3rd Year Project SF1: Data Analysis - Statistical Signal Processing with Applications in Audio

Project leader: Prof. Simon Godsill, sjg@eng.cam.ac.uk.

June 3, 2022

1 Demonstrator sessions

- Sessions will run in the DPO on Tuesdays and Fridays as follows: **Tuesdays 11am-1pm and Fridays 9-11am and pm**
- Location: In Person, DPO Clusters 3 and 4.
- **First session** is on Friday 13th May 9-11am. This will commence with a mini-lecture in LR10 for all students.
- Intermediate and Final reports to be handed in electronically via Moodle by 4pm each week, due on 20 May, 27 May and 10 June (**Final Report**). Do not miss the 4pm deadlines - no report = no marks.

2 Project Summary

Prerequisites: 3F1 and 3F3 recommended though not essential. If you have not done one of these then you will need to spend additonal preparation time going through the lecture notes on discrete time digital signal processing from these two courses.

This project introduces signal modelling/processing techniques and applies them to audio and musical signals. First non-parametric methods based on transforms such as the Discrete Fourier Transform (DFT) and its fast variant the FFT are studied and experiments are carried out with windowing, frequency resolution etc. These methods are then applied in an audio noise reduction setting, processing sound signals using overlap-add analysis/processing/synthesis to perform noise reduction. Then parametric models are introduced, with estimation using least squares, maximum likelihood and Bayesian techniques. The autoregressive (AR) model is used as an example here, and model choice is studied within likelihood and Bayesian probabilistic settings. Once again, the techniques are applied in audio signals, using models to perform packet loss concealment and interpolation of missing data in audio, as well as constrained interpolation for clipped and/or heavily quantised signals. Students will have

the opportunity to incude their own ideas into the applied schemes and there will be a competition for the best noise reduction performance from test audio datasets, evaluated using mean-sqaured error, perceptual criteria and (informal) listening tests.

You might like to bring your own preferred headphones for individual use so that you can assess the sound quality of your processed signals. Anyone who can't access headphones should let us know and we will provide some.

Example data and code is provided on the course's Moodle site:
https://www.vle.cam.ac.uk/course/view.php?id=70251,
and noisy data for the test processing will be provided in week 2.

3 Reports, Code and Audio Processings

You will make two short intermediate reports and one final report. These are to be submitted electronically via Moodle, along with the Matlab .m files that you wrote during the work and processed audio files for the examples given, saved in .wav format.

Intermediate short reports. These are each **2 sides maximum** plus figures plus code - due at end of weeks 1 and 2. They should contain summary answers to all questions posed, conclusions and plots of results from your experiments, as well as attaching the Matlab functions you have written along the way. Summarise results graphically wherever possible and keep text brief and to the point. Graphs/ figures must each be clearly captioned so the reader can interpret each one clearly, both from the main text and standalone. Make sure your code is clearly commented so that we can see what code generated what figures, etc. The number of figures is not limited, but figures should illustrate a point - marks will be deducted for acres of plotting of unnecessary results. Number your sections carefully corresponding to the questions asked below.

Final Report. This is **10 sides maximum** plus figures plus code - due at end of week 4. A detailed coverage of weeks 3-4 material. Same guidelines for figures and captions as for short reports. you will submit your full code and audio processings with this report.

4 Weeks 1-2

4.1 Spectrum estimation and Noise Reduction

4.1.1 Spectrum estimation - week 1

This section experiments with spectrum estimation using the Discrete Fourier Transform with Windowing. You will use the conclusions of this section to inform the design of your noise reducer software next week.

Recall the formula for the DFT of a signal x_n, and its inverse:

$$X_p = \sum_{n=0}^{N-1} x_n \, e^{-j\frac{2\pi}{N}np}$$

$$x_n = \frac{1}{N} \sum_{p=0}^{N-1} X_p \, e^{j\frac{2\pi}{N}pn}$$

Make sure you fully understand what this means and what frequency the pth component would correspond to in the DFT, see 1B Sig. and Data Analysis/ 3F3.

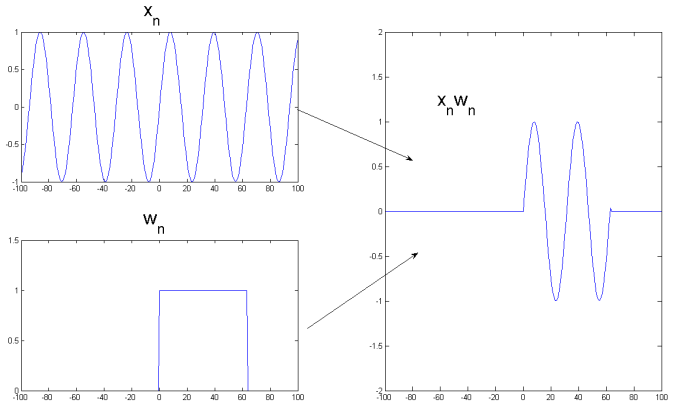

Figure 1: Windowing of Data prior to spectral analysis

Now, consider the DTFT of a windowed version of the signal $x_w = w_n x_n$

directly:

$$\begin{aligned} X_w(e^{j\omega T}) &= \sum_{n=-\infty}^{\infty} \{x_n\, w_n\}\, e^{-jn\omega T} \\ &= \sum_{n=-\infty}^{\infty} x_n \left\{ \frac{1}{2\pi} \int_0^{2\pi} W(e^{j\theta}) e^{jn\theta} d\theta \right\} e^{-jn\omega T} \\ &= \frac{1}{2\pi} \int_0^{2\pi} W(e^{j\theta}) \sum_{n=-\infty}^{\infty} x_n\, e^{-jn(\omega T-\theta)}\, d\theta \\ X_w(e^{j\omega T}) &= \frac{1}{2\pi} \int_0^{2\pi} W(e^{j\theta})\, X(e^{j(\omega T-\theta)})\, d\theta \end{aligned}$$

This is a reminder of the familiar result that the window function gets convolved with the spectrum of x when we take a windowed DFT.

Recall for example from 3F3 the Generalized Hamming Window:

$$w_n = \begin{cases} \alpha - \beta\, \cos\left(\frac{2\pi n}{N-1}\right), & n = 0, 1, ...N-1 \\ 0 & \text{Otherwise} \end{cases}$$

which has several common special cases:

- $\alpha = 1$, $\beta = 0$: the rectangle (or Boxcar or Dirichlet) window has

$$w_n = 1, \;\; n = 0, 1, ...N-1$$

- $\alpha = \beta = 0.5$ corresponds to the Hanning or Hann window
- $\alpha = 0.54$, $\beta = 0.46$ corresponds to the Hamming window

- all available as Matlab in-built functions. Refer to 3f3 lecture notes on Moodle for details of Windowed analysis with the DFT. Wikipedia also has a good reference entry under 'Window Functions'.

Type 'Help Window' to get started on this in Matlab.

You should now carry out experiments with hand-generated signals and real signals to determine the effects of windowing on spectrum analysis. You should particularly inlclude the following items in your explorations:

1. Write your own Matlab function 'DFT.m' to compute the DFT directly (i.e. not the Fast Fourier Transform (FFT)). Attempt to vectorise your code as much as possible to avoid expensive Matlab 'For' loops. Make sure your code works correctly by comparing your answer with that generated by Matlab's FFT function on the same data.

2. Use Matlab's timer functions e.g. 'tic' and 'toc' to test the time for running your function for different values of the data length N. Compare and plot the log-time against N with Matlab's in-built FFT function. How do these results compare with the theoretical expected order of computations from the DFT and FFT implemented in Matlab?

3. Now experiment with windowed spectrum analysis of some oscillating functions, e.g. $\exp(j\omega n)$, $\cos(\omega n)$, $\sin(\omega n)$ for various normalised frequencies from 0 to π. How does the FFT vary with data length N? Consider central lobe width, position of maximum, strength of sidelobes. You should consider several window functions, e.g. the Generalised Hamming Window, but also other families available within Matlab. It is generally easiest to study these properties in the log-spectral magnitude domain. *Note that it will be helpful to 'oversample' the FFT by zero-padding the data - Matlab allows this through the second argument in fft(X,N).*

4. Now add some noise to your data and see how the behaviour of the spectrum changes - use e.g. Matlab's randn function to generate Gaussian noise. Consider different levels of noise and how they effect the *position of the* maximum of the calculated spectrum.

5. Consider the effect of amplitude modulation of the signals, as would be present with real oscillations in e.g. speech data. Try signals of the form $a_n \exp(j\omega n)$, where a_n could be a linear gain increase $An + B$, a periodic modulation $1 + \beta \sin(\phi n)$ with $\phi << \omega$ and $\beta << 1$, or even a random modulation $a_n = a_{n-1} + v_n$ where v_n is a random noise (e.g. Gaussian - use 'randn' in Matlab). All of these are possible models for what happens to frequency components in speech/music signals over time.

6. Now load in some speech and music data into Matlab. Experiment with the functions Wavread and Wavwrite for reading/writing of .wav files. Isolate and plot some examples of consonants and vowels in speech, and steady notes/ transients in music. You can do this by careful plotting of the data and listening to short extracts over headphones - use the Matlab 'sound' command to listen to data.

7. Perform windowed spectrum analysis of the examples you have isolated *in the previous part.* Determine suitable window lengths and window types for each type of signal. Generally the aim will be to be able to clearly see as many separate frequency components in the data, with energy as well concentrated into as few frequency components as possible. This needs to be traded off against the time-varying nature of the data - too long a window length and you will see a mess of masses of overlapping/ broadened frequency components. What are the different characteristics of the spectrum for the different types of sound that you have isolated? *Note that for a steady state note or vowel we will be expecting to see something like the spectrum of a Fourier series in the DFT domain. For transients/consonants the effects will be much less localised in frequency.*

8. Challenge if time permits: How many musical notes can you identify in the dataset 'organ.wav' and what are their frequencies? Hint: each musical pitch is a 'near-periodic' waveform, so you should expect frequency components in the form of a Fourier series for each note in the mix. There

are lots in this example so you need a very high-resolution analysis - *use a very long frame of data!*

4.1.2 First short report - end of week 1

You can now write your first short report. Give brief answers to the above questions plus figures to provide evidence and any Matlab code you wrote (especially the DFT code). A good attempt at the Challenge question will gain some extra credit, but a perfectly adequate report can be done without the challenge question.

4.1.3 Noise Reduction - week 2

We now combine the spectrum analysis findings of last week with noise reduction functions in order remove background noise from sound recordings. A very simple and flexible way of doing this is via so-called overlap-add analysis/synthesis, see e.g.

Jont B. Allen (June 1977). 'Short Time Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform'. IEEE Transactions on Acoustics, Speech, and Signal Processing. ASSP-25 (3): 235–238.

