SIGNAL PROCESSING FOR ULTRASONIC FOETAL MONITORING

G.K. Manning B.Sc. A.M.I.E.E.

Thesis submitted for the degree of Doctor of Philosophy to the Faculty of Science, University of Edinburgh.

1987
ABSTRACT

This thesis examines a variety of techniques which may be used to derive foetal heart rate (FHR) estimates from ultrasound monitoring, using wide or narrow beam transducers. The clinical importance of FHR in antenatal and perinatal monitoring is discussed and the role of ultrasound as a monitoring technique applicable throughout pregnancy outlined.

The signal characteristics of the ultrasound returns from the foetal heart and surroundings are examined and together with known characteristics of FHR used to establish the required signal processing. The latter involves implementing a matched filter to the fundamental component of the foetal cardiac cycle. A number of techniques by which this may be achieved are shown. Analogue circuitry using a narrow bandpass switched capacitor filter has been designed and constructed. This circuit uses a control loop to position and hold the filter on the fundamental component. Problems associated with the tracking performance of the control loop, and "ringing" in the narrow bandpass filter produced by impulse like noise can cause artifacts in the FHR plot.

Time domain digital signal processing techniques, in particular autocorrelation have also been used to estimate the period of the foetal heart beat. The limitations of these techniques with reference to the quantisation of the FHR and the fast sampling rates and processing times required are discussed. Using frequency domain algorithms, this quantisation can be reduced, together with the computational load. Block and recursive Fourier spectral analysis algorithms have been used to determine FHR accurately with low sampling rates and using general purpose microprocessor hardware.

A number of autoregressive (AR) spectral analysis techniques have been used offline to determine FHR from the ultrasonic signal. The problems of selecting appropriate model order when using these techniques is covered and a comparison of the algorithms show that the gradient lattice, gradient LMS and optimum tapered Burg algorithms produce good spectral estimates for this data, and suffer from less peak bias than other AR algorithms. Examples of processed ultrasound recordings are given for each of the techniques applied.
DECLARATION OF ORIGINALITY

This thesis was composed entirely by myself. The work reported herein was conducted exclusively by myself, in the Department of Electrical Engineering, University of Edinburgh.
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. James H. Dripps for his encouragement, support and direction. Kevin Dalton for his assistance in providing medical expertise, and Burr-Brown (Livingston) Ltd. for financial support. I would also thank my mother-in-law Mrs J. Knowles for physical sustenance over the period of writing this work. Finally thanks to my patient and long-suffering wife, Gillian, for her for toleration of the disruption to family life, that has resulted from my long sojourn in the Department of Electrical Engineering, while completing this volume.
Contents

Title Page ........................................... i

Abstract ............................................... ii

Declaration of Originality ................................ iii

Acknowledgements ........................................ iv

Contents .............................................. v

List of abbreviations .................................... ix

1. Introduction ........................................ 1
   1.1. Importance of assessing foetal health .......... 2
       1.1.1. Life-Threatening Conditions .................. 2
           1.1.1.1. Foetal Hypoxia/Asphyxia ................. 3
           1.1.1.2. Foetal Cardiac Arrest ................... 4
       1.1.2. Antenatal Stress/Non-stress tests .......... 4
       1.2. Diagnostic Indicators of Foetal Distress .... 5
       1.3. Interpretation of cardiographs ............... 7
           1.4.2. Antenatal monitoring of FHR .............. 8
               1.4.2.1. Phonocardiography .................. 9
               1.4.2.2. Abdominal Electrocardiography ........ 9
               1.4.2.3. Doppler Ultrasound .................. 11
       1.4.3. Background to specific project ............. 12
           1.4.3.1. Foetal Home Telemetry ................. 12
           1.4.4. Thesis Overview ......................... 13
2. Analogue Techniques to measure foetal heart rate 15
   2.1. Signal Description .................................. 15
   2.2. Demodulation and Preprocessing ...................... 18
   2.3. Tracking Switched Capacitor Filters ................. 26
      2.3.1. Stagger Tuned Bandpass Filters ................. 27
      2.3.2. Tracking Problems ................................ 30
         2.3.2.1. Lock-in Time .............................. 30
         2.3.2.2. Biasing .................................. 31
         2.3.2.3. Tracking Performance ..................... 32
   2.4. Phase Lock Loop Tracking Filter .................... 32
      2.4.1. Circuit Description ............................ 33
      2.4.2. Phase Lock Loop Design ....................... 33
      2.4.3. Problems using Narrow Bandpass Filters ...... 38
   2.5. Results ............................................. 39
   2.6. Summary ............................................. 40

3. Time Domain Digital Signal Processing ............... 45
   3.1. Autocorrelation Definition .......................... 46
   3.2. SNR Improvement using the ACF ...................... 47
   3.3. Frequency estimates using Autocorrelation ....... 48
   3.4. Autocorrelation measurement of heart rate ....... 51
      3.4.1. Measurement of Fundamental Period ............. 51
      3.4.2. ACF Fundamental Peak Finding Algorithm ...... 56
      3.4.3. Signal Bandwidth Considerations .............. 56
      3.4.4. Signal Processing Development System ........ 57
5. Autoregressive Spectral Analysis Techniques ...... 114
  5.1. Introduction ...................................... 114
    5.1.1. Chapter Organisation .......................... 115
  5.2. AR Spectral Analysis ................................ 116
    5.2.1 ARMA, AR, MA Process Models .................. 117
    5.2.2 Power Spectral Density of an AR process .... 120
  5.3 AR Parameter Estimation Algorithms ................. 123
    5.3.1. Yule-Walker Solution .......................... 124
    5.3.2. Linear predictors .............................. 126
      5.3.2.1. Example of whitening a sinusoid .......... 129
    5.3.3. LMS Gradient Transversal Filter .............. 131
    5.3.4. Maximum Entropy Method - Burg Algorithm .... 134
    5.3.5. Weighted Burg Algorithms .................... 139
  5.4. AR Model Order selection .......................... 140
    5.4.1. AR Estimation of a sinusoid in white noise .... 143
  5.5. AR Analysis to determine FHR ...................... 149
    5.5.1. FM tracking of a sinusoid in white noise .... 149
    5.5.2. Performance Results ........................... 157
    5.5.3. FHR estimation using the AR PSD ............. 158
  5.6. Summary ........................................... 175

6. Conclusions ........................................... 177
  6.1. Limitations of the work ............................ 179
References ............................................. 180
Appendix A - Publications .............................. 193
Appendix B - Software ................................. 212
### List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2$</td>
<td>Oxygen</td>
</tr>
<tr>
<td>$CO_2$</td>
<td>Carbon Dioxide</td>
</tr>
<tr>
<td>3D</td>
<td>Three Dimensional</td>
</tr>
<tr>
<td>ACF</td>
<td>Autocorrelation Function</td>
</tr>
<tr>
<td>ADC</td>
<td>Analogue to Digital Converter</td>
</tr>
<tr>
<td>AGC</td>
<td>Automatic Gain Control</td>
</tr>
<tr>
<td>ALE</td>
<td>Adaptive Line Enhancement</td>
</tr>
<tr>
<td>AM</td>
<td>Amplitude modulate</td>
</tr>
<tr>
<td>AR</td>
<td>Autoregressive</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Moving Average</td>
</tr>
<tr>
<td>BPM</td>
<td>Beats per minute</td>
</tr>
<tr>
<td>CFT</td>
<td>Continuous Fourier Transform</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CW</td>
<td>Continuous Wave</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to Analogue Converter</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>ECG</td>
<td>Electrocardiogram</td>
</tr>
<tr>
<td>EPROM</td>
<td>Electrically Programmable Read Only Memory</td>
</tr>
<tr>
<td>EWB</td>
<td>Energy Weighted Burg</td>
</tr>
<tr>
<td>FECG</td>
<td>Foetal Electrocardiogram</td>
</tr>
<tr>
<td>FET</td>
<td>Field Effect Transistor</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FHP</td>
<td>Foetal Heart Period</td>
</tr>
<tr>
<td>FHR</td>
<td>Foetal Heart Rate</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>FM</td>
<td>Frequency Modulate</td>
</tr>
<tr>
<td>IFFT</td>
<td>Inverse Fast Fourier Transform</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Square</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Prediction</td>
</tr>
<tr>
<td>MA</td>
<td>Moving Average</td>
</tr>
<tr>
<td>MDF</td>
<td>Modulus Difference Function</td>
</tr>
<tr>
<td>MEM</td>
<td>Maximum Entropy Method</td>
</tr>
<tr>
<td>MLW</td>
<td>Main Lobe Width</td>
</tr>
<tr>
<td>OCT</td>
<td>Oxytocin Challenge Test</td>
</tr>
<tr>
<td>OTB</td>
<td>Optimum Tapered Burg</td>
</tr>
<tr>
<td>PARCOR</td>
<td>Partial Correlation</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Lock Loop</td>
</tr>
<tr>
<td>PSD</td>
<td>Power Spectral Density</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SBC</td>
<td>Single Board Computer</td>
</tr>
<tr>
<td>SCF</td>
<td>Switched Capacitor Filter</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>US</td>
<td>Ultrasound</td>
</tr>
<tr>
<td>VCO</td>
<td>Voltage Controlled Oscillator</td>
</tr>
<tr>
<td>YW</td>
<td>Yule Walker</td>
</tr>
<tr>
<td>w.r.t</td>
<td>with respect to</td>
</tr>
</tbody>
</table>
CHAPTER ONE

Introduction

This chapter outlines the importance of foetal monitoring and describes the main foetal monitoring techniques. Current clinically significant variables used in assessing foetal health are also discussed. The performance of the various non-invasive monitoring techniques for use in antenatal monitoring is indicated and the specific project to which this work was directed is outlined. The chapter concludes with an overview of the thesis and a description of the main subjects covered in the successive chapters.

Foetal monitoring originated early this century with the discovery that foetal heart sounds were audible through the maternal abdomen. The Pinard stethoscope — an ear trumpet like horn, became a standard piece of equipment for use by general practitioners, midwives, and obstetricians for simple auditory phonocardiography (often termed ascultation). Until the mid 1960’s, the detection of foetal distress during labour relied solely on listening for gross changes* in foetal heart rate (FHR) using the Pinard stethoscope and on the observation of foetal bowel movements (meconium) in the amniotic fluid. Antenatal monitoring of the foetus (monitoring before the onset of labour) consisted of listening for the foetal heart sounds — if present the foetus was still alive.

* The measurement of FHR by ascultation has subsequently shown[1] to be inaccurate with 20% of measurements in error by more than 15 BPM, and a tendency for observers to adjust rates of more than 150 BPM and less than 130 BPM to lie within that range.
During the 1960's a revolution took place in the clinical management of labour in most hospitals in the developed world. New electronic monitoring techniques enabled changes in FHR with maternal contractions to be observed in real-time during labour. Certain patterns of FHR with respect to the onset of maternal contractions were identified as indicating foetal distress, thus providing obstetricians and midwives with warning information which could quickly be acted upon to reduce the risk of handicap or death of the foetus. This is now routine practice in most labour wards today. Foetal monitoring is now also applied antenatally to assess foetal health prior to labour and in studying foetal development throughout pregnancy.

1.1. Importance of assessing foetal health

During the course of a woman's lifetime, the period of child birth is associated with one of relatively high health risk to both mother and child. The objective of foetal monitoring is to provide obstetricians and midwives with supplementary information on which they may act to reduce the risk of death or severe handicap of the newborn child. Antenatal monitoring of the foetus provides information on foetal development, and may also determine the foetus's ability to undergo the stresses involved in birth. This enables decisions to be made on the most appropriate form of delivery - Caesarian or otherwise.

1.1.1. Life-Threatening Conditions

The most serious complications which may arise during the pregnancy and labour are as follows
1.1.1.1. Foetal Hypoxia/Asphyxia

Lack of oxygen in foetal tissues is described as hypoxia. As this becomes more severe it is termed asphyxia. Insufficient blood supply to the foetus leads to a decrease in the oxygen supplied to the foetus. This may be caused by uteroplacental insufficiency (the inability of the placenta to provide adequate resources for healthy foetal development) causing poor oxygen and nutrient transfer from the mother to foetus over a long period antenatally. It may also be caused during labour by compression of the umbilical cord, or pressure on the foetal skull caused by uterine contractions. The foetus attempts to compensate for this deprivation by using its chemical reserves in the liver to indirectly maintain a supply of oxygen and energy (anaerobic metabolism). However, if the reduced blood supply persists, this results in a build up of lactic acid in the blood and a condition termed acidosis results. This is compounded further by the inability of the foetus to decrease the carbon dioxide ($CO_2$) levels in its blood since $CO_2$ is normally removed through the placenta. The brain is very sensitive and has a limited ability to withstand the effects of this deprivation thus, as this deprivation continues, brain damage of varying degrees will ensue and ultimately foetal death. For a growth retarded foetus (also associated with uteroplacental insufficiency) since it has poorer reserves of stored energy the effects of oxygen deprivation during labour will occur earlier than for a normal healthy foetus and there is thus a higher risk of brain damage and death. Mental handicap resulting from asphyxia during labour accounted for 2.4 per 1000 live births, as surveyed in a study of 1974.
1.1.2. Foetal Cardiac Arrest

The trauma to which the foetus is subjected by the uterine contractions, umbilical cord, or skull compression places strain on the foetal heart and its control mechanism. In some cases this may lead to cardiac arrest.[2] The heart control mechanism is thought to consist of a cardiac accelerator system* balanced by a cardiac decelerator system†. If for some reason this control system is not sufficiently developed or is unstable with the decelerator dominating then cardiac arrest is likely to occur when the foetus is stressed. The foetus will survive with little after effect if this is only temporary or if the cardiac arrest is detected and corrected by administering drugs. However if it persists or is repetitive brain damage and foetal death may ensue.

Both these conditions are associated with rapid decelerations in FHR.

1.1.2. Antenatal Stress/Non-stress tests

Antenatal monitoring provides an insight into the development of foetal central nervous systems and may aid in the detection and diagnosis of anecaphy — absence of a major portion of the foetal brain tissue. A major use is in the assessment of foetal health prior to labour and birth. This may be accomplished using foetal stress tests.

In the stress test the foetus is stressed by inducing maternal contractions[3] using the drug oxytocin - the Oxytocin Challenge Test (OCT), or by inducing moderate maternal hypoxia[4] through the mother breathing a 12% oxygen in nitrogen mixture

* the sympathetic automatic nervous system
† the parasympathetic automatic nervous system or vagal system. This is responsible for such involuntary actions as decrease in heart rate and constriction of the iris.
for 15 minutes. In both these tests FHR is recorded throughout and after the test period and changes in FHR noted in response to the stress. Both tests have their limitations. In the OCT the level of stress applied to the foetus is difficult to measure. Furthermore it may cause premature labour, and is awkward to perform since it requires drug infusion and up to 90 minutes monitoring. In the hypoxia test the risks to the foetus may be considered unacceptable, as urgent delivery of foetuses failing this test is required, otherwise foetal death may occur.

Non-stress tests involve the monitoring of FHR in relation to spontaneous foetal activity, or in response to acoustic or manual stimulation. There is a high incidence of increased FHR following foetal activity or stimulation. This indicates a healthy foetus. The absence of such changes are indicative of poor foetal health.

1.2. Diagnostic Indicators of Foetal Distress

The following physiological variables may be monitored in order to assess foetal health:

1. Foetal Breathing Movements.

2. Foetal Gross body Movements.

3. Foetal Heart Rate, antenatally in association with stress or non-stress tests, or during labour in association with maternal contractions.

4. Foetal heart valve timings.

* A more detailed explanation of their measurement and significance is given by Bennett, or may be found in the various papers referenced.
5. Foetal Electrocardiogram (FECG).\textsuperscript{[15,16]}

6. Foetal Body Size measurements

7. Foetal Blood $CO_2$ and $O_2$ gas levels

8. Maternal weight gain.

9. Maternal blood pressure, and blood analysis.

10. Maternal Urine Chemical Analysis

11. Amniotic Fluid Chemical Analysis

Any major decision taken concerning foetal health will usually be based on more than one of the above measurements. Each measurement will have an associated degree of confidence concerning how well the measurement reflects foetal health. In the routine antenatal screening for spina bifida, growth retardation and/or placental insufficiency, methods 2, 6, 8, 9, 10 and 11 are currently practiced. Scientific research continues on measuring, processing and determining the clinical significance of foetal breathing movements, and foetal heart valve timing, as these methods are not yet part of routine clinical monitoring. Examination of the foetal ECG and foetal blood gas levels are required on occasion during labour. The antenatal monitoring of FHR is mostly restricted to high risk patient categories. These are mothers with prior still-births, high blood pressure, diabetes or other complicating factors associated with a greater incidence of intrauterine growth retardation.

During Labour FHR may be monitored usually by means of Doppler ultrasound, or more reliably by means of an invasive electrode attached to the presenting part of the foetus (normally the scalp). In the case of twins both techniques may be applied
simultaneously. one technique for each foetus. Maternal contractions are measured either externally by a pressure transducer on the maternal abdomen or internally by a fluid filled catheter in the uterus connected externally to a strain gauge pressure transducer. The various patterns of FHR with respect to uterine pressure have been well studied by Hon. Lee and others, although automatic on-line classification of these has yet to be incorporated into commercial foetal monitoring equipment.

1.3. Interpretation of cardiographs

Features in the FHR cardiograph may be classified by the following:

(a) Long term average rate termed the baseline rate.
(b) Short term fluctuations in the beat to beat heart rate about the baseline termed the heart rate variability.
(c) Decelerations in the heart rate from the baseline value.
(d) Accelerations of heart rate from the baseline value.

During labour the cardiograph is viewed in relation to maternal uterine contractions and a number of patterns have been identified which relate to the condition of the foetus. Of these only those indicative of foetal distress will be considered here. Normal FHR is characterised during labour by:

1. A baseline rate between 120—160 BPM.
2. A variability of greater than ±5 BPM.
3. No change in heart rate during maternal contractions.

A baseline FHR below 120 BPM is termed bradycardia. Bradycardia significantly less than the 120 BPM may be indicative of congenital heart defects. Fetal hypoxia, asphyxia and/or acidosis may be revealed by the following FHR changes:

1. A baseline FHR above 160 BPM termed tachycardia, although other factors such as maternal anxiety can also produce foetal tachycardia. A FHR above 200 BPM are due most probably to heart arrhythmia.

2. Prolonged FHR decelerations which are decelerations of over 30 BPM and of duration greater than 2.5 minutes.

3. FHR decelerations which occur late or with variable delay, with reference to the onset of the maternal contractions.

4. Under certain circumstances the loss of FHR variability.

A sinusoidal pattern of FHR with period between 12 and 30 seconds is associated with severe foetal anaemia and a high probability of foetal death.

1.4.2. Antenatal monitoring of FHR

The foetal heart beat can be detected as early as 10 weeks gestation with the FHR tending to decrease with increasing gestational age. This is due mainly to the growth of the heart itself. Beat-to-beat variations in FHR appear after about 20 weeks gestation and are indicative of the performance of the cardiac-decelerator control of the foetal automatic nervous system. Antenatal monitoring of FHR must, by necessity, be non-invasive and one of three techniques may be used to determine FHR: Phonocardiography, Abdominal electrocardiography, or Doppler Ultrasound.
1.4.2.1. Phonocardiography

An electro-magnetic or piezo-electric microphonic transducer is placed at an appropriate location on the maternal abdomen to pick up the foetal heart sounds due to heart valve closures. Using fixed filtering and threshold crossing timing the FHR can be measured on a beat to beat basis.

Of the non-invasive monitoring techniques available this is the least commonly used in practice, due to the problems of picking up extraneous noise. Work by Solum[23] comparing FHR tracings from the same patients using all three monitoring techniques showed that good recordings were obtained in only 23.4% of the patients monitored from 34 weeks gestation to term, compared to 55.2% for abdominal ECG and 85.9% for ultrasound recordings. Also the signal quality may be affected by the position of the placenta and the obesity of the patient, thus restricting the class of patients to which it may usefully be applied. Heart sounds are also difficult to detect early in pregnancy.

Recently considerable improvement has been made in the development of suitable acoustic transducers[24] to perform phonocardiography and there is current renewed interest in this technique which has the advantage of being passive, relatively insensitive to foetal position within the uterus, and enables possible valve timing and beat-to-beat variability to be measured.

1.4.2.2. Abdominal Electrocardiography

Electrodes positioned on the maternal abdomen are used to pick up the FECG signal as shown in Figure 1.1 The electrodes also pick up the maternal ECG waveform and abdominal muscle (myoelectric) noise. Noise averaging[25] or cancellation[26,27] may
Figure 1.1
Abdominal foetal electrocardiographic signal

be used to reduce these extraneous signals, and obtain an averaged FECG waveform and also determine the FHR. However, these techniques tend to rely on a stable identifiable event marker which is usually taken to be the peak of the QRS complex, the R-wave.\[28\] This is often not clearly discernible in the abdominal ECG signal. FHR can be estimated either from auto-correlation processing (Chapter 3) or threshold crossing timing.

Abdominal ECG techniques give poor results from the 27th to 34th week of gestation,\[4\] during which time the amplitude of the ECG waveform is small. This results in poor recordings being obtained in 70% of patients at this stage in pregnancy.\[29\] It does however provide beat-to-beat measurement of FHR and the
quality of FHR recordings obtained later in pregnancy make the use of this technique complementary to that of ultrasound.

