Lecture

# Adaptive Filters

TECHNISCHE UNIVERSITÄT DARMSTADT

## Lecture 3: Linear Prediction

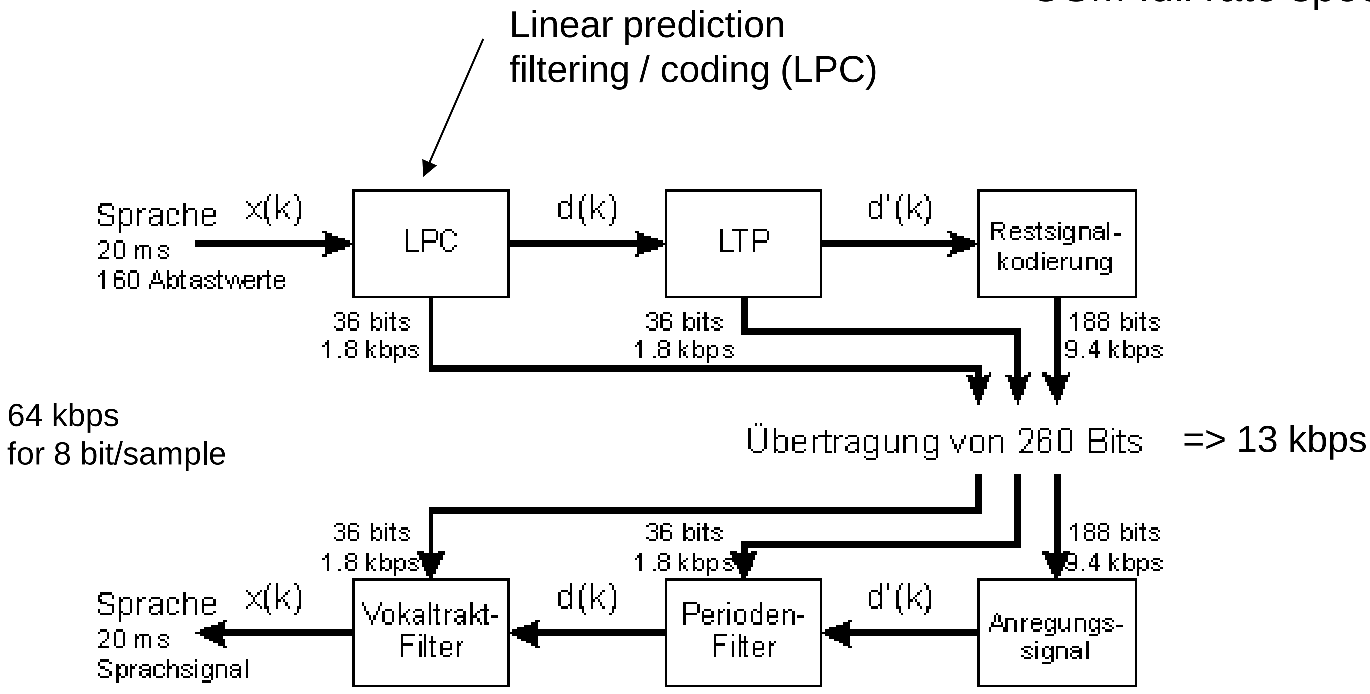
# Linear Prediction

- Linear prediction is one very strong instrument of adaptive signal processing
- Examples:
  - Source coding: eliminate redundant information
  - Speech signal processing: determine vocal tract filter
- Derivation of the optimum prediction filter coefficients based on known Wiener filter
- Prediction error analysis: power and correlation
- Examples of reduced amplitudes => source coding
- Examples of spectral envelope estimation
- Levinson-Durbin recursion
- Lattice filter structures

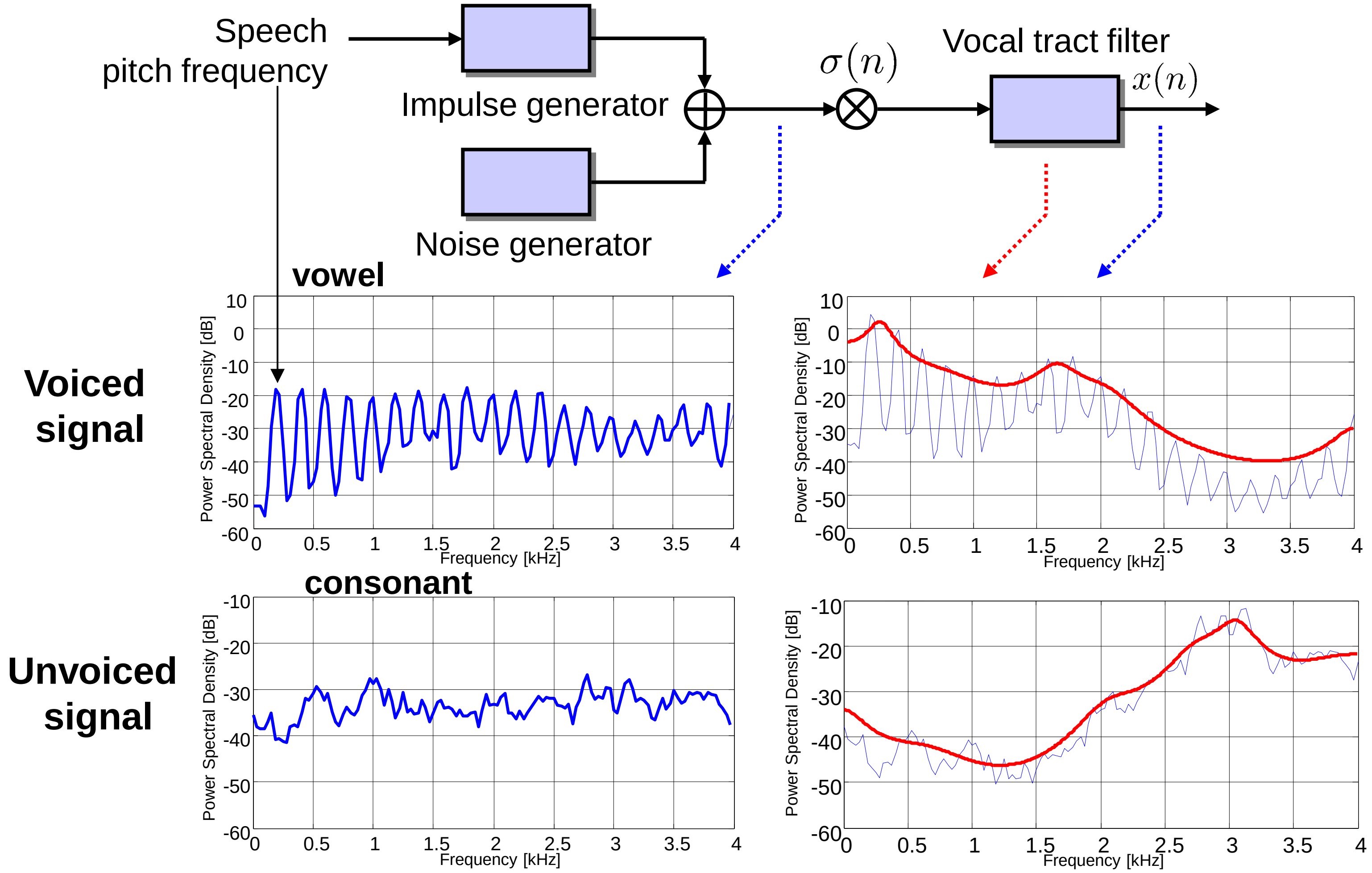
# Linear prediction: Application example (I): Source coding

**Target:**
Reduce the amount of data to be transmitted by removing redundant information

GSM full rate speech coder



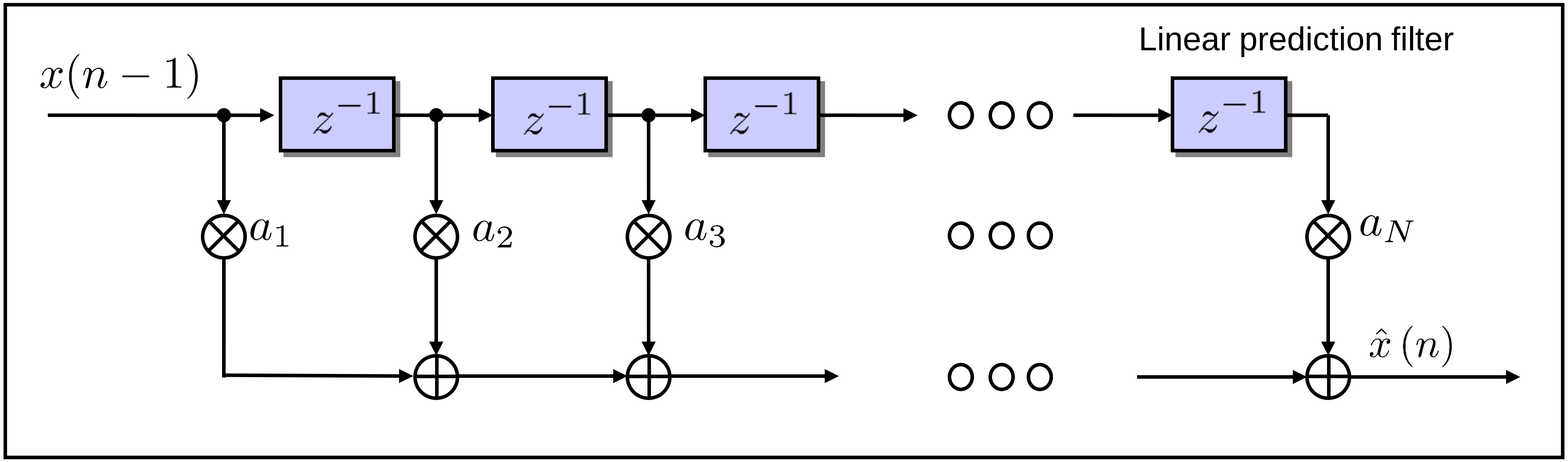
# Linear prediction:
# Application example (II): Estimation of vocal tract filters



Approach

**Prediction of the current signal sample based on the last $N$ signal samples:**

$$\hat{x}(n) = \sum_{i=1}^{N} a_i \, x(n-i)$$



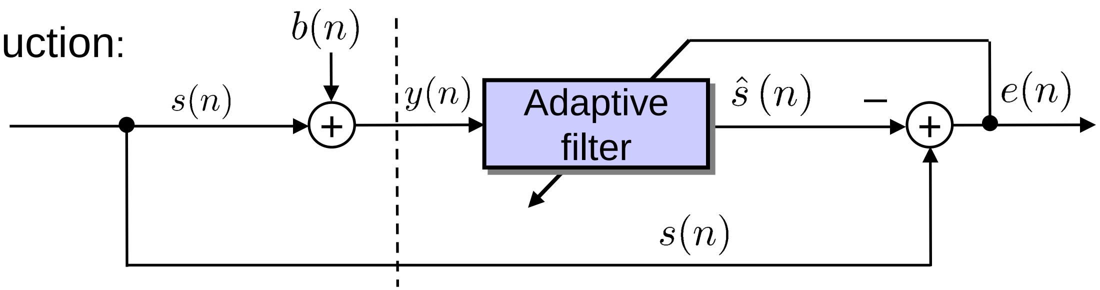

**With:**

- $\hat{x}(n)$ : Estimation for $x(n)$
- $a_i$ : prediction coefficients
- $N$ : Length / order of the prediction filter

# Wiener Filter

pergunta? pq colocar

- Noise reduction:




$$\mathrm{E}\left\{e^2(n)\right\} = r_{ss}(0) - 2\,\hat{\boldsymbol{h}}^{\mathrm{T}}\boldsymbol{r}_{ys}(0) + \hat{\boldsymbol{h}}^{\mathrm{T}}\boldsymbol{R}_{yy}\,\hat{\boldsymbol{h}}$$

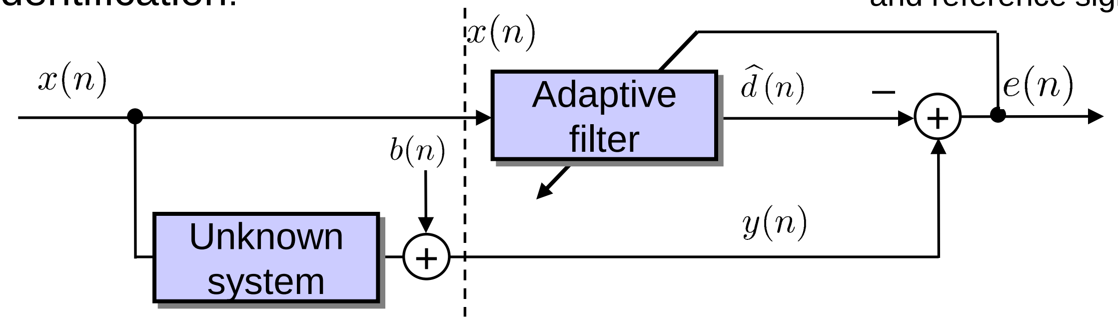
$$\mathrm{E}_{\min} = r_{ss}(0) - \boldsymbol{r}_{ys}^{\mathrm{T}}(0)\,\boldsymbol{R}_{yy}^{-1}\,\boldsymbol{r}_{ys}(0)$$

$$\hat{\boldsymbol{h}}_{\mathrm{opt}} = \boldsymbol{R}_{yy}^{-1}\boldsymbol{r}_{ys}(0)$$

autocorrelation of adaptive filter input

cross-correlation of adaptive filter input and reference signal

- System identification:



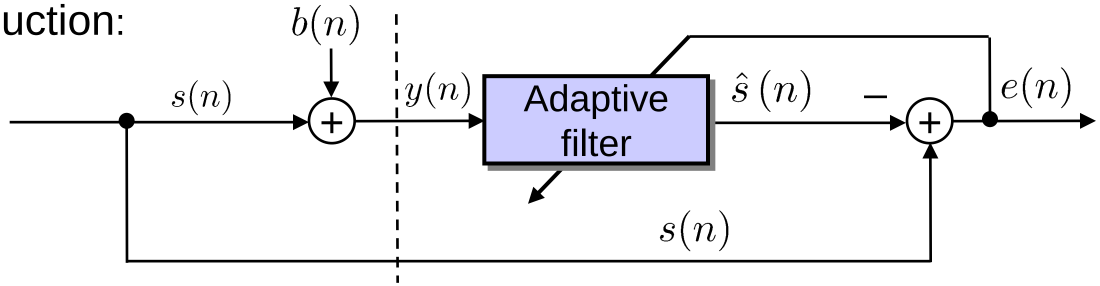

$$\hat{\boldsymbol{h}}_{\mathrm{opt}} = \boldsymbol{R}_{xx}^{-1}\boldsymbol{r}_{xy}(0)$$

$$\mathrm{E}\left\{e^2(n)\right\} = r_{yy}(0) - 2\,\hat{\boldsymbol{h}}^{\mathrm{T}}\boldsymbol{r}_{xy}(0) + \hat{\boldsymbol{h}}^{\mathrm{T}}\boldsymbol{R}_{xx}\,\hat{\boldsymbol{h}}$$

$$\mathrm{E}_{\min} = r_{yy}(0) - \boldsymbol{r}_{xy}^{\mathrm{T}}(0)\,\boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xy}(0)$$

# Wiener Filter

- Noise reduction:



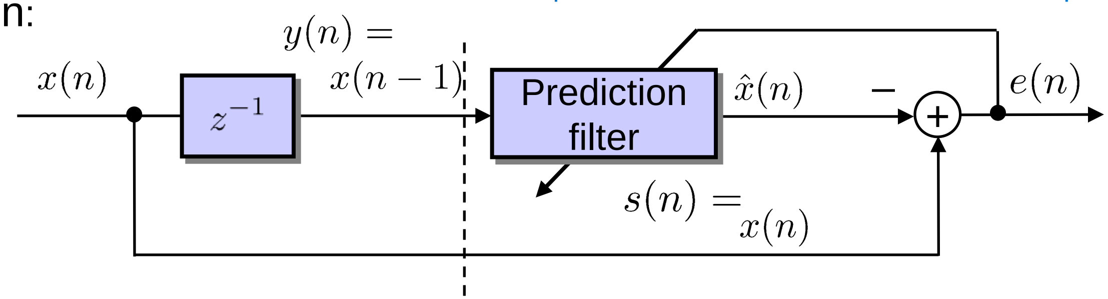

$$\hat{\boldsymbol{h}}_{\text{opt}} = \boldsymbol{R}_{yy}^{-1}\, \boldsymbol{r}_{ys}(0)$$

$$\boldsymbol{r}_{ys}(0) = [r_{ys}(0),\, r_{ys}(1),\, \cdots,\, r_{ys}(N-1)]^T$$

$$\boldsymbol{r}_{ys}(0) = [\mathrm{E}\{y(n)\, s(n)\},\, \mathrm{E}\{y(n)\, s(n+1)\},\, \cdots,\, \mathrm{E}\{y(n)\, s(n+N-1)\}]^T$$

pergunta: pq atrasar em 1 amostra só pra poder calcular ela mesma?
a potencia do erro é menor e também se distribui melhor na frequência - outras análises interessantes - quantização do erro é menor

- Prediction:




$$\hat{\boldsymbol{h}}_{\text{opt}} = \boldsymbol{R}_{xx}^{-1}\, \boldsymbol{r}_{xx}(1)$$

comparar com a formula do slide anterior

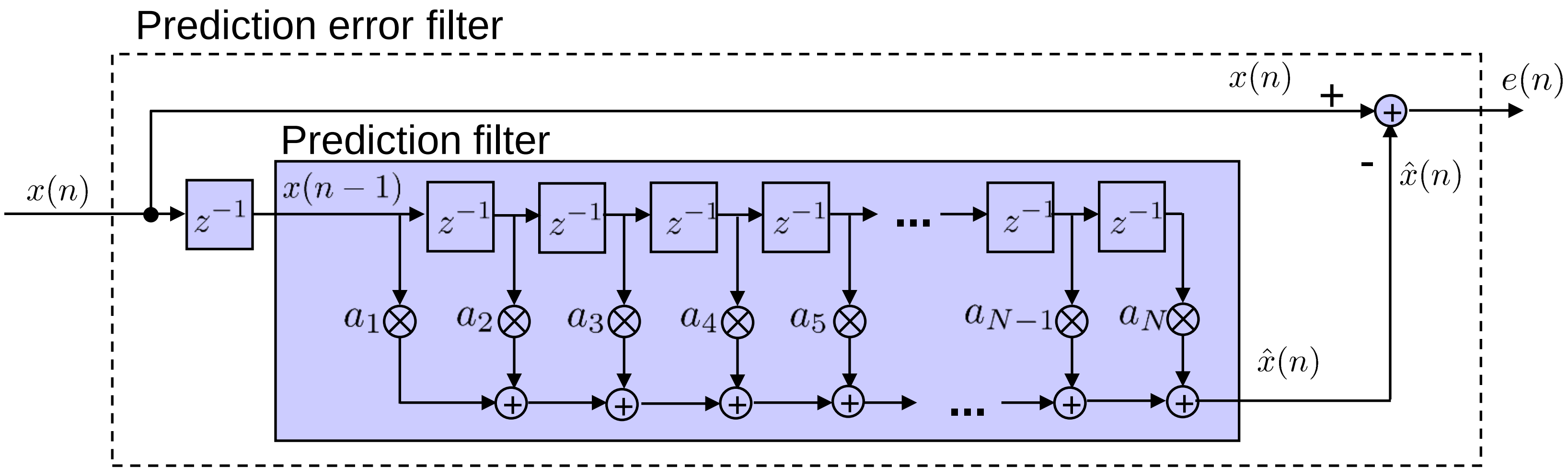
$$\boldsymbol{r}_{ys}(0) = [\mathrm{E}\{x(n-1)\, x(n)\},\, \mathrm{E}\{x(n-1)\, x(n+1)\},\, \cdots,\, \mathrm{E}\{x(n-1)\, x(n+N-1)\}]^T$$
$$= [r_{xx}(1),\, r_{xx}(2),\, \cdots,\, r_{xx}(N)]^T = \boldsymbol{r}_{xx}(1)$$

pergunta: pq pensar o vetor de autocorrelação é usado começando em 1 e indo até N?

## Prediction

$\hat{\boldsymbol{h}}_{\text{opt}} = \boldsymbol{a} = \boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1)$ : Yule-Walker equation

$$\boxed{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix}^{-1} \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(N) \end{bmatrix}}$$



# Example: White input signal

- Input signal $x(n)$ : white noise with the power $\sigma_0^2$ (mean value = 0)
- Prediction order: $N = 3$
- Prediction by one sample: $L = 1$

**Resulting in:**

$$\boldsymbol{R}_{xx} = \begin{bmatrix} \sigma_0^2 & 0 & 0 \\ 0 & \sigma_0^2 & 0 \\ 0 & 0 & \sigma_0^2 \end{bmatrix} \qquad \boldsymbol{R}_{xx}^{-1} = \frac{1}{\sigma_0^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

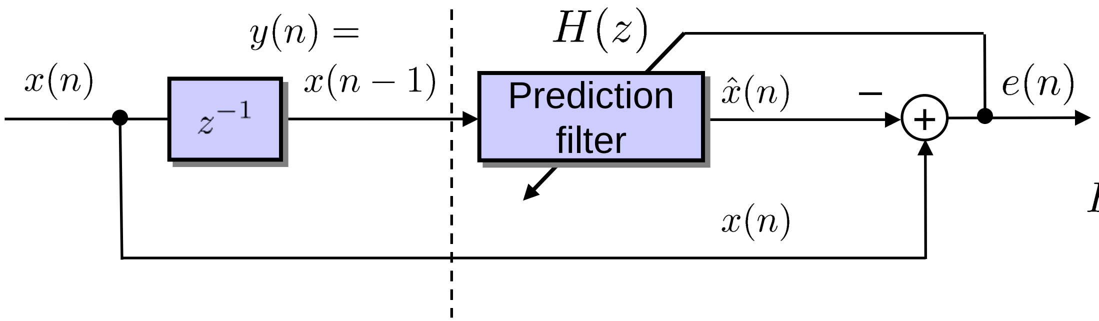
$$\boldsymbol{r}_{xx}(1) = \begin{bmatrix} 0, & 0, & 0 \end{bmatrix}^{\mathrm{T}}$$

$$\boldsymbol{a} = \boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1) = \begin{bmatrix} 0, & 0, & 0 \end{bmatrix}^{\mathrm{T}}$$

i.e., no prediction is possible, signal does not contain redundant information.

toda a informação de x(n) NÃO pode ser prevista a partir de amostras anteriores -> x(N) carrega nenhuma informação
É melhor não fazer nada quando o sinal é branco, pq nada pode ser previsto de x(n) a partir das amostras anteriores

# Prediction: Error signal power




$$\hat{\boldsymbol{h}}_{\mathrm{opt}} = \boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1)$$

$$H(z) = \sum_{i=0}^{N-1} \hat{h}_{\mathrm{opt},i}\, z^{-i} = \sum_{i=0}^{N-1} a_{i+1}\, z^{-i}$$

$$e(n) = x(n) - \sum_{i=0}^{N-1} \hat{h}_{\mathrm{opt},i}\, x(n-i-1) = x(n) - \sum_{i=1}^{N} a_i\, x(n-i)$$

**Wiener filter (known from last lecture):**

$$\mathrm{E}\left\{e^2(n)\right\} = r_{yy}(0) - 2\,\hat{\boldsymbol{h}}^{\mathrm{T}}\boldsymbol{r}_{xy}(0) + \hat{\boldsymbol{h}}^{\mathrm{T}}\boldsymbol{R}_{xx}\,\hat{\boldsymbol{h}}$$

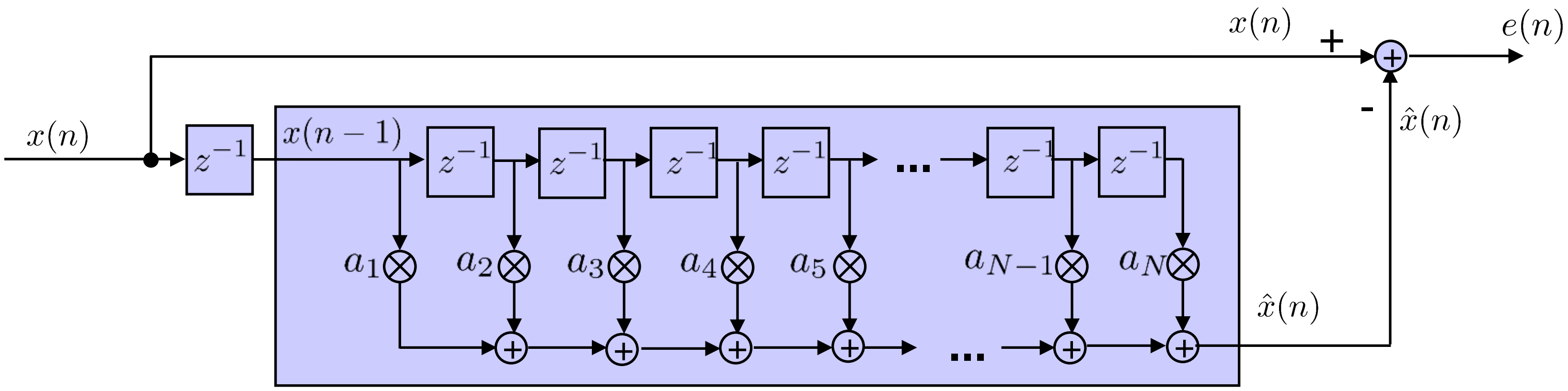
$$\mathrm{E}_{\min} = r_{yy}(0) - \boldsymbol{r}_{xy}^{\mathrm{T}}(0)\,\boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xy}(0)$$

**Predictor:**

$$\mathrm{E}\left\{e^2(n)\right\} = r_{xx}(0) - 2\,\boldsymbol{a}^{\mathrm{T}}\boldsymbol{r}_{xx}(1) + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{R}_{xx}\,\boldsymbol{a}$$

$$\boxed{\begin{aligned}\mathrm{E}_{\min} &= r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1)\\ \mathrm{E}_{\min} &= r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{a}\end{aligned}}$$

# Prediction error gain




$$\mathrm{E}_{\min} = r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1)$$

=> error signal power equal or reduced compared to input signal power.

