Advanced Digital Signal Processing Coursework

Prof. Danilo P. Mandic
TA: Sithan Kanna

January 25, 2016

Contents

Guidelines

The coursework comprises five assignments, whose individual scores yield 90% of the final mark. The remaining 10% accounts for presentation and organisation. Students are allowed to discuss the coursework but must code their own MATLAB scripts, produce their own figures and tables, and provide their own discussion of the coursework assignments.

General directions and notation:

- The simulations should be coded in MATLAB, a *de facto* standard in the implementation and validation of signal processing algorithms.
- The report should be clear, well-presented, and include the answers to the assignments in this handout with appropriate numbering. Students are encouraged to submit through Blackboard **(in PDF format only)**, although a hardcopy submission will also be accepted at the undergraduate office.
- The report should document the results and the analysis in the assignments, in the form of figures (plots) and tables, and not by listing MATLAB code as a proof of implementation.
- We adopt the following notation: boldface lowercase letters (e.g. $\mathbf{x}$) for vectors, lowercase letters with a (time) argument ($x[n]$) for scalar realisations of random variables and elements of a vector, and uppercase letters (X) for random variables. Column vectors will be assumed unless otherwise stated, that is, $\mathbf{x} \in \mathbf{R}^{N\times 1}$.
- The typewriter font, e.g. `mean`, is used for MATLAB functions.

Presentation:

- The length limit for the report (all parts) is 42 pages. This corresponds to eight pages per assignment in addition to one page for front cover and one page for the table of contents, however, there are no page restrictions per assignment but only for the full-report (42 pages).
- The final mark also considers the presentation of the report, this includes: legible and correct figures, tables, and captions, appropriate titles, table of contents, and front cover with student information.
- The figures and code snippets (only if necessary) included in the report must be carefully chosen, for clarity and to meet the page limit.
- Do not insert unnecessary MATLAB code or the assignment questions in the report.
- For figures, (i) decide which type of plot is the most appropriate for each signal (e.g. solid line, non-connected points, stems), (ii) export figures in a correct format: without grey borders and with legible captions and lines, and (iii) avoid the use of screentshots when providing plots and data, use figures and tables instead.
- Avoid terms like *good estimate, is (very) close, somewhat similar*, etc - use formal language and quantify your statements (e.g. in dB, seconds, samples, etc).
- Note that you should submit two files to Blackboard: the report in PDF format and all the MATLAB code files compressed in ZIP/RAR format. Name the MATLAB script files according to the part they correspond to (e.g. ASP_Part_X-Y-Z.m).

Honour code:

Students are strictly required to adhere to the College policies on students rights and responsibilities.The College has zero tolerance to plagiarism. Any suspected plagiarism or cheating (or prohibited collaboration on the coursework, see above) will lead to a formal academic dishonesty investigation. Being found responsible for an academic dishonesty violation results in a discipline file for the student and penalties, ranging from severe reduction in marks to expulsion from College.

1 Random signals and stochastic processes

Aims:

- To become acquainted with the generation of random signals in MATLAB.
- To investigate the effect of a linear system upon a random signal.
- To calculate and understand auto- and cross-correlation functions.

In most real-world signal processing applications the observed signals cannot be described by an analytical expression, however, by the Wold decomposition theorem, a stationary signal can be written as a sum of a deterministic signal and a stochastic signal. Our focus is on the statistical properties of such signals and their study paves the way for the design of estimation algorithms.

1.1 Statistical estimation

Using the MATLAB command `rand`, generate a 1000-sample vector $\mathbf{x} = [x[1], x[2], \ldots, x[1000]]^T$ where each sample $x[n]$ is a realisation of a uniform random variable $X \sim \mathcal{U}(0, 1)$ at time instant n. Plot the result and observe that despite its stochastic nature, $\mathbf{x}$ exhibits a degree of uniformity due to its time-invariant statistical properties, since the different samples $x[n], x[m]$ have been drawn from the same distribution. Such signals are referred to as **statistically stationary**. The vector $\mathbf{x}$ can be considered as a 1000-sample realisation of a stationary stochastic process X_n, whereby $X_n \sim \mathcal{U}(0, 1), \forall n$.

1. Calculate the expected value of X, denoted by $m = \mathbb{E}\{X\}$, also known as the *theoretical mean*. Using your 1000-sample realisation $\mathbf{x}$, also compute the *sample mean* using the MATLAB function `mean`, calculated as [5]

$$\widehat{m} = \frac{1}{N}\sum_{n=1}^{N} x[n],$$

   where the circumflex denotes an estimate. Comment on the accuracy of the sample mean as an estimator.

2. Repeat the analysis for the standard deviation: calculate the theoretical value $\sigma = \sqrt{\mathbb{E}\{X - \mathbb{E}\{X\}\}^2}$ and also its sample estimate from data $\mathbf{x}$ using the MATLAB function `std` which computes the *sample standard deviation* as[1] [5]

$$\widehat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x[n] - \hat{m})^2}. \tag{1}$$

   Comment on the accuracy of $\widehat{\sigma}$.

3. The *bias* of the sample mean estimation is given by $B = \mathbb{E}\{X\} - \widehat{m}$. Generate an ensemble of ten 1000-sample realisations of X, denoted by $\mathbf{x}_{1:10}$, and calculate the sample means $\widehat{m}_{1:10}$ and standard deviations $\widehat{\sigma}_{1:10}$ for each realisation. Plot these estimates of mean and standard deviation and comment on their bias, by showing how they cluster about their theoretical values. [5]
   Note: In general, when plotting quantities that are not indexed by time stamps (as in this case, where the index is the 'realisation') there is no reason to connect the plotted points.

4. The mean and standard deviation describe second order statistical properties of a random variable, however, to obtain a complete statistical description it is necessary to examine the probability density function (pdf) from which the samples are drawn. Approximate the pdf of X by showing in the same plot the histogram of $\mathbf{x}$ (see `hist`), normalised by the number of samples considered, and the theoretical pdf. Comment upon the result, in particular, on whether the estimate appears to converge as the number of generated samples increases, and how the number of histogram bins considered affects the analysis. [5]

   Note: As mentioned above, the theoretical pdf of X is $\mathcal{U}(0, 1)$.

5. Repeat Part 1–Part 4 using the MATLAB function `randn` to generate zero-mean, unit standard deviation, Gaussian random variables. [20]

[1]Note that the use of $(N-1)$, instead of N, in Eq. (1) is known as *Bessel's correction*, which aims to remove the bias in the estimation of population variance, and some (but not all) bias in the estimation of the population standard deviation. This way, Eq. (1) gives an *unbiased* estimator of σ.

1.2 Stochastic processes

In real-world applications, time-varying quantities are usually modelled by a stochastic process, that is, an ordered collection of random variables. For **ergodic** processes, the theoretical mean can be approximated by time averages, however, for **non-ergodic** processes time averages do not necessarily match averages in the probability space and therefore the theoretical statistics are not always well approximated using observed samples.