1.4.2.3. Doppler Ultrasound

A transducer coupled to the maternal abdomen through a water based gel transmits ultrasound at a frequency of between 2 to 5 MHz at a low power level of less than $10mW/cm^2$. The transducer consists of one or many transmitting and receiving piezoelectric crystals producing a narrow or wide beam of pulsed or continuous wave ultrasound. Reflections of the ultrasound from moving tissue within the beam are Doppler shifted and picked up at the transducer. When the beam encompasses the foetal heart the main Doppler sources are: arterial blood flow from around the heart, the opening and closing of the heart valves, and the pulsation of the foetal heart walls. Other components such as blood flow through the umbilical cord, venous blood flow, foetal breathing and maternal fluid flow add further modulating components to the received signal. The complexity and non-stationarity of this signal make it difficult to obtain beat-to-beat measurements of FHR and all monitors using on this technique produce a short term estimate of the FHR either from averaged threshold crossing timings[30] or autocorrelation processing.[31]

The main advantage of ultrasonic monitoring is the ability to monitor through pregnancy from early in gestation. With correct transducer positioning a high signal to noise ratio (SNR) signal can be obtained. However, with foetal movements, the heart can stray outside the beam resulting in total signal loss. The ease of application of ultrasound make it the most common antenatal monitoring technique.
1.4.3. Background to specific project

All the non-invasive monitoring techniques require improved signal processing in order to obtain consistent accurate FHR measurement. However, as ultrasound is simple to apply throughout pregnancy irrespective of gestational age and its use is routine in most antenatal clinics, improvement in its performance would be most beneficial. This project aimed to improve the determination of FHR estimates from ultrasound in order that good quality FHR recordings could be obtained earlier in pregnancy, and the technique could be used effectively by unskilled personnel. This would allow the study of possible FHR rhythms which may occur as indicators of foetal health at different stages during foetal development. A further motive was to enable patients to monitor themselves in their own home and still provide good FHR recordings.

1.4.3.1. Foetal Home Telemetry

The technology has existed for a number of years to enable data to be collected in one area and transferred over the telephone network to another location. In the foetal monitoring context, patients who are considered to be in the high-risk category currently spend the latter months of their pregnancy in hospital for observation. During this hospitalisation, FHR is usually monitored four times daily for ominous signs such as decelerations which would suggest possible hypoxia or other complication. This is a necessary preventive procedure. However, it is expensive in terms of the hospitalisation for this period and the disruption to the patient and her family. Foetal home telemetry could provide a simple means of minimising this disruption until it may actually be warranted.
The patient would be provided with the necessary monitoring equipment in her own home and some simple training in its use. FHR recordings could then be obtained by the patient as often as required and then transmitted digitally back to the hospital for inspection by the obstetrician in charge. The feasibility of this has already been demonstrated successfully,[32, 33, 34, 35] and operating costs have been assessed to be less than 6% of the daily cost of hospitalisation.[36] Other factors are also monitored during the patient’s hospital confinement such as maternal blood pressure and blood glucose levels in diabetic patients. Work continues to enable these to be monitored remotely and some trials of such monitoring equipment have been carried out.[37, 38]

1.4.4. Thesis Overview

This thesis describes a number of different techniques which have been applied to determine FHR from the Doppler Ultrasound foetal heart returns. Chapter 2 presents a detailed explanation of the origin and content of the received signal and describes the preprocessing and objectives of the signal processing to be preformed. Two analogue techniques are presented which use switched capacitor filters to perform this processing. Their limitations are discussed. Various autocorrelation algorithms and their use in estimating the short term FHR digitally in the time domain are explained in chapter 3 and results obtained using these and a faster modulus difference algorithm are presented. Chapter 4 and 5 extend the work to algorithms which estimate the FHR in the frequency domain. Chapter 4 presents various block and recursive Fourier Transform techniques that have been used to determine FHR by locating the fundamental component of the foetal heart beat from the power spectrum of the signal. Similarly a number of autoregressive spectral analysis algorithms have been used to determine the FHR from the power spectral density function and the mathematical theory upon which these estimators are based is presented in chapter 5.
The influence of factors such as model order choice and SNR on the performance of these spectral estimators is discussed and a comparison of these algorithms with that of conventional Fourier spectral estimation is made through the study their ability to track a frequency modulated sinusoid in white noise. The final chapter gives a summary of the work performed, conclusions and suggestions for possible further research. For ease of comparison, FHR results from ultrasound monitoring of a foetus at 40 weeks gestational age are presented for each of the techniques and algorithms studied.
CHAPTER TWO

Analogue Techniques to measure foetal heart rate

This chapter presents two analogue techniques which were applied in order to determine FHR from the ultrasonic heart valve returns. Section one discusses the characteristics of the ultrasonic returns and their origin. Section two outlines the basic assumptions used in deciding what processing may be preformed in order to improve SNR and hence FHR estimation, and describes the circuit used in the first stage of processing. Section three describes the use of three switched capacitor filters in implementing a tracking bandpass filter in order to approximately match filter the FHR fundamental component. The tracking and other limitations of this method is discussed, and an improved method using a Phase Locked Loop is presented in section four. The chapter concludes with a presentation of the results obtained using these methods in determining FHR from actual patient data, and a summary of the properties of the techniques.

2.1. Signal Description

As outlined in Chapter One currently the most reliable and widely used indicator of foetal well-being in early pregnancy is that of foetal heart rate, determined from noninvasive ultrasound monitoring. Using a wide or narrow beam transducer coherent continuous wave (CW) or pulsed ultrasound at a frequency of between 2 to 5 MHz is directed at the foetal heart through the maternal abdomen. Any movement of tissue within the beam with a different characteristic impedance to that of the
surroundings (notably the foetal heart walls and valves) Doppler shifts the frequency of the ultrasound. This Doppler shift \( f_v \) is given approximately by

\[
f_v = \frac{2v \cos \theta}{\lambda}
\]

where

\( v \) is the velocity of the ultrasound in soft tissue.

\( \lambda \) is the wavelength of the Ultrasound

\( \theta \) is the angle between the movement velocity vector and the isonating wavefronts of Ultrasound.

The back-scattered signal at the receiver is due to partial reflection of the ultrasound at the discontinuities in the tissue density. This back-scattered signal comprises of Doppler shifted components due to reflections from moving discontinuities together with a large component at the radiated frequency due to reflections from the stationary discontinuities. Additive mixing of the frequency components from the stationary and moving tissue occurs at the receiving transducer producing a signal containing both frequency and amplitude modulated terms. For example consider the result from the additive mixing of a single Doppler shifted ultrasonic return with the returns from stationary tissue.

\[
A_s \cos (\omega, t) + A_m \cos (\omega + \delta \omega_m) t
\]

Stationary Component Moving Component

\[
= A_m \left( 2 \cos (\omega_s + \frac{\delta \omega_m}{2}) t \cos \left( \frac{\delta \omega_m}{2} \right) t \right) + (A_s - A_m) \cos \omega, t
\]

AM/FM Term
where

\( A_s \) is the amplitude of the returns from stationary tissue.

\( A_m \) is the amplitude of the return from the moving tissue.

\( \omega_r \) is the radiated frequency.

\( \delta \omega_m \) is the Doppler shift in frequency due the the velocity of the moving tissue.

The received signal is amplitude demodulated and normally output to a loudspeaker. The frequencies of the Doppler shifted heart valve returns lie in the audio frequency range and the loudspeaker output provides audio feedback which aids in positioning the transducer to obtain a good signal to noise ratio (SNR). This signal was attenuated and recorded onto cassette tape to facilitate the further processing required to estimate foetal heart rate (FHR). Careful positioning of the narrow beam transducer in order to illuminate the foetal heart valves is essential to obtain a good SNR signal from which accurate estimates of the FHR can be made. This however is quite difficult particularly in early pregnancy where the size of the foetus makes the location and tracking of the foetal heart with the ultrasonic beam so much harder.

The problem of maintaining the illumination of the foetal heart by ultrasound has been tackled by Cousins[^39] using an array of ultrasonic transducers. In practice this is alleviated somewhat by the use of a wide beam of ultrasound. The wide beam allows foetal and maternal movement without loss of the signal from the foetal heart, however as the beam now encompasses a larger volume, more noise from the surroundings will be picked up by the transducer. The objective of the signal processing is to improve the SNR of the received signal and to provide reliable estimates of FHR even at poor SNR ratios.
2.2. Demodulation and Preprocessing

Figure 2.1 shows an example of the time domain signal obtained from a foetus at 40 weeks gestation using the transducer arrangement described in section 2.1. This shows clearly the Doppler returns produced during two heart beat cycles. The corresponding frequency domain waveform derived from Figure 2.1 is shown in Figure 2.2. As can be seen significant power is present in the signal from approximately 50 to 600 Hz with each heart valve movement producing a burst of sinusoid. Amplitude demodulation and low pass filtering to 50 Hz produces a waveform such as that in Figure 2.3. Here the individual valve movements are clearly

![Time Domain Ultrasonic Heart Valve Returns](image)

**Figure 2.1**
Time Domain Ultrasonic Heart Valve Returns
distinguishable as they perform the cycle that corresponds to one complete heart beat. The waveform produced however is highly dependent on the angle of illumination of the foetal heart valves by the ultrasound. This angle changes as the position of the foetus changes relative to that of the transducer, predominately due to foetal body movements. These foetal body movements are more prevalent early in pregnancy when the foetus has more room to move. Hence the wave shape can evolve in time as the isonation angle changes relative to the foetal heart.

Figure 2.4 shows how the wave shape has evolved after a few heart beats from that in Figure 2.3. These figures demonstrate the element of nonstationarity which exists in
the signal due to foetal body movements and transducer relocation. Also demonstrated is the fact that it is difficult to obtain a true beat to beat estimate of FHR from Doppler ultrasonic returns since it is difficult from one beat to the next to clearly associate a particular point in the waveform with a unique event in the foetal cardiac cycle. Although each peak corresponds to a valve movement it is difficult to determine which valve gives rise to the Doppler shift which results in a given peak nor can the exact time of valve movement be determined precisely from the ultrasonic returns due to the unknown factor of foetal heart and transducer orientation.
Before attempting to maximise the SNR of the 50-600 Hz wide bandwidth signal to enable determination of a good estimate for the short term FHR, it is necessary to make some general assumptions concerning foetal heart rate itself. These assumptions are based upon the clinical observations of obstetricians working in this field.\textsuperscript{[40, 41, 1, 29]}

1. The foetal heart rate lies in the range from 60 BPM (1 Hz) to 240 BPM (4 Hz), with normal foetal heart rate lying between 120 — 160 BPM
2. The foetal heart rate does not change significantly over a short time interval less than 4 seconds.

Assumption 1 limits the maximum and minimum FHR while assumption 2 limits the rate of change of the FHR (its slew rate) and thus governs the amount of averaging that may be performed without compromising FHR tracking ability.

The signal from the transducer contains heart rate information in both amplitude and frequency modulated forms. When a heart valve opens the amplitude of the Doppler returns will increase due to the movement causing a Doppler shift in the frequency of the incident ultrasound. Also since the heart valves move at a variable velocity as they go from fully open to fully shut so the frequency of the Doppler returns will vary. Over a complete cardiac cycle the sequence of valve openings and closings will manifest itself as frequency and amplitude modulation of the incident ultrasound. It is clear from assumption 1 that the passband of the Doppler returns, that of 50-600 Hz, is greatly in excess of that of the FHR 1-4 Hz. An overall improvement in the SNR can be accomplished by either amplitude or frequency demodulation and bandpass filtering to a bandwidth which contains the major signal components.

Figure 2.5 shows the successive spectra of the amplitude demodulated ultrasonic returns which have been bandpass filtered from 1 to 50 Hz. Each spectrum was computed over a 1 second period and displayed on a linear scale normalised by the power of the largest spectral component. This gives an indication of the minimum bandwidth that may be used. The processing techniques examined in this thesis have concentrated on estimating the period or frequency of the fundamental component to the exclusion of the other more variable harmonic components. At the times when the harmonic components contain significant power this does represent a loss in signal information and during these periods better estimates may be obtained if this
Figure 2.5
Spectrum of Demodulated 50Hz Bandwidth heart valve returns

information were retained. However, the fundamental of the FHR is always guaranteed to be present, whereas the other harmonics vary considerably in intensity depending on the heart rate itself and the transducer orientation. A pre-processing stage Figure 2.6 was designed to reduce the signal bandwidth to concentrate on this fundamental component. From assumption 1 this bandwidth was chosen to be from
1 to 4 Hz in which the fundamental component must lie.

Figure 2.7 shows the typical output after this amplitude demodulation and bandpass filtering has taken place. The resulting waveform would have been similar had frequency demodulation been used. A comparison\textsuperscript{[42]} between amplitude and frequency demodulation of the Doppler returns revealed occasions when the frequency demodulated output would totally collapse. This threshold in performance was possibly due to noise capture. When the frequency demodulation was above this threshold it provided a good SNR signal, however for robustness amplitude demodulation has been used throughout in this pre-processing stage. The signal at the

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.6.png}
\caption{Pre-processing — Demodulation and Filtering}
\end{figure}
Figure 2.7
Demodulated U.S. returns filtered from 1 to 4 Hz

The amplitude demodulator output consists of the fundamental component of the FHR, any harmonics which lie in the 1 Hz to 4 Hz passband, and low frequency noise. A possible component due to maternal blood flow was also noticeable during some recordings when it was suggested that the foetal heart had been illuminated through a maternal abdominal artery. To provide an approximately constant power signal an automatic gain control (AGC) stage follows the demodulation. This is constructed using a bandpass limiter circuit with similar bandwidth to that of the demodulated signal. Although the AGC circuit is nonlinear it has been shown that for a signal with an SNR above -3dB the bandpass limiter improves the SNR by up to 3dB.\cite{43,44}
At this stage a simple estimate of foetal heart rate could be made by use of a zero-crossing detector with the intervals between zero-crossings giving an indication of the beat to beat period of the foetal heart. However although the SNR has been improved further gains can be achieved by processing the signal using the second assumption. By assuming that the heart rate is quasi-stationary over a short time interval (less than 4 secs) a better estimate of FHR can be obtained by matched filtering to the corresponding bandwidth (in this case 0.25 Hz), before limiting and threshold detection to determine the FHR. In order to achieve this further degree of filtering a narrow bandpass filter must be positioned at a centre frequency equal to the current FHR. Since the latter can vary over a range from 1 to 4 Hz, the filter must track the FHR as it changes. Various analogue techniques have been examined in an attempt to solve this problem.

2.3. Tracking Switched Capacitor Filters.

With current advances in micro-electronics it is now possible to obtain integrated circuit tuneable filters based on switched capacitor technology. Lowpass, bandpass, and highpass filters of any order can be constructed by cascading such devices. In the case of the bandpass filters, the two external resistors required, determine the Q factor of the filter which is a constant independent of the centre frequency of the filter. The centre frequency itself is defined by an external clock as either one hundredth or one fiftieth of this clock frequency. Thus by varying the clock frequency the bandpass filter may be tuned to any desired centre frequency.

* National Semiconductor MF10CN.
2.3.1. Stagger Tuned Bandpass Filters

Using three such bandpass filters a simple circuit can be constructed to implement the tracking and further filtering required to reduce the 1 to 4 Hz signal bandwidth to that commensurate with the desired short term average FHR bandwidth of 0.25 Hz. The three filters Figure 2.8 have their respective centre frequencies \( f_1 \), \( f_2 \), and \( f_3 \), stagger tuned by an interval of 0.125 Hz with a Q factor of 10 at centre frequencies of 1.875, 2.0, and 2.125 Hz. The clocks to each filter were derived from three voltage controlled oscillators (VCO's), which enabled the full system to be steered up or down in frequency. By balancing the power in the upper and lower filters it was possible to position the centre frequency of the middle filter to lie directly over the fundamental

![Diagram](image_url)

Figure 2.8
Stagger Tuned Bandpass Filters.
copoot of the FHR. This bilaricing and owing is view through the coSI, Vlgun 23 as described below.

Figure 2.9 Block Diagram of Tracking Filtet Circuit
directly into all three bandpass filters. The power outputs from the upper and lower filters are obtained in each case by perfect full-wave rectification. An error which acts as a control signal is obtained by differencing the two rectified outputs. This error signal is integrated and added to the control voltages of the three VCO's which determine the centre frequencies of the filters. Fixed offsets in the VCO control voltages ensure that the filters retain their spacing with respect to one another at all times. This feedback control loop enables the system to lock onto and track the fundamental component of the FHR.

For example, if at switch on the FHR is higher than the initial centre frequency of the middle bandpass filter more power will be present at the output of the upper filter than the lower filter. The resulting error will cause the VCOs to change the centre frequencies of the corresponding filters in such a way as to steer the filters up in frequency until the power is balanced in the upper and lower filters and the error falls to zero. The output signal is taken from the middle filter which should be a 0.25 Hz bandpass filter centred on the fundamental of the FHR. This signal is limited and threshold detected to calculate the short term average beat to beat period and thence the short term average FHR. Advantage is taken of the 3dB asymptotic improvement in SNR of the limiter at SNRs greater than -3dB\cite{43,44} since the fundamental has been approximately matched filtered and the SNR increased relative to the wider 1 - 4 Hz signal.

Figure 2.10 shows the input and output signals obtained from using this technique.
2.3.2. Tracking Problems

The tracking SCF's technique is relatively simple and a number of major drawbacks became obvious with use. These are outlined as follows.

2.3.2.1. Lock-in Time

If the FHR is significantly higher or lower than the start-up frequency of the filter system, it can take several minutes for the filters to lock to the FHR fundamental. This problem is inherent in the bandpass nature of the filters, since if the FHR fundamental is, for example, much higher than the system's start-up frequency very little
power will be present at the output of the upper filter since the fundamental is in its stopband. Even less power is present at the lower filters output, thus the error is small and the filters move slowly towards the fundamental. This will continue until the FHR approaches the passband of the upper filter where a large error will be produced which could cause the filter to overshoot and the FHR to enter the lower filter depending upon the damping factor of the system.

A possible solution to this might be to widen the bandwidths of the outer filters from which the tracking information is derived, however, this leads to the further problem of biasing.

2.3.2.2. Biasing

Due to the presence of other components in the input signal, the centre frequency of the middle filter may not be the same as that of the FHR. These other components are notably harmonics of the FHR and possible maternal blood flow components. The power output from either the upper or lower filters may be increased depending on the position of these noise components relative to the centre frequency of the middle filter. This causes the filters to be biased or de-tuned in one direction and results in attenuation of the FHR fundamental. Also contributing to the bias is the fact that the SCF's maintain a constant Q factor, not a constant bandwidth, thus as the system moves up in frequency the bandwidth of the upper filter increases more than that of the lower.
2.3.2.3. Tracking Performance

The control loop which provides tracking capability for the filters is difficult to analyse with time varying filter responses and hence difficult to optimise. This could result in a poor tracking performance and thus limit the ability of the system to measure the FHR accurately. For example consider a signal consisting of a fundamental and second harmonic with the filter initially locked in. If a rapid deceleration in FHR occurs the three filters will begin to move down in frequency in order to balance the power in the upper and lower filters. However the second harmonic of the FHR will also be moving down in frequency and if the filters do not track the fundamental component fast enough the second harmonic may approach the passband of the upper filter and cause the system to move up in frequency to lock to the second harmonic. Thus what was in fact a rapid deceleration may be measured as a brief deceleration followed by an acceleration as the filters lock to the harmonic. This is clearly undesirable, and was observed in practice.

2.4. Phase Lock Loop Tracking Filter

In order to overcome the drawbacks of the tracking switched capacitor filters a phase locked loop (PLL) can be used to improve the tracking and lock-in performance.\cite{45,46} The performance of the PLL has been analysed extensively and its design rules well formulated.\cite{44,47,48} PLL’s have been applied in the area of foetal monitoring to the measurement of foetal breathing from ultrasound B-mode scanning.\cite{7,8}
2.4.1. Circuit Description

A PLL is used to replace the upper and lower SCF's and associated control circuitry used to position the bandpass SCF over the FHR fundamental in the previous method. The pre-processed 1 Hz to 4Hz signal is input to both the PLL and the SCF. The PLL will lock to the FHR fundamental and harmonics within the 1-4 Hz bandwidth. The control voltage to the PLL's VCO is filtered and used to control a further VCO\(^{[49]}\) operating at 100 times that of the PLL and this determines the position of the narrow 0.25 Hz bandpass SCF. The output from the SCF is then taken, limited and threshold detected to estimate the FHR. Alternatively the lowpass filtered PLL control voltage may be taken as an estimate of the FHR. A lock detector may also be incorporated into the PLL to act as a coherence estimator and be used to blank out results from periods when the loop is out of lock, however the variable amplitude of the input signal makes it difficult to set a meaningful threshold for the lock detector. The block diagram of this method is outlined in Figure 2.11.

2.4.2. Phase Lock Loop Design

The PLL consists of a phase detector, a loop filter, and a VCO. The phase detector chosen is a non-limiting type, and uses a FET switch under control of the VCO output to cause the input signal to become inverted or non-inverted through the operational amplifier (Figure 2.12). The phase detector has a response for small phase differences given by
Figure 2.11
Block Diagram of Phase Lock Loop Tracking filter

\[ V_e = \frac{4}{\pi}V_s (\theta_i - \theta_o) \]

= \( K_d (\theta_i - \theta_o) \), \( K_d = \frac{4}{\pi}V_s \)

where

- \( V_s \) is the input signal amplitude.
- \( \theta_i \) is the phase of the input signal.
\( \theta_o \) is the phase of the output from the VCO.

The amplitude of the input signal is taken to be 2 Volts in the design of this PLL.