Random additive background noise is a form of degradation common to all analogue measurement, storage and recording systems. In the case of audio signals the noise, which is generally perceived as 'hiss' by the listener, will be composed of electrical circuit noise, irregularities in the storage medium and ambient noise from the recording environment. The combined effect of these sources will generally be treated as one single noise process, although we note that a pure restoration should strictly not treat the ambient noise, which might be considered as a part of the original 'performance'. Random noise generally has significant components at all audio frequencies, and thus simple filtering and equalisation procedures are inadequate for restoration purposes.

Noise reduction has been of great importance for many years in many engineering disciplines. The classic least-squares work of Norbert Wiener [20] placed noise reduction on a firm analytic footing, and still forms the basis of many noise reduction methods. In the field of speech processing a large number of techniques has been developed for noise reduction, and many of these are more generally applicable to noisy audio signals.

Certainly the most popular methods for noise reduction in audio signals are still based upon short-time processing in the spectral domain. The reason for the success of these methods is that audio signals are usually composed of a number of line spectral components which correspond to fundamental pitch and partials of the note/ vowel being played. Although these line components are time-varying they can be considered as fixed over a short analysis window with a duration of perhaps 0.02 seconds or more. Hence analysing short blocks of data in the frequency domain will concentrate the signal energy into a relatively few frequency 'bins' with high signal-to-noise ratio. This means that noise reduction can be performed in the frequency domain while maintaining the important parts of the signal spectrum largely unaffected. Processing is performed in a transform domain, usually the discrete Fourier Transform (DFT) $Y(n,m)$ of sub-frames in the observed data $\{y_n\}$, a scheme known as teh Short-time Fourier Transform (STFT):

$$Y(n,m) = \sum_{l=0}^{N-1} w_l \; y_{(nM+l)} \; \exp(-jlm2\pi/N), \qquad m = 0, \ldots, N-1.$$

Index n is the sub-frame number within the data and m is the frequency component number. g_l is the time domain windowing function with suitable frequency domain characteristics, such as the Hanning or Hamming Window. N is

the sub-frame length and $M < N$ is the number of samples between successive sub-frames. Typically $M = N/2$ might be used here since it allows for simple reconstruction in the time domain by overlap-add, since for the generalised Hamming window we have always that:

$$w_l + w_{l+N/2} = 1$$

and hence the overall gain will always be unity.

Processing is then performed on the spectral components $Y(n, m)$ in order to estimate the spectrum of the 'clean' data $X(n, m)$:

$$\hat{X}(n, m) = f(Y(n, m))$$

where $f(.)$ is a function which performs noise reduction on the spectral components. We will discuss some possible forms for $f(.)$ in the next section.

The estimated spectrum $\hat{X}(n, m)$ is then inverse DFT-ed to obtain a time domain signal estimate for sub-frame n at sample number $nM + l$:

$$\hat{x}^n_{nM+l} = \frac{1}{N} \sum_{m=0}^{N-1} \hat{X}(n, m) \exp(jlm2\pi/N), \qquad l = 0, \ldots, N-1$$

Finally, the reconstructed signal in sub-frame n is obtained by an overlap-add method:

$$\hat{x}_{nM+l} = \sum_{m \in \mathcal{M}} \hat{x}^m_{nM+l}$$

where $\mathcal{M} = \{m;\ mM \leq nM + l < mM + N\}$. and $\mathcal{M}_0 = \{m;\ 0 \leq mM + l < N\}$. The summation sets in these two expressions are then over all adjacent windows which overlap with the current output sample number. In the case of the suggested $N/2$ overlap, it is just a case of summing together the current frame with the last $N/2$ samples from the previous frame, see code example given shortly.

4.2 Noise reduction functions

The previous section has described a basic scheme for analysis, processing and resynthesis of noisy audio signals using the DFT, a similar method to that first presented by Allen [2]. We have not yet, however, detailed the type of processing functions $f(.)$ which might be used to perform noise reduction for the spectral components. Many possible variants have been proposed in the literature, some based on heuristic ideas and others on a more rigorous basis such as the Wiener, maximum likelihood or maximum *a posteriori* estimation. See [6] for a review of the various suppression rules from a statistical perspective.

4.2.1 The Wiener solution

If the noise is assumed to be additive and independent of the signal, then the frequency domain Wiener filter which minimises the mean-squared error of the time domain reconstruction is given by [15] (see 3F3 notes on Wiener Filters):

$$H(\omega) = \frac{\mathcal{S}_X(\omega)}{\mathcal{S}_X(\omega) + \mathcal{S}_N(\omega)}$$

where $\mathcal{S}_X(\omega)$ is the power spectrum of the signal and $\mathcal{S}_N(\omega)$ is the power spectrum of the noise. If this gain can be calculated at the frequencies corresponding to the DFT bins then it can be applied as a noise reduction filter in the DFT domain. If we interchange the continuous frequency variable ω with the DFT bin number m the following 'pseudo-Wiener' noise reduction rule is obtained:

$$f(Y(m)) = \frac{\mathcal{S}_X(m)}{\mathcal{S}_X(m) + \mathcal{S}_N(m)} Y(m)$$

where we have dropped the sub-frame variable n for convenience.

The problem here is that we do not in general know any of the required terms in this equation except for the raw DFT data $Y(m)$. However, in many cases it is assumed that $\mathcal{S}_N(m)$ is known and it only remains to obtain $\mathcal{S}_X(m)$ ($\mathcal{S}_N(m)$ can often be estimated from 'silent' sections where no music is playing, for example). A very crude estimate for $\mathcal{S}_Y(m)$, the power spectrum of the noisy signal, can be obtained from the amplitude squared of the raw DFT components $|Y(m)|^2$. An estimate for the signal power spectrum is then:

$$\mathcal{S}_X(m) = \begin{cases} |Y(m)|^2 - \mathcal{S}_N(m), & |Y(m)|^2 > \mathcal{S}_N(m) \\ 0, & \text{otherwise} \end{cases}$$

where the second case ensures that the power spectrum is always greater than or equal to zero, leading to a noise reduction function of the form:

$$f(Y(m)) = \begin{cases} \frac{|Y(m)|^2 - \mathcal{S}_N(m)}{|Y(m)|^2} Y(m), & |Y(m)|^2 > \mathcal{S}_N(m) \\ 0, & \text{otherwise} \end{cases}$$

Defining the power signal-to-noise ratio at each frequency bin to be $\rho(m) = (|Y(m)|^2 - \mathcal{S}_N(m))/\mathcal{S}_N(m)$, this function can be rewritten as:

$$f(Y(m)) = \begin{cases} \frac{\rho(m)}{1+\rho(m)} Y(m), & \rho(m) > 0 \\ 0, & \text{otherwise} \end{cases}$$

4.2.2 Spectral subtraction and power subtraction

The Wiener solution has the appeal of being based upon a well-defined optimality criterion. Many other criteria are possible, however, including the well known spectral subtraction and power subtraction methods [5, 16, 4, 14, 12].

In the spectral subtraction method an amount of noise equal to the root mean-squared noise in each frequency bin, i.e. $\mathcal{S}_N(m)^{1/2}$, is subtracted from the spectral amplitude, while once again the phase is left untouched. In other words we have the following noise reduction function:

$$f(Y(m)) = \begin{cases} \frac{|Y(m)|-\mathcal{S}_N(m)^{1/2}}{|Y(m)|}Y(m), & |Y(m)|^2 > \mathcal{S}_N(m) \\ 0, & \text{otherwise} \end{cases}$$

Rewriting as above in terms of the power signal-to-noise ratio $\rho(m)$ we obtain:

$$f(Y(m)) = \begin{cases} \left(1 - \frac{1}{(1+\rho(m))^{1/2}}\right)Y(m), & \rho(m) > 0 \\ 0, & \text{otherwise} \end{cases}$$

Another well known variant upon the same theme is the power subtraction method, in which the restored spectral power is set equal to the power of the noisy input minus the expected noise power:

$$f(Y(m)) = \begin{cases} \left(\frac{|Y(m)|^2-\mathcal{S}_N(m)}{|Y(m)|^2}\right)^{1/2}Y(m), & |Y(m)|^2 > \mathcal{S}_N(m) \\ 0, & \text{otherwise} \end{cases}$$

and in terms of the power signal-to-noise ratio:

$$f(Y(m)) = \begin{cases} \left(\frac{\rho(m)}{1+\rho(m)}\right)^{1/2}Y(m), & \rho(m) > 0 \\ 0, & \text{otherwise} \end{cases}$$

The gain applied here is the square root of the Wiener gain.

Overlap add analysis/synthesis

```
overlap_add.m

x_in=0.5*cos([1:10000]*pi/4)+sin([1:10000]*pi/100)+randn(1,10000);
y_out=0*x_in;

N=512;
overlap=256;

x=buffer(x_in,N,overlap);

[N_samps,N_frames]=size(x);

x_w=repmat(hanning(N),1,N_frames).*x;

for frame_no=1:N_frames-2
```

```
    X_w(:,frame_no)=fft(x_w(:,frame_no));

    Y_w(:,frame_no)=X_w(:,frame_no);

    Y_w(2:N/8,frame_no)=0.1*X_w(2:N/8,frame_no);
    Y_w(N/4+1:N/2,frame_no)=0.2*X_w(N/4+1:N/2,frame_no);

    Y_w(N:-1:N/2+2,frame_no)=conj(Y_w(2:N/2,frame_no));

    y_w(:,frame_no)=ifft(Y_w(:,frame_no));

    y_out((frame_no-1)*overlap+1:(frame_no-1)*overlap+N)=...
y_out((frame_no-1)*overlap+1:(frame_no-1)*overlap+N)+y_w(:,frame_no)';

end
```

1. Above is a short piece of Matlab code for filtering of some audio data in the time-frequency domain. You can use this as the basis for your noise reduction functions later. It has been written by a technically competent student who doesn't know about commenting their code, and hence would have achieved a rather low mark for their report. You will of course do a much better job with your own submitted code.

   (a) What type of filter do you think this processing is trying to achieve?

   (b) Consider a time domain implementation of a filter with the same frequency response as the last part. Would this procedure be identical or different from a time domain version of the same filter and why?

   (c) What is the minimum processing delay that would be introduced by such a filtering procedure, assuming the code runs instantaneously and the audio samples are arriving one by one with no hardware delay due to A/D conversion, comms. etc.?

   (d) What is the reason for the line:

   ```
   Y_w(N:-1:N/2+2,frame_no)=conj(Y_w(2:N/2,frame_no));   ?
   ```

2. Now write your own noise reduction functions and experiment with their performance. You have freedom to choose all parameters, including the frame length, overlap, window function and noise reduction function. You should experiment with audio (music and/ or speech as you prefer) to which you have added noise, which may be white or non-white (how would you generate non-white noise?).