To illustrate this concept, consider the **deterministic** sinusoidal signal $x[n] = \sin(2\pi 5 \times 10^{-3} n)$ corrupted by independent and identically distributed (i.i.d.) Gaussian noise $\eta[n] \sim \mathcal{N}(0,1)$, to give $y[n] = x[n] + \eta[n]$. By averaging multiple realisations of the process $\mathbf{y} = [y[1], y[2], \ldots, y[N]]^T$, we aim to obtain reduced-noise estimates of the process $\mathbf{x}$, in terms of the Signal-to-Noise (SNR) ratio. If M independent realisations of the process $\mathbf{y}$, denoted by $\mathbf{y}_{1:M}$, are considered to compute an ensemble estimate, the variance of such an estimate is given by

$$\sigma_M^2 = \mathbb{E}\left\{\left(\mathbb{E}\{\mathbf{y}\} - \frac{1}{M}\sum_{i=1}^{M}\mathbf{y}_i\right)^2\right\} = \mathbb{E}\left\{\left(\frac{1}{M}\sum_{i=1}^{M}\boldsymbol{\eta}_i\right)^2\right\} \tag{2}$$

$$= \frac{1}{M^2}\mathbb{E}\left\{\left(\sum_{i=1}^{M}\sum_{j=1}^{M}\boldsymbol{\eta}_i^T\boldsymbol{\eta}_j\right)\right\} = \frac{1}{M^2}\left(\sum_{i=1}^{M}\sum_{j=1}^{M}\mathbb{E}\left\{\boldsymbol{\eta}_i^T\boldsymbol{\eta}_j\right\}\right), \tag{3}$$

where every noise sequence $\boldsymbol{\eta}_j$ comprises realisations of zero-mean and **uncorrelated** random variables $\eta[n]$. We know that $\mathbb{E}\left\{\boldsymbol{\eta}_i^T\boldsymbol{\eta}_j\right\} = \sigma_{\boldsymbol{\eta}}^2$ iff $i = j$, and zero otherwise, hence

$$\sigma_M^2 = \frac{1}{M^2}\left(M\sigma_{\boldsymbol{\eta}}^2\right) = \frac{\sigma_{\boldsymbol{\eta}}^2}{M}. \tag{4}$$

Therefore, the SNR of an M-member ensemble estimate increases linearly with the number of members of the ensemble

$$SNR = \frac{\sigma_{\mathbf{y}}^2}{\sigma_M^2} = \frac{\sigma_{\mathbf{y}}^2}{\sigma_{\boldsymbol{\eta}}^2}M, \quad \text{and in dB: } SNR_{dB} = \log_{10}\left(\frac{\sigma_{\mathbf{y}}^2}{\sigma_{\boldsymbol{\eta}}^2}M\right)\ [dB]. \tag{5}$$

Figure 1 shows a realisation of $\mathbf{y}$, together with ensemble averages for $M = 10, 50, 200, 1000$ and the original deterministic signal $\mathbf{x}$. Additionally, the bottom plot shows the SNR computed from the ensemble averages and its theoretical value in Eq. (5). Observe that for nonstationary signals, time-average will not provide meaningful approximations of the process statistics (e.g. sample mean).

We now study three stochastic processes generated by the following MATLAB codes, which give an ensemble of M realisations of N samples for each stochastic process.

a)
```
function v=rp1(M,N);
a=0.02;
b=5;
Mc=ones(M,1)*b*sin((1:N)*pi/N);
Ac=a*ones(M,1)*[1:N];
v=(rand(M,N)-0.5).*Mc+Ac;
```

b)
```
function v=rp2(M,N)
Ar=rand(M,1)*ones(1,N);
Mr=rand(M,1)*ones(1,N);
v=(rand(M,N)-0.5).*Mr+Ar;
```

c)
```
function v=rp3(M,N)
a=0.5;
m=3;
v=(rand(M,N)-0.5)*m + a;
```

Run the above MATLAB codes and explain the differences between the time averages and ensemble averages, together with the stationarity and ergodicity of the process generated by the following steps:

1. Compute the ensemble mean and standard deviation for each process and plot them as a function of time. For all the above random processes, use $M = 100$ members of the ensemble, each of length $N = 100$. Comment on the stationarity of each process. [10]
2. Generate $M = 4$ realisations of length $N = 1000$ for each process, and calculate the mean and standard deviation for each realisation. Comment on the ergodicity of each process. [10]
3. Write a mathematical description of each of the three stochastic processes. Calculate the theoretical mean and variance for each case and compare the theoretical results with those obtained by sample averaging. [10]

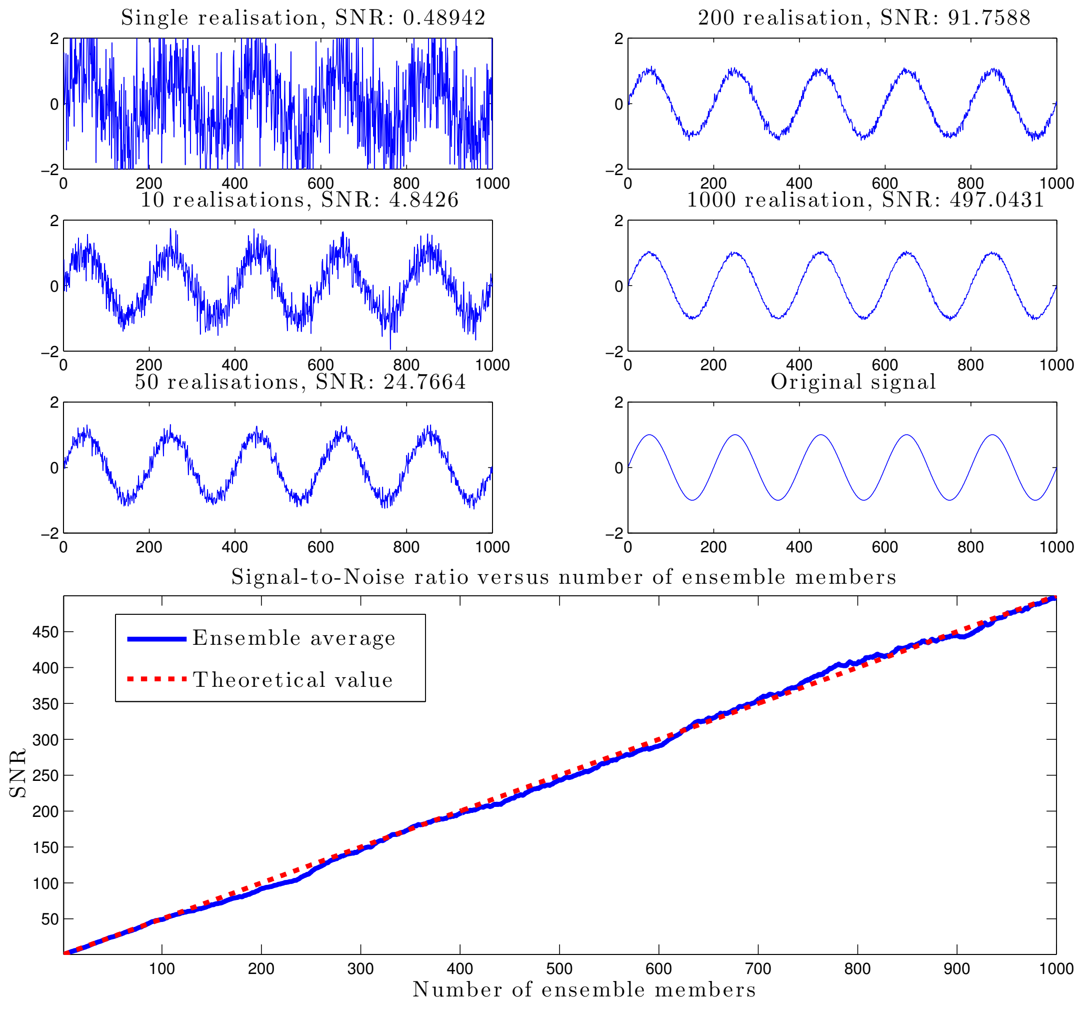

Figure 1: Ensemble estimates of a deterministic sequence corrupted by zero-mean uncorrelated white noise.

