

Spectral Estimation and Adaptive Signal Processing

Master - 21/4/08

Solutions:

Eq. 17

AS2

S015

Ex 4.31

1) a) [bookwork]

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

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

$$ii) E\{r_x(k) = \frac{N-k}{N} r_x(k)\} \Rightarrow E\{\hat{P}_{per}(e^{j\omega})\} = \frac{2}{\pi} P_x(e^{j\omega}) * W_B(e^{j\omega})$$

where W_B is the Bartlett window. Therefore the periodogram is a biased estimated, but since W_B converges to impulse as N goes to infinity, it is asymptotically unbiased.

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

b) [new example]

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

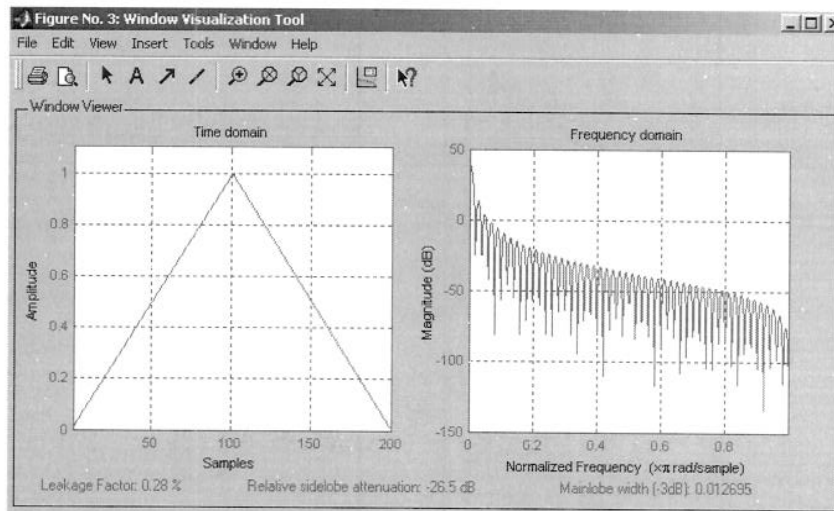
$$W_B(e^{j\omega}) = \frac{1}{N} \left[\frac{\sin(N\omega/2)}{\sin(\omega/2)} \right]^2$$

$$P_x(e^{j\omega}) = 1/2\pi A^2 [u_0(\omega - \omega_0) + u_0(\omega + \omega_0)] + \sigma_w^2$$

$$\begin{aligned} \Rightarrow E\{\hat{P}_{per}(e^{j\omega})\} &= \frac{2}{\pi} P_x(e^{j\omega}) * W_B(e^{j\omega}) = \\ &= \sigma_w^2 + 1/4A^2 [W_B(e^{j(\omega-\omega_0)}) + W_B(e^{j(\omega+\omega_0)})] \end{aligned}$$

the Bartlett (triangular) window is defined by

$$w_B(k) = \begin{cases} 1 - \frac{|k|}{N}; & |k| \leq N \\ 0; & |k| > N - 1 \end{cases}$$



The periodogram is biased and since the variance does not go to zero as $N \rightarrow \infty$ it is not a consistent estimate of the power spectrum. Averaging of periodogram reduces the variance but may also decrease frequency resolution.

c) **[new example]**

To get the maximum resolution from $N = 10,000$ we want to compute the periodogram of $x(n)$, by segmenting $x(n)$ into subsequences, however this reduces the resolution. The question, therefore, is how to compute the periodogram of $x(n)$ using 1024-point DFT's. Recalling how the FFT works, note that

$$X(e^{j\omega}) = \sum_{n=0}^{9999} x(n)e^{-jn\omega}$$

therefore the procedure is to pad $x(n)$ with zeros to form a sequence of length $N = 10240$, and then decimate $x(n)$ into 10 sequences $x_l(n)$ of length $M = 1024$,

$$x_l(n) = x(10n + l), \quad n = 0, \dots, 1023$$

Next, the 1024-point DFT's of these sequences, $X_l(k)$ are computed, and combined as follows

$$X(k) = \sum_{l=0}^9 e^{-jl\frac{2\pi k}{10240}} X_l(k), \quad k = 0, 1, \dots, 10239$$

Finally, squaring the magnitude of $X(k)$ and dividing by $N = 10240$, we have the periodogram with resolution $\Delta\omega = 0.89\frac{2\pi}{10000}$.

