Aalto University

T-61.3040 Statistical Signal Modeling

Autumn 2011

Amaury Lendasse & Yoan Miche

Today's Topics (18.10)

- More examples on modeling for AR, MA, ARMA models
- Power Spectrum estimation (a bit, discussed more in the next lectures)
- Some examples using Matlab

About last week's question

Yes, this is obviously linear in the coefficients $a_p(k)$ and $b_q(k)$

$$\left[\begin{array}{cccc} x(0) & 0 & \cdots & 0 \\ x(1) & x(0) & \cdots & 0 \\ x(2) & x(1) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ \hline x(q+1) & x(q) & \cdots & x(q-p+1) \\ \vdots & \vdots & \cdots & \vdots \\ x(q+p) & x(q+p-1) & \cdots & x(q) \end{array}\right] \left[\begin{array}{c} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{array}\right] = \left[\begin{array}{c} b_q(0) \\ b_q(1) \\ b_q(2) \\ \vdots \\ b_q(q) \\ \hline 0 \\ \vdots \\ 0 \end{array}\right]$$

About last week's question

But that is because we have reformulated the system function to be so:

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

as

$$H(z) A_p(z) = B_q(z)$$

So you could solve the whole system at once, or use the Padé approximation and solve the lower part in $a_p(k)$ and then the upper part in $b_q(k)$

For stochastic models

- This was all very beautiful, but meant for deterministic signals: we know the whole of $x(n)$ values or on some fixed known interval
- What of stochastic models?
- Well, we cannot use the previously defined errors meant for deterministic models, such as

$$\varepsilon_p = \sum_{n=q+1}^{\infty} |e(n)|^2 = \sum_{n=q+1}^{\infty} \left| x(n) + \sum_{k=1}^{p} a_p(k) x(n-k) \right|^2$$

because $x(n)$ is only known probabilistically

For stochastic models

- Need other criteria to minimize
- Also, to model we need a random process as an input to the system
- We used to have a unit sample, for deterministic signals
- Now, for probabilistic, use unit variance white noise (remember we used it when we discussed random processes)

Signal Modeling for ARMA Models

- General case: ARMA models
- Remember we can get an ARMA process by filtering the unit variance white noise $v(n)$ by a causal LSI filter with p poles and q zeroes

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

Signal Modeling for ARMA Models

- Now, using the same idea as for the Least Squares for deterministic signals, we could use a Mean Square error

$$\varepsilon_{MS} = E\left\{ \left| x(n) - \hat{x}(n) \right|^2 \right\}$$

- And we will have also (possibly) difficult to solve non-linear equations
- Now, remember the Yule-Walker equations?

Signal Modeling for ARMA Models

For an ARMA process, we had

$$r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = \begin{cases} \sigma_v^2 c_q(k) & , 0 \leq k \leq q \\ 0 & , k > q \end{cases}$$

with $c_q(k) = \sum_{l=0}^{q-k} b_q(l+k) h^*(l)$, and here $\sigma_v^2 = 1$

Signal Modeling for ARMA Models

So, for $k > q$, we have linear equations in $a_p(k)$

$$\begin{bmatrix} r_x(q) & r_x(q-1) & \cdots & r_x(q-p+1) \\ r_x(q+1) & r_x(q) & \cdots & r_x(q-p+2) \\ \vdots & \vdots & \ldots & \vdots \\ r_x(q+p-1) & r_x(q+p-2) & \cdots & r_x(q) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} r_x(q+1) \\ r_x(q+2) \\ \vdots \\ r_x(q+p) \end{bmatrix}$$

For which the matrix holding most of the autocorrelations in Toeplitz, as for the Padé approximation

Signal Modeling for ARMA Models

- In fact, these Modified Yule-Walker Equations are very similar to the Padé equations: The values of the sequence $x(n)$ are here replaced by the ones of a certain range of the autocorrelation sequence
- So, we get the AR coefficients $a_p(k)$ rather easily
- Now, for the MA coefficients $b_q(k)$, given that we have the $a_p(k)$, we can write the set of equations for $k < q$