The loop filter (Figure 2.13) is a first order active filter, and has a transfer function

\[
F(s) = \frac{s \tau_2 + 1}{s \tau_1} \quad \tau_1 = R_1.C \quad \tau_2 = R_2.C
\]

The VCO\textsuperscript{[50]} Figure 2.14 has output frequency given by

\[
f_o = f_{centre} - K_o \cdot V_{in}
\]

\[
f_{centre} = \frac{1}{4.R_e.C} \quad K_o = \frac{1}{4.R_s.C.V_s}
\]
The centre frequency of the VCO is chosen to be mid-range between the possible 1-4 Hz limits of the FHR, at 2.5 Hz. Overall the transfer function of this second order PLL is

\[ H(s) = \frac{2\zeta \omega_n s + \omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2} \]

\[ \omega_n^2 = \frac{K_a K_d}{\tau_1} \quad \zeta = \frac{\omega_n \tau_2}{2} \]

The PLL is designed such that it can track rapid variations in FHR and obtain lock-in quickly after signal loss. A worst case estimate\(^{[40]}\) for the rate of change of heart beat period is taken to be 50ms per beat, giving a maximum rate of change of FHR \(\Delta \omega\) of 120 BPM/sec. The PLL should also lock-in with no cycle slipping for normal heart rates and be capable of pulling in for all possible rates. Thus using the formulae given by Gardner\(^{[44]}\)

Maximum tracking rate \((rad/sec^2)\)
The Voltage Controlled Oscillator

Figure 2.14

The Voltage Controlled Oscillator
\[\Delta\omega = \omega_{c,2}\]

Lock-in Range (rad/sec)

\[\Delta\omega_L = 2\zeta\omega_s\]

Pull-in range (rad/sec)

\[\Delta\omega_p = \sqrt{2} (2\zeta K_v - \omega_s^2)\]

\[K_v < 0.4 \omega_s, \quad K_v = K_o K_d\]

A chosen damping factor of \(\zeta = 0.7071\) and \(\Delta\omega = 120 \text{ BPM/sec} = 12.57 \text{ rad/sec}\), determine the loop filter time constants \(\tau_1, \tau_2\) and the VCO gain constant \(K_o\) to be

\[\tau_1 = 2.2 \text{ sec} \quad \tau_2 \text{ sec} = 0.4 \quad K_o = 11 \text{ rad/V sec}\]

This gives the PLL the following properties

- Maximum Tracking rate: 120 BPM/sec
- Lock-in Range: 102 — 198 BPM
- Pull-in Range: 0 — 300 BPM

The control voltage of the PLL is low pass filtered to 0.25 Hz prior to using it to control the positioning of the SCF via the second VCO. This is required due to the limitations on rate at which the SCF can change position.

### 2.4.3. Problems using Narrow Bandpass Filters

The inherent narrow bandpass nature of SCF used in both the PLL and SCF tracking methods produces a problem of "ringing" which manifests itself in a number of ways.

If a narrow bandpass filter is subjected to a sudden impulse the filter will ring at a frequency determined by its centre frequency. The "ringing" is indistinguishable from the sinusoidal output produced when a fetal signal is actually present. Unfortunately
during transducer positioning while attempting to maintain the ultrasonic beam on the fetal heart valves impulsive noise spikes are produced particularly if air bubbles become trapped between the abdominal surface and the transducer. These noise spikes will cause the fetal monitor to record a FHR equal to the centre frequency of the bandpass filter rather than the actual FHR. In the case of the PLL system once the fetal signal is lost the PLL will return to its centre frequency which in turn steers the SCF to a corresponding centre frequency. If at the same time the noise spikes occur an acceleration or deceleration may erroneously be recorded. This "ringing" is also evident once the signal to the narrow bandpass filter is removed, as the filter continues to produce an output for a time dependent on its time constant.

This is a problem which is fundamental to the use of the narrow bandpass filters in these arrangements. The lock detector in the case of the PLL may be used to indicate the coherence of the input signal and determine when the PLL is in lock. However as the lock detector is based on quadrature sampling of the input signal to the phase lock loop, since the amplitude of this signal can vary considerably despite the bandpass limiter AGC it is difficult to set an appropriate threshold to guarantee lock detection.

2.5. Results

The results obtained using the SCF tracking filter and the PLL tracking filter techniques in measuring the FHR from a recorded 30 minute ultrasonic monitoring session of a foetus at 40 weeks gestational age are given in Figures 2.15, 2.16 respectively. These may be compared to that of Figure 2.17 which is the FHR plot obtained using an existing technique. This determines the FHR using threshold crossing interval measurements derived from the wide band 50 - 500Hz ultrasonic signal. The interval measurements are validated using a heuristic algorithm based
on clinical knowledge of the rate of change of beat-to-beat FHR. A interval measurement is validated as an estimate of foetal heart period (FHP) only if it differs from the past and future intervals by less than 50ms. This algorithm can also interpolate between validated measurements for a possible missing interval measurement.

The major acceleration and deceleration features apparent in the FHR plot derived from the interval measurements are reproduced in the plots obtained for the two tracking filter techniques. Results have also been obtained in the regions where no interval measurements were validated using the existing technique. These correspond well to estimates obtained using other analysis techniques discussed later.

2.6. Summary

The origin of the foetal ultrasonic heart returns have been explained and shown to contain information of events in the foetal cardiac cycle in both frequency and amplitude modulated forms. The received returns have a bandwidth of approximately 50 - 600Hz, corresponding to the Doppler returns from features which range from slow moving blood flow to the fast action of the foetal heart valves. The ultrasonic returns are highly dependent on the angle of illumination of the foetal heart. This causes the signal to become non-stationary as the foetus moves w.r.t the transducer. It is therefore difficult to obtain true beat to beat measurement of FHR using ultrasound.

The preprocessing of the signal reduces the signal bandwidth to that of the range of FHR, 1 - 4Hz, by amplitude demodulation and filtering, and uses a bandpass limiter to obtain some improvement in SNR and to implement a AGC. A further reduction of bandwidth to 0.25Hz using a narrow bandpass filter produces an approximate
matched filter to the fundamental component of the FHR. This filter requires to be positioned with a centre frequency corresponding to that of the FHR and be capable of tracking as the FHR changes. Two tracking methods have been implemented. The first method was based on power balancing between SCF’s positioned higher and lower than the centre frequency of the filter to be positioned on the fundamental. This suffered from problems of poor tracking, biasing, and slow initial lock-in. These problems were largely overcome by the use of a PLL to perform the tracking.

The narrow bandpass nature of this filtering technique makes it susceptible to "ringing" when impulsed by noise generated by transducer movement. This combined with the repositioning of the filter to attempt to hold to the fundamental or when the PLL loses lock and returns to its centre frequency can produce artifacts indistinguishable in the FHR plot from true FHR changes.
Figure 2.15
FHR plot obtained using SCF tracking filter
Figure 2.16
FHR plot obtained using PLL tracking filter
Figure 2.17
FHR plot obtained using existing interval timing method
CHAPTER THREE

Time Domain Digital Signal Processing

Autocorrelation Processing

This chapter is concerned with digital signal processing of the ultrasonic signal using algorithms which operate on the signal in the time domain. Digital signal processing has a number of fundamental advantages over analogue processing. Amongst the most relevant of these are accuracy of operation, the reproducibility of the processing, and the flexibility it affords in modifying the processing algorithms. The algorithm discussed in this chapter, to process the foetal ultrasound and produce FHR estimates, is that of correlation. This algorithm has been used in a wide variety of applications from speech processing to communications to enhance the SNR of a signal or aid in the signal detection.

Section one introduces the mathematical definition of correlation and the discrete time algorithms which may be used for digital implementation of the technique. This also gives an example of how it can improve the SNR of a signal. The estimation of frequency from the autocorrelation function (ACF) of a signal and the limitations of its application with regard to non-periodic signals is discussed in section two. This is extended to examine the problem of FHR estimation through the calculation of the ACF of the foetal signal. in section three. the fundamental period of the foetal heart being measured from the ACF. Section four details the signal processing hardware used in the implementation of the digital algorithms presented in this and subsequent
chapters. Section five is concerned with the implementation of the ACF processing using a number of methods of ACF estimation. A further algorithm similar to that of autocorrelation, but requiring less computation, that of the modulus difference function is also presented. The chapter ends with the summary and chapter conclusions, in section 6.

3.1.1. Definition

The autocorrelation function (ACF) of a stationary signal $x(t)$ is defined as:

$$R_{xx}(\tau) = E \left[ x(t) \cdot x(t+\tau) \right]$$

$$= \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t) \cdot x(t+\tau) dt$$

where $E[x]$ denotes the expectation or mean value of $x$.

To obtain the ACF requires by definition infinite time however in practice the ACF is computed over a specific time interval and this is taken to be an estimate of the actual ACF. For example, given the $N+1$ samples of a signal $x(0).....x(N)$ the ACF may be computed as:

$$R_{xx}(n) = \frac{1}{N+1} \sum_{k=0}^{N-n} x(k) \cdot x(n+k) \quad 0 \leq n \leq N$$

This produces a biased linear estimate of the ACF as it assumes $x(k) = 0$ for $k < 0 \cdot k > N$ and divides by $N+1$ (the number of samples in the block) even though zero valued samples are being included as $k$ increases. This estimate is best used where $N$ is large. The unbiased linear estimate of the ACF can be obtained from:

$$R_{xx}(n) = \frac{1}{N-n+1} \sum_{k=0}^{N-n} x(k) \cdot x(n+k) \quad 0 \leq n \leq N$$
Alternatively if the signal can be assumed periodic in \( N \) i.e. \( x(k) = x(N+k) \) where \( N \) is the block length then the circular ACF may be calculated as:

\[
R_{xx}(n) = \frac{1}{N+1} \sum_{k=0}^{N} x(k) \cdot x(k+n \mod N) \quad 0 \leq n < N
\]

Other methods for computing the ACF are based on sequential processing — recursively updating the previous estimate of the ACF with each new input sample, and also the relationship of the ACF to the power spectral density (PSD) of a signal. These techniques will be dealt with later in the chapter.

### 3.1.2. Signal to Noise Ratio Improvement using the ACF

Correlation processing provides one means of implementing a matched filter. A correlation receiver uses the correlation of a known ideal transmitted sequence with an incoming received data sequence which has been corrupted by additive noise. The incoming sequence can be recognised when the shift parameter \( \tau \) is such that the noise corrupted signal overlies the clean reference sequence. Often, as in the case of the FHR signal, no such reference waveform exists. Instead an approximation to this reference is used which is the noise corrupted signal itself. Despite this, the ACF may still be used to enhance the signal. Any one sample in the ACF is produced from the addition of the pairwise products of \( N \) data samples. Any uncorrelated additive random noise in the signal will add non-coherently while a stationary signal adds coherently thus improving the SNR. The resulting ACF is the superposition of the ACF of the signal and the ACF of the noise. Consider the signal \( x(t) \) contaminated by uncorrelated additive zero mean Gaussian noise \( w(t) \) of variance \( \sigma^2 \) to produce the signal \( y(t) \).
\[ y(t) = x(t) + w(t) \]

The ACF of this signal is given by:

\[
R_y(\tau) = E \left[ y(t).y(t+\tau) \right] = E \left[ x(t).x(t+\tau) \right] + E \left[ x(t).w(t+\tau) \right] + E \left[ w(t+\tau).w(t) \right]
\]

but since \( x(t) \) is uncorrelated with \( w(t) \)

\[
R_y(\tau) = E \left[ x(t).x(t+\tau) \right] + E \left[ w(t+\tau).w(t) \right]
\]

Also as the noise is assumed white and therefore uncorrelated from sample to sample

\[
E \left[ w(t+\tau).w(t) \right] = \sigma^2 \quad \tau = 0
\]

\[
= 0 \quad \tau \neq 0
\]

In this case of zero mean white Gaussian noise, only \( R_y(0) \) contains any contribution due to the noise.

### 3.2. Frequency estimates using Autocorrelation

Various techniques are available in order to determine the frequency of a periodic signal from its ACF. A number of spectral analysis techniques are based directly or indirectly on the ACF, examples of which are the Fourier Transform presented in chapter 4 or autoregressive spectral estimation presented in chapter 5.

An alternative simpler method is based on the fact that the ACF of a periodic signal is itself periodic. Consider a block of \( M \) samples containing at least two cycles of a stationary periodic signal as in Figure 3.1. If the linear ACF is calculated for \( \frac{M}{2} \) lags, Figure 3.2 the ACF will start at a maximum at \( \tau = 0 \) where all the samples align with themselves decrease to a minimum when \( \tau \) is such that the sliding and fixed waveforms are in antiphase and reach a local maxima again when \( \tau \) is equal to the
Figure 3.1
Test signal

Figure 3.2
Autocorrelation Function
period of the waveform. By measuring the time interval from $\tau=0$ to the first maxima, the period and hence the frequency of the waveform can be determined. The presence of higher harmonic components beyond the fundamental produce extra local maxima which may cause problems in peak detection although all the harmonics reinforce at the fundamental period to produce a larger correlation peak. This is illustrated in Figure 3.3 where a second harmonic is present in the signal from which this ACF was computed. Another problem arises due to the presence of non-harmonic components in the signal. These components cause the signal to be aperiodic and can bias the position of the maxima in the ACF and thus produce errors in the determination of the period of the fundamental of the waveform. Figure 3.4 shows the ACF for a signal $y(t)$ given by:

$$y(k) = \cos\left(\frac{2\pi k}{16}\right) + 0.5\cos\left(\frac{2\pi k}{11}\right)$$

This represents the fundamental of a desired signal and a non-harmonically related component due to the presence of another undesired signal. The resulting peak position is biased relative to the correlation peak which would be produced by the ACF's of either of the sinusoids by themselves. The accuracy of the determination of the peak in the ACF is limited by the sampling frequency $f_s$ of the original data and is also a function of the frequency $f$ to be measured:

$$\frac{1}{T} < \frac{1}{T_s} < \frac{1}{T - \frac{T_s}{2}}$$

where $T$ is the actual fundamental period of the signal, $T_s$ is the sampling period, and $T_e$ is the estimated period. Thus

$$-\frac{f}{2f_s + f} < f_{\text{err}} < +\frac{f}{2f_s - f}$$
where $f_{\text{err}}$ is the error in the measured frequency.

The higher the frequency to be measured for a given sample rate the lower the accuracy of the measurement of the position of the correlation peak. This is due to the period is being measured from the ACF which has to be quantised into units of the sampling period. The sampling interval becomes a greater percentage of the fundamental period as the fundamental frequency increases and thus the peak in the ACF cannot be determined as accurately in absolute terms.

3.3. Autocorrelation measurement of heart rate

3.3.1. Measurement of Fundamental Period

As already seen in Chapter 2 - Figure 2.5, significant power exists in the demodulated ultrasonic returns up to approximately the seventh harmonic which for the assumed maximum FHR of 4 Hz gives a total passband of around 30 Hz. Ideally the ACF would be computed using this signal bandwidth at the highest sampling rate attainable by the hardware used in order to finely quantise the time domain and increase the accuracy in locating the period of the foetal heart beat. This period will lie between 0.25 seconds and 1 second from the assumptions made of maximum and minimum FHR of 4 Hz and 1 Hz respectively. Therefore the ACF need only be computed for the range $\tau=0$ to $\tau=1$ second. A maximum will occur within this section of the ACF when $\tau$ is such that it has shifted the first heart beat cycle in a block of data so that it overlies the second heart beat cycle, the second cycle so that it overlies the third and so on. The value of $\tau$ at this point is the estimate of the foetal heart beat period over the duration of the data block.
Figure 3.3
ACF of a signal consisting of fundamental and second harmonic.

Figure 3.4
ACF of a signal consisting of fundamental and non-harmonic component
Unfortunately the harmonic content of the signal varies considerably with time as in the examples Figures 3.5 and 3.6. These show the time domain waveforms of the demodulated US returns in a 50 Hz bandwidth from the same patient monitoring session. Figure 3.5(i) consists of predominately the fundamental of the foetal heart beat while Figure 3.6 is composed of the fundamental, 3rd and 5th harmonics. The ACF's of these recordings are shown in Figures. 3.5(ii) and 3.6(ii) and in both examples the correlation peak corresponding to the fundamental period \( T_0 \) is clearly visible. With reference to Figure 3.6(ii) in particular, other features to note are the correlation peaks occurring at integer multiples (N) of the fundamental period ie. \( \tau = N.T_0 \). These peaks correspond to the position where the first heart beat cycle in the data block overlies the 3rd \( (N = 2) \) etc. In some circumstances these later peaks can be larger than the peak corresponding to the fundamental. If simply the position of the largest peak in the \( \tau=0 \) to 1 second region of the ACF is taken as the estimate of foetal heart period, this will lead to mistaken larger estimates of the fundamental period at times and hence a corresponding mistaken lower estimate of FHR. This type of error is termed a sub-harmonic error. In the application of this technique in estimating pitch period in speech\[51\] a window function may be applied to the ACF in order to reduce these harmonic errors.

The peaks occurring in Figure 3.6(ii) at positions \( \tau = \frac{T_0}{M} \) \( (M \text{- integer}) \) are due to the presence of the 3rd and 5th harmonics in this particular data block. These harmonic peaks will usually be of smaller amplitude than the correlation peak at the fundamental period however they preclude the use of a peak finding algorithm which just searched for the first maximum in the ACF. If such an algorithm were used a mistaken smaller period would be estimated in these cases giving a correspondingly larger estimate of FHR than actual. This type of error is termed a harmonic error.
Pre-processed signal consisting of predominately the fundamental

The ACF computed for Figure 3.5(i)
Figure 3.6(i)
Signal consisting of first, third and fifth harmonics

Figure 3.6(ii)
The ACF computed for the signal of Figure 3.6(i)
3.3.2. ACF Fundamental Peak Finding Algorithm

The algorithm used to determine the ACF peak corresponding to the fundamental period involves firstly a global search for the largest maximum in the ACF region \( \tau = 0.2 \) to 1 second. A threshold is set at 0.75 of the magnitude of this maximum and the position of first maximum above this threshold is taken as the measure of fundamental period. This threshold must be high enough to ensure the harmonic peaks are below it and low enough to ensure that the fundamental peak can cross it. Since the sub-harmonic peaks should be of comparable size to the fundamental peak and the harmonic peaks of lower amplitude, 0.75 times the global maximum is a reasonable value to take as this threshold. Further confidence of the estimate can be deduced from assessing the signal quality from its ACF. The total power is given by the first value of the ACF ie. \( R_n(0) \) and a threshold can be set to alert the user that a current estimate has been made from a data block of low signal power. Similarly the difference between the values of \( R_n(0) \) and the located fundamental correlation peak gives an indication of the SNR and this can also be used to alert the fact that the estimate has been derived from a poor quality signal.

3.3.3. Signal Bandwidth Considerations

As mentioned two types of error can result from the selection of the wrong correlation peak as the fundamental peak in the ACF:

- **Harmonic errors**: Selection of an ACF caused by harmonics in the signal which is earlier than the true fundamental peak.
Sub-harmonic errors. Selection of a ACF that occurs later in the ACF than the fundamental.

Sub-harmonic errors may be reduced at increased computational cost by using the windowing technique\(^1\) however this increases the harmonic errors as the fundamental peak will be reduced w.r.t the harmonic peaks, hence this technique was not used. The harmonic errors may be reduced by removal/attenuation of the harmonics in the signal by further filtering in the the pre-processing stage. By reducing the bandwidth from 1 to 30Hz to 1 to 4Hz, in the worst case only the 2nd and 3rd harmonics will be present at low FHRs. An undesirable consequence of the reduced bandwidth processing is a smoother ACF with less well defined peaks and an increased bias of the peaks if the signal becomes aperiodic, as discussed earlier in section 3.3. However the number and amplitude of the subharmonic peaks is substantially reduced making the performance of the peak finding algorithm more reliable.

3.4. Signal Processing Development System

The analogue and digital hardware used in analysing the ultrasonic heart valve returns to determine short term FHR, is based on the multi-processor VMEbus. This environment allows multiple 8, 16, or 32 bit processors, memory and devices to communicate via a common VME backplane if they conform to the VMEbus specification. The complete system comprises of the following commercially available VME-boards:

1. Systemforsceuag MP1000 68000 based VME CPU-board.
2. Systemforceung MP1300 776 Kbyte Static RAM / EPROM board.
   This is used to hold utility programs in EPROM, and collected, processed, or down-loaded data and program in battery backed up SRAM.

   This board provides the analogue to digital interface. It gives access to 32 single ended / 16 differential analogue input channels with 12 bit analogue to digital conversion, and two 12 bit digital to analogue output channels.

4. Burr Brown SPV100 TMS32010 D.S.P. VME board
   This is a VME board which uses the Texas Instruments TMS32010 DSP microprocessor. This is used to obtain real-time processing of the signal which the 68000 based CPU fails to achieve for some of the processing techniques. The TMS32010 has a 200ns instruction cycle with most instructions requiring only a single instruction cycle. It has a Harvard architecture with highly pipelined instruction execution which allows a 16 X 16 bit multiply and accumulate in 400ns, compared to the 10 μs multiply time in the case of the 68000. Thus in most DSP applications this gives a factor of at least 20 speed-up over the 68000 or other general purpose microprocessor approach.