2) a) [bookwork]

They are not consistent estimators, they give poor resolution and do not work well on short data records. They are also limited in their ability to resolve closely spaced narrowband processes when the number of data samples is limited. An advantage: they do not make any assumptions or place any constraints on the process and can be used for any type of processes.

b) [bookwork and new example]

i)

The ARMA power spectrum estimate is given by

$$\hat{P}_x(e^{j\omega}) = \frac{\left| \sum_{k=0}^q \hat{b}_q(k) e^{-jk\omega} \right|^2}{\left| 1 + \sum_{k=1}^p \hat{a}_p(k) e^{-jk\omega} \right|^2}$$

For the AR spectrum, we have only the denominator of the above expression, whereas for the MA spectrum the denominator of the above expression is constant.

The AR spectrum is more suitable for modelling peaks in spectrum (has poles in transfer function). The MA spectrum is more suitable for modelling flat spectra and spectra with zeros in transfer function.

ii) This is an $AR(4)$ spectrum, since the spectrum has two peaks and no zeros. These peaks are generated by 2 conjugate complex pairs of poles. This cannot be the frequency response of an MA spectrum since there are no finite zeros in the spectrum. This is also visible from the above expression for the ARMA spectrum. The positions of the poles can be deduced based on the frequencies of the peaks of the AR power spectrum (angle), whereas the magnitudes of the peaks can be used to deduce the distance of the poles from the origin in the z -plane.

b) [new example]

Finding the peaks of the MUSIC frequency estimation function is equivalent to finding the minima of

$$\sum_{i=p+1}^M |\mathbf{e}^H \mathbf{v}_i|^2$$

Since the eigenvectors \mathbf{v}_i are orthogonal, if we assume that they are normalised, then the identity matrix may be expanded in terms of these eigenvectors as follows

$$\mathbf{I} = \sum_{i=1}^M \mathbf{v}_i \mathbf{v}_i^H$$

Multiplying on the left by \mathbf{e}^H and on the right by \mathbf{e} we have

$$\mathbf{e}^H \mathbf{e} = \sum_{i=1}^M (\mathbf{e}^H \mathbf{v}_i) (\mathbf{v}_i^H \mathbf{e}) = \sum_{i=1}^M |\mathbf{e}^H \mathbf{v}_i|^2$$

Since $\mathbf{e}^H \mathbf{e} = M$, we have

$$M = \sum_{i=1}^p |\mathbf{e}^H \mathbf{v}_i|^2 + \sum_{i=p+1}^M |\mathbf{e}^H \mathbf{v}_i|^2$$

or

$$\sum_{i=p+1}^M |\mathbf{e}^H \mathbf{v}_i|^2 = M - \sum_{i=1}^p |\mathbf{e}^H \mathbf{v}_i|^2$$

Thus, minimising the left-hand side is equivalent to maximising the sum of the right as was to be shown.

3) a) [bookwork]

Start from cost function

$$J(k) = \frac{1}{2}e^2(k) \quad (1)$$

we have

$$e(k) = d(k) - \Phi(\mathbf{x}^T(k)\mathbf{w}(k)) \quad (2)$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \nabla_{\mathbf{w}(k)} J(k) \quad (3)$$

Gradient $\nabla_{\mathbf{w}(k)} J(k)$ can be calculated as

$$\frac{\partial J(k)}{\partial \mathbf{w}(k)} = e(k) \frac{\partial e(k)}{\partial \mathbf{w}(k)} = -e(k) \Phi'(\mathbf{x}^T(k)\mathbf{w}(k)) \mathbf{x}(k) \quad (4)$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta \Phi'(\mathbf{x}^T(k)\mathbf{w}(k)) e(k) \mathbf{x}(k) \quad (5)$$

⇒ This is the weight update equation for a direct gradient algorithm for a nonlinear FIR filter, called the **nonlinear gradient descent (NGD)**.

