Lecture

Speech and Audio Signal Processing

TECHNISCHE UNIVERSITÄT DARMSTADT

Lecture 5: Noise reduction & Dereverberation

GRADUATE SCHOOL computational engineering

Content

- Wiener filter
- Realization in the frequency domain
- Extensions of the basic approach
- Modified noise reduction procedure
- Dereverberation

Setup

Design of filters by means of minimizing the squared error (according to Gauß)

Independent development

1941: A. Kolmogoroff: Interpolation und Extrapolation von stationären zufälligen Folgen, Izv. Akad. Nauk SSSR Ser. Mat. 5, pp. 3 – 14, 1941 (in Russian)

1942: N. Wiener: *The Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Applications*, J. Wiley, New York, USA, 1949 (originally published in 1942 as MIT Radiation Laboratory Report)

Assumptions & Design criteria:

- One Wiener filter application: Separate a desired signal from an additive noise.
- The desired signal (typically speech) and noise are modeled as random processes.
- The filter is designed based on statistical properties up to the second order for speech and noise.

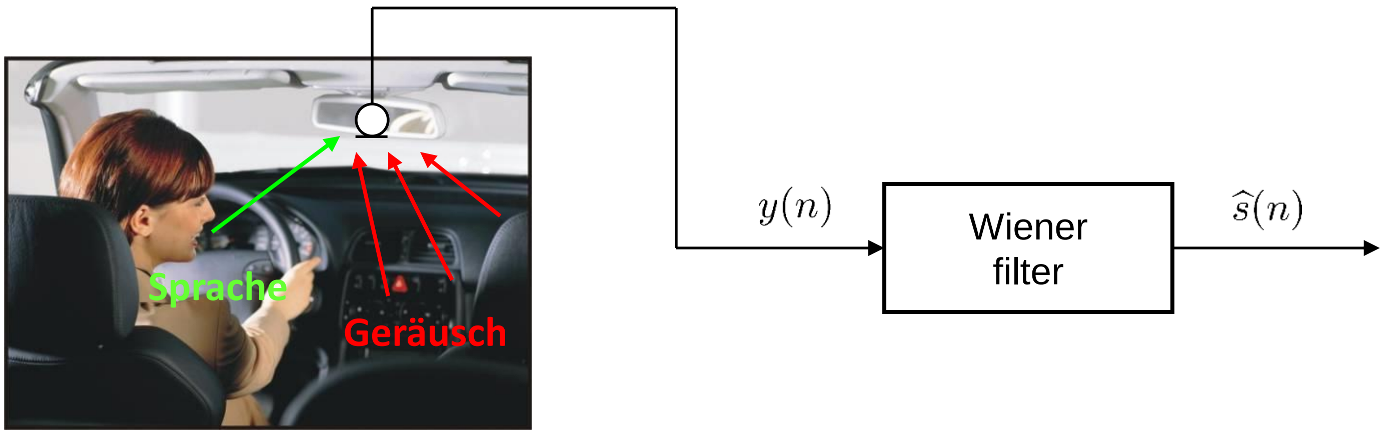
Literature

Basics of the Wiener filter:

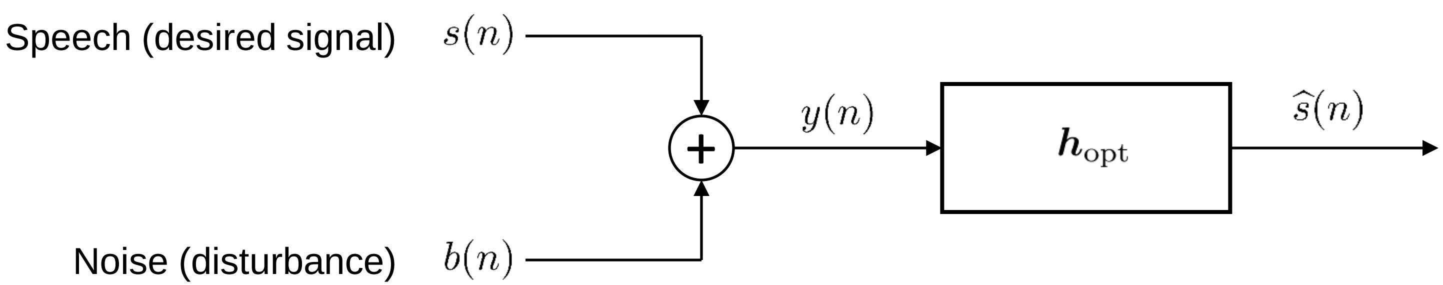
- E. Hänsler / G. Schmidt: Acoustic Echo and Noise Control – Kapitel 5 (Wiener Filter), Wiley, 2004
- E. Hänsler: Statistische Signale: Grundlagen und Anwendungen – Kapitel 8 (Optimalfilter nach Wiener und Kolmogoroff), Springer, 2001
- M. S.Hayes: Statistical Digital Signal Processing and Modeling – Kapitel 7 (Wiener Filtering), Wiley, 1996
- S. Haykin: Adaptive Filter Theory – Kapitel 2 (Wiener Filters), Prentice Hall, 2002

The Wiener filter – a noise reduction application example

Application:

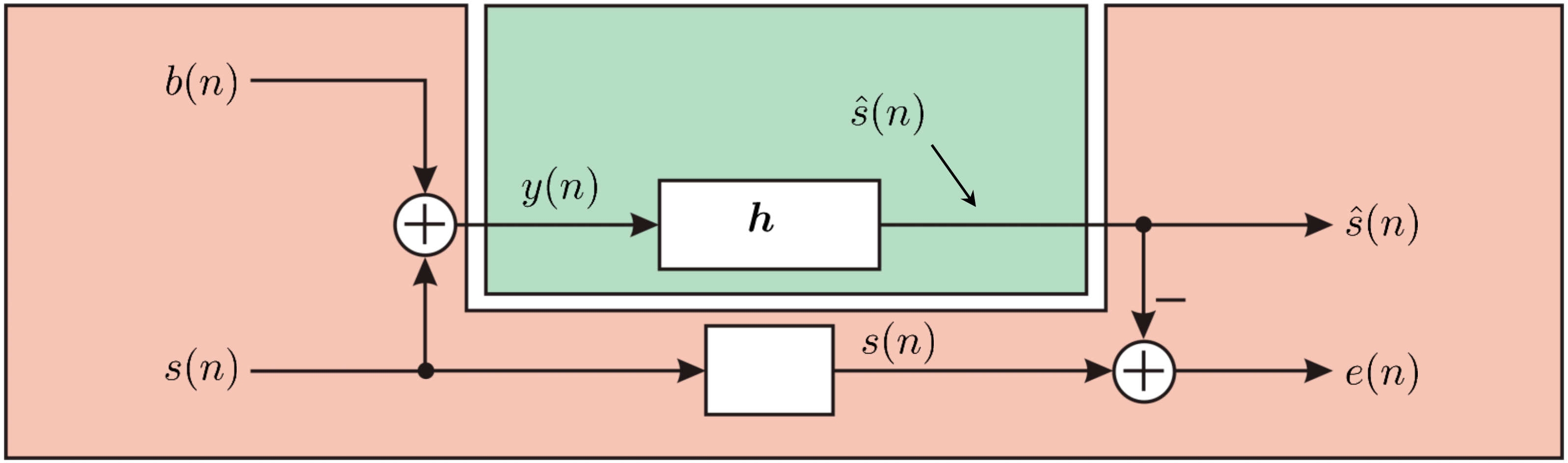


Model:



The Wiener filter

Structure in the time domain:



FIR filter structure:

$$\hat{s}(n) = \sum_{i=0}^{N-1} h_i \, y(n-i)$$

Optimization criterion:

$$\mathrm{E}\{e^2(n)\} \underset{h_i = h_{i,\mathrm{opt}}}{\longrightarrow} \min$$

The Wiener filter

Further assumptions:

- The target signal $s(n)$ and the noise $b(n)$ are zero-mean and uncorrelated, i.e. orthogonal:

$$m_s = m_b = 0,\ r_{sb}(l) = m_s \cdot m_b = 0$$

Calculation of the optimum filter coefficients:

$$\mathrm{E}\{e^2(n)\} \underset{h_i = h_{i,\mathrm{opt}}}{\longrightarrow} \min$$

$$\begin{aligned} \frac{\partial}{\partial h_i} \mathrm{E}\{e^2(n)\}\Big|_{h_i = h_{i,\mathrm{opt}}} &= 0 \\ 2\,\mathrm{E}\left\{e(n)\,\frac{\partial}{\partial h_i} e(n)\right\}\Big|_{h_i = h_{i,\mathrm{opt}}} &= 0 \end{aligned}$$

The Wiener filter

Calculation of the optimum filter coefficients:

$$2\,\mathrm{E}\left\{e(n)\,\frac{\partial}{\partial h_i}e(n)\right\}\bigg|_{h_i=h_{i,\mathrm{opt}}} = 0$$

Take the error signal: $$e(n) = s(n) - \sum_{i=0}^{N-1} h_i\, y(n-i)$$

$$\begin{aligned}
2\,\mathrm{E}\left\{\left(s(n) - \sum_{j=0}^{N-1} h_j\, y(n-j)\right) y(n-i)\right\}\Bigg|_{h_i=h_{i,\mathrm{opt}}} &= 0 \\
r_{sy}(i) - \sum_{j=0}^{N-1} h_{j,\mathrm{opt}}\, r_{yy}(i-j) &= 0
\end{aligned}$$

Target signal and noise are orthogonal: $$r_{sy}(l) = r_{ss}(l) + \underbrace{r_{sb}(l)}_{=0} = r_{ss}(l)$$

$$r_{ss}(i) - \sum_{j=0}^{N-1} h_{j,\mathrm{opt}}\, r_{yy}(i-j) = 0 \qquad \forall i \in [0,\ldots,N-1]$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Calculation of the optimum filter coefficients:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(1) & \dots & r_{yy}(N-1) \\ r_{yy}(1) & r_{yy}(0) & \dots & r_{yy}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}(N-1) & r_{yy}(N-2) & \dots & r_{yy}(0) \end{bmatrix} \begin{bmatrix} h_{0,\text{opt}} \\ h_{1,\text{opt}} \\ \vdots \\ h_{N-1,\text{opt}} \end{bmatrix} = \begin{bmatrix} r_{ss}(0) \\ r_{ss}(1) \\ \vdots \\ r_{ss}(N-1) \end{bmatrix}$$

Difficulties:

- The autocorrelation function of the speech signal cannot simply be measured.

  Solution: $r_{ss}(l) = r_{yy}(l) - r_{bb}(l)$ with a noise autocorrelation function to be measured in speech pauses.

- The inverse of the autocorrelation matrix does not necessarily exist since the matrix is only non-negative definite.

  Solution: Calculation in the frequency domain.

- The solution of the above matrix equation system is computational complex (and has to be redone every approx. 20 msec).

  Solution: Calculation in the frequency domain.

The Wiener filter – in the frequency domain

TECHNISCHE UNIVERSITÄT DARMSTADT

Time domain solution:

$$r_{ss}(i) - \sum_{j=0}^{N-1} h_{j,\mathrm{opt}}\, r_{yy}(i-j) \;=\; 0$$

Frequency domain solution:

$$\begin{aligned} S_{ss}(\Omega) - H_{\mathrm{opt}}(e^{j\Omega})\, S_{yy}(\Omega) &= 0 \\ H_{\mathrm{opt}}(e^{j\Omega}) &= \frac{S_{ss}(\Omega)}{S_{yy}(\Omega)} \end{aligned}$$

Orthogonality of speech and noise: $S_{ss}(\Omega) = S_{yy}(\Omega) - S_{bb}(\Omega)$

$$H_{\mathrm{opt}}(e^{j\Omega}) \;=\; 1 - \frac{S_{bb}(\Omega)}{S_{yy}(\Omega)}$$

The Wiener filter – in the frequency domain

TECHNISCHE UNIVERSITÄT DARMSTADT

Frequency domain solution:

$$H_{\mathrm{opt}}(e^{j\Omega}) \quad = \quad 1 - \frac{S_{bb}(\Omega)}{S_{yy}(\Omega)}$$

Approximation with short-term estimates:

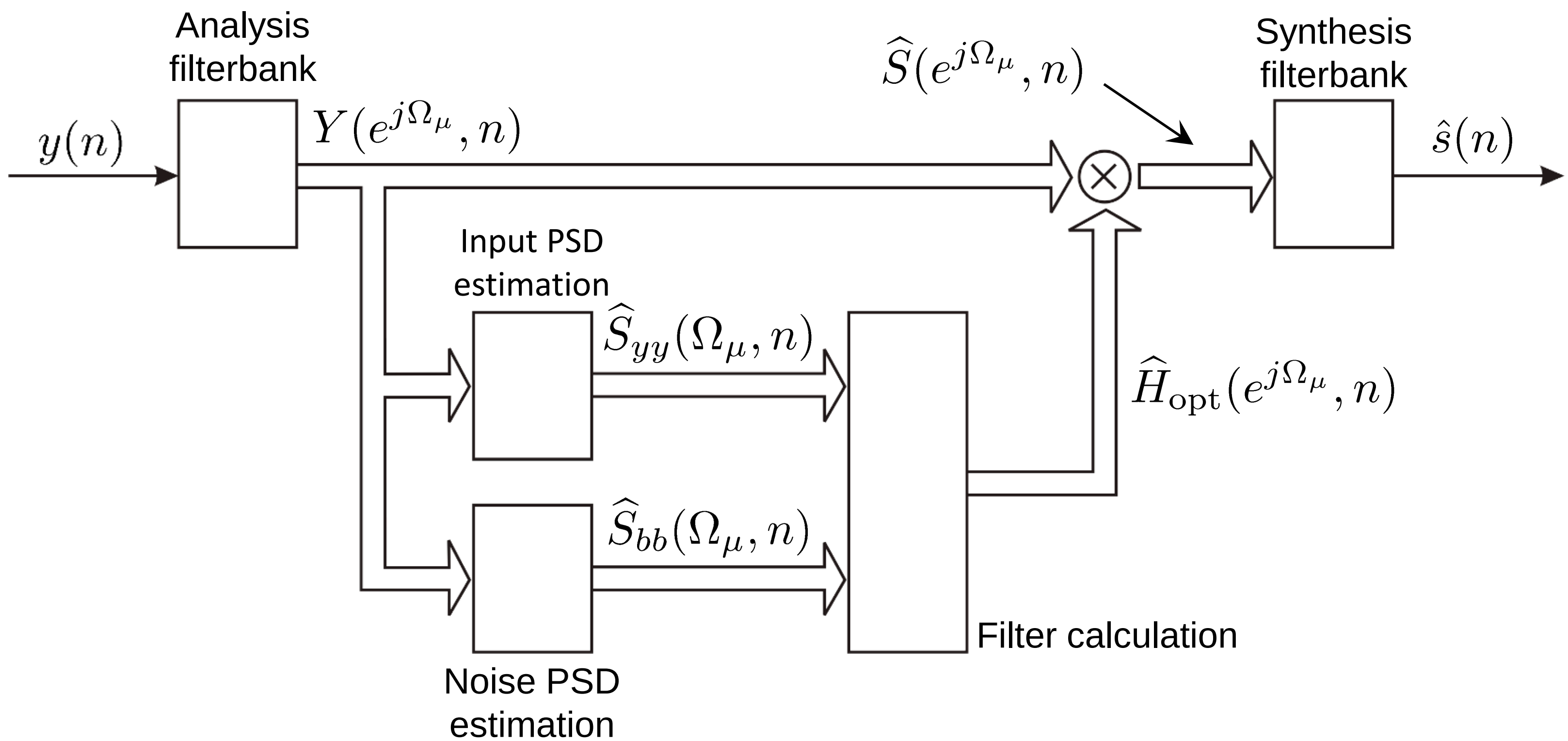
$$\widehat{H}_{\mathrm{opt}}(e^{j\Omega}, n) \quad = \quad \max\left\{0,\, 1 - \frac{\widehat{S}_{bb}(\Omega, n)}{\widehat{S}_{yy}(\Omega, n)}\right\}$$

Typical solution:

- Realization with a filter bank system (Application of adaptive attenuation factors in each subband)
- The prototype low-pass of the filter-bank should have a length between 15 and 100 msec.
- The subsampling rate (sample time of the sub-band signals) should be between 1 and 20 msec.
- The basic Wiener formula will be modified in order to be suitable for practical applications: Over-estimation, Limitation of the attenuation, etc.