3. Measure the Mean-squared error for your filters as a function of the different parameters. *Note that the MSE compares the denoised signal with the original 'clean' signal. You will need to ensure that these two signals are perfectly aligned in time to calculate the MSE*. Does the MSE tally with how the results actually sound when you listen? What are the characteristics of the results in terms of distortion levels to the original signal? Are any processing artefacts introduced by the noise reduction?

4. Challenge if time permits:

   Now try and do some clever stuff to improve performance. Some things you might try include (but are not limited to):

   - Over-estimation of the noise floor
   - Leaving some of the noise behind in order to improve sound quality.
   - Averaging of power spectrum estimates over time and/or frequency - you could consider standard averagers, or how about nonlinear ones like median values and other order statistics?
   - Difficult: How about combining different frame lengths N for different pieces of data (vowel, consonant, etc.)?
   - Can you devise a simple method to automatically work out the noise power? How about if the noise levels are time-varying? Clues: look for times when there is no speech/ audio activity, or calculate the order statistics of the spectral powers over time (e.g. the minimum power levels over a number of frames).

   You can use ideas from the internet/ research papers if you wish, but you must cite any ideas that you have re-used in this way.

5. Testing. Present your best results on the test examples. We will then evaluate these by 'blind' listening tests across the whole group to evaluate the ranking of the methods you have generated - this will be carried out during weeks 3 and 4.

6. Code delivery. Your final Matlab code should be submitted in a format that enables the demonstrators to immediately run it on the test examples without code modification.

4.3 Parameter Estimation - weeks 3-4

In this final two weeks we will investigate parameter estimation methods in the time domain, and their application to data measured directly in the time domain ('time series'). We will explore probabilistic inference methods, especially Maximum Likelihood and Bayesian methods. There is a heavier mathematical content to this section, so read the background material carefully and ask a demonstrator if you don't follow. Some of the basics are familiar to you from parts of 3F1 anbd 3F3.

In parameter estimation we suppose that a random process $\{X_n\}$ depends in some well-defined stochastic manner upon an unobserved parameter vector $\boldsymbol{\theta}$. If we observe N data points from the random process, we can form a vector $\mathbf{x} = [x_1\, x_2\, \ldots x_N]^T$. The parameter estimation problem is to deduce the value of $\boldsymbol{\theta}$ from the observations $\mathbf{x}$. In general it will not be possible to deduce the parameters exactly from a finite number of data points since the process is random, but various schemes will be described which achieve different levels of performance depending upon the amount of data available and the amount of prior information available regarding $\boldsymbol{\theta}$.

We now define a general parametric model which will be used as the basis of our work in this last two weeks. This model is a fundamental building block for many more complex and sophisticated non-linear or non-Gaussian models in signal processing. However, although we use these models exclusively in this project, it should be borne in mind that there are indeed many classes of model in practical application that have none, or very little, linear Gaussian structure, and for these results specialised techniques are required for inference; none of the analytic results here will apply in these cases, but the fundamental structure of the likelihood and Bayesian inference methodology does indeed carry through unchanged, and inference can still be successfully performed, albeit with more effort and typically greater computational burdens. The Monte Carlo method, and in particular particle filtering and Markov chain Monte Carlo (MCMC) approaches, are well suited to inference in the hardest models with no apparent linear or Gaussian structure [neither of these are covered here, but look them up on Wikipedia if you are interested].

4.4 The Linear Gaussian model

In the general model it is assumed that the data $\mathbf{x}$ are generated as a function of the parameters $\boldsymbol{\theta}$ with an additive random modelling error term e_n:

$$x_n = g_n(\boldsymbol{\theta}, e_n)$$

where $g_n(.)$ is a deterministic and possibly non-linear function of the parameter $\boldsymbol{\theta}$ and the random error, or 'disturbance' e_n. Now we will consider the important special case where the function is linear and the error is additive, so we may write

$$x_n = \mathbf{g}_n^T \boldsymbol{\theta} + e_n$$

where $\mathbf{g}_n$ is a P-dimensional column vector, and the expression may be written for the whole vector $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{G}\boldsymbol{\theta} + \mathbf{e} \tag{1}$$

Linear model

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1^T \\ \mathbf{g}_2^T \\ \vdots \\ \mathbf{g}_N^T \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \dots & \mathbf{h}_P \end{bmatrix}$$

The columns $\mathbf{h}_i$ of $\mathbf{G}$ form a fixed basis vector representation of the data, for example sinusoids of different frequencies in a signal which is known to be made up of pure frequency components in noise. A variant of this model will be seen later in the section, the autoregressive (AR) model, in which previous data points are used to predict the current data point x_n. The error sequence $\mathbf{e}$ will here (but not necessarily in general) be assumed drawn from an independent, identically distributed (i.i.d.) noise distribution, that is,

$$p(\mathbf{e}) = p_e(e_1).p_e(e_2).\dots.p_e(e_N) \tag{2}$$

where $p_e(.)$ denotes some noise distribution which is identical for all n. All of the 'p' terms here are to be interpreted as probability densiy functions. Note then that we are considering here discrete time *white noise* processes as defined in 3F3, though not necessarily zero mean in the definition so far.

$\{e_n\}$ can be viewed as a modelling error, 'innovation' or observation noise, depending upon the type of model. When $p_e()$ is the normal distribution and g_n is a linear function, we have the Linear Gaussian model.

There are many examples of the use of such a linear Gaussian modelling framework. Two very simple and standard cases are the sinusoidal model and the autoregressive (AR) model.

Example: Sinusoidal model. This is a model that can be used to derive a maximum likelihood or Bayesian version of the DFT for noisy signals. If we write a single sinusoid as the sum of sine and cosine components we have:

$$x_n = a\cos(\omega n) + b\sin(\omega n)$$

Thus we can form a second order ($P = 2$) linear model from this if we take:

$$\mathbf{G} = \begin{bmatrix} \mathbf{c}(\omega) & \mathbf{s}(\omega) \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} a \\ b \end{bmatrix}$$

where

$$\mathbf{c}(\omega) = [\cos(\omega), \cos(2\omega), \dots \cos(N\omega)]^T$$

and

$$\mathbf{s}(\omega) = [\sin(\omega), \sin(2\omega), \ldots \sin(N\omega)]^T$$

And similarly, if the model is composed of J sinusoids at different frequencies ω_j we have

$$x_n = \sum_j a_j \cos(\omega_j n) + b_j \sin(\omega_j n)$$

and the linear model expression is

$$\mathbf{G} = \begin{bmatrix} \mathbf{c}(\omega_1) & \mathbf{s}(\omega_1) & \mathbf{c}(\omega_2) & \mathbf{s}(\omega_2) & \ldots & \mathbf{c}(\omega_J) & \mathbf{s}(\omega_J) \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \\ \vdots \\ a_J \\ b_J \end{bmatrix}$$

Thus we could make up a very complicated signal composed of lots of 'sinusoids' (with known frequencies) all added together. If we estimate the parameters $\boldsymbol{\theta}$ from some data then we will be doing a kind of probabilistic 'spectrum estimation' - this gets explored later in the project.

Example: Autoregressive (AR) model - this one was introduced at the end of my course in 3F3. The AR model is a standard time series model based on an all-pole filtered version of the noise residual:

$$x_n = \sum_{i=1}^{P} a_i x_{n-i} + e_n. \tag{3}$$

The coefficients $\{a_i;\ i = 1...P\}$ are the filter coefficients of the all-pole filter, henceforth referred to as the AR parameters, and P, the number of coefficients, is the order of the AR process. The AR model formulation is closely related to the linear prediction framework used in many fields of signal processing such as speech synthesis and coding (see e.g. [18, 13]). The AR modelling equation of (41) is now rewritten for the block of N data samples as

$$\mathbf{x} = \mathbf{G}\ \mathbf{a}\ +\ \mathbf{e} \tag{4}$$

where $\mathbf{e}$ is the vector of $(N - P)$ error values and the $((N - P) \times P)$ matrix $\mathbf{G}$ is given by

$$\mathbf{G} = \begin{bmatrix} x_P & x_{P-1} & \cdots & x_2 & x_1 \\ x_{P+1} & x_P & \cdots & x_3 & x_2 \\ \vdots & & \ddots & & \vdots \\ x_{N-2} & x_{N-3} & \cdots & x_{N-P} & x_{N-P-1} \\ x_{N-1} & x_{N-2} & \cdots & x_{N-P+1} & x_{N-P} \end{bmatrix} \tag{5}$$

Note that the data vector is comprised of just the last $N - P$ data points in order to make the equation work, i.e. $\mathbf{x} = [x_{P+1}\, x_{P+2}, ..., x_N$.

Notice also that this is a very special version of the linear model in which the matrix $\mathbf{G}$ is not fixed, but is actually made up of observed data points, owing to the feedback all-pole structure of the filter involved.

4.5 Maximum likelihood (ML) estimation

We first present the maximum likelihood (ML) estimator, which treats the parameters as unknown constants about which we incorporate no prior information. The observed data $\mathbf{x}$ is, however, considered random and we can often then obtain the PDFfor $\mathbf{x}$ when the value of $\boldsymbol{\theta}$ is known. This PDFis termed the *likelihood* $L(\mathbf{x}; \boldsymbol{\theta})$, which is defined as

$$L(\mathbf{x}; \boldsymbol{\theta}) = p(\mathbf{x} \mid \boldsymbol{\theta}) \tag{6}$$

The likelihood is of course implicitly conditioned upon all of our modelling assumptions $\mathcal{M}$, (e.g. the structure of $\mathbf{G}$, the number of model coefficients P, the noise distribution p_e, etc.) which could more properly be expressed as $p(\mathbf{x} \mid \boldsymbol{\theta}, \mathcal{M})$ - we will condsider model choice explicily later on, but for now just consider this as fixed and known.

The ML estimate for $\boldsymbol{\theta}$ is then that value of $\boldsymbol{\theta}$ which maximises the likelihood for given observations $\mathbf{x}$:

$$\boldsymbol{\theta}^{\mathrm{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\{p(\mathbf{x} \mid \boldsymbol{\theta})\} \tag{7}$$

Maximum likelihood (ML) estimator

The rationale behind this is that the ML solution corresponds to the parameter vector which would have generated the observed data $\mathbf{x}$ with highest probability. The maximisation task required for ML estimation can be achieved using standard differential calculus for well-behaved and differentiable likelihood functions, and it is often convenient analytically to maximise the log-likelihood function $l(\mathbf{x}; \boldsymbol{\theta}) = \log(L(\mathbf{x}; \boldsymbol{\theta}))$ rather than $L(\mathbf{x}; \boldsymbol{\theta})$ itself. Since log is a monotonically increasing function the two solutions are identical.