The middle region of the nonlinearity is approximately linear and the derived algorithm is similar to LMS. When the filter operates in the tails of the nonlinearity, the gradient is close to zero and the filter is not updated. b) [new example]

[new example]

Differentiation performed by $u(k) = x(k) - x(k-1)$ reduces the dynamical range, since it only operates on two consecutive samples. However there is no guarantee that this will suit the range of the nonlinearity within the filter. Also if the nonlinearity has only a positive range (logistic function), this transformation may not be suitable, since it can produce transformed inputs with both positive and negative values. One alternative transformation is the application of the logarithm, however, this transformation suffers from similar problems as in the case of differentiation.

c) [bookwork and worked example]

$$E(k) = \frac{1}{2}e^2(k), \quad e(k) = d(k) - \Phi(\mathbf{x}^T(k)\mathbf{w}(k))$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta \mathbf{x}(k) \Phi'(\mathbf{x}^T(k)\mathbf{w}(k)) e(k)$$

$$\Phi(\mathbf{x}^T(k)\mathbf{w}(k)) = \lambda(k) \bar{\Phi}(\mathbf{x}^T(k)\mathbf{w}(k))$$

$$\lambda(k+1) = \lambda(k) - \rho \nabla_{\lambda} E$$

$$\frac{\partial E(k)}{\partial \lambda(k)} = \frac{\partial 1/2e^2(k)}{\partial e(k)} \frac{\partial e(k)}{\partial y(k)} \frac{\partial y(k)}{\partial \lambda(k)}$$

$$\lambda(k+1) = \lambda(k) + \rho e(k) \bar{\Phi}(\mathbf{x}^T(k)\mathbf{w}(k))$$

d) [new example]

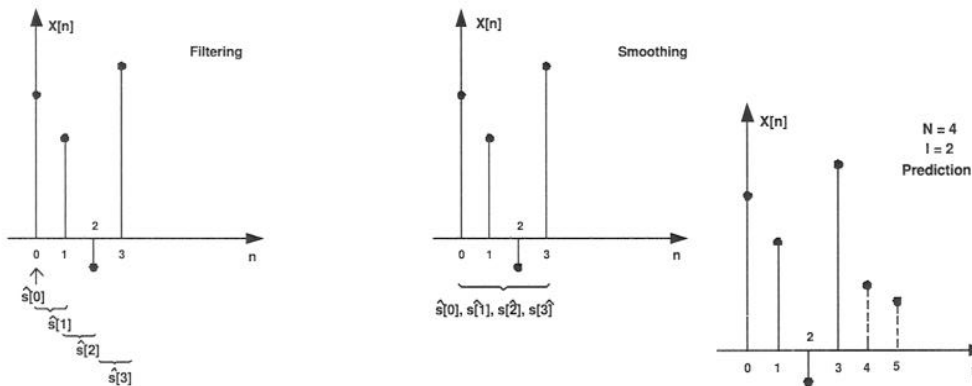
The bilinear model is a feedback model and the FIR structure is not suitable for its realisation. A better way would be to use a nonlinear feedback structure, e.g. a recurrent perceptron.

4) a) [bookwork and worked example]

- **Parametric** modelling assumes a fixed structure for the model. The model identification problem then simplifies to estimating a finite set of parameters of this fixed model. An example of this technique is the broad class of ARIMA/NARMA models.
- **Nonparametric** modelling seeks a particular model structure from the input data. The actual model is not known beforehand. We look for a model in the form of $y(k) = f(x(k))$ without knowing the function $f(\cdot)$.
- **Semiparametric** modelling is the combination of the above. Part of the model structure is completely specified and known beforehand, whereas the other part of the model is either not known or loosely specified.

b) [bookwork and worked example]

Filtering: $\theta = s[n]$ to be estimated based on $x[m] = s[m] + w[m] \quad m = 0, 1, \dots, n$.