The Wiener filter – in the frequency domain

Processing structure:



PSD = power spectral density

M sub-bands with a discrete frequency index:

Ω_μ with: $0 \leq \mu \leq M$

The Wiener filter – in the frequency domain

Power spectral density estimation for the input signal:

$$\widehat{S}_{yy}(\Omega_\mu, n) = \left|Y(e^{j\Omega_\mu}, n)\right|^2$$

Theory behind:
Estimation of PSDs with „periodograms“

Power spectral density estimation for the noise:

- Estimation schemes using voice activity detection (VAD)
- Tracking of minima of short-term power estimations

The Wiener filter – in the frequency domain

Two alternatives:

1) Schemes with voice activity detection:

$$\widehat{S}_{bb}(\Omega_\mu, n) = \begin{cases} \beta\, \widehat{S}_{bb}(\Omega_\mu, n-1) + (1-\beta)\, \widehat{S}_{yy}(\Omega_\mu, n), & \text{during speech pauses,} \\ \widehat{S}_{bb}(\Omega_\mu, n-1), & \text{else.} \end{cases}$$

2) Tracking of minima of the short-term power (s. lecture 1, p.45) :

1) Smoothing:

$$\overline{S_{yy}(\Omega_\mu, n)} \quad = \quad \beta\, \overline{S_{yy}(\Omega_\mu, n-1)} + (1-\beta)\, \widehat{S}_{yy}(\Omega_\mu, n)$$

2) Minimum value, with a slight increase to avoid a freezing of the estimate:

$$\widehat{S}_{bb}(\Omega_\mu, n) = \min\left\{\overline{S_{yy}(\Omega_\mu, n)}, \widehat{S}_{bb}(\Omega_\mu, n-1)\right\}\, (1+\epsilon) \text{ with: } \epsilon << 1$$

ϵ : determines the tracking capabilities of the estimator

Noise power spectral density estimation

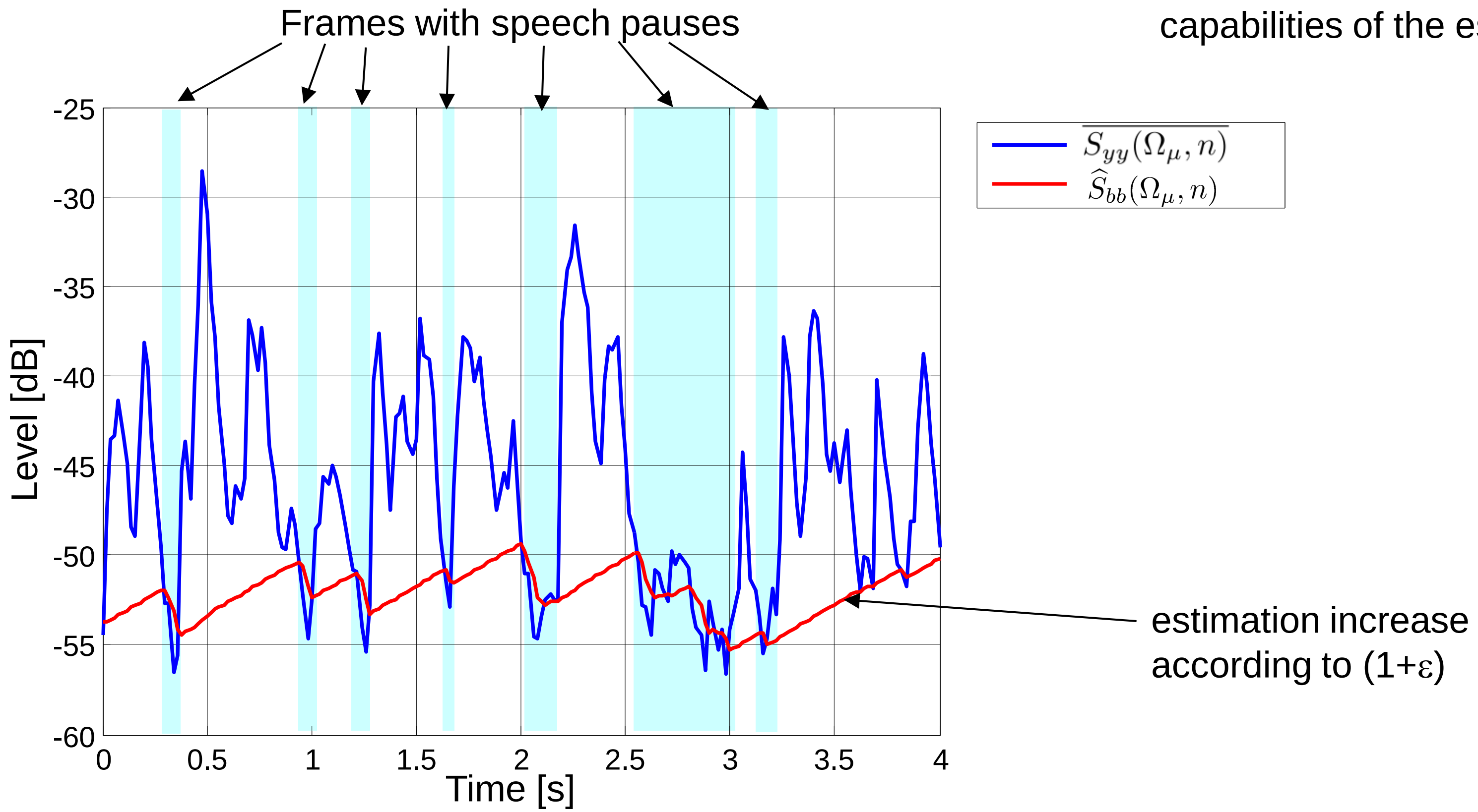
TECHNISCHE UNIVERSITÄT DARMSTADT

2) Tracking of minima of the short-term power:

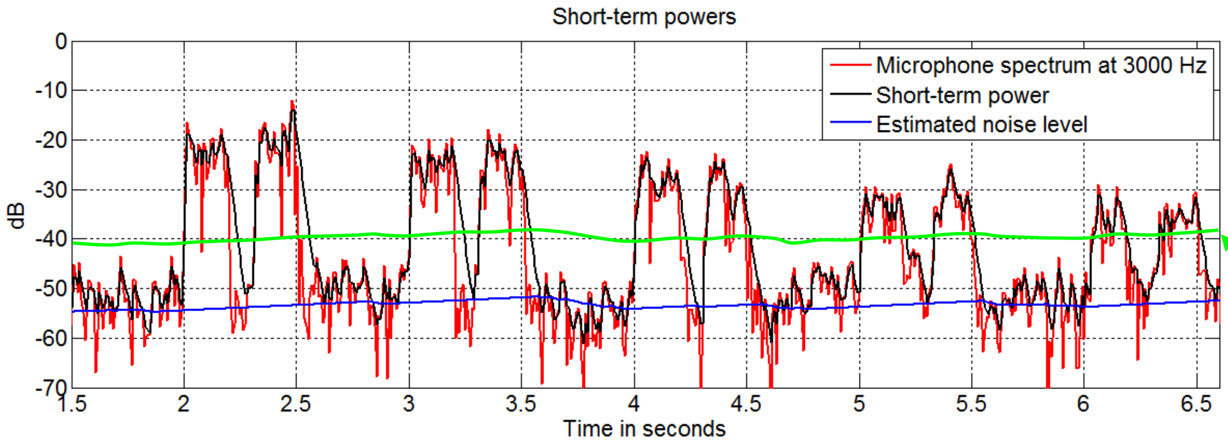
$$\widehat{S}_{bb}(\Omega_\mu, n) = \min\left\{\overline{S_{yy}(\Omega_\mu, n)}, \widehat{S}_{bb}(\Omega_\mu, n-1)\right\}(1+\epsilon)$$

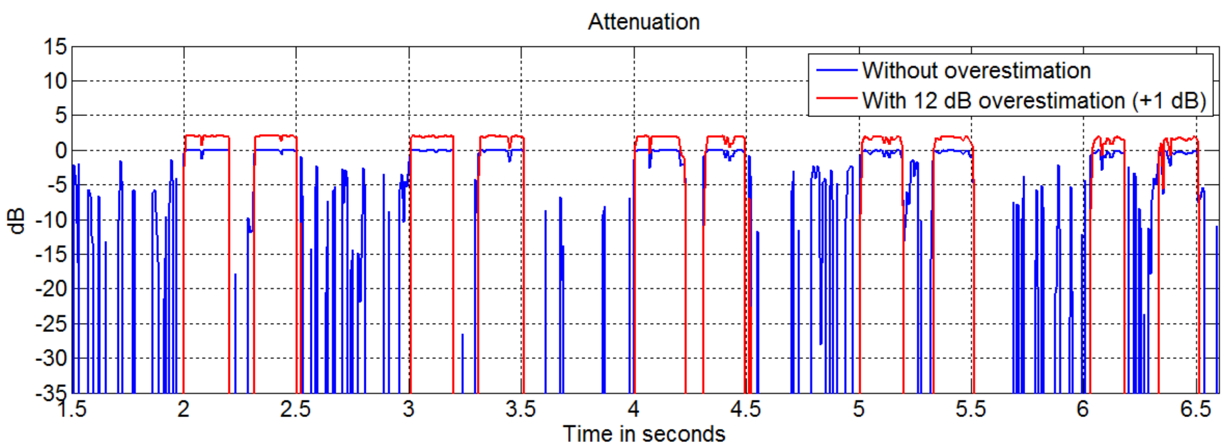
with: $\epsilon << 1$

ϵ : determines the tracking capabilities of the estimator



Noise reduction





: Microphone signal

: Output without over-estimation

: Output with 12 dB over-estimation

Noise reduction

Limiting the maximum attenuation:

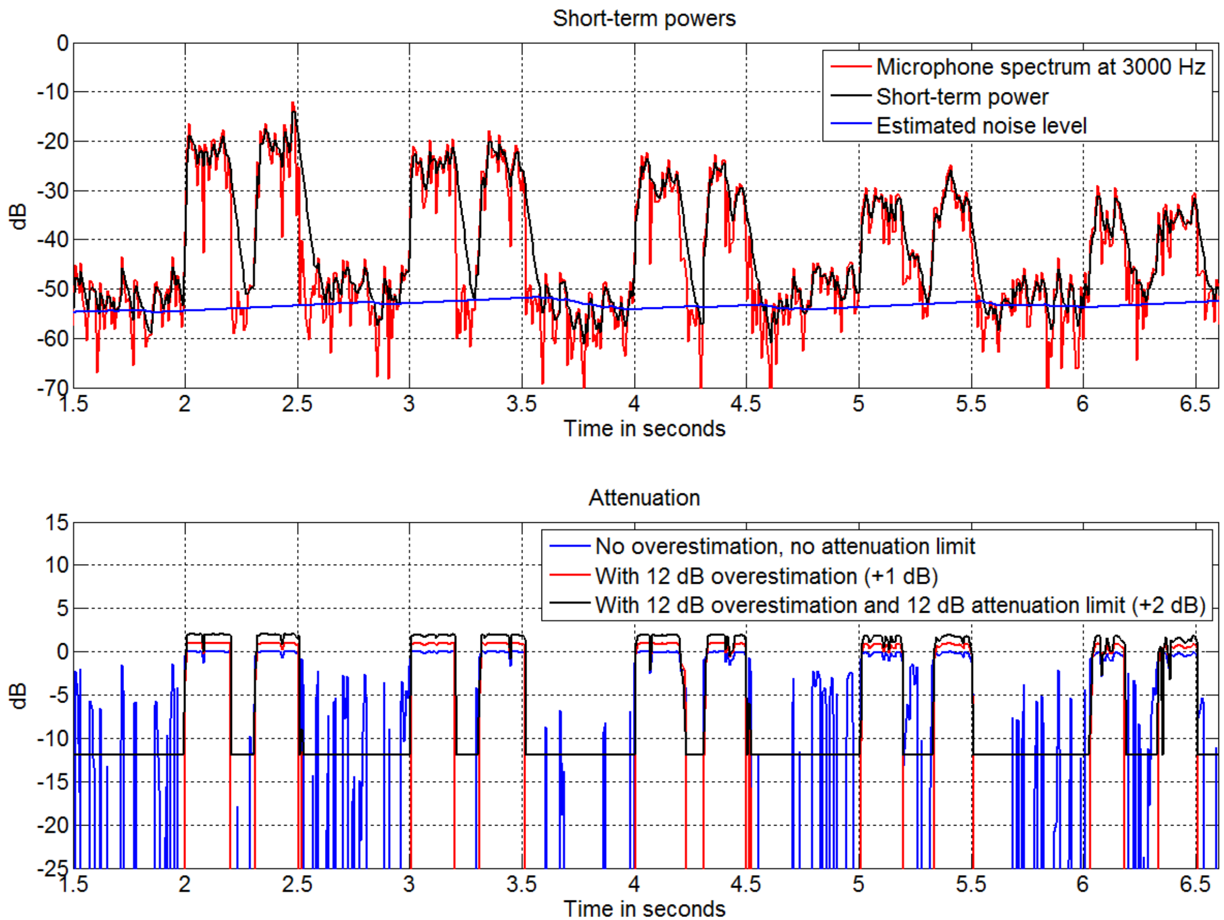
- For several application the original shape of the noise should be preserved (the noise should only be attenuated but not completely removed). This could be achieved by inserting a maximum attenuation:

$$H_{\min}(e^{j\Omega_\mu}, n) \quad = \quad H_{\min}.$$

$$\widehat{H}_{\text{opt}}(e^{j\Omega}, n) \quad = \quad \max\left\{1 - K_{\text{over}} \frac{\widehat{S}_{bb}(\Omega, n)}{\widehat{S}_{yy}(\Omega, n)},\ H_{\min}\right\}$$

