Lesson 1

Optimal Signal Processing

LTH

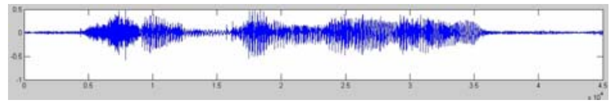
August 2008

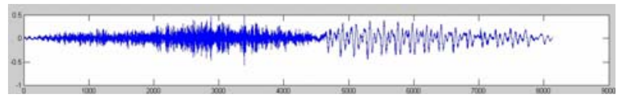
Statistical Digital Signal Processing and Modeling,
Hayes, M:

John Wiley & Sons, 1996. ISBN 0471594318

Bengt Mandersson
Department of Electroscience
Lund University

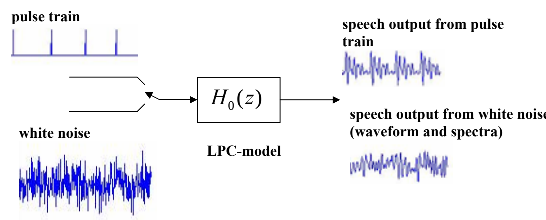
The sound of 'Signalbehandling'





How can this be generated as output from a linear filter?
Determine the filter and the input signal.

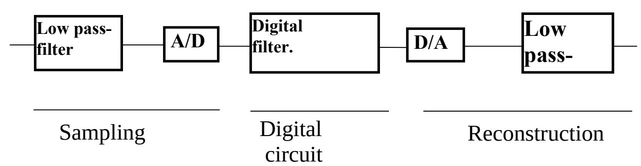
LPC model of syntetic sound production



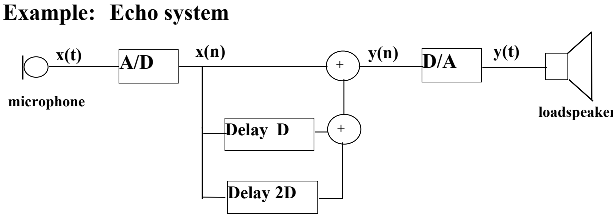
In syntetic speech production, the parameters often are updated every 5 milliseconds.

Chapter 2.	Digital signal processing	
	impulse response, convolution,	
	system function, Fourier, z-transforms	page 7-20
	Matrix description.	page 20-52
	Hints.	page 8-18, 21, 49.
Chapter 3.	Random processing, such as	
	correlation functions, correlation matrices.	
	Random variables	page 58-74
	Random Processes	page 74-119
	Hints.	page 77, 79, 80, 85, 95, 99, 100, 101, 106
Chapter 4.	Signal models, Deterministic and Stochastic approach.	
	Padé, Prony	page 133-154
	Shank	page 154-160
	All-pole Modeling	page 160,165
	Linear prediction	page 165-174
		4.5 not included
	4.6	page 178-188
	4.7 Stochastic Models	page 188-200
	Hints.	page 130, 135, 138, 147, 148,149 195, 195
Chapter 5.	Levinson-Durbin recursion.	page 215-225, 233-241
		page 242 – 276 not included
	Hints.	Table 5.1 – 5.4, figure 5.10
Chapter 6.	Latttice FIR and IIR filters,	
	only 6.2 and 6.4.1, 6.4.3	page 289-293, 297, 298, 304-307
	6.5	page 308-324
Chapter 7.	Optimal filters. Linear prediction.	
	Wiener filters. Specially FIR filters.	
	FIR- Wiener filter	page 335-345
	IIR- Wiener filter	page 353-371
		7.4 not included
	Hints.	page 337-339, 354, 355, 358-363, 370
Chapter 8.	Spectrum estimation.	
	Nonparametric methods	page 393-399, 408-425
	8.3 (8.5 see chap 4) , 8.6	page 426-429, 451-472
	Hints.	page 394, 408

Digital Signal Processing



Example: Echo system

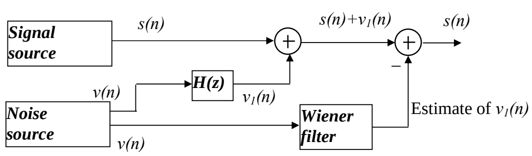


An application from the text book

Noise cancellation (chapter 7, page 349)

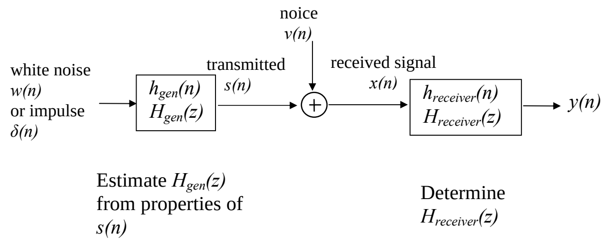
A signal is disturbed by additive noise $v_1(n)$.

Try to measure the noise $v(n)$ from the source and estimate the noise $v_1(n)$ added to the signal. Then subtract the noise $v_1(n)$ from the received signal.



Optimal signal processing in Hay's book

Chapter 2: **Brief review of digital signal processing.**
Chapter 3: **Brief review of random signals.**



The filters $H_{gen}(z)$ and $H_{receiver}(z)$ are of type

FIR
IIR
all-pole IIR

Chapter 4, 5 and 6: **Make a model $H_{gen}(z)$ from the properties of $s(n)$.**

Chapter 7: **Determine $H_{receiver}(z)$.**

Chapter 8: **Estimation of spectra.**

Chapter 2 Digital Signal Processing

Difference equation

$$y(n) = -\sum_{k=1}^{p} a(k)\ y(n-k) + \sum_{k=0}^{q} b(k)x(n-k)$$

MATLAB: A=[1 0.5 0.5]; B=[1 1]; y=filter(B,A,x);

Convolution

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)\, x(n-k)$$

impulse: $\delta(n) = [0\ 0\ 0\ \underset{\uparrow}{1}\ 0\ 0\ 0]$

unit step: $u(n) = [0\ 0\ 0\ \underset{\uparrow}{1}\ 1\ 1\ 1 \ldots]$

System function

$$H(z) = \frac{B(z)}{A(z)}$$

Frequency function

$$H(e^{j\omega}) = \frac{B(e^{j\omega})}{A(e^{j\omega})}$$

FIR, IIR filters

FIR: Circuit with impulse response with finite length

Example

$$y(n) = x(n) + x(n-1), \quad h(n) = \delta(n) + \delta(n-1)$$

IIR: Circuit with impulse response with infinite length

Example

$$y(n) = 0.5\, y(n-1) + x(n), \quad h(n) = 0.5^n\, u(n)$$

All-pole IIR-filters

IIR-filters with poles only (all zeroes in origin, $B(z)$=constant)

Example

$$H(z) = \frac{1}{1 - 0.5\, z^{-1}}$$

Solvning the convolution sum.

$$y(n)=\sum_{k=-\infty}^{\infty} h(k)\,x(n-k)=\sum_{k=-\infty}^{\infty} x(k)\,h(n-k)$$

$$y(n)=\cdots h(0)\,x(n)+h(1)\,x(n-1)+h(2)\,x(n-2)\cdots$$

Example $x(n)=[\underset{\uparrow}{1}\ 2\ 3\ 4],\quad h(n)=[\underset{\uparrow}{4}\ 2\ 2]$

Method A: Vector notation

$$y(n)=\left[h(0)\,h(1)...h(N-1)\right]^T\begin{bmatrix} x(n)\\ x(n-1)\\ .\\ .\\ x(n-N+1)\end{bmatrix}=\underline{h}^T\ \underline{x}(n)$$

Method B: Graphical solution
Write

$x(k):\quad \underset{\uparrow}{1}\ 2\ 3\ 4$

$h(0-k):\quad 2\ 2\ \underset{\uparrow}{4}\qquad y(0)=4\cdot 1=4$

$h(1-k):\quad 2\ \underset{\uparrow}{2}\ 4\qquad y(1)=2\cdot 1+4\cdot 2=10$

Gives the output $y(n)=[\underset{\uparrow}{4}\ 10\ 18\ 26\ 14\ 8]$

MATLAB: x=[1 2 3 4]; h=[4 2 2]; y=conv(x,h)

Method C: Convolution matrix

Use matrix notations

$$x(n)=[\underset{\uparrow}{1}\ 2\ 3\ 4],\quad h(n)=[\underset{\uparrow}{4}\ 2\ 2]$$

$$\begin{bmatrix} x(0) & 0 & 0\\ x(1) & x(0) & 0\\ x(2) & x(1) & x(0)\\ x(3) & x(2) & x(1)\\ 0 & x(3) & x(2)\\ 0 & 0 & x(3)\end{bmatrix}\cdot\begin{bmatrix} h(0)\\ h(1)\\ h(2)\end{bmatrix}=\begin{bmatrix} y(0)\\ y(1)\\ y(2)\\ y(3)\\ y(4)\\ y(5)\end{bmatrix}$$

$$\begin{bmatrix} 1\ 0\ 0\\ 2\ 1\ 0\\ 3\ 2\ 1\\ 4\ 3\ 2\\ 0\ 4\ 3\\ 0\ 0\ 4\end{bmatrix}\cdot\begin{bmatrix} 4\\ 2\\ 2\end{bmatrix}=\begin{bmatrix} 4\\ 10\\ 18\\ 26\\ 14\\ 8\end{bmatrix}\qquad X\,h=y$$

In Matlab: **x=[1 2 3 4]'; X=convmtx(x,3)**
h=[4 2 2]', y=X*h

(In signal processing, all vectors are column vectors)

Properties of matrices

The square matrix $A\ \ (n\times n)$ **is:**

symmetrical if $A=A^T$

Hermitian if $A=(A^T)^*=A^H$

invertable if $AA^{-1}=I$

Toeplitz if all diagonals are identical

$$A=\begin{bmatrix} 3\ 4\ 5\\ 2\ 3\ 4\\ 1\ 2\ 3\end{bmatrix}$$

Hermitian (symmetrical) Toeplitz if

$$A=\begin{bmatrix} 3 & 2 & 1\\ 2 & 3 & 2\\ 1 & 2 & 3\end{bmatrix}\qquad A=Toep[3,2,1]$$

orthogonal if $A^T A\ =I$

Linear equation (page 31-34)

$A\ \ is\ \ a\ \ [n*m]\ \ matrix$

$A\,x=b$ **gives**

$$x=A^{-1}b\qquad if\quad n=m,(A\ \ invertable)$$

$$x=(A^H A)^{-1}A^H\,b\qquad if\quad n>m$$

(overdetermined, more equations than variables.) Described more in chapter 4

$$x=A^H\,(A\,A^H)^{-1}b\quad if\quad n<m$$

(underdetermined, less equations than variables)

Eigenvalue:

$$A\,v=\lambda\,v,\quad (A-\lambda I)=0$$

$$\lambda\ \ eigenvalues,\ \ v\ \ eigenvectors$$

$$A=V\,\Lambda V^{-1}\ \ with\ eigenvectors\ in\ the\ columns\ of\ V,$$
$$eigenvalues\ in\ the\ diagonal\ of\ \Lambda$$

Optimisation (minimizing): (page 49)

If z real: $f(z) = z^2$

$$\frac{d}{dz} f(z) = \frac{d}{dz} z^2 = 2z; \quad \frac{d}{dz} z^2 = 0$$
$$gives \; z = 0 \; as \; \min imum;$$

If z is complex: $f(z) = |z|^2 = z^* z$

$(z^* \; is \; the \; conjugate \; of \; z)$

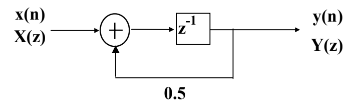
Derivate with respect to z^* *or* z **separately while treating the other as a constant.**

$$\frac{d}{dz} |z^2| = \frac{d}{dz} z^* z = z^*$$
$$\frac{d}{dz^*} |z^2| = \frac{d}{dz^*} z^* z = z$$

Setting this derivatives equal to zero gives the same minimum (page 49). This is used sometimes in the textbook.

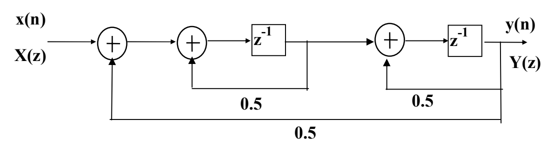
Example on circuits

A

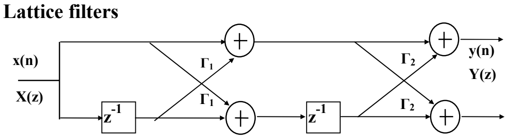


$$y(n) = 0.5\, y(n-1) + x(n-1)$$
$$Y(z) = 0.5 z^{-1}\, Y(z) + z^{-1}\, X(z)$$

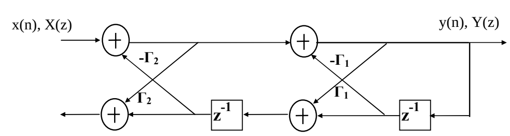
B



C Lattice filters



FIR-lattice filter



IIR-lattice filter

Correlation functions (deterministic)

Autocorrelation function

$$r_x(l) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-l) \quad \left(= r_{xx}(l)\right)$$

Cross-correlation function

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n)\, x(n-l)$$

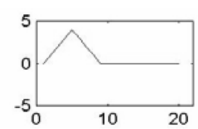
$$r_x(l) = x(l) * x(-l)$$
$$r_{yx}(l) = y(l) * x(-l)$$

Relation between input and output

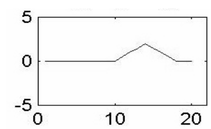
$$r_{yx}(l) = h(l) * r_x(l)$$

$$r_y(l) = r_h(l) * r_x(l)$$

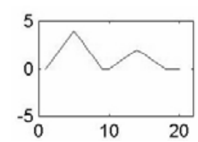
Example on correlation, echo



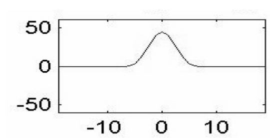
$\mathbf{x_1}$



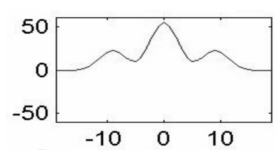
$\mathbf{x_2}$



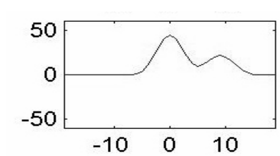
$\mathbf{y = x_1 + x_2}$



$\mathbf{r_{x1}}$

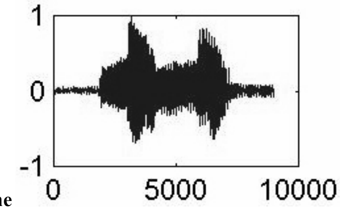


$\mathbf{r_y}$

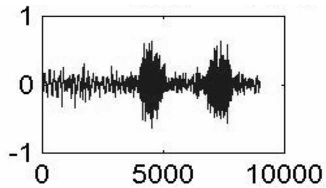


$\mathbf{r_{x1y}}$

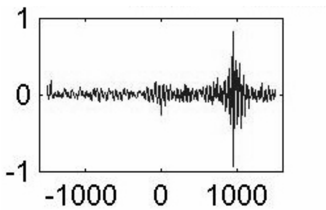
Example of correlation, delay in mobile phones (GSM)



Input signal to the GSM phone



Output signal after GSM



Crosscorrelation

In Matlab: rxy=xcorr(input,output)

Chapter 3 Discrete-Time Random Processes

Random variables (3.2 page 58-74)

Probability density function $f_X(x)$

Probability distribution function: $F_X(x)$

Expected value (mean): $m = E\{x\} = \int x \; f_X(x)\; dx$

Mean-square value: $E\{x^2\} = \int x^2 \; f_X(x)\; dx$

Variance: $Var[x] = \sigma_x^{\,2} = E\{[x-m]^2\} = \int [x-m]^2 \; f_X(x)\; dx$

General: $y = g(x); \quad E\{y\} = E\{g(x)\} = \int g(x)\; f_X(x)\; dx$

Relation: $Var[x] = E\{[x-m]^2\} = E\{X^2\} - m^2$

Correlation. **Dependency between random variables** ***x* and *y***

Correlation: $r_{xy} = E\{x\,y\}$

Covariance: $c_{xy} = E\{[x-m_x][\,y-m_y]\}$

Stochastic processes (3.3 page 74)

(Wide-sense stationary processes, WSS)

Example A: Sinusoids with random phase

$x(n) = A\sin(\omega_0\, n \; + \Phi)$,

Φ **is a random variable and**

$x(n)$ **is a random process.**

Example B: Noise (white noise, colored noise).

Example C: Speech signals.

The autocorrelation sequence and the cross-correlation sequence and their Fourier transforms are important in this course.

Autocorrelation sequence:

$$r_x(m) = E\{x(k)\, x^*(k-m)\}$$

Cross-correlation sequence.

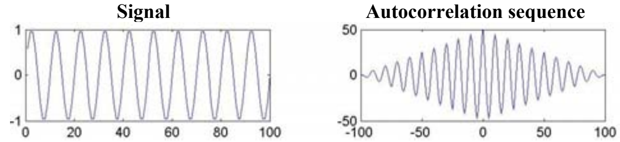
$$r_{xy}(m) = E\{x(k)\, y^*(k-m)\}$$

Estimation of the autocorrelation sequence (ergodic processes)

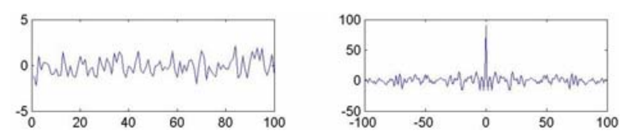
$$r_x(m) = E\{x(k)x(k-m)\} = \frac{1}{N}\sum_{\substack{sum\ over\\ N\, values}} x(k)x(k-m)$$

Interpreting of autocorrelation sequence:

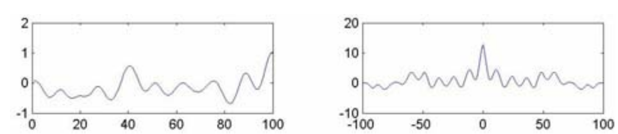
Signal | **Autocorrelation sequence**



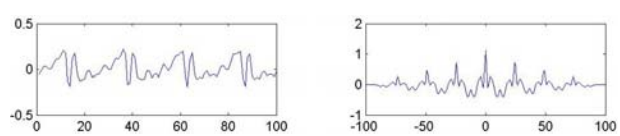
Sinusoid:



White noise.



Colored noise



Speech signal: Vowel.