A subset of this system consisted of just a FORCE 68000 profi-kit single board computer (SBC) which was used to implement the algorithms which did not required the computational power of the TMS32010. This SBC was used to evaluate some of the algorithms in the hospital.
Software* was developed on a VAX 11/750 running UNIX† or on a TORCH UNICORN UNIX system. Both systems allow programming in the high level language 'C'[53] and compilation to 68000 machine code to run on the MP1000 CPU board. Using the many bit-wise operators available in 'C' and the register declarations which enable variables which are used considerably to be assigned to an actual machine register, machine code can be produced which can be as efficient as hand written assembler. A macro cross-assembler was also written in 'C' to enable machine code to be developed on the UNIX systems for the TMS32010 DSP. This code could be debugged on a Texas Instruments TM32010 evaluation module and altered easily to run on the SPV100 VME board.

The system was used in one of two modes, signal acquisition in which signals were sampled, collected, and transferred to the UNIX systems for off-line analysis, or signal analysis in which the algorithm to determine the short term FHR estimates was run in real-time and the FHR results collected and transferred to the UNIX system for storage and display.

3.5. ACF Processing Implementation

The US signal is preprocessed as described in chapter 2, by amplitude demodulation and bandpass filtering from 1 to 4 Hz followed by a bandpass limiter filtering again from 1 to 4 Hz. The bandpass limiter provides a form of approximate automatic gain control on the demodulated signal. This signal is then sampled and processed digitally.

* Some of this software given in Appendix
† UNIX is a Trademark of Bell Telephone Laboratories, Inc.
3.5.1. ACF Time Domain Block Processing

In this method the CPU assembles the incoming data from the ADC on an interrupt driven basis into blocks of 3 second duration (300 samples at the sample rate of 100 Hz used). The linear unbiased ACF is computed using equation 3.2 for \( \tau = 0 - 1 \) second in steps of 0.01 seconds by sliding the first two seconds of the data block by up to one second.

\[
R_x(k) = \sum_{n=0}^{200} x(k) \cdot x(k+n) \quad k = 0...100
\]

The peak in the ACF corresponding to the fundamental of the fetal heart beat is determined using the peak finding algorithm discussed earlier and the value of shift at which this occurs is logged as the fetal heart period (FHP). The tail one second of the 3 second block is prepended to the next 2 second block before this block is processed. The sample rate of 100 Hz implies a quantisation limitation of \( \pm 1.2 \) bpm at a FHR of 120 bpm. The size of the data block governs the amount of time averaging performed for each estimate of FHR. In this case the short term FHR is estimated over a 2 second interval since 200 samples contribute to any one value in the ACF. Increasing the length of the data block would improve the SNR of the ACF estimate but reduces the ability to track rapid changes in the FHR. In effect if the estimates were perfect the continuous time FHR waveform is being sampled at 0.5 Hz by Nyquist's theorem assuming a sinusoidal variation in FHR between the limits of 1 and 4 Hz

\[
FHR(t) = 2.5 + 1.5 \cos t
\]

the fastest change in rate that can be tracked is

\[
\frac{\partial}{\partial t} FHR(t) = 1.5 \cos t \quad \rightarrow \quad 90 \text{ bpm/sec maximum}
\]
If a larger block size were used this value of maximum tracking rate would be reduced. The maximum deceleration likely to occur in practice is during a condition of foetal cardiac arrest\textsuperscript{[2]} were the fetal heart rate drops from its current rate to an abnormal low rate in a few heart beats. It is important that this condition is detected as early as possible, particularly if it persists beyond about 10 seconds. This is the justification for the processing of 2 second blocks. It is essential that the tracking ability of the algorithm be matched to the slew rate of the signal. This permits optimisation of the conflicting requirements of improving the SNR by processing larger time segments of data and tracking rapid changes in heart rate which requires processing shorter time segments.

The results of this processing on the same patient data analysed earlier using the analogue techniques is shown in Figure 3.7. The effect of the limited sampling rate of 100Hz is clearly evident producing a noticeable quantisation in the FHR estimates. Given this, the plot displays the same acceleration and deceleration features shown in the earlier analysis. However a number of spikes are present due to the peak finding algorithm estimating a subharmonic or harmonic peak as the fundamental.

3.5.2. Thinned ACF Processing

The computation of the ACF for a block of N samples by the previous technique requires $N^2$ multiplications and N-1 additions. In this application however, the signal of bandwidth 1 to 4 Hz is being highly oversampled at the 100 Hz sample rate, thus successive samples are highly correlated. This fact can be exploited to reduce the computational load by amending the algorithm to implement "thinned autocorrelation".
Figure 3.7

FHR plot using ACF processing
The block of N samples is correlated with a "thinned" version of itself by taking only every $M^m$ sample in the original data block

$$T_{m,n} = \frac{N}{M} \sum_{k=0}^{N/M} x(Mk) \cdot x(Mk + n) \quad n = 0.1...100$$

This thinned data set must still meet Nyquist's criterion but adjacent samples are less correlated than those in the highly oversampled data set. This thinning operation is demonstrated in Figure 3.8 which shows the original data set, an example thinned set, and the "Thinned ACF".

Thinning the data by a factor M reduces the number of multiplications required to $\left(\frac{N}{M}\right)^2$ compared to the $N^2$ required in the full ACF. The penalty for this reduced computation is that since fewer data samples are now contributing to each estimated ACF value the variance of these values may increase. Whether this occurs or not depends solely on the underlying statistics of the signal. In general the thinned ACF appears coarser than the full ACF, particularly if the data is thinned to near the Nyquist rate. However this technique can be implemented in real time on simpler microprocessor systems.

3.5.3. Fast Correlation Method

The Wiener-Khinchin relationship, equation 3.16 establishes that the ACF and the Power Spectral Density Function are a Fourier Transform pair.[54]

$$\int_{-\infty}^{\infty} R_{xx}(\tau) \exp(-j\omega \tau) d\tau = S(\omega)$$

3.16
Figure 3.8
(i) Original Data block. (ii) A Thinned data (iii) Thinned ACF
Thus, it is possible to obtain the ACF with reduced computational load by calculating the power spectrum of the signal and inverse Fourier transforming to obtain the ACF. This technique is equivalent to Fast Convolution. Here the time domain output of a filter may be calculated by the pairwise multiplication of corresponding samples of the Fourier transform of the impulse response of the filter by the transform of the input signal, followed by an inverse transform of the result to obtain the output time domain waveform. By replacing the filter impulse response by the input signal itself the ACF of the input signal may be calculated.

\[
R_x(t) = \frac{1}{T} \left( x(t) * x(-t) \right) \\
S(\omega) = \frac{1}{T} X(\omega) X^*(\omega) \\
R_x(t) \xrightarrow{\text{FFT}} S(\omega)
\]

where

- \( x(t) \) is the time domain signal
- \( X(\omega) \) is the Fourier transform of \( x(t) \)
- \( S(\omega) \) is the Power spectrum
- \( \xrightarrow{\text{FFT}} \) denotes a Fourier transform pair.

The Fast Fourier Transform (FFT) algorithm provides an efficient means of transforming a signal from the time domain to the frequency domain and vice-versa. Using this algorithm and the Weiner-Khinchin relationship the ACF can be found with less computation than using the direct approach of equation 3.2. To calculate the linear ACF function for a block of \( N \) samples it is necessary to pad the \( N \) data samples in the block by at least \( N \) zero valued samples otherwise the result will be that of the circular ACF. The FFT sizes required are thus \( 2N \). The number of
multiplications required in implementing this method using a radix-2 FFT to calculate
the linear ACF is

\[ 2N \cdot (4 \cdot N \log_2 N + 5) \]

compared to the \( N^2 \) multiplications required previously, where \( N \) is the number of lags
of the linear ACF to be calculated. Thus for real data and \( N > 32 \) this technique can
offer considerable computational savings (Table 1).

In this application since real (not complex) data is used the transform sizes for both
the forward and inverse transforms can be reduced from \( 2N \) to \( N \) using the Doubling
algorithm.\[36] This reduces the computation further to

\[ 4(N+1) \cdot \log_2 N + 2N \]

Table 1 shows the number of multiplications required for the various methods
discussed and Table 2 presents the times taken to perform the FFT using fixed point
16-bit arithmetic, on various microprocessors for different transform sizes. All times
are in milliseconds.

<table>
<thead>
<tr>
<th>Size</th>
<th>Direct</th>
<th>FFT Method</th>
<th>Doubling Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>( N^2 )</td>
<td>( 2N \cdot \left[ 4 \cdot \log_2 N + 5 \right] )</td>
<td>( 4(N+1) \cdot \log_2 N + 2N )</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>272</td>
<td>124</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>672</td>
<td>304</td>
</tr>
<tr>
<td>32</td>
<td>1024</td>
<td>1600</td>
<td>724</td>
</tr>
<tr>
<td>64</td>
<td>4096</td>
<td>3712</td>
<td>1688</td>
</tr>
<tr>
<td>128</td>
<td>16384</td>
<td>8448</td>
<td>3868</td>
</tr>
<tr>
<td>256</td>
<td>65536</td>
<td>18944</td>
<td>8736</td>
</tr>
<tr>
<td>512</td>
<td>262144</td>
<td>41984</td>
<td>19492</td>
</tr>
</tbody>
</table>
Table 2 — FFT Timings in milliseconds for various microprocessors

<table>
<thead>
<tr>
<th>FFT Size</th>
<th>6809 (1MHz)</th>
<th>68000 (8MHz)</th>
<th>TMS32010 (20MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>17.2</td>
<td>3.65</td>
<td>0.0696</td>
</tr>
<tr>
<td>16</td>
<td>45</td>
<td>8.8</td>
<td>0.1872</td>
</tr>
<tr>
<td>32</td>
<td>108</td>
<td>20</td>
<td>0.4608</td>
</tr>
<tr>
<td>64</td>
<td>260</td>
<td>46</td>
<td>2.42</td>
</tr>
<tr>
<td>128</td>
<td>600</td>
<td>104</td>
<td>5.19</td>
</tr>
<tr>
<td>256</td>
<td>1360</td>
<td>230</td>
<td>11.08</td>
</tr>
<tr>
<td>512</td>
<td>3000</td>
<td>500</td>
<td>23.8</td>
</tr>
<tr>
<td>1024</td>
<td>6800</td>
<td>1100</td>
<td>60.4</td>
</tr>
</tbody>
</table>

The complete method involves the following:

1. Assembling a data block of 200 samples (2 seconds at 100 Hz) on an interrupt driven basis.

2. Padding the data block to be transformed with zeros to 512 points. This is to ensure that the linear ACF is being calculated for the first 1 second of the ACF, since otherwise this technique will produce the circular correlation function. The data is transformed using a 256 point complex radix-2 FFT with the post-processing according to the doubling algorithm, from which the first 256 values of the power spectrum are calculated. This is all that is necessary since, for real data, the power spectrum is symmetric around half the sampling frequency.
3. Inverse transforming the power spectrum which, as it is also real, may also be performed using a 256 point complex IFFT with post-processing according to the Doubling Algorithm. The required result is the 256 values of the linear ACF for $\tau = 0 \cdots 2$ seconds.

4. Determine the peak corresponding to the fundamental of the heart beat cycle from the ACF.

3.5.3.1. Interpolation through Zero-padding

Using the fast correlation technique it is possible to reduce the quantisation of the peak finding algorithm by an alternative method to that of increasing the sample rate discussed earlier. This can be achieved by the use of zero-padding prior to the forward or inverse FFT's. Zero-padding is the inclusion of zero valued data samples to a block of data to increase the overall block length. The above implementation padded the 200 samples in the data block to 512 samples by appending zero valued samples to the block. When zero-padding prior to a forward transform whether the zeros are appended, prepended or inserted either side of the data is immaterial since only the resulting phase of the spectrum will be affected and this is neglected when only the power spectrum is evaluated. The zeros however should not be inserted between samples in the data as this will alter the power spectrum.* The motives behind padding in this example are firstly to ensure that the linear ACF is calculate rather than the circular ACF. This requires that the block be zero-padded by at least its own length, 200 zeros in this case. Also since a radix-2 FFT is to be used the block length is also required to be a size which is an integer power of 2 thus a further 112 zeros are appended giving a block length of 512 ($2^9$) samples. After

* This is explained by Cooley, Lewis, and Welch[56] as theorem 8

\[
\text{Stretch}_k(jX) < \frac{A(n)}{K} n = 0,1 \ldots NK -1
\]
transforming, calculation of the Power Spectrum and inverse transforming, the first 200 real values of the inverse transform give the ACF for the time interval 0 to 2 seconds. However if the power spectrum itself was zero-padded prior to the inverse transform by inserting \((2K-1)N\) (K - integer) zeros at the frequency bin corresponding to half the sampling frequency, then the resulting ACF after inverse transforming will be interpolated by a factor \(K\). Neglecting the precise magnitude of the frequency components this method of zero-padding is similar to having sampled the signal by \(K\) times the original sample rate. The signal is known to be band-limited from 1 - 4 Hz by the preprocessing stage and therefore will have zero valued frequency components from 0 - 1 Hz and 4 Hz to half the sampling frequency. This suggests that the signal could have been sampled at just above the Nyquist rate, a smaller forward transform performed and the power spectrum calculated. Prior to inverse transforming the frequency bins representing 0 to 1 Hz and 4 Hz to half the sample rate should be forced to zero and the zero-padding at half the sample rate performed, expanding the length of the power spectrum so that the ACF is interpolated by the required amount. Figure 3.9 depict the stages involved in this computation. This processing has been implemented in real-time using the 68000 system with a sample rate of 16 Hz, collecting 32 samples per data block calculating a 64 point power spectrum and interpolating by a factor of 16 and using a 1024 point inverse transform. The results of this processing on the patient data with the reduced sampling rate is shown in Figure 3.10 which is similar to that obtained with the 100Hz sample rate.
3. Further zero-padding to 512 points for interpolation

2. 32 point FFT on real data using the doubling algorithm

4. 256 point Inverse FFT with doubling algorithm to obtain ACF.

Figure 3.9
Fast correlation with interpolation
Figure 3.10

FHR plot obtained using Fast correlation with interpolation.
3.5.3.2. Modification to peak finding algorithm

When using the fast correlation method the amplitude of the linear ACF decays to zero as $\tau$ increases since for each shift a sample from the block will be "lost" as it becomes aligned with one of the appended zero-valued samples. This was not the case in the direct time domain method of calculating the ACF since the block size was extended by appending the actual sampled data values to the block not zeros. This type of data block extension cannot be done using the frequency domain method. Instead the peak detection algorithm must be modified to take into account the decaying nature to the ACF. Figure 3.11 shows the principle operation of the modified peak detector. The new algorithm determines the position of the first minimum and this is taken as approximately half of the fundamental period and is used as the default estimate of FHP if no peak is determined by the next stage. The global maximum occurring from this position to $\tau=1$ second is determined and the

![Figure 3.11](image_url)
gradient of the line joining $0.75R_x(0)$ to 0.75 of this determined maximum. This line now acts as the threshold level and the first maximum above the threshold is taken as the fundamental peak.

### 3.5.4. Recursive Autocorrelation

Figure 3.12 shows the structure of a recursive formulation\(^{[57]}\) to compute the ACF by updating the ACF values with each new input sample, equation 3.18.

$$R_x(n) = \beta R_x(n-1) + \alpha x(k)x(k+n)$$  \hspace{1cm} 3.18

$\alpha < 1 \hspace{0.5cm} \beta = 1 - \alpha$

---

Figure 3.12
Recursive ACF structure
This algorithm replaces the block mean estimator by a simple first order recursive estimator,[58] which is the digital approximation to a simple first order RC low pass filter. The coefficient $\beta$ determines the RC time constant of the filter and hence the cut-off frequency of the low pass filter.

$$\beta = 1 - e^{-\frac{L}{\tau}} \approx 1 - \frac{L}{\tau} \quad \tau \gg \text{sampling period} \quad 3.19$$

The use of this recursive equation can be viewed as replacing a sliding rectangular window by a exponentially decaying window. The equivalent block length $L$ of this recursive window relative to the block length of the rectangular window is defined as the inverse of the equivalent noise bandwidth (ENBW) of the window.[59,60]

$$L = \frac{\left[ \sum_{k=0}^{N-1} w(k) \right]^2}{\sum_{k=0}^{N-1} w^2(k)} \quad 3.20$$

where the exponential window is given by

$$w(k) = a \cdot \beta^k \quad 3.21$$

thus

$$L = \frac{2 - \beta}{\beta} = \frac{1 + a}{1 - a} \quad 3.22$$

or rearranging

$$\beta = \frac{L-1}{L+1}, \quad a = 1 - \beta \quad 3.23$$

Thus for an estimate of the ACF over a 2 second interval, at 100 Hz sampling rate, $L=200$ giving $\beta = 0.995, a = 0.005$. This algorithm requires more computation than the previous methods — $2N$ multiplies per input sample where $N$ is the number of ACF values to be computed. However it has the advantage that the equivalent block length over which the estimate is made can be varied easily by changing one
parameter $\beta$. The ACF may also be displayed after each update providing a visual indication of the signal quality. A recursive thinned ACF can also be formulated which reduces the computation in a similar manner to that for the block ACF case.

3.5.5. The Modulus Difference Function

In a processing environment with the absence of a hardware multiplier, multiplications are computational more expensive than simple additions or subtractions typically (for the Motorola 68000 microprocessor) by a factor of 10. An alternative time domain processing algorithm has been proposed [61, 62, 63] which is based on the sum of the modulus of the differences between corresponding signal samples as opposed to the products of corresponding samples used in the ACF. This Modulus Difference function (MDF) is given by:

$$\Phi_s(n) = \frac{1}{N-n+1} \sum_{k=0}^{N-n} |x(k) - x(k+n)|$$

where $|y|$ denotes the absolute magnitude of $y$ i.e., the sign is ignored.

This function now uses $N^2$ subtraction and modulus operations for a block of $N$ samples rather than $N^2$ multiplies. An example of the MDF computed for the sinusoidal signal of Figure 3.13(i) is shown in Figure 3.13(ii). Here the fundamental period is located at a minima in the MDF as compared to a maxima in the ACF. The algorithm used to determine the position of this minimum firstly determines the maximum value of the function in the 0 - 1 second region and the global minimum in the 0.2 - 1 second region. These values are used to set a threshold as in Figure 3.13(ii) and the position of the first minimum below this threshold is taken as the estimate of FHP. Measures of signal quality are more indirect than the signal power and correlation coefficient obtainable from the ACF. However, the value of the global maximum and the minimum at the estimated FHP reflects something of the
Figure 3.13(i)
Sinusoidal signal

Figure 3.13(ii)
Modulus difference function of a sinusoid
signal quality. This method can be used to determine FHR in real-time on simpler microprocessor systems. The results obtained from the application of this technique to the processing of the patient data is shown in Figure 3.14. The thinned and recursive MDF's can be formulated in the same manner as those described for the ACF. A comparison between the MDF and ACF processing shows the MDF to be slightly inferior to the ACF at low SNRs however, this algorithm is more amenable to implementation on simple microprocessors in applications which have limited processing time. Further discussion of this technique may be found in Manning 1986, Appendix A.

3.5.6. Summary

The ACF provides a means by which the SNR of a signal may be improved and thus in the case of FHR measurement better estimates can be made from signal contaminated by noise. The accuracy of these estimates have been shown to be a function of the sampling rate of the signal and the frequency of the heart rate to be measured. A maximum quantisation of ±1 BPM at a FHR of 240 BPM, can only be achieved using sample rates of the order of 1 KHz, two orders of magnitude above the Nyquist rate of the FHR itself. Using the fast correlation method, zero-padding in the Fourier transform can be used to interpolate between the time samples of the ACF to possibly reduce this sampling rate, and retain accuracy. Non-periodic components in the signal can also bias the estimates obtained for the FHR.

There are conflicting requirements in the signal processing. Processing of short data block lengths gives improved tracking of the FHR as it changes but the use of longer data block lengths gives improved SNR but impaired tracking ability. For FHR variability to be represented in the cardiograph the block length should be short
Figure 3.14

FHR plot obtained using MDF processing
otherwise this is obscured by the increased time averaging involved in the processing associated with larger block lengths. However the slower changes of FHR such as the FHR decelerations which occur over a longer time span than the variability changes, more noise robust estimates would be obtained by processing longer data blocks.

For microprocessor implementation the Fast correlation, thinned correlation, and particularly the thinned modulus difference function are computational less expensive than the direct or recursive ACF calculation. However the thinned algorithms are associated with less SNR improvement.

The results obtained using these techniques show quite high quantisation of the FHR estimates, but these are consistent with the expected maximum accuracy of ±3.2 BPM at a FHR of 140 BPM when using a sampling rate of 100Hz. This limitation is set by the 68000 microprocessor hardware used. Modern DSP microprocessor such as the TMS32010 enable faster calculation of the ACF and thus higher sample rates may be used giving improved FHR estimation. However operation at faster sample rates requires more ACF lags to be evaluated to cover the 0.25 to 1 second region of the ACF producing an increasing computational burden.

A major problem with the determination of FHR from these time domain techniques is the identification of the particular peak/dip which corresponds to the fundamental period of the cardiac cycle. A number of spikes are apparent in the FHR plots where either a subharmonic or harmonic error has occurred in the estimation. These may be reduced by using an improved fundamental peak finding algorithm such as comb filtering,[31] or using the past history of FHR estimates to confine the peak search to a defined region of the ACF.[31]
In summary the key elements in the implementation of these time domain algorithms are the choice made concerning data block length, and sampling rate and an algorithm to determine the fundamental period minimising the harmonic and subharmonic errors which can occur.
CHAPTER FOUR

Fourier Transform Frequency Domain Techniques

The fast correlation method described in the previous chapter used the Fourier Transform and Inverse Fourier transform to estimate the FHR in the time domain. It is clear that the estimation of FHR could be made in the frequency domain directly rather than inverse transforming back to the time domain. Since FHR is the measurement of the frequency at which the fetal heart beats it is more natural to make this measurement in the frequency domain as opposed to making the measurement of period in the time domain. The transformation from time to frequency domain is made through the Fourier transform. Considerable research has been undertaken into computationally efficient methods of performing this transformation and several recursive algorithms based on the DFT are presented in this chapter as well as the more commonly used FFT algorithm.