1.3 Estimation of probability distributions

Stochastic processes can be uniquely represented by their probability density functions (pdf), since the pdf contains all the statistical information about a random variable. When the process at hand is stationary, the statistical properties are time-invariant, and the process is uniquely defined by a time-invariant pdf. Furthermore, for ergodic processes such distributions can be estimated via time averages.

Design and implement a pdf estimator and test it on the three stochastic processes studied in Part 1.2.

1. Write an m-file named `pdf`, which gives an estimate of the pdf of a collection of samples based on the MATLAB function `hist`. Test your code for the case of a stationary process with a Gaussian pdf, `v=randn(1,N)`. The data length `N` should be at least 100. [10]

2. For those processes in Part 1.2 (a,b,c) that are stationary and ergodic, run your MATLAB code to approximate the pdf for $N \in \{100, 1000, 10000\}$. Compare the estimated densities with the theoretical pdf using the MATLAB `subplot` function, and comment on the results as data length N increases. [10]

3. Is it possible to estimate the pdf of a nonstationary process with your function `pdf`? Comment on the difficulties encountered for the case when the mean of a 1000-sample-long signal changes from 0 to 1, after sample point $N = 500$. Explain how you would compute the correct pdf in this case? [10]

2 Linear stochastic modelling

Aims:

- To introduce practical estimators of the autocorrelation and cross-correlation functions.
- To give insight into linear stochastic modelling and its stability.
- To understand linear prediction models and optimal model order selection.

The autocorrelation function (ACF) of a stochastic process X_n is given by

$$R_X(n,s) = \mathbb{E}\{X_n X_s\}. \tag{6}$$

By assuming ergodicity, the statistical expectation operator can be approximated by a time average over the samples $x[0], x[1], \ldots, x[N-1]$, that is

$$R_X(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} x[n]x[n+\tau], \qquad \tau \in \mathbb{Z}. \tag{7}$$

However, as in real-world applications only a finite number of samples is available, we consider the so-called **unbiased** estimate of the autocorrelation function given by

$$\widehat{R}_X(\tau) = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} x[n]x[n+\tau], \qquad \tau = -N+1, \ldots, N-1. \tag{8}$$

Note: For an N-sample realisation $\mathbf{x} = [x[0], \ldots, x[N-1]]^T$, the estimated ACF has $(2N-1)$ samples.

2.1 ACF of uncorrelated sequences

For independent stochastic processes, the autocorrelation function has the form of a discrete Dirac function, since knowledge of one sample has no bearing on explaining any other samples. You will now verify this property when the ACF estimates are calculated based on Eq. (8).

1. Employ the function `xcorr(x,'unbiased')` to calculate the unbiased estimate of the ACF for a 1000-sample realisation[2] of WGN denoted by $\mathbf{x}$. Display for $\tau \in [-999 : 999]$ using the function `axis`, explain the result, and comment on the symmetry of the ACF. [5]
2. Use the `zoom` command to focus onto the region $|\tau| < 50$ and compare with the shape of the ACF for large τ. Explain the differences. [5]
3. Explain the effects of a large autocorrelation lag $|\tau|$ on the accuracy of ACF estimates according to Equation (8). Elaborate on whether the ACF estimates for large $|\tau|$ are statistically reliable, and suggest an empirical bound on $|\tau|$. [5]

2.2 ACF of correlated sequences

1. Generate a 1000-sample WGN vector $\mathbf{x}$ and filter it by a moving average (MA) filter with unit coefficients of order 9 by executing the command `y=filter(ones(9,1),[1],x)`, and plot the ACF estimate for $\mathbf{y}$ using the function `stem` for lags between -20 and 20. Comment upon the shape of the ACF and its relation to the MA coefficients (impulse response) of the considered filter. Additionally, comment on the effect of the filter order on the ACF; is it possible to calculate the (local) sample mean in this way? [10]
2. The vector $\mathbf{y}$ is a realisation of the stochastic process Y_n, which is a filtered version of the process X_n. The ACF of Y_n, denoted by R_Y, is given by [5]

$$R_Y(\tau) = R_X(\tau) * R_h(\tau),$$

where R_X is the autocorrelation of the input, R_h is the autocorrelation of the impulse response, and the symbol $*$ denotes the convolution operator. If X_n is an uncorrelated process, what does R_Y represent?

[2]White Gaussian noise, a zero-mean and unit-variance process generated using the MATLAB function `randn`.

2.3 Cross-correlation function

As an extension to the ACF given in Eq. (6), the cross-correlation function (CCF) is defined as the expectation of a product of delayed samples of two different stochastic processes, that is

$$R_{XY}(n,s) = \mathbb{E}\{X_n Y_s\}. \tag{9}$$

For ergodic data, as shown in Eq. (8) for the autocorrelation function, an unbiased estimate of the CCF is given by

$$\hat{R}_{XY}(\tau) = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} x[n]y[n+\tau], \qquad \tau = -N+1, ..., N-1$$

which again has the length of $2N-1$.

1. Estimate the CCF for the sequences $\mathbf{x}$ and $\mathbf{y}$ generated in Part 2.2 using the command `xcorr(x,y,'unbiased')`. Plot the result using `stem` for autocorrelation lags in the region [-20, 20] and comment upon the shape of the cross-correlation function. [5]

   The cross-correlation function R_{XY} between the input and the output of a filter is given by

   $$R_{XY}(\tau) = h(\tau) * R_X(\tau),$$

   where $h(\tau)$ is the impulse response of the filter. If X_t is an uncorrelated stochastic process, what shape will R_{XY} take?

2. Suggest how this result can be used for system identification. What is the effect of the order of the filter that generated $\mathbf{y}$ on the shape of the cross-correlation function? [5]

2.4 Autoregressive modelling

1. Generate 100 samples of the uniformly distributed random variables $a_1 \in [-2.5, 2.5]$ and $a_2 \in [-1.5, 1.5]$. Use the pairs (a_1, a_2) as the coefficients of the AR(2) process of length 1000, given by [10]

   $$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n], \qquad w[n] \sim \mathcal{N}(0,1). \tag{10}$$

   If the signal $x[n]$ converges, plot the symbol $*$ at the position (a_1, a_2) on the canvas[3]. Comment on the values of the pairs (a_1, a_2) which preserve the stability of this process. Give a theoretical justification for the shape of the resulting convergence region?

2. Load the real world sunspot time series[4] and plot its ACFs for data lengths $N = 5,\ 20,\ 250$. Comment on any differences in the shape of the ACFs. Explain the effects of the mean of the series. Verify this by computing the ACF of a zero-mean version of the sunspot series and comment on the differences from the original (non-centred) signal. [10]

3. Use the Yule-Walker equations to calculate the partial correlation functions up until the model order $p = 10$ and comment on the most likely model order of the sunspot time series. Repeat the procedure for the sunspot series standardised to zero mean and unit variance. Explain the differences. [5]

4. Now use the minimum description length (MLD) and the Akaike information criterion (AIC) to determine the correct model order for the standardised data. These criteria are defined as [5]

   $$\text{MDL} = \log E_p + \frac{p \log N}{N} \tag{11}$$
   $$\text{AIC} = \log E_p + \frac{2p}{N}, \tag{12}$$

   where

   - E_p is the loss function (typically the cumulative squared error) for the model with p parameters,
   - p is the number of estimated parameters, that is, the model order,
   - N is the number of estimated data points.