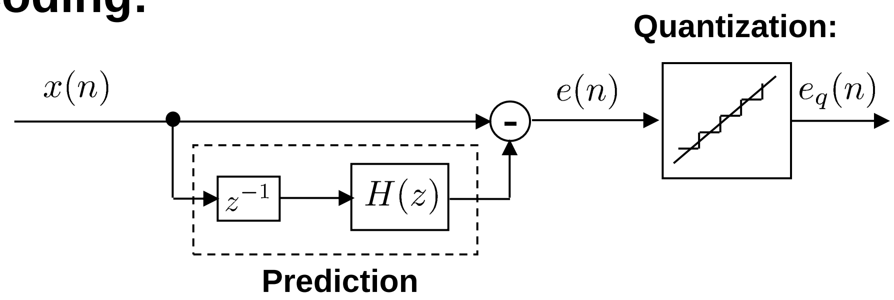
Prediction error gain:

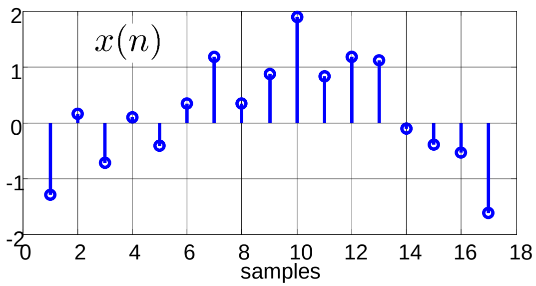
$$\boxed{\frac{\mathrm{E}\{x^2(n)\}}{\mathrm{E}_{\min}} = \frac{r_{xx}(0)}{r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1)}}$$

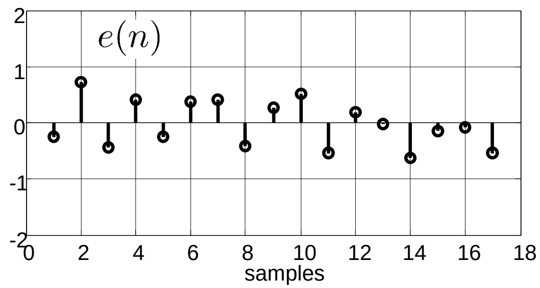
RANGE: [1, +inf [

# Linear prediction application example: source coding

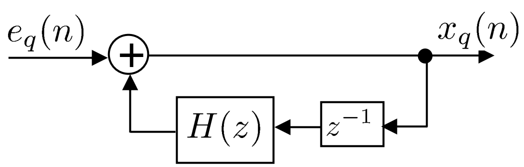
## Coding:



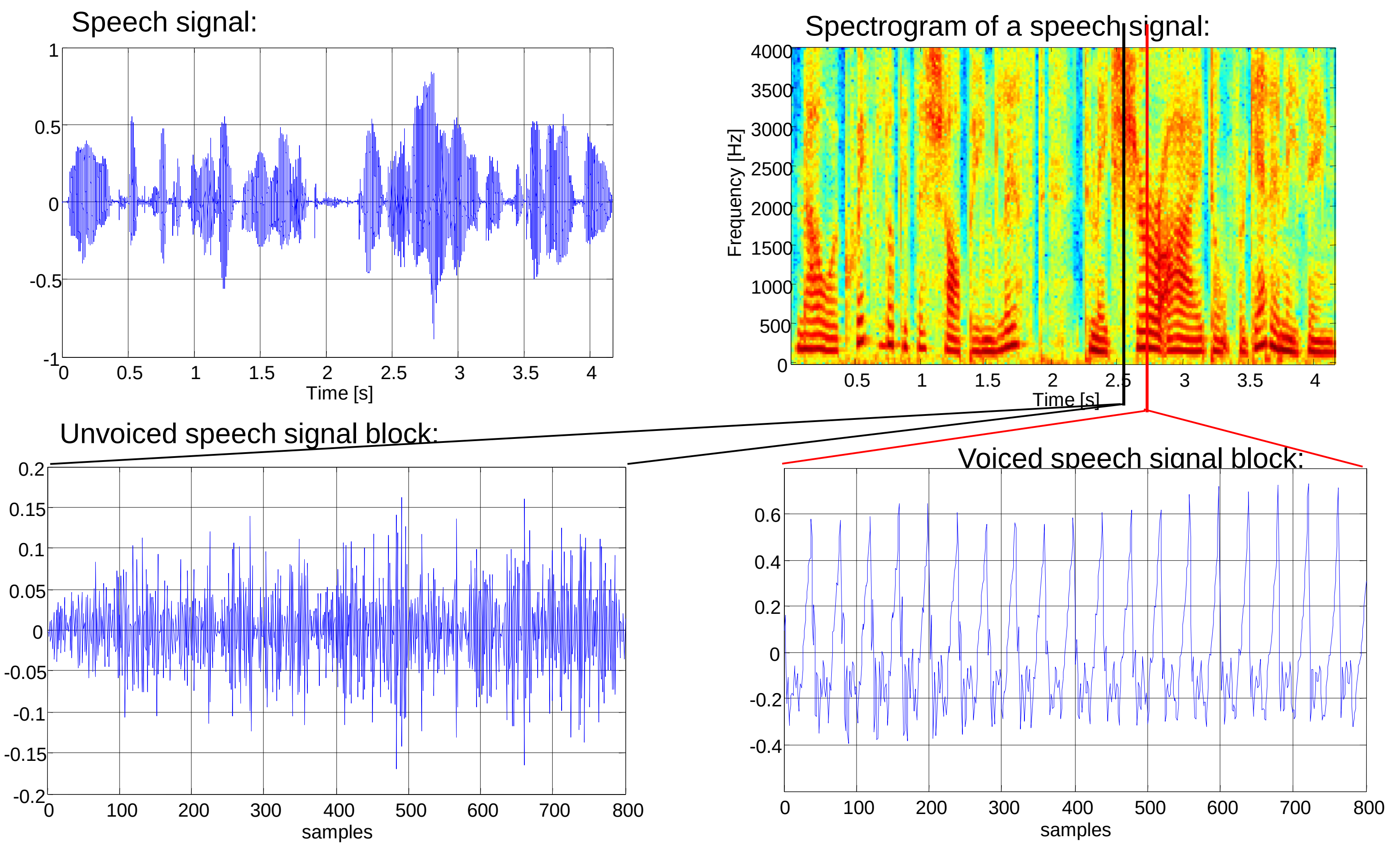

## Decoding:




Reduced amplitude of signal to code
=> reduced number of coding bit required

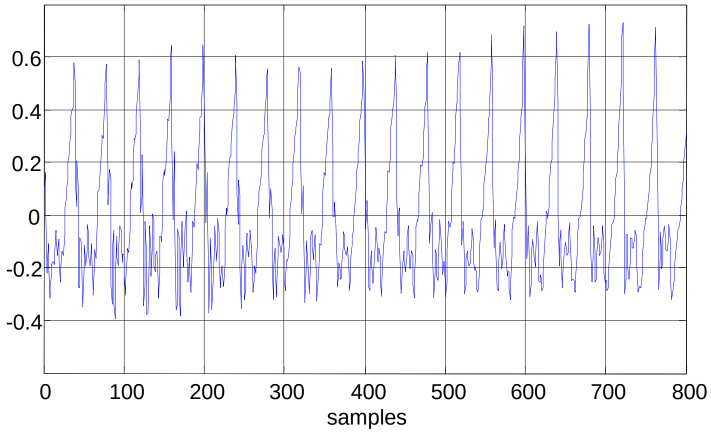
Example: Power reduced by 6 dB => 1 Bit less per signal sample required at quantization

# Linear prediction: source coding

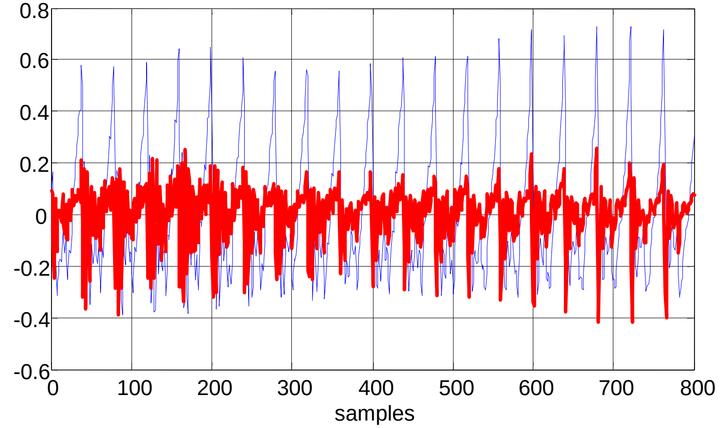


# Linear prediction: source coding

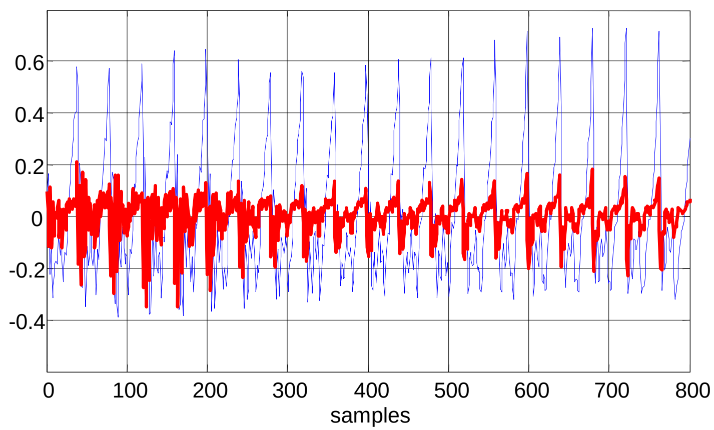
Voiced input signal block:



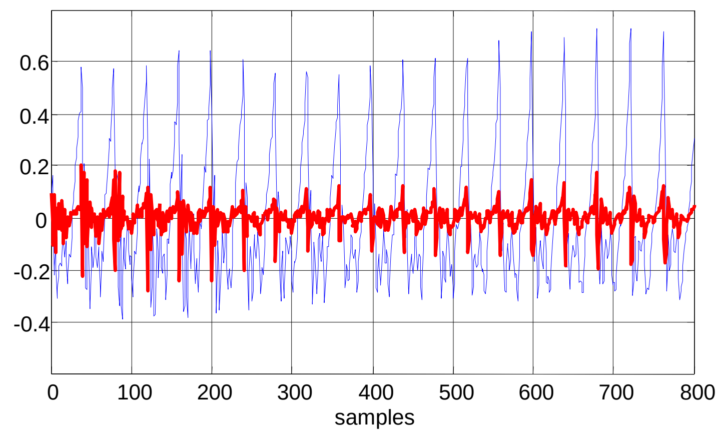

Error signal: prediction order 1; prediction gain: 7.2 dB



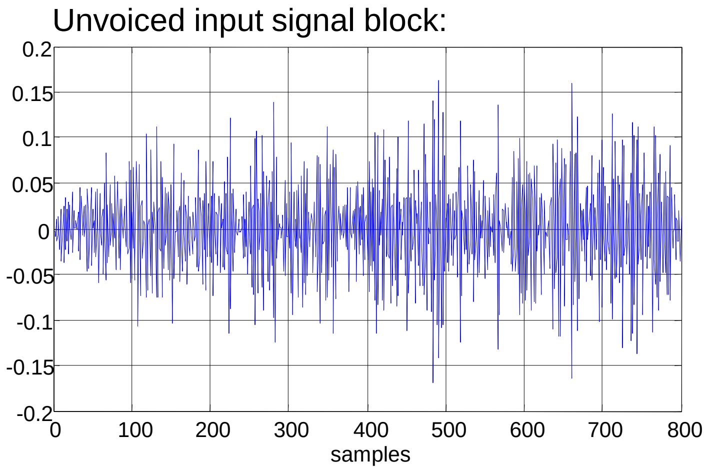

Error signal: prediction order 2; prediction gain: 10.4 dB

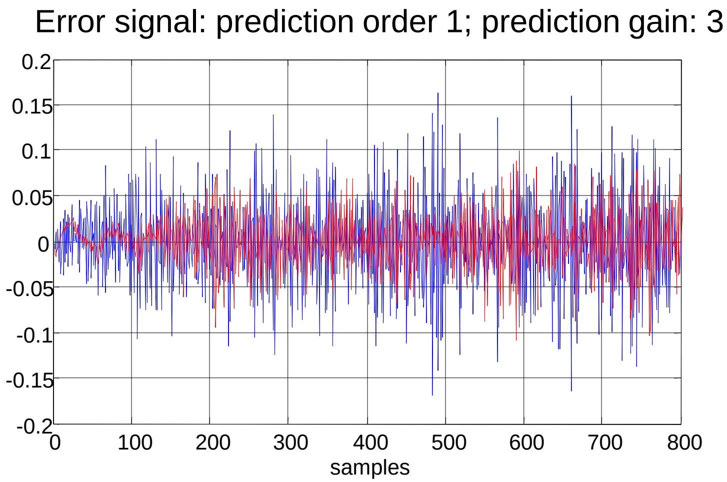


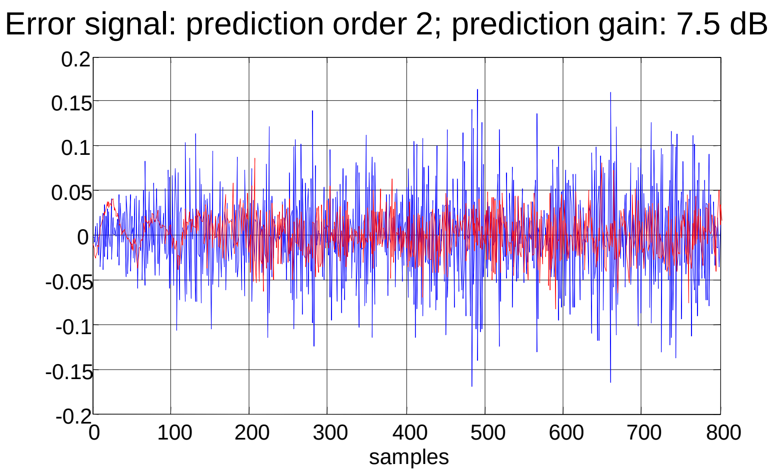

Error signal: prediction order 10; prediction gain: 13.6 dB

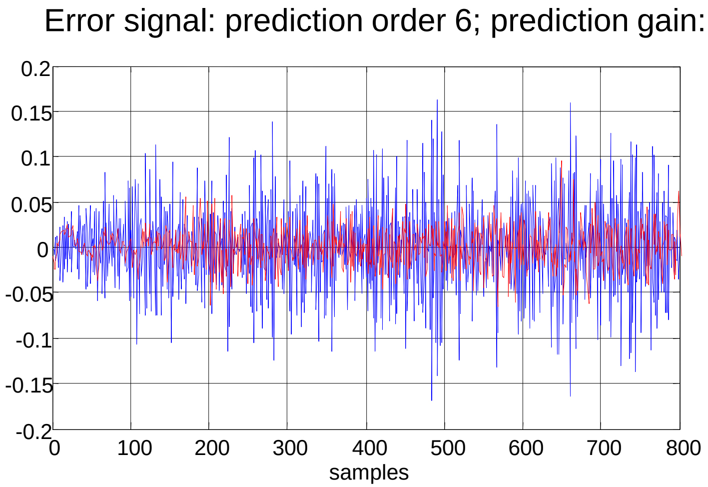


# Linear prediction: source coding



# Correlation analysis of the prediction error signal

quando não há correlação/quando não há redundancia -> sinal branco

se vc consegue remover toda a redundancia então não haveria correlação entre as amostras de erro -> erro seria branco

$$
\begin{aligned}
r_{ee}(l) &= \mathrm{E}\{(x(n) - \sum_{i=1}^{N} a_i\, x(n-i))\,(x(n+l) - \sum_{j=1}^{N} a_j\, x(n-j+l))\} \\
&= r_{xx}(l) - \sum_{i=1}^{N} a_i\, r_{xx}(l-i) - \sum_{i=1}^{N} a_i\, r_{xx}(l+i) + \sum_{i=1}^{N}\sum_{j=1}^{N} a_i\, a_j r_{xx}(l+i-j) \\
&= \underbrace{r_{xx}(l) - \sum_{i=1}^{N} a_i\, r_{xx}(l-i)}_{=0 \text{ for } l=1,\ldots,N} - \sum_{i=1}^{N} a_i \underbrace{\left[ r_{xx}(l+i) - \sum_{j=1}^{N} a_j r_{xx}(l+i-j) \right]}_{=0 \text{ for } l+i=1,\ldots,N}
\end{aligned}
$$

with the relation for optimum prediction coefficients:

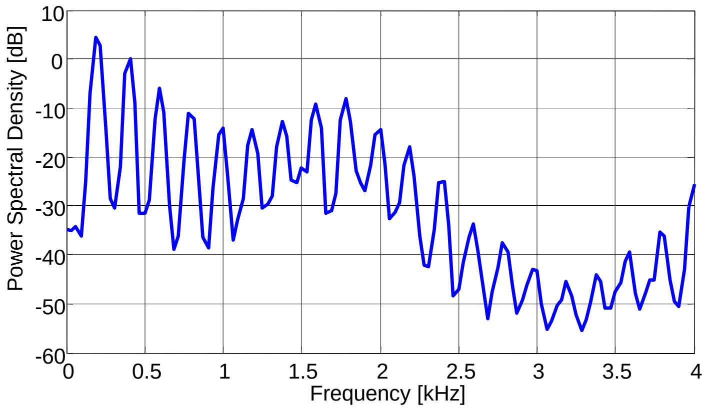
$$
r_{xx}(l) \stackrel{!}{=} \sum_{i=1}^{N} a_i\, r_{xx}(l-i) \qquad \text{for } l = 1, \ldots, N
$$

$$
\boxed{r_{ee}(l) = 0 \qquad \text{for } |l| > 0 \text{ and } N \to \infty}
$$

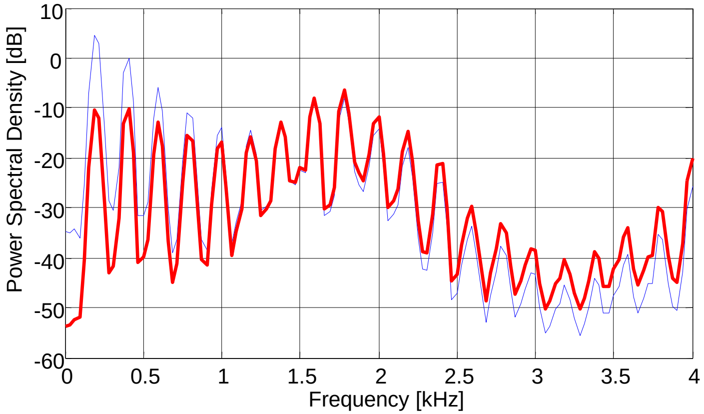
=> whitening of output signal spectrum

# Linear prediction: spectral envelope calculation

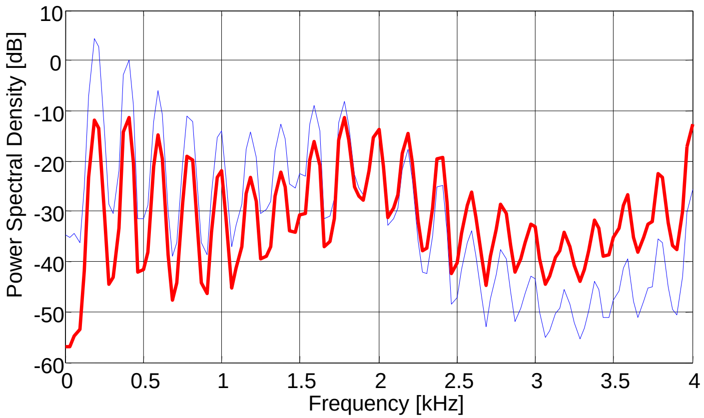
Voiced input signal block:



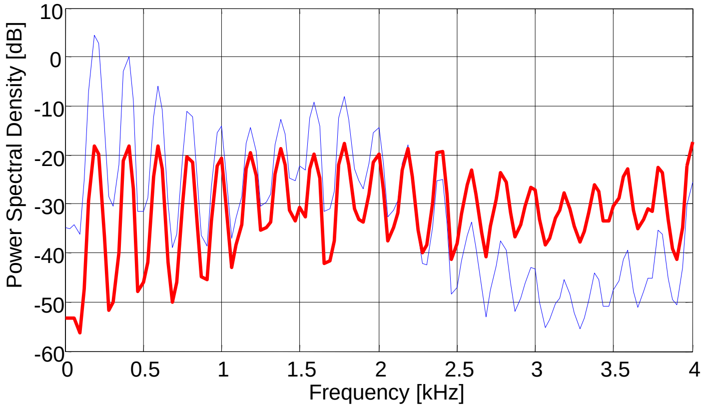

Error signal: prediction order 1; prediction gain: 7.2 dB




Error signal: prediction order 2; prediction gain: 10.4 dB

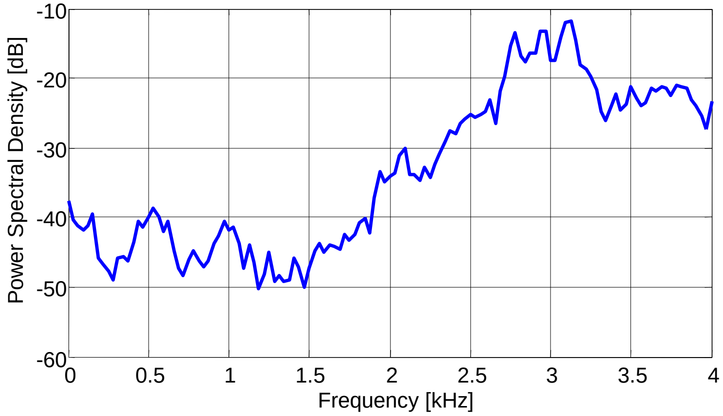



Error signal: prediction order 10; prediction gain: 13.6 dB

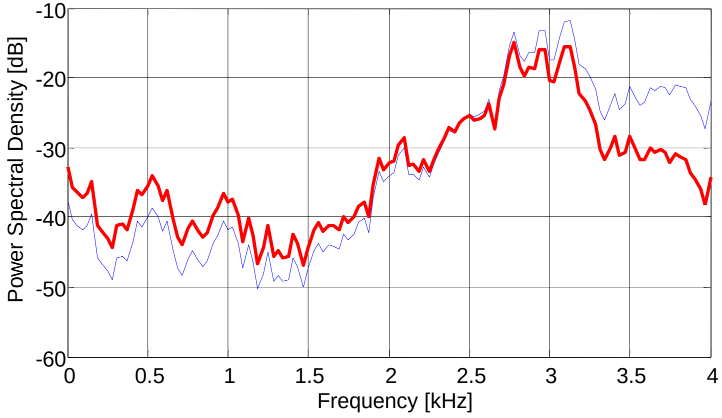


# Linear prediction: spectral envelope calculation

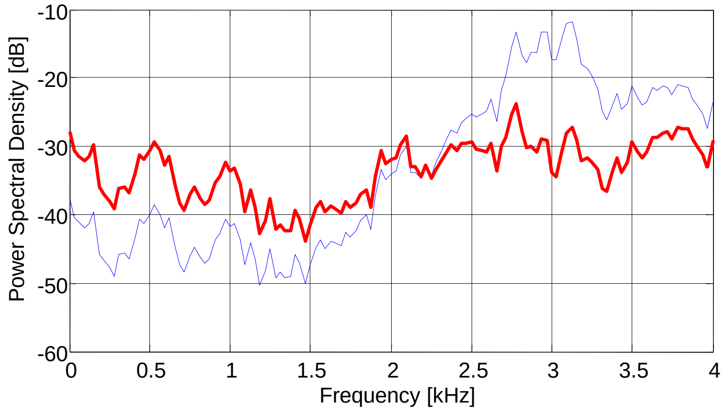
Unvoiced input signal block:



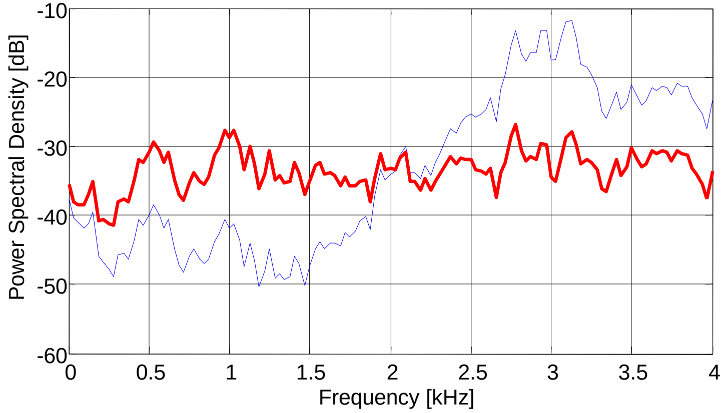

Error signal: prediction order 1; prediction gain: 3.0 dB




Error signal: prediction order 2; prediction gain: 7.5 dB




Error signal: prediction order 6; prediction gain: 8.9 dB



# Linear prediction: spectral envelope calculation

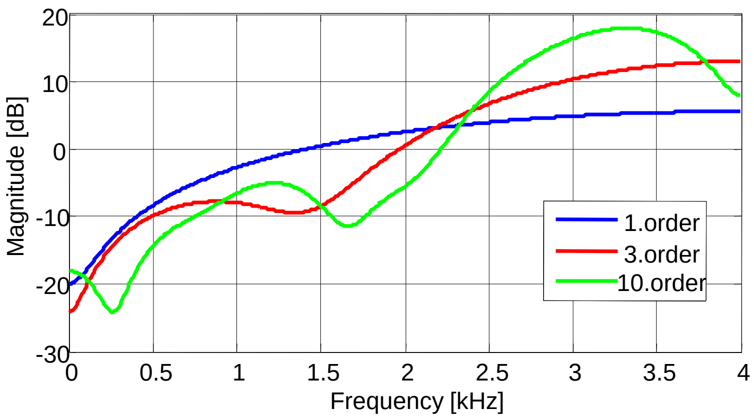
PEF modela o INVERSO do envelope -------- O inverso do PEF modela o envelope ->
Explicação: PEF faz com que o sinal*PEF = sinal branco = constante

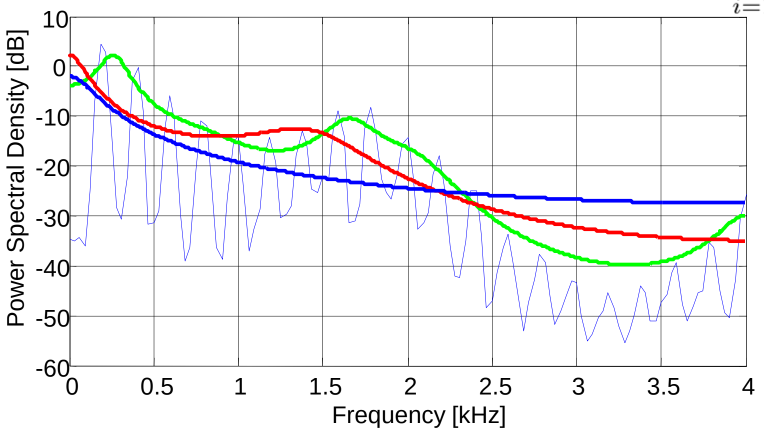
**Voiced input signal block:**

Prediction error filter: $H_{\text{PEF}}\left(e^{j\Omega}\right) = 1 - \sum_{i=1}^{N} a_i \, e^{-j\Omega i}$

moving average

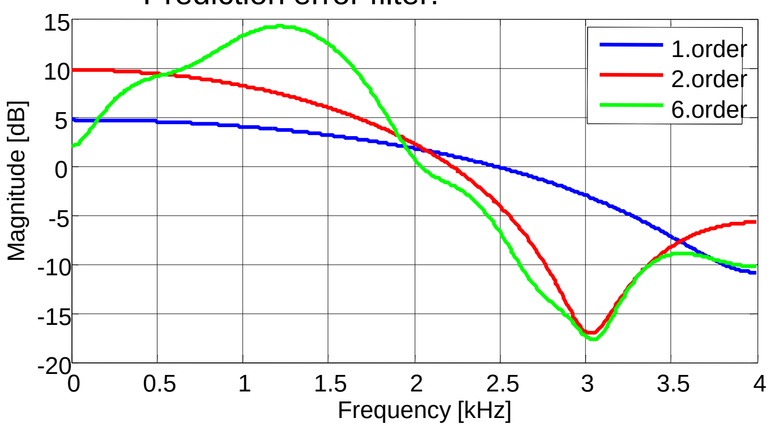
Inverse prediction error filter / approx. the PSD: $H_{\text{inv. PEF}}\left(e^{j\Omega}\right) = \dfrac{1}{1 - \sum_{i=1}^{N} a_i \, e^{-j\Omega i}}$

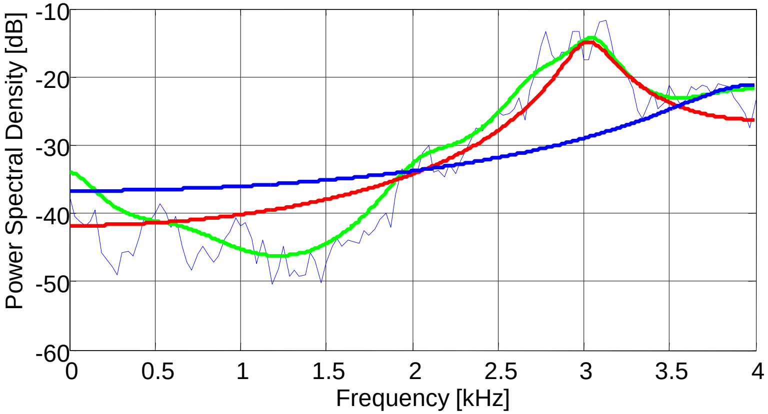



**Unvoiced input signal block:**

Prediction error filter:



# Estimation of the autocorrelation function

**Problem:**

Ensemble averages are unknown in most applications.