4.1. Considerations of Frequency Domain Processing

As for the correlation processing, the Fourier transform methods can give a SNR improvement by a factor \( N \) where \( N \) is the number of data samples in the transformed block. Thus estimates of FHR made in the frequency domain should be more reliable and noise robust than those made directly from the time domain signal.

The major advantage of working in the frequency domain is the reduction in sampling rate it affords. In applying these techniques an estimate of the FHR can be determined by finding the position of the spectral peak corresponding to the
fundamental component of the foetal heart signal. As this is assumed to be in the range 1 to 4 Hz it is only necessary to compute the Fourier power spectrum for this region, thus it is only required to sample in the limit at 8 Hz from Nyquist's theorem. This represents at least two orders of magnitude reduction on what was required from working strictly in the time domain using correlation processing. The frequency quantisation $f_q$ introduced by using the DFT is given by

$$f_q = \frac{f_s}{N}$$

where $f_s$ is the sampling frequency and $N$ the transform size. From this it is clear that the accuracy with which a single peak in the spectrum may be located is increased by either increasing the transform size $N$ or reducing the sampling rate $f_s$, the lower bound on the sampling frequency being set by Nyquist's theorem. Working solely in the time domain computing either the MDF or ACF it was required to increase the sampling rate to improve accuracy. This accuracy was also a function of the heart rate to be measured whereas using frequency domain methods the accuracy is independent of the heart rate itself. In the case of a single processor system, the reduction in sampling rate reduces the time spent in data collection and correspondingly increases the time available for processing.

The resolution $f_{res}$ of the FFT is given by

$$f_{res} \approx \frac{1}{T}$$

where $T$ is the length of time for which the data has been observed, i.e. $\frac{N}{f_s}$ for a sample rate of $f_s$ and transform size $N$. This resolution sets a limit on the ability of the transform to identify two closely spaced sinusoids. In this application the resolution is not critical since the peak due to the fundamental in the FHR spectrum should be the major single component in the frequency range of interest, although at
times harmonics of the maternal heart rate may be close to that of the foetal heart rate. Similarly the various windowing techniques [61] that are normally associated with the reduction of spectral leakage in the Fourier Transform are not necessary in this application. The performance of the rectangular window (implicitly used) is adequate where only one main spectral component or multiple harmonically related components are present, particularly since the spectrum is viewed on a linear scale. Use can be made of the doubling algorithm once again to reduce the size of transform to be computed by a factor of 2 since the data is real.

4.2. Block FFT Processing

The implementation of this processing technique is identical to the first stages of the fast correlation method of computing the ACF. The preprocessed bandlimited signal is sampled at 10 Hz collecting a block of samples on an interrupt driven basis. The data is zero padded to 512 points in order to decrease the quantisation in the frequency domain. The transformation is performed with a 256 point FFT, using the doubling algorithm. The power spectrum is then calculated for the first 256 points and the global spectral peak present in the 1 to 4 Hz region of the spectrum is then taken as the short term average of the FHR for that data block. The result has a maximum quantisation of

\[ \pm \frac{f_r}{2N} = \pm \frac{10}{1024} \text{Hz} = \pm 0.54 \text{ BPM} \]

as this is the accuracy to which the spectral global peak can be located. Figure 4.1 shows a typical spectrum produced from processing a 2 second block of bandlimited signal. The peaks are of a broad sinc function shape. This is the result of the rectangular window which has a sinc function transform, with the main lobe width (MLW) governed by the number of actual signal samples in the data block. This
MLW is approximately given by:

\[
MLW = \frac{2 \cdot f_s}{M} \text{ Hz}
\]

where \( M \) is the length of the rectangular window (i.e., the number of observed data samples) and \( f_s \) the sampling frequency. This can be obtained from consideration of the convolution property of the Fourier transform with the spectrum resulting from the convolution of the window's sinc function transform with the transform of the signal, a dirac delta function.

\[
\text{rect} \left( \frac{t}{\tau} \right) \quad <--------> \quad \tau \cdot \text{sinc} \left( \frac{\omega \tau}{2} \right)
\]
where $\tau$ is the width of the window. From this it should be clear that the mainlobe width decreases as more samples are included in the data block at the same sampling frequency, the observation period $T$ increases. With a sample rate of 10 Hz, processing 2 second data blocks results in a MLW of 1 Hz whereas processing 8 second blocks the MLW will be only 0.25 Hz, as shown in Figure 4.2. It is the width of this main lobe that limits the resolution in the transform. By increasing the block size better estimates of FHR can be made from poor SNR signals as more samples will add coherently in a single frequency bin. However, as $T$ is increased the tracking ability of the system is diminished as estimates are made over a longer time interval. This point is demonstrated in Figures 4.3(i),(ii) and (iii). A sinusoidal carrier at 2.5 Hz is frequency modulated by another sinusoid of period 20 seconds. The resultant signal simulates a FHR variation from 240 BPM to 60 BPM.

![Figure 4.2](image)

Figure 4.2
Frequency Transform of Rectangular Windows
Figure 4.3(i)
FM Tracking using 2 second block processing

Figure 4.3(ii)
FM Tracking using 4 second block processing
The simulated signal is sampled at 10 Hz. The resulting samples are then processed as described earlier using a 256 point FFT on data blocks of length 2, 4, and 8 seconds. Successive blocks are fully overlapped so that only one new sample is incorporated into the next data block to be processed. An estimate of the short term FHR is produced for each block. The results show a degradation in the tracking performance as the block size is increased, with only the estimates from the 2 second blocks tracking the changes accurately. Figures 4.4(i) (ii) and (iii) show examples of the individual spectra which gave rise to the first estimate of FHR for each of the block sizes. It is clear from these that only the spectrum derived from the 2 second
Figure 4.4(i)
Example Spectra from 2 second block Processing

Figure 4.4(ii)
Example Spectra from 4 second block Processing
data block is representative of the signal's instantaneous power spectrum. This simulated signal has been modelled on what is considered to be an acceptable model for the worst case tracking problem; that of fetal heart block.\textsuperscript{2} In Figure 4.4 the FHR is estimated every 100ms, however if the estimation rate is reduced to every 2 seconds, the impairment in the tracking ability associated with the larger block processing is obscured by this lower sampling rate and all the processing lengths (ie. 2, 4, or 8 secs) will register such dramatic changes in heart rate.

Figure 4.5 show the results obtained from processing the actual patient data with these different block lengths and a FHR sample rate of 0.5 Hz. It can be seen from these that the clarity of this cardiograph is improved as a longer data block is used in the computation of the spectrum. No obvious impairment in tracking ability is discernible.
Figure 4.30
FHR plot using Fourier analysis on successive 2 second data blocks.
Figure 4.5(ii) FHR plot using Fourier analysis on successive 4 second data blocks, overlapped by 2 seconds.
for this data as the block length is increased. A consequence of this processing with longer block lengths is the loss of the heart rate variability information. Ideally this should be measured on a beat to beat basis however as explained early in chapter one the considerable variation in the waveform shape derived from the ultrasonic foetal heart returns and the unknown factor of transducer orientation makes this extremely difficult to obtain from ultrasound monitoring. Nevertheless provided the size of the data blocks from which the FHR is estimated are of the order of a few heart beats some indication of the FHR variability may be obtained. This may suggest the use of two concurrent processes; one providing FHR estimates by working with relatively long data blocks and thus achieving quite high SNR estimates, and another working with shorter data blocks providing estimates suitable for FHR variability analysis. The results obtained from the longer time constant processing could aid in the determination of the correct fundamental spectral peak from the shorter time constant process. The successive spectra of Figures 4.6(i),(ii) have been power normalised for ease of presentation on this waterfall diagram, and show the spectra which gave rise to the results obtained in the FHR plots from approximately 15 minutes onwards in the recording. These should be compared with the autoregressive spectra for the same data segment, presented in the next chapter. Figure 4.6(i) is the data analysed using 2 second block processing, while that for Figure 4.6(ii) is for 4 second block processing, with 2 seconds of data overlap between each successive block. The longer data block reduces the MLW and the FHR changes are more apparent in this diagram.
Figure 4.6(i)
Normalised Fourier Power Spectra from foetal data using 2 second block processing
Figure 4.6(i)

Normalised Fourier Power Spectra from foetal data using 4 second block processing
4.2.1. Data Reconstruction and Median Filtering

Ideally the data of the plots Figure 4.5 should be passed through a reconstruction filter in order to remove the effect of sampled waveform. The reconstruction filter would have a characteristic in the frequency domain of a "brick-wall" filter with a cut-off frequency of 0.25 Hz. This filter would produce interpolated data samples between those actual sampled values to give a clearer presentation of the data. The presence, however, of inevitable incorrect estimates of FHR make accurate reconstruction of the data impossible with these drop-out samples affecting the possibly correct FHR estimates on either side of the incorrect estimate thus making the reconstructed FHR plot worse than the sampled version. In general however the sample rate of 0.5 Hz is very much greater than the Nyquist rate for the accelerations and decelerations that occur in the majority of recordings. Thus reconstruction of the data would not greatly enhance the display of the FHR.

One form of filtering which does improve the presentation of the FHR data is that of median filtering. Median filtering is a non-linear processing technique used in image processing to remove speckled noise. In applying an N point median filter to data a sliding block of N points is assembled and ranked in ascending or descending order of value. The middle point of this ordered N point block ie. the median of this data is taken as the estimate of FHR. The block is moved forward by one sample and the process repeated*. The resulting filtered data has a lower variance than the original data and the number of data spikes (artifacts) produced by incorrect estimates of FHR are reduced. The FHR plot, estimated from FFTs on 2 second data blocks, when median filtered using a three point filter is shown in Figure 4.7. The order N of the median filter determines the number of successive spike samples which

* See Appendix B
may be removed. Care must be taken in the choice of the filter order since if a large
filter order is used the data can be significantly altered. The median filtering at this
stage merely emulates what a doctor viewing such a FHR plot does when encountering
spikes in the data, however a low order filter of 3 or 5 points should be used as higher
filter orders removing more consecutive data spikes could convey a false impression of
the estimates validity resulting in possible clinical misinterpretation. This type of
over-processing should be avoided leaving the doctor with the final interpretation of
the data, irrespective of quality. Again the effect of the median filtering will be to
greatly alter any estimate of FHR variability produced from the median filtered FHR
plot.

To provide the doctor with some indication of the validity of estimates of FHR a
simultaneous plot of the current signal power could be provided as in Figure 4.8.
This can be obtained from the short term power spectra by logging the value of the
peak determined as the fundamental. Estimates of FHR obtained from sections
where the signal power is low can then be treated more dubiously. Better presentation
methods [64] exist to produce a running spectrum display. These may be in the form
of a pseudo-three dimensional (3D) display frequency, power, time or as a grey-scale
plot where the third-dimension of power is represented as the intensity of the darkness
of a pixel. Both these outputing techniques are slow requiring about 12 seconds per
plot for the 3D display. A digital storage oscilloscope was used as a the display
device in the fixed point implementation of the various Fourier Transform techniques.
This enabled the user to obtain visual information concerning the coherence of the
ultrasonic signal by viewing the current power spectrum of the processed signal. This
together with the audible feedback obtained from the loudspeaker output from the
foetal heart detector aided the positioning of the transducer to maintain a good
received signal.
Figure 4.8
Plot of fundamental power estimates associated with the FHR plot of Figure 4.5(i).
4.3. Recursive D.F.T Methods

The efficiency of the FFT algorithm with respect to the direct evaluation of the DFT is reduced when:

1. The data is highly zero-padded. If M data samples are zero-padded to N samples the number of complex multiplications in calculating the DFT can be reduced to M·N whereas a radix 2 FFT still will require \( \frac{N}{2} \cdot \log_2 N \) complex multiplications.

2. Only a particular region of the frequency spectrum is of interest. The FFT will compute all the frequency bins up to half the sample rate whereas using the DFT only that portion of the spectrum required need be determined.

3. Successive data blocks are highly overlapped.

In terms of FHR estimation from the short term Fourier Spectrum all the above factors apply. The data is highly zero-padded with 20 to 80 data samples being zero-padded to 512 samples in order to decrease the frequency domain quantisation interval. Only the 1 to 4 Hz region of the spectrum is of interest and in the previous block processing FFT method successive data blocks were overlapped by as much as 75%. The techniques described in this section are based on recursively updating the short term spectrum on each incoming new data sample. This is effectively working with fully overlapped data blocks. Using these recursive methods an estimate of FHR is obtained from the global peak in the 1 to 4 Hz region of the spectrum at the sample rate of the incoming data. All the techniques as with the FFT algorithm are based on the DFT given by
\[ X(n) = \sum_{k=0}^{N-1} x(k)W_{n-k} \quad n = 0\ldots N-1 \]  \hspace{1cm} 4.5

\[ X = W \cdot x \]

\[ X = [X_0, X_1, \ldots, X_{N-1}]^T \quad \text{Frequency domain vector} \]

\[ x = [x_0, x_1, \ldots, x_{N-1}]^T \quad \text{Time domain vector} \]

\[
W = \begin{bmatrix}
W_N^0 & W_N^0 & W_N^0 \\
W_N^0 & W_N^1 & W_N^1 \\
& & \\
W_N^0 & W_N^{N-1} & W_N^{N-1}
\end{bmatrix} \quad \text{Twiddle Factor Matrix}
\]

where

\[ W_N = \exp\left(-\frac{j2\pi}{N}\right) \]

However rather than exploiting the symmetries of the DFT twiddle factor matrix as in the case of the FFT algorithm, these techniques assume that the data is being input sequentially and the Fourier coefficients are updated with each new sample.

### 4.3.1. Halberstein's Technique

Consider an infinite sampled time series (Figure 4.9) \(x(i)\) \(i = 0, 1, 2, \ldots\) at time instant \(i=k\) a block of \(N\) samples are obtained from which the Fourier coefficients are calculated using the DFT.

\[ X_k(n) = \sum_{i=k}^{N-k-1} x(i)W_N^{i-k} \]  \hspace{1cm} 4.6

At time \(i = k+1\) the new coefficients will be given similarly by:

\[ X_{k-1}(n) = \sum_{i=1}^{N-k-1} x(i)W_N^{i-k-1} \]  \hspace{1cm} 4.7
combining equations 4.6 and 4.7 we obtain

\[ X_{k+1}(n) = W^k N \cdot (X_k(n) + x(N+k) - x(k)) \]  \hspace{1cm} 4.8

This algorithm was presented by Halberstein\[65\] in 1966 for calculating the DFT as can be seen from Figures 4.10.

### 4.3.2. The Goertzel Algorithm

The Goertzel\[66\] algorithm is derived from the DFT by multiplying both sides of equation 4.5 by the factor \( W_N^{-n} = 1 \), thus
Figure 4.10
Structure of Halberstein Algorithm

\[ X(n) = \sum_{k=0}^{N-1} x(k) \cdot W_N^{-(n-k)m} \]

\[ y_n(k) = \sum_{m=0}^{N-1} x(m) \cdot W_N^{-(k-m)m} \]

then

\[ X(n) = y_n(k) \mid_{k=N} \]

Equation 4.11 represents the discrete convolution of the sequence \( x(j) \) with a system with impulse response \( W_N^{-mn} \). This can be represented as a recursive digital filter, similar to that in the Halberstein formulation as shown in Figure 4.10 with transfer function

\[ H_n(z) = \frac{1}{1 - W_N^{-m} \cdot z^{-1}} \]
If the transfer function \( H_n(z) \) is multiplied, numerator and denominator by the factor 
\[
1 - W_n^{-n}z^{-1}
\]
then this filter can be realised using only 2 real multiplications, for each frequency bin 
to evaluate the denominator, as compared with the 4 multiplications involved in the complex multiplication required previously. The Goertzel algorithm, in this form, can be used only in the calculation of the DFT of a block of \( N \) samples since the spectrum is obtained only after the \( N^{th} \) iteration of the filter whereas the Halberstein method produces an updated spectrum of the past \( N \) input samples with each iteration. The recursive calculation of the DFT can be obtained using a modification of the Goertzel algorithm based on the similarity to the Halberstein algorithm and equation 4.12. This is shown in the structure of Figure 4.11.

**Figure 4.11**
Structure of Modified Goertzel Algorithm
Problems arise in the implementation of the Halberstein algorithm in fixed point arithmetic. Each frequency bin output is calculated as the output of a recursive filter which has poles on the unit circle, and thus acts as a perfect integrator. The algorithm becomes unstable as arithmetic errors accumulate with each iteration. Ting [67] modified Halberstein's equation to make it more suitable for implementation in fixed point arithmetic.

4.3.3. Ting's Continuous Fourier Transform

Ting reformulated equation 4.8 by multiplying both sides by the factor $W_n^{(k+1)}$:

$$W_n^{(k+1)} X_{k+1}(n) = W_n^{(k)} X_k(n) + W_n^{(k)}(x(n+k) - x(k))$$

4.14

and introducing a new variable

$$T_k(n) = W_n^{(k)} X_k(n)$$

to give

$$T_{k-1}(n) = T_k(n) + W_n^{(k)}(x(N+k) - x(k))$$

4.15

$T_k(n)$ differs from the Fourier coefficients $X_k(n)$ only by a phase rotation of $W_n^{(k)}$. This can be neglected however where only the power spectrum is of interest as in this application. The structure of this algorithm is shown in Figure 4.12. This expression can be viewed as sliding a rectangular window of width N samples through the time domain data sample by sample with the Fourier coefficients being updated by removing the contributions of the oldest sample which has just dropped out of the window and adding in the contribution due to the new sample entering the window. This algorithm is more stable than equation 4.8 since errors introduced by the calculation when a sample first enters the sliding rectangular window are exactly removed when it leaves the sliding window again N samples later.
This equation can be further modified to compute \( N \) frequency bins from only \( M \) data samples where \( N > M \) i.e. the same effect as zero-padding in block processing.

\[
T_{k+1}(n) = T_k(n) + W_N^{nT^k} - W_N^{nT}x(k)
\]  

4.4. Exponential sliding Window

Rather than using the sliding rectangular window on the time domain data a recursive exponential window as shown in Figure 4.13 can be used given by:

\[
W_k(n) = \alpha \cdot \beta^{n-1+k-1} \quad n = k, k+1, \ldots
\]  

\[
\alpha = 1 - \beta \quad 0 < \beta < 1
\]

This can be implemented as a simple recursive low pass filter with transfer function:
This is the same recursive filter used in the calculation of the recursive ACF in the previous chapter. Examples of this exponential window are shown in Figure 4.14, for different equivalent block lengths. This window results in the older samples being gradually weighted out rather than, as in Ting's algorithm being subtracted out once and for all as the window moves forward in time. Also the resulting transform of a sinusoidal input signal is changed from that of sinc function shown earlier in Figure 4.2 of the sliding rectangular window to that shown in the examples of Figure 4.15. This modification changes equation 4.13 to

\[ T_{k-1}(n) = \beta T_k(n) + \alpha W_{N_k} x(N+k) \]
Figure 4.14
Exponential window

Figure 4.15
Frequency Transform of exponential Windows
and has the structure shown in Figure 4.16. This is equivalent to a baseband analyser in which a local oscillator in this case \( W_n \) demodulates the incoming signal to baseband following which a low-pass filter rejects out-band product terms. In this algorithm the incoming signal sample regarded as a positive frequency phasor is rotated by different amounts by each of a set of negative frequency phasors. One of these will de-rotate (or render stationary) the input phasor. This de-rotated phasor will add coherently to previous de-rotated inputs forming a global peak in the spectrum at the corresponding frequency. With reference to the structure for Ting's CFT algorithm it can be seen that the poles of the transfer function of this structure have been moved from \( |Z| = 1 \) for the CFT to \( |Z| = \beta \) for the baseband analyser.

An alternative algorithm may be formulated by similarly multiplying equation 4.8 by the phase rotation \( W_n \) and defining a new variable

\[
\hat{T}_k(n) = W_n X_k(n)
\]

![Figure 4.16](image)

Structure for Baseband Analyser
to give

\[ \hat{T}_{k-1}(n) = W_n^{-n} \hat{T}_k(n) + x(N+k) - x(k) \]  

4.20

Applying the recursive exponential window, now leads to equation 4.19

\[ \hat{T}_{k+1}(n) = \beta W_n^{-n} \hat{T}_k(n) + \alpha x(N+k) \]  

4.21

where this time the particular phasor is rotated to align with the next incoming sample in the bin. This has the structure shown in Figure 4.17. This structure is very similar to that for the Halberstein algorithm however the poles of the transfer function have been moved from the unit circle in the Z-plane to \( Z = \beta W_n^{-n} \). Both equations 4.17 and 4.18 yield the same results, however equation 4.18 has simpler rotational factors and is easier to implement.

These recursive algorithms essentially compute the full DFT however they have, like the DFT, the advantage over the FFT that the Fourier coefficients are independently calculated and thus the individual frequency channels can be divided to take account of the frequency range of interest and/or to implement a parallel processing structure. Also the effective block length \( L^{[59]} \) of the data can be easily changed by altering the

\[ \begin{align*}
\alpha X(N+k) & \rightarrow X \rightarrow + \rightarrow \hat{T}_{k+1}(n) \\
\beta W_n^{-n} & \rightarrow \end{align*} \]

Figure 4.17

Alternative Recursive structure for Baseband Analyser
factor $\beta$ which for this simple first order case is given by

$$L = \frac{1 + \beta}{1 - \beta}, \quad 0 < \beta < 1$$

as shown early.