5. An AR model is used to predict the sunspot time series m steps ahead, in the form [10]

$$\hat{x}[n+m] = a_1 x[n-1] + \cdots + a_p x[n-p], \tag{13}$$

for the prediction horizons $m = 1, 2, 5, 10$. Use the AR(1), AR(2) and AR(10) models of the sunspots calculated in Parts 4 and 5 of this section, and compare the actual with the predicted values. Give your own interpretation and justification for the results, and explain how a model order larger than the optimal one (overmodelling) affects the statistical properties of this predictor. Focus on the trade-off between the model error (interpolation) and prediction error (extrapolation).

2.5 Real world signals: ECG from iAmp experiment

***See also Problem 3.4 in your Problem and Answer Sheets.**
Before attempting this section, you should have MATLAB data files corresponding to the RR interval (RRI) data from three trials – unconstrained breathing, constrained breathing at 50 beats per minute and constrained breathing at 15 beats per minute. (*Further details on splitting a single ECG recording into three segments and converting them to RRI data are given in your experiment handouts*)

Heart rate probability density estimate (PDE).
Using the RRI signal $rr[n]$ from Trial 1 (unconstrained breathing), obtain the heart rate $h[n]$ from

$$h[n] = \frac{60}{rr[n]} \tag{14}$$

To obtain a smoother estimate of the heart rate, use the following method to average every 10 samples of the heart rate:

$$\widehat{h}[1] = \frac{1}{10}\sum_{i=1}^{10} \alpha h[i], \;\; \widehat{h}[2] = \frac{1}{10}\sum_{i=11}^{20} \alpha h[i], \;\; \ldots \tag{15}$$

where α is a scalar.

a) Plot the probability density estimate (PDE) of the original heart rates $h[n]$ and the averaged heart rates $\widehat{h}[n]$ for $\alpha = 1$ and $\alpha = 0.6$. [5]

b) Comment on the shape of the PDE of the averaged heart rates compared to the original heart rates. How does the constant α affect the PDE? [5]

AR modelling of heart rate.

c) Find the autocorrelation sequence for the RRI data for the three trials. From the shape of the autocorrelation sequence, can you infer whether the RRI data is an AR or an MA process? (To ensure that your data is zero-mean, use the MATLAB command `detrend`). [5]

d) Using the tools and analyses for the AR modelling of the sunspot series in Section 2.4, argue whether the RRI signal for the three trials can be modelled by an AR(p) process. If so, what is the optimal AR model order? [5]

[3]**Hint:** When plotting the symbol $*$ on the positions $\{(a_1^i, a_2^i), i = 1, \ldots, n\}$, do not connect the points using lines. Define the vectors $a1 = [a_1^1, a_1^2, a_1^3, \ldots, a_1^n]$ and $a2 = [a_2^1, a_2^2, a_2^3, \ldots, a_2^n]$, and execute the command `plot(a1,a2,'*')`.

[4]Use `load sunspot.dat` in your MATLAB distribution. The data are contained in the second column of the variable `sunspot`.

3 Spectral estimation and modelling

Aims:

- To introduce the Power Spectral Density (PSD) of a random signal.
- To study classical methods for PSD estimation.
- To investigate model-based methods for PSD estimation.
- To gain experience with practical applications of spectrum estimation.

The power spectral density (PSD) of an ergodic stochastic process X_n refers to the distribution of the power of X_n across the frequency components of such process. More formally, the PSD of a stochastic process X_n is given by the absolute value of the Fourier transform of its autocorrelation function, that is

$$P_X(f) = \left| \sum_{\tau=-\infty}^{\infty} R_X(\tau) e^{-\jmath 2\pi f \tau} \right|, \qquad f \in [0,1] \tag{16}$$

where $R_X(\cdot)$ is the ACF of the stochastic process X_n, and f is the normalised frequency. This relation is also known as the Wiener–Khintchine theorem.

The calculation of $P_X(f)$ according to (16) requires complete knowledge of the autocorrelation function of X_n, however, in real world applications this quantity is approximated based on the available samples, hence, the PSD can only be estimated.

A method to estimate the PSD based on the fast Fourier transform (FFT) is the so-called *periodogram*, and is defined by

$$\widehat{P}_X(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-\jmath 2\pi f \frac{n}{N}} \right|^2. \tag{17}$$

The PSD is estimated at N discrete frequencies, which in terms of normalised frequencies, corresponds to $f = 0, 1/N, \ldots, (N-1)/N$. For convenience, the sampling interval T is implicitly assumed to be unity.

1. Write a MATLAB script called pgm which calculates the periodogram according to Eq. (17). The input and output of the MATLAB function should be, respectively, a sequence $\mathbf{x} = [x[1], x[2], \ldots, x[N]]$ and its periodogram $\widehat{P}_X(f)$, both of the same length N. Test your code with an N-sample realisation of WGN $\mathbf{x}$, for $N = 128, 256$, and 512. Inspect and comment upon the results. [10]

3.1 Averaged periodogram estimates

The theoretical autocorrelation function for a stochastic ergodic Gaussian process X_n is given by $\sigma^2\delta(\tau)$, meaning that there is no correlation between the samples $x[s]$ and $x[n], n \neq s$. Furthermore, the magnitude of its only non-zero element $R_X(0) = \sigma^2$ equals the variance of the process; for the standard Gaussian case, the variance $\sigma^2 = 1$ (see `help randn`). As a consequence, the true PSD of X_t is a constant, unity, for all frequencies. Such a signal is referred to as **white noise** because it has a constant spectrum independent of frequency, in an, albeit approximate, analogy with white light. The difference between this ideal PSD and those estimated from the datasets generated in MATLAB is due, in part, to their finite lengths. One method to improve these estimates is to apply frequency domain smoothing.

1. Use a zero-phase FIR filter with impulse response 0.2*[1 1 1 1 1] to smooth the PSD estimate (periodogram) computed in the previous exercise (use the `filter` command). Does this improve the apparent PSD estimate? [5]
2. Generate a 1024-sample sequence of WGN and subdivide it into eight non-overlapping 128-sample segments. Calculate the PSD estimates for all eight segments, analyse their variation and comment. [5]
   Hint: You do not need to show all eight time series and periodograms, however, if you choose to show all the segments, use the `subplot` function.
3. Average these eight results to yield a new PSD estimator called the *averaged periodogram*. Display this result and comment on the differences between this estimate and the individual PSDs. [5]

Mathematical Comments

The periodogram and averaged periodogram are estimates of the PSD at N discrete frequencies. The estimate of the PSD, $\widehat{P}_X(f)$, at each frequency, is a random variable which has an associated probability density function. For instance, for the vector $\mathbf{x}$ for which the elements have been drawn from a Gaussian pdf, the estimate of its PSD at each frequency is distributed according to a χ^2-distribution.

3.2 Spectrum of autoregressive processes

Filter a 1064-sample WGN sequence $\mathbf{x}$ using the MATLAB command `filter` with the coefficients $\mathbf{b} = 1$ and $\mathbf{a} = [1,\ 0.9]$, and remove the first 40 values of the resulting signal **y** as these are affected by the transient effects of the filter. Plot and compare **x** and **y** in the time domain.