**Remedy:**

Assume ergodic processes: replace ensemble averages by time averages:

$$\mathrm{E}\Big\{x(n)\,x(n+l)\Big\} \qquad\qquad \sum_n x(n)\,x(n+l)$$

geralmente não é o caso, mas...
?da pra assumir localmente?
geralmente não, não tente fazer isso kkkk use outros processos de estimation

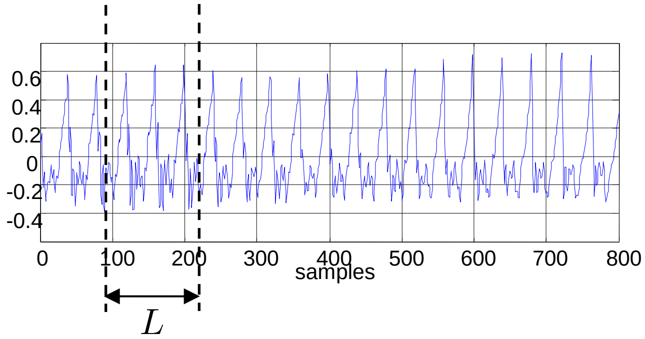
**Estimation procedures:**

There are some estimation procedures, differing by the properties of the estimated autocorrelation function (biased / unbiased; ACF matrix which is positive-definite)

# Estimation of the autocorrelation function

## Example – „autocorrelation method“:

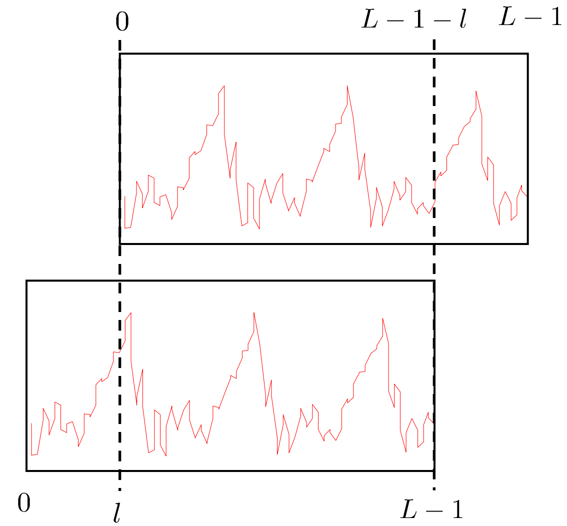
- Windowing:




- Calculation:

$$\hat{r}_{xx}(l) = \begin{cases} \frac{1}{L} \sum\limits_{n=0}^{L-1-l} x(n)\, x(n+l), & \text{for } l \geq 0, \\ \frac{1}{L} \sum\limits_{n=-l}^{L-1} x(n)\, x(n+l), & \text{for } l < 0 \end{cases}$$

- Correlation:
  => Time shifted blocks; element-wise multiplication; summation




- Note: L should be chosen larger than the prediction order N!

# Estimation of the autocorrelation function

**Properties of the „autocorrelation method“:**

- Biased estimation => Bias calculation:

embora tenha esse problema, esse bias até que ajuda (acho que no sentido de que a autocorrelação para l maior geralmente é menor mesmo

$$\mathrm{E}\{\hat{r}_{xx}(l)\} = \begin{cases} \frac{1}{L}\sum\limits_{n=0}^{L-1-l} \mathrm{E}\{x(n)\,x(n+l)\}, & \text{for } l \geq 0, \\ \frac{1}{L}\sum\limits_{n=-l}^{L-1} \mathrm{E}\{x(n)\,x(n+l)\}, & \text{for } l < 0 \end{cases} = \frac{L-|l|}{L}\, r_{xx}(l) \leq r_{xx}(l)$$

- Properties:

$$\begin{aligned} \hat{r}_{xx}(l) &= 0, \text{ for } |l| \geq L \\ \hat{r}_{xx}(l) &= \hat{r}_{xx}(-l) \\ \hat{r}_{xx}(l) &\leq \hat{r}_{xx}(0) \end{aligned}$$

para um prediction order de 10, seria interessante uma janela de mais ou menos 100 ou mais

- The ACF matrix estimated with the autocorrelation method is positive-definite.
- The ACF matrix estimated with the autocorrelation method has Toeplitz structure.

pergunta: seria diferente se nós na verdade usassemos o fator correto de normalização??´e exatamente o que ta sendo dito abaixo kkkk

**Other methods:**

- Covariance method
- Modified covariance method

=> Unbiased estimates;
But: ACF matrices have no Toeplitz structure
and are not positive-definite

# Levinson-Durbin recursion - Motivation

**Problem:**

Solving the matrix equation

$$\boldsymbol{a} = \boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1)$$

requires a large computational effort proportional to $N^2$ in case of a Toeplitz structure and proportional to $N^3$ otherwise. Additional problems may occur during matrix inversion in case of a bad conditioning of the ACF matrix.

**Target:**

Robust method which solves the above equation without calculating a matrix inversion of: $\boldsymbol{R}_{xx}$

**Solution:**

Taking advantage of the Toeplitz structure of the matrix $\boldsymbol{R}_{xx}$.

- Recursion with the prediction order
- Combining forward and backward prediction

**Literature (original):**


- J. Durbin: *The Fitting of Time Series Models*, Rev. Int. Stat. Inst., No. 28, pages 233 - 244, 1960
- N. Levinson: *The Wiener RMS Error Criterion in Filter Design and Prediction*, J. Math. Phys., Nr. 25, Pages 261 - 268, 1947

# Levinson-Durbin recursion – General Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

- Recursion: Three steps: Initialization, Iteration, Stop

- **Initialization:** Predictor of order 1:

  $$\boldsymbol{a} = \boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1) \qquad \text{=> Order 1:} \qquad a_1 = \frac{r_{xx}(1)}{r_{xx}(0)}$$

- **Iteration:** Increasing the order:

  Not identical!

  Differentiation by upscript (1), (2), … (N)

  Order 1: $$\boldsymbol{a}^{(1)} = a_1^{(1)} = \frac{r_{xx}(1)}{r_{xx}(0)}$$

  Order 2: $$\boldsymbol{a}^{(2)} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix}^{-1} \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \end{bmatrix}$$

  However: Recursion does not solve the Yule-Walker equation (matrix inversion) but uses $\boldsymbol{a}^{(N)} = [a_1^{(N)}, a_2^{(N)}, ..., a_N^{(N)}]^T$ and $\boldsymbol{r}_{xx}(l)$ to generate

  $$\boldsymbol{a}^{(N+1)} = [a_1^{(N+1)}, a_2^{(N+1)}, ..., a_N^{(N+1)}, a_{N+1}^{(N+1)}]^T$$

- Stop: If iteration target is reached

# Levinson-Durbin recursion – Extension of the matrix

(according to Monson H. Hayes: “Statistical Digital Signal Processing and Modeling“)

ele não quer que explique as deduções na prova - talvez não precise decorar com detalhe

- **Prediction equation:**

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(N) \end{bmatrix}$$

$$\begin{bmatrix} r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} -1 \\ a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

with: $\boxed{\mathrm{E}_{\min} = r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\, \boldsymbol{a}}$ a power value

=>

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(N) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} -1 \\ a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} -\mathrm{E}_{\min} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

# Levinson-Durbin recursion - Extension of the matrix

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(N) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} -1 \\ a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} -\mathrm{E}_{\min} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\mathrm{E}_{\min} = r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{a} \qquad \boldsymbol{a} = [a_1\, a_2\, \ldots\, a_N]^T$$

- **Extension of the order:**

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(N) & r_{xx}(N+1) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) & r_{xx}(N) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) & r_{xx}(N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(N+1) & r_{xx}(N) & r_{xx}(N-1) & \cdots & r_{xx}(1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} -1 \\ a_1 \\ a_2 \\ \vdots \\ a_N \\ 0 \end{bmatrix} = \begin{bmatrix} -\mathrm{E}_{\min} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -V \end{bmatrix}$$

Last row of the matrix:

$$V = r_{xx}(N+1) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\tilde{\boldsymbol{a}} \qquad \tilde{\boldsymbol{a}} = [a_N\, a_{N-1}\, \ldots\, a_1]^T$$

# Levinson-Durbin recursion

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(N) & r_{xx}(N+1) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) & r_{xx}(N) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) & r_{xx}(N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(N+1) & r_{xx}(N) & r_{xx}(N-1) & \cdots & r_{xx}(1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} -1 \\ a_1 \\ a_2 \\ \vdots \\ a_N \\ 0 \end{bmatrix} = \begin{bmatrix} -\mathrm{E}_{\min} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -V \end{bmatrix}$$

- **Reorder the equations (flip all rows and columns), profit from Toeplitz structure:**

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(N) & r_{xx}(N+1) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) & r_{xx}(N) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) & r_{xx}(N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(N+1) & r_{xx}(N) & r_{xx}(N-1) & \cdots & r_{xx}(1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} 0 \\ a_N \\ a_{N-1} \\ \vdots \\ a_1 \\ -1 \end{bmatrix} = \begin{bmatrix} -V \\ 0 \\ 0 \\ \vdots \\ 0 \\ -\mathrm{E}_{\min} \end{bmatrix}$$

Levinson-Durbin recursion

**Note with (N), the N-th order of prediction:**

$$\boldsymbol{a}^{(N)} = \left[a_1^{(N)}\ a_2^{(N)}\ \dots\ a_N^{(N)}\right]^T \quad \boldsymbol{a}^{(N+1)} = \left[a_1^{(N+1)}\ a_2^{(N+1)}\ \dots\ a_{N+1}^{(N+1)}\right]^T$$

**Linear combination of previous matrix equations:**

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(N) & r_{xx}(N+1) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) & r_{xx}(N) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) & r_{xx}(N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(N+1) & r_{xx}(N) & r_{xx}(N-1) & \cdots & r_{xx}(1) & r_{xx}(0) \end{bmatrix} \left\{ \begin{bmatrix} -1 \\ a_1^{(N)} \\ a_2^{(N)} \\ \vdots \\ a_N^{(N)} \\ 0 \end{bmatrix} - \Gamma^{(N+1)} \begin{bmatrix} 0 \\ a_N^{(N)} \\ a_{N-1}^{(N)} \\ \vdots \\ a_1^{(N)} \\ -1 \end{bmatrix} \right\} = \begin{bmatrix} -\mathrm{E}_{\min}^{(N)} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -V^{(N)} \end{bmatrix} - \Gamma^{(N+1)} \begin{bmatrix} -V^{(N)} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -\mathrm{E}_{\min}^{(N)} \end{bmatrix}$$

agora ele só iguala

**Matrix equation for predictor of order (N+1):**

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(N) & r_{xx}(N+1) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) & r_{xx}(N) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) & r_{xx}(N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{xx}(N) & r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(N+1) & r_{xx}(N) & r_{xx}(N-1) & \cdots & r_{xx}(1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} -1 \\ a_1^{(N+1)} \\ a_2^{(N+1)} \\ \vdots \\ a_N^{(N+1)} \\ a_{N+1}^{(N+1)} \end{bmatrix} = \begin{bmatrix} -\mathrm{E}_{\min}^{(N+1)} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\Gamma^{(N+1)} = \frac{V^{(N)}}{\mathrm{E}_{\min}^{(N)}} = \frac{r_{xx}(N+1) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\tilde{\boldsymbol{a}}^{(N)}}{r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{a}^{(N)}}$$

$$a_{N+1}^{(N+1)} = \Gamma^{(N+1)}$$

$$\begin{bmatrix} a_1^{(N+1)} \\ a_2^{(N+1)} \\ \vdots \\ a_N^{(N+1)} \end{bmatrix} = \begin{bmatrix} a_1^{(N)} \\ a_2^{(N)} \\ \vdots \\ a_N^{(N)} \end{bmatrix} - \Gamma^{(N+1)} \begin{bmatrix} a_N^{(N)} \\ a_{N-1}^{(N)} \\ \vdots \\ a_1^{(N)} \end{bmatrix}$$

# Levinson-Durbin recursion - Results

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Parcor coefficient, (N+1)th coefficient of the predictor filter of order (N+1):**

$$\Gamma^{(N+1)} = \frac{V^{(N)}}{\mathrm{E}_{\min}^{(N)}} = \frac{r_{xx}(N+1) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\tilde{\boldsymbol{a}}^{(N)}}{r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{a}^{(N)}}$$

$$a_{N+1}^{(N+1)} = \Gamma^{(N+1)}$$

- **All other coefficients can be updated based on the Parcor coefficient and the previous prediction coefficients:**

$$\begin{bmatrix} a_1^{(N+1)} \\ a_2^{(N+1)} \\ \vdots \\ a_N^{(N+1)} \end{bmatrix} = \begin{bmatrix} a_1^{(N)} \\ a_2^{(N)} \\ \vdots \\ a_N^{(N)} \end{bmatrix} - \Gamma^{(N+1)} \begin{bmatrix} a_N^{(N)} \\ a_{N-1}^{(N)} \\ \vdots \\ a_1^{(N)} \end{bmatrix} = \boldsymbol{a}^{(N)} - a_{N+1}^{(N+1)}\tilde{\boldsymbol{a}}^{(N)}$$

with:

$$\boldsymbol{a}^{(N)} = [a_1\, a_2\, \ldots\, a_N]^T$$

$$\tilde{\boldsymbol{a}}^{(N)} = [a_N\, a_{N-1}\, \ldots\, a_1]^T$$

# Levinson-Durbin recursion - Results

- **Update of the prediction error power:**

$$\mathrm{E}_{\min}^{(N+1)} = \mathrm{E}_{\min}^{(N)} - \Gamma^{(N+1)}\, V^{(N)} \quad \textbf{with: } V^{(N)} = \Gamma^{(N+1)}\, \mathrm{E}_{\min}^{(N)}$$

$$\boxed{\mathrm{E}_{\min}^{(N+1)} = \mathrm{E}_{\min}^{(N)}(1 - |\Gamma^{(N+1)}|^2)}$$