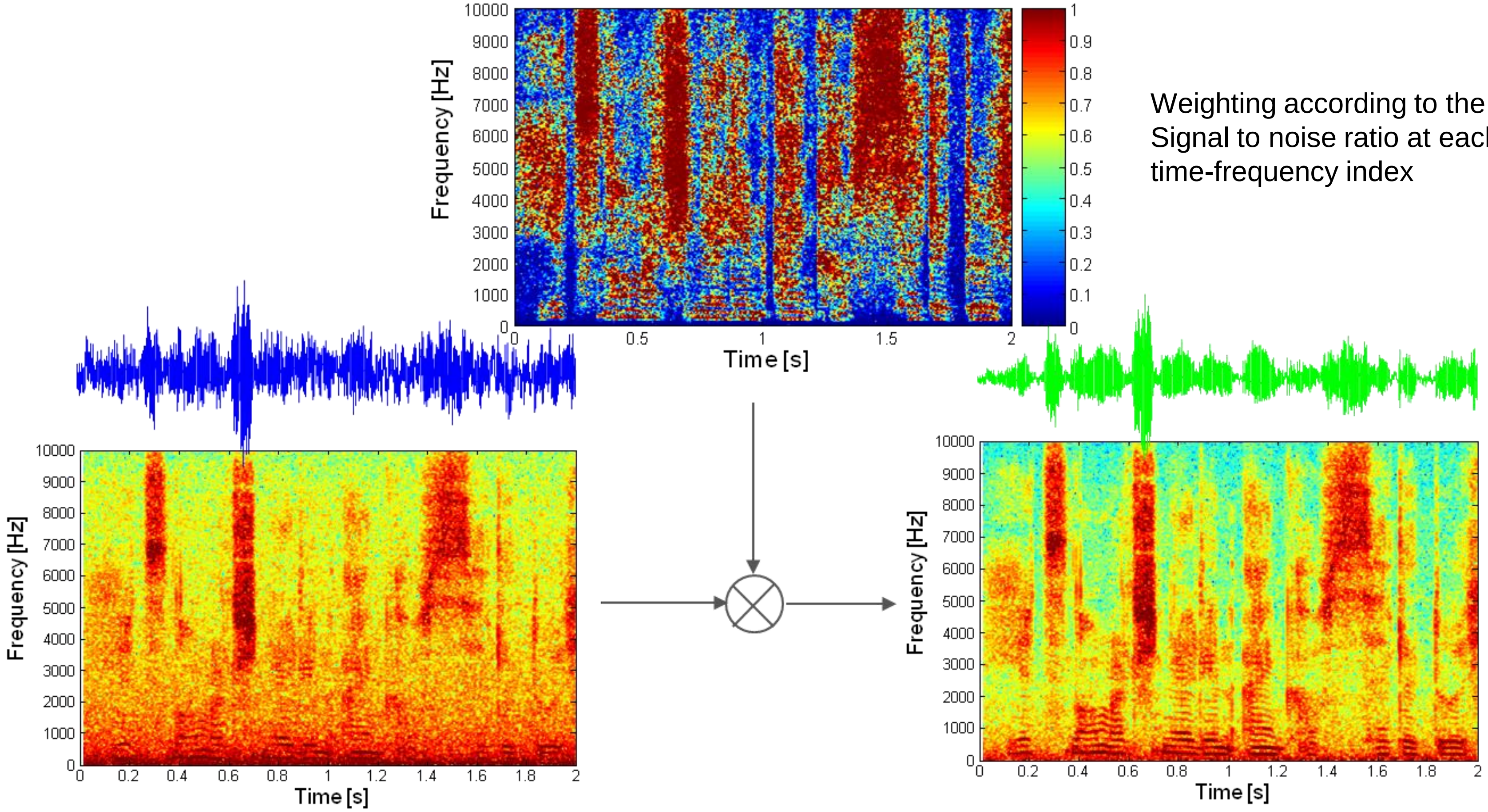
- In addition, this attenuation limits can be varied slowly over time (slightly more attenuation during speech pauses, less attenuation during speech activity).

Noise reduction



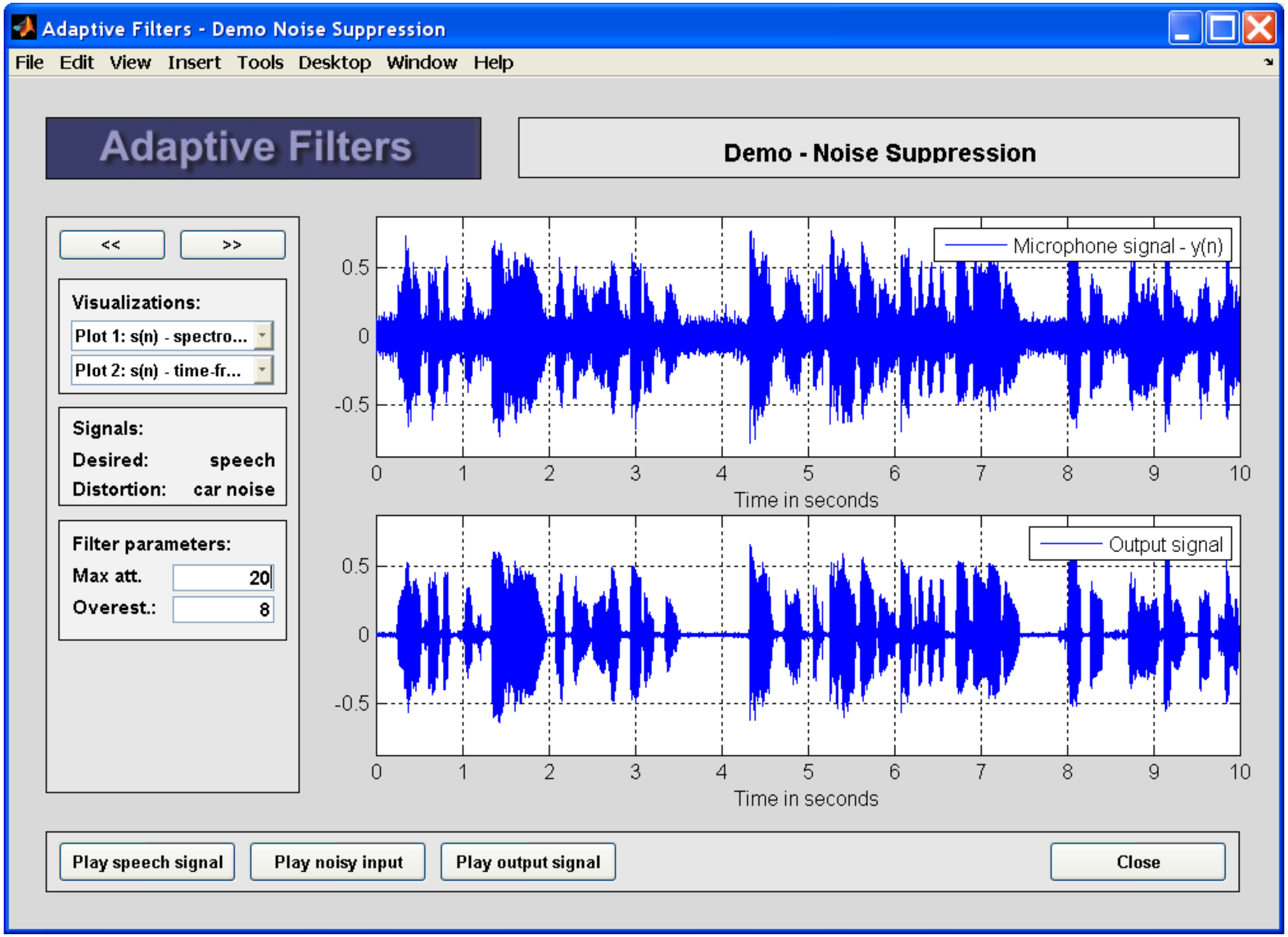
- Microphone signal
- Output without attenuation limit
- Output with attenuation limit

Noise reduction: Spectrogram view

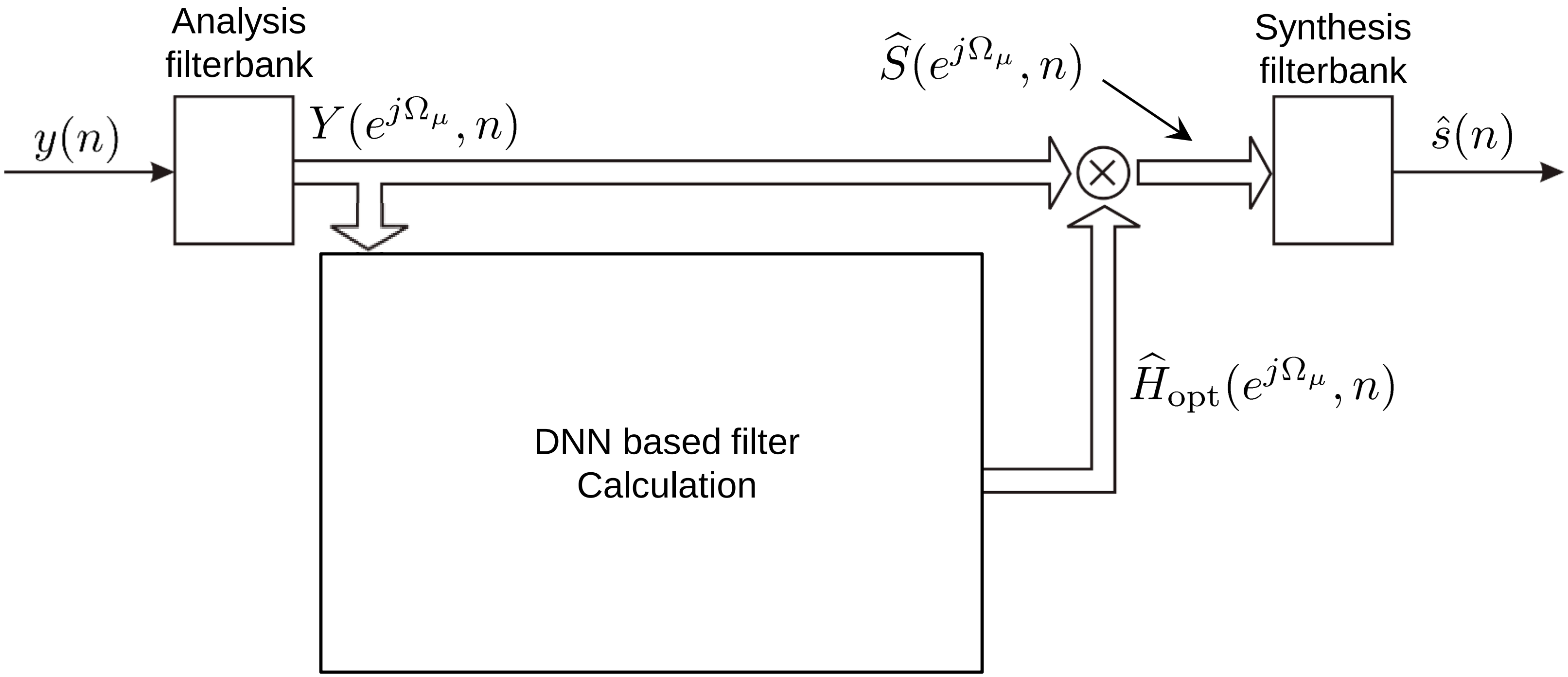


Weighting according to the Signal to noise ratio at each time-frequency index

Noise reduction: Matlab-Demo

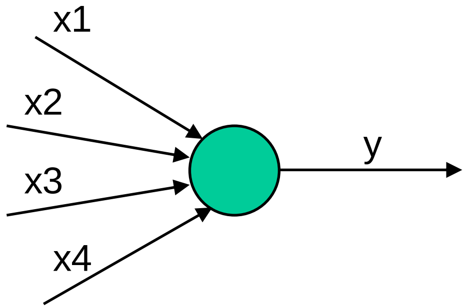


Noise Reduction with DNN



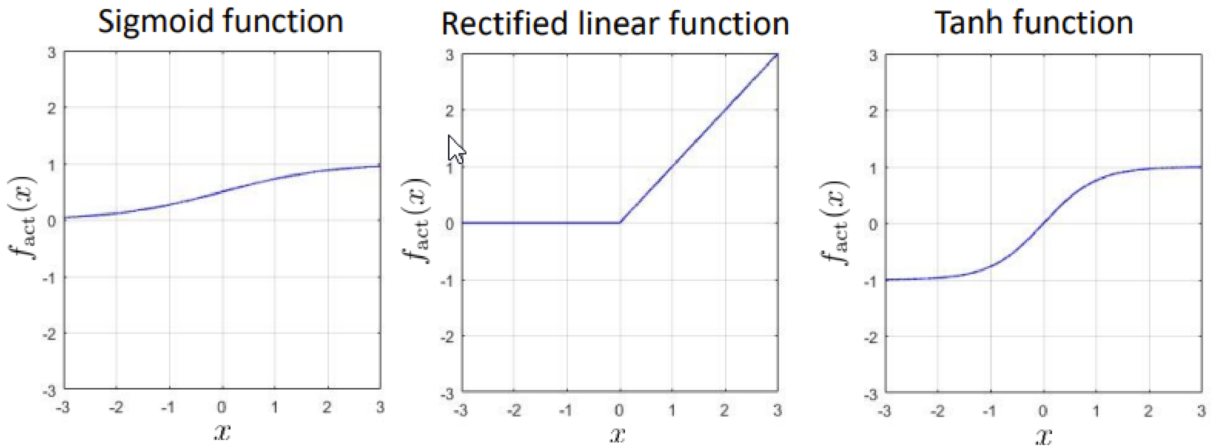
DNN – Basic concept

- The Neuron as basis unit of a DNN:
  Weighted combination a non-linear mapping to the output.

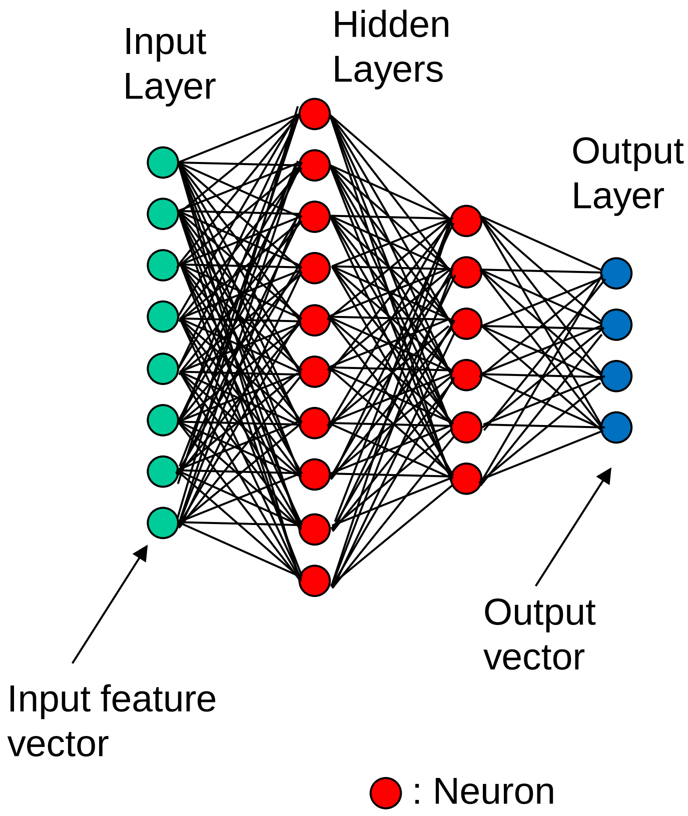


$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

- Different non-linear functions f() may be used.
  Most applies ones: Sigmoid, ReLU (rectified linear unit), Tanh, etc.