The sequence **y** is said to be generated by a first-order AutoRegressive model, AR(1), defined by two parameters: the power of **x** and the value of the AR filter parameter a_1. Recall that the theoretical PSD of an AR(p) process has the form

$$P_Y(f) = \frac{\sigma_X^2}{\left|1 + \sum_{k=1}^{p} a_k e^{-\jmath 2\pi k f}\right|^2}, \tag{18}$$

and the exact PSD of the considered filtered signal is given by (since the random input **x** has unit variance $\sigma_X^2 = 1$)

$$P_Y(f) = \frac{1}{\left|1 + 0.9 e^{-\jmath 2\pi f}\right|^2}, \tag{19}$$

which can be generated with `[h,w]=freqz([1],[1 0.9],512)`.

1. Plot the result with `plot(w/(2*pi),abs(h).^2)`. Explain the shape and the cutoff frequency of this spectral estimate. [5]

2. Now calculate the periodogram of $\mathbf{y}$ with your function `pgm` and show it **on the same plot in red**. Explain the differences. [5]

3. Zoom in on the interval $f = [0.4, 0.5]$. Why are the results different? Explain the role of the underlying rectangular windowing within the periodogram estimator. [5]

Note: The AR model is a filter whose input X is WGN and output Y has a PSD of the form

$$P_Y(f) = |H(f)|^2\, P_X(f),$$

where $P_X(f)$ is the PSD of the input X, and $H(f)$ is the frequency response of the AR model. Since the input sequence is zero–mean unit–variance white noise, we have $P_X(0) = \sigma^2 = 1$, while $H(f)$ can be found from the filter vectors $\mathbf{a} = [1, 0.9]$ and $\mathbf{b} = [1]$.

Model-based methods for PSD estimation assume that the available samples, as **y** generated above, are the output of a dynamical model. This principle is used in low-bit rate speech coding, where the speech data is assumed to be autocorrelation ergodic and generated by an AR model, with typically 12 parameters, driven by either periodic or random noise.

Model based PSD. By assuming that the sequence **y** is generated by an e.g. AR(1) model with two parameters (a_1, σ_X^2), the calculation of the PSD simplifies into the estimation of these two quantities, that is, $\widehat{a}_1$ and the input variance $\widehat{\sigma}_X^2$. These estimates are, in turn, calculated from the estimate of the correlation function of Y, $\widehat{R}_Y$, according to

$$\widehat{a}_1 = -\widehat{R}_Y(1)/\widehat{R}_Y(0) \tag{20}$$

$$\widehat{\sigma}_X^2 = \widehat{R}_Y(0) + \widehat{a}_1 \widehat{R}_Y(1). \tag{21}$$

4. Use the `xcorr` function to estimate $\widehat{a}_1$ and $\widehat{\sigma}_X^2$ for the sequence $\mathbf{y}$ as generated above, and calculate the model-based PSD estimate according to [5]

$$\widehat{P}_{\mathbf{y}}(f) = \frac{\widehat{\sigma}_X^2}{\left|1 + \widehat{a}_1 e^{-\jmath 2\pi f}\right|^2},$$

where $\widehat{P}_{\mathbf{y}}(f)$ may again be generated with `freqz`. Compare the result with that obtained from the periodogram estimate.

5. Repeat the above procedure for the sunspot time series, that is, compare its periodogram with a model-based PSD. Evaluate the results for the original series, its mean-centred version, and for different model orders. How does under/overmodelling affect the PSD estimate? [10]

3.3 Spectrogram for time-frequency analysis: dial tone pad

Time-frequency analysis studies the time evolution of the spectral contents of signals. This is particularly useful in everyday life, such as when using a mobile phone, watching a film from the web, or even in your ear canal: a very sophisticated time-frequency analysis is performed which lets you enjoy music and at the same time communicate with others. In this section, we employ fast finite Fourier transform to illustrate the usefulness of time-frequency analysis in touch-tone devices.

The underlying concept of touch-tone telephone dialling is the Dual Tone Multi-Frequency (DTMF) system, which assigns a signal composed of two sinusoids to each button of the keypad, that is

$$y[n] = \sin(2\pi f_1 n) + \sin(2\pi f_2 n), \tag{22}$$

where f_1, f_2 are the frequencies in Hz, and n is the time index.

The pairs of frequencies corresponding to each key are shown in Table 1 and an example of the signal corresponding to the digit 0 is shown in Figure 2.

	1209 Hz	1336 Hz	1477 Hz
697 Hz	**1**	**2**	**3**
770 Hz	**4**	**5**	**6**
852 Hz	**7**	**8**	**9**
941 Hz	$*$	**0**	**#**

Table 1: Dial pad frequencies

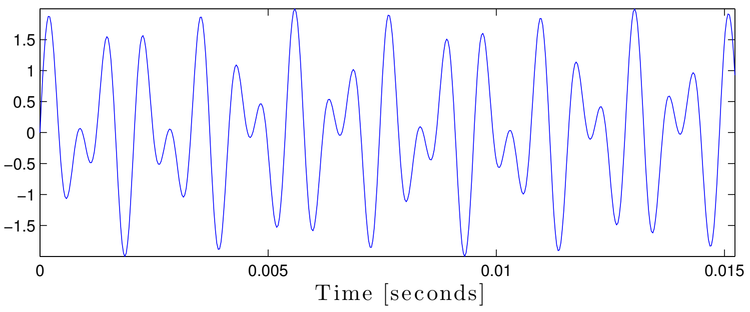

Figure 2: A two-tone signal for the digit 0 within the DTMF system.

1. Generate a random London landline number, i.e. a sequence of the form: `020 XXXX XXXX`, where the last eight digits are drawn from the set $\{0,1,2,3,4,5,6,7,8,9\}$ (with uniform probability). Compute the discrete time sequence $\mathbf{y}$ according to Eq. (22) and Table 1 using the sampling rate of 32768 [Hz], assume that each digit is pressed for 0.25 [s] followed by an idle time of 0.25 [s] so that your dialled sequence lasts for $0.25 * 21 = 5.25$ [s]. Explain why the proposed sampling rate is appropriate and plot the signal $\mathbf{y}$ for any two different keys and the idle sequence, adjust zoom and axis parameters to show the difference between the tones. [5]

2. Familiarise yourself with the function `spectrogram` within the MATLAB documentation, and use it to analyse the spectral components of the sequence $\mathbf{y}$. Choose the parameters of the function carefully so that the FFT is computed for non-overlapping segments containing touch-tone information corresponding to either a single key, or to the idle time in between pressed keys (it might be useful to consider a Hanning window in this part). Analyse the results and plot the corresponding FFT segments. [10]

3. Explain, using Table 1 and your spectrum estimates, whether it is possible to identify the sequence generated by a key press, and elaborate on how you would perform key classification. [5]

4. In real-world conditions, the signal is corrupted by channel noise. Repeat Parts 1-3 while adding white noise to the sequence $\mathbf{y}$. Consider three cases, starting from low-variance noise until the original signal is, at least visually, completely immersed in noise. Explain both analytically and from simulations how the noise affects the tone identification. Refer to Figure 3 for guidance. [10]

3.4 Real world signals: Respiratory sinus arrhythmia from RR-Intervals

Note: Use the RRI data from Section 2.5

Respiratory sinus arrhythmia (RSA) refers to the modulation of cardiac function by respiratory effort. This can be readily