In data analysis and signal processing applications the likelihood function is arrived at through knowledge of the stochastic model for the data. For example, in the case of the Gaussian model (1) the likelihood can be obtained easily if we know the form of $p(\mathbf{e})$, the joint PDFfor the components of the error vector. The likelihood $p(\mathbf{x} \mid \boldsymbol{\theta})$ is then found from a transformation of variables $\mathbf{e} \rightarrow \mathbf{x}$ where $\mathbf{e} = \mathbf{x} - \mathbf{G}\boldsymbol{\theta}$. The Jacobian for this transformation is unity, so the likelihood is:

$$L(\mathbf{x}; \boldsymbol{\theta}) = p(\mathbf{x} \mid \boldsymbol{\theta}) = p_{\mathbf{e}}(\mathbf{x} - \mathbf{G}\boldsymbol{\theta}) \tag{8}$$

The ML solution may of course be posed for any error distribution, including highly non-Gaussian and non-independent cases. Here we give the most straightforward case, for the linear model where the elements of the error vector $\mathbf{e}$ are assumed to be i.i.d.and Gaussian and zero mean. If the variance of the Gaussian is σ_e^2 then we have:

$$p_{\mathbf{e}}(\mathbf{e}) = \frac{1}{(2\pi\sigma_e^2)^{N/2}} \exp\left(-\frac{1}{2\sigma_e^2}\mathbf{e}^T\mathbf{e}\right)$$

The likelihood is then

$$L(\mathbf{x};\boldsymbol{\theta}) = p(\mathbf{x} \mid \boldsymbol{\theta}) = p_{\mathbf{e}}(\mathbf{x} - \mathbf{G}\boldsymbol{\theta}) = \frac{1}{(2\pi\sigma_e^2)^{N/2}} \exp\left(-\frac{1}{2\sigma_e^2}(\mathbf{x} - \mathbf{G}\boldsymbol{\theta})^T(\mathbf{x} - \mathbf{G}\boldsymbol{\theta})\right) \tag{9}$$

which leads to the following log-likelihood expression:

$$\begin{aligned} l(\mathbf{x};\boldsymbol{\theta}) &= -(N/2)\log(2\pi\sigma_e^2) \; - \; \frac{1}{2\sigma_e^2}(\mathbf{x} - \mathbf{G}\boldsymbol{\theta})^T(\mathbf{x} - \mathbf{G}\boldsymbol{\theta}) \\ &= -(N/2)\log(2\pi\sigma_e^2) \; - \; \frac{1}{2\sigma_e^2}\sum_{n=1}^{N}(x_n - \mathbf{g}_n^T\boldsymbol{\theta})^2 \end{aligned}$$

Maximisation of this function w.r.t. $\boldsymbol{\theta}$ is equivalent to minimising the sum-squared of the error sequence $E = \sum_{n=1}^{N}(x_n - \mathbf{g}_n^T\boldsymbol{\theta})^2$. This is exactly the criterion which is applied in the familiar *least squares* (LS) estimation method. The ML estimator is obtained for example by taking derivatives w.r.t. $\boldsymbol{\theta}$ and equating to zero:

$$\boldsymbol{\theta}^{\text{ML}} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\,\mathbf{x} \tag{10}$$

Maximum likelihood for the Linear Gaussian model

which is, as expected, the familiar linear least squares estimate for model parameters calculated from finite length data observations. Thus we see that the ML estimator under the i.i.d.Gaussian error assumption is exactly equivalent to the well known least squares (LS) parameter estimator.

4.6 Bayesian Inference

The ML methods treat parameters as unknown constants. If we are prepared to treat parameters as random variables it is possible to assign prior PDFs to the parameters. These PDFs should ideally express some prior knowledge about the relative probability of different parameter values *before the data are observed.* Of course if nothing is known *a priori* about the parameters then the prior

distributions should in some sense express no initial preference for one set of parameters over any other. Note that in many cases a prior density is chosen to express some highly qualitative prior knowledge about the parameters. In such cases the prior chosen will be more a reflection of a *degree of belief* concerning parameter values than any true modelling of an underlying random process which might have generated those parameters. This willingness to assign priors which reflect subjective information is a powerful feature and also one of the most fundamental differences between the Bayesian and 'classical' inferential procedures. For various expositions of the Bayesian methodology and philosophy see, for example [11, 7, 3, 17]. The precise form of probability distributions assigned *a priori* to the parameters requires careful consideration since misleading results can be obtained from erroneous priors, but in principle at least we can apply the Bayesian approach to any problem where statistical uncertainty is present.

Bayes' Theorem is now stated as applied to estimation of random parameters $\boldsymbol{\theta}$ from a random vector $\mathbf{x}$ of observations, known as the posterior or *a posteriori* probability for the parameter:

$$p(\boldsymbol{\theta} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \boldsymbol{\theta})\ p(\boldsymbol{\theta})}{p(\mathbf{x})} \tag{11}$$

Posterior probability

Note that all of the distributions in this expression are implicitly conditioned upon all prior modelling assumptions, as was the likelihood function earlier. The distribution $p(\mathbf{x} \mid \boldsymbol{\theta})$ is the likelihood as used for ML estimation, while $p(\boldsymbol{\theta})$ is the prior or *a priori* distribution for the parameters. This term is one of the critical differences between Bayesian and 'classical' techniques. It expresses in an objective fashion the probability of various model parameters values *before* the data $\mathbf{x}$ has been observed. As we have already observed, the prior density may be an expression of highly subjective information about parameter values. This transformation from the subjective domain to an objective form for the prior can clearly be of great significance and should be considered carefully when setting up an inference problem.

The term $p(\boldsymbol{\theta} \mid \mathbf{x})$, the posterior or *a posteriori* distribution, expresses the probability of $\boldsymbol{\theta}$ given the observed data $\mathbf{x}$. This is now a true measure of how 'probable' a particular value of $\boldsymbol{\theta}$ is, given the observations $\mathbf{x}$. $p(\boldsymbol{\theta} \mid \mathbf{x})$ is in a more intuitive form for parameter estimation than the likelihood, which expresses how probable the *observations* are given the *parameters*. The generation of the posterior distribution from the prior distribution when data $\mathbf{x}$ is observed can be thought of as a refinement to any previous ('prior') knowledge about the parameters. Before $\mathbf{x}$ is observed $p(\boldsymbol{\theta})$ expresses any information previously obtained concerning $\boldsymbol{\theta}$. Any new information concerning the parameters contained in $\mathbf{x}$ is then incorporated to give the posterior distribution. Clearly if we

start off with little or no information about $\boldsymbol{\theta}$ then the posterior distribution is likely to obtain information obtained almost solely from $\mathbf{x}$. Conversely, if $p(\boldsymbol{\theta})$ expresses a significant amount of information about $\boldsymbol{\theta}$ then $\mathbf{x}$ will contribute less new information to the posterior distribution.

The denominator $p(\mathbf{x})$, referred to as the marginal likelihood, or the 'evidence' in machine learning circles, is a fundamentally useful quantity in model selection problems (see later), and is constant for any given observation $\mathbf{x}$; thus it may be ignored if we are only interested in the relative posterior probabilities of different parameters. As a result of this, Bayes' theorem is often stated in the form:

$$p(\boldsymbol{\theta} \mid \mathbf{x}) \propto p(\mathbf{x} \mid \boldsymbol{\theta})\ p(\boldsymbol{\theta}) \tag{12}$$

Posterior probability (proportionality)

$p(\mathbf{x})$ may be calculated in principle by integration:

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})\ d\boldsymbol{\theta} \tag{13}$$

and this effectively serves as the normalising constant for the posterior density (in this and subsequent results the integration would be replaced by a summation in the case of a discrete random vector $\boldsymbol{\theta}$).

It is worth reiterating at this stage that we are here implicitly conditioning in this framework on many pieces of additional prior information beyond just the prior parameters of the model $\boldsymbol{\theta}$. For example, we are assuming a precise form for the data generation process in the model; if the linear Gaussian model is assumed, then the whole data generation process *must* follow the probability law of that model, otherwise we cannot guarantee the quality of our answers; the same argument applies to ML estimation, although in the Bayesian setting the distributional form of the Bayesian prior must be assumed in addition. Some texts therefore adopt a notation $p(\boldsymbol{\theta} \mid \mathbf{x}, \mathcal{M})$ for Bayesian posterior distributions, where $\mathcal{M}$ denotes all of the additional modelling and distributional assumptions that are being made. We will omit this convention for the sake of notational simplicity. However, it will be partially reintroduced when we augment a model with a model index $\mathcal{M}_i$, when we are considering several competing models (and asssociated prior distributions) for explaining the data. In these cases, for example, the notation will be naturally extended as required, e.g. $p(\boldsymbol{\theta} \mid \mathbf{x}, \mathcal{M}_i)$ in the model selection case.

4.6.1 Posterior inference and Bayesian cost functions

The posterior distribution gives the probability for any chosen $\boldsymbol{\theta}$ given observed data $\mathbf{x}$, and as such optimally combines our prior information about $\boldsymbol{\theta}$ and any

additional information gained about $\boldsymbol{\theta}$ from observing $\mathbf{x}$. We may in principle manipulate the posterior density to infer any required statistic of $\boldsymbol{\theta}$ conditional upon $\mathbf{x}$. This is a significant advantage over ML and least squares methods which strictly give us only a single estimate of $\boldsymbol{\theta}$, known as a 'point estimate'. However, by producing a posterior PDFwith values defined for all $\boldsymbol{\theta}$ the Bayesian approach gives a fully interpretable probability distribution. In principle this is as much as one could ever need to know about the inference problem. In signal processing problems, however, we usually require a single point estimate for $\boldsymbol{\theta}$, and a suitable way to choose this is via a 'cost function' $C(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta})$ which expresses objectively a measure of the cost associated with a particular parameter estimate $\hat{\boldsymbol{\theta}}$ when the true parameter is $\boldsymbol{\theta}$ (see e.g. [8, 3, 17]). The form of cost function will depend on the requirements of a particular problem. A cost of 0 indicates that the estimate is perfect for our requirements (this does not necessarily imply that $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$, though it usually will) while positive values indicate poorer estimates. The simplest and most intuitive form of cost function is perhaps the so-called uniform cost function (details not worked through here), which leads to the maximum *a posteriori* (MAP) estimate, the value of $\hat{\boldsymbol{\theta}}$ which maximises the posterior distribution:

$$\boldsymbol{\theta}^{\mathrm{MAP}} = \underset{\boldsymbol{\theta}}{\mathrm{argmax}}\{p(\boldsymbol{\theta} \mid \mathbf{x})\} \tag{14}$$

Maximum a posteriori (MAP) estimator

In other words, just like with ML, we solve to find rhe parameters $\boldsymbol{\theta}$ which *maximise* the posterior probabiity.