The Levinson-Durbin recursion guarantees:

$$|\Gamma^{(N+1)}| \leq 1$$

- **Start of the recursion:**

$$\boxed{\Gamma^{(1)} = \frac{V^{(0)}}{\mathrm{E}_{\min}^{(0)}} = \frac{r_{xx}(1)}{r_{xx}(0)}}$$

$$\boxed{a_1^{(1)} = \Gamma^{(1)} = \frac{r_{xx}(1)}{r_{xx}(0)}}$$

**with:** $\mathrm{E}_{\min} = r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\, \boldsymbol{a}$

$$\boxed{\mathrm{E}_{\min}^{(1)} = r_{xx}(0) - r_{xx}(1)\, a_1^{(1)}}$$

- The property $|\Gamma^{(N+1)}| \leq 1$ guarantees phase minimum prediction error filters (i.e., stable inverse Prediction error filters)
  => Will be shown with lattice structure realization

# Levinson-Durbin recursion - Summary

**Initialization:**

- Predictor: $$a_1^{(1)} \;=\; \tilde{a}_1^{(1)} \;=\; r_{xx}(1)/r_{xx}(0)$$
- Error signal power (minimum): $$E_{\min}^{(0)} \;=\; r_{xx}(0)$$

**Recursion:**

pergunta de prova: Para uma construção de lattice structure, só precisa do reflection coefficient da pra economizar computação no cálculo

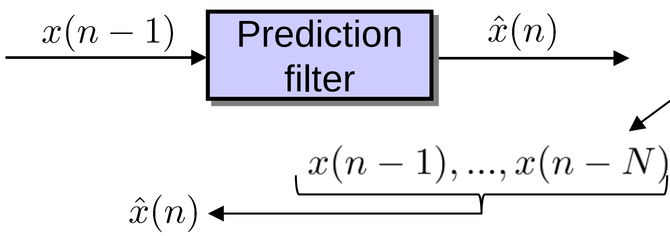
- Reflection coefficient: $$a_{N+1}^{(N+1)} \;=\; \frac{r_{xx}(N+1) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\tilde{\boldsymbol{a}}^{(N)}}{r_{xx}(0) - \boldsymbol{r}_{xx}^{\mathrm{T}}(1)\,\boldsymbol{a}^{(N)}}$$
- Forward prediction: $$[a_1^{(N+1)}\;a_2^{(N+1)}\cdots\;a_N^{(N+1)}]^T = \boldsymbol{a}^{(N)} - a_{N+1}^{(N+1)}\tilde{\boldsymbol{a}}^{(N)}$$
- Backward prediction: $$\tilde{a}_i^{(N)} = a_{N-i}^{(N)}$$
- Error power (minimum): $$\mathrm{E}_{\min}^{(N+1)} = \mathrm{E}_{\min}^{(N)}(1 - |a_{N+1}^{(N+1)}|^2)$$

**Stop criteria:**

- Numeric criterion: When $|a_{N+1}^{(N+1)}|^2 < \epsilon$ , use the previous recursion step and stop the recursion.
- Order criterion: In case $N$ has reached the desired order => recursion stop.

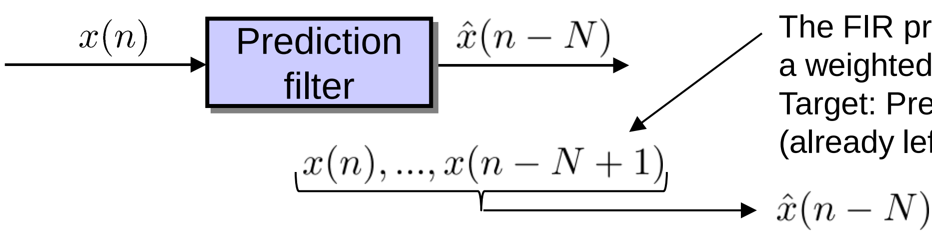
# Backward Prediction: Comparison to forward prediction

- Forward Prediction:




The FIR prediction filter output is
a weighted sum of these elements.
Target: Predict the next sample

- Backward Prediction:

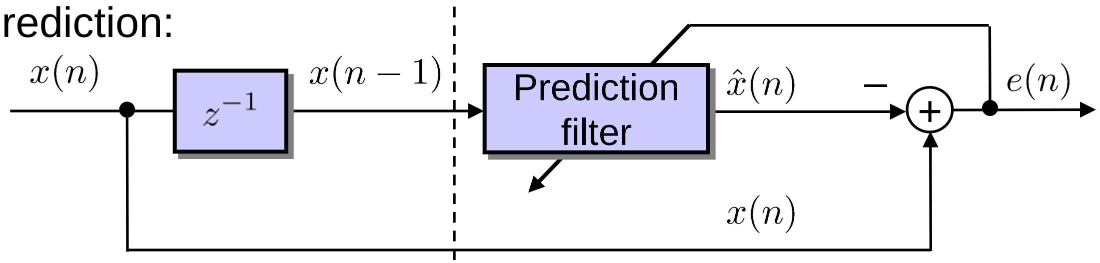



The FIR prediction filter output is
a weighted sum of these elements.
Target: Predict the previous sample
(already left the FIR filter buffer)

# Backward Prediction

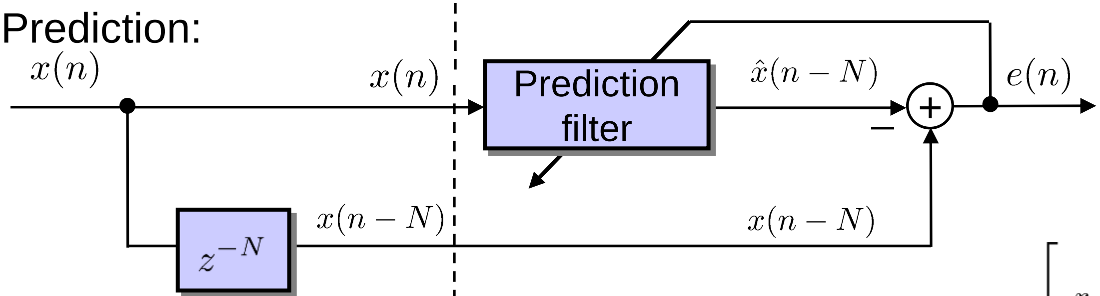


- Forward Prediction:




$$\boldsymbol{h}_{\text{opt}} = \boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(1)$$

- Backward Prediction:




$$\boldsymbol{h}_{\text{opt}} = \boldsymbol{R}_{xx}^{-1}\,\boldsymbol{r}_{xx}(-N)$$

using: $\boldsymbol{r}_{xx}(-N) = \begin{bmatrix} r_{xx}(-N) \\ r_{xx}(-N+1) \\ \vdots \\ r_{xx}(-1) \end{bmatrix} = \begin{bmatrix} r_{xx}(N) \\ r_{xx}(N-1) \\ \vdots \\ r_{xx}(1) \end{bmatrix}$

results in:

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_{r,1} \\ a_{r,2} \\ \vdots \\ a_{r,N} \end{bmatrix} = \begin{bmatrix} r_{xx}(N) \\ r_{xx}(N-1) \\ \vdots \\ r_{xx}(1) \end{bmatrix}$$

# Backward Prediction

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_{r,1} \\ a_{r,2} \\ \vdots \\ a_{r,N} \end{bmatrix} = \begin{bmatrix} r_{xx}(N) \\ r_{xx}(N-1) \\ \vdots \\ r_{xx}(1) \end{bmatrix}$$

Flip the rows and columns:

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_{r,N} \\ a_{r,N-1} \\ \vdots \\ a_{r,1} \end{bmatrix} = \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(N) \end{bmatrix}$$

- Forward Prediction:

$$\boldsymbol{a} = [a_1\, a_2\, \dots\, a_N]^T$$

$$e^{(N)}(n) = x(n) - \sum_{i=1}^{N} a_i^{(N)}\, x(n-i)$$

- Backward Prediction:

$$\boldsymbol{a}_r = \tilde{\boldsymbol{a}} = [a_N\, a_{N-1}\, \dots\, a_1]^T$$

$$\begin{aligned} e_r^{(N)}(n-N) &= x(n-N) - \sum_{i=1}^{N} a_{r,i}^{(N)}\, x(n-i+1) \\ &= x(n-N) - \sum_{i=1}^{N} a_{N+1-i}^{(N)}\, x(n-i+1) \end{aligned}$$

# Backward Prediction: Recursive error calculation

$$\hat{x}^{(N)}(n) = \sum_{i=1}^{N} a_i^{(N)}\, x(n-i)$$

$$= \sum_{i=1}^{N-1} \left(a_i^{(N-1)} - a_N^{(N)} a_{N-i}^{(N-1)}\right) x(n-i) + a_N^{(N)}\, x(n-N)$$

$$= \underbrace{\sum_{i=1}^{N-1} a_i^{(N-1)}\, x(n-i)}_{\substack{\text{Forward prediction}\\ \text{of oder N-1}\\ = \hat{x}^{(N-1)}(n)}} + a_N^{(N)} \underbrace{\left( \underbrace{x(n-N)}_{\substack{\text{additional}\\ \text{signal value}}} - \underbrace{\sum_{i=1}^{N-1} a_{N-i}^{(N-1)} x(n-i)}_{\substack{\text{Backward prediction}\\ \text{of oder N-1}\\ = \hat{x}_r^{(N-1)}(n-N)}} \right)}_{\text{Innovation}}$$

$$= \hat{x}^{(N-1)}(n) + a_N^{(N)} \left( x(n-N) - \hat{x}_r^{(N-1)}(n-N) \right)$$

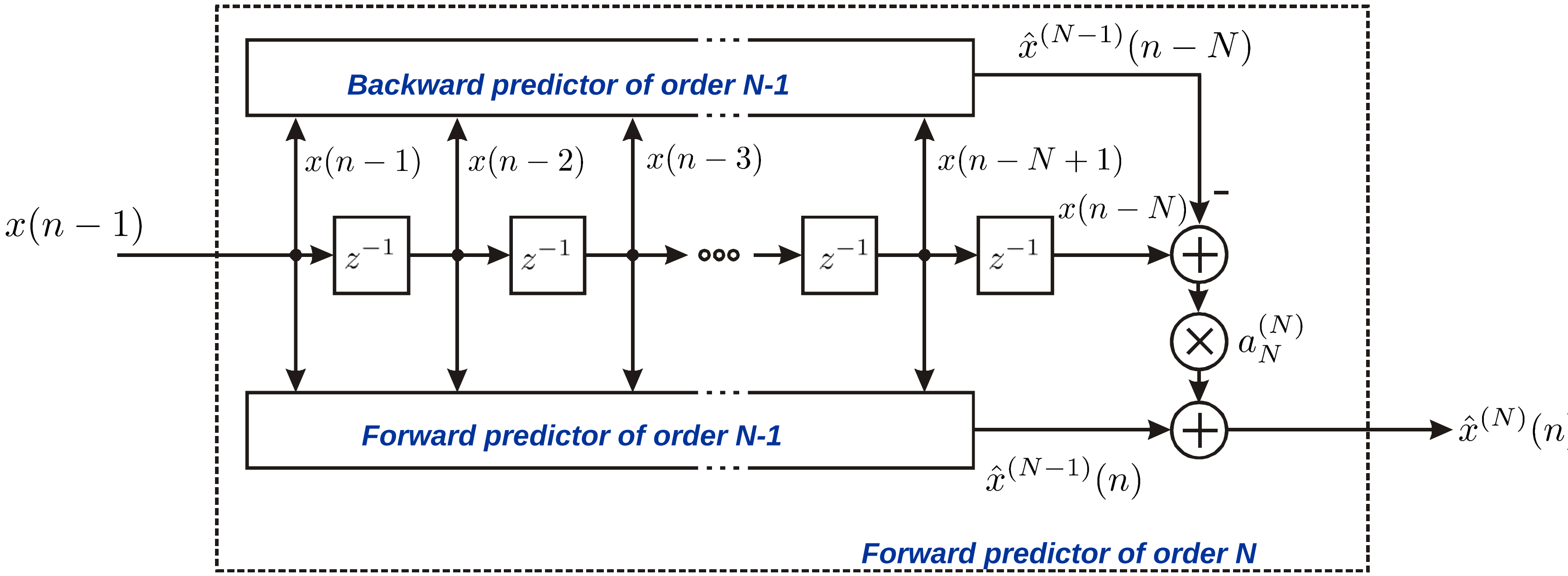
# Backward Prediction: Recursive error calculation

Short Form:

$$\boxed{\hat{x}^{(N)}(n) = \hat{x}^{(N-1)}(n) + a_N^{(N)} \Big( x(n-N) - \hat{x}_r^{(N-1)}(n-N) \Big)}$$

***New estimate = Old estimate + weighting * (new – prediction of new)***

Structure of the order recursion:



# Backward Prediction: Recursive error calculation

$$
\begin{aligned}
e^{(N)}(n) &= x(n) - \hat{x}^{(N)}(n) \\
&= x(n) - \hat{x}^{(N-1)}(n) - a_N^{(N)} \left( x(n-N) - \hat{x}_r^{(N-1)}(n-N) \right) \\
&= e^{(N-1)}(n) \; - \; a_N^{(N)}\, e_r^{(N-1)}(n-N)
\end{aligned}
$$