Properties of autocorrelation sequence (page 83)

(Wide-sense stationary processes, WSS)

Definition:

$$r_x(k) = E\left[x(n)x^*(n-k)\right] = E\left[x^*(n)\ x(n-k)\right]^* =$$
$$= E\left[x(n-k)x^*(n)\right]^* = E\left[x(n)\ x^*(n+k)\right]^* = r_x(k)^*$$

Symmetry:

$$r_x(k) = r_x^*(-k)$$

Mean-square value:

$$r_x(0) = E[|\,x(n)|^2] \geq 0 \qquad (positive)$$

Maximum value:

$$r_x(0) \geq |\,r_x(k)\,|$$

Non-stationary processes

For signals that are not wide-sense stationary processes, (not WSS), we have to use the definitions (see chapter 4)

$$r_x(k,l) = E\{x(k)\ x^*(l)\}$$
$$r_{yx}(k,l) = E\{y(k)\ x^*(l)\}$$

Correlation matrix (WSS)

$$x = [x(0)\ \ x(1) \ldots x(N-1)]^T$$

$$R_x = E[x\,x^H] =$$

$$= \begin{bmatrix} r(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-1) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-2) \\ & & . & & \\ r_x(p) & r_x(p-1) & r_x(p-2) & \cdots & r_x(0) \end{bmatrix}$$

Properties of the correlation matrix

Hermitian Toeplitz
Toeplitz if real-valued process
Eigenvalues are real and non-negative

Estimate of the correlation function

$$\hat{r}_x(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\,x^*(n-k)$$

Estimate of the cross-correlation function

$$\hat{r}_{xy}(k) = \frac{1}{N}\sum_{n=0}^{N-1}\ x(n)\ y^*(n-k)$$

Power spectrum of random process (3.3.8 page 94):

(Wide-sense stationary processes, WSS)

x(n) **is a wide sense stationary random process (WSS,** ***x(n)*** **real-valued,** ***h(n)*** **real) with autocorrelation** $r_x(k)$

The Fourier transform and the z-transform are given by:

The Fourier transform of $r_x(k)$ **:**

$$P_x(e^{j\omega}) = \sum r_x(k)\ e^{-j\omega k}$$

The Z-transform of $r_x(k)$ **:**

$$P_x(z) = \sum r_x(k)\ z^{-k}$$

Properties

Symmetry (real processes)

:

$$P_x(e^{j\omega}) = P_x(e^{-j\omega})$$

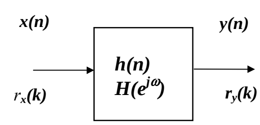
Positive:

$$P_x(e^{j\omega}) \geq 0$$

Total power:

$$r_x(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})\,d\omega$$

Filtering of random processes, (3.4 page 99, 100, 101):



Input-output relation

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Autocorrelation function for the output

$$r_y(k) = E\{y(n)\,y(n-k)\} = \sum_{l=-\infty}^{\infty}\ \sum_{m=-\infty}^{\infty} h(l)\,r_x(m-l+k)\,h(m)$$

Cross correlation functions

$$r_{yx}(k) = E\{y(n)\,x(n-k)\} = \sum_{l=-\infty}^{\infty}\ h(l)\,r_x(k-l)$$
$$r_{xy}(k) = E\{x(n)\,y(n-k\} = \sum_{l=-\infty}^{\infty} h(l)\ r_x(k+l)$$

Using convolution and power spectra

$$Define \quad r_h(k) = \sum h(l)h(l+k) = h(k) * h(-k)$$

Correlation functions

$$r_y(k) = r_x(k) * h(k) * h(-k) = r_x(k) * r_h(k)$$
$$r_{yx}(k) = r_x(k) * h(k)$$
$$r_{xy}(k) = r_x(k) * h(-k)$$

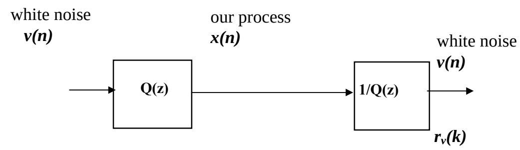
Spectra

$$P_y(e^{j\omega}) = P_x(e^{j\omega})|H(e^{j\omega})|^2$$
$$P_{yx}(e^{j\omega}) = P_x(e^{j\omega})H(e^{j\omega})$$
$$P_{xy}(e^{j\omega}) = P_x(e^{j\omega}) \; H^*(e^{j\omega})$$

$$P_y(z) = P_x(z)\, H(z)\, H(\frac{1}{z})$$

Spectral factorization (3.5 page 104)

***x(n)* is a WSS process with autocorrelation $r_x(k)$. We assume that the process are generated from white noise v(n) filtered in a filter with system function Q(z), Then, *v(n)* is called the innovation process of the process x(n).**



$$r_v(k) = \sigma_0^2 \, \delta(k) \qquad r_x(k)$$
$$P_v(z) = \sigma_0^2 \qquad P_x(z) = \sigma_0^2 Q(z) Q^*(1/z^*)$$

Can we find the filter *Q(z)* from *x(n)* and $r_x(k)$?
Is *Q(z)* stable and causal?
Is *1/Q(z)* stable and causal?

Lesson 2

Chapter 4. Signal Modeling

LTH

August 2008

Bengt Mandersson
Department of Electroscience
Lund University

Chapter 4 Signal modeling

In Chapter 2, we have given a brief review of digital signal processing and some basic matrix definitions.

Then, in chapter 3, the basics of random processes was given, specially autocorrelation sequence, power spectra (power spectral density) and filtering random processes.

Now, we will use our knowledge of random processes to analyze signals which could be described as random processes such as speech signals.

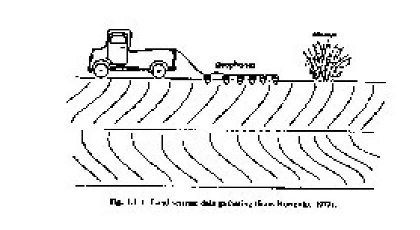
We assume that we have a random process such as speech signals and we want to describe this process in terms of the output from digital filters.

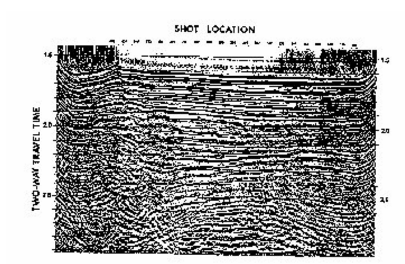
We will have matrix equations and then, in chapter 5, we will describe a well-known algorithm (the Levinson-Durbin algorithm) to solve the equations.

Applications:

- **Speech coding in Mobile phones**
- **Synthetic speech**
- **Seismology**
- **Biomedical applications**
- **Radar**
- **Sonar**
- **Designing optimum filters for noise reduction**

Seismology





Deterministic signals.

Padé	**chapter 4.3 page 134-138**
Prony	**chapter 4.4 page 145-148**
Shanks method	**chapter 4.4.2 page 154-158**
All-pole model	**chapter4.4.3**

Random signals.

All-pole model	**chapter 4.7.2 page 194**

The all-pole model is the most common method and we will concentrate us in the use of this method.

Padés approximation (chap 4.3, page 133 - 141)

Start with the difference equation and let the input be *δ(n)* and the output *x(n)*. Then,

$$x(n)+\sum_{k=1}^{p}a(k)\;x(n-k)=\sum_{k=0}^{q}b(k)\delta(n-k)=b(n)$$

This can be written in matrix forms,

$$\begin{bmatrix} x(0) & 0 & \cdots & 0 \\ x(1) & x(0) & \cdots & 0 \\ & . & & \\ x(q) & x(q-1) & \cdots & x(q-p) \\ \multicolumn{4}{c}{\cdots\cdots\cdots} \\ x(q+1) & x(q) & \cdots & x(q-p+1) \\ & . & & \\ x(q+p) & x(q+p-1) & \cdots & x(q) \\ & . & & \end{bmatrix}\cdot\begin{bmatrix}1\\ a_p(1)\\ a_p(2)\\ .\\ a_p(p)\end{bmatrix}=\begin{bmatrix}b_q(0)\\ b_q(1)\\ .\\ b_q(q)\\ \cdots\\ 0\\ 0\end{bmatrix}$$

We divide the equation in two parts (row *1* to *q* and *q+1* to *q+p*)

$$X\,a_p=b_q \quad \textbf{or} \quad \begin{bmatrix}X_0\\ X_{q+1}\end{bmatrix}\begin{bmatrix}1\\ \bar{a}_p\end{bmatrix}=\begin{bmatrix}b_q\\ 0\end{bmatrix}$$

Now, we use the lower part to determine *a(n)*. Then, we use these values of *a(n)* to determine *b(n)* from the upper part.

We illustrate the method with an example.

Example of Pade's approximation

Use Padé to determine $H(z)$ for $p=2,\ q=2$ for the signal

$$x(n)=n\;0.5^n\;u(n)=[\underset{\uparrow}{0}\;\;1/2\;\;1/2\;\;3/8\;\;1/4\;\;5/32\;\;3/32\;\;7/128\;\;...]$$

We know the system function (use table for z-transform)

$$H(z)=\frac{0.5\;z^{-1}}{1-z^{-1}+0.25\,z^{-2}}$$

We now use method of Pade' and see if we got the same solution.

In matrix form, we have

$$\begin{bmatrix} x(0) & 0 & 0 \\ x(1) & x(0) & 0 \\ x(2) & x(1) & x(0) \\ \multicolumn{3}{c}{\cdots\cdots\cdots} \\ x(3) & x(2) & x(1) \\ x(4) & x(3) & x(2) \\ x(5) & x(4) & x(3) \\ x(6) & x(5) & x(4) \end{bmatrix}\cdot\begin{bmatrix}1\\ a(1)\\ a(2)\end{bmatrix}=\begin{bmatrix}b(0)\\ b(1)\\ b(2)\\ \cdots\\ 0\\ 0\\ 0\\ 0\end{bmatrix}$$

Pade': Use the rows 4 and 5 to solve *a(1),a(2)*, Then rows 1,2,3 to solve *b(0),b(1),b(2.)*

$$\begin{bmatrix} x(3) & x(2) & x(1) \\ x(4) & x(3) & x(2) \end{bmatrix}\cdot\begin{bmatrix}1\\ a(1)\\ a(2)\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix} \Longrightarrow \underbrace{\begin{bmatrix} x(2) & x(1) \\ x(3) & x(2) \end{bmatrix}}_{X_q}\cdot\underbrace{\begin{bmatrix}a(1)\\ a(2)\end{bmatrix}}_{\bar{a}}=-\begin{bmatrix}x(3)\\ x(4)\end{bmatrix}$$

This gives $\begin{bmatrix}1/2 & 1/2\\ 3/8 & 1/2\end{bmatrix}\begin{bmatrix}a(1)\\ a(2)\end{bmatrix}=-\begin{bmatrix}3/8\\ 1/4\end{bmatrix}\Rightarrow\begin{bmatrix}a(1)\\ a(2)\end{bmatrix}=\begin{bmatrix}-1\\ 1/4\end{bmatrix}$

Then, use row 1,2 and 3 to determine *b(n).*

$$b(0)=x(0)=0$$
$$b(1)=x(1)+a(1)x(0)=0.5$$
$$b(2)=x(2)+a(1)x(1)+a(2)x(0)=0$$

which gives the filter $H(z)=\dfrac{0.5\;z^{-1}}{1-z^{-1}+0.25\,z^{-2}}$

Prony's method (chap 4.4, page 144 – 149)

In Pade's approximation, we use a square matrix to determine *a(n)*. If we use more equations, then we got an overdetermined equation system but we know from the first session how to solve this. This method is called Prony's method. We use the same example to illustrate this.

Example of the Prony method.

We restrict us to use 3 rows because we solve it manually.
Then use the row 4,5,6 and solve it as an overdetermined equation system.
The formula for this from chapter 2.

$$A\,x=b\quad(n>m) \Longrightarrow x=(A^H A)^{-1}A^H\,b$$

Now, we use this formula for row 4,5 and 6.

$$X_q=\begin{bmatrix}x(2) & x(1)\\ x(3) & x(2)\\ x(4) & x(3)\end{bmatrix},\quad X_q^T X_q=\begin{bmatrix}x(2) & x(3) & x(4)\\ x(1) & x(2) & x(3)\end{bmatrix}\begin{bmatrix}x(2) & x(1)\\ x(3) & x(2)\\ x(4) & x(3)\end{bmatrix}=\begin{bmatrix}0.45 & 0.53\\ 0.53 & 0.64\end{bmatrix}$$

gives $\begin{bmatrix}a(1)\\ a(2)\end{bmatrix}=-(X_q^T X_q)^{-1}X_q^T\begin{bmatrix}x(3)\\ x(4)\\ x(5)\end{bmatrix}=...=\begin{bmatrix}-1\\ .25\end{bmatrix}$

Then *b(n)* the same as in a), which gives $H(z)=\dfrac{0.5\;z^{-1}}{1-z^{-1}+0.25\,z^{-2}}$.

Comment: The z-transform of *x(n)* can be found in a formula table to be just $H(z)=\dfrac{z^{-1}}{1-z^{-1}+0.25\,z^{-2}}$ **and due to no noise , both methods gives the exact solution.**
The disadvantage of these two methods is that the correlation matrix is not a Toeplitz matrix. Now, we restrict us now to use an all-pole model.

All-pole model. (chap 4.4.3, page 162 – 165)

This is the most common model used in practical applications (synthetic speech, speech coding in mobile phones). We assume that the signal *x(n)* can be modeled as output from an *p*-order all-pole filter.
The difference equation for the input $\delta(n)$ is

$$x(n)+\sum_{k=1}^{p}a_p(k)\;x(n-k)=b(0)\delta(n)$$

and the system function

$$H(z)=\frac{b(0)}{1+a_p(1)z^{-1}+a_p(2)z^{-2}+\cdots+a_p(p)z^{-p}}=\frac{b(0)}{1+\sum_{k=1}^{p}a_p(k)z^{-k}}$$

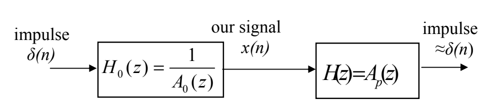
The output should be zero for all $n\neq 0$. We define an error

$$e(n)=x(n)+\sum_{k=1}^{p}a_p(k)\;x(n-k)$$

and we minimize

$$\varepsilon_p=\sum_{n=0}^{\infty}|\,e(n)\,|^2$$

This can be described by the following figure (*b(0)=1*).



$$A_p(z)=1+\sum_{k=1}^{p}a_p(k)z^{-k}$$

The filter $A_p(z)$ is called the predicting error filter (PEF).

We use a least squares solution to solve the problem.
Take the derivative (for simplicity, we assume real valued signals).

$$\frac{\partial \varepsilon_p}{\partial a_p(k)} = \frac{\partial}{\partial a_p(k)} \sum_{n=0}^{\infty} |e(n)|^2 = 2\sum_{n=0}^{\infty} e(n) \frac{\partial}{\partial a_p(k)} e(n) =$$

$$= 2\sum_{n=0}^{\infty} e(n) \frac{\partial}{\partial a_p(k)} [x(n) + \sum_{l=1}^{p} a_p(l)\ x(n-l)] =$$

$$= 2 \underbrace{\sum_{n=0}^{\infty} e(n)x(n-k)}_{e(n) \text{ and given data orthogonal}} = 0 \quad k = 1,2,..,p$$

Then
$$\sum_{n=0}^{\infty} [x(n) + \sum_{l=1}^{p} a_p(l)\ x(n-l)]\, x(n-k) = 0$$

With

$$r_x(k) = \sum_{n=0}^{\infty} x(n)x(n-k)$$

we got the result

$$r(k) + \sum_{l=1}^{p} a_p(l)\ \underbrace{r_x(l-k)}_{r_x(k-l)} = 0$$

or rewritten

$$\sum_{l=1}^{p} a_p(l)\ r_x(k-l) = -r_x(k) \qquad k = 1,...,p$$

This equation is called the normal equation or the Yule-Walker equation.

In matrix form

$$\underbrace{\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3) \\ & & \cdots & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix}}_{R_x} \cdot \underbrace{\begin{bmatrix} a_p(1) \\ a_p(2) \\ a_p(3) \\ \cdot \\ a_p(p) \end{bmatrix}}_{a_p} = - \begin{bmatrix} r_x(1) \\ r_x(2) \\ r_x(3) \\ \cdot \\ r_x(p) \end{bmatrix}$$

Orthogonality principle.

We can derive the filter in a slightly different way.
Writing

$$\varepsilon_p = \sum_{n=0}^{\infty} |e(n)|^2 = \sum_{n=0}^{\infty} e(n)e(n) = \sum_{n=0}^{\infty} e(n)[x(n) + \sum_{k=1}^{p} a_p(k)x(n-k)] =$$

$$= \underbrace{\sum_{n=0}^{\infty} e(n)x(n)}_{\varepsilon_{p,\min} \text{ called model error}} + \sum_{k=1}^{p} a_p(k) \underbrace{\sum_{n=0}^{\infty} e(n)x(n-k)}_{=0,\ e(n) \text{ and given data must be orthogonal}}$$

The minimum error (model error) is now found as

$$\varepsilon_{p,\min} = \varepsilon_p = \sum_{n=0}^{\infty} e(n)x(n) = \sum_{n=0}^{\infty} [x(n) + \sum_{k=1}^{p} a_p(k)x(n-k)]x(n) =$$

$$= r_x(0) + \sum_{k=1}^{p} a(k)\ r_x(k)$$

$$\varepsilon_{p,\min} = \varepsilon_p = r_x(0) + \sum_{k=1}^{p} a(k)\ r_x(k)$$

This equation can be added to the matrix equation described above.
Then, we got (for real signals $r_x^*(k) = r_x(k)$ **)**

$$\underbrace{\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3) \\ & & \cdots & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix}}_{R_x} \cdot \underbrace{\begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \cdot \\ a_p(p) \end{bmatrix}}_{a_p} = \underbrace{\begin{bmatrix} \varepsilon_p \\ 0 \\ 0 \\ \cdot \\ 0 \end{bmatrix}}_{\varepsilon_p u_1}$$

$$R_x\ a_p = \varepsilon_p u_1$$

This is a symmetrical Toeplitz matrix equation system and can be solve with the method described in chapter 5.

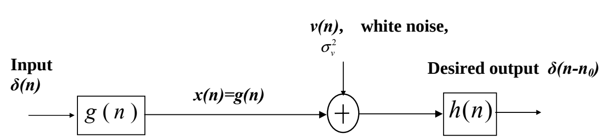
This all-pole model is often called Prediction Error Filter (PEF) or Linear Prediction Coding (LPC).