We now work through the MAP estimation scheme under the Linear Gaussian model (1). Suppose that the prior on parameter vector $\boldsymbol{\theta}$ is the multivariate Gaussian (35):

$$p(\boldsymbol{\theta}) = \mathrm{N}(\mathbf{m}_{\boldsymbol{\theta}}, \mathbf{C}_{\boldsymbol{\theta}}) = \frac{1}{(2\pi)^{P/2} \mid \mathbf{C}_{\boldsymbol{\theta}} \mid^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \mathbf{m}_{\boldsymbol{\theta}})^T \mathbf{C}_{\boldsymbol{\theta}}{}^{-1}(\boldsymbol{\theta} - \mathbf{m}_{\boldsymbol{\theta}})\right) \tag{15}$$

where $\mathbf{m}_{\boldsymbol{\theta}}$ is the prior parameter mean vector, $\mathbf{C}_{\boldsymbol{\theta}}$ is the parameter covariance matrix and P is the number of parameters in $\boldsymbol{\theta}$. If the likelihood $p(\mathbf{x} \mid \boldsymbol{\theta})$ takes the same form as before (8), the posterior distribution is as follows:

$$\begin{aligned} p(\boldsymbol{\theta} \mid \mathbf{x}) \propto & \frac{1}{(2\pi)^{P/2} \mid \mathbf{C}_{\boldsymbol{\theta}} \mid^{1/2}} \frac{1}{(2\pi\sigma_e^2)^{N/2}} \\ & \exp\Bigg(-\frac{1}{2\sigma_e^2}(\mathbf{x} - \mathbf{G}\boldsymbol{\theta})^T(\mathbf{x} - \mathbf{G}\boldsymbol{\theta}) \\ & \qquad - \frac{1}{2}(\boldsymbol{\theta} - \mathbf{m}_{\boldsymbol{\theta}})^T \mathbf{C}_{\boldsymbol{\theta}}{}^{-1}(\boldsymbol{\theta} - \mathbf{m}_{\boldsymbol{\theta}})\Bigg) \end{aligned} \tag{16}$$

and the MAP estimate $\boldsymbol{\theta}^{\text{MAP}}$ is obtained by differentiation of the log-posterior and finding its unique maximiser as:

$$\boldsymbol{\theta}^{\text{MAP}} = \left(\mathbf{G}^T\mathbf{G} + \sigma_e^2\ \mathbf{C}_{\boldsymbol{\theta}}^{-1}\right)^{-1}\left(\mathbf{G}^T\mathbf{x} + \sigma_e^2\ \mathbf{C}_{\boldsymbol{\theta}}{}^{-1}\mathbf{m}_{\boldsymbol{\theta}}\right) \tag{17}$$

MAP estimator - Linear Gaussian model

In this expression we can clearly see the 'regularising' effect of the prior density on the ML estimate of (9). As the prior becomes more 'diffuse', i.e. the diagonal elements of $\mathbf{C}_{\boldsymbol{\theta}}$ increase both in magnitude and relative to the off-diagonal elements, we impose 'less' prior information on the estimate. In the limit the prior tends to a uniform ('flat') prior with all $\boldsymbol{\theta}$ equally probable. In this limit $\mathbf{C}_{\boldsymbol{\theta}}^{-1} = 0$ and the estimate is identical to the ML estimate (9). This useful relationship demonstrates that the ML estimate may be interpreted as the MAP estimate with a uniform prior assigned to $\boldsymbol{\theta}$. The MAP estimate will also tend towards the ML estimate when the likelihood is strongly 'peaked' around its maximum compared with the prior. Once again the prior will then have little influence on the shape of the posterior density. It is in fact well known [11] that as the sample size N tends to infinity the Bayes solution tends to the ML solution. This of course says nothing about small sample (i.e. small N) parameter estimates where the effect of the prior may be very significant.

The choice of a multivariate Gaussian prior may well be motivated by physical considerations about the problem, or it may be motivated by subjective prior knowledge about the value of $\boldsymbol{\theta}$ (before the data $\mathbf{x}$ are seen!) in terms of a rough value $\mathbf{m}_{\boldsymbol{\theta}}$ and a confidence in that value through the covariance matrix $\mathbf{C}_{\boldsymbol{\theta}}$ (a 'subjective' prior). In fact the choice of Gaussian also has the very special property that it makes the Bayesian calculations straightforward and available in closed form. Such a prior is known as a 'conjugate' prior [3].

4.6.2 Posterior distribution for parameters in the Linear Gaussian model

Reconsidering the form of (15), we can obtain a lot more information than simply the MAP estimate of (16). A fully Bayesian analysis of the problem will study the whole posterior distribution for the unknowns, computing measures of uncertainty, confidence intervals, and so on. The required distribution can be obtained by rearranging the exponent of (15) using the result of (38),

$$\begin{aligned}&\frac{1}{\sigma_e^2}(\mathbf{x}-\mathbf{G}\boldsymbol{\theta})^T(\mathbf{x}-\mathbf{G}\boldsymbol{\theta}) + (\boldsymbol{\theta}-\mathbf{m}_{\boldsymbol{\theta}})^T\mathbf{C}_{\boldsymbol{\theta}}{}^{-1}(\boldsymbol{\theta}-\mathbf{m}_{\boldsymbol{\theta}})\\ &\quad= \frac{1}{\sigma_e^2}\left((\boldsymbol{\theta}-\boldsymbol{\theta}^{\text{MAP}})^T\boldsymbol{\Phi}(\boldsymbol{\theta}-\boldsymbol{\theta}^{\text{MAP}}) + \mathbf{x}^T\mathbf{x} + \sigma_e^2\mathbf{m}_{\boldsymbol{\theta}}{}^T\mathbf{C}_{\boldsymbol{\theta}}{}^{-1}\mathbf{m}_{\boldsymbol{\theta}} - \boldsymbol{\Theta}^T\boldsymbol{\theta}^{\text{MAP}}\right)\end{aligned} \tag{18}$$

with terms defined as

$$\boldsymbol{\theta}^{\text{MAP}} = \boldsymbol{\Phi}^{-1}\boldsymbol{\Theta} \tag{19}$$

$$\boldsymbol{\Phi} = \mathbf{G}^T\mathbf{G} + \sigma_e^2 {\mathbf{C}_{\boldsymbol{\theta}}}^{-1} \tag{20}$$

$$\boldsymbol{\Theta} = \mathbf{G}^T\mathbf{x} + \sigma_e^2 {\mathbf{C}_{\boldsymbol{\theta}}}^{-1}\mathbf{m}_{\boldsymbol{\theta}} \tag{21}$$

Now we can observe that the first term in (17), $\frac{1}{\sigma_e^2}\Big((\boldsymbol{\theta}-\boldsymbol{\theta}^{\text{MAP}})^T\boldsymbol{\Phi}(\boldsymbol{\theta}-\boldsymbol{\theta}^{\text{MAP}})$, is in exactly the correct form for the exponent of a multivariate Gaussian, see (35), with mean vector and covariance matrix as follows,

$$\mathbf{m}_{\boldsymbol{\theta}}^{\text{post}} = \boldsymbol{\theta}^{\text{MAP}} \quad \text{and } \mathbf{C}_{\boldsymbol{\theta}}^{\text{post}} = \sigma_e^2\boldsymbol{\Phi}^{-1}.$$

Since the remaining terms in (15) do not depend on $\boldsymbol{\theta}$, and we know that the multivariate density function must be proper (i.e. integrate to 1), we can conclude that the posterior distribution is itself a multivariate Gaussian,

$$p(\boldsymbol{\theta} \mid \mathbf{x}) = \text{N}(\boldsymbol{\theta}^{\text{MAP}}, \sigma_e^2\boldsymbol{\Phi}^{-1}). \tag{22}$$

This result will be fundamental for the construction of inference methods in later sections.

4.6.3 Marginal Likelihood

In problems where model choice or classification are of importance, and in inference algorithms which marginalise ('Rao-Blackwellise') a Gaussian parameter (not needed for this project) it will be required to compute the marginal likelihood for the data, i.e.

$$p(\mathbf{x}) = \int_{\boldsymbol{\theta}} p(\mathbf{x} \mid \boldsymbol{\theta})\; p(\boldsymbol{\theta})\; d\boldsymbol{\theta} \tag{23}$$

Now, filling in the required density functions for the Linear Gaussian case, we have:

$$\begin{aligned} p(\mathbf{x}) = \int_{\boldsymbol{\theta}} & \frac{1}{(2\pi)^{P/2} \mid \mathbf{C}_{\boldsymbol{\theta}} \mid^{1/2}} \frac{1}{(2\pi\sigma_e^2)^{N/2}} \\ & \exp\left(-\frac{1}{2\sigma_e^2}(\mathbf{x}-\mathbf{G}\boldsymbol{\theta})^T(\mathbf{x}-\mathbf{G}\boldsymbol{\theta})\right. \\ & \left. -\frac{1}{2}(\boldsymbol{\theta}-\mathbf{m}_{\boldsymbol{\theta}})^T{\mathbf{C}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{\theta}-\mathbf{m}_{\boldsymbol{\theta}})\right)\; d\boldsymbol{\theta} \end{aligned} \tag{24}$$

This multivariate Gaussian integral can be performed after some rearrangement using result (40) to give:

$$\begin{aligned} p(\mathbf{x}) = & \frac{1}{(2\pi)^{P/2} \mid \mathbf{C}_{\boldsymbol{\theta}} \mid^{1/2} \quad \mid \boldsymbol{\Phi} \mid^{1/2} (2\pi\sigma_e^2)^{(N-P)/2}} \\ & \exp\left(-\frac{1}{2\sigma_e^2}(\mathbf{x}^T\mathbf{x} + \sigma_e^2{\mathbf{m}_{\boldsymbol{\theta}}}^T{\mathbf{C}_{\boldsymbol{\theta}}}^{-1}\mathbf{m}_{\boldsymbol{\theta}} - \boldsymbol{\Theta}^T\boldsymbol{\theta}^{\text{MAP}})\right) \end{aligned} \tag{25}$$

with terms defined exactly as for the posterior distribution calculation above in (18)-(20).

4.7 Model Uncertainty and Bayesian Decision theory

In many problems of practical interest there are also issues of model choice and model uncertainty involved. For example, how can one choose the number of basis functions P in the linear model (1)? This could amount to a choice of how many sinusoids are necessary to model a given signal, or how many coefficients are required in an autoregressive model formulation. These questions should be answered automatically from the data and can as before include any known prior information about the models. There are a number of frequentist approaches to model choice, typically involving computation of the maximum likelihood parameter vector in each possible model order and then penalising the likelihood function to prevent over-fitting by inappropriately high model orders. Such standard techniques include the AIC, MDL and BIC procedures [1], which are typically based on asymptotic and information-theoretic considerations. In the fully Bayesian approach we aim to determine directly the posterior probability for each candidate model. An implicit assumption of the approach is that the true model which generated the data lies within the set of possible candidates; this is clearly an idealisation for most real-world problems.