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(p) \\ r_x(1) & r_x(0) & \cdots & r_x^*(p-1) \\ \vdots & \vdots & \cdots & \vdots \\ r_x(q) & r_x(q+1) & \cdots & r_x(q) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ \vdots \\ a_p(p) \end{bmatrix} = \begin{bmatrix} c_q(0) \\ c_q(1) \\ \vdots \\ c_q(q) \end{bmatrix}$$

Signal Modeling for ARMA Models

- Now, we are going to use the power spectrum properties and spectral factorization to solve the $b_q(k)$
- We know that $c_q(k) = 0, \forall k > q$, so $c_q(k)$ is known for all $k \geq 0$. Denote the "positive time" portion of the z-transform of $c_q(k)$ by $[C_q(z)]^+$

$$[C_q(z)]^+ = \sum_{k=0}^{\infty} c_q(k) z^{-k}$$

and similarly the negative time part by

$$[C_q(z)]^- = \sum_{k=-\infty}^{-1} c_q(k) z^{-k} = \sum_{k=1}^{\infty} c_q(-k) z^{k}$$

Signal Modeling for ARMA Models

- We have defined $c_q(k)$ as the convolution of $b_q(k)$ with $h^*(-k)$, so

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

- Recognize the power spectrum of an MA(q) process in the upper part?

Signal Modeling for ARMA Models

- We have

$$P_y(z) = C_q(z)A_p^*(1/z^*) = B_q(z)B_q^*(1/z^*)$$

- And $a_p(k) = 0, \forall k < 0$, so we have

$$P_y(z) = [C_q(z)]^+ A_p^*(1/z^*) + [C_q(z)]^- A_p^*(1/z^*)$$

- Of which the causal part is

$$[P_y(z)]^+ = \left[[C_q(z)]^+ A_p^*(1/z^*)\right]^+$$

since $[C_q(z)]^-$ and $A_p^*(1/z^*)$ only contain positive powers of z

Signal Modeling for ARMA Models

- Then, using the conjugate symmetry of the power spectrum, we can determine the whole $P_y(z)$
- Finally, using spectral factorization on the obtained equation, we can get

$$P_y(z) = B_q(z)B_q^*(1/z^*)$$

and solve the B_q

Signal Modeling for ARMA Models: Example

- For an ARMA(1,1) model
- Assume we have the autocorrelation values

$$r_x(0) = 26;\ r_x(1) = 7;\ r_x(2) = 7/2$$

- We have the Yule-Walker equations

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

Signal Modeling for ARMA Models: Example

- And the Modified Yule-Walker equations are

$$r_x(1)a_1(1) = -r_x(2)$$

  from which $a_1(1) = -r_x(2)/r_x(1) = -1/2$

- And for the MA coefficients, begin by computing the $c_q(k)$ coefficients using the Yule-Walker equations:

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

Signal Modeling for ARMA Models: Example

- Which given the values for $r_x(k)$ and that $a_1(1) = -1/2$, we get $c_1(0) = 45/2$ and $c_1(1) = -6$ so that we can express

$$[C_1(z)]^+ = 45/2 - 6z^{-1}$$

- Multiplying by $A_1^*(1/z^*) = 1 - 0.5z$, the causal part of the power spectrum $[P_y(z)]^+ = [C_1(z)]^+ A_p^*(1/z^*)$ is expressed by

$$[P_y(z)]^+ = \left[[C_1(z)]^+ A_p^*(1/z^*)\right]^+ = 51/2 - 6z^{-1}$$

Signal Modeling for ARMA Models: Example

- With the conjugate symmetry property of $P_y(z)$, we have

$$P_y(z) = C_1(z)A_1^*(1/z^*) = -6z + 51/2 - 6z^{-1} = B_1(z)B_1^*(1/z^*)$$

- And using a spectral factorization on this polynomial, we get the $B_1(z)$ coefficients, to get the final system function

$$H(z) = 2\sqrt{6}\frac{1 - 0.25z^{-1}}{1 - 0.5z^{-1}}$$

Signal Modeling for AR Models

- This time, we know we can generate an AR process by filtering unit variance white noise $v(n)$ by an all pole filter of the form