Shank's method (4.4.2, see Hayes)

Application: FIR Least Squares Inverse Filters: Chap. 4.4.5

Exercise 3, problem 4.19 , Exercise 4 (Computer exercise 1)

The following system is given



The input signal is an impulse,

$$input = \delta(n),$$

and the desired output from our receiver is a delayed version of the input impulse,

$$desired\ \ output = \delta(n-n_0).$$

This means that we want to have the overall impulse response

$$g(n) * h(n) \approx \delta(n-n_0)$$

We define the error signal

$$e(n) = \delta(n-n_0) - g(n) * h(n)$$

Determine the receiver impulse response $h(n)$ **.which we will minimize**

$$\varepsilon_A = \sum_{n=0}^{\infty} e^2(n)$$

Solution: See Exercise 3 and exercise4 (Computer exercise)

Finite Data Records, all-pole modeling (4.6, see Hayes)

The error is defined as

$$\varepsilon_p = \sum_{n=0}^{\infty} |e(n)|^2$$

but $x(n)$ **is known only for n in the interval [0 N], Then,**

$$\varepsilon_p^C = \sum_{n=p}^{N} |e(n)|^2$$

Autocorrelation Method (most common used method)

Determine $r_x(k)$ **assuming** $x(n)=0$ **outside the interval [0 N].**

Exactly as the Prony's all-pole method with the autocorrelation matrix a Toeplitz matrix.

Covariance Method (used sometimes)

Use only values of $x(n)$ **in the interval [0 N]. Like the Prony's method but the autocorrelation matrix is now not a Toeplitz matrix.**

Stochastic model 4.7, All-pole model 4.7.2 page 194.

An all-pole stochastic model is called an autoregressive model (AR). The equations are identical to the all-pole model we had before. The only difference is the definition of the autocorrelation sequence.

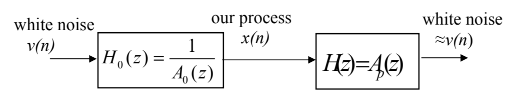
$$\sum_{l=1}^{p} a_p(l)\ r_x(n-l) = -r_x(k) \qquad k = 1,\ldots,p$$

$$r_x(k) = E\{x(n)\,x(n-l)\}$$

The minimum error (model error) is

$$\varepsilon_{p,\min} = \varepsilon_p = r_x(0) + \sum_{k=1}^{p} a(k)\ r_x(k)$$

We now write the model as (predicting error filter, PEF)



$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k)\, z^{-k}$$

Stochastic model with both poles and zeroes. ARMA-model (4.7 page 189.

The solution with both poles and zeroes is more difficult.
Solve the problem in 2 steps like before (first $a_p(k)$**, then** $b_q(k)$ **)**

The differential equation is $(white\ \ input\ \ noise\ with\ \ \sigma_v^2 = 1)$

$$x(n) + \sum_{l=1}^{p} a_p(l) x(n-l) = \sum_{l=0}^{q} b_q(l)\, v(n-l)$$

Multiply with $x^*(n-k)$ **and take** $E\{\ldots\}$ **gives** $(a_p(0)=1)$

$$\sum_{l=0}^{p} a_p(l) r_x(k-l) = \underbrace{\sum_{l=0}^{q} b_q(l) \quad \underbrace{r_{vx}(k-l)}_{r_{vx}(k-l)=h^*(l-k)\,\sigma_v^2}}_{c_q(k)}$$

The right side is $(l \ge k)$

$$c_q(k) = \sum_{l=k}^{q} b_q(l) h^*(l-k)$$

This gives the equations

$$\sum_{l=0}^{p} a_p(l)\ r_x(k-l) = \begin{cases} c_q(k) & 0 \le k \le q \\ 0 & k > q \end{cases}$$

or in matrix form

$$\begin{bmatrix} R_a \\ R_b \end{bmatrix} a_p = \begin{bmatrix} c_q \\ 0 \end{bmatrix}$$

- **Determine** a_p **from the lower part, then** c_q **from the upper part.**
- **From** c_q **back to** b_q **we use spectral factorization.**

From chapter 3, we have

$$P_x(z) = \sigma_v^2 \, Q(z) \; Q^*(\frac{1}{z^*})$$

Take the transform of YWE:

$$A_p(z)\, P_x(z) = C_q(z) = B_q(z) \;\; H^*(\frac{1}{z^*})$$

$$A_p(z)\, P_x(z) = C_q(z) = B_q(z) \;\; \frac{B_q^{\ *}(1/z^*)}{A_p^{\ *}(1/z^*)}$$

An finally

$$C_q(z)\, A_p^{\ *}(1/z^*) = B_q(z)\, B_q^{\ *}(1/z^*)$$

The left side is known and hopefully we can identify the factors in the right side.

Example - ARMA model

Problem: We will estimate a first order ARMA model from

$$r_x(0) = 3, \;\; r_x(1) = 2, \;\; r_x(2) = 1,$$

Solution: We have $p = q = 1$ **which gives the equations**

$$\begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \end{bmatrix} = \begin{bmatrix} c_1(0) \\ c_1(1) \\ 0 \end{bmatrix}$$

Then, we found $a_1 = -1/2$ **from the lower equation**

and

$$\begin{bmatrix} c_1(0) \\ c_1(1) \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ -1/2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1/2 \end{bmatrix}$$

$$\begin{pmatrix} c_1(k) = 0 & k \geq 2 \\ c_1(k) & k < 0 \quad not \quad used \end{pmatrix}$$

Now, we have to identify $B_q(z)$

We have

$$C_q(z)\, A_p^{\ *}(1/z^*) = \underbrace{\underbrace{(\ldots + 2 + \frac{1}{2} z^{-1})}_{C_q(z)} \underbrace{(1 - \frac{1}{2} z)}_{A_p^{\ *}(1/z^*)}}_{\ldots \; \frac{7}{4} + \frac{1}{2} z^{-1}}$$

But

$B_q(z)\, B_q^{\ *}(1/z^*)$ **must be symmetrical so we can write**

$$B_q(z) B_q^{\ *}(1/z^*) = \frac{1}{2} z + \frac{7}{4} + \frac{1}{2} z^{-1} =$$

$$= (c_1 + c_2 z^{-1})(c_1 + c_2 z) =$$

$$= \begin{cases} (1.26 + 0.4 z^{-1})(1.26 + 0.4 z) & minimum \;\; phase \\ (0.4 + 1.26 z^{-1})(0.4 + 1.26 z) & not \;\; minimum \; phase \end{cases}$$

which gives the filter (choose minimum phase)

$$H(z) = \frac{1.26 + 0.4 z^{-1}}{1 - \frac{1}{2} z^{-1}} = 1.26 \;\; \frac{1 + 0.31 z^{-1}}{1 - \frac{1}{2} z^{-1}}$$

Lesson 3

Chapter 5.
Levinson-Durbin Recursion

LTH

September 2008

Bengt Mandersson
Department of Electroscience
Lund University

Chapter 5 Levinson-Durbin recursion

In chapter 4, we derive the normal equations or Yule-walker equations for an all-pole model (chap 4.4.3, page 162 – 165).

The difference equation for the input $\delta(n)$ **is**

$$x(n)+\sum_{k=1}^{p}a_p(k)\;x(n-k)=b(0)\delta(n)$$

and the system function

$$H(z)=\frac{b(0)}{1+a_p(1)z^{-1}+a_p(2)z^{-2}+\cdots+a_p(p)z^{-p}}=\frac{b(0)}{1+\sum_{k=1}^{p}a_p(k)z^{-k}}$$

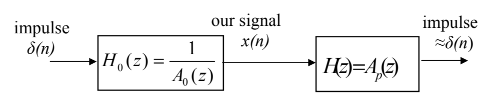
The output should be zero for all $n\neq 0$. **We define an error**

$$e(n)=x(n)+\sum_{k=1}^{p}a_p(k)\;x(n-k)$$

and we minimize

$$\varepsilon_p=\sum_{n=0}^{\infty}|\,e(n)\,|^2$$

This can be described by the following figure (*b(0)=1*).



$$A_p(z)=1+\sum_{k=1}^{p}\;a_p(k)\,z^{-k}$$

The solution derived in chapter 4 is

$$\sum_{l=1}^{p}a_p(l)\;r_x(k-l)=-r_x(k)\qquad k=1,...,p$$

In matrix form

$$\underbrace{\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1)\\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2)\\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3)\\ & & \cdots & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0)\end{bmatrix}}_{R_x}\cdot\underbrace{\begin{bmatrix} a_p(1)\\ a_p(2)\\ a_p(3)\\ .\\ a_p(p)\end{bmatrix}}_{a_p}=-\begin{bmatrix} r_x(1)\\ r_x(2)\\ r_x(3)\\ .\\ r_x(p)\end{bmatrix}$$

The optimal coefficients can be found just by inverting the correlation matrix. The value of resulting minimum cost function was

$$\varepsilon_{p,\min}=\varepsilon_p=r_x(0)+\sum_{k=1}^{p}a(k)\;r_x\;(k)$$

The minimum cost is decreasing if the order p increases and can be used to chose an appropriate the value of the order p.

Combining the two equations into one matrix equation gives

$$\underbrace{\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1)\\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2)\\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3)\\ & & \cdots & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0)\end{bmatrix}}_{R_x}\cdot\underbrace{\begin{bmatrix} 1\\ a_p(1)\\ a_p(2)\\ .\\ a_p(p)\end{bmatrix}}_{a_p}=\underbrace{\begin{bmatrix} \varepsilon_p\\ 0\\ 0\\ .\\ 0\end{bmatrix}}_{\varepsilon_p u_1}$$

$$R_x\;\;a_p\;=\varepsilon_p u_1$$

We will now derive an iterative solution this equations.

Levinson-Durbin recursion

The autocorrelation matrix for this system is Hermitian Toeplitz (symmetrical Toeplitz for real valued signals). For solving this types of matrix equation, a very well known algorithm is the Levinson-Durbin recursion. We assume here real valued signals (for complex signal, see the textbook).

We start with the normal equation from chapter 4 (page 216-219)

$$r_x(k)+\sum_{l=1}^{p}a_p(l)\;r_k(l-k)=0;\qquad k=1,2\ldots p$$

and the error

$$\varepsilon_p=r_x(0)+\sum_{l=1}^{p}a(l)\;r_x(l)$$

In matrix form (index *p* denotes the order of the filter)

$$\underbrace{\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(p)\\ r_x(1) & r_x(0) & r_x(1) & \cdots & r_x(p-1)\\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x(p-1)\\ & & \cdots & & \\ r_x(p) & r_x(p-1) & r_x(p-2) & \cdots & r_x(0)\end{bmatrix}}_{R_p}\cdot\underbrace{\begin{bmatrix} 1\\ a_p(1)\\ a_p(2)\\ .\\ a_p(p)\end{bmatrix}}_{a_p}=\underbrace{\begin{bmatrix} \varepsilon_p\\ 0\\ 0\\ .\\ 0\end{bmatrix}}_{\varepsilon_p u_1}$$

$$R_p\;a_p=\varepsilon_p u_1$$

Now we will derive an algorithm to solve this iteratively . The idea is to solve it in a recursive procedure starting a_1, then a_2, a_3, up to a_p.

Then, in step j we have

$$\underbrace{\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(j) \\ r_x(1) & r_x(0) & r_x(1) & \cdots & r_x(j-1) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x(j-2) \\ & & \cdots & & \\ r_x(j) & r_x(j-1) & r_x(j-2) & \cdots & r_x(0) \end{bmatrix}}_{R_j} \cdot \underbrace{\begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ . \\ a_j(j) \end{bmatrix}}_{a_j} = \underbrace{\begin{bmatrix} \varepsilon_j \\ 0 \\ 0 \\ . \\ 0 \end{bmatrix}}_{\varepsilon_j u_1}$$

$$R_j\, a_j = \varepsilon_j u_1$$

Add one row and one column including the following equation

$$\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j+1-i)$$

The new matrix is

$$\underbrace{\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(j+1) \\ r_x(1) & r_x(0) & r_x(1) & \cdots & r_x(j) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x(j-1) \\ & & \cdots & & \\ r_x(j) & r_x(j-1) & r_x(j-2) & \cdots & r_x(1) \\ r_x(j+1) & r_x(j) & r_x(j-1) & \cdots & r_x(0) \end{bmatrix}}_{R_{j+1}} \cdot \underbrace{\begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ . \\ a_j(j) \\ 0 \end{bmatrix}} = \underbrace{\begin{bmatrix} \varepsilon_j \\ 0 \\ 0 \\ . \\ 0 \\ \gamma_j \end{bmatrix}}$$

Use the symmetry to write this as

$(R_j = R_j^T,\ symmetrical)$

$$\underbrace{\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(j+1) \\ r_x(1) & r_x(0) & r_x(1) & \cdots & r_x(j) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x(j-1) \\ & & \cdots & & \\ r_x(j) & r_x(j-1) & r_x(j-2) & \cdots & r_x(1) \\ r_x(j+1) & r_x(j) & r_x(j-1) & \cdots & r_x(0) \end{bmatrix}} \cdot \underbrace{\begin{bmatrix} 0 \\ a_j(j) \\ a_j(j-1) \\ . \\ a_j(1) \\ 1 \end{bmatrix}} = \underbrace{\begin{bmatrix} \gamma_j \\ 0 \\ 0 \\ . \\ 0 \\ \varepsilon_j \end{bmatrix}}$$

Now, make a linear combination of these two equations

$$R_{j+1} \cdot \underbrace{\left\{ \underbrace{\begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ . \\ a_j(j) \\ 0 \end{bmatrix}} + \Gamma_{j+1} \begin{bmatrix} 0 \\ a_j(j) \\ a_j(j-1) \\ . \\ a_j(1) \\ 1 \end{bmatrix} \right\}}_{a_{j+1}} = \underbrace{\underbrace{\begin{bmatrix} \varepsilon_j \\ 0 \\ 0 \\ . \\ 0 \\ \gamma_j \end{bmatrix}} + \Gamma_{j+1} \begin{bmatrix} \gamma_j \\ 0 \\ 0 \\ . \\ 0 \\ \varepsilon_j \end{bmatrix}}_{\varepsilon_{j+1} u_1}$$

This new matrix equation must satisfy

$$R_{j+1}\, a_{j+1} = \varepsilon_{j+1} u_1$$

Then, the lowest element in the vector on the right side must be zero.

Then we got

$$\gamma_j + \Gamma_{j+1}\ \varepsilon_j = 0$$

$$\Gamma_{j+1} = -\frac{\gamma_j}{\varepsilon_j}$$

and also $\varepsilon_{j+1} = \varepsilon_j + \Gamma_{j+1}\ \gamma_j = \varepsilon_j\,(1 - |\,\Gamma_{j+1}|^2)$

This results in the update equation for *a*

$$\begin{bmatrix} 1 \\ a_{j+1}(1) \\ a_{j+1}(2) \\ . \\ a_{j+1}(j) \\ a_{j+1}(j+1) \end{bmatrix} = \begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ . \\ a_j(j) \\ 0 \end{bmatrix} + \Gamma_{j+1} \begin{bmatrix} 0 \\ a_j(j) \\ a_j(j-1) \\ . \\ a_j(1) \\ 1 \end{bmatrix}$$

Alternatively, we can write

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1}\, a_j(j-i+1) \qquad i = 1, 2, \ldots, j+1$$

Note that

$$a_j(0) = 1$$

$$a_j(j+1) = 0$$

$$a_{j+1}(j+1) = \Gamma_{j+1}$$

$$\varepsilon_0 = r_x(0)$$

This algorithm is easy to implement in a computer program. This is shown in table 5.1 in the textbook. The parameters Γ_j **are called the reflection parameters. For stable filters (all poles inside the unit circle), the reflection parameters are bounded by**

$$|\,\Gamma_j\,| < 1.$$

Table 5.1 *The Levinson-Durbin Recursion*

1. Initialize the recursion
   (a) $a_0(0) = 1$
   (b) $\epsilon_0 = r_x(0)$
2. For $j = 0, 1, \ldots, p-1$
   (a) $\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j-i+1)$
   (b) $\Gamma_{j+1} = -\gamma_j / \epsilon_j$
   (c) For $i = 1, 2, \ldots, j$
   $$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$$
   (d) $a_{j+1}(j+1) = \Gamma_{j+1}$
   (e) $\epsilon_{j+1} = \epsilon_j \left[1 - |\Gamma_{j+1}|^2\right]$
3. $b(0) = \sqrt{\epsilon_p}$

The relation between a_j and Γ_j can be written as (page 234)

$$a_0 = 1$$

$$a_1 = \begin{bmatrix} a_1(0) \\ a_1(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \Gamma_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \Gamma_1 \end{bmatrix}$$

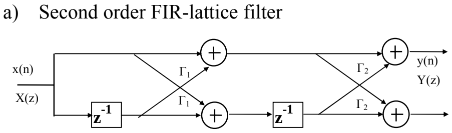
$$a_2 = \begin{bmatrix} a_2(0) \\ a_2(1) \\ a_2(2) \end{bmatrix} = \begin{bmatrix} 1 \\ a_1(1) \\ 0 \end{bmatrix} + \Gamma_2 \begin{bmatrix} 0 \\ a_1(1) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \Gamma_1 \\ 0 \end{bmatrix} + \Gamma_2 \begin{bmatrix} 0 \\ \Gamma_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \Gamma_1 + \Gamma_1 \cdot \Gamma_2 \\ \Gamma_2 \end{bmatrix}$$

$$a_3 = \begin{bmatrix} a_3(0) \\ a_3(1) \\ a_3(2) \\ a_3(2) \end{bmatrix} = \begin{bmatrix} 1 \\ a_2(1) \\ a_2(2) \\ 0 \end{bmatrix} + \Gamma_3 \begin{bmatrix} 0 \\ a_2(2) \\ a_2(1) \\ 1 \end{bmatrix}$$

Lattice filter

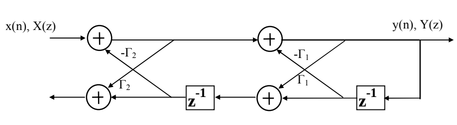
The parameters Γ_j can be interpreted in a specific structure of digital filters, called the lattice filters (see page 225 and chapter 6 and also homework 1).

a) Second order FIR-lattice filter



$$H(z) = 1 + (\Gamma_1 + \Gamma_1\Gamma_2) z^{-1} + \Gamma_2 z^{-2}$$

b) Second order IIR-lattice filter



$$H(z) = \frac{1}{1 + (\Gamma_1 + \Gamma_1\Gamma_2) z^{-1} + \Gamma_2 z^{-2}}$$

Relation between the polynomial a_j and the reflection coefficients Γ_j using z-transform

We have $\quad a_j = \begin{bmatrix} 1 & a_j(1) & a_j(2) \cdots a_j(j) \end{bmatrix}^T$

Then define $\quad a_j^R = \begin{bmatrix} a_j(j) & \cdots & a_j(2) & a_j(1) & 1 \end{bmatrix}^T$

Then we can write the update equation (page 224, 235, 236)

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} \, a_j^R(i-1)$$

Make a variable substitution $i \Rightarrow j - i + 1$ **gives**

$$\underbrace{a_{j+1}(j-i+1)}_{a_{j+1}^R(i)} = \underbrace{a_j(j-i+1)}_{a_j^R(i-1)} + \Gamma_{j+1} \underbrace{a_j^R(j-i)}_{a_j(i)}$$

Taking the z-transform of these equations gives

Forwards: (from gamma to polynomial)

$$\begin{cases} A_{j+1}(z) = A_j(z) + z^{-1}\,\Gamma_{j+1}\, A_j^R(z) \\ A_{j+1}^R(z) = z^{-1}\, A_j^R + \Gamma_{j+1}\, A_j(z) \end{cases}$$

In matrix form this can be written (page 224 and page 236

$$\begin{bmatrix} A_{j+1}(z) \\ A_{j+1}^R(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{-1}\,\Gamma_{j+1} \\ \Gamma_{j+1} & z^{-1} \end{bmatrix} \begin{bmatrix} A_j(z) \\ A_j^R(z) \end{bmatrix}$$

Backwards: (From polynomial to gamma)

$$\begin{bmatrix} A_j(z) \\ A_j^R(z) \end{bmatrix} = \frac{1}{z^{-1}(1-|\Gamma_{j+1}|^2)} \begin{bmatrix} z^{-1} & -z^{-1}\,\Gamma_{j+1} \\ -\Gamma_{j+1} & 1 \end{bmatrix} \begin{bmatrix} A_{j+1}(z) \\ A_{j+1}^R(z) \end{bmatrix}$$

$$A_j(z) = \frac{1}{(1-|\Gamma_{j+1}|^2)}\left[A_{j+1}(z) - \Gamma_{j+1} A_{j+1}^R(z)\right]$$

Backwards ireratively

If the filter is given by $a_p(n)$, the reflection parameters Γ_j can be determined, see textbook page 235, page 236, table 5.3.