4.5. Implementation of recursive DFT Algorithms

Programs were written in 'C' and TMS32010 assembly language to implement the sliding and exponentially weighted recursive DFT. 16-bit fixed point arithmetic was used throughout. The 'C' program was developed to execute on a Motorola 68000 SBC with interfacing hardware constructed to enable the demodulated filtered ultrasonic returns to be sampled by this microprocessor system, and the Fourier Spectrum displayed via a DAC on a digital storage oscilloscope.

This software allowed the user to alter the value of the equivalent block length by changing the exponential decay constant $\beta$, so that the spectrum could be calculated over a longer or shorter period as desired. Using a sampling rate of 10Hz, the spectrum was updated with each new input sample, and the position and value of the global peak identified. This information was then used to update a corresponding position in a further array. This array constituted a histogram whose contents decayed exponentially in a manner similar to the spectrum. Every $\frac{1}{16}$ minute an estimate of FHR was taken from this histogram as the position of the largest value. This array was thus used to hold the occurrence of the FHR estimates obtained for each new input sample over a selected period. Therefore, the final estimate chosen corresponded to the statistical mode of the sample by sample FHR estimates, each being weighted by the power of the spectral peak located. This power was thus used as a confidence measure of the validity of the estimate, and also used to inform the
user of the quality of the input signal by illuminating red, yellow, and green indicator
lights signifying poor, average, and good signal levels respectively. The threshold
levels for the action of these lights could be adjusted by the user.

This microprocessor system and associated interface hardware formed the basis of a
remote home foetal monitoring system, CUESPEC.\cite{34} The ultrasonic foetal heart
returns from the patient in her own home, were received at the hospital over the
public telephone network. This was achieved by placing the telephone mouth piece
close to the loudspeaker output of the ultrasonic foetal heart detector. At the hospital
the analogue signal was inductively coupled from the telephone ear piece and input to
the amplitude demodulation and filtering circuitry previously described before
sampling and processing by the microprocessor system.

The recursive DFT of the signal was computed for the 1-4 Hz region using 164
frequency bins and the FHR estimate and power derived as explained above and
logged every $\frac{1}{16}$ of a minute for the 30 minute duration of the monitoring session.
At the end of this period the stored results were transferred to a microcomputer to be
plotted and for further analysis.

The main problem encountered in the use of the system was in selecting the
appropriate equivalent block length. This involved a compromise between the
tracking ability and noise reduction of the FHR estimates.
4.6. Summary

This chapter has dealt with the determination of FHR using Fourier spectral analysis. By performing the FHR estimation in the frequency domain rather than time domain, the sampling rate may be considerably reduced. The maximum accuracy achievable also becomes independent of the FHR to be measured. Block processing using zero-padded FFTs have been used to determine FHR, and the impairment of the tracking performance of increased data block length illustrated. The display of the FHR plots may be improved using median filtering to remove spikes however this will affect any FHR variability measurement based on the filtered data. A number of recursive DFT algorithms have been presented and implemented in fixed point arithmetic. These can provide a continually updated display of the short term Fourier spectrum. In particular, those based on a sliding exponential window give flexibility in choosing the length of the data block to be used, which governs the tracking ability and SNR improvement of the FHR estimates.
CHAPTER FIVE

Autoregressive Spectral Analysis Techniques

5.1. Introduction

The Fourier Transform techniques covered in the previous chapter, and in particular the Fast Fourier transform, is the most commonly used algorithm for spectral analysis in a wide variety of application areas. The Fourier Transform however is just one example of a parametric approach to signal analysis in which a signal model is postulated and the model parameters determined for a particular data set using the properties of the signal model. The success of such parametric techniques lie in the closeness of the proposed model to the actual data set.

In the case of the Fourier transform the time domain data \( x(0) \ldots x(N-1) \) is modelled as \( N \) sinusoids of harmonically related frequency. For a given data set the amplitudes and phases of these sinusoids are determined by a least squares fit of the sinusoids to the data. An alternative model is to assume the signal is composed of a known number of sinusoids \( P \), of arbitrary amplitude, phase, and frequency. These parameters are likewise determined by a least squares fit of the model to the data. This is the essence of the Pisarenko harmonic decomposition\(^{54} \) algorithm. This may be further modified by replacing the \( P \) undamped sinusoids by damped sinusoids with arbitrary damping factors as proposed by Prony.\(^{68} \) Thus, in the Prony model, the four parameters amplitude, phase, frequency, and damping for each of the \( P \) sinusoids must be determined from the time domain data. In both the Pisarenko and Prony
models the choice of the number of sinusoids present is critical to the of the model's accurate representation of the data.

In this chapter another modelling technique will be presented which originated in economic time series analysis and has found wide application where it may bemiscellaneously refered to as

MEM Maximum Entropy Method (Geophysics)
ALE Adaptive Line Enhancement (Adaptive Filtering)
AR Autoregressive Analysis
LP Linear prediction (Speech Processing)

5.1.1. Chapter Organisation

This chapter is organised as follows.

Section two introduces the Autoregressive Moving average (ARMA), Moving Average (MA), and Autoregressive (AR) process models. It discusses: the problems associated with conventional FFT analysis, and improvements offered by AR analysis, the problems associated with AR spectral estimation, the relationship between the various models, and the determination of the Power Spectral Density (PSD) function from the model parameters.

Section three provides the background theory to the AR parameter estimation algorithms examined and implemented, these include the Yule-Walker Solution, the LMS gradient adaptive filter, the Burg Maximum entropy method, and windowed
Burg methods. It also discusses the problem of selecting an appropriate AR model order, and examines the effects of SNR, and model order in the spectral estimation of a sinusoid in white noise.

Section four provides a simulation study examining the performance of the AR techniques in tracking a frequency modulated sinusoid, as a basis for their use in estimation of FHR.

Section five presents the results obtained in analysing the foetal ultrasound recording to obtain estimates of FHR and the chapter concludes with a summary in section six.

5.2. AR Spectral Analysis

AR Spectral analysis is an alternative technique to the more commonly used Fourier Transform approach to spectral analysis. The AR spectrum has better frequency resolution, and none of the distortion effects caused by the sidelobe leakage from the windowing involved in the Fourier estimation. The length of the data record to be analysed has less of an effect on the AR spectrum than that in the Fourier spectrum. Thus better spectral estimates may be obtained from short data records, as used in the case of FHR estimation. AR analysis produces a smooth spectral estimate not requiring noise averaging as is required to produce the periodogram from Fourier spectra.

Problems associated with AR spectral estimation are

1. Spectral Line splitting\cite{69,70}

   where one actual frequency component becomes represented as two distinct components in the AR spectrum. This occurs usually in the analysis of high SNR data, or data which contain sinusoidal components whose starting or ending phase within the record is an odd multiple of 45°. Its most common cause is
probably the incorrect selection of AR model order with this being chosen to be a high percentage of the data record length.

2. Spurious peaks
Spectral peaks occur indicating frequency components which do not exist in the data. Its cause is generally an incorrect choice of too high a model order.

3. Bias in the spectral peak positions
The frequency of spectral components revealed in the AR spectrum differ from those present in the actual data. This can be as much\[^{70}\] as 16% of the frequency of observation \(\frac{f_s}{N}\) where \(N\) is the number of samples in the data record and \(f_s\) is the sampling frequency.

5.2.1. ARMA, AR, MA Process Models

These models, which are used to depict the data, are drawn from filtering theory. Here the continuous time input/output relationship of a linear time invariant filter can be described in terms of a differential equation relating the output \(y(t)\) and its derivatives to the input \(x(t)\) and its derivatives. In discrete time, the derivatives are replaced by finite differences and in its most general form may be represented as

\[
y_k = \sum_{m=0}^{q} b_m x_{k-m} - \sum_{n=1}^{p} a_n y_{k-n} \tag{5.1}
\]

where \(q\) is the order of the highest derivative of the input \(x(t)\) and \(p\) is the order of the highest derivative of the output \(y(t)\). Thus output of the filter is a linear combination of the present and past \(q\) input samples and the past \(p\) output samples. Using the Z-transform, Figure 5.1 shows this more clearly as a digital filter.
Figure 5.1
ARMA Process Generation Model

comprising an autoregressive* section A(z), and a moving average† section B(z), and is thus overall termed autoregressive moving average (ARMA), where

* It may also be referred to as infinite impulse response or all-pole.
† It may also be referred to as Finite Impulse Response or all-zero filter.
The ARMA transfer function can have both poles and zeros, characterised by the coefficients $a_0, \ldots, a_p$ and $b_1, \ldots, b_q$, respectively. The positions of these poles and zeros determine the spectral response of the filter in terms of both magnitude and phase. As illustrated already, two important subsets of the ARMA filter are the AR filter when $B(z)=1$ and the MA filter when $A(z)=1$.

The ARMA, AR, and MA process generators model the observed data sequence $x(n)$ as the output from the appropriate filter type when driven by a zero mean white noise input process $w(n)$. This results in the flat PSD function of the white noise being coloured by the filter to produce the PSD of the data sequence. The parameters of the model which vary depending on the individual data sequence are the coefficients $a_n$ and $b_m$ and the input noise power $\sigma^2$. The PSD of the general ARMA process output $P(z)$ for a real (non-complex) signal is given by

$$P(z) = \frac{B(z)B(z^{-1})}{A(z)A(z^{-1})} \sigma^2$$
As to be expected, PSD estimates from AR processes which model the data using only poles in the filter produce sharp peaks whereas those from an MA process have characteristic sharp nulls and the ARMA PSD contain both. All these models produce smooth PSD functions unlike the Fourier transform which must be averaged as the periodogram to reduce the noise effects on the spectral estimate.

The Wold decomposition theorem relates all three model types and shows that the ARMA and MA processes can be modelled by an AR process of possibly infinite order* Thus an AR model of suitably high order can still be used to approximately model a process even if it is not strictly AR. An example of this is given in section 5.3. Therefore only the use of the AR model in analysing real data sequences will be considered further.

Having established the model for the data for a given choice of model order the model parameters $a_n, n=1, \ldots, P$ and $\sigma^2$ must be determined from the data and thence the PSD.

5.2.2. Power Spectral Density of an AR process

From equation 5.1 if a data sequence $x(n)$ can be modelled as an AR process order $P$, it will be governed by the following relationship

$$x(n) = - \sum_{k=1}^{P} a_k \cdot x(n-k) + w(n)$$  \hspace{1cm} 5.6

where $w(n)$ is the zero mean white noise driving source and $a_k, k=1,2, \ldots, P$ are the autoregressive coefficients. The AR process generation model is shown in Figure 5.2.

* Similarly an ARMA or AR process may be modelled as a possibly infinite order MA process.
Thus the next data sample $x(n)$ of an autoregressive process can be predicted as a weighted sum of the $P$ past samples and a white noise term $w(n)$. If the input noise has variance $\sigma^2$ then rearranging equation 5.6 and taking the $Z$ transform

$$X(z) = \frac{\sigma^2}{A(z)}$$  

5.7

The PSD $S(\omega)$ of the data $x(n)$ can be determined by evaluating the $Z$ transform around the unit circle in the $Z$ plane.

$$S(\omega) = \frac{\sigma^2}{|A(z)|^2}_{z=e^{j\omega}}$$  

5.8

$$S(\omega) = \frac{\sigma^2}{\left| \sum_{k=0}^{P} a_k \exp(-j\omega k) \right|^2}$$  

5.9

$\rho_0 = 1$
The denominator in equation 5.9 is simply the squared magnitude of the Fourier transform of the impulse response of the AR filter and may be evaluated using the DFT or, with zero-padding, the FFT algorithm.

It is clear that the computation of the AR PSD is more intensive than that involved in determining the Fourier power spectrum, since as well as finding the AR coefficients, a Fourier transform of these coefficients and a further division for each required frequency point must be performed. If the FFT of the zero-padded AR coefficients is used to calculate the denominator of equation 5.9 then some of the redundancy of the zero-padding can be exploited to reduced the computation.[72] It should also be noted that the AR PSD differs from the Fourier power spectrum in that the measure of power is obtained as the area under the PSD and not the actual magnitude as in the case of the FFT. For the case of well resolved sinusoidal frequency components, and where power estimates are required, the following methods may be employed:

1. Numerical integration between appropriate limits around the spectral peak of interest.[73]

2. Where the relative power distribution over a region is of interest rather than the actual magnitude the product of peak height and bandwidth may approximate the area measure or alternatively the square root of the peak value may be used.[73]

3. Determine the roots of the polynomial A(z) and calculate the power using the residues of the roots close to the unit circle in the Z plane.[74]

In many applications, including the current one of determining FHR, only the position of the spectral peaks and their relative amplitude are of interest and the AR PSD may be modified as
\[ \hat{S}(\omega) = \frac{1}{\left| \sum_{k=0}^{\rho} a_k \exp(-j\omega k) \right|^2} \]

which retains the frequency information but expends with the absolute magnitude information.

### 5.3. AR Parameter Estimation Algorithms

A number of algorithms\[^{75,76,77,79,80}\] are available from which the AR parameters may be determined for a particular data sequence. These may take block or sequential forms, and vary in complexity and performance. Algorithms which have been examined and implemented to approach the problem of FHR estimation are

1. Yule-Walker Equation Solution\[^{79}\]
2. LMS Gradient Transversal Filter\[^{81}\]
3. Burg Maximum Entropy Method\[^{82}\]
4. Gradient Lattice Algorithm\[^{83}\]
5. Optimum Tapered Burg Algorithm\[^{84}\]
6. Energy Weighted Burg Algorithm\[^{85}\]

These algorithms are generally based on the least mean square fit of the data sequence to the AR model to determine the AR coefficients. Algorithms based on the least squares estimation offer improved performance over these algorithms but at added computational cost.
5.3.1. Yule-Walker Solution

For a real stationary AR process of order \( P \) and input white noise \( w(n) \) with variance \( \sigma^2 \)

\[
x(n) = - \sum_{m=1}^{P} a_m x(n-m) + w(n) \tag{5.11}
\]

Multiplying both sides by \( x(n+i) \) and taking expectations

\[
E[x(n) x(n+i)] = - \sum_{m=1}^{P} a_m E[x(n-m) x(n+i)] + E[w(n) x(n+i)] \tag{5.12}
\]

The final term in equation 5.12 can be expanded as

\[
E[w(n) x(n+i)] = E[w(n) h_i w(n+i)] \tag{5.13}
\]

where \( h_i \) is the impulse response of the AR IIR filter

\[
E[w(n) x(n+i)] = E[w(n) \sum_{j=0}^{\infty} h_j w(n+i-j)]
\]

where \( h_j \) is the impulse response of the AR IIR filter

\[
= \sum_{j=0}^{\infty} h_j \sigma^2 \delta(i+j) = \sigma^2 h_i
\]

Since the process is causal,

\[
h_i = \begin{cases} 
0 & i > 0 \\
\sigma^2 & i = 0 
\end{cases}
\]

as \( h_0 = \lim_{i \to \infty} H(z) = 1 \)

Thus

\[
R(i) = - \sum_{m=1}^{P} a_m R(i-m) + \sigma^2 \delta(i) \tag{5.14}
\]

where \( R(i) \) represents the \( i^{th} \) autocorrelation lag and \( \delta(i) \) is the discrete delta function.

Thus if a process is autoregressive its autocorrelation function is also autoregressive.

Equation 5.14 define a set of linear simultaneous equations termed the Yule-Walker equations, the solution of which determines the AR coefficients from the autocorrelation lags. As the ACF for a stationary signal is an even function
\( R(-i) = R(i) \)

this autocorrelation matrix takes a special form of a symmetric Toeplitz matrix.

\[
\begin{bmatrix}
R(0) & R(1) & \ldots & R(P) \\
R(1) & R(0) & \ldots & R(P-1) \\
R(2) & R(1) & \ldots & R(P-2) \\
\vdots & \vdots & \ddots & \vdots \\
R(P) & R(P-1) & \ldots & R(0)
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_P
\end{bmatrix}
= \begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix}
\]

Because of this, the simultaneous equations can be solved most efficiently using the Levinson algorithm as opposed to using matrix inversion techniques such as Gaussian elimination or Cholesky decomposition. The Levinson algorithm calculates the AR parameters for ascending model orders (m) from the ACF values as

\[
a_{1,1} = -\frac{R(1)}{R(0)}
\]

\[
\sigma^2 = (1 - a_{1,1}^2).R(0)
\]

For each successive model order \( m = 2, \ldots, P \)

\[
a_{m,i} = a_{m-1,i} + a_{m,m} \cdot a_{m-1,m-i} \quad i = 1, 2, \ldots, m-1
\]

\[
\sigma_m^2 = (1 - a_{m,m}^2).\sigma_{m-1}^2
\]

At each iteration the coefficients along the rows of the AR parameter matrix \( a_{1,1}, a_{2,2}, \ldots, a_{m,m} \) and the residual error power \( \sigma_m^2 \) represent the estimates for the AR model parameters for model order \( m \). The coefficients along the diagonal, \( a_{1,1}, a_{2,2}, \ldots, a_{m,m} \) are termed the partial correlation (PARCOR) coefficients, or reflection coefficients.\(^{[86]}\) For the autocorrelation matrix to be valid and the AR coefficients to realise a stable minimum phase causal filter\(^*\) it is a necessary condition

\(^*\) This is a key requirement in speech synthesis for vocal tract modelling how-
that $|a_{m,m}| \leq 1$ for $m=1,2,\ldots,P$. The ACF values may be calculated from any of the equations given in chapter 3, however for short data records it is advisable to use the unbiased ACF estimate. As will be seen in section 5.4 the residual noise power $\sigma^2$ is an important parameter in selecting an AR model order which is appropriate for the given data.

5.3.2. Linear predictors

The AR generation model has an IIR structure as shown in Figure 5.2 and thus will place poles in the Z plane at the appropriate positions to colour the input white noise sequence so as to approximate the PSD of the given data. For this to be a realisable, stable, causal filter the poles of $A(z)$ must lie within the unit circle and correspondingly the poles of $A(z^{-1})$ must lie outside the unit circle. One approach to the determination of the AR parameters is to find the coefficients of the inverse filter which, when driven by the data sequence, will produce a white noise output. To whiten the signal the inverse filter must place zeros in the Z plane at the pole locations introduced in the generation model. The inverse filter can take two basic FIR forms, that of the forward predictor Figure 5.3 or the backward predictor Figure 5.4. This corresponds to implementing the whitening filter as either the causal $A(z)$ or the anti-causal $A(z^{-1})$.

$$A(z) = 1 - a_1z^{-1} - a_2z^{-2} \ldots - a_pz^{-p}$$
$$A(z^{-1}) = 1 - a_1z^{-1} - a_2z^{-2} \ldots - a_pz^p$$

$$= z^p \cdot (z^{-p} - a_1z^{-p+1} - a_2z^{-p+2} \ldots - a_p)$$

ever is not of consequence in spectral estimation.
Figure 5.3
One Step Forward Predictor

Figure 5.4
One step Backward Predictor
For a real stationary signal the backward predictor coefficients of Figure 5.4, $b_0, \ldots, b_p$ are simply the time reversed forward predictor coefficients.*

$$b_m = a_{P-m} \quad m = 0, 1, \ldots, P$$

5.17

The forward predictor is the direct implementation of the AR generation model shown earlier equation 5.11 in which the negative of the predicted next sample is calculated as a weighted sum of the $P$ past samples to which the actual next sample is added, just leaving a white noise term. Similarly the backward predictor is the direct implementation of the alternative form of the AR generation model $A(z^{-1})$ in which a past sample is predicted from it’s $P$ future samples. Both these predictors are one step predictors, predicting the next future or past sample.

For pure AR processes there is no necessary constraint on the predictor to be one step. In some applications[87, 88] improvement in AR spectral estimation can be achieved by using a larger predictor delay than that used in the single step predictor. In the case of the one step predictor the single delay is all that is required to decorrelate the white noise component in the input signal, however in data analysis where coloured noise is a problem this delay can be optimised to attempt to decorrelate the coloured noise, and a larger step predictor used. The delay, however, can only be optimised for a particular frequency of interest and as a result its use tends to be limited to the enhancement of known fixed frequency sinusoids in coloured noise.

Once the whitening filter has been determined in either of these forms the coefficients of the appropriate AR generation model will correspond to those of the whitening filter.

* For a complex stationary signal the backward predictor coefficients are the complex conjugated time reversed forward predictor coefficients.
5.3.2.1. Example of whitening a sinusoid

As an example of this whitening process consider a real sinusoid

\[ x(k) = A \cdot \cos \left( \frac{2\pi \omega k}{\omega_s} \right) \]

where

\[ \omega \] is the frequency of the sinusoid.

\[ \omega_s \] is the sampling frequency.

In order to generate this sequence from white noise the AR model would place poles in the z plane at

\[ z = \cos \left( \frac{\omega}{\omega_s} \right) \pm j \sin \left( \frac{\omega}{\omega_s} \right) \]

as shown in Figure 5.5. With poles on the unit circle the IIR filter constitutes an oscillator. In order to whiten the signal, the inverse FIR filter must place zeros at the pole locations, and thus takes the form of a notch filter as shown in Figure 5.6.