Filters signal from noise, based on current and past data, i.e. casual filtering.

Smoothing: $\theta = s[n]$ to be estimated based on entire dataset $\{x[0], x[1], \dots, x[N-1]\}$. Requires all data to be collected.

Prediction: $\theta = x[N-1+l]$ for l a positive integer based on $\{x[0], x[1], \dots, x[N-1]\}$ "1-step" forward prediction.

- it is smoothing, $y(n)$ is estimated based on previous and future values of y .
- filtering

c) [bookwork and intuitive reasoning]

i) Least squares techniques are based on the **exact minimisation** of the sum of square errors. The cost function for the Least Squares algorithm is given by

$$J(n) = \sum_{i=1}^n e^2(i) = \sum_{k=1}^n [d(k) - \mathbf{x}^T(k)\mathbf{w}(k)]$$

This is a **totally deterministic** cost function.

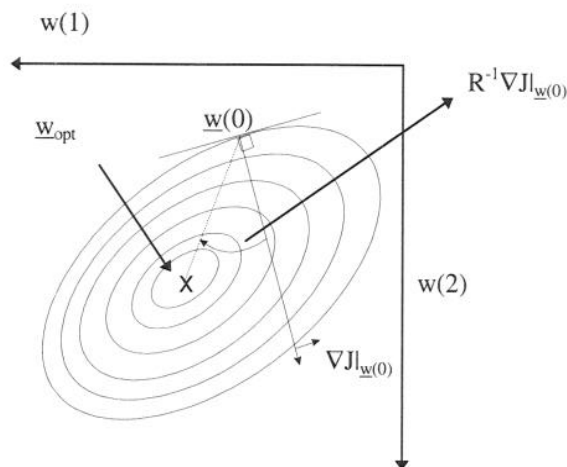
ii) We desire

$$\mathbf{w}(n+1) = f[\mathbf{w}(n), \mathbf{x}(n+1)]$$

To achieve this, we partition the data matrix

$$\mathbf{X}(n+1) = \begin{pmatrix} \mathbf{x}^T(n+1) \\ \dots \\ \mathbf{X}(n) \end{pmatrix}$$

iii) The LMS is based on a stochastic cost function. The role of \mathbf{R}^{-1} in RLS is to rotate the direction of descent of the LMS algorithm towards the minimum of the cost function independent of the nature, or colouration, of the input. The



operation $\mathbf{R}^{-1}\mathbf{x}[n]$ is a **pre-whitening operation**.

The convergence of the RLS in a high SNR environment ($> 10dB$) is of an order $\mathcal{O}(2p)$, whereas for LMS $\mathcal{O}(10p)$.

The misadjustment performance of RLS is essentially zero because it is deterministic and matches the data where $\lambda = 1$.

iv) A sample correlation matrix \mathbf{R} can be written as

$$\mathbf{R}(n+1) = \mathbf{X}^T(n+1)\mathbf{X}(n+1) = \mathbf{R}(n) + \mathbf{x}(n+1)\mathbf{x}^T(n+1)$$

We wish to solve

$$\mathbf{w}(n+1) = \mathbf{R}^{-1}(n+1)\mathbf{p}(n+1)$$

where

$$\mathbf{p}(n+1) = \sum_{k=1}^{n+1} \mathbf{x}(k)d(k)$$

v) The basic cost function for the RLS algorithm assumes a statistically stationary environment. There, all the errors are weighted equally. In order to deal with a nonstationary environment, modify the LS error criterion

$$J(n) = \sum_{k=1}^n \lambda^{n-k} e^2(k)$$

This way old information is forgotten. The forgetting factor $\lambda \in (0, 1]$, but typically > 0.95 . The forgetting factor introduces an effective window length of $\frac{1}{1-\lambda}$.