The step down recursion (see table 5.3)

Identify $\Gamma_p = a_p(p)$

Loop j=p-1, p-2, ... 1

Then, determine a_j from a_{j+1}

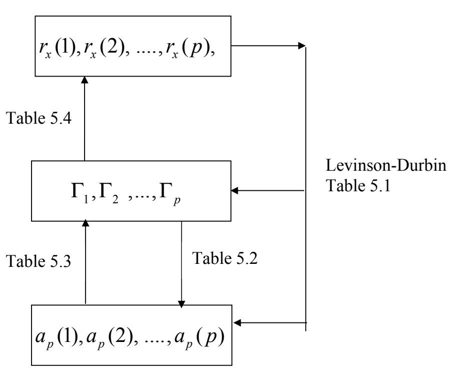
$$a_j(i) = \frac{1}{1-\Gamma_{j+1}^2}\left[a_{j+1}(i) - \Gamma_{j+1}\, a_{j+1}(j-i+1)\right] \quad i = 1,2,\dots,j$$

Identify $\Gamma_j = a_j(j)$

Table 5.3 ***The Step-down Recursion***

1. Set $\Gamma_p = a_p(p)$
2. For $j = p-1, p-2, \dots, 1$
   (a) For $i = 1, 2, \dots, j$
   $$a_j(i) = \frac{1}{1-|\Gamma_{j+1}|^2}\left[a_{j+1}(i) - \Gamma_{j+1} a_{j+1}^*(j-i+1)\right]$$
   (b) Set $\Gamma_j = a_j(j)$
   (c) If $|\Gamma_j| = 1$, Quit.
3. $\epsilon_p = b^2(0)$

The Levinson-Durbin algorithm determine relation between autocorrelation *r(x)*,polynomial *a(k)* and the reflection coefficients. This can be summarized in the figure below and in the table 5.1 –5.4.



c

Table 5.4 ***The Inverse Levinson-Durbin Recursion***

1. Initialize the recursion
   (a) $r_x(0) = \epsilon_p / \prod_{i=1}^{p}(1-|\Gamma_i|^2)$
   (b) $a_0(0) = 1$
2. For $j = 0, 1, \dots, p-1$
   (a) For $i = 1, 2, \dots, j$
   $a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$
   (b) $a_{j+1}(j+1) = \Gamma_{j+1}$
   (c) $r_x(j+1) = -\sum_{i=1}^{j+1} a_{j+1}(i) r_x(j+1-i)$
3. Done

Table 5.2 ***The Step-up Recursion***

1. Initialize the recursion: $a_0(0) = 1$
2. For $j = 0, 1, \dots, p-1$
   (a) For $i = 1, 2, \dots, j$
   $a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$
   (b) $a_{j+1}(j+1) = \Gamma_{j+1}$
3. $b(0) = \sqrt{\epsilon_p}$

Lesson 4

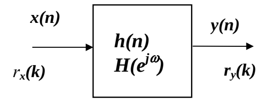
Chapter 6.
Lattice Filters

LTH

September 2008

Bengt Mandersson
Department of Electroscience
Lund University

Chapter 3 Review of filtering random processes



Input-output relation (convolution)

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k) h(n-k)$$

Autocorrelation function (deterministic)

$$r_x(k) = \sum_n x(n)\ x(n-k) = (r_{xx}(k))$$

Autocorrelation function (random processes)

$$r_x(k) = E\{x(n)\ x(n-k)\} = (r_{xx}(k))$$

Cross correlation function (random processes)

$$r_{yx}(k) = E\{y(n)\ x(n-k)\}$$

Autocorrelation function for the output

$$r_y(k) = E\{y(n)\,y(n-k)\} = \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h(l)\, r_x(m-l+k)\, h(m)$$

Cross correlation functions

$$r_{yx}(k) = E\{y(n)\,x(n-k)\} = \sum_{l=-\infty}^{\infty} h(l)\, r_x(k-l)$$

$$r_{xy}(k) = E\{x(n)\,y(n-k)\} = \sum_{l=-\infty}^{\infty} h(l)\ r_x(k+l)$$

Correlation functions

$$r_y(k) = r_x(k) * h(k) * h(-k)$$

$$r_{yx}(k) = r_x(k) * h(k)$$

$$r_{xy}(k) = r_x(k) * h(-k)$$

Spectra

$$P_y(e^{j\omega}) = P_x(e^{j\omega}) |H(e^{j\omega})|^2$$

$$P_{yx}(e^{j\omega}) = P_x(e^{j\omega}) H(e^{j\omega})$$

$$P_{xy}(e^{j\omega}) = P_x(e^{j\omega})\ H^*(e^{j\omega})$$

$$P_y(z) = P_x(z)\ H(z) H(z^{-1})$$

$$P_{yx}(z) = P_x(z)\ H(z)$$

$$P_{xy}(z) = P_x(z)\ H(z^{-1})$$

Chapter 3 Review of the All-pole model.

The difference equation for the input $\delta(n)$ **is (deterministic)**

$$x(n) + \sum_{k=1}^{p} a_p(k)\ x(n-k) = b(0)\delta(n)$$

and the system function

$$H(z) = \frac{b(0)}{1 + a_p(1) z^{-1} + a_p(2) z^{-2} + \cdots + a_p(p) z^{-p}} = \frac{b(0)}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}}$$

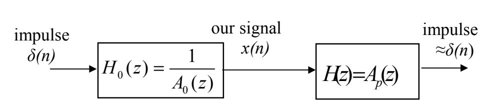
The output should be zero for all $n \neq 0$**. We define an error**

$$e(n) = x(n) + \sum_{k=1}^{p} a_p(k)\ x(n-k)$$

and we minimize

$$\varepsilon_p = \sum_{n=0}^{\infty} |\,e(n)\,|^2$$

This can be described by the following figure (*b(0)=1*).



$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k) z^{-k}$$

The filter $A_p(z)$ is called the predicting error filter (PEF).

We use a least squares solution to solve the problem.

Take the derivative (for simplicity, we assume real valued signals).

$$\frac{\partial \varepsilon_p}{\partial a_p(k)} = \frac{\partial}{\partial a_p(k)} \sum_{n=0}^{\infty} |e(n)|^2 = 2\sum_{n=0}^{\infty} e(n) \frac{\partial}{\partial a_p(k)} e(n) =$$

$$= 2\sum_{n=0}^{\infty} e(n) \frac{\partial}{\partial a_p(k)} [x(n) + \sum_{l=1}^{p} a_p(l)\, x(n-l)\,] =$$

$$= 2\underbrace{\sum_{n=0}^{\infty} e(n)x(n-k)}_{e(n)\text{ and given data orthogonal}} = 0 \quad k = 1,2,..,p$$

Then

$$\sum_{n=0}^{\infty} [x(n) + \sum_{l=1}^{p} a_p(l)\, x(n-l)\,] x(n-k) = 0$$

With

$$r_x(k) = \sum_{n=0}^{\infty} x(n)\, x(n-k)$$

we got the result

$$r_x(k) + \sum_{l=1}^{p} a_p(l) \underbrace{r_x(l-k)}_{r_x(k-l)} = 0$$

or rewritten

$$\sum_{l=1}^{p} a_p(l)\, r_x(k-l) = -r_x(k) \qquad k = 1,...,p$$

This equation is called the normal equation or the Yule-Walker equation.

In matrix form

$$\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \ldots & r_x(p-1) \\ r_x(1) & r_x(0) & r_x(1) & \ldots & r_x(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \ldots & r_x(p-3) \\ & & & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \ldots & r_x(0) \end{bmatrix} \cdot \begin{bmatrix} a_p(1) \\ a_p(2) \\ a_p(3) \\ \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} r_x(1) \\ r_x(2) \\ r_x(3) \\ \\ r_x(p) \end{bmatrix}$$

Orthogonality principle.

We can derive the filter in a slightly different way.
Writing

$$\varepsilon_p = \sum_{n=0}^{\infty} |e(n)|^2 = \sum_{n=0}^{\infty} e(n)e(n) = \sum_{n=0}^{\infty} e(n)[x(n) + \sum_{k=1}^{p} a_p(k)x(n-k)] =$$

$$= \underbrace{\sum_{n=0}^{\infty} e(n)x(n)}_{\varepsilon_{p,\min}\text{ called model error}} + \sum_{k=1}^{p} a_p(k) \underbrace{\sum_{n=0}^{\infty} e(n)x(n-k)}_{=0,\ e(n)\text{ and given data must be orthogonal}}$$

The minimum error (model error) is now found as

$$\varepsilon_{p,\min} = \varepsilon_p = \sum_{n=0}^{\infty} e(n)x(n) = \sum_{n=0}^{\infty} [x(n) + \sum_{k=1}^{p} a_p(k)x(n-k)]x(n) =$$

$$= r_x(0) + \sum_{k=1}^{p} a(k)\, r_x(k)$$

$$\varepsilon_{p,\min} = \varepsilon_p = r_x(0) + \sum_{k=1}^{p} a(k)\, r_x(k)$$

This equation can be added to the matrix equation described above.

Then, we got (for real signals $r_x^*(k) = r_x(k)$)

$$\underbrace{\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \ldots & r_x(p) \\ r_x(1) & r_x(0) & r_x(1) & \ldots & r_x(p-1) \\ r_x(2) & r_x(1) & r_x(0) & \ldots & r_x(p-2) \\ & & & & \\ r_x(p) & r_x(p-1) & r_x(p-2) & \ldots & r_x(0) \end{bmatrix}}_{R_x} \cdot \underbrace{\begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \\ a_p(p) \end{bmatrix}}_{a_p} = \cdot \underbrace{\begin{bmatrix} \varepsilon_p \\ 0 \\ 0 \\ \\ 0 \end{bmatrix}}_{\varepsilon_p u_1}$$

$$R_x\ a_p = \varepsilon_p u_1$$

This is a symmetrical Toeplitz matrix equation system and can be solve with the Levinson-Durbin algorithm described in chapter 5.

This all-pole model is often called Prediction Error Filter (PEF) or Linear Prediction Coding (LPC).

Chapter 6 Lattice Filters

In chapter 4, we derive the normal equations or Yule-walker equations for an all-pole model. And in chapter 5 we derived an algorithm (Levinson-Durbin) to solve the equations. In this chapter we will interpret the signals direct in a Lattice FIR structure.

The difference equation for the input $\delta(n)$ is

$$x(n) + \sum_{k=1}^{p} a_p(k)\, x(n-k) = \underbrace{b(0)\ \delta(n)}_{e(n)}$$

and the system function

$$H(z) = \frac{b(0)}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}} = \frac{b(0)}{A_p(z)}$$

The output should be zero for all $n \neq 0$. The error was defined as

$$e(n) = x(n) \ - \underbrace{(-\sum_{k=1}^{p} a_p(k)\ x(n-k))}_{\text{prediction or estimate of } x(n)}$$

We minimized the cost

$$\varepsilon_p = \sum_{n=0}^{\infty} |\, e(n)\,|^2$$

The solution was given by the normal equation (chapter 4, page 216-219)

$$r_x(k) + \sum_{l=1}^{p} a_p(l)\; r_k(l-k) = 0; \qquad k = 1, 2 \ldots p$$

and the error

$$\varepsilon_p = r_x(0) + \sum_{l=1}^{p} a(l)\; r_x(l)$$

In matrix form this can be written as

$$\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \ldots & r_x(p-1) \\ r_x(1) & r_x(0) & r_x(1) & \ldots & r_x(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \ldots & r_x(p-3) \\ & & & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \ldots & r_x(0) \end{bmatrix} \cdot \underbrace{\begin{bmatrix} a_p(1) \\ a_p(2) \\ a_p(3) \\ \\ a_p(p) \end{bmatrix}}_{\bar{a}_p} = - \begin{bmatrix} r_x(1) \\ r_x(2) \\ r_x(3) \\ \\ r_x(p) \end{bmatrix}$$

$$\varepsilon_{p,\min} = \varepsilon_p = r_x(0) + \sum_{k=1}^{p} a(k)\; r_x\;(k)$$

or as

$$\underbrace{\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \ldots & r_x(p) \\ r_x(1) & r_x(0) & r_x(1) & \ldots & r_x(p-1) \\ r_x(2) & r_x(1) & r_x(0) & \ldots & r_x(p-2) \\ & & & & \\ r_x(p) & r_x(p-1) & r_x(p-2) & \ldots & r_x(0) \end{bmatrix}}_{R_x} \cdot \underbrace{\begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \\ a_p(p) \end{bmatrix}}_{a_p} = \underbrace{\begin{bmatrix} \varepsilon_p \\ 0 \\ 0 \\ \\ 0 \end{bmatrix}}_{\varepsilon_p u_1}$$

$$R_x \;\; a_p = \varepsilon_p u_1$$

Levinson-Durbin (chapter 5) solves the normal equations iteratively.

The solution gives $a_j(k)$ **and** Γ_j **in each step j=1,..,p**

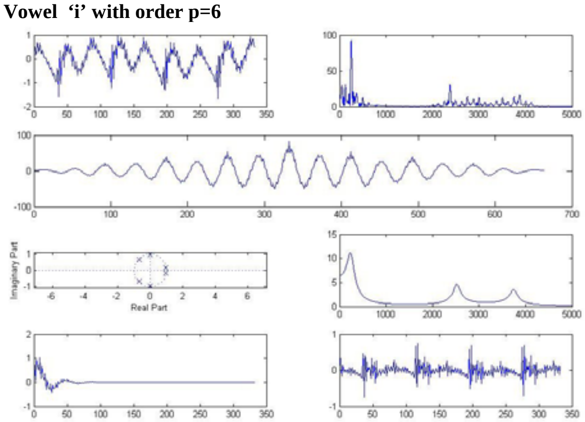
Table 5.1 ***The Levinson-Durbin Recursion***

1. Initialize the recursion
   - (a) $a_0(0) = 1$
   - (b) $\epsilon_0 = r_x(0)$
2. For $j = 0, 1, \ldots, p-1$
   - (a) $\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j-i+1)$
   - (b) $\Gamma_{j+1} = -\gamma_j / \epsilon_j$
   - (c) For $i = 1, 2, \ldots, j$

     $a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$
   - (d) $a_{j+1}(j+1) = \Gamma_{j+1}$
   - (e) $\epsilon_{j+1} = \epsilon_j \left[1 - |\Gamma_{j+1}|^2\right]$
3. $b(0) = \sqrt{\epsilon_p}$

Example of all-pole model of vowels
Example 1

Vowel 'i' with order p=6



Upper left: Signal , Upper right: Fourier transform (DFT) of the signal

Middle: Autocorrelation sequence of the signal
Pole-zero plot Spectrum from poles

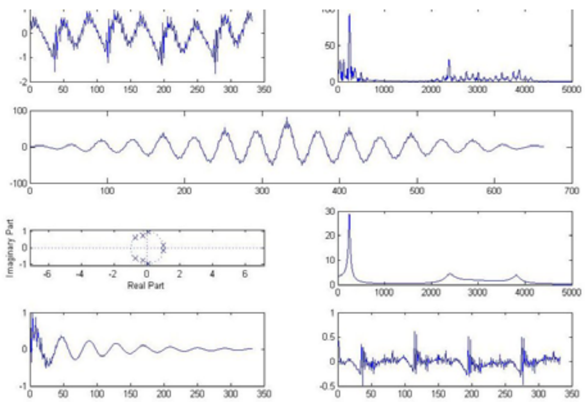
Lower left: Impulse response to $H_{IIR}(z)=1/A(z)$
Lower right : Output from $H_{FIR}(z)=A(z)$.

Coefficients $a_p(k)$ for order p=6.
a=[1.0000 -0.4912 0.1714 -1.0041 0.1397 -0.4127 0.7529]

Reflection coefficients:
Γ= [-0.7021 -0.2154 -0.5704 -0.0168 -0.0990 0.7529]

Example 2

Vowel 'i' with order p=8



Upper left: Signal , **Upper right:** Fourier transform (DFT) of the signal

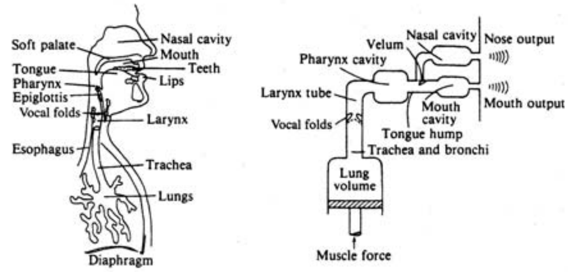
Middle: Autocorrelation sequence of the signal
Pole-zero plot Spectrum from poles

Lower left: Impulse response to $H_{IIR}(z)=1/A(z)$
Lower right : Output from $H_{FIR}(z)=A(z)$.