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

- In the same fashion as for an ARMA process, we have the Yule-Walker equations for the autocorrelation sequence

$$r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = |b(0)|^2 \delta(k), \forall k \geq 0$$

Signal Modeling for AR Models

- So writing the equations in matrix form for $k > 0$, we have

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

- So given the autocorrelation sequence, we can have directly the $a_p(k)$

Signal Modeling for AR Models

- Getting the coefficient $b(0)$ is not so hard, from the Yule-Walker equations:

$$|b(0)|^2 = r_x(0) + \sum_{k=1}^{p} a_p(k) r_x(k)$$

- Note (check yourself): If you need to estimate the autocorrelation, using e.g. the sample autocorrelation

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

then the deterministic and stochastic all-pole modeling (i.e. AR) are equivalent

Signal Modeling for MA Models

- This time, to obtain a MA model, we filter unit variance white noise $v(n)$ by an FIR filter of order q as

$$x(n) = \sum_{k=0}^{q} b_q(k) v(n-k)$$

- The Yule-Walker equations are in this case

$$r_x(k) = b_q(k) * b_q^*(-k) = \sum_{l=0}^{q-|k|} b_q(l+|k|) b_q^*(l)$$

which are nonlinear in the filter coefficients

Signal Modeling for MA Models

- We will use the same idea of spectral factorization as for the ARMA model
- Since the autocorrelation is zero for $|k| > q$ (by design of the system), the power spectrum is of the form

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

Signal Modeling for MA Models

- And using spectral factorization, we can have

$$P_x(z) = \sigma_0^2 Q(z) Q^*(1/z^*) = \sigma_0^2 \prod_{k=1}^{q} (1 - \alpha_k z^{-1}) \prod_{k=1}^{q} (1 - \alpha_k^* z)$$

- In the end, one can model the output process $x(n)$ as the output of the FIR filter

$$H(z) = \sigma_0 Q(z) = \sigma_0 \sum_{k=0}^{q} q(k) z^{-k}$$

with the $q(k)$ the coefficients of $Q(z)$

Power Spectrum Estimation

- We play with the power spectrum quite much to find the system coefficients
- How about estimating it?
- We have defined the power spectrum $P_x(e^{j\omega})$ as

$$P_x(e^{j\omega}) = \sum_{k=-\infty}^{\infty} r_x(k) e^{-jk\omega}$$

with

$$r_x(k) = E\left\{x(n)x^*(n-k)\right\}$$

Power Spectrum Estimation

- In the general case, the autocorrelation sequence is unknown and we have to estimate the power spectrum from a sample realization $x(n), n = 1, \ldots, N$
- Estimating $P_x(e^{j\omega})$:
  - Using estimates of the autocorrelation: If we indeed have only $N + 1$ values of $x(n)$, the autocorrelation can only be estimated for $|k| \leq N$
  - So, since we use estimates for the autocorrelations, the power spectrum based on them is also approximate
  - Also, since we only have a subset of all required autocorrelation values for the power spectrum, it will be limited in resolution
  - Better solution: Have a priori knowledge on the process at hand

Power Spectrum Estimation: Example

- If we know $x(n)$ to be an AR(p) process, then we have the power spectrum form instantly

$$P_x(e^{j\omega}) = \frac{|b(0)|^2}{\left|1 + \sum_{k=1}^{p} a_p(k) e^{jk\omega}\right|^2}$$

- And then we can use the Yule-Walker method with estimated autocorrelations to estimate the b_q and a_p