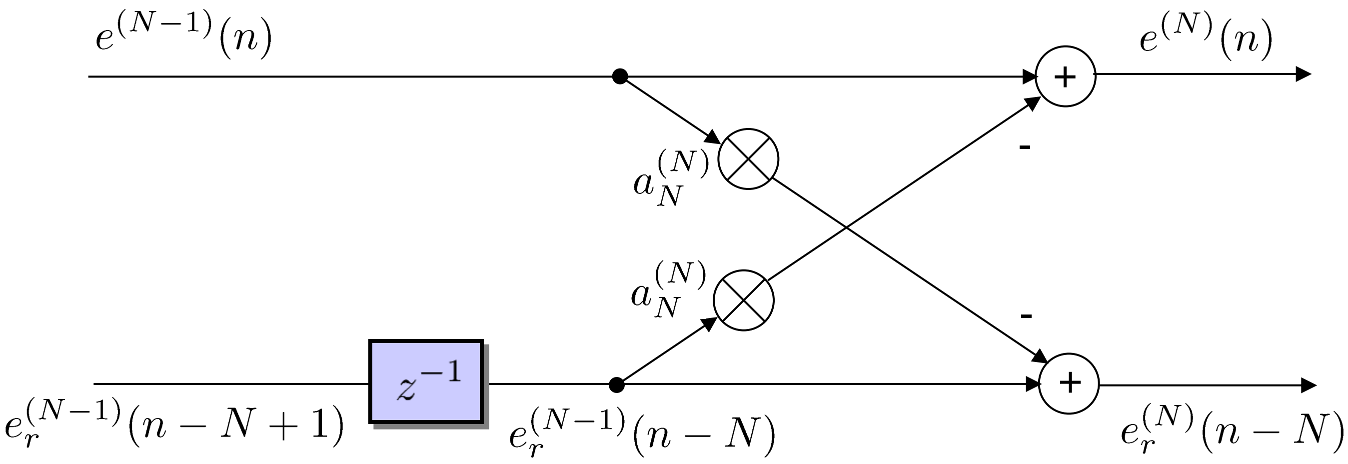
$$
\begin{aligned}
e_r^{(N)}(n-N) &= x(n-N) - \hat{x}^{(N)}(n-N) \\
&= x(n-N) - \sum_{i=1}^{N} a_{N+1-i}^{(N)}\, x(n-i+1) \\
&= x(n-N) - \sum_{i=1}^{N} a_i^{(N)}\, x(n-N+i) \\
&= x(n-N) - \sum_{i=1}^{N-1} [a_i^{(N-1)} - a_N^{(N)}\, a_{N-i}^{(N-1)}]\, x(n-N+i) - a_N^{(N)}\, x(n) \\
&= x(n-N) - \sum_{i=1}^{N-1} a_i^{(N-1)}\, x(n-N+i) - a_N^{(N)} \left[ x(n) - \sum_{i=1}^{N-1} a_{N-i}^{(N-1)}\, x(n-N+i) \right] \\
&= e_r^{(N-1)}(n-N) - a_N^{(N)}\, e^{(N-1)}(n)
\end{aligned}
$$

Flip summation index

# Backward Prediction: Recursive error calculation

- **Previous equations resulting in the following recursion:**

$$
\begin{aligned}
e^{(N)}(n) \;&=\; e^{(N-1)}(n) \;-\; a_N^{(N)}\, e_r^{(N-1)}(n-N) \\
e_r^{(N)}(n-N) \;&=\; e_r^{(N-1)}(n-N) - a_N^{(N)}\, e^{(N-1)}(n)
\end{aligned}
$$



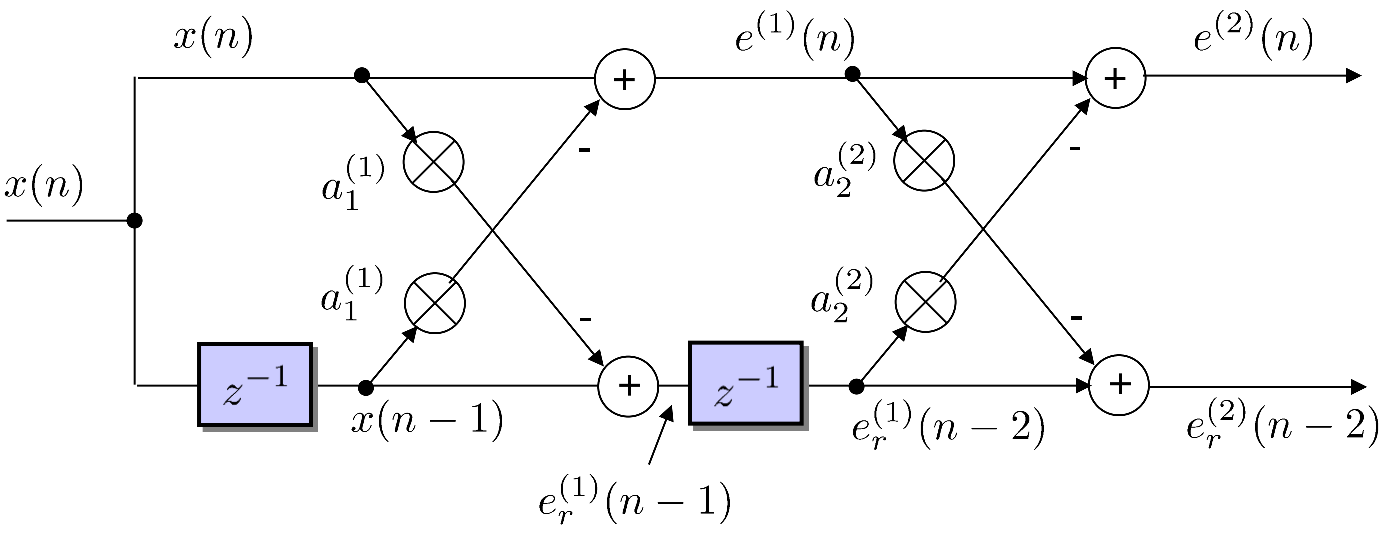
# Lattice filter structure of the prediction error filter

$N = 1:$

$$e^{(1)}(n) \;=\; e^{(0)}(n) \;-\; a_1^{(1)}\, e_r^{(0)}(n-1) \quad\longrightarrow\quad e^{(1)}(n) \;=\; x(n) \;-\; a_1^{(1)}\, x(n-1)$$

$$e_r^{(1)}(n-1) \;=\; e_r^{(0)}(n-1) - a_1^{(1)}\, e^{(0)}(n) \quad\longrightarrow\quad e_r^{(1)}(n-1) \;=\; x(n-1) - a_1^{(1)}\, x(n)$$

- **Lattice structure of the prediction error filter:**



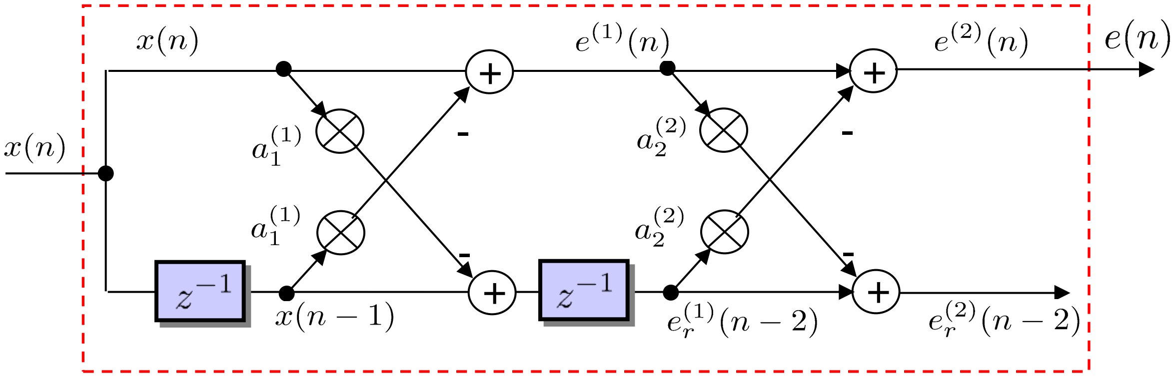
# Different realizations of the prediction error filter

ele parou aqui

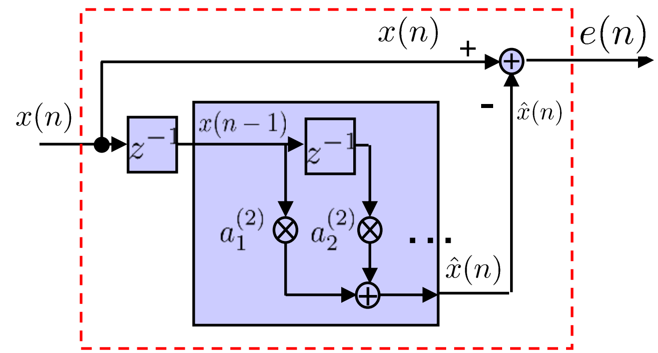
~~PERGUNTA: esse modelo parece pior em termos de hardware pq vc precisa de 2x mais multiplicadores - RESPONDIDO~~

r - vai depender da sua aplicação, mas essencialmente as multiplicações são feitas com SHIFT-AND-ADD

- **Lattice structure:**




- **Normal FIR filter:**




não da pra checar se é minimum phase
não da pra checar se o inverso é estavel

- **Advantage of the lattice structure:**

Since the Parcor coefficients are of magnitude < 1, the stability of the inverse prediction error filter (IIR filter !) is guaranteed

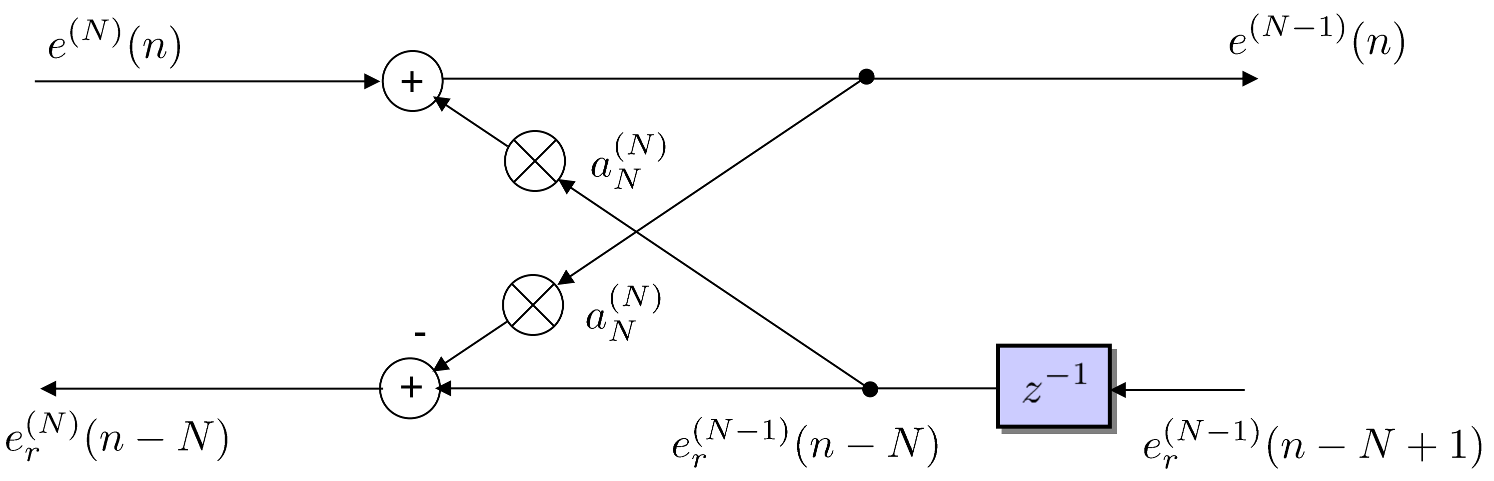
# IIR inverse prediction lattice filter structure:

- **FIR equations:**

$$\begin{aligned} e^{(N)}(n) &= e^{(N-1)}(n) - a_N^{(N)}\, e_r^{(N-1)}(n-N) \\ e_r^{(N)}(n-N) &= e_r^{(N-1)}(n-N) - a_N^{(N)}\, e^{(N-1)}(n) \end{aligned}$$

- **IIR equations:**

$$\begin{aligned} e^{(N-1)}(n) &= e^{(N)}(n) + a_N^{(N)}\, e_r^{(N-1)}(n-N) \\ e_r^{(N)}(n-N) &= e_r^{(N-1)}(n-N) - a_N^{(N)}\, e^{(N-1)}(n) \end{aligned}$$



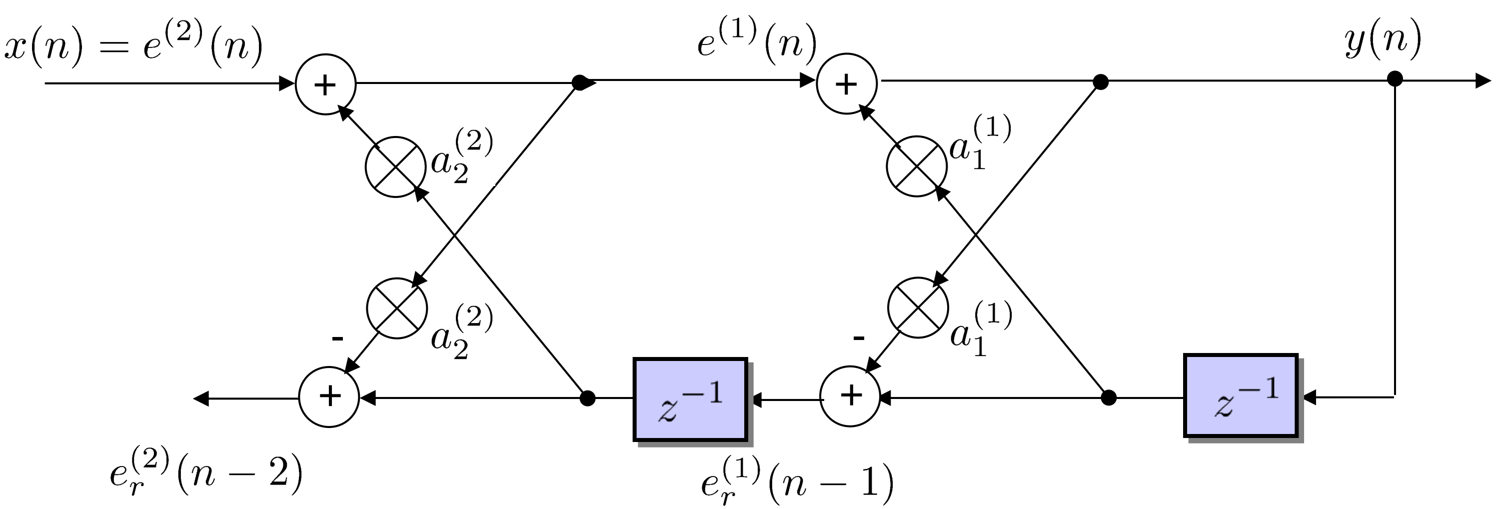
# IIR inverse prediction lattice filter structure

