) / 2

    v = gamma * ksi / (1 + ksi)

    j_0 = sp.iv(0 , v/2)
    j_1 = sp.iv(1 , v/2)
    C = np.exp(-0.5 * v)
    A = ((c * (v ** 0.5)) * C) / gamma
    B = (1 + v) * j_0 + v * j_1
    hw = A * B     #[7.40]
    return hw
```

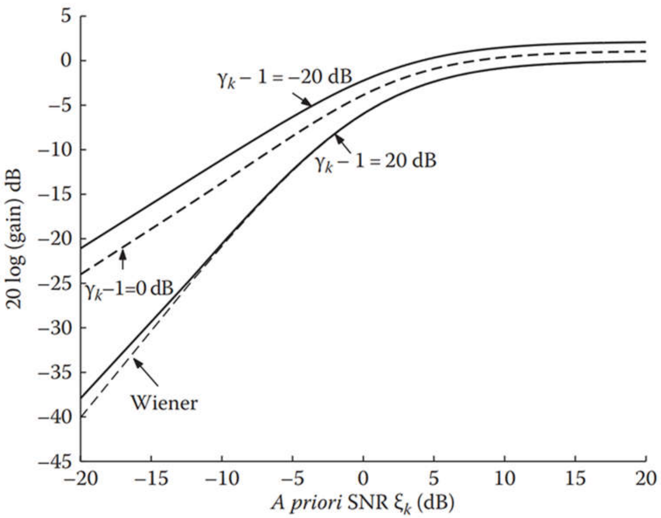
$$\hat{X}_k = \frac{\sqrt{\pi}}{2}\frac{\sqrt{v_k}}{\gamma_k}\exp\left(-\frac{v_k}{2}\right)\left[(1+v_k)I_o\left(\frac{v_k}{2}\right)+v_k I_1\left(\frac{v_k}{2}\right)\right]Y_k$$

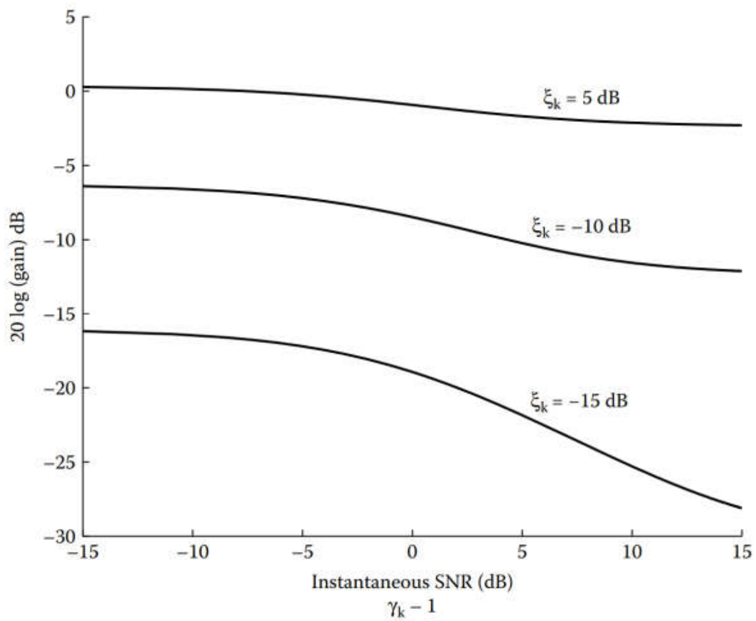
```python
if __name__ == "__main__":

    # 读取干净语音
    clean_wav_file = "sp01.wav"
    clean,fs = librosa.load(clean_wav_file,sr=None)
    print(fs)
    # 读取读取噪声语音
    noisy_wav_file = "in_SNR5_sp01.wav"
    noisy,fs = librosa.load(noisy_wav_file,sr=None)