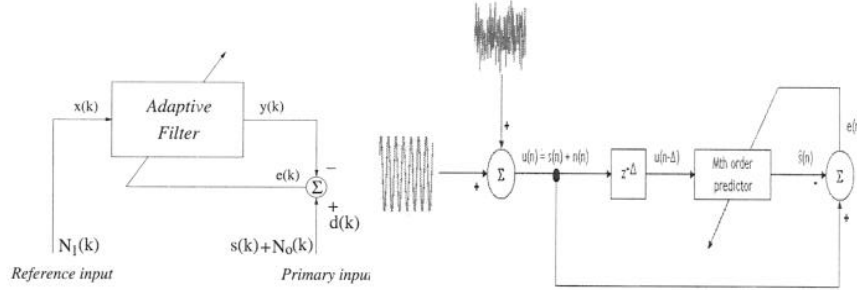


Figure 1: Left: Adaptive noise cancellation configuration. Right: Adaptive Line Enhancement configuration

5) a) [bookwork and intuitive reasoning]

Both configurations are designed to suppress noise. The ANC configuration achieves this by a reference noise input which is correlated with the noise within the signal. The larger the degree of correlation, the more successful noise cancellation. The ALE configuration does not have a reference input and relies instead on the different correlation lags of signal and noise.

b) [bookwork]

The NLMS algorithm uses an instantaneous estimate of the input correlation matrix to “normalise” the error surface, that is, it aims to produce a modified surface for which the contours are concentric circles. This contributes to the speed of convergence and stability of the NLMS algorithm. Parameter β is the learning rate as is usually set to unity, whereas ε is a regularisation factor which prevents the filter from becoming unstable for very small values of input.

c) [new example]

The a posteriori error update is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{1}{\frac{1}{\mu} + \|\mathbf{x}(k)\|_2^2} e(k) \mathbf{x}(k)$$

Compare with NLMS, that is

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\beta e(k) \mathbf{x}(k)}{\varepsilon + \|\mathbf{x}(k)\|_2^2}$$

The equivalence is achieved for $\beta = 1$ and $\varepsilon = 1/\mu$. This also involves a compromise, since for fast adaptation parameter ε should be small, however for stability of NLMS, $0 < \mu < 2$. For typical values of $\mu = 0.1, 0.2, etc$ the algorithm becomes over-regularised, i.e. the effect of the normalisation of the error surface is reduced.

d)i) and ii) [**new example**]

By minimising the L successive a posteriori errors, we ensure fast convergence and small error. This cannot be achieved by NLMS only, since NLMS has no memory. The AP algorithm has memory and this way it is better suited for nonstationary environments (albeit at a higher computational complexity). Similar to the derivation of NLMS, it follows that

$$\mathbf{e}_p = \mathbf{d} - \begin{bmatrix} \mathbf{x}^T(k) \\ \vdots \\ \mathbf{x}^T(k-L) \end{bmatrix} \mathbf{w}(k+1) = \mathbf{0}$$

that is

$$\begin{aligned} \mathbf{X}^T \mathbf{w}(k+1) &= \mathbf{d} \\ \mathbf{X}^T (\mathbf{w}(k) + \Delta \mathbf{w}(k)) &= \mathbf{d} \\ \Rightarrow \mathbf{X}^T \Delta \mathbf{w}(k) &= \mathbf{d} - \mathbf{X}^T \mathbf{w}(k) = \mathbf{e}_p \end{aligned}$$

This is an underdetermined set of equations and we have to use the pseudoinverse

$$\Delta \mathbf{w}(k) = (\mathbf{X}^T)^{\#-1} \mathbf{e}$$

The weight update is therefore given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{e}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(k) \\ \vdots \\ \mathbf{x}^T(k-L) \end{bmatrix}$ and $\mathbf{e} = \begin{bmatrix} e(k) \\ \vdots \\ e(k-L) \end{bmatrix}$.