$N = 1:$

$$e^{(0)}(n) = e^{(1)}(n) + a_1^{(1)} e_r^{(0)}(n-1) \longrightarrow y(n) = e^{(1)}(n) + a_1^{(1)} y(n-1)$$

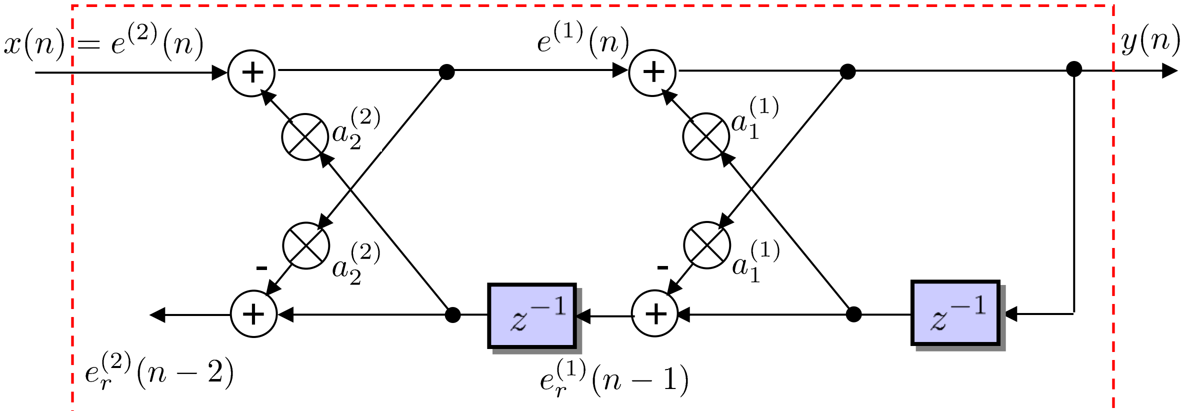
$$e_r^{(1)}(n-1) = e_r^{(0)}(n-1) - a_1^{(1)} e^{(0)}(n) \longrightarrow e_r^{(1)}(n-1) = y(n-1) - a_1^{(1)} y(n)$$

- **IIR filter of second order in lattice structure:**

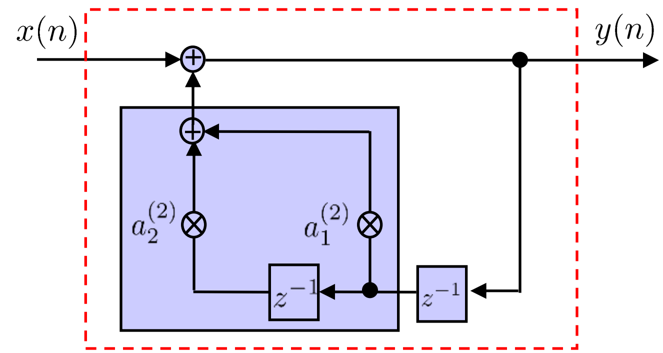


# Different realizations of IIR inverse prediction filters

**Lattice structure:**




**Normal IIR filter:**



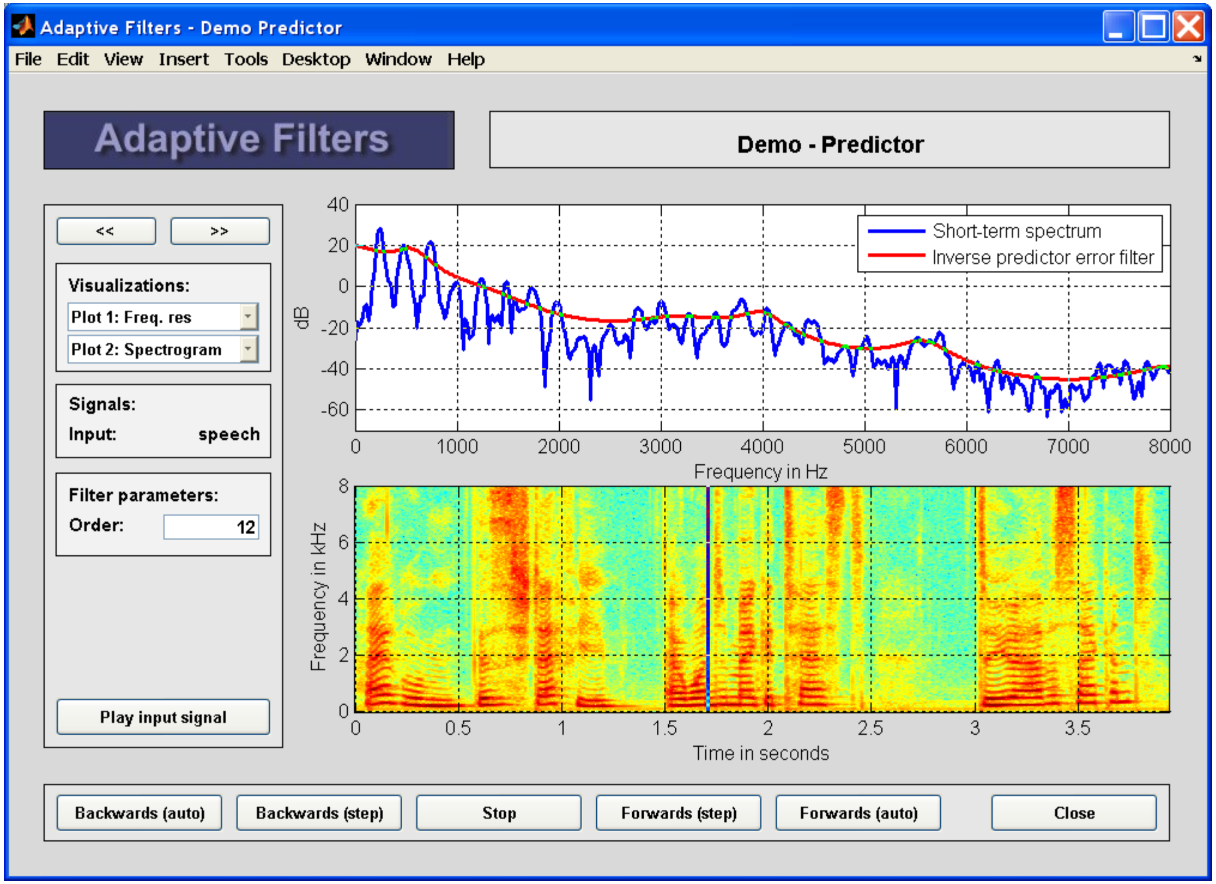

**Advantage of the lattice structure:**

Since the Parcor coefficients are of magnitude < 1, the stability of the IIR filter is guaranteed.

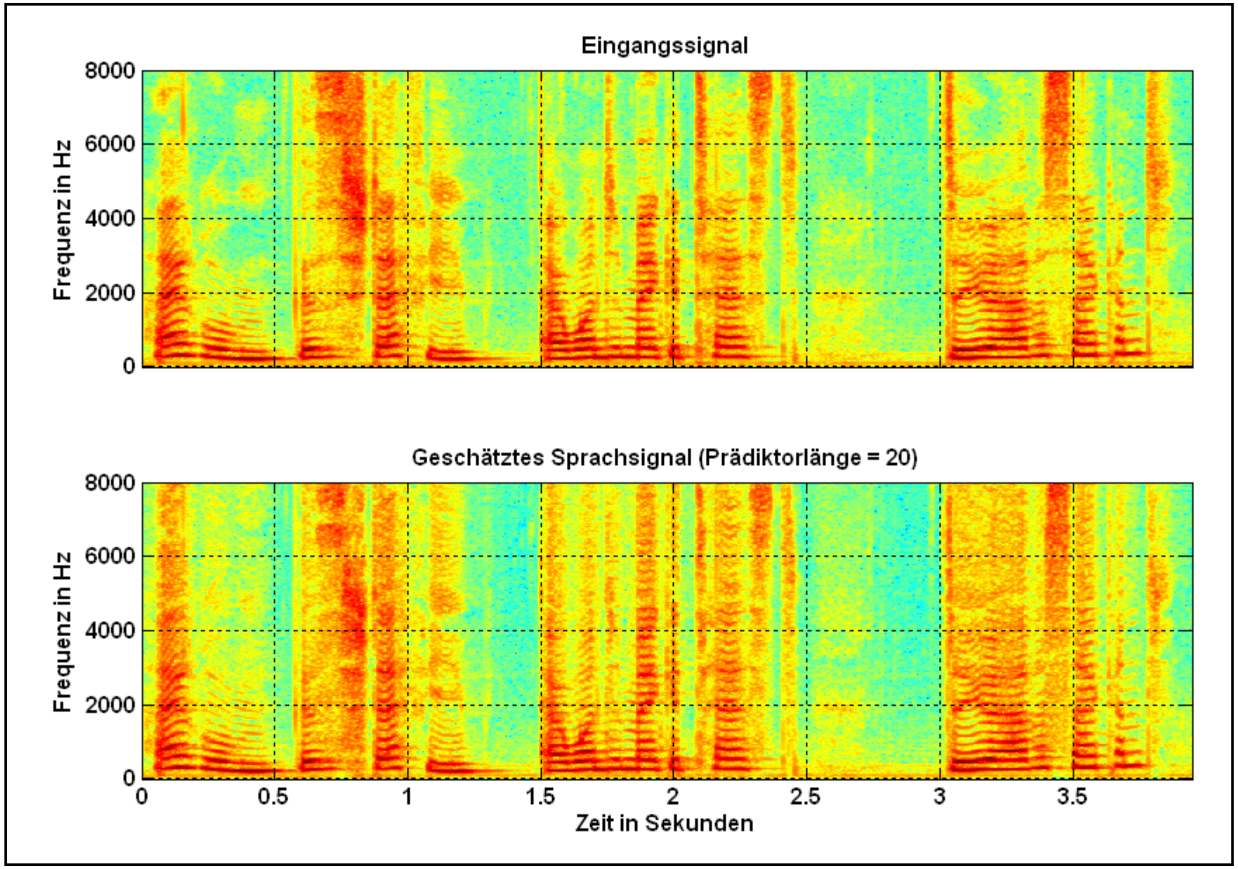
os coefs a(j)_i podem ser maiores que 1, só a)i)_i tem que ter magnitude menor que 1

# Matlab example

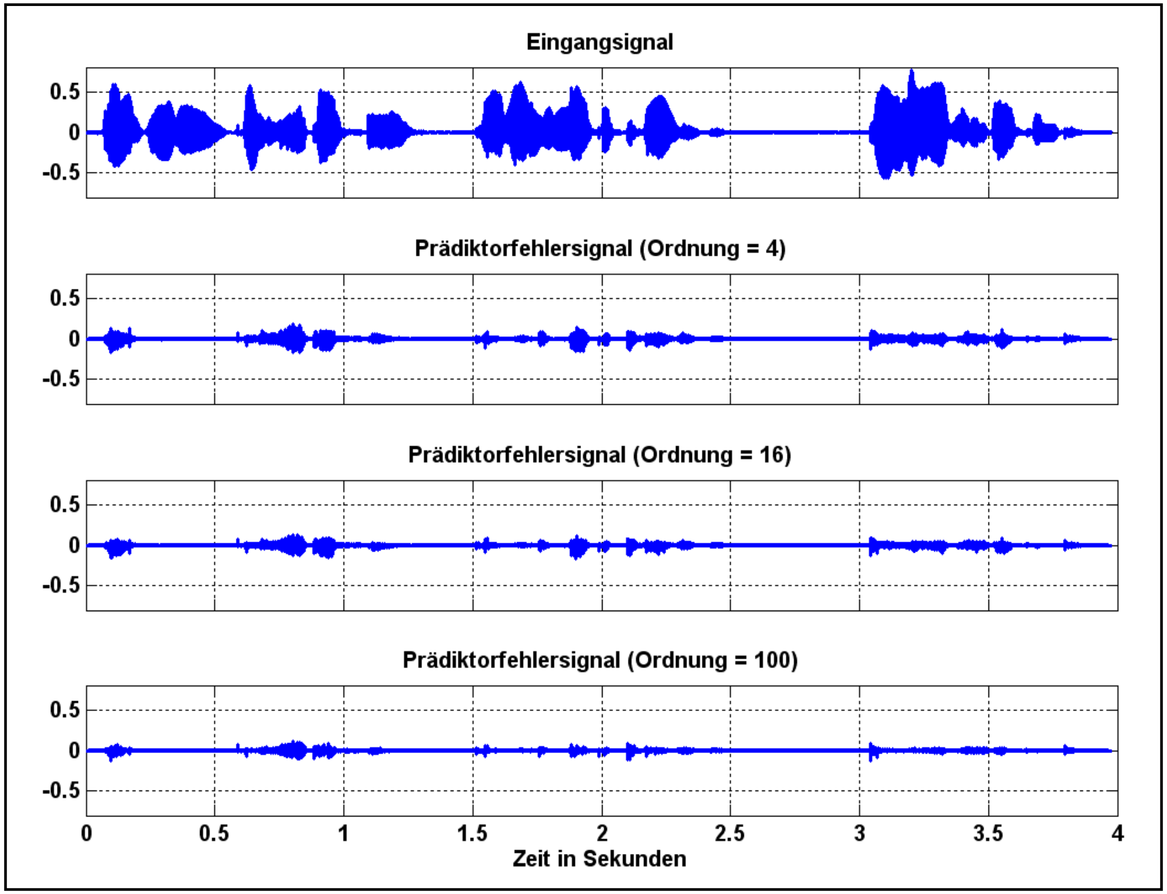
PEF FILTER: [1 -a1 -a2 ...



# Matlab example – estimated speech signal



# Matlab example – prediction error signal



# Summary

***This week:***

- Linear prediction as tool to predict next signal signal sample bases on previous samples => utilizing redundancy
- Application are removal of redundancy for efficient source signal coding and spectral envelope estimation
- The Levinson-Durbin recursion (order recursion!) allows an efficient calculation of the prediction coefficients.

***Next week:***

- Applications of linear prediction