In this example since the poles were on the unit circle and hence the AR coefficients are symmetric, the forward and backward predictors are identical. However, for a more general example of a damped sinusoid shown in Figure 5.7, with the PSD given by

\[ S(\omega) = \frac{\sigma^2}{A(z)A(z^{-1})|_{z=e^{j\omega}}} \]

as before, the AR model transfer function \( \frac{1}{A(z)} \) could take any of the following forms:
Figure 5.5
AR generation model for a real sinusoid

Figure 5.6
Whitening Inverse filter for a sinusoid

Figure 5.7
Pole positions for AR model of damped sinusoid
where \( q_k = \frac{1}{p_k} \) \( k=1,2 \) Resulting in the AR generation model taking forms of minimum phase with all the poles located inside the unit circle, through non-minimum phase with a mixture of poles inside and outside the unit circle, to maximum phase with all the poles outside the unit circle. Likewise the inverse filter could take any of the above forms placing a zero either inside or outside the unit circle to cancel one of each pole pair \( p_k, q_k \). The corresponding function \( A(z^{-1}) \) will place a zero over the remaining pole of the pair to whiten the PSD.

5.3.3. LMS Gradient Transversal Filter

This algorithm is one of the simplest of a wide range of adaptive filtering algorithms that may be used in a system identification mode to determine the AR coefficients from a given data sequence. It found early application in the measurement of instantaneous frequency, and in spectral analysis.\[81,89,90,91\]

Consider the model order \( P \), one step forward predictor of Figure 5.3, with time-varying AR coefficients \( a_m(n) \) \( m=1,...,P \). An estimate of the next input sample \( \hat{x}(n) \) is given in matrix form by

\[
\hat{x}(n) = -A_n^T x_n = -x_n^T A_n
\]

where

\[
A_n^T = [ a_1(n), a_2(n), \ldots, a_m(n), \ldots, a_P(n) ]
\]
The forward prediction error $e(n)$ is thus

$$e(n) = x(n) - \hat{x}(n) = x(n) + A_n^T X_n$$

The mean square value of $e(n)$ will be minimum when the coefficient vector $A_n$ corresponds to that of the AR generation model which gave rise to the data sequence $x(n)$, and $e(n)$ is white noise. Hence

$$E\left[ e^2(n) \right] = E\left[ (x(n) + A_n^T X_n)^2 \right] \leq E\left[ x^2(n) \right] + 2E\left[ x(n)X_n^T \right]A_n + A_n^T E\left[ X_n X_n^T \right] A_n$$

defining

$$P_n = E\left[ X_n X_n^T \right] \text{ — the autocorrelation vector.}$$

$$R_n = E\left[ X_n X_n^T \right] \text{ — the autocorrelation matrix.}$$

then

$$E\left[ e^2(n) \right] = E\left[ x^2(n) \right] + 2P_n A_n + A_n^T R_n A_n = 0$$

Minimising this w.r.t $A_n$

$$\nabla_n = \frac{\partial}{\partial A_n} E\left[ e^2(n) \right] = 2P_n + 2R_n A_n = 0$$

$$A_n^* = -R_n^{-1} P_n$$

$A_n^*$ is the Weiner optimum solution for the coefficient vector. The LMS algorithm uses the method of steepest descent[92] to iteratively update the current coefficient vector $A_n$ with each new input sample. Ideally an amount proportional to the negative gradient $-\nabla_n$ of the mean square error surface defined in equation 5.20 along the $a_m$ coordinate should be added to the current value of $a_m$ to reduce the mean square error and approach its minimum value. Since the mean square error
gradient is unknown it is estimated as the instantaneous squared error gradient although this results in slower convergence of the algorithm in the presence of contaminating coloured noise in the signal.

\[
\hat{\mathbf{v}}_n = \frac{\partial}{\partial \mathbf{A}_n} e^2(n) = \frac{\partial}{\partial \mathbf{A}_n} (x(n) + \mathbf{A}_n^T \mathbf{X}_n)^2 \\
= 2(x(n) + \mathbf{A}_n^T \mathbf{X}_n) \mathbf{X}_n \\
= 2 e(n) \mathbf{X}_n
\]

thus

\[
\mathbf{A}_{n+1} = \mathbf{A}_n - \mu \hat{\mathbf{v}}_n
\]

Convergence of \( \mathbf{A}_n \) to \( \mathbf{A}_n^* \) is guaranteed for a stationary signal provided

\[
0 < \mu < \frac{1}{\lambda_{max}}
\]

where \( \lambda_{max} \) is the largest eigenvalue of the autocorrelation matrix \( \mathbf{R}_n \). For implementation purposes it is more convenient to use

\[
\mu = \frac{\alpha}{P \cdot R(0)} \quad 0 < \alpha < 1
\]

where \( P \) is the AR model order and \( R(0) \) is the mean power of the data sequence \( x(n), \ldots, x(n-P) \). This likewise guarantees convergence. More detailed discussion on the convergence properties of the LMS algorithm may be found in the literature.\[93,92,94,81\]

The selection of the value of \( \alpha \) determines the speed of convergence of the LMS algorithm and also the mis-adjustment of the weight vector, ie. its variance about the optimal value. The adaption time constant\[90,95\] \( T_a \) measured in number of iterations is given approximately by
\[ T_a = \frac{-1}{\ln \left( 1 - \frac{\alpha}{P} \right)} \]  

thus

\[ \alpha = P \left( 1 - e^{-\frac{1}{T_a}} \right) \]

This provides a means of determining an appropriate value for \( \alpha \) given that convergence should be obtained within a given number of iterations.

The variance of the weights \( \alpha_m(n) \) about their mean optimal value is also dependent on the chosen adaption time \( T_a \) and results in excess mean square error. The misadjustment\(^{[93]} \) \( M \) is measured as a ratio of this excess mean square error to the minimum mean square error and is given approximately by

\[ M = \frac{P}{4T_a} \]

As can be seen the choice of a long adaption time results in a smaller mis-adjustment of the weight vector but a slower convergence. Once converged the PSD can be calculated from the weights \( \alpha_m(n) \) using equation 5.9. The program used to implement the LMS Adaptive line enhancement algorithm also given in Appendix B.

5.3.4. Maximum Entropy Method — Burg Algorithm

The LMS algorithm may be used with either a forward or backward predictor structure to obtain the AR coefficients, for a given model order. These predictors implement the all-zero inverse filter \( A(z) \) in their direct form as FIR filters. An alternative to this realisation is the lattice filter which incorporates both forward and backward predictor types in the one structure shown in Figure 5.8. This type of filter is often termed a prediction-error filter since it is the forward and backward prediction
Figure 5.8
Structure of the Lattice Filter
errors for increasing model orders which propagate along the lattice rather than the actual data samples. The Maximum Entropy PSD is equivalent to the AR PSD in the restricted case of a stationary Gaussian process.[71]

Consider a block of real data \(x(0), \ldots, x(N-1)\) originating from a stationary process. The one step forward prediction error \(f_m(n)\) for a order \(m\) predictor may be defined as[96]

\[
f_m(n) = \sum_{i=0}^{m} a_{m,i} x(n-i) \quad n = m, \ldots, N-1
\]

Similarly the one step backward prediction error \(b_m(n)\) may be defined as

\[
b_m(n) = \sum_{i=0}^{m} a_{m,i} x(n-m+i) \quad n = m, \ldots, N-1
\]

Since the process is stationary the AR coefficients must satisfy the Levinson recursion, equation 5.16. Using this recursion it can be shown that the forward and backward prediction errors are related by the recursive lattice equations depicted in Figure 5.8

**Initialisation**

\[
f_0(n) = b_0(n) = x(n) \quad n = 0, \ldots, N-1
\]

*For each model order* \(m = 1, \ldots, P\)

\[
f_m(n) = f_{m-1}(n) + K_m b_{m-1}(n-1)
\]

\[
b_m(n) = b_{m-1}(n-1) + K_m f_{m-1}(n) \quad \ldots \quad n = m, \ldots, N-1
\]

where

\[
K_m = a_{m,m}
\]

\(K_m\) are termed the PARCOR or reflection coefficients.[86]
Burg\textsuperscript{[97]} minimised the sum of the forward and backward prediction errors powers

$$E\left[f_m^2(n) + b_m^2(n)\right]$$ \hspace{1cm} 5.34

w.r.t $K_m$. Substituting equations 5.33, differentiating w.r.t $K_m$ and equating to zero to find the function minimum gives

$$K_m = -\frac{2E\left[f_{m-1}(n) b_{m-1}(n-1)\right]}{E\left[f_{m-1}^2(n) + b_{m-1}^2(n-1)\right]}$$ \hspace{1cm} 5.35

For the given data sequence the expectations in equation 5.35 may be approximated by summations to give

$$K_m = \frac{-2\sum_{n=m}^{N-1} f_{m-1}(n) b_{m-1}(n-1)}{\sum_{n=m}^{N-1} f_{m-1}^2(n) + b_{m-1}^2(n-1)}$$ \hspace{1cm} 5.36

The summations in the equation 5.36 start at $m$ since the prediction errors are inaccurate until the one step predictors of order $m$ have undergone their initial transients of $m$ input samples.

Other functions\textsuperscript{[98,99]} of the forward and backward prediction error powers may be minimised leading to different expressions for $K_m$ however the Burg minimisation is guaranteed to produce a stable minimum phase filter, as the previously stated condition for stability, \( |K_m| \leq 1 \) is ensured by equation 5.36. The Burg PARCOR estimates can be shown to be the harmonic mean of the separate PARCOR estimates obtained by minimising the forward and backward prediction errors individually.\textsuperscript{[99]}

The AR PSD can be determined from the PARCOR coefficients by first transforming them into their equivalent AR coefficients. This can be done using the Levinson recursion equation 5.16 replacing the Yule-Walker estimates of the PARCOR
coefficients $a_{m,m}$ with the Burg estimates. The PSD can then be calculated as before, using equation 5.9.

The algorithm can be modified from this block processing form to a sequential form by either:

1. Replacing the block summations in equation 5.35 by the simple recursive estimator[58] as used previously in Chapter 3 to estimate the ACF recursively and in Chapter 4 to calculate the DFT recursively.

$$C_m(n) = \beta . C_m(n-1) - 2.\alpha . f_{m-1}(n) . b_{m-1}(n-1)$$

$$D_m(n) = \beta . D_m(n-1) + \alpha . ( f_{m-1}^2(n) + b_{m-1}^2(n-1) ) \quad m = 1, \ldots, P$$

where

$$\alpha = 1 - \beta \quad 0 < \alpha < 1$$

where

$C_m(n)$ is the estimate of the mean cross-correlation between $f_{m-1}(n)$ and $b_{m-1}(n-1)$. The numerator of equation 5.35.

$D_m(n)$ the estimate of the mean of the sum of the forward and backward prediction powers entering at the $m^{th}$ lattice stage. The denominator of equation 5.35.

$m$ is the model order index (or lattice stage index),

$n$ is the time index.

$\alpha$ determines the equivalent block length of the recursive estimator as from equation 3.22 earlier.
2. Use the method of steepest descent to implement a gradient lattice filter,\cite{96,83} in a similar manner to that of the LMS gradient transversal filter. The error surface of which the minima must be sought is now that of the sum of the forward and backward prediction error powers.

Both these methods can be shown to be equivalent.\cite{99}

5.3.5. Weighted Burg Algorithms

The Burg Algorithm has been shown\cite{69,100,85,84} to exhibit bias in the estimates of the peak positions of sinusoidal frequencies particularly for short data records, where the starting phase of the sinusoid is shown to influence the bias of the peak. Windowing techniques may be used to overcome this problem. The sum of the windowed forward and backward prediction error powers is minimised.

\[ E \left[ W_m(n).\left( f_n^2(n) + b_n^2(n) \right) \right] \] 5.37

The window \( W(n) \) may take many forms which may be either fixed\cite{59} or data dependent. For the case of the fixed coefficient type, Swingler\cite{101} first showed that the bias could be reduced by using a Hamming window.

\[ W_m(n) = 0.54 - 0.46\cos \left( \frac{2\pi(n-m)}{N-m} + \frac{\pi}{N-m} \right) \quad n = m, \ldots, N-1 \] 5.38

However a theoretical "optimum"\cite{102} parabolic window proposed by Kaveh\cite{84} which minimises the mean variance of the estimated frequency of a real sinusoid and is given by

\[ W_m(n) = \frac{6(n-m+1)(N-n)}{(N-m)(N-m+1)(N-m+2)} \quad n = m, \ldots, N-1 \] 5.39
or recursively as

\[ W_m(m-1) = 0 \]

\[ W_m(m) = \frac{(N-m)\lambda}{2} \]

\[ W_m(n) = 2W_m(n-1) - W_m(n-2) + \lambda \quad n = m+1, \ldots, N-1 \]

\[ \lambda = \frac{12}{(N-m)(N-m+1)(N-m+2)} \]

Scott\textsuperscript{[85,103]} formulated a data dependent window (equation 5.40) which used the power in the prediction error filter to weight the forward and backward prediction errors individually. The forward and backward PARCOR coefficients are determined separately using this weighting and combined as a harmonic mean to obtain the equivalent Burg estimate.

**Forward Predictor Window**

\[ W^f_m(n) = \sum_{i=1}^{m} x^2(n-i) \]

**Backward Predictor Window**

\[ W^b_m(n) = \sum_{i=1}^{m} x^2(n-m+i) \quad n = m, \ldots, N-1 \]

The programs used to implement the optimum tapered Burg algorithm and this energy weighted Burg algorithm are given in Appendix B.

### 5.4. AR Model Order selection

The problem of choosing an appropriate AR model order for a given data set is important since this choice can significantly affect the performance of the spectral estimation. Various criteria\textsuperscript{[54]} are available on which to base initial estimates of model order. All of these involve examination of the residual error power, \( \sigma_m^2 \), as it diminishes with increasing model order \( m \). In a pure AR process \( \sigma_m^2 \) will reach a
minimum value at the correct model order, and will then remain constant even if higher order estimators are used. AR parameter estimation algorithms which involve the Levinson recursion use a model order update for this error power as

$$\sigma_m^2 = (1 - a_{m,m}^2)\sigma_{m-1}^2$$

The above equation shows that provided $a_{m,m} \neq 0$, this error power to be a monotonically decreasing function with increasing model order, since $|a_{m,m}| \leq 1$ is a necessary condition for a valid autocorrelation matrix and the realisation of a stable causal filter. Various algorithms have been proposed which are functions of error power $\sigma_m^2$, model order $m$ and the number of samples $N$ in the block of data to be analysed. These have been found to work well with simulated AR processes but tend to underestimate the model order when used to analyse actual data.[68] However, they may be used to provide initial estimates for the model which can be improved on empirically.

Factors which must be considered in selecting an appropriate model order are

**Spectral content**

If the number of sinusoidal frequency components present in the data is known the initial model order should be chosen to be at least twice the number of components since each real sinusoid will required two poles in the AR generation model.[69]
Spectral Resolution

If the spectral components are closely spaced a higher order model should be used in order to resolve these components since with only a few available poles the AR generation model will tend to model two closely spaced sinusoids with a single pole pair, thus leaving them unresolved.

Signal to Noise Ratio

At low SNR a higher order model will be required in order to maintain frequency resolution, since sinusoids in white noise constitute an ARMA process and thus require to be modelled using a high order AR process.

Number of data samples to be analysed

When based on a small data segment, AR estimates from a high order model tend to become unstable producing spurious peaks in the spectra. In this case as a general guide, a model order should be chosen between $\frac{N}{3}$ and $\frac{N}{2}$ where $N$ is the number of samples in the segment. The effect of data record length on the estimates of the PARCOR coefficients has been examined by Kay and Makhoul.[104]

Assuming a known ACF, the effect of SNR and model order $P$ on the resolution $\delta f$ (normalised to the sampling rate) of two equi-amplitude sinusoids in white noise has been quantified by Marple[68] as

$$\delta f = \frac{1.03}{P [ \text{SNR} \cdot (P + 1)]^{0.31}}$$
where SNR is measured linearly not in dB. This shows that the ability to resolve the sinusoids with a fixed order model decreases as the SNR decreases. A similar empirical rule is given by Linkens,[105] from observations in analysing biomedical data. He states that the model order should be greater than twice the number of samples in one beat period of the signal.

\[ P > \frac{2}{B_f} \]  

5.43

It is tempting to select high order AR models to alleviate the effects of poor SNR. This can however, result in the classic AR phenomena of spectral peak splitting\(^{[69]}\) and spurious peaks.

### 5.4.1. AR Estimation of a sinusoid in white noise

The estimation of a real sinusoid at various SNR's and model orders is considered, in order to illustrate the effect of model order and SNR on the AR PSD. This example is a reasonable approximation to the signal under investigation to determine FHR.

The true ACF of the sinusoid is used with the total signal power \( R(0) \) being increased by \( \sigma^2 \) to include the required level of white noise. Thus

\[ R(0) = 1 + \sigma^2 \]
\[ R(k) = \cos \left( \frac{2\pi k}{\omega} \right) \quad k = 1, \ldots, P \]

A sinusoid in white noise can be shown to be an ARMA process\(^{[54]}\) and hence requires more than the initial estimate of two poles to model it accurately as an AR process. The AR parameters are calculated from the theoretical ACF using the Yule-Walker equations* and the roots of the polynomial \( A(z) \) are then determined,\(^{[106]}\)

* For this case the Burg Algorithm produces the same estimates\(^{[104]}\) for the PARCOR coefficients as the Yule-Walker solution as a data segment of infinite length is implicitly assumed in using the theoretical ACF.
Figure 5.9 shows the estimated location of the poles of the AR process, for the model order 2 case when estimating a sinusoid in white noise at SNR's of 20 dB to 0 dB in steps of 2 dB. The corresponding normalised AR PSD are also shown. Figures 5.10 to 5.12 repeat the above for model orders of 4, 8, and 16. As the SNR decreases the spectral peaks increase in bandwidth. This is as a result of the trajectory of the poles from close to the unit circle in towards the centre of the unit circle. This is most marked in the case of the two pole model which effectively fails to resolve the positive and negative frequency phasor components of the real sinusoid at 0 dB SNR. In doing so the 2 pole model degenerates from a bandpass filter response to that of a low pass filter. The closer these phasors are together ie. the sinusoidal frequency being close to D.C or to half the sampling frequency, the harder it is to resolve them in the presence of the additive noise. With increased model order the spectral peaks become sharper although more ripple is introduced as the extra poles are placed approximately equi-distanced around the unit circle to model the white noise floor. This floor level increases as the SNR decreases since the total power has increased while the power in the sinusoid has remained unchanged. As a result of this increase in the white noise power, the area under the PSD estimate must increase accordingly. With higher order models, the peak positions become more distinct and less biased by the effect of the noise. Figure 5.13 shows how the residual noise power changes as a function of model order. The noise power rapidly decreases as the model order increases initially but after model order 6 there is only further gradual reduction.
Figure 5.9
Pole Trajectories and spectra for sinusoid in white noise using a 2 Pole AR Model.
Figure 5.10
Pole Trajectories and spectra for sinusoid in white noise using a 4 Pole AR Model
Figure 5.11
Pole Trajectories and spectra for sinusoid in white noise using a 8 Pole AR Model.
Figure 5.12
Pole Trajectories and spectra for sinusoid in white noise using a 16 Pole AR Model
5.5. AR Analysis to determine FHR

5.5.1. FM tracking of a sinusoid in white noise

To compare the frequency tracking performance of the various AR PSD estimation algorithms a simulation study was performed. This consisted of the analysis of a simulated worst case FHR sinusoidal variation of 240 BPM to 60 BPM in a period of 20 seconds. This test signal, which was used earlier in examination of the performance of the Fourier transform technique, consisted of a frequency modulated 2.5Hz carrier sampled at 10Hz. The SNR was set at 20dB and an AR model order of

![Graph showing variation of residual error power with increasing model order](image)

Figure 5.13 Variation of Residual error power with increasing model order
10 chosen on the basis of the results from the previous section. This showed little improvement could be gained in the residual noise level for the fixed sinusoid for model orders greater than 10. This model order was used in all the following AR estimation algorithms in analysing this data.

Using data blocks of 20 samples, the AR PSD was obtained from which the spectral peak corresponding to the FM sinusoid was determined. For the block processing algorithms of the Yule-Walker (YW), Burg, Optimum tapered Burg (OTB), and Energy Weighted Burg (EWB) methods successive data blocks were fully overlapped by 19 samples. This produced an output frequency estimate for every input sample point, providing an easier comparison with the estimates produced by the sequential techniques of the gradient Lattice and LMS algorithms. The adaption times of these sequential algorithms were fixed to give guaranteed convergence in 20 iterations using equation 5.29. The AR PSD was computed using a 512 point FFT to obtained the same frequency domain quantisation as was used in the study of chapter 4. The results from this simulation are presented in the following diagrams, Figures 5.14-5.20. The variation in the frequency of the sinusoid based on the AR estimation is shown against its actual frequency for each estimation algorithm. Example spectra produced for every 5 samples of input data are shown in a corresponding waterfall diagram.

The block algorithms incorporated a check for numerical ill conditioning based on the stability criterion the the PARCOR coefficients. For each ascending model order these must not have magnitude greater than unity. If this test failed the the AR coefficients derived from the previous stable model order were used to derived the AR PSD. Thus, although the model order selected was 10, in some cases a lower model order may have been used depending on the data to be analysed.
Figure 5.14
AR Tracking of FM sinusoid using YW algorithm with examples of AR spectra obtained.
Figure 5.15
AR Tracking of FM sinusoid using Burg algorithm with examples of AR spectra obtained
AR Tracking of FM sinusoid using EW Burg algorithm with examples of AR spectra obtained
Figure 5.17
AR Tracking of FM sinusoid using OT Burg algorithm with examples of AR spectra obtained
Figure 5.18
AR Tracking of FM sinusoid using gradient Lattice algorithm with examples of AR spectra obtained.
Figure 5.19
AR Tracking of FM sinusoid using gradient LMS algorithm with examples of AR spectra obtained.
This is particularly evident for the results obtained using the YW algorithm. These contain a number of spikes where the frequency was estimated to be either 0 or 5Hz. Here the algorithm used a model order as low as 1 in some instances as a result of numerical instability.