As for Bayesian parameter estimation, we consider the unobserved variable (in this case the model $\mathcal{M}_i$) as being generated by some random process whose prior probabilities are known. These prior probabilities are assigned to each of the possible model states using a probability mass function (PMF) $p(\mathcal{M}_i)$, which expresses the prior probability of occurrence of different states given all information available except the data $\mathbf{x}$. The required form of Bayes' theorem for this discrete estimation problem is then

$$p(\mathcal{M}_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathcal{M}_i)\ p(\mathcal{M}_i)}{p(\mathbf{x})} \tag{26}$$

$p(\mathbf{x})$ is constant for any given $\mathbf{x}$ and will serve to normalise the posterior probabilities over all i in the same way that the marginal likelihood, or 'evidence', normalised the posterior parameter distribution (10). In the same way that Bayes rule gave a posterior distribution for parameters $\boldsymbol{\theta}$, this expression gives the posterior probability for a particular model given the observed data $\mathbf{x}$. It would seem reasonable to choose the model $\mathcal{M}_i$ corresponding to maximum posterior probability as our estimate for the true state (we will refer to this state estimate as the MAP estimate), and this can be shown to have the desirable property of minimum classification error rate P_E (see e.g. [8]), that is, it has minimum probability of choosing the wrong model. Note that determination of the MAP model estimate will usually involve an exhaustive search of $p(\mathcal{M}_i \mid \mathbf{x})$ for all feasible i.

These ideas are formalised by consideration of a 'loss function' $\lambda(\alpha_i \mid \mathcal{M}_j)$ which defines the penalty incurred by taking action α_i when the true state is $\mathcal{M}_j$. Action α_i will usually refer to the action of choosing model $\mathcal{M}_i$ as the estimate - we do not pursue this further as it is not required for the project.

4.7.1 Calculation of the marginal likelihood, $p(\mathbf{x} \mid \mathcal{M}_i)$

The term $p(\mathbf{x} \mid \mathcal{M}_i)$ is equivalent to the marginal likelihood term $p(\mathbf{x})$ which was encountered in the parameter estimation section, since $p(\mathbf{x})$ was implicitly conditioned on a particular model structure in that scheme.

If one uses a uniform model prior $p(\mathcal{M}_i) = \frac{1}{N_{\mathcal{M}}}$, then, according to equation (25), it is only necessary to compare values of $p(\mathbf{x} \mid \mathcal{M}_i)$ for model selection since the remaining terms are constant for all models. $p(\mathbf{x} \mid \mathcal{M}_i)$ can then be viewed literally as the relative 'evidence' for a particular model, and two candidate models can be compared through their Bayes Factor:

$$BF_{ij} = \frac{p(\mathbf{x} \mid \mathcal{M}_i)}{p(\mathbf{x} \mid \mathcal{M}_j)}$$

Typically each model or state $\mathcal{M}_i$ will be expressed in a parametric form whose parameters $\boldsymbol{\theta}_i$ are unknown. As for the parameter estimation case it will usually be possible to obtain the state conditional parameter likelihood $p(\mathbf{x} \mid \boldsymbol{\theta}_i, \mathcal{M}_i)$. Given a model-dependent prior distribution for $\boldsymbol{\theta}_i$ the marginal likelihood may be obtained by integration to eliminate $\boldsymbol{\theta}_i$ from the joint probability $p(\mathbf{x}, \boldsymbol{\theta}_i \mid \mathcal{M}_i)$. The marginal likelihood is then obtained as

$$p(\mathbf{x} \mid \mathcal{M}_i) = \int_{\boldsymbol{\theta}_i} p(\mathbf{x} \mid \boldsymbol{\theta}_i, \mathcal{M}_i)\ p(\boldsymbol{\theta}_i \mid \mathcal{M}_i)\ d\boldsymbol{\theta}_i \tag{27}$$

If the Linear Gaussian model of (1) is extended to the multi-model scenario we obtain:

$$\mathbf{x} = \mathbf{G}_i\boldsymbol{\theta}_i \ + \ \mathbf{e}_i \tag{28}$$

where $\mathbf{G}_i$ refers to the state-dependent basis matrix and $\mathbf{e}_i$ is the corresponding error sequence. For this model the state dependent parameter likelihood $p(\mathbf{x} \mid \boldsymbol{\theta}_i, \mathcal{M}_i)$ is (see (8)):

$$p(\mathbf{x} \mid \boldsymbol{\theta}_i, \mathcal{M}_i) = \frac{1}{(2\pi\sigma_{e_i}^2)^{N/2}} \exp\left(-\frac{1}{2\sigma_{e_i}^2}(\mathbf{x} - \mathbf{G}_i\boldsymbol{\theta}_i)^T(\mathbf{x} - \mathbf{G}_i\boldsymbol{\theta}_i)\right) \tag{29}$$

If we for example take the same Gaussian form for the state conditional parameter prior $p(\boldsymbol{\theta}_i \mid \mathcal{M}_i)$ (with P_i parameters) as we used for $p(\boldsymbol{\theta})$ in (14) the marginal likelihood is then given with minor modification as:

$$p(\mathbf{x} \mid \mathcal{M}_i) = \frac{1}{(2\pi)^{P_i/2} \mid \mathbf{C}_{\boldsymbol{\theta}_i} \mid^{1/2} \ \mid \boldsymbol{\Phi} \mid^{1/2} (2\pi\sigma_{e_i}^2)^{(N-P_i)/2}}$$
$$\exp\left(-\frac{1}{2\sigma_{e_i}^2}(\mathbf{x}^T\mathbf{x} + \sigma_{e_i}^2 \mathbf{m}_{\boldsymbol{\theta}_\mathbf{i}}{}^T \mathbf{C}_{\boldsymbol{\theta}_i}{}^{-1} \mathbf{m}_{\boldsymbol{\theta}_\mathbf{i}} - \boldsymbol{\Theta}^T\boldsymbol{\theta}_i^{\text{MAP}})\right) \tag{30}$$

with terms defined as

$$\boldsymbol{\theta}_i^{\text{MAP}} = \boldsymbol{\Phi}^{-1}\boldsymbol{\Theta} \tag{31}$$

$$\boldsymbol{\Phi} = \mathbf{G}_i{}^T\mathbf{G}_i + \sigma_{e_i}^2 \mathbf{C}_{\boldsymbol{\theta}_i}{}^{-1} \tag{32}$$

$$\boldsymbol{\Theta} = \mathbf{G}_i{}^T\mathbf{x} + \sigma_{e_i}^2 \mathbf{C}_{\boldsymbol{\theta}_i}{}^{-1}\mathbf{m}_{\boldsymbol{\theta}_i} \tag{33}$$

Notice that $\boldsymbol{\theta}_i^{\text{MAP}}$ is simply the model-dependent version of the MAP parameter estimate given by (16).

4.8 Structures for model uncertainty

Model uncertainty may often be expressed in a highly structured way: in particular the models may be *nested* or *subset* structures. In the nested structure for the linear model, the basis matrix for model $\mathcal{M}_{i+1}$ is obtained by adding an additional basis vector to the $\mathbf{G}_i$:

$$\mathbf{G}_{i+1} = \left[\mathbf{G}_i \mathbf{h}_{i+1}\right]$$

and hence the model of higher order inherits part of its structure from the lower order models. This would be the case, for example, in the AR modelling example given earlier.

5 Experiments to carry out

In the experiments you will try out the basics of ML and Bayesian estimation, and then move onto applying them to some real and synthetic data sets.

1. Consider perhaps the simplest case of the linear model, just an unknown level plus noise:

   $$y_n = \theta + e_n$$

   where θ is a constant but unknown parameter for all n.

   Write code to generate a vector of observations $\mathbf{y}$ from this model.

   Now, formulate both the ML estimate, the Bayesian posterior density and Bayesian estimates for θ, using the same Gaussian assumptions as in the earlier text. Give the resulting derivation and equations in your final report.

   What happens to the Bayesian method compared to the ML method as the prior gets more and more 'spread out', i.e. its variance parameter gets larger and larger so it looks like a flat ('uniform') distribution?

   Code up the ML and Bayesian methods in Matlab.

   Explore the properties of the methods. Consider as a function of N, including the case $N = 1$, just one data point, and as N gets large. Plot likelihood functions and posterior densities as functions of θ, giving examples of different prior parameters and noise variances σ_e^2, showing how the prior may or may not influence the Bayesian solution, and showing what happens to ML and Bayes for small, medium and large N.

2. Now add a linear trend term to the model:

   $$y_n = \theta_1 + \theta_2 n + e_n$$

Again, write code to generate data from the model.

Code up the general linear model ML and Bayesian estimators with Gaussian priors and likelihoods for this model, as in the main text - probably at this stage best to do it for a general value of P ($P = 2$ in this question) so you can re-use it for later questions.

Give some examples of generated data sets and the corresponding ML/ Bayesian estimates of trend/ DC level.

3. Now, consider Bayesian model choice, choosing between three models:

$$\mathcal{M}_1 : y_n = e_n$$
$$\mathcal{M}_2 : y_n = \theta_1 + e_n$$
$$\mathcal{M}_3 : y_n = \theta_1 + \theta_2 n + e_n$$

Formulate the Bayesian posterior model probabilties for these models and code up in Matlab.

Show some examples of model choice and parameter estimation for datasets you have generated from each of the model classes. It would be sensible to have a simple zero mean independent Gaussian prior for the θ terms, each with constant variance σ_θ^2.

Since it might be hard to specify the variance σ_θ^2 in advance for a real problem, it is tempting to use a flat prior, i.e. set σ_θ very large or even infinite. Repeat your experiments with increasingly large σ_θ and see what happens to the model probabilities. Can you explain the effect? This is an example of a paradox sometimes termed 'Lindley's paradox', and is one weakness of a fully Bayesian model choice method [there are ways around it using hyperparameters and hierarchical models, but we do not consider them here].

4. Challenge dataset and modelling problem - finding a Bayesian needle in a haystack.

A piece of Matlab code was used to hide a signal at an unknown time location and buried in noise. The code that generated the data is given below:

```
N=100;

sigma_n=2;

sigma_theta=1;

signal=[-3 5 -2 4 1 3 5 -1 2 4 6 5 -2 -2 1];

noise=sigma_n*randn(N,1);
```

```
theta=sigma_theta*randn(1);

offset=round(90*rand(1));

y=noise;

signal_offset=0*noise;
signal_offset(offset:offset+14)=signal'*theta;


y=y+signal_offset;
```

This code is available for downloading on the Moodle SF1 site, along with the dataset it generated,

```
gen_hidden_data.m
hidden_data.mat
```

Use Bayesian model choice and parameter estimation to determine the most probable location of the signal, i.e. the value of the 'parameter' offset and its corresponding parameter theta. Determine and plot the posterior probability for the offset random variable and for the theta corresonding to the most probable offset. Determine also the probability of the 'null' hypothesis that there is no hidden signal at all, equivalent to theta=0 for any offset value. Test the sensitivity of your analysis to the setting of the prior parameters in the model.