- The MultiLayer Perceptron (MLP):



MLP:
Each neuron is connected to all neurons of the previous layer:
=> **Feedforward and unidirectional DNN.**

MLP training:
The weights and the bias values are trained with training data based on stochastic gradient descent.
Overfitting needs to be avoided. E.g. by dropout (randomly removing some neurons during training). This avoids “co-adaptation of neurons”.

DNN – different concepts

- The Convolutive Neural Network (CNN):

Example of the application of a convolutional layer:

$$
\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}
\xrightarrow{\text{Convolution}}
\underset{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}}{*}
\longrightarrow
\begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix}
$$

DNN – different concepts

- Pooling layer (as part of CNNs):

Example of the application of a pooling layer (non-linear „max“ – function):

1	3	5	1
5	2	6	7
3	5	2	3
1	7	1	4

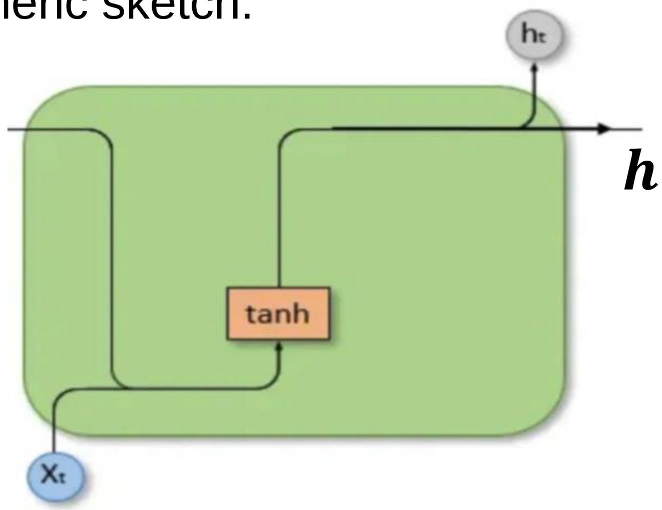
2x2
max
pooling

5	7
7	4

RNN – Recurrent Neural Networks

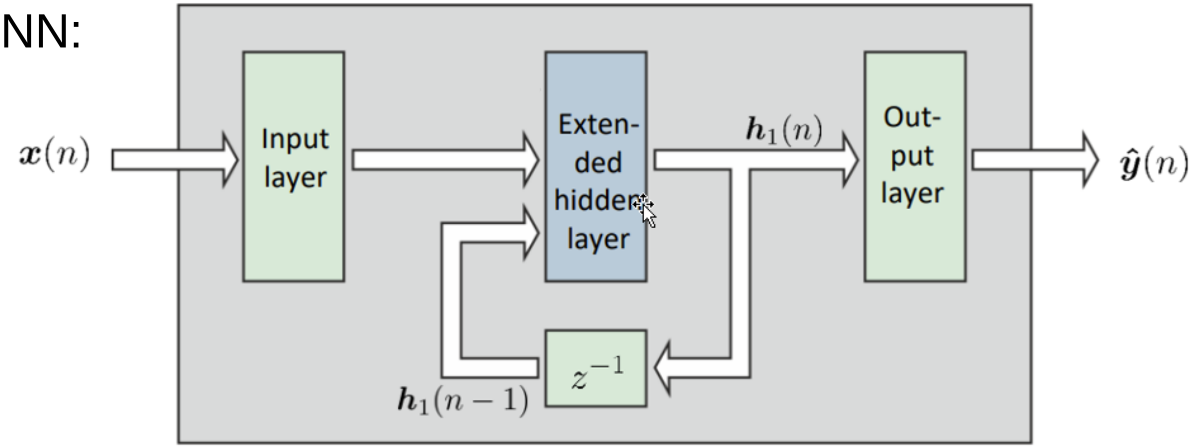
TECHNISCHE UNIVERSITÄT DARMSTADT

- Generic sketch:



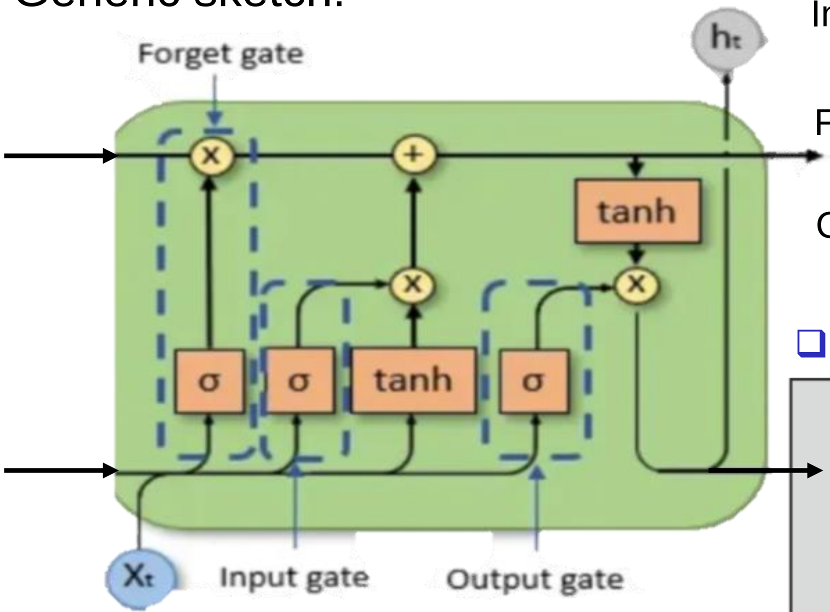
$$\boldsymbol{h}_t(n) = \tanh(\boldsymbol{W}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b})$$

- Integration in a DNN:



LSTM – Long Short-Term Memory Networks

- Generic sketch:



Input gate:

$$\boldsymbol{i}_t(n) = \sigma(\boldsymbol{W}_{\text{in}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{in}})$$

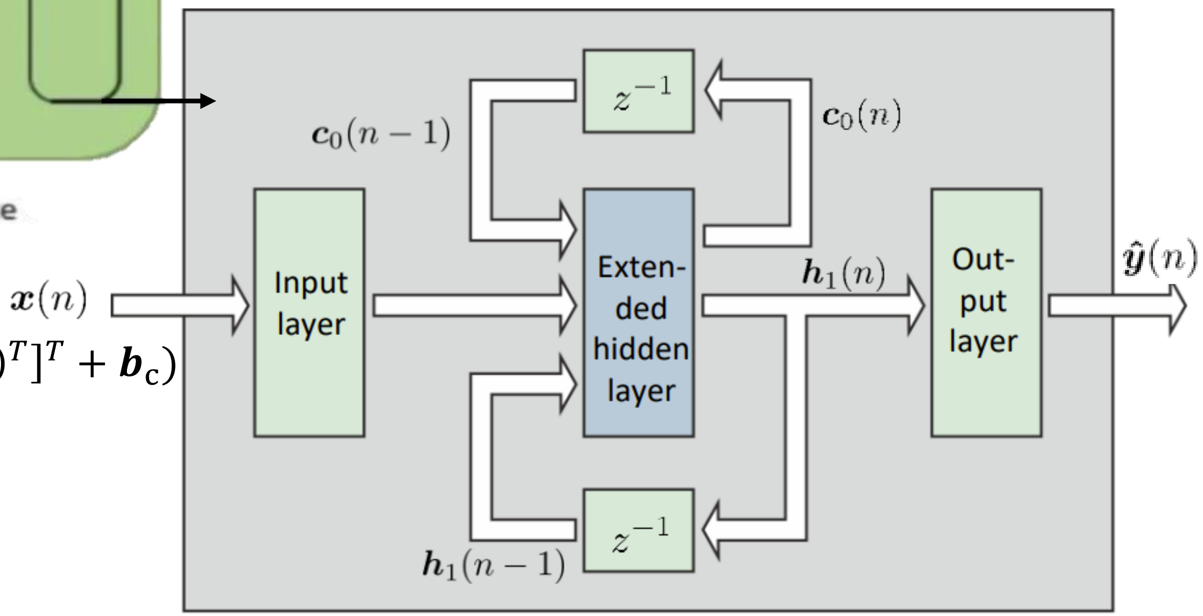
Forget gate:

$$\boldsymbol{f}_t(n) = \sigma(\boldsymbol{W}_{\text{in}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{for}})$$

Output gate:

$$\boldsymbol{o}_t(n) = \sigma(\boldsymbol{W}_{\text{out}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{out}})$$

- Integration in a DNN:



Cell state update:

$$\bar{\boldsymbol{c}}_t(n) = \tanh(\boldsymbol{W}_{\text{c}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{c}})$$

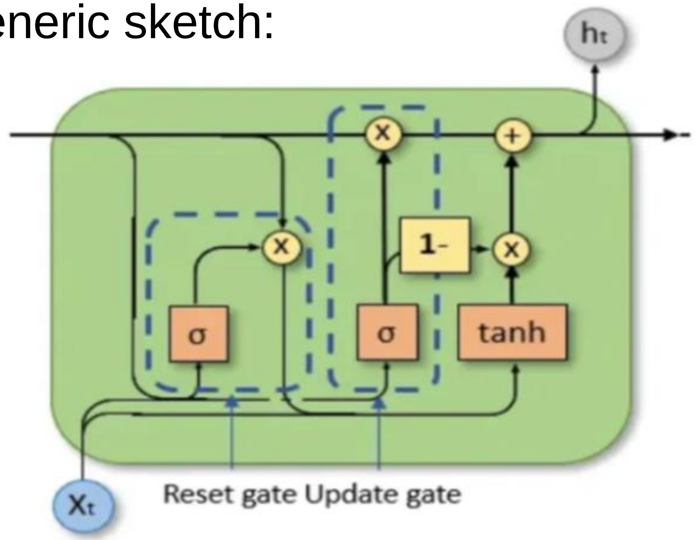
$$\boldsymbol{c}_t(n) = \text{diag}\{\boldsymbol{f}_t(n)\}c_t(n-1) + \text{diag}\{\boldsymbol{i}_t(n)\})\ \bar{\boldsymbol{c}}_t(n)$$

Hidden state update:

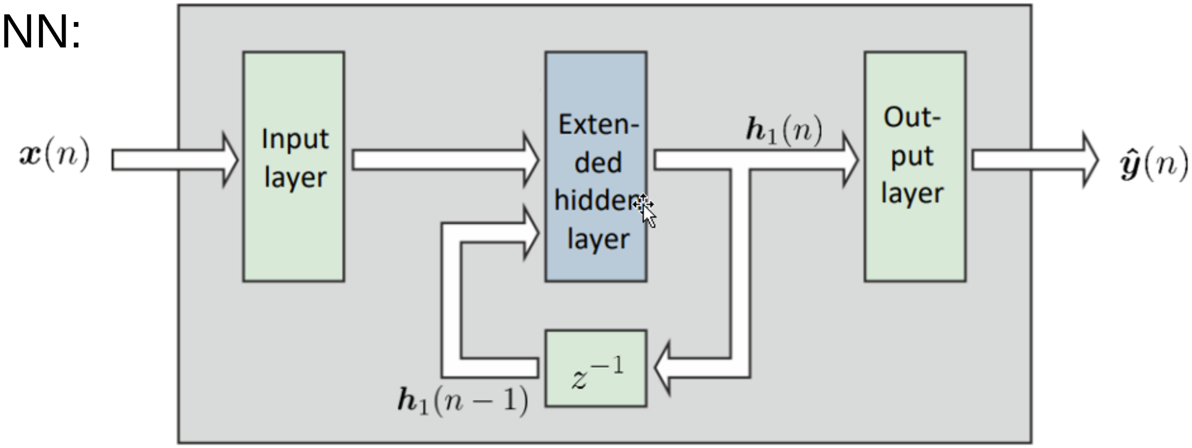
$$\boldsymbol{h}_t(n) = \text{diag}\{\tanh(\boldsymbol{c}_t(n)\}\boldsymbol{o}_t(n)$$

GRU – Gated Recurrent Units

- Generic sketch:



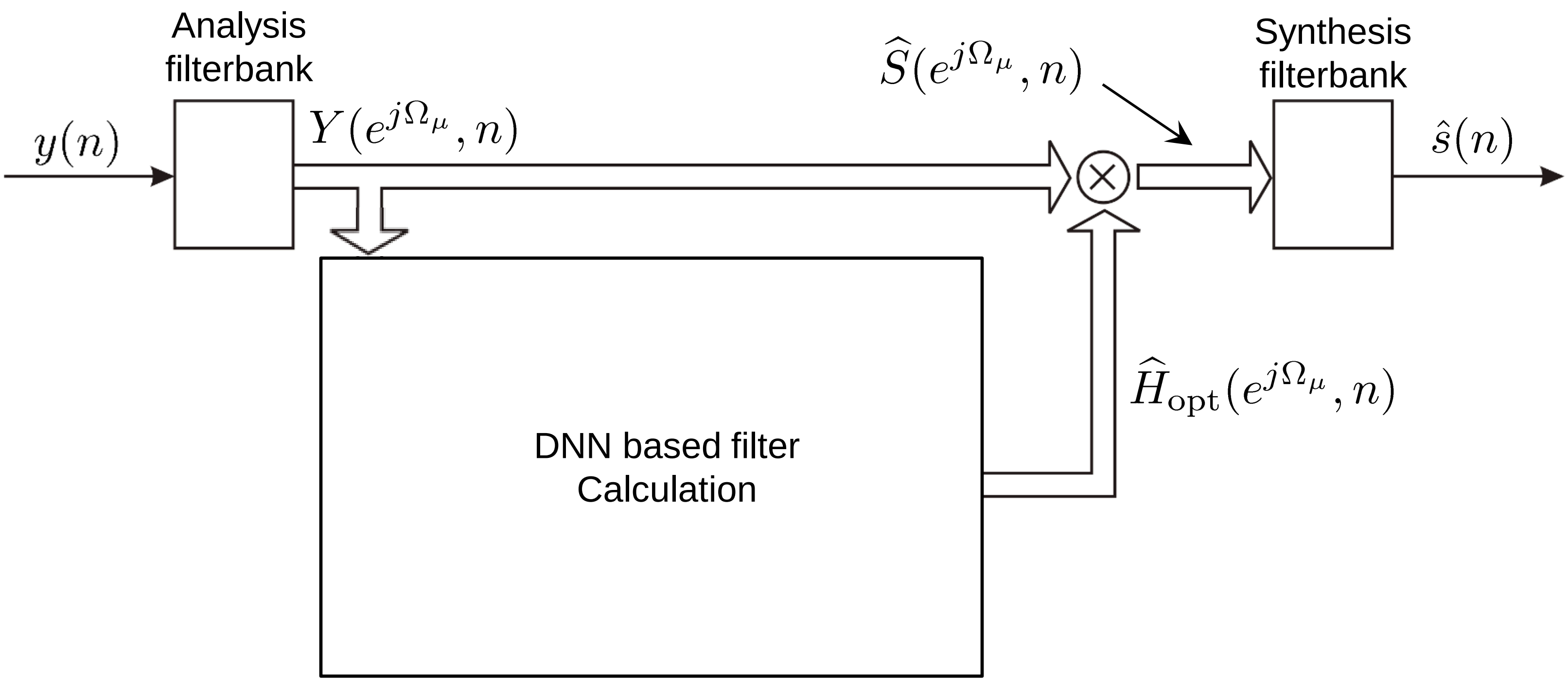
- Integration in a DNN:



Noise Reduction with DNN

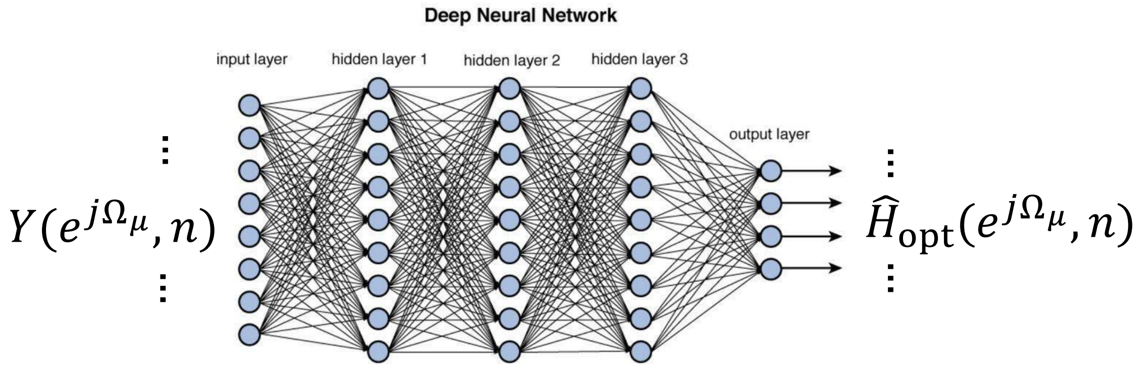
TECHNISCHE UNIVERSITÄT DARMSTADT

- Noise reduction gains based on DNN optimization:

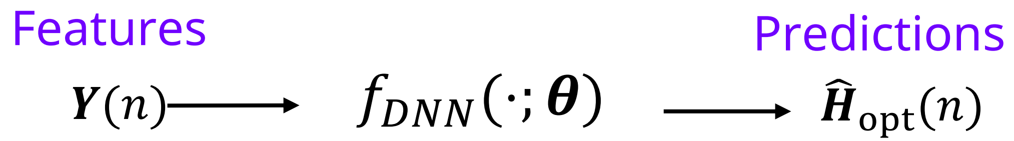


Noise Reduction with DNN

- Example of a DNN mapping a feature vector to a noise reduction gain vector:



- Used feature vector: Noisy spectrum



Feature vector: $\boldsymbol{Y}(n)$
DNN parameter vector: $\boldsymbol{\theta}$
DNN architecture: $f_{DNN}(\cdot\,;\cdot)$
Predictions: $\widehat{\boldsymbol{H}}_{\text{opt}}(n)$

Optimizing cost function

TECHNISCHE UNIVERSITÄT DARMSTADT

- Cost function:

essa função custo pode n ser a melhor para representar a audição humana

$$C(\boldsymbol{\theta}) = \sum_{\mu,n} \mathbb{E}\left\{ \left|S(e^{j\Omega_\mu}, n) \ - \widehat{H}(e^{j\Omega_\mu}, n) Y(e^{j\Omega_\mu}, n)\ \right|^2 \right\}$$

for optimizing the following gain vector:

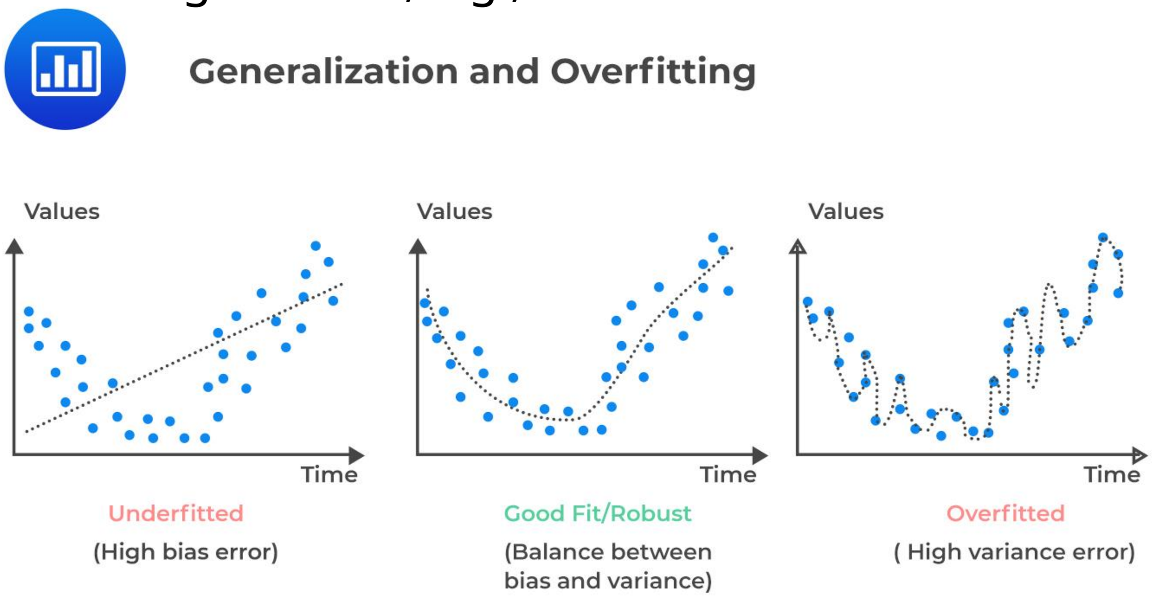
$$\widehat{\boldsymbol{H}}_{\text{opt}}(n) = \left(\widehat{H}_{\text{opt}}(e^{j\Omega_0}, n), \ldots, \widehat{H}_{\text{opt}}(e^{j\Omega_M}, n)\right)^T$$

$$= f_{DNN}(\boldsymbol{Y}(n); \boldsymbol{\theta})$$

Data Modeling Accuracy

Motivation:

- DNNs provide great modeling capability due to large number of parameters
- Risk of overfitting: "DNN memorizes the training data and does not learn its structure“
- Use large training data set, e.g., for a VAD: 150 h



Realization

- Replace the cost function:

$$C(\boldsymbol{\theta}) = \sum_{\mu,n} \mathbb{E}\left\{ \left| S(e^{j\Omega_\mu}, n) - \widehat{H}(e^{j\Omega_\mu}, n) Y(e^{j\Omega_\mu}, n) \right|^2 \right\}$$

by an arithmetic average of the training samples:

$$C(\boldsymbol{\theta}) = \sum_{\mu,n} \frac{1}{L} \sum_{l} \left| S^{(l)}(e^{j\Omega_\mu}, n) - \widehat{H}^{(l)}(e^{j\Omega_\mu}, n) Y^{(l)}(e^{j\Omega_\mu}, n) \right|^2$$

$$\widehat{\boldsymbol{H}}_{\text{opt}}^{(l)}(n) = f_{DNN}\left(\boldsymbol{Y}^{(l)}(n); \boldsymbol{\theta}\right)$$

Iterative optimization

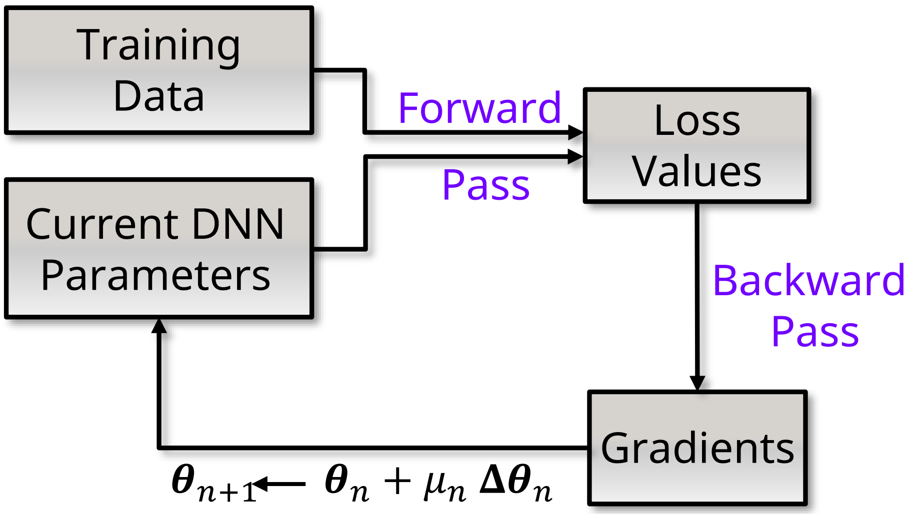
- According to classic adaptive rulesets:

$$\boldsymbol{\theta}_{n+1} \longleftarrow \boldsymbol{\theta}_n + \mu_n \, \boldsymbol{\Delta\theta}_n$$

with:
- iteration index n
- batch gradient $\boldsymbol{\Delta\theta}_n$
- step-size μ_n

- Parameter optimization by iterative gradient descent

- Gradients are computed in backward pass



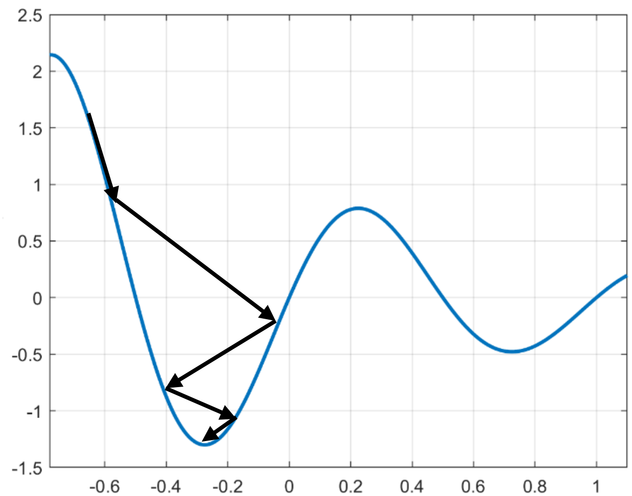
Iterative optimization

- Suitable selection of the step-size important:

$$\boldsymbol{\theta}_{n+1} \longleftarrow \boldsymbol{\theta}_n + \mu_n\, \boldsymbol{\Delta\theta}_n$$

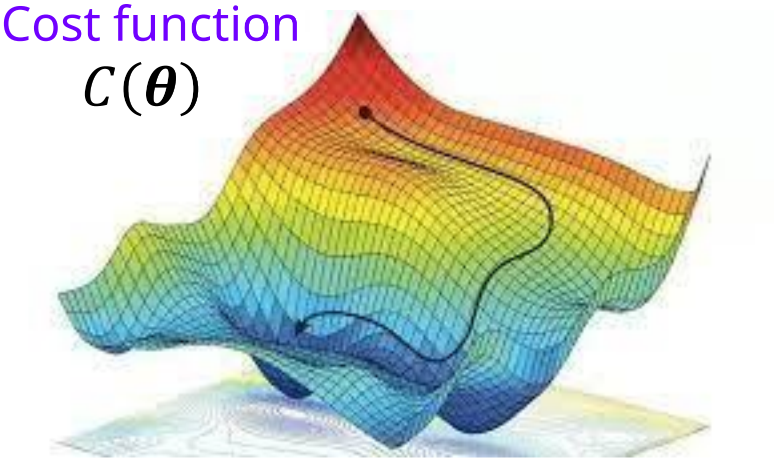
with:
- iteration index n
- batch gradient $\boldsymbol{\Delta\theta}_n$
- step-size μ_n

Cost function $C(\theta)$



Scalar DNN parameter θ

Cost function $C(\boldsymbol{\theta})$



2-dim DNN parameter vector $\boldsymbol{\theta}$

Iterative optimization

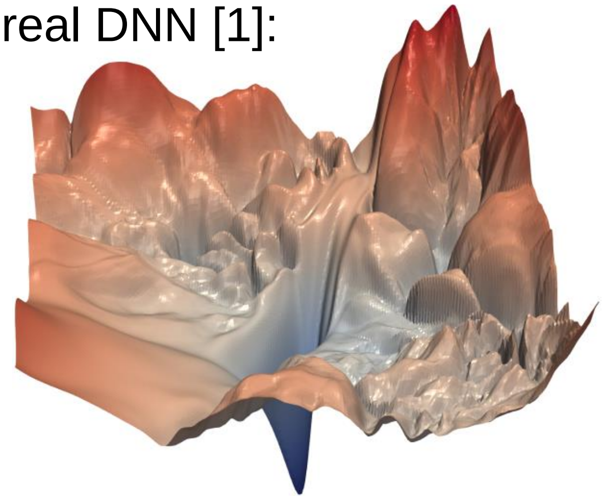
- Suitable selection of the step-size important:

$$\boldsymbol{\theta}_{n+1} \longleftarrow \boldsymbol{\theta}_n + \mu_n \, \boldsymbol{\Delta\theta}_n$$

with:
- iteration index n
- batch gradient $\boldsymbol{\Delta\theta}_n$
- step-size μ_n

- Cost function of a real DNN [1]:

Cost function $C(\boldsymbol{\theta})$



2-dim DNN parameter vector $\boldsymbol{\theta}$

[1]: Li et al., *Visualizing the Loss Landscape of Neural Nets*, NIPS 2018.

Mini batch approximation

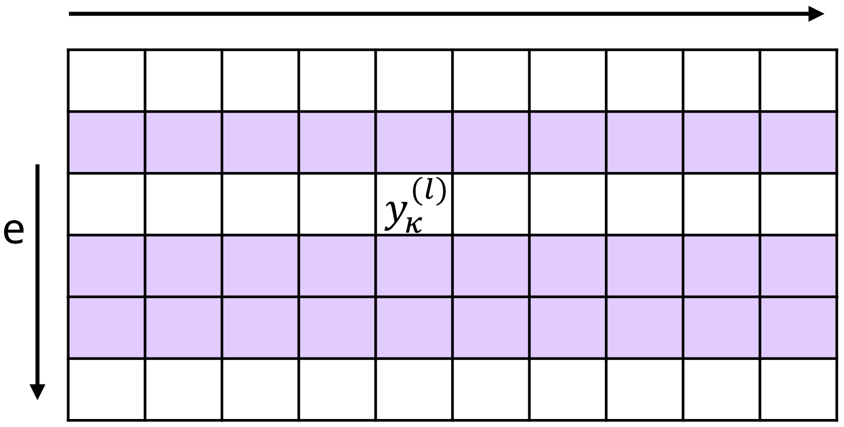
- **Problem:** Entire training data set does usually not fit in memory

- **Solution**: Sequential updating with randomly-sampled small subsets of the complete training data set (=mini-batch)

Training data set with L sequences

Time index κ

Sequence index l



Batch size: 3
First index typically corresponds to (mini-)batch dimension

Example of a simple DNN architecture for Noise Reduction in pytorch

TECHNISCHE UNIVERSITÄT DARMSTADT

- Architecture definition:

```
# dnn architecture
self.encoder = torch.nn.Sequential(
    torch.nn.Linear(self.feat_dim, self.hidden_size_gru),
    torch.nn.ReLU()
)

self.gru_inf = torch.nn.GRU(self.hidden_size_gru, self.hidden_size_gru,
                            self.num_layers_gru, dropout=self.dropout_gru,
                            batch_first=True)

self.decoder = torch.nn.Sequential(
    torch.nn.Linear(self.hidden_size_gru, self.mask_freq_size),
    torch.nn.Sigmoid()
)
```