    # 设置模型参数
    para_mmse = {}
    para_mmse["n_fft"] = 256
    para_mmse["hop_length"] = 128
    para_mmse["win_length"] = 256
    para_mmse["max_gamma"] =40 # gamma 的最大值
    para_mmse["a_DD"] = 0.98  # 利用 decision-direct 进行 ksi更新
    para_mmse["ksi_min"] = 10 ** (-25 / 10)   # ksi最小值 -25dB

    para_mmse["mu_VAD"] = 0.98 # VAD噪声跟踪的参数
    para_mmse["th_VAD"] = 3  # VAD 判定阈值 3db

    para_mmse["fun_GAN"] = Gan_mmse
    # mmse 增强
    enhance = enh_mmse(noisy,noisy[:1000],para_mmse)

    sf.write("enhce_sqr_mmse.wav",enhance,fs)
```

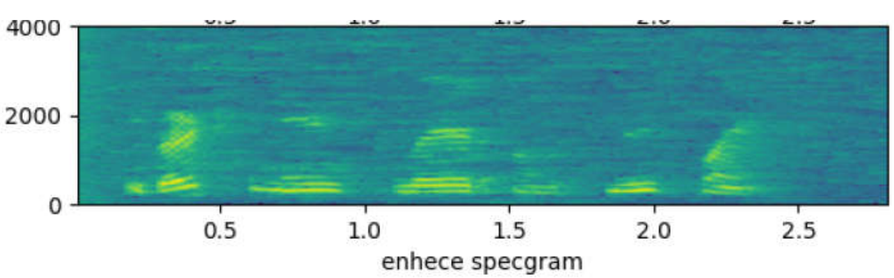
```python
plt.subplot(3,1,1)
plt.specgram(clean,NFFT=256,Fs=fs)
plt.xlabel("clean specgram")
plt.subplot(3,1,2)
plt.specgram(noisy,NFFT=256,Fs=fs)
plt.xlabel("noisy specgram")
plt.subplot(3,1,3)
plt.specgram(enhance,NFFT=256,Fs=fs)
plt.xlabel("enhece specgram")
plt.show()
```

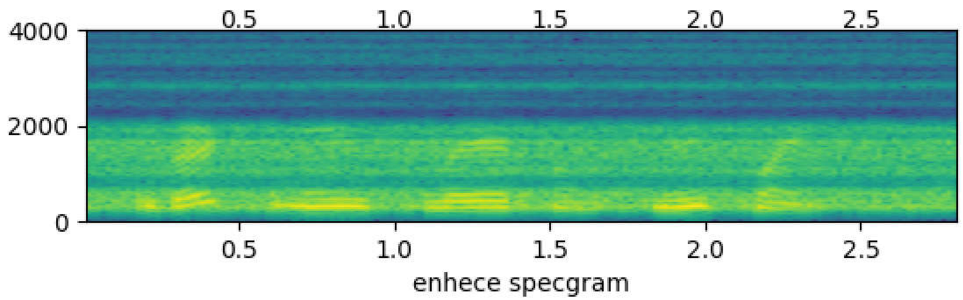


MMSE 与维纳滤波的区别









Log-MMSE

$$E\{(\log X_k - \log \hat{X}_k)^2\}$$

根据最小均方误差估计

$$\log \hat{X}_k = E\{\log X_k \mid Y(\omega_k)\}$$

$$\hat{X}_k = \exp(E\{\log X_k \mid Y(\omega_k)\})$$

矩母函数**moment-generating function**

$$MGF_X(t) := E[e^{tX}] = \begin{cases} \sum_x e^{tx} \cdot \overset{\text{PMF}}{P(x)} & X: \text{discrete} \\ \int_x e^{tx} \cdot \underset{\text{PDF}}{f(x)}\, dx & X: \text{continuous} \end{cases}$$

知乎 @jinzhao

$$E(X) = \frac{d}{dt} MGF_X(t) \Big|_{t=0} = MGF'_X(0)$$

① $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$

then,