Hint: each possible offset value can be considered as a different Bayesian linear 'model', with its own parameter theta. The code gives you the parameters of the actual prior distribution which generated the data.

Challenge: Finally, can you establish a link in the maths between your detector and another method you have already studied in 3F3 for detecting a signal in noise?

This material should all be completed around the end of week 3. For week 4 some longer mini-projects are to be carried out:

5. The Autoregressive (AR) model.

   Code up the AR model and generate some example datasets from, say, the 2nd order ($P = 2$) case. Take a look at the magnitude squared of the DFT of the data and check whether it agrees with your expectations for the power spectrum of such a process (see 3F3). Remember you need to check the pole positions of the filter for stability. What does the data look like when the poles are unstable?

Code up the linear model for parameter estimation, both ML and Bayesian *with Gaussian prior*, for AR models of a specified order P. Experiment with priors and see *the effects of the Gaussian prior compared with the ML estimate.*

Examine the mean sum squared residual error, i.e. $1/N \sum_{n=1}^{N} e_n^2$ for different model orders. *Note that e_n needs to ba carefully calculated using your estimated AR parameters and Eq. (??), and not using data synthesized from your model.* Do you see any interesting behaviour around the true model order? You should try, say, order 10 or higher to see these effects properly.

Now code up and perform Bayesian model choice for the AR model. Examine the effect of the parameter prior distribution on the model order estimates and compare estimates with the plots of residual error from the previous parts.

Apply the model selection method to short segments of real audio, trying vowels, consonants, steady notes, transients, as before, comparing and contrasting the selected model orders and the parameter estimates obtained (plot the power spectrum of the estimated process). Note that for real audio you will need to specify the residual variance σ_e^2, which will be different for each extract. This can also be estimated in a Bayesian/ ML scheme, but we will not consider the details here. One simple possibility for the project would be to iteratively estimate σ_e^2 from the current estimate of the residuals $\{e_n\}$ and then re-estimate the AR parameters - experiment and see what happens. Would such a procedure have any effect on the ML parameter estimate?

Now, use your AR models and corresponding parameter estimates to interpolate missing packets of data in audio signals. We can define missing packets by simply zeroing out some of the data points and assuming they were lost during digital transmission:

$$y_n = \begin{cases} x_n, & \text{received OK} \\ 0, & \text{otherwise} \end{cases}$$

Some example datasets with missing packets are provided on the Moodle site.

The simplest way to do this is to run the AR model in 'prediction' mode from the start of the gap to the end using the forward prediction formula with/ without residual noise:

$$\hat{x}_n^f = \sum_{i=1}^{P} a_i \hat{x}_{n-i}^f + e_n^f.$$

In order to get continuity at the end of the missing packet you can combine

this with the reverse-time ('anti-causal') prediction:

$$\hat{x}_n^b = \sum_{i=1}^{P} a_i \hat{x}_{n+i}^b + e_n^b.$$

using a weighted sum of the two predictions,

$$\hat{x}_n = \alpha \hat{x}_n^f + (1-\alpha)\hat{x}_n^b$$

where α is a coefficient which would smoothly vary from 1 to 0 as you move across the missing data.

Experiment with your combinations of model choice, parameter estimation and interpolation to see if you can find a general technique that will sound good for a long segment of audio with many missing packets (try, say missing chunks of 100 data points out of each run of 1000 transmitted data points). How can you estimate the AR model when some of the data is lost? Measure the mean-squared error in interpolation as a function of the % of missing data, the length of missing data, whether you include noise $e_n^{f/b}$ in your interpolations, etc. Of course, as for the noise reducers, the final judgement of quality will be on the sound quality of the reconstructed audio.

Challenge: The simple forward/backward interpolator above is not firmly grounded in any estimation theory. A more sophisticated scheme which is more satisfactory carries out a ML/ Bayesian interpolation according to the formula

$$\mathbf{x}_{(\mathrm{i})}^{\mathrm{LS}} = -(\mathbf{A}_{(\mathrm{i})}{}^T \mathbf{A}_{(\mathrm{i})})^{-1} \mathbf{A}_{(\mathrm{i})}{}^T \mathbf{A}_{-(\mathrm{i})}\ \mathbf{x}_{-(\mathrm{i})} \tag{34}$$

where the terms are explained in the appendix. You could also code up this method if you want a challenge/ have time spare, and include it in your experiments/evaluations above.

6. Challenge if time permits: The Bayesian DFT.

   Consider the example in the text, the Sinusoidal Model. Derive and Code up the ML and Bayesian methods for J sinuoids, assuming independent zero mean Gaussian priors for the parameters.

   For $J = 1$, i.e. just one sinusoid, can you spot in the mathematics a close link between the ML method and the DFT magnitude, i.e. $|X_p|$? Hint: consider the formula for the ML estimate for the parameters of the sinusoid, a and b, evaluated on a DFT bin frequency, and simplify the expression for the log-likelihood at this optimum. You will need to consider the special form of the matrix $\mathbf{G}^T\mathbf{G}$ and its inverse in this case. You should now be able to see a close link between the DFT at this frequency and the log-likelihood. Remember that terms which do not depend on ω are unimportant in any optimisation task with respect to ω.

   Apply the methods to (a segment of) the challenge audio file organ.wav. Try it first for a single sinusoid to check you get a reasonable amplitude

spectrum as ω is varied - include a fine grid of frequencies that are not at DFT bin frequencies so as to get the most accurate estimates of the peak locations. Now consider a multiple frequency model in which the sinusoids have frequencies set equal to the most significant frequencies just identified. Can you use the ML/Bayesian model to confirm/ refine your estimates of the musical pitches present in the extract and make a model-based reconstruction of the signal? It is quite informative to listen to the effects of adding in the frequency components one by one and hearing (hopefully) the sound gradually build up into a fully formed set of musical notes. Now explore the incorporation of Bayesian priors in your model. For example, it is well known that frequency components decay in amplitiude exponentially at higher frequencies - can this be used to 'regularise' your estimated frequency amplitudes in the Bayesian model?

Background:

For the Bayesian single frequency model, you are really computing a probability conditioned on the frequency value:

$$p(a, b|\mathbf{y}, \omega)$$

You could do Bayesian inference across frequency by evaluating and plotting the following *marginal* probaility:

$$p(\omega|\mathbf{y}) \propto p(\omega) \int p(y|a, b, \omega)p(a, b|\omega)dadb$$

where the integral required is essentially the same as the marginal likelihood for model selection in the linear Gaussian model, and $p(\omega)$ is a prior for the frequency, which could be uniform, or shaped to the likely spectral content of an audio signal.

If you need to estimate frequency *and* amplitude, you could do joint Bayesian inference about the probability of frequency *and* amplitudes by calculating:

$$p(a, b, \omega|\mathbf{y}) \propto p(a, b|\omega, \mathbf{y})p(\omega|\mathbf{y}).$$

The first term of this you already have in the Bayesian linear Gaussian model posterior. The second is just the marginal from above.

A Probability Densities and Integrals

A.1 Univariate Gaussian

The univariate Gaussian, or normal, density function with mean μ and variance σ^2 is defined for a real-valued random variable as:

$$\mathrm{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{35}$$

Univariate normal density

A.2 Multivariate Gaussian

The multivariate Gaussian probability density function (PDF) for a column vector $\mathbf{x}$ with N real-valued components is expressed in terms of the mean vector $\mathbf{m_x}$ and the covariance matrix $\mathbf{C_x} = E[(\mathbf{x}-\mathbf{m_x})(\mathbf{x}-\mathbf{m_x})^T]$ as:

$$\mathcal{N}_N(\mathbf{x}|\mathbf{m},\mathbf{C_x}) = \frac{1}{(2\pi)^{N/2} \mid \mathbf{C_x} \mid^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m_x})^T \mathbf{C_x}^{-1}(\mathbf{x}-\mathbf{m_x})\right) \tag{36}$$

Multivariate Gaussian density

An integral which is used on many occasions throughout the text is of the general form:

$$I = \int_{\mathbf{y}} \exp\left(-\frac{1}{2}\left(a + \mathbf{b}^T\mathbf{y} + \mathbf{y^T C\, y}\right)\right) d\mathbf{y} \tag{37}$$

where $d\mathbf{y}$ is interpreted as the infinitesimal volume element:

$$d\mathbf{y} = \prod_{i=1}^{N} dy_i$$

and the integral is over the real line in all dimensions, i.e. the single integration sign should be interpreted as:

$$\int_{\mathbf{y}} \equiv \int_{y_1=-\infty}^{\infty} \cdots \int_{y_N=-\infty}^{\infty}$$

For non-singular symmetric $\mathbf{C}$ it is possible to form a 'perfect square' for the exponent and hence simplify the integral. Take negative of twice the exponent:

$$a + \mathbf{b}^T\mathbf{y} + \mathbf{y^T C\, y} \tag{38}$$

and try to express it in the form:

$$(\mathbf{y}-\mathbf{m_y})^T\mathbf{C}\,(\mathbf{y}-\mathbf{m_y}) + k$$

for some constants k and $\mathbf{C}$, to be determined. Multiplying out this expression leads to the required form:

$$\mathbf{y}^T\mathbf{C}\,\mathbf{y} + \mathbf{m_y}^T\mathbf{C}\,\mathbf{m_y} - 2\mathbf{m_y}^T\mathbf{C}\,\mathbf{y} + k.$$

Here we have used the fact that $\mathbf{m}_\mathbf{y}^T\mathbf{C}\ \mathbf{y} = \mathbf{y}^T\mathbf{C}\ \mathbf{m}_\mathbf{y}$, since $\mathbf{C}$ is assumed symmetric.