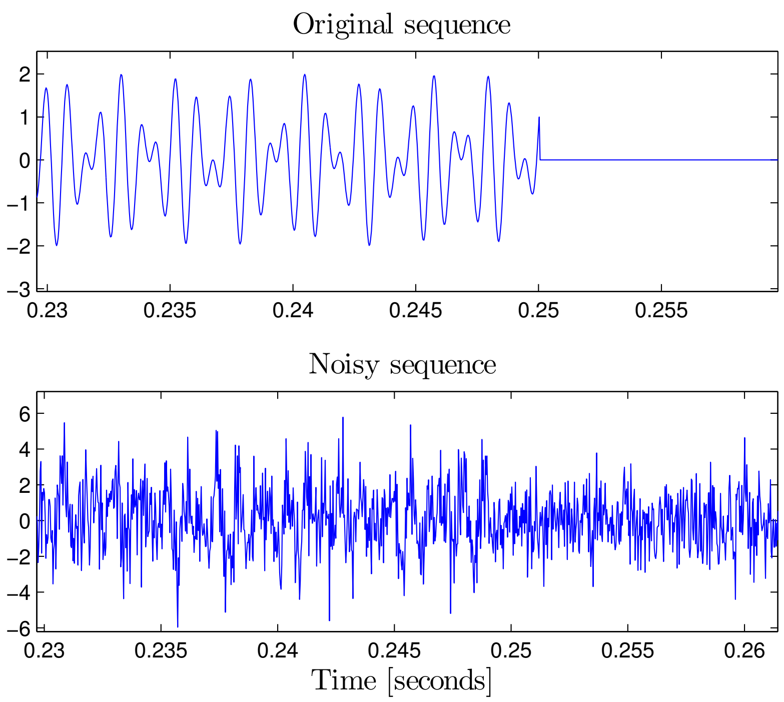

Figure 3: Example of original and noisy tone dialling signals.

observed by the speeding up of heart rate during inspiration ("breathing in") and the slowing down of heart rate during expiration ("breathing out"). The strength of RSA in an individual can be used to assess cardiovascular health. Breathing at regular rates will highlight the presence of RSA in the cardiac (ECG) data.

a) Apply the standard periodogram as well as the averaged periodogram with different window lengths (e.g. 50 s, 150 s) to obtain the power spectral density of the RRI data. Plot the PSD of the RRI data obtained from the three trials separately. [10]

b) What is the difference between the PSD estimates of the RRI data from the three trials? Can you identify the peaks in the spectrum corresponding to frequencies of respiration (and their harmonics) for the three experiments? [5]

4 Optimal filtering - fixed and adaptive

Aims:

- To review adaptive system identification.
- To introduce the Wiener filter as the optimal least squares filter for stationary signals.
- To consider online estimation of optimal filter coefficients based upon adaptive filtering in system identification and prediction setting.
- To illustrate the utility of adaptive filtering on real world speech (your own voice).

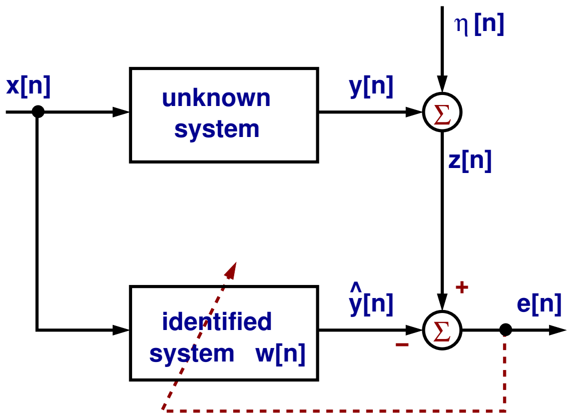

Figure 4: Real world system identification.

A system identification configuration is shown in Figure 4. The signal $x[n]$ is the input to an unknown system represented by a moving average (FIR) filter with fixed parameters $\{w[0], w[1], \ldots, w[N_w]\}$. The output of the unknown system is given by

$$y[n] = \sum_{i=0}^{N_w} w[i]x[n-i], \tag{23}$$

or in a more concise notation

$$y[n] = \mathbf{w}_n^T \mathbf{x}[n],$$

where $\mathbf{w}_n = [w[n], w[n-1], \ldots, w[n-N_w]]^T$ and $\mathbf{x}[n] = [x[n], x[n-1], \ldots, x[n-N_w]]^T$ are (N_w+1)-element column vectors[5]. A noise signal $\eta[n]$ is added to $y[n]$ at the output of the system to yield a realistic measured output signal $z[n]$.

An optimum filter, operating on the input $x[n]$ and measured output $z[n]$, is to be designed to produce the closest match to the unknown system. It is assumed that such filter is a moving average filter of the same order as the unknown system, and its (optimal) coefficients are denoted by $\mathbf{w}_{opt}$.

The filtering problem arises in a number of applications such as optimal control, acoustic dereverberation, and channel identification in telecommunications. In the linear case, the Wiener solution is found by minimising the mean-square error between $z[n]$ and $\hat{y}[n]$, where $\hat{y}[n]$ is the output of the optimum filter driven by $x[n]$ (Fig. 4).

If $x[n]$ and $\eta[n]$ are assumed to be uncorrelated stochastic processes, then the mean square error takes the form

$$\begin{aligned}
\mathbb{E}\left\{e^2[n]\right\} &= \mathbb{E}\left\{(z[n]-\hat{y}[n])^2\right\} \\
&= \mathbb{E}\left\{(z[n]-\mathbf{w}_{opt}^T\mathbf{x}[n])^2\right\} \\
&= \mathbb{E}\left\{z^2[n]-2\mathbf{w}_{opt}^T z[n]\mathbf{x}[n]+\mathbf{w}_{opt}^T\mathbf{x}[n]\mathbf{x}^T[n]\mathbf{w}_{opt}\right\} \\
&= \mathbb{E}\left\{z^2[n]\right\}-2\mathbf{w}_{opt}^T\mathbb{E}\left\{z[n]\mathbf{x}[n]\right\}+\mathbf{w}_{opt}^T\mathbb{E}\left\{\mathbf{x}[n]\mathbf{x}^T[n]\right\}\mathbf{w}_{opt}.
\end{aligned} \tag{24}$$

[5]Recall that boldface letters are used to denote vector quantities and the superscript $(\cdot)^T$ represents a vector transpose, also $\mathbf{x}(n) = [x[n], x[n-1], x[n-2], \ldots, x[n-N_w]]^T$ is the value of the vector $\mathbf{x}$ at a discrete time n.