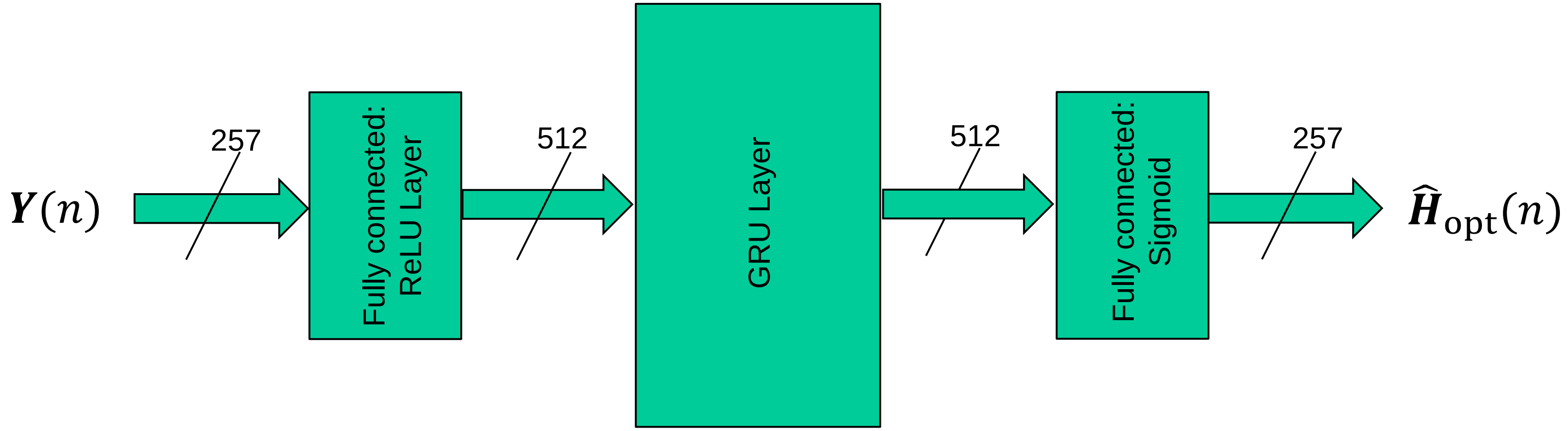
- Parameters:

```
nfft: 512                                             # DFT length
frameshift: 256
hidden_size_gru: 512

self.feat_dim = self.dnn_cfg['nfft'] // 2 + 1         # => 257
self.hidden_size_gru = self.dnn_cfg['hidden_size_gru’] # => 512
self.mask_freq_size = self.dnn_cfg['nfft'] // 2 + 1   # => 257
```

Example of a simple DNN architecture for Noise Reduction in pytorch

- Simple noise reduction based on a combination of fully connected and GRU layers:



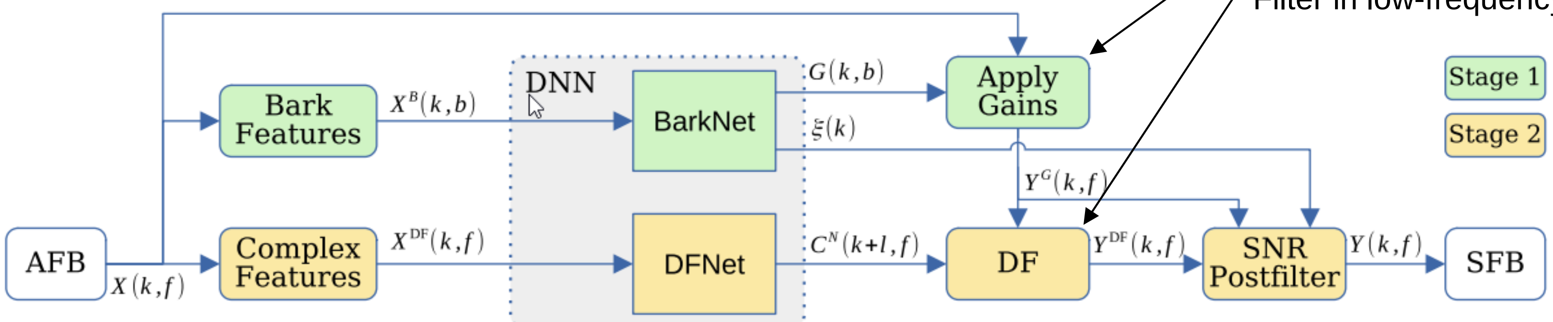
State-of-the-art Noise Reduction with DNNs [1]: Combination of complex-valued gains and sub-band filters

TECHNISCHE UNIVERSITÄT DARMSTADT

- Processing structure:

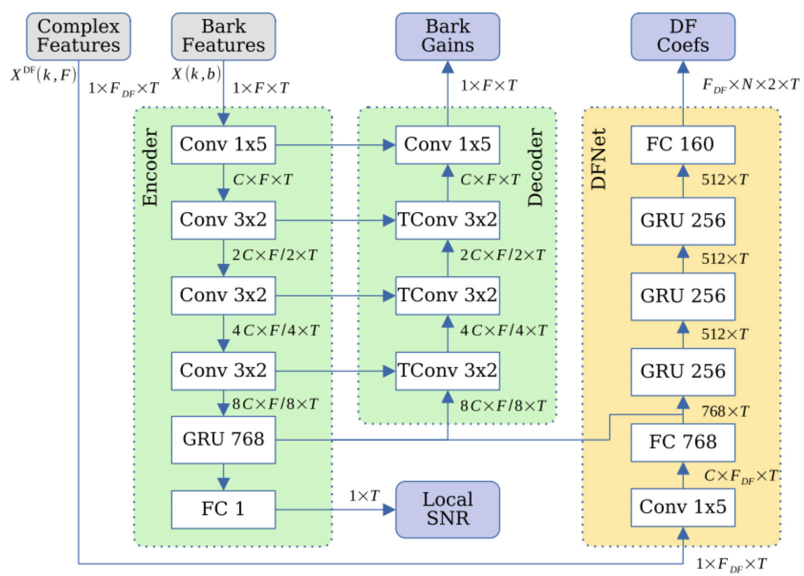
Gain in sub-bands

Filter in low-frequency sub-bands



- Design of the corresponding DNN structure:

A combination of
- fully connected (FC)
- convolutional (Conv) and
- GRU layers



[1] H. Schröter, T. Rosenkranz, A. -N. Escalante and A. Maier, "Low Latency Speech Enhancement for Hearing Aids Using Deep Filtering," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2716-2728, 2022

Dereverberation

- Speech recordings in large rooms sound reverberant, and this the larger the distance is between the signal source and the recording microphone.

- This provokes the following effects:
  - The recorded sound quality is perceived as low.
  - For large reverberation even the speech intelligibility may be reduced. Here, first hearing impaired people are concerned (=> demands for dereverberation techniques in hearing aids)
  - Automatic speech recognition systems tend to fail in reverberant environments.

- Reverberation may also contribute to a good and natural speech quality. Early reflections (~ 30 – 50 ms) are typically desired.

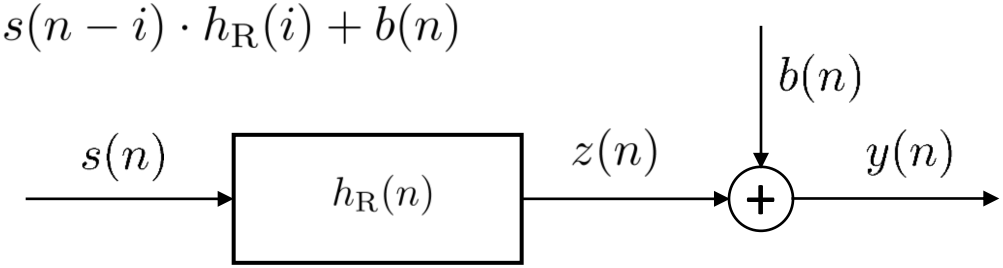
- Ideally the room impulse response is known and an inverse filtering is applied. This approach, however, has mainly a theoretical importance.

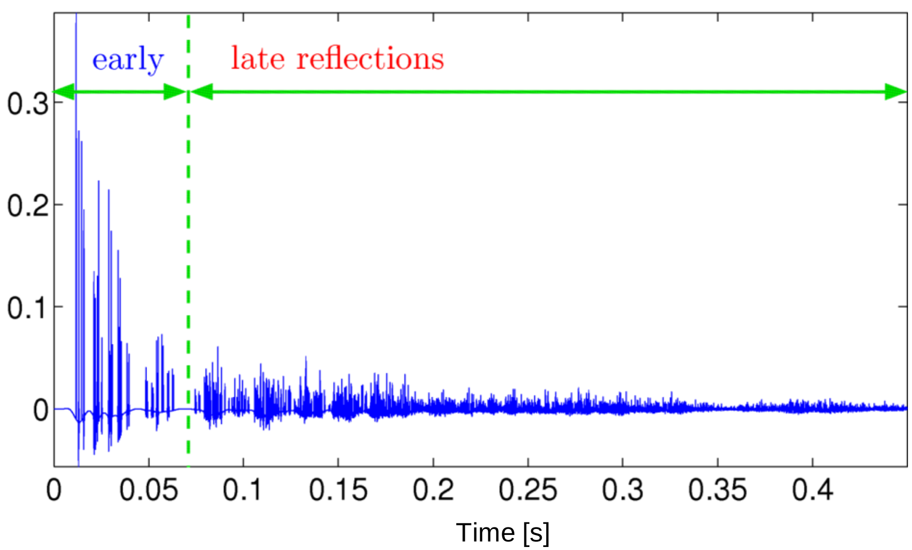
- The procedure sketched here tries to apply a Wiener filter approach comparable to the noise reduction.

Dereverberation

- Convolution with room impulse response + additive noise

$$y(n) = \sum_{i=0}^{L_R-1} s(n-i) \cdot h_{\mathrm{R}}(i) + b(n)$$

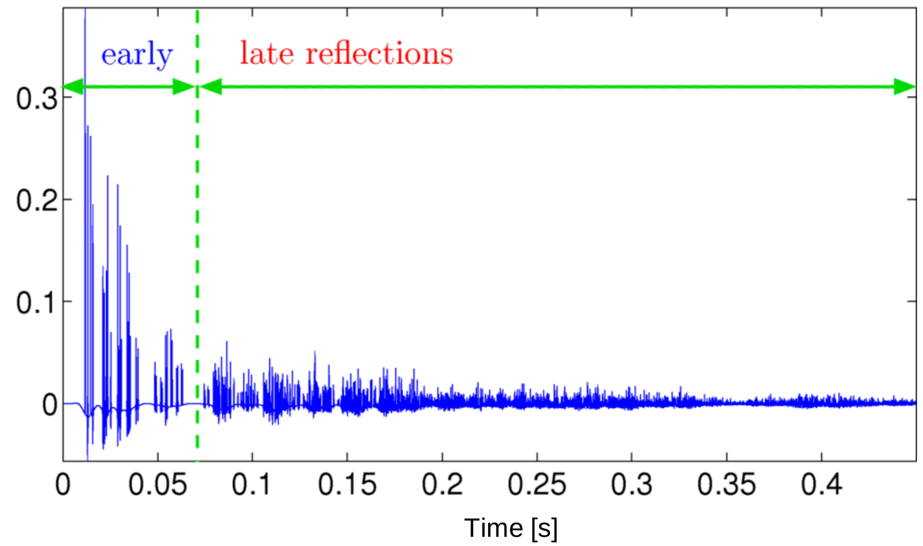




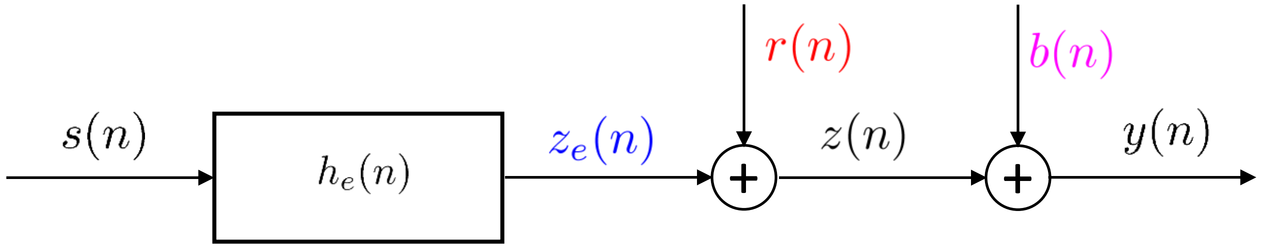
- Early reverberant components are desired and contribute to a natural sound and even to a good speech intelligibility.
- Late reverberant components should be cancelled

Dereverberation

- Model late reflections as additive noise component

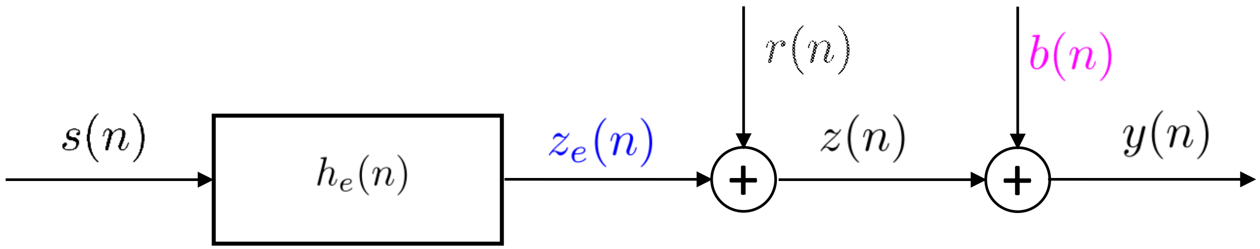


$$y(n) = \underbrace{\sum_{i=0}^{L_e-1} s(n-i)\cdot h_e(i)}_{z_e(n):\ \text{early reverberant speech}} + \underbrace{\sum_{i=L_e}^{L_R-1} s(n-i)\cdot h_l(i)}_{r(n):\ \text{late reverberant speech}} + \underbrace{b(n)}_{\text{noise}}$$



Dereverberation

- Model late reflections as additive noise component:



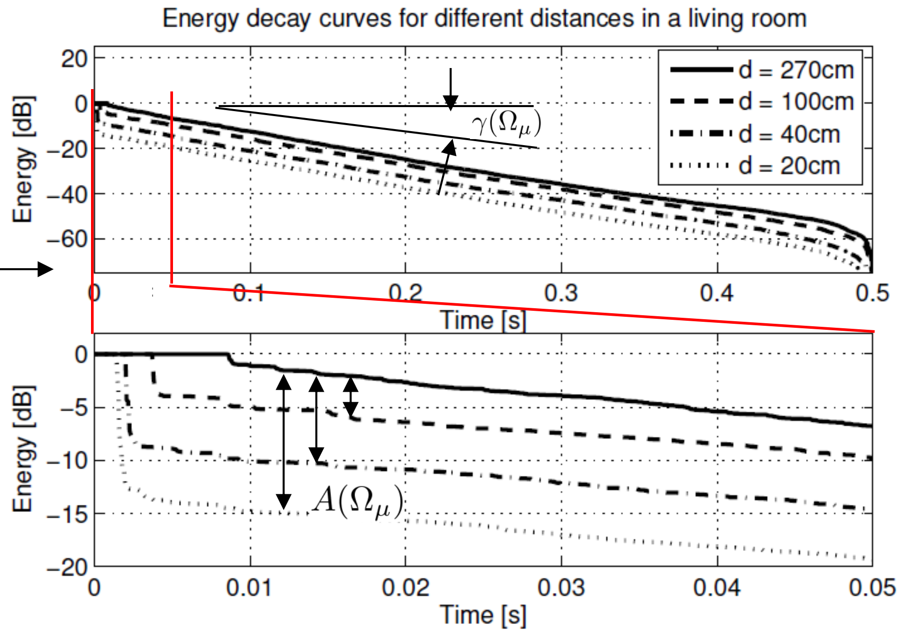
- Incorporation in the Wiener formula:

$$\widehat{S}_{bb}(\Omega_\mu, n) \quad \longrightarrow \quad \widehat{S}_{bb}(\Omega_\mu, n) + \widehat{S}_{rr}(\Omega_\mu, n)$$

$$\widehat{H}_{\text{opt}}(e^{j\Omega_\mu}, n) = \max\left\{ H_{\min},\ 1 - \frac{K_{bb,\text{over}}\,\widehat{S}_{bb}(\Omega_\mu, n) + K_{rr,\text{over}}\,\widehat{S}_{rr}(\Omega_\mu, n)}{\widehat{S}_{yy}(\Omega_\mu, n)} \right\}.$$

Dereverberation

- Estimation of the PSD of the reverberant signal.
- Two main properties which determine the reverberant signal:
  - Direct-to-reverberant ratio which depends on the distance *d* between the audio source and the audio sink:
  - Decay parameter: $\gamma(\Omega_\mu)$
  - Decay of the reverberation energy over time normalized by overall reverberation energy.