Power Spectrum Estimation: Example

Assume we have $N = 64$ samples of an AR(4) process generated by filtering unit variance white noise by the fourth order all pole filter $H(z) = \frac{b(0)}{1+\sum_{k=1}^{4} a(k)z^{-k}}$

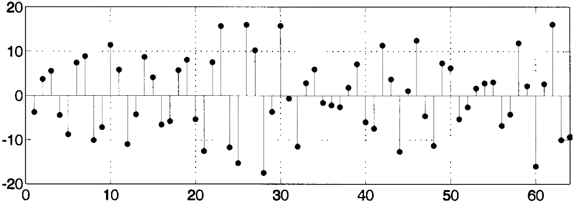

Figure: AR(4) random process samples

Power Spectrum Estimation: Example

Estimating the autocorrelation $\hat{r}_x(k)$ and substituting these estimates directly in the power spectrum formula gives the following power spectrum (dashed is true psd)

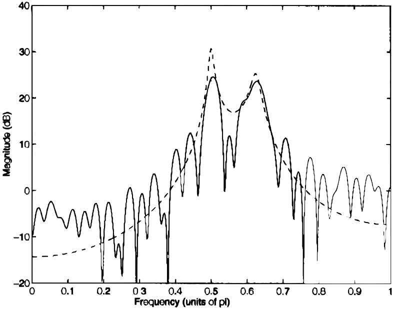

Figure: Power spectrum obtained using the estimated autocorrelations

Power Spectrum Estimation: Example

While using the estimated autocorrelations and the Yule-Walker equations for the filter coefficients (knowing its form)

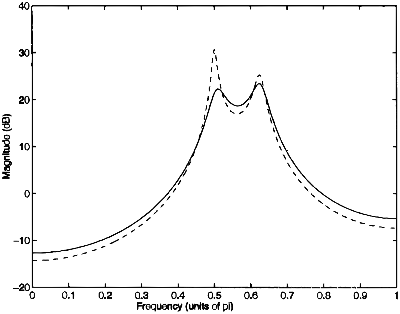

Figure: Power spectrum obtained using the estimated filter coefficients and Yule-Walker equations

Power Spectrum Estimation: Example

- But to do well, you need a lot of a priori information: type/form of the system and order (AR(p) for example)
- More on power spectrum estimation in two weeks

Levinson-Durbin Recursion

- Not mandatory to know or anything
- Manner of computing solution to equations involving a Toeplitz matrix
- Runs in $O(n^2)$ instead of the usual $O(n^3)$ for the Gauss-Jordan elimination
- Some special versions even faster for large n
- Try to play with Matlab and the `levinson` command

A summary of what we have so far

- Following slides are a quick summary of the previous lectures
- Look the original lecture for more explanations

Stationarity

- "Statistical time-invariance"
- L-th order stationarity: "$x(n)$ and $x(n+k)$ have the same L-th order joint density function"
- *Wide-Sense Stationarity* (WSS) requires all three:
  - Mean of the process is a constant, $m_x(n) = m_x$
  - Autocorrelation $r_x(k,l)$ depends only on the difference $k-l$ (and not k and l separately)
  - The variance $c_x(0)$ is finite
- Wide-Sense Stationarity (WSS) is *weaker* than second-order stationarity (constraints on moments, not on density functions directly)

Autocorrelation

- General probabilistic sense: $r_x(k,l) = E\left\{x(k)x^*(l)\right\}$
- General deterministic sense: $r_x(k,l) = \sum_{n=0}^{\infty} x(n-k)x^*(n-l)$
- Second (and above) order stationary process: $r_x(k,l) = r_x(k-l,0) \equiv r_x(k-l)$
- For a WSS process:
    - $r_x(k) = r_x^*(-k)$
    - *Mean-square value*: $r_x(0) = E\left[|x(n)|^2\right] \geq 0$
    - *Maximum value*: $|r_x(k)| \leq r_x(0)$
    - *Periodicity*: If $r_x(k_0) = r_x(0)$ for some k_0, then $r_x(k)$ is *periodic* with period k_0