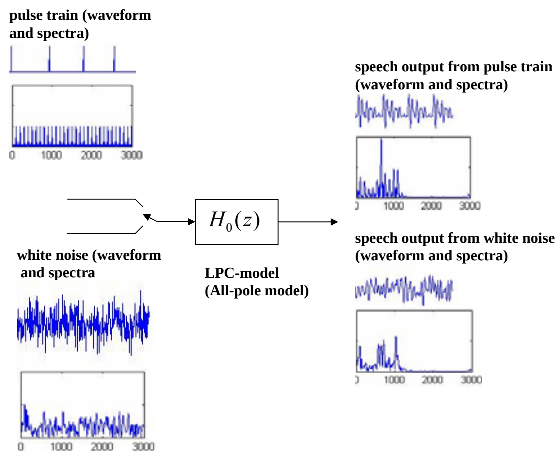
Coefficients $a_p(k)$ for order p=8.
a=[1.0000 -0.1191 0.3997 -1.1694 -0.2256 -0.9065 0.6446 0.1265 0.5622]

Reflection coefficients:
Γ= [-0.7021 -0.2154 -0.5704 -0.0168 -0.0990 0.7529 0.2829 0.5622]

LPC Speech encoding

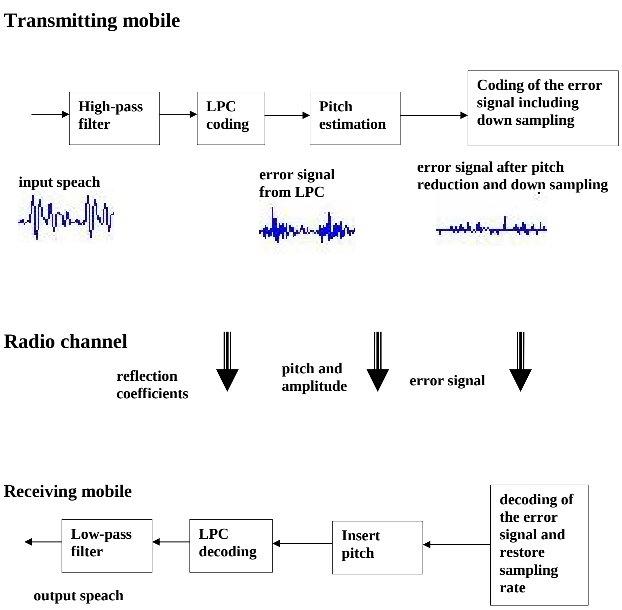


LPC model of syntetic sound production



In syntetic speech production, the parameters often are updated every 5 milliseconds.

The principles of the speech coding in GSM



In the laboratory work 1, we listen to the signals after each block and we also plot the waveforms and the spectra after each step.

FIR Lattice Filter structure

Now we look at signals direct in the FIR Lattice Filter structure. We determine the coefficients direct from the signal not determining the autocorrelation function r(k).

The error signal (output signal) is

$$e(n) = x(n) + \underbrace{\sum_{k=1}^{p} a_p(k)x(n-k)}_{-\hat{x}(n)} = x(n) - \hat{x}(n)$$

with
$$\hat{x}(n) = -\sum_{k=1}^{p} a_p(k)x(n-k)$$

We refer this error as the forward prediction error and use the notation

$$e^{+}(n) = x(n) + \underbrace{\sum_{k=1}^{p} a_p(k)x(n-k)}_{-\hat{x}(n)} = x(n) - \hat{x}(n)$$

This signal is found as the output from the upper branch in the Lattice FIR filter.
Now, we also define the signal in the lower branch as the backward prediction error.

Forward/Backward Prediction Error

From chapter 5, we have (page 224, 235, 236) (real signals)

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1}\ a_j^R(i-1)$$
$$a_{j+1}^R(i) = a_j^R(i-1) + \Gamma_{j+1}\ a_j(i)$$

and the transforms

$$A_{j+1}(z) = A_j(z) + z^{-1}\,\Gamma_{j+1}\ A_j^R(z)$$
$$A_{j+1}^R(z) = z^{-1} A_j^R(z) + \Gamma_{j+1}\ A_j(z)$$

The output in each step is

$$E_j^+(z) = A_j(z)\ X(z)$$
$$E_j^-(z) = A_j^R(z)\ X(z)$$

The error signal is the

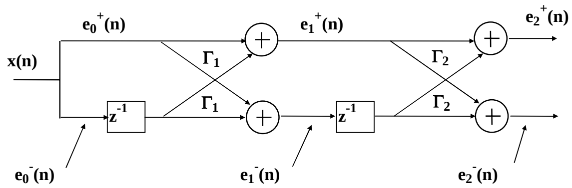
$$e_{j+1}^+(n) = e_j^+(n) + \Gamma_{j+1}\ e_j^-(n-1)$$

$$e_{j+1}^-(n) = e_j^-(n-1) + \Gamma_{j+1}\ e_j(n)$$

Second order Lattice-FIR-filter (real signals)

We can interpret the forward prediction error in the upper branch and the backward prediction error in the lower branch in the Lattice FIR filter shown below.

FIR *H(z)=A (z)*



We now will briefly present methods using the Lattice structure.

Forward Covariance Method, page 308

Backward Covariance Method, page 313

Burgs Method, page 317

Another method is the modified covariance method (page 322) using only FIR structure.

6.4 IIR Lattice filters. Some examples in the exercises

Forward Covariance method, page 308

Given: **The Lattice FIR structure (real signals).**

Task: **Determine the predictor, which minimize the forward prediction error**

$$\varepsilon_j^+ = \sum_{n=j}^{N} |\,e_j^+(n)\,|^2$$

where

$$e_j^+(n) = e_{j-1}^+(n) + \Gamma_j^+\, e_{j-1}^-(n-1)$$

Solution:

Take the derivative of ε_j^+ **with respect to** Γ_j^+

$$\frac{\delta \varepsilon_j^+}{\delta\ \Gamma_j^+} = 2\sum_{n=j}^{N} e_j^+(n)\ \underbrace{\frac{\delta e_j^+}{\delta\ \Gamma_j^+}}_{e_{j-1}^-(n-1)} =$$

$$= 2\sum_{n=j}^{N}\ \left[e_{j-1}^+(n) + \Gamma_j^+ e_{j-1}^-(n-1)\ \right] e_{j-1}^-(n-1) = 0$$

This gives the solution

$$\Gamma_j^+ = -\frac{\sum_{n=j}^{N}\ e_{j-1}^+(n)(e_{j-1}^-(n-1))}{\sum_{n=j}^{N}\ |\,(e_{j-1}^-(n-1))\,|^2}$$

Backward Covariance method (page 313-314)

Given: **The Lattice FIR structure (real signals).**

Task: **Determine the predictor, which minimize the forward prediction error**

$$\varepsilon_j^- = \sum_{n=j}^{N} |\,e_j^-(n)\,|^2$$

with

$$e_j^-(n) = e_{j-1}^-(n-1) + \Gamma_j^-\, e_{j-1}^+(n)$$

Solution:

Take the derivative of ε_j^- **with respect to** Γ_j^-

This gives the solution

$$\Gamma_j^- = -\frac{\sum_{n=j}^{N}\ e_{j-1}^+(n)(e_{j-1}^-(n-1))}{\sum_{n=j}^{N}\ |\,e_{j-1}^+(n)\,|^2}$$

Burgs method (page 317-319)

Given: The Lattice FIR structure (real signals).

Task: Determine the predictor, which minimize the forward prediction error

$$\varepsilon_j^B = \varepsilon_j^+ + \varepsilon_j^- = \sum_{n=j}^{N} (|e_j^+(n)|^2 + |e_j^-(n)|^2)$$

with

$$e_j^+(n) = e_{j-1}^+(n) + \Gamma_j^B e_{j-1}^-(n-1)$$

$$e_j^-(n) = e_{j-1}^-(n-1) + \Gamma_j^B \; e_{j-1}^+(n)$$

Solution:

Take the derivative of ε_j^B **with respect to** Γ_j^B

This gives the solution

$$\Gamma_j^B = -\frac{2\sum_{n=j}^{N} e_{j-1}^+(n)(e_{j-1}^-(n-1))}{\sum_{n=j}^{N} (|e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2)}$$

Burgs method step by step

Step 1:

$$\Gamma_1^B = -\frac{2\sum_{n=1}^{N} x(n)(x(n-1))}{\sum_{n=1}^{N} (|x(n)|^2 + |x(n-1)|^2)}$$

Step 2:

$$e_1^+(n) = x(n) + \Gamma_1^B x(n-1), \quad n = 1,\ldots,N$$

$$e_1^-(n) = x(n-1) + \Gamma_1^B x(n)$$

Step 3

$$\Gamma_2^B = -\frac{2\sum_{n=2}^{N} e_1^+(n)(e_1^-(n-1))}{\sum_{n=2}^{N} (|e_1^+(n)|^2 + |e_1^-(n-1)|^2)}$$

Step 4

$$e_2^+(n) = e_1^+(n) + \Gamma_2^B e_1^-(n-1), \quad n = 2,\ldots,N$$

$$e_2^-(n) = e_1^-(n-1) + \Gamma_2^B e_1^+(n)$$

and so on

Which method is the best?

Useful for short data sequences.

The forward and backward covariance methods can give reflection coefficients not always less than 1 and then, the signal model is not stable.

The reflection coefficients estimated using the Burg method are always less than 1 and signal model is stable.

Burgs method modified with a window

Given: The Lattice FIR structure (real signals).

Task: Determine the predictor, which minimize the forward prediction error

$$\varepsilon_j^B = \varepsilon_j^+ + \varepsilon_j^- = \sum_{n=j}^{N} w_j(n)(|e_j^+(n)|^2 + |e_j^-(n)|^2)$$

Solution:

This gives the solution (see exercise 6.18)

$$\Gamma_j^B = -\frac{2\sum_{n=j}^{N} w_{j-1}(n) e_{j-1}^+(n)(e_{j-1}^-(n-1))}{\sum_{n=j}^{N} w_{j-1}(n)(|e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1))|^2)}$$

Forward/Backward Covariance method, page 322.
Gives the coefficients $a_p(k)$ direct.

Given: **The Lattice FIR structure (real signals).**

Task: **Determine the predictor, which minimize the forward prediction error**

$$\varepsilon_p^M = \sum_{n=p}^{N} (|\, e_p^+(n)\,|^2 + |\, e_p^-(n)\,|^2)$$

with

$$e_p^+(n) = x(n) + \sum_{k=1}^{p} a_p(k) x(n-k)$$

$$e_p^-(n) = x(n-p) + \sum_{k=1}^{p} a_p(k) x(n-p+k)$$

Solution:

Take the derivative of ε_p^M **with respect to** $a_p^M(k)$

This gives the solution

$$\sum_{k=1}^{p} (r_x(l,k) + r_x(p-k, p-l))\ a_p(k) = -(\, r_x(l,0) + r_x(p, p-l))$$

Lesson 5

Chapter 7.
Wiener Filters

LTH

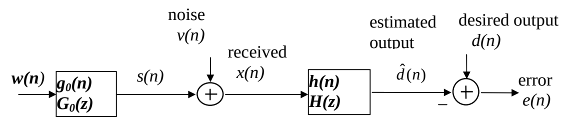
September 2008

Bengt Mandersson
Department of Electroscience
Lund University

Chapter 7 Wiener Filters

In this chapter we will use the model shown below.

The signal into the receiver is *x(n)* (received signal). Normally, this signal is disturbed by additive white noise *v(n)*. The information is in *s(n)*. Also, we often used the approach that the information signal is modeled as white noise w(n) filtered in a filter *g(n)*.



We will minimize the output error *e(n)*, which we describe as the difference between the desired output *d(n)* and the estimated output.

$$\text{minimize} \quad \xi = E[e^2(n)] = E[(d(n) - \hat{d}(n))^2]$$

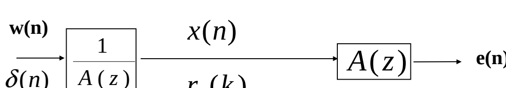
Applications.

Filtering s(n): The desired signal is *s(n)* and we will determine the optimum filter for noise reduction.
Smoothing: Like filtering but we allow an extra delay in the output signal (specially image processing).
Prediction: The output is a prediction of future values of *s(n)*. One step predictor. predict next value *s(n+1)*.
Equalization: The desired signal is *w(n)* and we will determine the optimum filter for whitening the output spectrum (inverse filtering, deconvolution).

Other applications: Echo cancellation. Noise cancellation. Pulse shaping.

Prediction Error Filter PEF (second order) from chapter 4

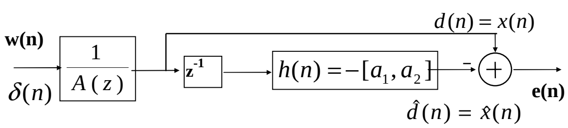
Model of the signal x(n).
Input: white noise $w(n)$ or impulse $\delta(n)$



$$A(z) = 1 + a_2(1)z^{-1} + a_2(2)z^{-2}$$

$$e(n) = x(n) + \sum_{l=1}^{2} a_2(l)x(n-l) =$$
$$= x(n) - \hat{x}(n) \qquad with$$
$$\hat{x}(n) = -\sum_{l=1}^{2} a_2(l)x(n-l)$$

We can rewrite the figure

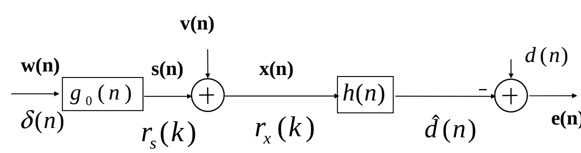


$d(n)$ *desired signal (önskad)*

$\hat{d}(n) = \hat{x}(n)$ *estimated signal*

$e(n) = x(n) - \hat{x}(n)$ *error signal*

Optimum Filters (process the received signal x(n))



We assume uncorrelated noise v(n).

$d(n)$ **could be:**	$s(n)$,	**filtering noisy signal** $x(n)$
	$s(n-n_0)$,	**smoothing (allow delay)**
	$s(n+n_0)$,	**predict future values**
	$\delta(n-n_0)$,	**inverse filtering, deconvolution**

h(n) causal FIR filter:	**easy, useful (chap. 7.2)**
h(n) noncausal IIR filter:	**easy, less useful (chap. 7.3.1)**
h(n) causal IIR filter	**more difficult, useful (chap. 7.3.2)**

We assume that correlation functions $r_x(k)$, $r_{dx}(k)$ and $r_d(k)$ are known or could be estimated.

Derivation of the optimal solution (Wiener filter).

Real-valued random signals.

We start with

$$\xi = E[e^2(n)] = E[(d(n)-\hat{d}(n))^2]$$

with

$$\hat{d}(n) = \sum_{l=-\infty}^{\infty} h(l)x(n-l) \qquad \textbf{(in general noncausal filter)}$$

and

$$e(n) = d(n)-\hat{d}(n) = d(n) - \sum_{l=-\infty}^{\infty} h(l)x(n-l)$$

Set the derivative of ξ with respect to h(k) equal to zero for all k.

$$\xi = \frac{\partial}{\partial h(k)} E[e^2(n)] = 2E[e(n)\frac{\partial}{\partial h(k)} e(n)] = 2E[e(n)(-x(n-k))] = 0$$

which gives

$$E[e(n)\,x(n-k)] = 0 \qquad \textbf{(The orthogonality principle)}$$

Replace *e(n)* and then

$$E[(d(n) - \sum_{l=-\infty}^{\infty} h(l)x(n-l))x(n-k)] = 0$$

and we got the Wiener-Hopf equations

$$\sum_{l=-\infty}^{\infty} h(l) r_x(k-l) = r_{dx}(k)$$

Derivation of the minimum error

Writing

$$\xi = E[e^2(n)] = E[e(n)\,e(n)] = E\{e(n)[d(n) - \sum_{l=-\infty}^{\infty} h(l)x(n-l)]\} =$$

$$= \underbrace{E\{e(n)\,d(n)\}}_{\xi_{\min}} + \sum_{l=-\infty}^{\infty} h(l) \underbrace{E\{e(n)\,x(n-l)\}}_{=0}$$

e(n) and given data orthogonal

This gives the minimum error

$$\xi_{\min} = E[e(n)d(n)] = E\{[d(n) - \sum_{l=-\infty}^{\infty} h(l)x(n-l)]d(n)\} =$$

$$= r_d(0) - \sum_{l=\infty}^{\infty} h(l) r_{xd}(-l) = r_d(0) - \sum_{l=\infty}^{\infty} h(l) r_{dx}(l)$$

and

$$\xi_{\min} = r_d(0) - \sum_{l=-\infty}^{\infty} h(l) r_{dx}(l)$$

The Wiener filter was derive from random signals.
For a deterministic approach we have to use the definition of autocorrelation and cross correlation

$$r_x(k) = \sum_{n=-\infty}^{\infty} x(n)x(n-k)$$

$$r_{dx}(k) = \sum_{n=-\infty}^{\infty} d(n)x(n-k)$$

Then, minimize

$$\varepsilon = \sum_{n=0}^{\infty} e^2(n) = \sum_{n=0}^{\infty} (d(n)-\hat{d}(n))^2$$

The Wiener-Hopf equations will be the same.