Dereverberation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Estimation of the PSD of the reverberant signal.
  - Disturbing reverberation after L_e samples considering the attenuation of the direct path $A(\Omega_\mu)$and the decay parameter $\gamma(\Omega_\mu)$:

$$S_{rr}(\Omega_\mu, n) \approx \sum_{k=L_e}^{\infty} S_{ss}(\Omega_\mu, n-k)\, A(\Omega_\mu)\, e^{-\gamma(\Omega_\mu)\, k}$$

  - Typically, the clean speech is not available
    => take the noisy spectrum

$$\widehat{S}_{ss}(\Omega_\mu, n) \approx |Y(e^{j\,\Omega_\mu}, n)|^2$$

    => leads to an overestimation of the reverberation in noisy environments.

  - Summed estimation:

$$\widehat{S}_{rr}(\Omega_\mu, n) = \sum_{k=L_e}^{\infty} |Y(e^{j\,\Omega_\mu}, n-k)|^2\, A(\Omega_\mu)\, e^{-\gamma(\Omega_\mu)\, k}$$

  - Recursive estimation:

$$\widehat{S}_{rr}(\Omega_\mu, n) = \widehat{S}_{rr}(\Omega_\mu, n-1)\, e^{-\gamma(\Omega_\mu)} + |Y(e^{j\,\Omega_\mu}, n-L_e)|^2\, A(\Omega_\mu)\, e^{-\gamma(\Omega_\mu)\, L_e}$$

- Estimation of the of the direct-to-reverberant ratio and the decay parameter:
  - Rather complicated procedures.
  - A simple approach is sketched in [2]:
  - 1) Determine **decay rate** (assumption: T_60 or T_40 etc. time is known, s. next slide for its definition):

$$10\,\log_{10}\left(e^{-\gamma(\Omega_\mu)\,T_{60}\,f_s}\right) = -60\,\mathrm{dB} \qquad \Rightarrow \qquad \gamma(\Omega_\mu) = \frac{6\,\ln(10)}{T_{60}\,f_s}$$

  - 2) Determine the **direct-to-reverberant ratio**:

$$\tilde{S}_{rr}(\Omega_\mu, n) = \tilde{S}_{rr}(\Omega_\mu, n-1)\,e^{-\gamma(\Omega_\mu)} + |Y(e^{j\,\Omega_\mu}, n - L_e)|^2\,e^{-\gamma(\Omega_\mu)\,L_e}$$

$$\widehat{Q}_A(\Omega_\mu, n) = \frac{|Y(e^{j\,\Omega_\mu}, n)|^2}{\tilde{S}_{rr}(\Omega_\mu, n)}$$

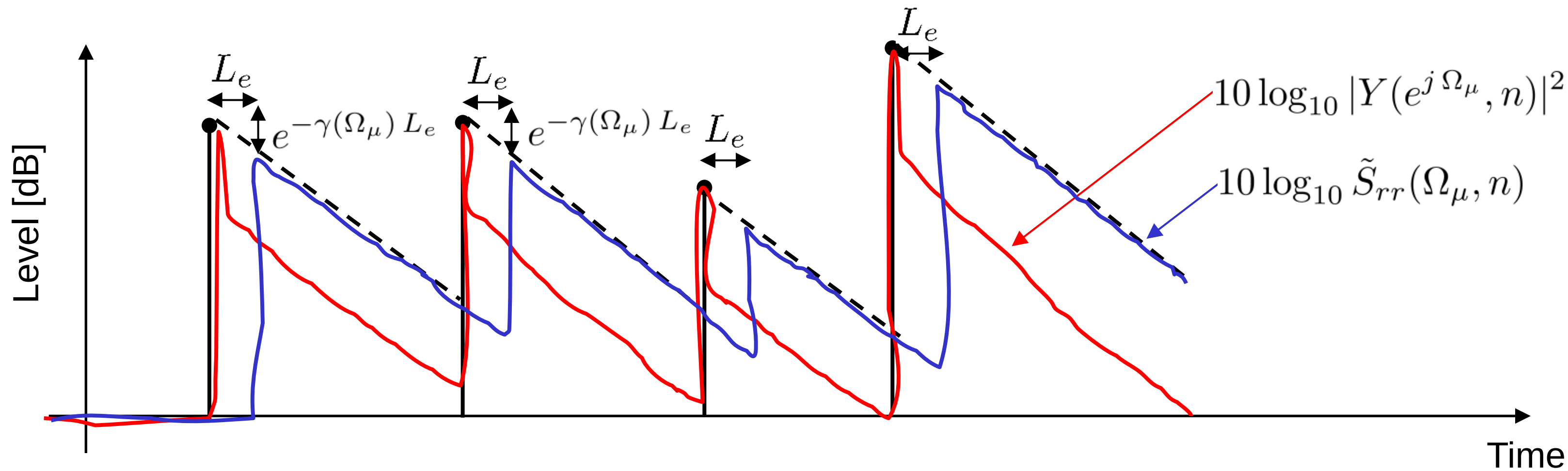
Minimum search in speech pauses:

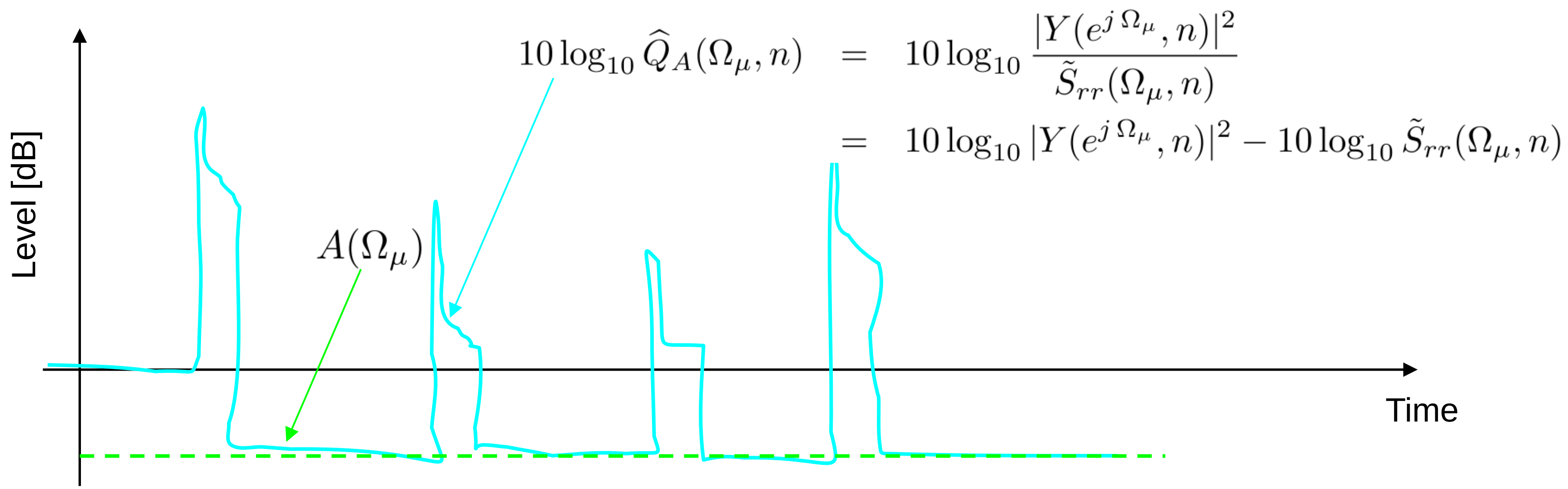
$$\widehat{A}(\Omega_\mu, n) = \min\left\{(1+\epsilon)\,\widehat{A}(\Omega_\mu, n-1), \widehat{Q}_A(\Omega_\mu, n)\right\}$$

$$\boxed{\Rightarrow \quad \widehat{S}_{rr}(\Omega_\mu, n) = \widehat{A}(\Omega_\mu, n)\,\tilde{S}_{rr}(\Omega_\mu, n)}$$

[2]: M. Buck, A. Wolf: Model Based Dereverberation for Speech Recognition: ITG-Fachtagung Sprachkommunikation, Aachen, Oct. 2008

Example with impulses as excitation





Repetition (Lecture 1, page 23):

- Reverberation after a time t = N*Ts

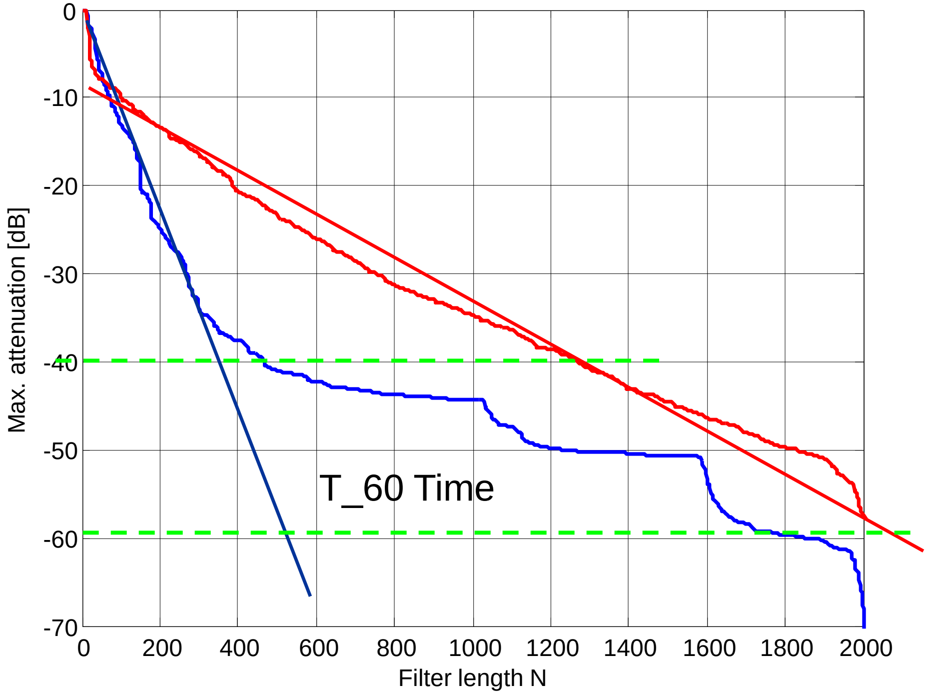
$$att_{max} = \frac{\sigma_e^2(N)}{\sigma_y^2} = \frac{\sum_{v=N}^{\infty} h_v^2}{\sum_{v=0}^{\infty} h_v^2}$$

40 dB attenuation:
N = 450 for a car cabin (example)
N = 1250 for an office room (example)

- Determine reverberation time:
  T_60 is a value which typically characterizes the reverberation:

  - Set att_max to 60 dB and calculate corresponding N, or t.

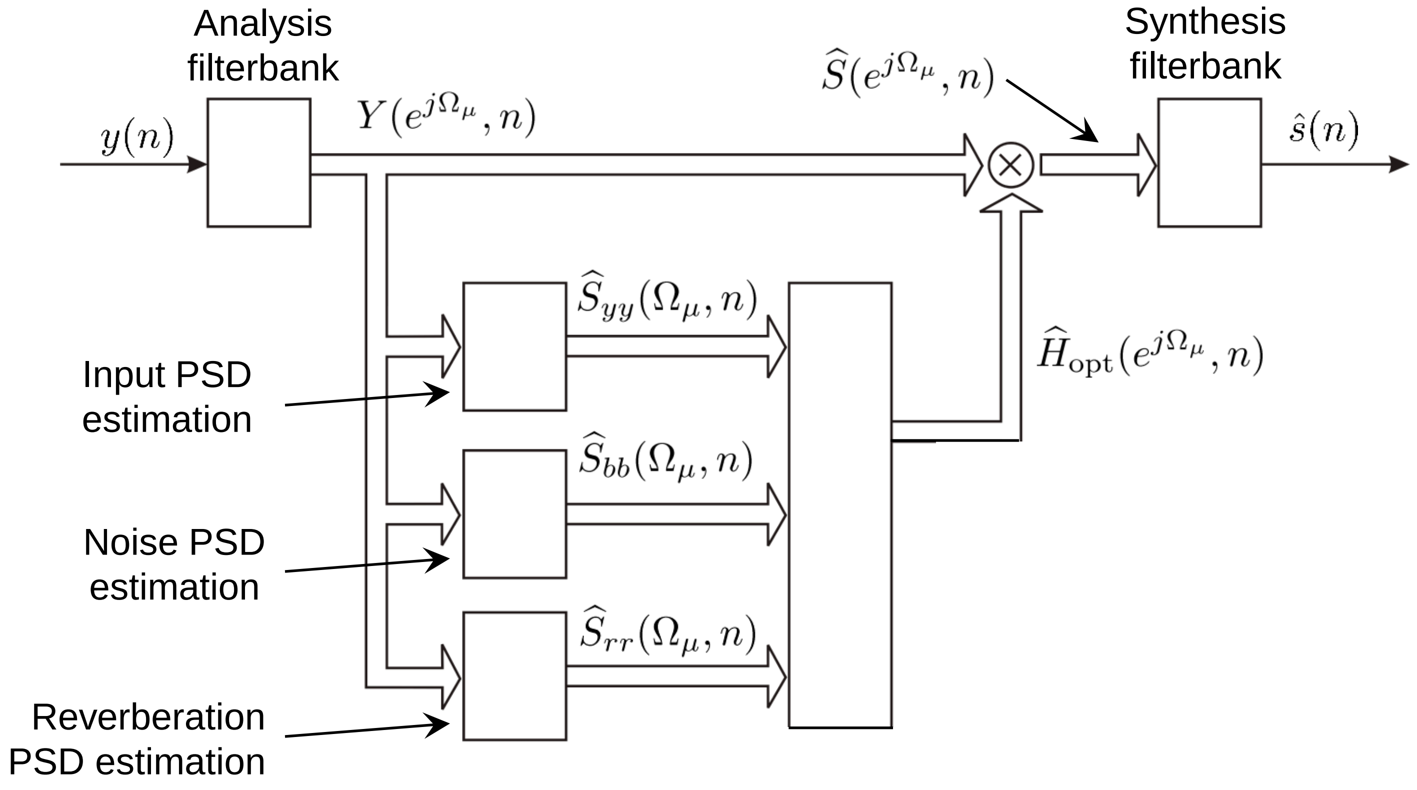
Attenuation in dependence of N



red: office room
blue: car cabin

Dereverberation

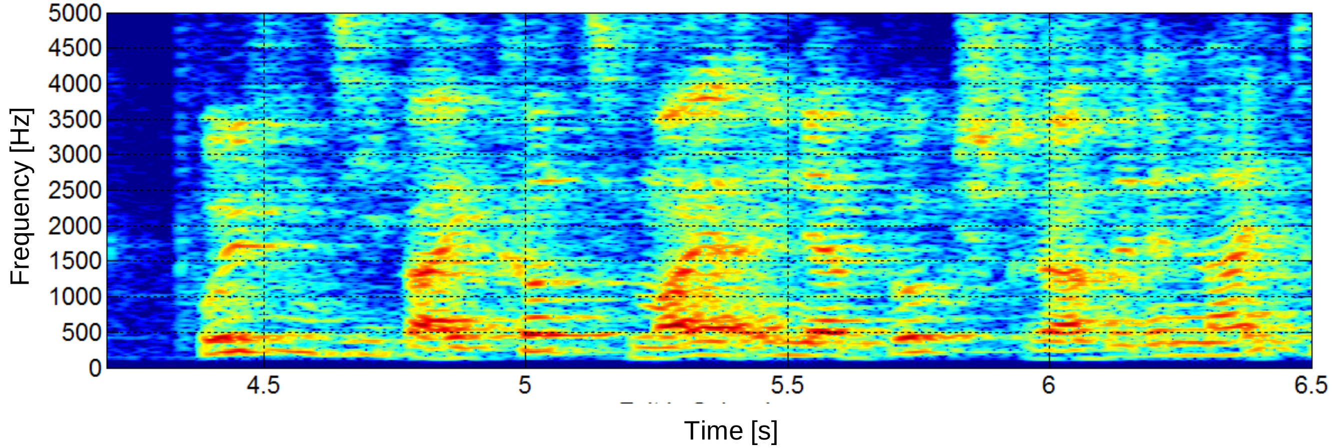
Combined noise reduction and dereverberation:



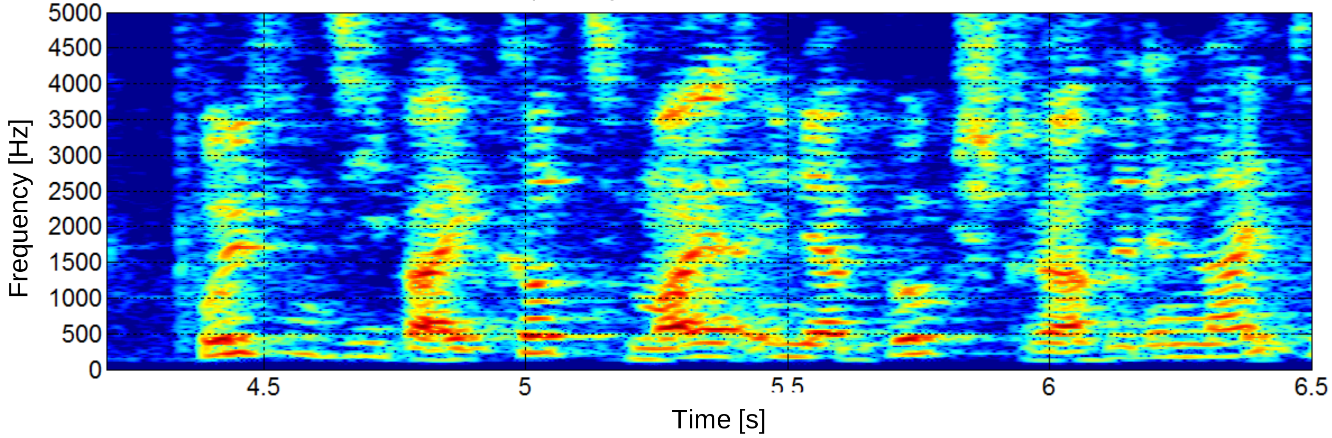
PSD = power spectral density

Dereverberation

Spectrogram of the reverberant input signal

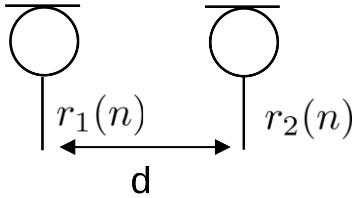


Spectrogram after dereverberation



Two microphone based dereverberation

- The late reflections are modeled as diffuse noise



- Definition of the coherence function:

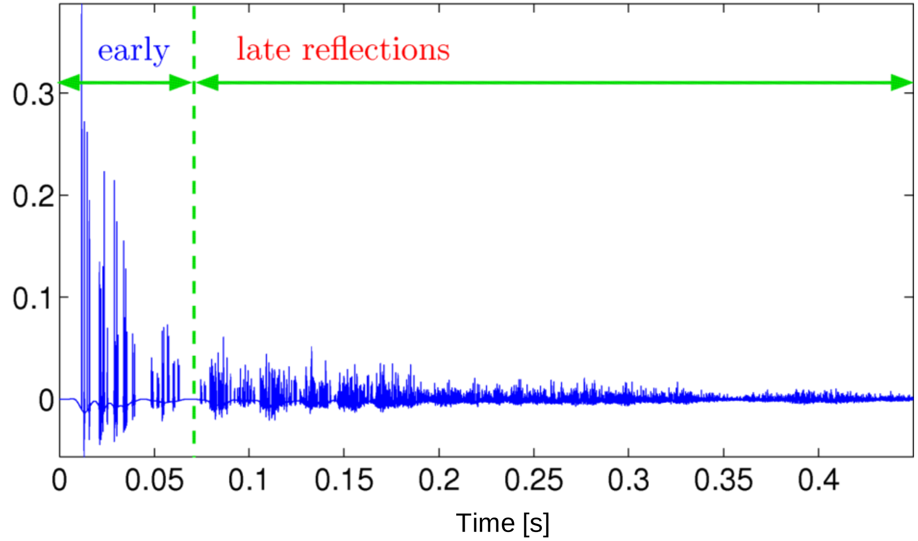
$$\gamma_{r_1 r_2}(\Omega) = \frac{S_{r_1 r_2}(\Omega)}{\sqrt{S_{r_1 r_1}(\Omega)\, S_{r_2 r_2}(\Omega)}}$$

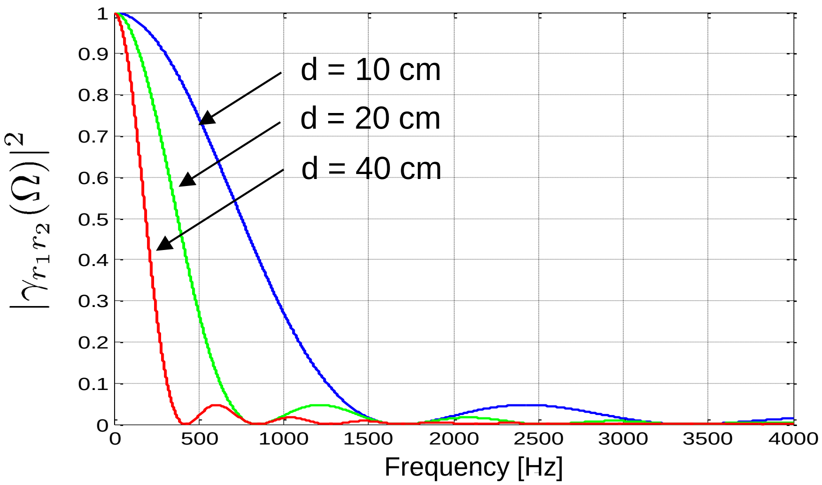
- For diffuse noise fields one obtains:

$$|\gamma_{r_1 r_2}(\Omega)|^2 = \frac{\sin^2(\Omega\, f_s\, d/c)}{(\Omega\, f_s\, d/c)^2}$$

f_s : sampling rate

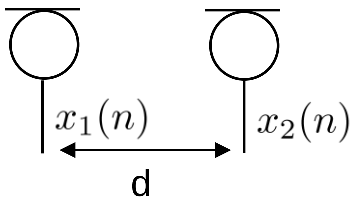
c : sound propagation speed





Two microphone based dereverberation

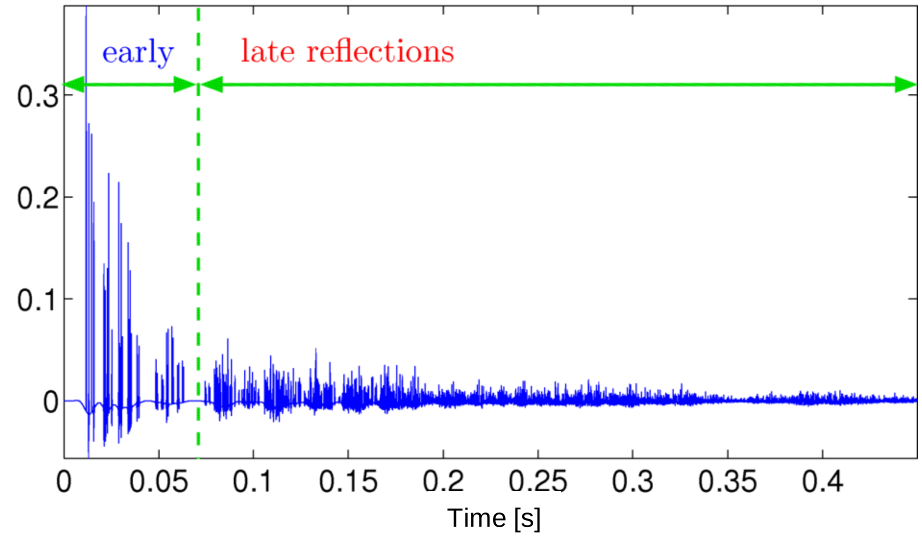
- Target signal + reverberation:



$$x_1(n) = s(n) + r_1(n)$$
$$x_2(n) = s(n) + r_2(n)$$

$$\gamma_{x_1x_2}(\Omega) = \frac{S_{x_1x_2}(\Omega)}{\sqrt{S_{x_1x_1}(\Omega)\,S_{x_2x_2}(\Omega)}} = \frac{S_{ss}(\Omega) + S_{r_1r_2}(\Omega)}{S_{ss}(\Omega) + S_{rr}(\Omega)}$$

with: $S_{rr}(\Omega) = S_{r_1r_1}(\Omega) = S_{r_2r_2}(\Omega)$



- For high frequencies no correlation for diffuse noise: $S_{r_1r_2}(\Omega) = 0$

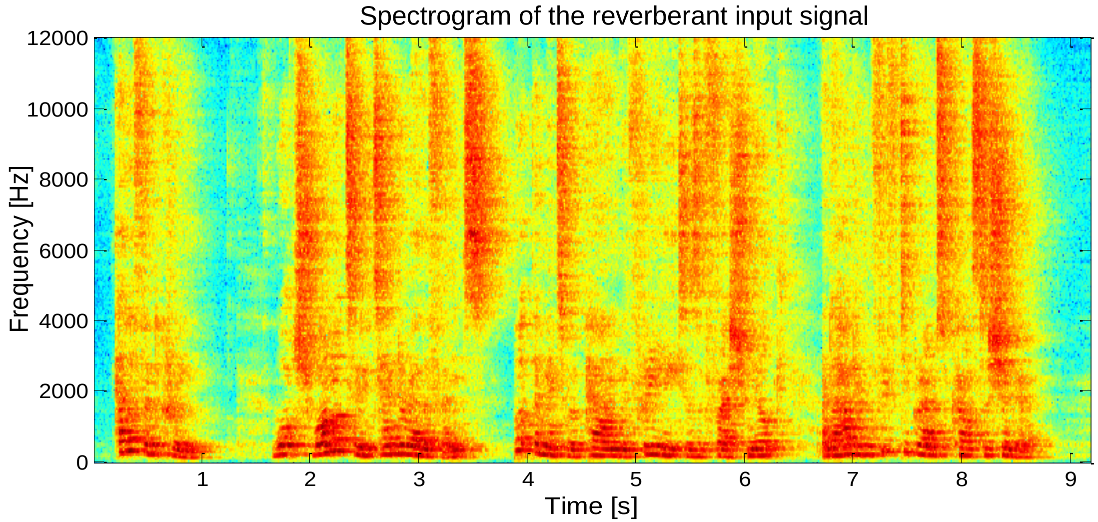
$$\Rightarrow \gamma_{x_1x_2}(\Omega) = \frac{S_{ss}(\Omega)}{S_{ss}(\Omega) + S_{rr}(\Omega)} = \frac{S_{ss}(\Omega)}{S_{xx}(\Omega)}$$

identical to the Wiener filter.

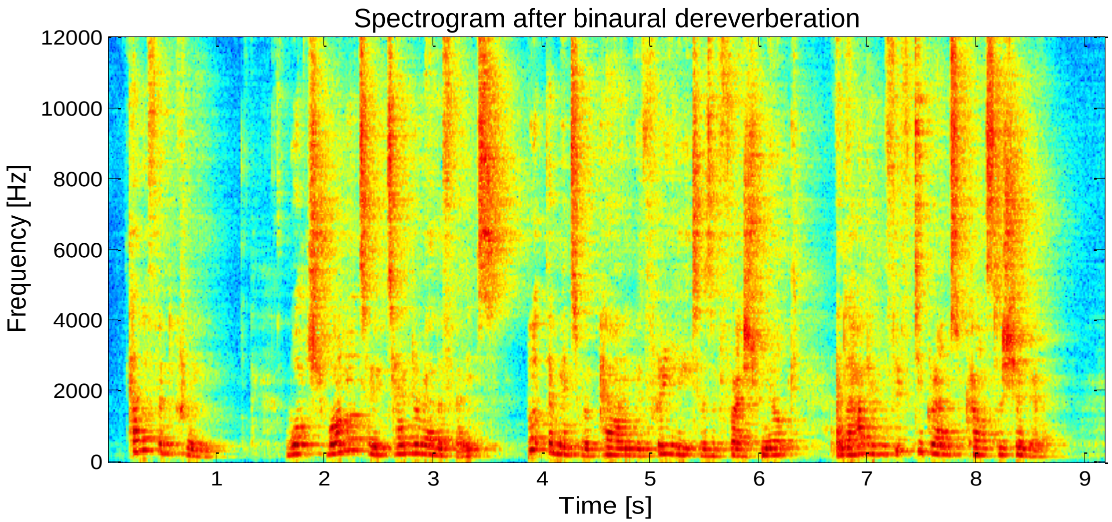
=> $\widehat{H}_{\text{opt}}(e^{j\Omega_\mu}, n) = \gamma_{x_1x_2}(\Omega_\mu, n)$

Coherence function allows filter design for the reverberation reduction. For low frequencies the diffuse coherence has to be considered.

Two microphone based dereverberation



Reverberant speech



Monaural dereverberation

Binaural dereverberation

One microphone on each side of the head
=> d = 17 cm

Summary & Outlook

Summary

- Wiener filter
- Realization in the frequency domain
- Modified basic filter approach
- Modified noise reduction approaches
- Dereverberation methods

Outlook to next week:

- *Beamforming*