Ergodicity

- "Each member of the process has the same statistical behavior as the entire process"
- Required property to estimate autocorrelation and mean using time estimates
- Usually, we will assume ergodicity if we need to estimate mean, autocorrelation, and such, using time averages

Ergodicity

For WSS processes:

1. *Mean Ergodic Theorem 1*: With $x(n)$ a WSS process of autocovariance $c_x(k)$. It is necessary and sufficient for $x(n)$ to be ergodic in the mean that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} c_x(k) = 0$$

2. *Mean Ergodic Theorem 2*: With $x(n)$ a WSS process of autocovariance $c_x(k)$. It is sufficient for $x(n)$ to be ergodic in the mean that $c_x(0) < \infty$ and that

$$\lim_{k\to\infty} c_x(k) = 0$$

Autocovariance

- $c_x(k,l) = E[(x(k) - m_x(k))(x(l) - m_x(l))^*]$
- $c_x(k,l) = r_x(k,l) - m_x(k)m_x^*(l)$

White noise

- A WSS process $v(n)$ is said to be *white* if the autocovariance is 0 for all $k \neq 0$, i.e.

$$c_v(k) = \sigma_v^2 \delta(k)$$

  with $\delta(k)$ the unit sample
- Sequence of uncorrelated random variables, each with variance σ_v^2
- PSD is then a constant: $P_v(e^{j\omega}) = \sigma_v^2$

Power spectrum/Power Spectral Density

- Wiener–Khinchin theorem states that the Power Spectral Density $P_x(e^{j\omega})$ is the Fourier transform of the autocorrelation function $r_x(k)$

$$P_x(e^{j\omega}) = \sum_{k=-\infty}^{\infty} r_x(k) e^{-jk\omega}$$

- PSD and autocorrelation function form a Fourier transform pair, i.e.

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

Power spectrum/Power Spectral Density

- *Symmetry*: If $x(n)$ is WSS, then the PSD is real-valued (i.e. $P_x(e^{j\omega}) = P_x^*(e^{j\omega})$) and $P_x(z)$ (z-transform version of the PSD) satisfies the symmetry

$$P_x(z) = P_x^*(1/z^*)$$

- *Positivity*: The PSD of a WSS process is nonnegative: $P_x(e^{j\omega}) \geq 0$

Power spectrum/Power Spectral Density

- *Total Power*: The power in a zero mean WSS process is proportional to the area under the PSD curve,

$$E\left[|x(n)|^2\right] = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})d\omega$$

- The spectral factorization of $P_x(e^{j\omega})$ aims at expressing it as

$$P_x(z) = \sigma_0^2 Q(z) Q^*(1/z^*)$$

Power spectrum/Power Spectral Density

- ARMA(p,q): if the filter $H(z)$ is stable, output process $x(n)$ is WSS and if $P_v(z) = \sigma_v^2$, we have the power spectrum

$$P_x(z) = \sigma_v^2 \frac{B_q(z)B_q^*(1/z^*)}{A_p(z)A_p^*(1/z^*)}$$

- AR(p): with same assumptions

$$P_x(z) = \sigma_v^2 \frac{|b(0)|^2}{A_p(z)A_p^*(1/z^*)}$$

- MA(q): same assumptions

$$P_x(z) = \sigma_v^2 B_q(z)B_q^*(1/z^*)$$

Linear Shift-Invariance

- Linearity of a discrete-time system: with two inputs $x_1(n)$ and $x_2(n)$ and two constants a and b, we have

$$T\left[ax_1(n) + bx_2(n)\right] = aT\left[x_1(n)\right] + bT\left[x_2(n)\right]$$