Now, we will look at the three types of filters *H(z)*

- **FIR Wiener filter (in the textbook denoted *W(z))***
- **Noncausal IIR filter**
- **Causal Wiener filter (at the end of this chapter)**

FIR Wiener filter (pp. 337-339, table 7.1 page 339)

The Wiener-Hopf equations are now

$$\sum_{l=0}^{p-1} h(l) r_x(k-l) = r_{dx}(k) \qquad k = 0,1,\dots,p-1$$

or in matrix form

$$\underbrace{\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r(p-1) \\ r_x(1) & r_x(0) & r_x(1) & \cdots & r_x(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x(p-3) \\ & & \cdots & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix}}_{R_X} \cdot \underbrace{\begin{bmatrix} h(0) \\ h(1) \\ h(2) \\ \cdot \\ h(p-1) \end{bmatrix}}_{h} = \underbrace{\begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ r_{dx}(2) \\ \cdot \\ r_{dx}(p-1) \end{bmatrix}}_{r_{dx}}$$

$$R_x \; h = r_{dx}$$

The solution is

$$h = R_x^{-1} \, r_{dx}$$

and the minimum error

$$\xi_{\min} = r_d(0) - \sum_{l=0}^{p-1} h(l) r_{dx}(l)$$

which also can be written

$$\xi_{\min} = r_d(0) - \sum_{l=0}^{p-1} h(l)\; r_{dx}(l) = r_d(0) - r_{dx}{}^T \; h_{opt} = r_d(0) - r_{dx}{}^T \; R_x^{-1} \, r_{dx}$$

Noncausal IIR Wiener filter (pp. 353-356, table 7.2)

The Wiener-Hopf equation are here

$$\sum_{l=-\infty}^{\infty} h(l) r_x(k-l) = r_{dx}(k) \quad all \;\; k$$

Here we have a complete convolution and it can be solved using z-transform or Fourier transform

$$H(z)P_x(z) = P_{dx}(z)$$
$$H(z) = \frac{P_{dx}(z)}{P_x(z)};$$
$$H(e^{j\omega}) = \frac{P_{dx}(e^{j\omega})}{P_x(e^{j\omega})}$$

The minimum error is

$$\xi_{\min} = r_d(0) - \sum_{l=-\infty}^{\infty} h(l) \; r_{dx}(l)$$

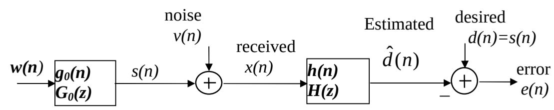
We can use the Parseval's relation and also write this in the frequency domain. Then, (see properties of the Fourier transform, see page 356, Table 7.2)

$$\xi_{\min} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left[P_d(e^{j\omega}) - H(e^{j\omega}) P_{dx}^{\;*}(e^{j\omega}) \right] d\omega$$

Filtering received signal for noise reduction

The received signal is disturbed by additive zero mean white noise

$$x(n) = s(n) + v(n)$$



Desired signal is now *s(n)*. Then

$$r_x(k) = r_s(k) + r_v(k)$$
$$r_{dx}(k) = E[d(n)x(n-k)] = E[s(n)(s(n-k)+v(n-k))] = r_s(k)$$
$$P_x(z) = P_s(z) + P_v(z)$$
$$P_{dx}(z) = P_s(z)$$

Causal FIR-filter for noise reduction

The FIR-filter equations are

$$\sum_{l=0}^{p-1} h(l) r_x(k-l) = r_{dx}(k) \quad k = 0,1,\ldots,p-1$$

Now, they will be

$$\sum_{l=0}^{p-1} h(l)(r_s(k-l) + r_v(k-l)) = r_s(k)$$

or

$$(R_s + R_v)\; h = r_s$$

and

$$h_{opt} = (R_s + R_v)^{-1} \;\; r_s$$

The spectrum we find from the Fourier Transform

$$H(e^{j\omega}) = Fourier\{h_{opt}(n)\}$$

Noncausal IIR-filter for noise reduction

For non-causal IIR filter, we have

$$H(z)P_x(z) = P_{dx}(z)$$
$$H(z) = \frac{P_{dx}(z)}{P_x(z)};$$
$$H(e^{j\omega}) = \frac{P_{dx}(e^{j\omega})}{P_x(e^{j\omega})}$$

In the filtering problem the power spectra are

$$P_x(z) = P_s(z) + P_v(z)$$
$$P_{dx}(z) = P_s(z)$$

which gives the Wiener filter

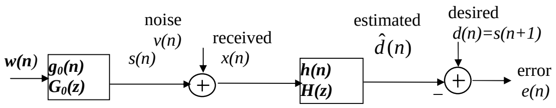
$$H(z) = \frac{P_s(z)}{P_s(z) + P_v(z)};$$
$$H(e^{j\omega}) = \frac{P_s(e^{j\omega})}{P_s(e^{j\omega}) + P_v(e^{j\omega})};$$

We see that for frequencies with low noise,

$$|\,H(e^{j\omega})\,| \approx 1$$

Prediction

In a one-step predictor, the desired signal is *s(n+1)*.



Desired signal is now *s(n+1)*. Then

$$r_{dx}(k) = E[d(n)x(n-k)] = E[s(n+1)(s(n-k)] = r_s(k+1)$$
$$P_{dx}(z) = z\,P_s(z)$$

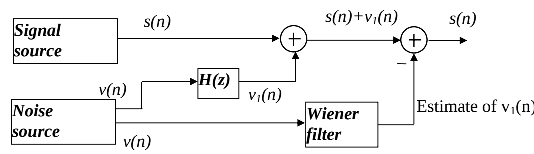
This gives the Wiener-Hopf equation

$$\sum_{l=0}^{p-1} h(l) r_s(k-l) = r_s(k+1) \quad k = 0,1,\ldots, p-1$$

Noise cancellation (page 349)

A signal is disturbed by additive noise $v_1(n)$.

Try to measure the noise v(n) from the source and estimate the noise $v_1(n)$ added to the signal. Then subtract the noise $v_1(n)$ from the received signal.

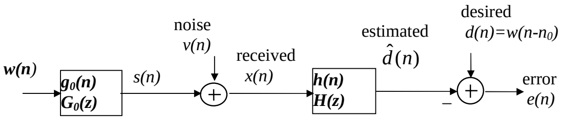


Deconvolution (equalizing, inverse filtering)

Desired signal here is *w(n)* (or allow delay, *w(n-n₀)*).
(see also problem 4.19)

This means that

$$g(n) * h(n) \approx \delta(n - n_0)$$

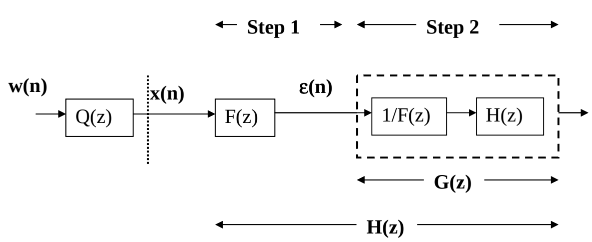


Causal IIR Wiener filter –1 (page 358-362)

Derivation of the causal filter is more difficult.
The Wiener solution is

$$\sum_{l=0}^{\infty} h(l) r_x(k-l) = r_{dx}(k)$$

We divide the solution into two steps.



In step 1, we whitening the input signal *x(n)*. From chapter 3, we have spectral factorization

$$P_x(z) = \sigma_0^2 \; Q(z)\; Q(z^{-1})$$

If the chose $F(z) = \dfrac{1}{\sigma\, Q(z)}$ **the signal *ε(n)* will be white with variance equal to 1.**

IIR, causal filter – 2

Step 2

In step 2 we know have (Wiener-Hopf equation)

$$\sum_{l=0}^{\infty} g(l) r_{\varepsilon}(k-l) = r_{d\varepsilon}(k)$$

with $$r_{\varepsilon}(k) = \delta(k)$$

The optimal filter (the causal filter, k≥0) is then

$$g(k) = r_{d\varepsilon}(k)\ u(k)$$

with the z-transform

$$G(z) = \left[P_{d\varepsilon}(z)\right]_{+}$$

The notation […]$_{+}$ means the causal part of the argument.

IIR, causal filter – 3

Vi have to determine $P_{d\varepsilon}(z)$. **Then**

$$r_{d\varepsilon}(k) = E\{d(n)\varepsilon(n-k)\} =$$
$$= E\{d(n)(\sum_{l} f(l)\, x(n-k-l))\} =$$
$$= \sum_{l} f(l)\, r_{dx}(k+l)$$

where

$$f(n) = Z^{-1}\{F(z)\}$$

Then

$$P_{d\varepsilon}(z) = P_{dx}(z) F(z^{-1}) = \frac{P_{dx}(z)}{\sigma_0\, Q(z^{-1})}$$

To find G(z) we take the causal part

$$G(z) = \left[\frac{P_{dx}(z)}{\sigma_0\, Q(z^{-1})}\right]_{+}$$

Combining step 1 and step 2 gives finally

$$H(z) = F(z) G(z) = \frac{1}{\sigma_0^2\, Q(z)} \left[\frac{P_{dx}(z)}{Q(z^{-1})}\right]_{+}$$

Relation between causal and non causal IIR Wiener filter

Non causal IIR Wiener filter

$$H(z) = \frac{P_{dx}(z)}{P_x(z)} = \frac{1}{\sigma_0^2\, Q(z)}\ \frac{P_{dx}(z)}{Q(z^{-1})}$$

Causal IIR Wiener filter

$$H(z) = \frac{1}{\sigma_0^2\, Q(z)} \left[\frac{P_{dx}(z)}{Q(z^{-1})}\right]_{+}$$

We can see both filters as a cascade two filters there the first is a whitening filter .

The minimum error is as before

$$\xi_{\min} = r_d(0) - \sum_{l=-0}^{\infty} h(l)\ r_{dx}(l)$$

Adaptive filtering.
Chapter 9 or the course ´Adaptive Signal Processing´.

We want to minimize the error

$$\xi = E[e^2(n)] = E[(d(n) - \hat{d}(n))^2]$$

Iterative solution

We can solve this iteratively using the update equation

$$h_{n+1}(k) = h_n(k) - \mu' \frac{\delta \xi}{\delta h_n(k)} =$$
$$= h_n(k) + 2\mu' E\{e(n)\, x(n-k)\}$$

there μ' **is the step size.**

Adaptive solution (Least Mean Square, LMS)

Use the approximation

$$E\{e(n)\, x(n-k)\} \approx e(n)\, x(n-k)$$

which gives

$$h_{n+1}(k) = h_n(k) + 2\mu' e(n)\, x(n-k)$$

How to chose step size μ' **?**
Does the algorithm converge? How fast?

Lesson 6

Chapter 8.
Spectrum estimation

LTH

October 2008

Bengt Mandersson
Department of Electroscience
Lund University

Spectrum estimation, Chapter 8

Nonparametric methods:

The periodogram
The modified Periodogram (windowing)
Averaging periodogram
- **Bartlett**
- **Welch**

The Blackman-Tukey method

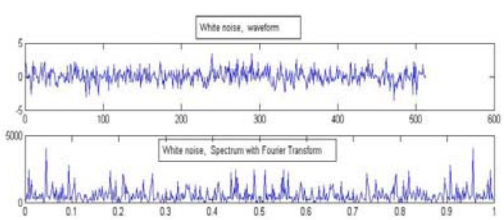
Parametric methods:

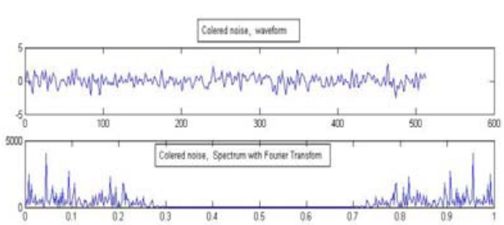
Described in chapter 4

Frequency estimation (Estimation of sinusoids), lesson 7

The well known methods like Pisarenco Harmonic Decomposition and the MUSIC algorithm are presented here. These methods are based on
the eigenvectors of the correlation matrix.

Examples of waveforms and Fourier Transforms





Row 1: White noise (N=512 values)
Row 2: Fourier transform of the signal in row 1 (magnitude) (N=512 values)

Row 3: Coloured noise (output from 4th order Butterworth filter)
Row 4: Fourier transform of the signal in row 3 (magnitude) (N=512 values)

Estimation of power spectra – periodogram
(page 393-394)

We want to estimate $r_x(k)$ **from** ***x(n)*** **in the interval** $0 \le n \le N-1$.
In chapter 3 we had

$$\hat{r}_x(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\,x(n-k)$$

To ensure that the values that fall outside the interval are excluded, we write

$$\hat{r}_x(k) = \frac{1}{N}\sum_{n=0}^{N-1-k} x(n+k)\,x(n) \qquad 0 \le k \le N-1$$

Using a rectangular window

$$w_R(n) = [\underbrace{1\ 1\ 1\cdots 1}_{N}] \quad \text{rectangular window}$$

this can be written

$$x_N(n) = x(n)\cdot w_R(n)$$

or

$$x_N(n) = \begin{cases} x(n) & 0 \le n \le N-1 \\ 0 & otherwise \end{cases}$$

The estimated autocorrelation can now be written

$$\hat{r}_x(k) = \frac{1}{N}\sum_{n=-\infty}^{\infty} x_N(n+k)x_N(n)$$
$$= \frac{1}{N}\sum_{n=-\infty}^{\infty} x(n+k)w_R(n+k)x(n)w_R(n)$$
$$= \frac{1}{N} x_N(k) * x_N(-k)$$

Then $\hat{r}_x(k)$ **is defined for** $-N+1 \le k \le N-1$

Now, we take the Fourier Transform of $\hat{r}_x(k)$**, and then we get**

$$\hat{P}_{per}(e^{j\omega}) = \sum_{k=-N+1}^{N-1} \hat{r}_x(k)\, e^{-j\omega k}$$

which is called the periodogram.

We see that it also can be written

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{N} X_N(e^{j\omega})\, X_N{}^*(e^{j\omega}) = \frac{1}{N} |\, X_N(e^{j\omega})\,|^2$$

Using DFT (FFT), the periodogram will be

$$\hat{P}_{per}(e^{j2\pi k/N}) = \frac{1}{N} X_N(k)\, X_N{}^*(k) = \frac{1}{N} |\, X_N(k)\,|^2$$

The Performance of the Periodogram (page 398-399)

The estimate is unbiased if

$$E\{\hat{P}_x(e^{j\omega})\} = P_x(e^{j\omega})$$

The estimate is consistent if it is (asymptotically) unbiased and if

$$\lim_{N\to\infty} \operatorname{var}\{\hat{P}_x(e^{j\omega})\} = 0$$

Taking the mean of $\hat{r}_x(k)$**, we got** ($k \ge 0$) **(page 398-399)**

$$E\{\hat{r}(k)\} = \frac{1}{N}\sum_{n=-\infty}^{\infty} E\{x_N(n+k)x_N(n)\} =$$
$$= \frac{1}{N}\sum_{n=0}^{N-1-k} E\{x(n+k)x(n)\} = \frac{1}{N}\sum_{n=0}^{N-1-k} r_x(k) = \frac{N-k}{N}\, r_x(k)$$

Defining the Bartlett (triangular) window

$$w_B(k) = \begin{cases} \dfrac{N-|k|}{N} & |k| \le N \\ 0 & |k| > N \end{cases}$$

we can write

$$E\{\hat{r}_x(k)\} = w_B(k)\, r_x(k)$$

Using this, we have (page 399)

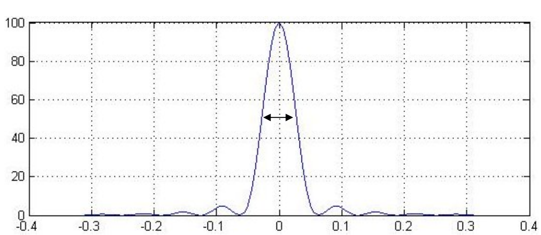
$$E\{\hat{P}_{per}(e^{j\omega})\} = E\{\sum_{k=-N+1}^{N-1} \hat{r}_x(k)\, e^{-j\omega k}\} =$$
$$= \sum_{k=-N+1}^{N-1} E\{\hat{r}_x(k)\} e^{-j\omega k} = \sum_{k=-\infty}^{\infty} r_x(k)\, w_B(k)\, e^{-j\omega k}$$

or

$$E\{\hat{P}_{per}(e^{j\omega})\} = \frac{1}{2\pi} P_x(e^{j\omega}) * W_B(e^{j\omega})$$

The Bartlett (triangular) window can be seen as the convolution of two rectangular windows. The window is

$$W_B(e^{j\omega}) = \frac{1}{N}\left[\frac{\sin(N\omega/2)}{\sin(\omega/2)}\right]^2$$



Plot of $W_B(e^{j\omega})$**, N=100, bandwidth 0.89*2π/N**

The estimate in asymptotically unbiased due to

$$\lim_{N\to\infty} E\{\hat{P}_{per}(e^{j\omega})\} = P_x(e^{j\omega})$$

The variance is (textbook page 404, 405)

$$\operatorname{var}\{\hat{P}_{per}(e^{j\omega})\} \approx P_x{}^2(e^{j\omega})$$

so the periodogram is not a consistent estimate of the power spectrum.

Table 8.1 ***Properties of the Periodogram***

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-jn\omega}\right|^2$$

Bias

$$E\left\{\hat{P}_{per}(e^{j\omega})\right\} = \frac{1}{2\pi} P_x(e^{j\omega}) * W_B(e^{j\omega})$$

Resolution

$$\Delta\omega = 0.89\frac{2\pi}{N}$$

Variance

$$\operatorname{Var}\left\{\hat{P}_{per}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})$$

The Modified Periodogram (windowing *x(n)*)

The periodogram use a rectangular window $w_R(n)$

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{N} | X_N(e^{j\omega}) |^2 = \frac{1}{N} | \sum_{n=-\infty}^{\infty} x(n)\, w_R(n)\; e^{-j\omega n}) |^2$$

If we use other windows, we got the modified periodogram

$$\hat{P}_M(e^{j\omega}) = \frac{1}{NU} | \sum_{n=-\infty}^{\infty} x(n)\, w(n)\; e^{-j\omega n}) |^2$$

$$U = \frac{1}{N} \sum_{n=0}^{N-1} | w(n)\; e^{-j\omega n}) |^2$$

Table 8.2 Properties of a Few Commonly Used Windows. Each Window is Assumed to be of Length N.

Window	Sidelobe Level (dB)	3 dB BW $(\Delta\omega)_{3dB}$
Rectangular	−13	$0.89(2\pi/N)$
Bartlett	−27	$1.28(2\pi/N)$
Hanning	−32	$1.44(2\pi/N)$
Hamming	−43	$1.30(2\pi/N)$
Blackman	−58	$1.68(2\pi/N)$

Properties of the modified periodogram

Table 8.3 Properties of the Modified Periodogram

$$\hat{P}_M(e^{j\omega}) = \frac{1}{NU} \left| \sum_{n=-\infty}^{\infty} w(n)x(n)e^{-jn\omega} \right|^2$$

$$U = \frac{1}{N} \sum_{n=0}^{N-1} |w(n)|^2$$

Bias

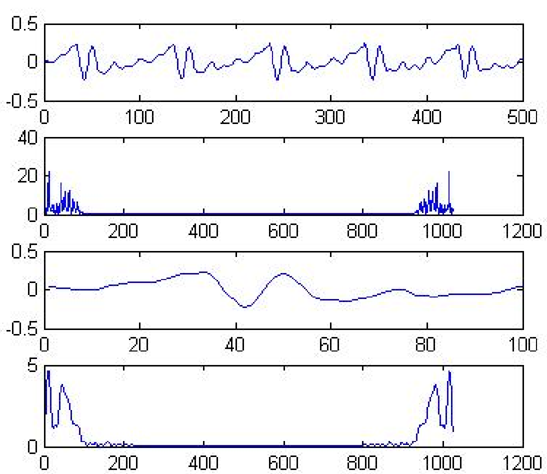
$$E\left\{\hat{P}_M(e^{j\omega})\right\} = \frac{1}{2\pi NU} P_x(e^{j\omega}) * |W(e^{j\omega})|^2$$

Resolution Window dependent

Variance

$$\mathrm{Var}\left\{\hat{P}_M(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})$$

Example of the resolution



Row 1: Waveform of a vowel 'a', N=500 (50 ms)-
Row 2: Fourier transform of the N=500 values in row 1.
Row 3: Part of the waveform in row 1, N=100, (10 ms)
Row 4: Fourier transform of the N=100 values in row 3.