Hence, equating the constant, linear and quadratic terms in $\mathbf{y}$, we arrive at the result:

$$a + \mathbf{b}^T\mathbf{y} + \mathbf{y}^\mathbf{T}\mathbf{C}\ \mathbf{y} = (\mathbf{y} - \mathbf{m}_\mathbf{y})^T\mathbf{C}\ (\mathbf{y} - \mathbf{m}_\mathbf{y}) + k \tag{39}$$

where

$$\mathbf{m}_\mathbf{y} = -\frac{\mathbf{C}^{-1}\mathbf{b}}{2}$$

and

$$k = (a\ -\ \frac{\mathbf{b}^\mathbf{T}\mathbf{C}^{-1}\mathbf{b}}{4})$$

Thus the integral I can be re-expressed as:

$$I = \int_\mathbf{y} \exp\left(-\frac{1}{2}\left((\mathbf{y} - \mathbf{m}_\mathbf{y})^T\mathbf{C}\ (\mathbf{y} - \mathbf{m}_\mathbf{y})\right)\right) \exp\left(-\frac{1}{2}\left(a\ -\ \frac{\mathbf{b}^\mathbf{T}\mathbf{C}^{-1}\mathbf{b}}{4}\right)\right) d\mathbf{y} \tag{40}$$

where, again

$$\mathbf{m}_\mathbf{y} = -\frac{\mathbf{C}^{-1}\mathbf{b}}{2}$$

Comparison with the multivariate PDFof 35 which has unity volume leads directly to the result:

$$\int_\mathbf{y} \exp\left(-\frac{1}{2}\left(a + \mathbf{b}^T\mathbf{y} + \mathbf{y}^\mathbf{T}\mathbf{C}\ \mathbf{y}\right)\right) d\mathbf{y} = \frac{(2\pi)^{N/2}}{|\ \mathbf{C}\ |^{1/2}} \exp\left(-\frac{1}{2}\left(a\ -\ \frac{\mathbf{b}^\mathbf{T}\mathbf{C}^{-1}\mathbf{b}}{4}\right)\right) \tag{41}$$

Multivariate Gaussian integral

This result can be also be obtained directly by a transformation which diagonalises $\mathbf{C}$ and this approach then verifies the normalisation constant given for the PDFof 35.

B LS/ML/ Bayesian interpolation details

B.1 Autoregressive Modelling

In previous sections we have developed the principles of block-based and recursive estimation within a Bayesian framework. We now introduce a particular time series model which is fundamental to much of the work in this dissertation. This is the autoregressive (AR) model in which the data is modelled as the output of an all-pole filter excited by white noise. This model formulation is a special case of the innovations representation for a stationary random signal in which the signal x_m is modelled as the output of a linear shift-invariant filter driven by white noise. In the AR case the filtering operation is restricted to a weighted sum of past output values and a white noise innovations input e_m:

$$x_m = \sum_{i=1}^{P} a_i x_{m-i} + e_m. \tag{42}$$

The coefficients a_i, $(i = 1...P)$ are the filter coefficients of the all-pole filter, henceforth referred to as the AR parameters, and P, the number of coefficients, is the order of the AR process. The AR model formulation is identical to the linear prediction framework used in many fields of signal processing (see e.g. [13]). AR modelling has some very useful properties as will be seen later and these will often lead to analytical results where a more general model such as ARMA (autoregressive moving-average where the output contains in addition a weighted sum of previous innovation values) does not. In addition the AR model has a reasonable basis from physical considerations for many speech and audio signals.

Consider now an alternative form for the vector model equation which treats the missing data as the linear 'parameter' in the model, and which will be used in subsequent work for interpolation of missing packets in AR data:

$$\mathbf{e} = \mathbf{A}\mathbf{x} \tag{43}$$

where $\mathbf{A}$ is the $((N-P) \times (N))$ matrix:

$$\mathbf{A} = \begin{bmatrix} -a_P & -a_{P-1} & \cdots & -a_1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -a_P & -a_{P-1} & \cdots & -a_1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -a_P & -a_{P-1} & \cdots & -a_1 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & -a_P & -a_{P-1} & \cdots & -a_1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & -a_P & -a_{P-1} & \cdots & -a_1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & -a_P & -a_{P-1} & \cdots & -a_1 & 1 \end{bmatrix} \tag{44}$$

This form will be useful for modelling cases where a particular AR model $\mathbf{a}$ with fixed parameters is assumed but some elements of $\mathbf{x}$ are unknown (these are then treated as the 'parameters' of the problem).

An interpolation procedure which has proved highly successful is the Least Squares AR-based (LSAR) method [19, 10, 9].

Consider first a block of data samples $\mathbf{x}$ drawn from an AR process with parameters $\mathbf{a}$. As in (42) and (43) we can write the excitation $\mathbf{e}$ in terms of the data vector:

$$\mathbf{e} = \mathbf{A}\mathbf{x} \tag{45}$$

Now suppose that a section of data of length l samples starting at sample m is known to be missing or irrevocably corrupted by noise. Now partition the vector $\mathbf{x}$ into 'known' and 'unknown' sections. The unknown section is denoted by a vector $\mathbf{x}_{(\mathbf{i})}$ of l unknown samples, whilst the preceding and following known data samples are denoted $\mathbf{x}_{-(\mathbf{i})a}$ and $\mathbf{x}_{-(\mathbf{i})b}$, respectively, as follows:

$$\mathbf{x} = \left[\mathbf{x}_{-(\mathbf{i})\mathbf{a}}{}^T \; \mathbf{x}_{(\mathbf{i})}{}^T \; \mathbf{x}_{-(\mathbf{i})\mathbf{b}}{}^T\right]^T \tag{46}$$

Matrix $\mathbf{A}$ is partitioned correspondingly by columns so the excitation equation of (44) becomes:

$$\mathbf{e} = \left[\begin{array}{ccc} \mathbf{A}_{-(\mathbf{i})\mathbf{a}} & \mathbf{A}_{(\mathbf{i})} & \mathbf{A}_{-(\mathbf{i})\mathbf{b}} \end{array}\right] \left[\begin{array}{c} \mathbf{x}_{-(\mathbf{i})\mathbf{a}} \\ \mathbf{x}_{(\mathbf{i})} \\ \mathbf{x}_{-(\mathbf{i})\mathbf{b}} \end{array}\right] \tag{47}$$

Multiplying this expression out and grouping the known sections of $\mathbf{x}$ together such that $\mathbf{x}_{-(\mathbf{i})}^T = \left[\begin{array}{cc} \mathbf{x}_{-(\mathbf{i})a}{}^T & \mathbf{x}_{\mathcal{K}b}{}^T \end{array}\right]$ and $\mathbf{A}_{-(\mathbf{i})} = \left[\begin{array}{cc} \mathbf{A}_{-(\mathbf{i})a} & \mathbf{A}_{-(\mathbf{i})b} \end{array}\right]$ gives:

$$\mathbf{e} = \mathbf{A}_{-(\mathbf{i})}\mathbf{x}_{-(\mathbf{i})} + \mathbf{A}_{(\mathbf{i})}\mathbf{x}_{(\mathbf{i})} \tag{48}$$

where we have now separated the excitation $\mathbf{e}$ into known and unknown components. The least-squares solution is then found from minimization of $E = \mathbf{e}^T\mathbf{e}$ w.r.t. the unknown data $\mathbf{x}_{(\mathbf{i})}$ using standard differential calculus of vectors to be:

$$\mathbf{x}_{(\mathbf{i})}^{\mathrm{LS}} = -(\mathbf{A}_{(\mathbf{i})}{}^T\mathbf{A}_{(\mathbf{i})})^{-1}\mathbf{A}_{(\mathbf{i})}{}^T\mathbf{A}_{-(\mathbf{i})}\,\mathbf{x}_{-(\mathbf{i})} \tag{49}$$

This formula can be also interpreted as a ML or Bayesian estimate for the 'missing' samples, see [9].

Note that the LS estimate can be generalized to include cases where samples are missing at random (known) positions in the data. $\mathbf{x}$ is once again partitioned into two vectors $\mathbf{x}_{-(\mathbf{i})}$ and $\mathbf{x}_{(\mathbf{i})}$ containing known and unknown samples respectively. This is as before except that there is now no constraint that elements of $\mathbf{x}_{(\mathbf{i})}$ should correspond to one contiguous 'burst' of degradation. $\mathbf{x}_{(\mathbf{i})}$ and $\mathbf{x}_{-(\mathbf{i})}$ can now contain any combinations of all the sample numbers between 1 and N. $\mathbf{A}$ can be partitioned by columns in exactly the same way to give two matrices $\mathbf{A}_{-(\mathbf{i})}$ and $\mathbf{A}_{(\mathbf{i})}$ which contain columns of $\mathbf{A}$ corresponding to elements of $\mathbf{x}_{-(\mathbf{i})}$ and $\mathbf{x}_{(\mathbf{i})}$ respectively.

References

[1] H. Akaike. A new look at the statistical model identification. 19(6):716–723, 1974.

[2] J. Allen. Short term spectral analysis, synthesis and modification by discrete Fourier transform. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-25:235–239, June 1977.

[3] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. John Wiley & Sons, 1994.

[4] M. Berouti, R. Schwartz, and J. Makhoul. Enhancement of speech corrupted by additive noise. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 208–211, 1979.

[5] S. F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Trans. Acoustics, Speech and Signal Processing*, 27(2):113–120, April 1979.

[6] S.F. Boll. Speech enhancement in the 1980's: Noise suppression with pattern matching. In S. Furui and M.M. Sondhi, editors, *Advances in Speech Signal Processing*, pages 309–325. Marcel Dekker, Inc., 1992.

[7] G. E. P. Box and G. C. Tiao. *Bayesian Inference in Statistical Analysis*. Addison-Wesley, 1973.

[8] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[9] S. J. Godsill and P. J. W. Rayner. *Digital Audio Restoration: A Statistical Model-Based Approach*. Berlin: Springer, ISBN 3 540 76222 1, September 1998.

[10] A. J. E. M. Janssen, R. Veldhuis, and L. B. Vries. Adaptive interpolation of discrete-time signals that can be modeled as AR processes. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-34(2):317–330, April 1986.

[11] H. Jeffreys. *Theory of Probability*. Oxford University Press, 1939.

[12] J. S. Lim and A. V. Oppenheim. Enhancement and bandwidth compression of noisy speech. *Proc. IEEE*, 67(12), 1979.

[13] J. Makhoul. Linear prediction: A tutorial review. *Proc. IEEE*, 63(4):561–580, 1975.

[14] R. J. McAulay and M. L. Malpass. Speech enhancement using a soft-decision noise suppression filter. *IEEE Trans. Acoustics, Speech and Signal Processing*, 28(2):137–145, April 1980.

[15] A. Papoulis. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York, 3rd edition, 1991.

[16] R.D. Preuss. A frequency domain noise cancelling preprocessor for narrowband speech communications systems. *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 212–215, April 1979.

[17] C. P. Robert. *The Bayesian Choice*. Springer, New York, 2nd edition, 2001.

[18] C. W. Therrien. *Discrete Random Signals and Statistical Signal Processing*. Prentice-Hall, 1992.

[19] S. V. Vaseghi. *Algorithms for Restoration of Archived Gramophone Recordings*. PhD thesis, University of Cambridge, 1988.

[20] N. Wiener. *Extrapolation, Interpolation and Smoothing of Stationary Time Series with Engineering Applications*. MIT Press, 1949.