- Shift-invariance of a discrete-time system: If a shift in the input results in the same shift in the output. That is, if we input $x(n - n_0)$, the output is $y(n - n_0)$.
- If input $x(n)$ is WSS, then output $y(n)$ is also WSS if $\sigma_y^2 < \infty$ (which requires the filter to be stable)
- If $h(n)$ is finite in length and zero outside $[0, N-1]$ on which $x(n)$ is defined

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

Wold Decomposition Theorem

Wold Decomposition Theorem: any WSS random process $x(n)$ can be written as the sum of two processes $x_{\text{pred}}(n)$ and $x_{\text{reg}}(n)$, where $x_{\text{pred}}(n)$ is a *predictable* process and $x_{\text{reg}}(n)$ a *regular* process, with $x_{\text{reg}}(n)$ and $x_{\text{pred}}(n)$ *orthogonal*, i.e.

$$E\left[x_{\text{reg}}(m)x^*_{\text{pred}}(n)\right] = 0$$

Yule-Walker equations

- With σ_v^2 the variance of the white noise $v(n)$ filtered by a LSI causal filter

$$H(z) = \frac{B_q(z)}{A_p(z)}$$

- ARMA(p, q) process:

$$r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = \begin{cases} \sigma_v^2 c_q(k) & , 0 \leq k \leq q \\ 0 & , k > q \end{cases}$$

with $c_q(k) = \sum_{l=0}^{q-k} b_q(l+k) h^*(l)$

Yule-Walker equations

- AR(p) process:

$$r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = \sigma_v^2 \left|b(0)\right|^2 \delta(k), k \geq 0$$

- MA(q) process:

$$r_x(k) = \sigma_v^2 b_q(k) * b_q^*(-k) = \sigma_v^2 \sum_{l=0}^{q-|k|} b_q(l+|k|) b_q^*(l)$$

Signal Modeling: LS Method

- Minimize

$$\varepsilon_{LS} = \sum_{n=0}^{\infty} \left| e'(n) \right|^2$$

- Meaning

$$\begin{cases} \frac{\partial \epsilon_{LS}}{\partial a_p^*(k)} = 0, & \forall k \in [\![1, p]\!] \\ \frac{\partial \epsilon_{LS}}{\partial b_q^*(k)} = 0 & \forall k \in [\![0, q]\!] \end{cases}$$

- Hard and mathematically intractable usually

Signal Modeling: Padé approximation

- Set of linear equations solved approximately
- Express system function as

$$H(z)A_p(z) = B_q(z)$$

- Expressed in time domain, set of equations

$$x(n) + \sum_{k=1}^{p} a_p(k)x(n-k) = \begin{cases} b_q(n) & , n \in [\![0, q]\!] \\ 0 & , n \in [\![q+1, q+p]\!] \end{cases}$$

Signal Modeling: Padé approximation

In matrix form

$$\left[\begin{array}{cccc} x(0) & 0 & \cdots & 0 \\ x(1) & x(0) & \cdots & 0 \\ x(2) & x(1) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ \hline x(q+1) & x(q) & \cdots & x(q-p+1) \\ \vdots & \vdots & \cdots & \vdots \\ x(q+p) & x(q+p-1) & \cdots & x(q) \end{array}\right] \left[\begin{array}{c} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{array}\right] = \left[\begin{array}{c} b_q(0) \\ b_q(1) \\ b_q(2) \\ \vdots \\ b_q(q) \\ \hline 0 \\ \vdots \\ 0 \end{array}\right]$$