$$e^{tx} = 1 + tx + \frac{(tx)^2}{2!} + \frac{(tx)^3}{3!} + \cdots + \frac{(tx)^n}{n!}$$

知乎 @jinzhao

② $E(e^{tx}) = E\left(1 + tx + \frac{(tx)^2}{2!} + \frac{(tx)^3}{3!} + \cdots + \frac{(tx)^n}{n!}\right)$

$$= E(1) + tE(x) + \frac{t^2}{2!}E(x^2) + \frac{t^3}{3!}E(x^3) + \cdots + \frac{t^n}{n!}E(x^n)$$

知乎 @jinzhao

③ $\frac{d}{dt}E(e^{tx}) = \frac{d}{dt}\left(E(1) + tE(x) + \frac{t^2}{2!}E(x^2) + \frac{t^3}{3!}E(x^3) + \cdots + \frac{t^n}{n!}E(x^n)\right)$

plug $t=0$ in...

$$= 0 + E(x) + 0 + 0 + \cdots + 0$$

$$= E(x)$$

知乎 @jinzhao

$$Z_k = \log X_k$$

矩母函数

$$\Phi_{Z_k|Y(\omega_k)}(\mu) = E\{\exp[\mu Z_k] | Y(\omega_k)\}$$

$$= E\left\{X_k^{\mu} | Y(\omega_k)\right\}$$

$$E\{\log X_k | Y(\omega_k)\} = \frac{d}{d\mu}\Phi_{Z_k|Y(\omega_k)}(\mu)\Big|_{\mu=0}$$

$$\Phi_{Z_k|Y(\omega_k)}(\mu) = E\left\{X_k^{\mu} | Y(\omega_k)\right\}$$

$$= \frac{\int_0^{\infty}\int_0^{2\pi} x_k^{\mu} p(Y(\omega_k) | x_k, \theta_x) p(x_k, \theta_x) d\theta_x dx_k}{\int_0^{\infty}\int_0^{2\pi} p(Y(\omega_k) | x_k, \theta_x) p(x_k, \theta_x) d\theta_x dx_k}$$

$$\Phi_{Z_k|Y(\omega_k)}(\mu) = \lambda_k^{\mu/2}\Gamma\left(\frac{\mu}{2}+1\right)\Phi\left(-\frac{\mu}{2}, 1; -v_k\right)$$

$$E\{\log X_k | Y(\omega_k)\} = \frac{1}{2}\log\lambda_k + \frac{1}{2}\log v_k + \frac{1}{2}\int_{v_k}^{\infty}\frac{e^{-t}}{t}dt$$

$$\hat{X}_k = \frac{\xi_k}{\xi_k + 1} \exp\left\{ \frac{1}{2} \int_{v_k}^{\infty} \frac{e^{-t}}{t} dt \right\} Y_k$$

$$\triangleq G_{LSA}(\xi_k, v_k)\, Y_k$$