The right hand side of equation (24) contains a vector of crosscorrelation components and a matrix of autocorrelation components, i.e.

$$\mathbf{p}_{zx} = \mathbb{E}\{z[n]\mathbf{x}[n]\} = \begin{bmatrix} \mathbb{E}\{z[n]x[n]\} \\ \mathbb{E}\{z[n]x[n-1]\} \\ \vdots \\ \mathbb{E}\{z[n]x[n-N_w]\} \end{bmatrix} = \begin{bmatrix} r_{zx}(0) \\ r_{zx}(-1) \\ \vdots \\ r_{zx}(-N_w) \end{bmatrix} \tag{25}$$

$$\mathbf{R}_{xx} = \mathbb{E}\left\{\mathbf{x}[n]\mathbf{x}^T[n]\right\} = \begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \cdots & r_{xx}(-N_w) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(-N_w+1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N_w) & r_{xx}(N_w-1) & \cdots & r_{xx}(0) \end{bmatrix}. \tag{26}$$

Differentiating equation (24) with respect to $\mathbf{w}_{opt}$ and setting the result to zero yields the **optimum Wiener filter** solution

$$\mathbf{w}_{opt} = \mathbf{R}_{xx}^{-1}\mathbf{p}_{zx}.$$

4.1 Wiener filter

Generate a 1000-sample WGN sequence $\mathbf{x}$ and present it as an input to the *unknown system* in Figure 4. The unknown system is described by a filter with coefficients $\mathbf{b} = [1, 2, 3, 2, 1], \mathbf{a} = [1]$, you should normalise the unknown system output $y[n]$ so that its variance is unity. Now, generate another WGN sequence $\eta[n]$ of $N = 1000$ samples and with standard deviation $\sigma = 0.1$, and generate $z[n]$ by adding the noise $\eta[n]$ to the filter output $y[n]$. Calculate the Signal-to-Noise Ratio (SNR) in dB for $z[n]$.
Note: The power of zero mean white noise is σ^2. When a signal is scaled by a factor c, its standard deviation σ gets scaled by the same factor.

1. Calculate the statistics $\mathbf{R}_{xx}$ and $\mathbf{p}_{zx}$ as given in Eqs. (25) and (26) and find the optimal coefficients of the Wiener filter. Are they similar to those of the unknown system? [10]

2. Repeat the experiment by varying the scaling applied to the additive noise so that its variance takes six different values in the region $\sigma^2 \in [0.1, 10]$ and re-calculate the SNR for each case. Explain the effects of different noise powers upon the Wiener solution. Also, analyse the effect upon the Wiener solution of assuming N_w greater than 4. [5]

3. The Wiener solution is block-based, therefore, it requires that all the input and output samples are available, and one of its drawbacks is its computational complexity. Estimate the number of multiplications and additions in the calculation of the Wiener solution. [5]
   Note: The number of operations to find the inverse of an arbitrary $N \times N$ matrix is of the order of magnitude $\mathcal{O}(N^3)$.

4.2 The least mean square (LMS) algorithm

The Wiener solution assumes statistical stationarity, however, in real-world applications the unknown system may be time-varying and, consequently, its output becomes nonstationary. To study such signals, adaptive filters that approximate, in a recursive fashion, the Wiener solution are considered. The simplest adaptive filter is based on the least mean square (LMS) algorithm and is defined by

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e[n]\mathbf{x}(n), \qquad n = 0, 1, \ldots \tag{27}$$

where μ is the adaptation gain which controls the stability of the algorithm, and $\mathbf{w}(0) = \mathbf{0}$. Notice that the weight vector of the adaptive filter, that is, $\mathbf{w}(n)$ is time-varying.

The error $e[n]$ is calculated as the difference between $z[n]$ and the output of the adaptive filter $\hat{y}[n]$, given by

$$\hat{y}[n] = \mathbf{w}^T(n)\mathbf{x}(n) = \sum_{m=0}^{N_w} w_m(n)x(n-m) \tag{28}$$

$$e[n] = z[n] - \hat{y}[n]. \tag{29}$$

Note: $y[n], z[n], \hat{y}[n]$ and $e[n]$ are scalar quantities, whereas $\mathbf{w}(n)$ and $\mathbf{x}(n)$ are $(N_w+1)\times 1$ column vectors.

1. Write a MATLAB routine called `lms` whose inputs are the N-sample vectors $\mathbf{x} = [x[1], \ldots, x[N]]^T$ and $\mathbf{z} = [z[1], \ldots, z[N]]^T$, the adaptation gain μ, and the order of the adaptive filter (N_w+1), while the output is the LMS estimate $\hat{y}[n]$ for $n = 1, \ldots, N$. The function must also output the error vector $e[n]$ and the $(N_w+1)\times N$ matrix containing the evolution of the adaptive weights in time. Apply this algorithm to the $\mathbf{x}$ and $\mathbf{z}$ signals used in Section 4.1 and explain the result. [10]

2. Choose the adaptation gain μ to be 0.01. Plot and examine the time evolution of the coefficients and show the (squared) estimate error. Explain the effect of increasing and decreasing μ? When does the adaptive filter converge to the Wiener solution? [5]
   Note: Plot the evolution of all five coefficients on one graph and vary the adaptation gain within the interval $[0.002, 0.5]$.

3. What is the computational complexity of the LMS algorithm? (How many multiplications and additions are required at each iteration?) [5]

4.3 Gear shifting

For statistically stationary environments it would be advantageous to use *gear shifting*, that is, to reduce the adaptation gain in time so as to obtain improved estimates in the steady state. To illustrate this, implement a time-varying adaptation gain and observe and plot the behaviour of both the filter weights and the error. If the error is high, what values of the adaptation gain would reduce the error? Incorporate a subroutine in your `lms` function to calculate a suitable adaptation gain for the $(n+1)$-coefficient update based on the n^{th} error $e[n]$. Carefully explain the criteria for manipulating the adaptation gain. [10]

Note: To compare different adaptation gain policies, always initialise the seed before generating random numbers. Also, assume that the maximum permissible overshoot for each coefficient is 20% of its true value; then compare the different methods by considering the rise time in each case. The best method will give the shortest rise time.

4.4 Identification of AR processes

Linear adaptive filters have a wide range of applications. Speech recognition, for example, involves non-stationary signals, and therefore requires adaptive estimation. Consider the structure shown in Figure 5 consisting of two separate stages: the **synthesis** of a signal from white noise using an autoregressive model, and the **analysis** stage performs the adaptation of coefficients to recreate the original signal.

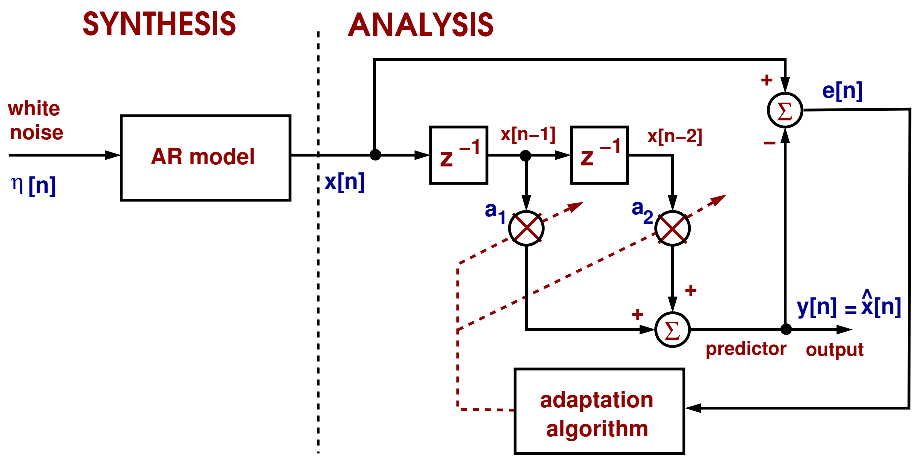

Figure 5: Synthesis and analysis structure for an AR model

1. Write a programme to implement the structure in Fig. 5 and use an AR model with parameters $\mathbf{a} = [1, 0.9, 0.2]$ (in MATLAB notation). If the adaptive LMS algorithm is successfully implemented, to what values should a_1 and a_2 converge? Explain your results. [10]

2. Use your code to analyse and explain the evolution of a_1 and a_2 for four different adaptation gains of your choice (starting from an adaptation gain of 0.01 would be appropriate). [5]

4.5 Speech recognition

In speech recognition, only the right part of the structure in Fig. 5 is used. A characteristic sound is used as an input, and the filter finds the coefficients $a_1, a_2, \ldots, a_p$, so that the average output squared error is minimised; this way, when the input is a delayed version of the sound, the estimator will output the sound itself - and therefore it is referred to as a predictor. A p^{th}-order predictor uses samples delayed by $1, 2, \ldots, p$ time units to create its output; the predictor in Figure 5 is a second order predictor.