Signal Modeling: Padé approximation

Solve the lower part for the $a_p(k)$ (reformulated)

$$\begin{bmatrix} x(q) & x(q-1) & \cdots & x(q-p+1) \\ x(q+1) & x(q) & \cdots & x(q-p+2) \\ \vdots & \vdots & \ldots & \vdots \\ x(q+p-1) & x(q+p-2) & \cdots & x(q) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} x(q+1) \\ x(q+2) \\ \vdots \\ x(q+p) \end{bmatrix}$$

expressed as $\mathbf{X}_q \mathbf{a}_p = -\mathbf{x}_{q+1}$

Signal Modeling: Padé approximation

Then get the $b_q(k)$ by the upper part

$$\begin{bmatrix} x(0) & 0 & \cdots & 0 \\ x(1) & x(0) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ x(q) & x(q-1) & \cdots & x(q-p) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ \vdots \\ a_p(p) \end{bmatrix} = \begin{bmatrix} b_q(0) \\ b_q(1) \\ \vdots \\ b_q(q) \end{bmatrix}$$

Prony's method

- Relax the constraint to match exactly the values on the interval $[\![0, p+q]\!]$
- Express another error:

$$E(z) = A_p(z)E'(z) = A_p(z)X(z) - B_q(z)$$

- Find the $a_p(k)$ coefficients that minimize the LS error

$$\varepsilon_{p,q} = \sum_{n=q+1}^{\infty} |e(n)|^2 = \sum_{n=q+1}^{\infty} \left| x(n) + \sum_{l=1}^{p} a_p(l)x(n-l) \right|^2$$

Prony's method

- Gets down to

$$\begin{bmatrix} r_x(1,1) & r_x(1,2) & \cdots & r_x(1,p) \\ r_x(2,1) & r_x(2,2) & \cdots & r_x(2,p) \\ \vdots & \vdots & \cdots & \vdots \\ r_x(p,1) & r_x(p,2) & \cdots & r_x(p,p) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} r_x(1,0) \\ r_x(2,0) \\ \vdots \\ r_x(p,0) \end{bmatrix}$$

  or $\mathbf{R}_x \mathbf{a}_p = -\mathbf{r}_x$

- Solved in the same way as for the Padé approximation

Prony's method

- Minimum error expression

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

- Get the $b_q(k)$

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

Finite records

- $x(n)$ known only on the interval $[0, N]$
- Methods to find all-pole models

Finite records: Autocorrelation method

- Autocorrelation method: Force the signal to zero outside interval
- Create new signal, $\tilde{x}(n)$ based on $x(n)$, by applying a rectangular window:

$$\tilde{x}(n) = x(n)w(n)$$

with

$$w(n) = \begin{cases} 1 & , n = 0, \dots, N \\ 0 & , \text{otherwise} \end{cases}$$

Finite records: Autocorrelation method

Use Prony's method with the autocorrelation replaced by

$$r_{\tilde{x}}(k) = \sum_{n=0}^{\infty} \tilde{x}(n)\tilde{x}^*(n-k) = \sum_{n=k}^{N} x(n)x^*(n-k), k = 0, \ldots, p$$

Finite records: Covariance method

- Covariance method: forget about the values outside interval
- Covariance normal equations:

$$\begin{bmatrix} r_x(1,1) & r_x(1,2) & \cdots & r_x(1,p) \\ r_x(2,1) & r_x(2,2) & \cdots & r_x(2,p) \\ \vdots & \vdots & \vdots & \vdots \\ r_x(p,1) & r_x(p,2) & \cdots & r_x(p,p) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ \vdots \\ a_p(3) \end{bmatrix} = - \begin{bmatrix} r_x(1,0) \\ r_x(2,0) \\ \vdots \\ r_x(p,0) \end{bmatrix}$$

Finite records: Covariance method

- Autocorrelation sequence $r_x(k, l)$ expressed as

$$r_x(k, l) = \sum_{n=p}^{N} x(n-l)x^*(n-k)$$

- Minimum error:

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

And now for some Matlab

- Some Matlab examples
- http://users.ece.gatech.edu/~mhayes/stat_dsp/matlab.html

Next week

- Optimum/optimal filters: filters that "isolate" the signal you want from another signal, e.g. noisy, containing other components. . .
- Later: Wiener Filters, Kalman Filters