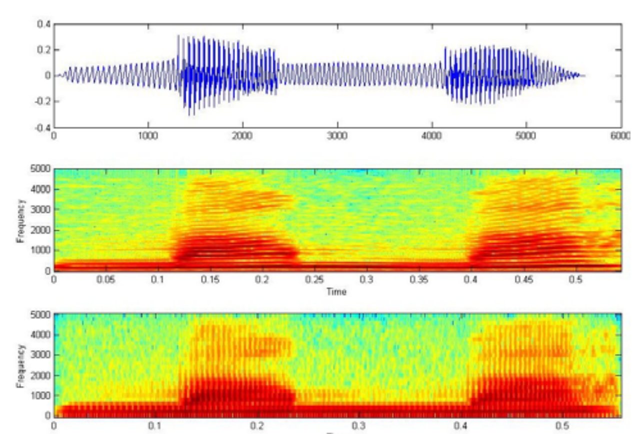
Spectrogram

Spectrogram is a plot of spectrum as function of the time using a sliding window. The command in Matlab is

specgram(x,Nfft,Fs);

Nfft is the length of the time window (length of the fft).
Fs is the sample frequency.

Example of spectrogram of the word 'mamma'.



Top: Waveform of the word 'mamma'.
Middle: Specgram with wide time window, N=200 (20 ms)
Bottom: Specgram with narrow time window, N=50 (5 ms)

Averaging periodogram. Bartlett's Method

(page 412.414)

In order to reduce the variance we must use averaging.

We divide the input sequence *x(n)* of length N into K blocks of length L,

$$K = \frac{N}{L}$$

Then, determine the power spectra for each block and take the average. The variance will decrease but also the resolution will decrease.

The variance will be

$$\text{var}\{\hat{P}_B(e^{j\omega})\} \approx \frac{1}{K} P_x^2(e^{j\omega})$$

Table 8.4 *Properties of Bartlett's Method*

$$\hat{P}_B(e^{j\omega}) = \frac{1}{N} \sum_{i=0}^{K-1} \left| \sum_{n=0}^{L-1} x(n+iL) e^{-jn\omega} \right|^2$$

Bias

$$E\left\{\hat{P}_B(e^{j\omega})\right\} = \frac{1}{2\pi} P_x(e^{j\omega}) * W_B(e^{j\omega})$$

Resolution

$$\Delta\omega = 0.89 K \frac{2\pi}{N}$$

Variance

$$\text{Var}\left\{\hat{P}_B(e^{j\omega})\right\} \approx \frac{1}{K} P_x^2(e^{j\omega})$$

Averaging periodogram. Welch's Method

(page 419)

The method of Welch is similar to the Bartlett's method but we allow overlapping of the blocks and using windows *w(n)*.

The estimated properties of Welch's method is found in table 8.5

Table 8.5 *Properties of Welch's Method*

$$\hat{P}_W(e^{j\omega}) = \frac{1}{KLU} \sum_{i=0}^{K-1} \left| \sum_{n=0}^{L-1} w(n) x(n+iD) e^{-jn\omega} \right|^2$$

$$U = \frac{1}{L} \sum_{n=0}^{L-1} |w(n)|^2$$

Bias

$$E\left\{\hat{P}_W(e^{j\omega})\right\} = \frac{1}{2\pi LU} P_x(e^{j\omega}) * |W(e^{j\omega})|^2$$

Resolution Window dependent

Variance†

$$\text{Var}\left\{\hat{P}_W(e^{j\omega})\right\} \approx \frac{9}{16} \frac{L}{N} P_x^2(e^{j\omega})$$

Blackman-Tukey Method

(page 420-423)

With the Blackman-Tukey method we calculate $\hat{r}_x(k)$ from all N data. But for large *k*, the estimate is not so god.

Multiply $\hat{r}_x(k)$ with a window symmetric around *k=0* and take the Fourier Transform. This gives

$$\hat{P}_{BT}(e^{j\omega}) = \sum_{k=-N+1}^{N-1} \hat{r}_x(k) w(k) e^{-j\omega k}$$

The spectrum of the window must be positive for all frequencies, i.e. $W(e^{j\omega}) \geq 0$, to guarantee that $P_{BT}(e^{j\omega}) \geq 0$. This is not true for a rectangular time window.

Table 8.6 *Properties of the Blackman-Tukey Method*

$$\hat{P}_{BT}(e^{j\omega}) = \sum_{k=-M}^{M} \hat{r}_x(k) w(k) e^{-jk\omega}$$

Bias

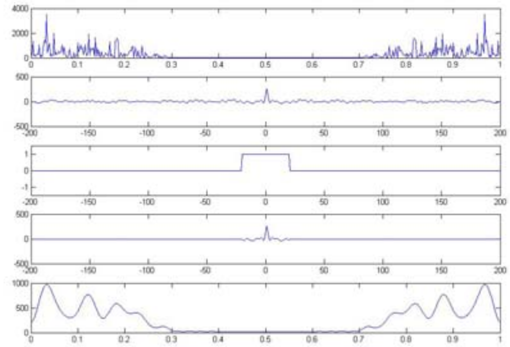
$$E\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx \frac{1}{2\pi} P_x(e^{j\omega}) * W(e^{j\omega})$$

Resolution Window dependent

Variance

$$\mathrm{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega}) \frac{1}{N} \sum_{k=-M}^{M} w^2(k)$$

Example of the Blackman-Tukey Method



Row 1: Spectrum from FFT.
Row 2: Autocorrelation
Row 3: Time window
Row 4: Windowed autocorrelation
Row 5: Blackman-Tukey Spectrum (from windowed autocorr)

Conclusion

We have always a trade-off between resolution and variance.

Time windows

Rectangular window has the best resolution but also highest leakage (highest side lobes)

Averaging

Averaging decreases the variance but for fix length of data the resolution also will decrease.

Performance comparisons

Definitions see page 424-426

Resolution: $\Delta\omega$

Variability: $\nu = \dfrac{\mathrm{var}\{\hat{P}_x(e^{j\omega})\}}{(E\{\hat{P}_x(e^{j\omega})\})^2}$

Figure of merit: $M = \nu \cdot \Delta\omega$

Quality factor: $Q = \dfrac{1}{\nu}$

Filter bank implementation of periodogram

We can interpret the periodogram as the output from of bank of band pass filters.

$$P_x(e^{j\omega}) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-j\omega n} \right|^2$$

For the frequency ω_i, this can be written

$$P_x(e^{j\omega_i}) = | y(n) |^2_{n=0} = N \left| \sum_{n=0}^{N-1} x(n) h_i(n-k) \right|^2_{n=0}$$

i.e. the squared of the output from the filter at n=0;

The band pass filters are then

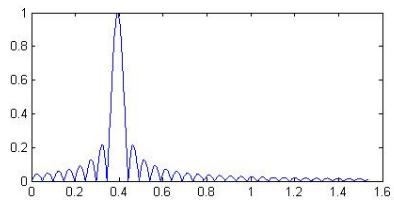
$$h_i(k) = \begin{cases} \dfrac{1}{N} e^{j\omega_i k} & k = -(N-1),\ldots,0 \\ 0 & \text{otherwise} \end{cases}$$

The Fourier transform of the filters

$$h_i(k) = \begin{cases} \frac{1}{N} e^{j\omega_i k} & k = -(N-1),\ldots,0 \\ 0 & otherwise \end{cases}$$

are

$$H_i(e^{j\omega_i}) = \frac{\sin(N(\omega-\omega_i)/2)}{N\sin((\omega-\omega_i)/2)} e^{-j(\omega-\omega_i)(N-1)/2}$$



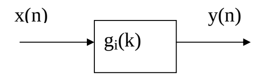
Conclusion: The value of the spectrum at this frequency is the output at n=0 from the band pass filter. The bandwidth is approximately

$$\Delta\omega = 2\pi / N$$
$$\Delta f = 1/N$$

Minimum variance spectral estimation

(page 426-429)

We use the idea of band pass filters



The output y(n) is an narrowband signal out from the band pass filter.

$$g_i(k), \quad G_i(e^{j\omega_i})$$

1. **Design a bank of band pass filters $g_i(k)$ with center frequency ω_i so that each filter rejects the maximum out-of-band power while passing component at ω_i with no distortions.**

2. **Filter x(n) with each filter and estimate the output power.**

3. **Set $\hat{P}_x(e^{j\omega_i})$ equal to the estimated power in step 2 divided by the filter bandwidth.**

The band pass filter depends on the properties of the signal x(n).

Use the vector notation:

Band pas sfilter: $g_i = [g_i(0), g_i(1), \ldots, g_i(p)]^T$

Sinusoids: $e_i = [1,\ e^{j\omega_i},\ e^{j\omega_i 2},\ \ldots,\ e^{j\omega_i p}\ ]^T$

Output: $y_i(n) = \sum_{k=0}^{p} g_i(k)x(n-k) = g_i^T x$

With the definition of e_i the Fourier transform of g at frequency ω_i can be written

$$G(e^{j\omega_i}) = \sum_{k=0}^{p} g(k)e^{-j\omega_i k} = e_i^H g = (g^H e_i)^H$$

Now, the spectrum estimate can be written (complex signals)

$$P(e^{j\omega_i}) = E\{y_i(n)y_i^*(n)\} = E\{g_i^H x x^H g_i\} =$$
$$= g_i^H R_x g_i$$

We must also normalize the band pass filters so that

$$G(e^{j\omega_i}) = \sum_{k=0}^{p} g_i(k)e^{-j\omega_i k} = e_i^H g_i = (g_i^H e_i)^H = 1$$

Then, we now want to minimize

$$P(e^{j\omega_i}) = g_i^H R_x g_i$$

due to the linear constraints

$$G_i(e^{j\omega_i}) = g_i^H e_i = 1$$

This can be done using Lagrange multipliers (page 50-52)

Introduce the Lagrange multiplier μ **and minimize (page 50-52)**

$$L(g_i,\mu)=\underbrace{\frac{1}{2}g_i^{\,H}R_x g_i}_{\text{minimize this}} + \mu\underbrace{(1-g_i^{\,H}e_i)}_{\text{this should be zero}}$$

Differentiate $L(g_i,\mu)$ **with respect to** $g_i^{\,H}$**. Then**

$$\nabla_{g_i^*}L(g_i,\mu)=R_x g_i - \mu e_i = 0$$

and

$$g_i=\mu R_x^{-1}e_i$$

Differentiate $L(g_i,\mu)$ **with respect to** μ gives

$$\frac{\delta}{\delta\mu}L(g_i,\mu)=1-g_i^{\,H}e_i=0$$

Then using

$$g_i=\mu R_x^{-1}e_i$$

we have

$$\mu=\frac{1}{e_i^H R_x^{-1}e_i}$$

This gives the filter

$$g_i=\frac{R_x^{-1}e_i}{e_i^H R_x^{-1}e_i}$$

The power at frequency ω_i **is estimated as**

$$P(e^{j\omega_i})=g_i^{\,H}R_x g_i=\frac{1}{e_i^H R_x^{-1}e_i}$$

We normalized the band pass filter but we must also normalize for the bandwidth of the band pass filter (length p+1).

A correction factor (p+1) (see page 429) finally gives the minimum variance estimate for any ω

$$P_{MV}(e^{j\omega})=\frac{p+1}{e^H R_x^{-1}e}$$

Lesson 7

Chapter 8.
Frequency estimation

LTH

October 2008

Bengt Mandersson
Department of Electroscience
Lund University

Chapter 8, Spectrum estimation

Nonparametric methods: lesson 6

The periodogram
The modified Periodogram (windowing)
Averaging periodogram
- **Bartlett**
- **Welch**

The Minimum variance method
The Blackman-Tukey method

Parametric methods:

Described in chapter 4
- **Pade**
- **Prony**
- **All-pole model**

Lattice structures in chapter 6

Frequency estimation (Estimation of sinusoids), lesson 7

The well known methods like Pisarenco Harmonic Decomposition and the MUSIC algorithm are presented here. These methods are based on the eigenvectors of the correlation matrix.

Pisarenco Harmonic Decomposition
The MUSIC algorithm
The Eigenvector method (EV)
(Minimum norm)
Principal components Blackman-Tukey frequency estimation
Minimum variance Frequency estimation

Frequency estimation

The model is that we have sinusoids in white noise.

$$x(n) = \sum_{i=1}^{p} A_i \, e^{j\omega_i n} + w(n)$$

with the complex amplitude

$$A_i = |A_i| e^{j\phi_i}$$

The phase is randomly distributed in the interval $-\pi \le \phi_i \le \pi$

We want to estimate

I: **The amplitudes** $|A_i|$; $\quad A_i = |A_i| e^{j\phi_i}$

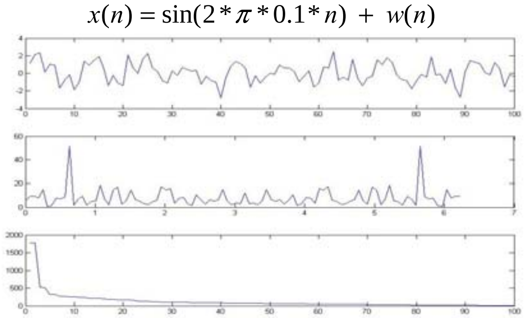
II: **The frequency** ω_i, f_i

III: **Number of sinusoids p**

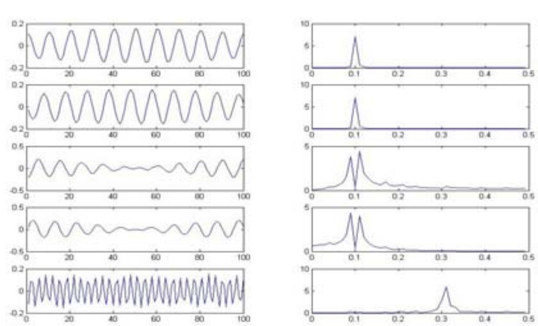
Frequency estimation,

Examples on eigenvectors and eigenvalues of the correlation matrix.
Sinusoid in white noise

$$x(n) = \sin(2*\pi*0.1*n) + w(n)$$



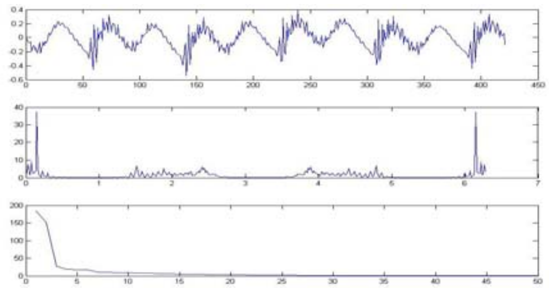
Upper: Waveform of a sinusoid in white noise
Middle: Spectrum from DFT
Lower: The eigenvalues of the correlation matrix R_x.



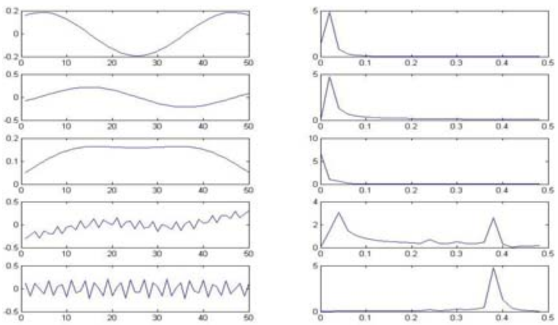
The first 5 eigenvectors (left) and their spectra (right)

Frequency estimation,

Examples on eigenvectors and eigenvalues of the correlation matrix.
Vowel ‘i’.



Upper: Waveform of a vowel ‘i’.
Middle: Spectrum from DFT
Lower: The eigenvalues of the correlation matrix R_x.



The first 5 eigenvectors (left) and their spectra (right)

Frequency estimation, correlation matrix.

We assume first that p=1,

$$x(n) = A_1 e^{j\omega_1 n} + w(n) \quad n = 0,\dots,N-1$$

or

$$x = A_1 e_1 + w$$

with

$$x = \left[x(0),\ x(1),\ \dots\ ,x(N-1)\right]^T$$
$$e_1 = \left[1,\ e^{j\omega_1},\ e^{j\omega_1 2},\ \dots\ ,e^{j\omega_1 (N-1)}\ \right]^T$$
$$w = \left[w(0),\ w(1),\ \dots\ ,w(N-1)\right]^T$$

The correlation matrix is

$$R_x = E\{x\,x^H\} =$$
$$= E\{(A_1 e_1 + w)(A_1 e_1 + w)^H\} =$$
$$= P_1 e_1 e_1^H + \sigma_w^2\ I$$

The power of the sinusoids is $P_1 = |A_1|^2$.

Frequency estimation, eigenvalues and eigenvectors.

Eigenvalues and eigenvectors for sinusoids in white noise.

Multiply R_x **with** e_1 ,

$$R_x e_1 = (P_1 e_1 e_1^H + \sigma_w^2\ I)\, e_1 =$$
$$= (P_1 e_1 e_1^H\ e_1 + \sigma_w^2\ \ e_1) =$$
$$= (P_1 N + \sigma_w^2\ )\, e_1$$

We now identify one eigenvalue and corresponding eigenvector

$$R_x e_1 = (P_1 N + \sigma_w^2\ )\, e_1$$

$$\lambda_1 = P_1 N + \sigma_w^2 \qquad (\lambda_{\max})$$
$$v_1 = e_1 \qquad (only\ if\ \ p = 1)$$

Frequency estimation, eigenvalues and eigenvectors.

The other eigenvectors must be orthogonal to eigenvector 1.

$$R_x v_i = (P_1 e_1 e_1^H + \sigma_w^2\, I\,)\, v_i =$$
$$= \sigma_w^2\, v_i \qquad\qquad i = 2,3,\dots,N$$
$$\lambda_2 = \lambda_3 = \dots = \lambda_N = \sigma_w^2 \qquad (\lambda_{\min})$$

The signal subspace is determined by v_1

The noise subspace is determined by v_i ,i=2,...,N

Frequency estimation.

A: Estimate R_x and determine the eigenvalues and eigenvectors.