1. Test the performance of such a predictor on real-world audio signals. Record your own voice corresponding to the sounds "e", "a","s", "t" and "x". Consider the sampling frequency of $f_s = 44100$Hz and $N = 1000$ samples. Use each file as an input to the predictor and investigate the selection of the adaptation gain and the order of the predictor. Also consider gear shifting. [5]

2. How would you find an optimal filter length? Suggest some heuristic and analytical approaches. [5]

   A standard measure for the performance of a predictor is the prediction gain

$$R_p = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right), \tag{30}$$

   where σ_x^2 and σ_e^2 denote the variance of the input and error signals respectively.

3. Assess the performance of the predictor for each audio recording (typical prediction gain values are on the order of 1-25). Calculate and explain the prediction gains of the corresponding predictors for a 1000-sample recording of your own speech, as in Part 4.5.1 above, but this time at the sampling frequency of $f_s = 16000$ Hz. Comment on the relationship between the sampling frequency and the number of data samples in order to obtain quasi-stationary vowel sounds[6], and for the learning curves to converge. [10]

4.6 Dealing with computational complexity: sign algorithms

A simplified version of the LMS algorithm is the class of the sign LMS algorithms, given by [15]

$$\begin{aligned}
\text{signed}-\text{error}\ \ \text{(sign algorithm)} \qquad & \mathbf{w}(n+1) = \mathbf{w}(n) + \mu\text{sign}\big(e[n]\big)\mathbf{x}(n) \\
\text{signed}-\text{regressor} \qquad & \mathbf{w}(n+1) = \mathbf{w}(n) + \mu e[n]\text{sign}\big(\mathbf{x}(n)\big) \\
\text{sign}-\text{sign} \qquad & \mathbf{w}(n+1) = \mathbf{w}(n) + \mu\text{sign}\big(e[n]\big)\text{sign}\big(\mathbf{x}(n)\big)
\end{aligned} \tag{31}$$

Write a MATLAB routine to simulate these algorithms and compare their performance with that of the basic LMS algorithm for the AR parameter identification in Part 4.4, and for one speech file in Part 4.5. Comment upon the convergence properties and accuracy of the sign LMS algorithms.

[6]When their statistics vary slowly, the signal can be considered stationary for a fixed period of time.

5 A Real World Case Study: Vinyl Denoising

While walking around Camden market, you bought a vinyl copy of the fourth Led Zeppelin album, released in 1971, which seemed to be in mint condition. While listening at home, at the start of the fourth track—Stairway To Heaven—you hear an unpleasant whistling tickling due to a small but irregular scratch across the grooves in Track 4. You decide to put your best signal processing knowledge forward and remove this tick digitally.

1. Download the file `http://www.commsp.ee.ic.ac.uk/~mandic/vinyl.mat`, load it into MATLAB and analyse the matrix `s2h` containing the corrupted segment of the album sampled at 44.1 kHz in stereo. Listen to both channels in MATLAB using the command `sound` and the appropriate sampling rate included in the loaded data, and plot the periodogram for each channel. Is it possible to differentiate between the spectral components corresponding to the song and to the tick? [10]

2. A friend gives you their own digital copy of Stairway to Heaven, contained in the matrix `s2h_original`. Listen to the recording, plot the periodogram for both channels and compare to those of your own copy corrupted by the tick. How would you identify the spectral components of the tick? [10]

3. Now that you know the spectral information of the ticks in the recording, design a fixed-coefficient digital filter to remove this distortion. Carefully choose the limits of the spectral region to be filtered out and apply as many such digital filters as necessary to remove all the tick components. Assess your method by listening to the filtered signals (qualitative performance measure). Explain your choice of filters, and the criteria you used to set their parameters. [20]

4. After the tick has been removed, plot the periodograms (for each channel) of the filtered signal and compare to the periodograms of the original and corrupted recordings. Use the blue, red, and green lines for the original, corrupted and filtered signals, respectively. Compute the relative error of your estimate with respect to the *ground truth* recording, provided by your friend, according to (quantitative performance measure) [20]

$$\frac{\|P_c - \widehat{P}_c\|}{\|P_c\|}, \tag{32}$$

   where P_c is the (non-corrupted) periodogram of channel c and $\widehat{P}_c$ is your estimate of P_c, $c \in \{\text{left, right}\}$. Comment on the effects of removing the unwanted tick in terms of the spectral content of the song.

   Note: Alternatively, consider the prediction gain R_p in this part.

5. In order to develop a supervised adaptive denoising algorithm, which does not require the analysis of the whole content of the disc, implement a least mean square (LMS) estimator that filters the recording of the faulty song and uses the original recording as a teaching signal to eliminate the noise in your copy. Use any noise cancelling adaptive filtering configuration, and implement the normalised LMS algorithm (NLMS), given by [20]

$$\hat{y}[n] = \mathbf{w}_n^T \mathbf{x}_n \tag{33}$$

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\Big(y[n] - \hat{y}[n]\Big)\frac{\mathbf{x}_n}{\|\mathbf{x}_n\|_2^2} \tag{34}$$

   where $\mathbf{x}_n = [x[n-1], \ldots, x[n-p]]^T$ is the vector of corrupted signal samples, y is the teaching signal, $\hat{y}$ is the NLMS estimate, and $\mathbf{w}_n \in \mathbf{R}^{p\times 1}$ is the weight vector at time instant n. Exploit different filter parameters to validate the algorithm (see Part 4).

   Explain whether the normalised LMS provides a better estimate in terms of the square error than the standard LMS.

6. Your fellow audiophiles have heard about your successful audio enhancement of vinyl albums; they now require your expertise to remove ticks with their own albums. This time you receive a recording from a fairly newer album of a slightly different music genre: Liquid Tension Experiment's eponymous debut (1998). Note that LTE's recording is much richer in frequency components than the analysed section of Led Zeppelin IV, this is due to the use of distorted guitars (with high harmonic content due to the rectification stage of tube amplifiers), analogue synthesisers (composed of wide band harmonic oscillators), and the multiple layers recorded for each instrument. Apply the adaptive filter **designed in Part 5.5** to LTE's recordings (find the data in the matrices `um`, and `um_original`), and analyse the performance of your method as in Part 5.4. How does this differ from the previous result? What can you say about the spectral loss related to removing the ticks in this song? [20]

References

Hayes, M.H., Statistical Digital Signal Processing and Modeling, Wiley, 1996.

Haykin, S., Adaptive Filter Theory, Third Edition, Prentice-Hall, 1996.

MATLAB Documentation: User's and Reference Guides, and Signal Processing Toolbox User's Manual.

Burrus, C.S., et. al, Computer-Based Exercise for Signal Processing Using MATLAB, Prentice Hall,1994

Mandic, D.P and Chambers, J.A., Recurrent Neural Networks for Prediction, Wiley, 2001

Mandic, D.P and Goh, V.S.L, Complex Valued Nonlinear Adaptive Filters, Wiley, 2009