```python
def Gan_log_mmse(ksi,gamma):
    def integrand(t):
        return np.exp(-t) / t
    A = ksi / (1 + ksi)
    v = A * gamma
    ei_v = np.zeros(len(v))
    for i in range(len(v)):
        ei_v[i] = 0.5 * inte.quad(integrand,v[i],np.inf)[0]
    hw = A * np.exp(ei_v)
    return hw
```

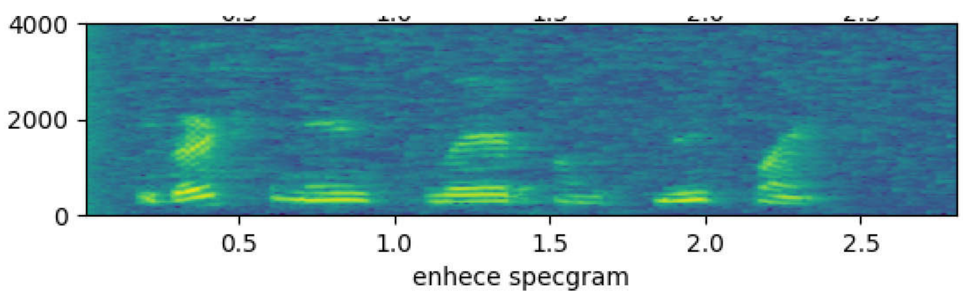
$$Ei(x) = \int_{x}^{\infty} \frac{e^{-x}}{x} dx \approx \frac{e^{x}}{x} \sum_{k} \frac{k!}{x^{k}}$$

近似计算:

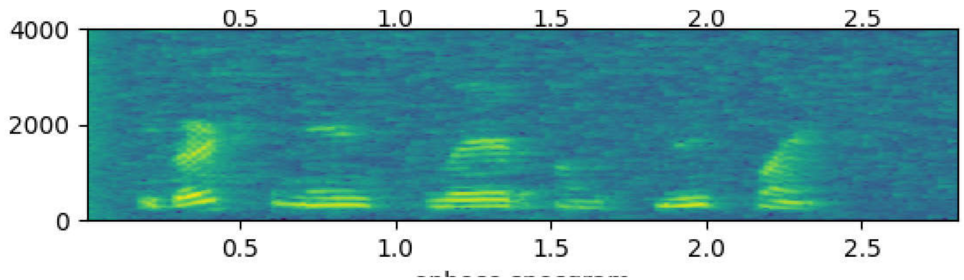
$$\int_{\nu(n,t)}^{\infty} \frac{e^{-t}}{t} \mathrm{dt} \approx \begin{cases} \nu(n,t) < 0.1 & -2.3 * log_{10}(\nu(n,t)) - 0.6 \\ 0.1 \le \nu(n,t) < 1 & -1.544 * log_{10}(\nu(n,t)) + 0.166 \\ \nu(n,t) > 1 & 10^{-0.52 * \nu(n,t) - 0.26} \end{cases}$$

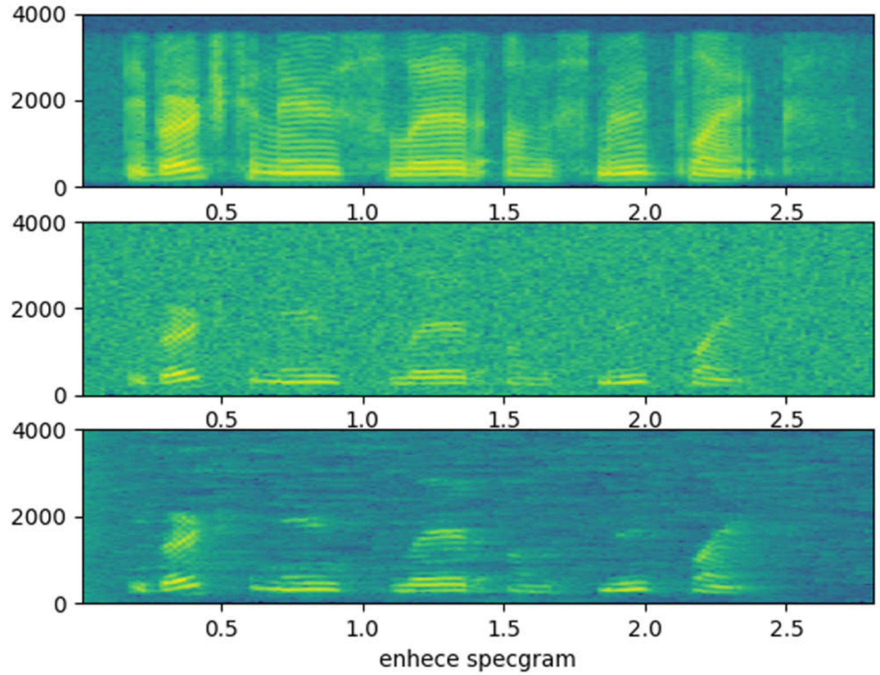
```python
def Gan_log_mmse2(ksi,gamma):

    A = ksi / (1 + ksi)
    v = A * gamma
    ei_v = np.zeros(len(v))
    for i in range(len(v)):

        if v[i]<0.1:
            ei_v[i] = -2.3*np.log10(v[i])-0.6
        elif v[i]>=0.1 and v[i]<1:
            ei_v[i] = -1.544*np.log10(v[i]) + 0.166
        else:
            ei_v[i] = np.power(10,-0.53*v[i]-0.26)

    hw = A * np.exp(0.5*ei_v)
    return hw
```

Log-MMSE



Log-MMSE2





MMSE

幅度平方估计

$$\hat{X}_k^2 = E[X_k^2|Y_k]$$

$$\hat{X}_k^2 = \frac{\xi_k}{1+\xi_k}\left(\frac{1+v_k}{\gamma_k}\right)Y_k^2$$

$$H_k = \sqrt{\frac{\xi_k}{1+\xi_k}\left(\frac{1+v_k}{\gamma_k}\right)}$$

```
def Gan_sqr_mmse(ksi,gamma):

    A = ksi / (1 + ksi)
    v = A * gamma

    B = (1 + v) / gamma
    hw = np.sqrt(A * B)


    return hw
```