B: Estimate the variance of the noise as $\sigma_w^2 = \lambda_{\min}$ **.**

C: Estimate the signal power as

$$\hat{P}_1 = \frac{\lambda_{\max} - \lambda_{\min}}{N}$$

Note that

$$\lambda_1 = P_1 N + \sigma_w^2$$
$$v_1 = e_1$$

D: Estimate the frequency from the eigenvector 1.

$$\omega_1 = \arg\{v_1(1)\}$$ **(second index)**

Frequency estimation,
Frequency estimation function

The eigenvectors v_2 **to** v_N **are orthogonal to** $v_1 = e_1$.

$$e_1^H v_i = 0, \quad i = 2, \dots, N$$

But

$$V_i(e^{j\omega}) = e^H v_i = \sum_{k=0}^{N-1} v_i(k) e^{-j\omega k}$$

For $\omega = \omega_1$ **we have**

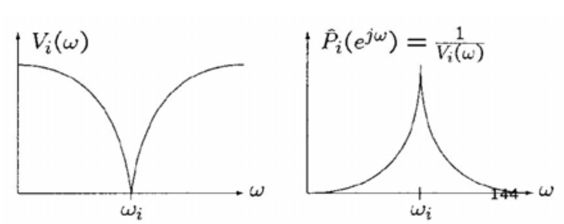
$$V_i(e^{j\omega_1}) = e_1^H v_i = \sum_{k=0}^{N-1} v_i(k) e^{-j\omega_1 k} = 0$$

This is valid for all eigenvectors v_2 **to** v_N.

We define the frequency estimation function as

$$\hat{P}_i(e^{j\omega}) = \left| \frac{1}{V_i(e^{j\omega k})} \right|^2 = \frac{1}{\left| \sum_{k=0}^{N-1} v_i(k) e^{-j\omega k} \right|^2} =$$

$$= \frac{1}{|e^H v_i|^2}$$



We can also compute the Z-transform

$$V_i(z) = \sum_{k=0}^{N-1} v_i(k) z^{-k}$$

and determine the zeroes of $V_i(z)$

Frequency estimation.

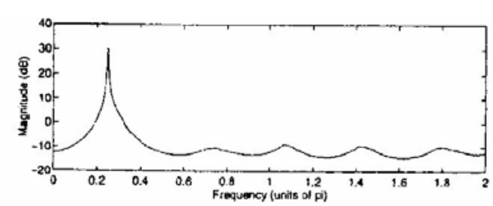
Averaging over all noise eigenvectors yield

$$\hat{P}(e^{j\omega}) = \frac{1}{\sum_{i=2}^{N} \alpha_i |e^H v_i|^2}$$

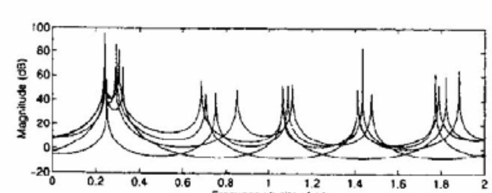
Example from the textbook page 455

Upper figure: **Averaging over the noise eigenvectors with the weight equal to one.**

Lower figure: **Overlay plot over the frequency estimation function from each of the noise eigenvectors**



(a)



Frequency estimation.
Several sinusoids in white noise

$$x(n) = \sum_{i=1}^{p} A_i \, e^{j\omega_i n} + w(n)$$

and for p=2

$$R_x = E\{x\, x^H\} =$$
$$= E\{(A_1 e_1 + A_2 e_2 + w)(A_1 e_1 + A_2 e_2 + w)^H\} =$$
$$= P_1 e_1 e_1^H + P_2 e_2 e_2^H + \sigma_w^2 \; I$$

Eigenvaules

$$\lambda_i \approx \begin{cases} \upsilon_i + \sigma_w^2 & signal \;\; subspace \\ \sigma_w^2 & noise \;\; subspace \end{cases}$$
$$\upsilon_i \; are \; the \; eigenvalues \; in \; the \; signal \;\; subspace$$

The frequency estimation function is now

$$\hat{P}\,(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{N} \alpha_i \, | \, e^H \, v_i \, |^2}$$

We will now look at some methods using the frequency estimation function above.

The first is called the Pisarenco Decomposition method. This method is very sensitive to the noise but describe the principle for the methods.

A well known method is the MUSIC algorithm.

The frequency estimation function is sometimes called the pseudospectrum or eigenspectrum.

Frequency estimation: Pisarenco

1: Assume *p* complex sinusoids in white noise

2: Assume the dimension of R_x (p+1)*(p+1), i.e. only one noise eigenvector.

This assumptions means that only one eigenvector corresponds to the noise subspace.

Then $\lambda_{\min} = \lambda_{p+1} = \sigma_w^2$

and the frequency estimation function (pseudospectrum) is defined

$$\hat{P}_{PHD}(e^{j\omega}) = \frac{1}{| \sum_{k=0}^{p} v_{\min}(k) \, e^{-j\omega k} \, |^2} =$$
$$= \frac{1}{| \, e^H \, v_{\min} \, |^2}$$

$$V_{\min}(z) = \sum_{k=0}^{p} v_{\min}(k) \, z^{-k}$$

Frequency estimation: MUSIC
Page 464, 465

MUSIC: MUltiple SIgnal Characterization

The frequency estimation is achieved by averaging the pseudospectra over the noise eigenvectors.

$$\hat{P}_{MU}(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{M} | \, e^H \, v_i \, |^2}$$

Then estimate the position of the peaks in $\hat{P}_{MU}(e^{j\omega})$

Principal Components Spectrum Estimation.

These methods use the signal subspace. (page 470-471)

The Blackman-Tukey power spectrum was determined from a windowed autocorrelation sequence

$$\hat{P}_{BT}(e^{j\omega}) = \sum_{k=-N+1}^{N-1} \hat{r}_x(k)\, w(k)\, e^{-j\omega k}$$

If w(k) is a Bartlett window, the Blackman-Tukey estimate can be written in terms of the autocorrelation matrix

$$\hat{P}_{BT}(e^{j\omega}) = \frac{1}{M}\sum_{k=-M}^{M}(M-|k|)\hat{r}_x(k)e^{j\omega k} = \frac{1}{M}e^H R_x e$$

In terms of eigenvectors (eigendecomposition) this is

$$\hat{P}_{BT}(e^{j\omega}) = \frac{1}{M}\sum_{i=1}^{M} \lambda_i\, |e^H v_i|^2$$

Now, use only the eigenvectors corresponding to the sinusoids. Then the Blackman-Tukey principal frequency estimation is defined by

$$\hat{P}_{PC-BT}(e^{j\omega}) = \frac{1}{M}\sum_{i=1}^{p} \lambda_i\, |e^H v_i|^2$$

The minimum variance power spectrum estimate was defined by

$$P_{MV}(e^{j\omega}) = \frac{M}{e^H R_x^{-1} e}$$

Rewrite this in terms of eigenvectors and only use eigenvectors corresponding to the sinusoids gives the minimum variance frequency estimation

$$P_{PC-MV}(e^{j\omega}) = \frac{M}{\sum_{i=1}^{p} \frac{1}{\lambda_i} e^H v_i}$$

Table 8.10 ***Noise Subspace Methods for Frequency Estimation***

Pisarenko	$\hat{P}_{PHD}(e^{j\omega}) = \frac{1}{\lvert\mathbf{e}^H \mathbf{v}_{\min}\rvert^2}$
MUSIC	$\hat{P}_{MU}(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{M} \lvert\mathbf{e}^H \mathbf{v}_i\rvert^2}$
Eigenvector Method	$\hat{P}_{EV}(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{M} \frac{1}{\lambda_i}\lvert\mathbf{e}^H \mathbf{v}_i\rvert^2}$
Minimum Norm	$\hat{P}_{MN}(e^{j\omega}) = \frac{1}{\lvert\mathbf{e}^H \mathbf{a}\rvert^2}$; $\mathbf{a} = \lambda \mathbf{P}_n \mathbf{u}_1$

Table 8.11 ***Signal Subspace Methods***

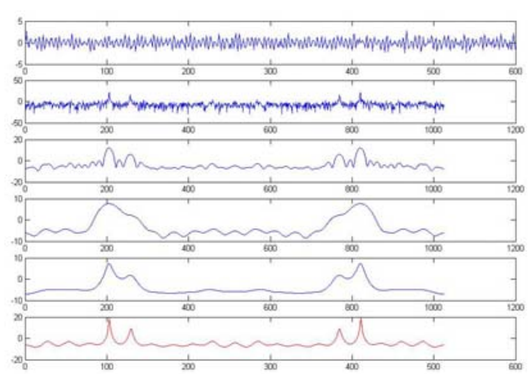
Blackman-Tukey	$\hat{P}_{PC-BT}(e^{j\omega}) = \frac{1}{M}\sum_{i=1}^{p} \lambda_i \lvert\mathbf{e}^H \mathbf{v}_i\rvert^2$
Minimum variance	$\hat{P}_{PC-MV}(e^{j\omega}) = \frac{M}{\sum_{i=1}^{p} \frac{1}{\lambda_i}\lvert\mathbf{e}^H \mathbf{v}_i\rvert^2}$
Autoregressive	$\hat{P}_{PC-AR}(e^{j\omega}) = \frac{1}{\left\lvert \sum_{i=1}^{p} \alpha_i \mathbf{e}^H \mathbf{v}_i \right\rvert^2}$

Example of sinusoids in white noise

Power spectrum estimation

$x(n) = \sin(2\pi \cdot 0.20 \cdot n) + 0.5\sin(2\pi \cdot 0.25 \cdot n) + v(n)$

with $v(n)$ = white noise with varaiance 1



Row 1: Waveform of input signal x(n)
Row 2 FFT of x(n), N=1024 (Periodogram)

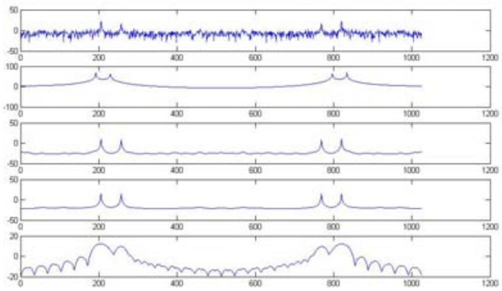
Row 3 Averaging with the Welch method (10 subintervals, rectangular time window)
Row 4 Blackman-Tukey estimate with M=20 (hamming window)
Row 5 Minimum variance method with M=20,
Row 6 All-pole model of order M=20 (Levinson Durbin algorithm).

(All spectra in 1024 frequency points. y axis in dB)

Example of sinusoids in white noise

Frequency estimation methods

$$x(n) = \sin(2\pi \cdot 0.20 \cdot n) + 0.5\sin(2\pi \cdot 0.25 \cdot n) + v(n)$$
$$\text{with} \quad v(n) = \text{white noise with varaiance } 1$$



Row 1 FFT of x(n), N=1024 (Periodogram).
Row 2 Pisarenco Harmonic Decomposition p=4, M=5.
Row 3 The MUSIC algorithm p=4, M=30.
Row 4 The Eigenvector method (EV) p=4, M=30.
Row 5 Principal components Blackman-Tukey frequency estimation (PC-BT) p=4, M=30.

(All spectra in 1024 frequency points, y axis in dB)

Optimal Signal Processing 2008

A brief review

We have focused on methods used in practical applications

All-pole modeling in chapter 4 (LPC, Prediction Error Filter)

Levinson Durbin Recursion using the reflection parameters Γ **in chapter 5.**

Lattice structure in chapter 6, Burgs algorithm

Wiener FIR Filters in chapter 7

Power Spectrum Estimation using the Periodogram in chapter 8
Frequency estimation (MUSIC)

Chapter 3

Filtering random processes (real signals)

input: $x(n)$
output: $y(n)$

$$y(n) = x(n) * h(n) = \sum_k x(k)h(n-k)$$

$$r_y(k) = r_x(k) * h(k) * h(-k)$$
$$r_{yx}(k) = h(k) * r_x(k)$$
$$r_{xy}(k) = h(-k) * r_x(k)$$

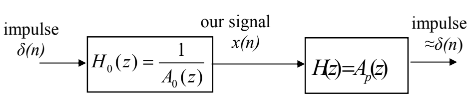
$$P_y(e^{j\omega}) = P_x(e^{j\omega}) H(e^{j\omega}) H^*(e^{j\omega})$$
$$P_{yx}(e^{j\omega}) = P_x(e^{j\omega}) H(e^{j\omega})$$
$$P_{xy}(e^{j\omega}) = P_x(e^{j\omega}) H^*(e^{j\omega})$$

$$P_y(z) = P_x(z) H(z) H(z^{-1})$$
$$P_{yx}(z) = P_x(z) H(z)$$
$$P_{xy}(z) = P_x(z) H(z^{-1})$$

Chapter 4 System modeling

All-pole modeling

This can be described by the following figure.



$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k) z^{-k}$$

$$H_0(z) = \frac{1}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}}$$

Minimize $\varepsilon_p = \sum_{n=0}^{\infty} |e(n)|^2$ **with** $e(n) = x(n) + \sum_{k=1}^{p} a_p(k)\, x(n-k)$

gives the normal equations

$$\sum_{l=1}^{p} a_p(l)\, r_x(k-l) = -r_x(k) \qquad k = 1, \ldots, p$$

$$\underbrace{\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3) \\ & & \cdots & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix}}_{R_x} \cdot \underbrace{\begin{bmatrix} a_p(1) \\ a_p(2) \\ a_p(3) \\ \cdot \\ a_p(p) \end{bmatrix}}_{a_p} = - \begin{bmatrix} r_x(1) \\ r_x(2) \\ r_x(3) \\ \cdot \\ r_x(p) \end{bmatrix}$$

$$\varepsilon_{p,\min} = \varepsilon_p = r_x(0) + \sum_{k=1}^{p} a(k)\, r_x(k)$$

Chapter 5 Levinson-Durbin recursion

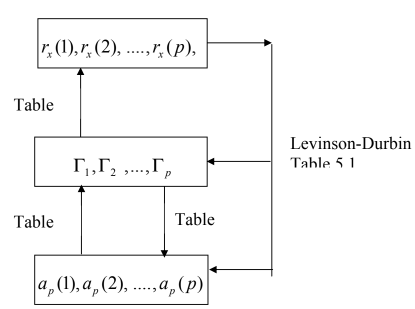
The Levinson-Durbin algorithm determine relation between autocorrelation ***r(x)*****,polynomial** ***a(k)*** **and the reflection coefficients.**

The matrix equation can be written

$$\underbrace{\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3) \\ & & \cdots & & \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix}}_{R_x} \cdot \underbrace{\begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \cdot \\ a_p(p) \end{bmatrix}}_{a_p} = \underbrace{\begin{bmatrix} \varepsilon_p \\ 0 \\ 0 \\ \cdot \\ 0 \end{bmatrix}}_{\varepsilon_p u_1}$$

$$R_x \; a_p = \varepsilon_p u_1$$

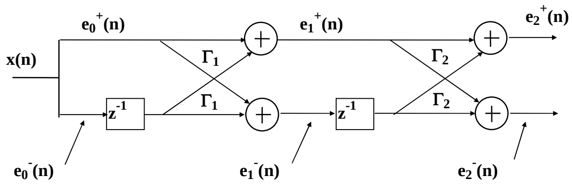
This can be summarized in the figure below and in the table 5.1 –5.4.



c

Chapter 6 Lattice Structure

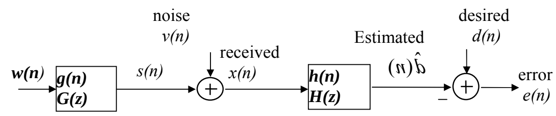
FIR $H(z)=A(z)$



Burgs algorithm

Lattice FIR
Lattice IIR

Chapter 7. Wiener filters



We will minimize the output error *e(n)*, which we describe as the difference between the desired output *d(n)* and the estimated output.

Applications.

Filtering s(n): **The desired signal is *s(n)* and we will determine the optimum filter for noise reduction.**
Smoothing: **Like filtering but we allow an extra delay in the output signal (specially image processing).**
Prediction: **The output is a prediction of future values of *s(n)*. One step predictor. predict next value *s(n+1)*.**
Equalization: **The desired signal is *w(n)* and we will determine the optimum filter for whitening the output spectrum (inverse filtering, deconvolution).**

Other applications: **Echo cancellation. Noise cancellation. Pulse shaping.**

Minimizing

$$\xi = E[e^2(n)] = E[(d(n)-\hat{d}(n))^2]$$

gives the wiener-Hopf equations

$$\sum_{l=-\infty}^{\infty} h(l) r_x(k-l) = r_{dx}(k)$$

$$\xi_{\min} = r_d(0) - \sum_{l=-\infty}^{\infty} h(l) r_{dx}(l)$$

FIR Wiener filter

$$\sum_{l=0}^{p-1} h(l) r_x(k-l) = r_{dx}(k) \quad k = 0,1,\ldots,p-1$$

Non-causal IIR Wiener filter

$$H(e^{j\omega}) = \frac{P_{dx}(e^{j\omega})}{P_x(e^{j\omega})}$$

Causal IIR Wiener filter

$$H(z) = \frac{1}{\sigma^2 Q(z)} \underbrace{\left[\frac{P_{dx}(z)}{Q(z^{-1})}\right]_+}_{\text{Take the causal part of this}}$$

FIR-filter for noise reduction

$$h_{opt} = (R_s + R_v)^{-1} \; r_s$$

Non-causal IIR-filter for noise reduction

$$H(e^{j\omega}) = \frac{P_s(e^{j\omega})}{P_s(e^{j\omega}) + P_v(e^{j\omega})};$$

Chapter 8 Power Spectrum Estimation

The Fourier Transform of $\hat{r}_x(k)$ **is called the periodogram**

$$\hat{P}_{per}(e^{j\omega}) = \sum_{k=-N+1}^{N-1} \hat{r}_x(k)\; e^{-j\omega k}$$

We see that it also can be written

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{N} X_N(e^{j\omega}) X_N^*(e^{j\omega}) = \frac{1}{N} |X_N(e^{j\omega})|^2$$

Using DFT (FFT), the periodogram will be

$$\hat{P}_{per}(e^{j2\pi k/N}) = \frac{1}{N} X_N(k) X_N^*(k) = \frac{1}{N} |X_N(k)|^2$$

Averaging periodogram.
- **Bartlett's Method**
- **Welch's method**

Blackman-Tukey method

Minimum variance method

Frequency estimation (Estimation of sinusoids)

- **Pisarenco Harmonic Decomposition**
- **The MUSIC algorithm**
- **The Eigenvector method (EV)**
- **Principal components Blackman-Tukey frequency estimation**