5.5.2. Performance Results

From Figures 5.14 — 5.19 the estimation algorithms may be ranked in order of tracking performance as:

1. Optimum Tapered Burg
2. Gradient LMS
3. Gradient Lattice
4. Energy Weighted Burg
5. Burg
6. Yule-Walker

There is ripple evident on the tracking results obtained using the Burg, EWB, and particularly the gradient lattice algorithm. This ripple is due to the initial phase of the sinusoid in the block contributing to the bias in the spectral peak. The good performance of the OTB algorithm is not surprising since the window used in this modification of the Burg algorithm has been specifically designed to minimise the variance of the position of the spectral peak. The performance of the gradient lattice lies between that of the Burg and OTB algorithm. This is also understandable given that the gradient lattice algorithm could be viewed as windowing the data and hence the forward and backward prediction errors with an exponential window. Data
windowing has been shown to reduce the bias effect of initial phase on the peak position.

The tracking performance of the LMS algorithm is good for this case where the signal has been contaminated with white noise. However, in the presence of coloured noise its convergence rate is reduced and this results in poorer tracking performance. This is not the case with the lattice algorithm which has a faster convergence rate than the LMS and is more spectrally robust. The poor performance of the YW algorithm is as a direct result of the ACF estimates produced from the short data record. Using only 20 samples in a block these ACF estimates will be noise contaminated estimates of the true ACF of the signal. These are then used directly by the YW algorithm to form and solve a set of simultaneous equations.

In general the AR spectra have a narrower bandwidth than the those produced by similar processing using the Fourier transform.* Although for all the extra computation little if any improvement in the tracking has been accomplished, with only the LMS and OTB results being comparable with those obtained using the simpler Fourier analysis.

5.5.3. FHR estimation using the AR PSD

The example patient data has been similarly analysed using a 10th order AR model. The position of the global peak in the 1 to 4Hz region of the PSD is taken as the FHR estimate. The AR PSD is calculated using a 512 point FFT giving a ±0.54 BPM quantisation accuracy as for the results presented using the Fourier transform in chapter 4. Figures 5.20–5.25 show the results obtained using the various AR

* Chapter 4 - Figures 4.5,6 Page 94-95
† See Chapter 4 - Figure 4.3
algorithms processing 20 sample (2 second) data blocks. The same general trends in FHR are present as seen in earlier analysis methods, but most contain sharp spikes in the FHR plot indicating incorrect estimates of FHR. One cause of this is the limitations of the simple global peak finding algorithm as a method of estimating FHR from the PSD. Other more sophisticated algorithms incorporating the past history of the FHR estimates and the rate of change of FHR could improve on the general quality of the estimates produced both in the Fourier and AR techniques. In the case of the block processing algorithms a further cause of the errors is due to numerical instability which, as explained earlier, cause the last stable model order estimates of the AR parameters to be used to determine the AR PSD. For low model orders this PSD may be quite a poor representation of the actual signal spectrum.

From these results it would appear that the gradient lattice, and OTB algorithms have performed well. The result obtained using the gradient lattice method is the best and has the fewest incorrect estimates of FHR of all the digital techniques applied to determine FHR from this recording. An improved algorithm to determine the spectral peak corresponding to the fundamental frequency of the foetal heart could enhance all these spectra based measurement techniques.

Figures 5.26 - 5.31 show example spectra from approximately 15 minutes into the recording. It can be seen from these that the width of the spectral peak could provide further information on the SNR of the data and consequently confidence of the FHR estimate from a given data block. A power estimate from the AR PSD can be taken as the height of the spectral peak times its bandwidth. In the calculation of the PSD the absolute magnitude information has been dispensed with by using a numerator of unity rather than the residual error power (equation 5.10). The spectrum thus shows by the bandwidth of the spectral peak how close the poles of the AR model are to the
unit circle. As seen already in the section 5.4 as the noise level increases so the bandwidth of the spectral peak increases. This reflects the movement of the poles away from the unit circle towards the centre to simulate the increase in background noise level. Data blocks which are noise corrupted show up as having a wider bandwidth spectral peak.

None of these AR spectra show any signs of spectral line splitting or spurious peaks, other than those accounted for in terms of interfering noise. This confirms that the model order of 10 is not too high.

The totally blank spectra produced in the analysis using the EWB algorithm are due to ill conditioning in the algorithm caused by the data from these blocks producing PARCOR coefficients greater than unity. In these cases this most likely occurs at model order 1 producing a totally flat spectrum.

Figures 5.32 and 5.33 give examples of the results of processing using a longer block length of 4 seconds using the OTB and LMS algorithms. The results from processing using the other algorithms show the same general characteristic of reduced variance in the position of the spectral peak. This was also noted earlier for the Fourier Transform results and is due to the heart rate variability changes being averaged over this longer time duration.
Figure 5.20
FHR plot derived using the YW AR PSD
— 2 second block processing
Figure 5.21
FHR plot derived using the Burg AR PSD
— 2 second block processing
Figure 5.22
FHR plot derived using the EWB AR PSD
— 2 second block processing
Figure 5.23
FHR plot derived using the OTB AR PSD
— 2 second block processing
Figure 5.24
FHR plot derived using the gradient lattice AR PSD
— 2 second block processing
Figure 5.25
FHR plot derived using the gradient LMS AR PSD
— 2 second block processing
Figure 5.26
Example YW AR spectra from patient data
Example Burg AR spectra from patient data
Figure 5.28
Example EWB AR spectra from patient data
Figure 5.29
Example OTB AR spectra from patient data
Figure 5.30 — Example Gradient Lattice AR spectra from patient data
Figure 5.31 - Example Gradient LMS AR spectra from patient data
Figure 5.32
AR Spectra using OTB algorithm
processing 4 second data blocks
Figure 5.33
AR Spectra using LMS algorithm
processing 4 second data blocks
5.6. Summary

This chapter has introduced the autoregressive spectral analysis technique, which offers increased spectral resolution and smooth spectral shape. The digital filtering models upon which this technique is based and the calculation of the PSD from the model parameters have been outlined. A number of block and sequential algorithms to determine the AR coefficients have been described and concepts of linear prediction and the whitening of the PSD performed by these algorithms has been covered.

The problems associated with incorrect choice of AR model order have been illustrated by means of a study of the AR performance in the spectral estimate of a sinusoidal signal in white noise. General guidelines for the selection of an appropriate model order have been given.

A simulated signal of a FM sinusoid in white noise formed the basis of a comparison of the tracking abilities of the various algorithms. This revealed the problem of spectral peak bias due to starting and ending phase effects in the data. This resulted in most of the algorithms performing worse that the Fourier transform approach used earlier. The performance of the LMS algorithm and OTB algorithm were the exceptions in the algorithms studied. Their performance was found to be as good if not better than that of the Fourier transform method.

FHR plots have been produced from the example foetal data for all the AR techniques discussed. Example AR spectra from analysis of a portion of this data have also been shown, and have verified a reasonable choice of model order. The
higher spectral resolution attainable using these techniques over that of the Fourier transform is evidenced by the narrower bandwidth peaks in these spectra. The peak bandwidth measure is also indicative of the SNR of the signal from which the FHR estimate is to be made. Of the examined AR algorithms the gradient Lattice, gradient LMS and OT burg have been found to be the most suitable for the problem of FHR estimation although their performance has not been found to be greatly superior to that of the Fourier transform methods discussed in chapter 4.
CHAPTER SIX

Conclusions

This thesis has investigated a variety of analogue and digital techniques for determining FHR from ultrasonic foetal heart returns. Improvements in signal processing are required if remote self monitoring of patients is to become clinically useful.

The characteristics of the ultrasound signal and the FHR itself have been used to decide the most appropriate signal processing to be performed. This has concentrated on matched filtering of the fundamental component of the foetal heart beat signal. Two analogue circuits have been designed and constructed to approximately match filter this signal using a switched capacitor bandpass filter. The first circuit used a control loop based on the power output from adjacent SCFs. Problems of tracking, biasing, and initial start-up of this filter were overcome using a PLL design. However the narrow bandwidth inherent in both these designs rendered them prone to "ringing" following impulsive noise produced during relocation of the ultrasonic transducer. Incorrect estimates of FHR were produced due to this effect and there would be no distinguishable change in the signal characteristics.

The time domain algorithm of autocorrelation was examined and its accuracy shown to be dependent on the digital sampling frequency and the heart rate itself. Various ACF algorithms have been surveyed and implemented, and the fast correlation technique shown to be able to produce interpolation of the time domain ACF through the use of data zero-padding. This is also the fastest technique of calculating the ACF, making use of the doubling algorithm to further reduce the computation. The
harmonic and subharmonic peaks produced in the ACF make the identification of the fundamental period difficult. Errors can also result due to non-periodic components being present in the signal. Thinned ACF and a simpler MDF have been examined as a means of reducing the computation involved in this period estimation.

Spectral analysis techniques based on block FFT and sequential DFT algorithms have been implemented using a general purpose microprocessor system. These techniques offer improvement in the accuracy of the FHR estimates over those derived from the period measurement in the time domain. A much lower sample rate can be used to achieve the same quantisation accuracy as the ACF method and this accuracy is independent of the FHR. The reduction in sample rate allows a single processor system more time to process the data. The resulting spectra also give a visual indication of the coherence and SNR of the signal. This may be used as an aid in transducer relocation in addition to the audible output from the foetal monitor itself.

The computations required by these Fourier analysis techniques are less than those required in the determination of the ACF.

A number of AR spectral analysis algorithms have been used offline to determine FHR from the resulting AR PSD. Of these, the most suitable in terms of performance and implementation were found to be the gradient LMS, gradient lattice and optimum tapered burg algorithms. These exhibited the least spectral peak bias due to starting and ending phase of the data. Spectral peak bias was the major limitation of the AR techniques. The performance of the AR algorithms in this specific application were deemed not sufficiently superior to the Fourier transform techniques to warrant the considerable increased in computation required to derive the AR PSD.
The improved resolution of these AR techniques is not fully utilised in this application where accuracy in spectral peak location is of more paramount concern. The AR techniques may be used in the further analysis of periodicities in the FHR plot itself, where spectral resolution becomes more important. They can also be used in a data reconstruction role to provide estimates for FHR which are incorrect in the FHR plot. The data reduction properties of these algorithms also makes them suitable to be used in a linear predictive coding mode where the AR parameters of the signal would be determined on-line in the patient's home, and transmitted back to the hospital for the further computation of the AR spectral or reconstruction of the signal itself.

6.1. Limitations of the work

The algorithms and techniques studied have concentrated on the implementation of a matched filter to the fundamental component of the heart beat signal. In so doing so the power which also exists in the signal harmonics has been rejected. An improved approach to this problem would take this into account and attempt to match filter the complete signal making use of all the available signal power. This may be accomplished using extensions of the frequency domain techniques to determine the major harmonic components present and derive FHR estimates from the positions of all these harmonic rather than simply that of the fundamental.

The algorithms used in conjunction with the time and frequency domain techniques to determine the fundamental period or frequency of the signal have been simple. These used only the information based on the current ACF or spectrum. By incorporating knowledge of the past history of FHR estimates the performance of these signal processing techniques could be improved.
References


42. Dripps, J.H and Manning, G.K, "Microprocessor Based Signal Processing for the determination of fetal heart rate by Non-invasive Ultrasonic Techniques.,” *ACTAS del II Simposium de ingeniera biomedica*, pp. 293-299, Madrid, (5-7 October 1983).


Appendix A

(* denotes paper reprinted here *)

Publications


Comparison of correlation and modulus difference processing algorithms for the determination of foetal heart rate from ultrasonic Doppler signals

G. K. Manning  J. H. Dripps
Electrical Engineering Department, University of Edinburgh, The King's Buildings, Mayfield Road, Edinburgh EH9 3JL, UK.

Abstract—Microprocessor-based correlation processing is now widely used for the estimation of foetal heart rate from ultrasonic Doppler signals. The paper describes a technique having a similar performance to correlation but which will run three times faster on a microprocessor without an onboard hardware multiplier. This speed improvement results from replacing each multiplication operation of the correlation algorithm with a subtraction and a modulus operation. The algorithm is thus referred to as the modulus difference algorithm. Typical waveforms are presented illustrating the responses of the two processors to continuous-wave Doppler foetal heart signals for both wide (50 Hz) and narrow (4 Hz) preprocessor bandwidths. A quantitative comparison is given of the performance of the two methods in estimating the period of a synthetic signal with additive white Gaussian noise. This simulation represents the wideband preprocessor case. A theoretical derivation of the outputs for the narrowband preprocessor case is given, and actual outputs are shown for comparison. A tutorial approach is adopted throughout the paper.

Keywords—Correlation, Foetal heart rate, Modulus difference, Ultrasonic foetal monitoring.


1 Introduction

There are three dynamic foetal activities which are known to reflect foetal well-being. These are fetal heart rate (FLYNN and KELLY, 1971), fetal breathing movements (PATTERSON et al., 1976) and gross body movements (PEARSON and WEAK, 1976). Of these three, only fetal heart rate (FHR) is presently capable of precise definition and reliable measurement in human pregnancy. For this reason FHR is now widely established as the principal method of foetal monitoring.

The most successful and most commonly used signal for FHR estimation is an ultrasonic Doppler signal. This is usually continuous wave (CW), but recently pulsed Doppler (PD) systems have been introduced (ATKINSON and WOODCOCK, 1982). The latest generation of foetal monitors use matched filtering to maximise signal-to-noise ratio (SNR). The matched filter is realised by a correlation algorithm. The audiofrequency Doppler signal is amplitude demodulated and low-pass filtered to a few hertz or tens of hertz before being analogue-to-digital converted. The resulting digital signal is correlation processed in blocks of 1-2 s duration. This provides one new estimate of FHR per block.

The correlation processing now being offered should provide a useful improvement over previous techniques.

Since a correlation processor is the optimum linear processor for stationary signals in additive Gaussian noise, the reader may be forgiven for thinking that there is no room for further improvement. This is not the case, however, since there are many forms of matched or approximate matched filtering and one or more of these forms may prove more computationally efficient or less susceptible to periods of nonstationarity in the signal. The latter point is very important since the demodulated ultrasonic Doppler signal is extremely sensitive to changes in the relative positions of the transducer and foetal heart. This frequently leads to collapse of the autocorrelation signal during these periods of nonstationarity. This is not surprising because the autocorrelation function of a nonstationary signal is undefined. There is considerable scope for work in this area.

Implementation of correlation processing in real time imposes a heavy computational load on a microprocessor. This is due to the large number of multiplication operations involved. One possible solution is to use a hardware multiplier or correlator to perform the 'number crunching' or to use a special signal-processing microprocessor such as the TMS320. This approach is not yet economically attractive and is not considered further. The technique investigated here is the replacement of the multiplication operation with a subtraction and modulus difference operation. The FHR may be estimated by looking for a dip or global minimum in the output waveform as opposed to the...
usual correlation peak or global maximum. The use of the modulus operation makes this strategy nonlinear and therefore not as amenable to analysis as the corresponding correlation-based strategy. The conclusion, based on simulation studies, is that the modulus difference approach runs approximately three times faster than the normal correlation approach with negligible performance degradation.

2 Correlation processing

Let \( x(t) \) and \( y(t) \) be two real signals. Let the linear cross-correlation function (BENDAT and PIERSOL, 1980) derived from \( x(t) \) and \( y(t) \) be denoted by \( \phi_{xy}(\tau) \) where \( \tau \) is a time shift between the two signals. \( \phi_{xy}(\tau) \) is defined as

\[
\phi_{xy}(\tau) = \frac{1}{T} \int_{0}^{T} x(\tau - t)y(t) \, dt
\]

or

\[
\phi_{xy}(\tau) = x(\tau) * y(t)
\]

where \( \ast \) is the convolution symbol and eqn. 2 is a symbol notation method of writing eqn. 1.

For implementation on a digital computer it is convenient to use the discrete time approximation:

\[
\phi_{xy}(m) = \frac{1}{N} \sum_{n=0}^{N-1} x(n - m)y(n)
\]

where \( x(i) = 0 \) for \( i < 0 \) or \( i > N - 1 \) and \( -N < m < N \).

This corresponds physically to shifting one waveform relative to the other. For each value of shift parameter \( m \) a point on the cross-correlation function is obtained by summing the products of all the pairs of corresponding samples. For an \( N \) sample block, the computational load involves \( N \) multipliers and \( N - 1 \) additions and leads to an output \( (2N + 1) \) points long. It is often convenient to implement a circular correlation instead of the linear correlation described above. This is mainly because the output vector contains \( N \) points for the circular case as opposed to \( (2N + 1) \) points for the linear case. In circular correlation the two signals to be cross-correlated may be thought of as being wrapped around a cylindrical surface. One of the two signals is rotated (circular shifting) and again the sum of the products of corresponding sample pairs is computed after each shift. Figs. 1a and 1c show two identical sets of triangles which match each other exactly as they are shown here. These were considered to

![Graphs showing correlation and modulus difference](image-url)
be idealised forms of the 50 Hz bandwidth ultrasonic base-
band signals. Figs. 1b and 1d show the circular correlation
and circular modulus difference functions for these two
signals. Since they initially overlap a peak occurs at
shift = 0 in Fig. 1b. When one signal shifts relative to the
other until the displacement is such that there is no
overlap the correlation function reaches a minimum value
(shift = 45). The secondary peak at shift = 100 is due to
partial overlaps between triangles 3 and 2, and between
triangles 4 and 1 at a point half way through the circular
shifting process. In this case, which contained 200 samples,
the signals match up again after 200 shifts round a cylin-
drical surface 200 samples in circumference.

3 Modulus difference processing
For the same two signals $x(t)$ and $y(t)$ which were used
in Section 2 above, the modulus difference algorithm
(NEILSON, 1974; ROSS et al., 1974; MOORER, 1974) is given
mathematically as:

$$\Delta_x(t) = \frac{1}{T} \int_0^T |x(t)-y(t)| dt$$

The corresponding discrete approximation becomes

$$\Delta_x(m) = \frac{1}{N} \sum_{n=0}^{N-1} |x_n - y_{n+m}|$$

$$m = 0, 1, \ldots, N-1$$

Again we are considering circular waveforms hence the
mod $t$ subscript indicating computation modulo $N$. In
eqn. 5 the number obtained is the sum of the modulus of
the differences between corresponding samples on the cir-
cular surface after $m$ shifts. Fig. 1d shows the modulus
difference function for the triangular waveforms of Figs. 1a
and 1c. As can be seen, a minimum occurs in the modulus
difference function at shift = 0. This minimum shows
where the two waveforms align exactly. The subsidiary
minima either side of shift = 100 are due to partial over-
laps, as for the correlation case. The algorithm structure is
identical to the correlation case but the $N$ mod $t$ multiplications
are replaced with $N^2$ difference operations and $N^2$ 2
modulus operations, assuming the differences between cor-
responding samples are equally likely to be positive or
negative (the modulus operation need only be performed
on negative differences). The desired parameter in both
cases is the shift which best matches the two waveforms.
For the correlation function this point is given by the
position of the absolute maximum and for the modulus
difference function by the position of the absolute

Fig. 2 White Gaussian noise added to the triangular waveform of Fig. 1a

Medical & Biological Engineering & Computing March 1986
minimum. Given the same data set, the position of the maximum of the correlation function and the minimum of the modulus difference function should be the same.

4 Statistical comparison of correlation and modulus difference methods

For the purpose of this study it is convenient to define signal-to-noise ratio as the ratio of the peak signal power to the mean square noise power (peak SNR). The peak signal power is the square of the voltage at the apex of the triangular pulses of Fig. 1a. To determine the relative performance of the two techniques the following statistical tests were carried out for a range of peak SNRs. Independent blocks of white Gaussian noise were added to two replicas of the triangular waveform of Fig. 1a for various peak SNRs as shown in Fig. 2. The value of shift which best matched the two waveforms was found using the correlation and modulus difference algorithms. Since the replicas were shifted by 12 samples with respect to each other, then in the absence of noise this shift should have been 12. In the presence of noise a different value of shift may appear to give a better match. The difference between the two shifts represents an error in the shift estimate. 1000 values of shift error were determined.

The histograms (Fig. 3) show how both the correlation and modulus difference results have been degraded by the presence of noise. However, they still perform well even at a peak SNR of 5 dB. Table I gives the standard deviation and mean for the shift error distributions obtained by

![Histograms showing shift error distributions for 20dB, 15dB, 10dB, and 5dB SNR.](image)

**Table I** Comparison of peak location performance in units of sample interval

<table>
<thead>
<tr>
<th>SNR</th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5.5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean modulus difference</td>
<td>0.005</td>
<td>0.003</td>
<td>0.002</td>
<td>0.000</td>
<td>0.010</td>
</tr>
<tr>
<td>Mean correlation</td>
<td>0.018</td>
<td>0.024</td>
<td>0.028</td>
<td>0.14</td>
<td>0.52</td>
</tr>
<tr>
<td>Standard deviation modulus difference</td>
<td>0.876</td>
<td>1.516</td>
<td>2.605</td>
<td>3.43</td>
<td>15.55</td>
</tr>
<tr>
<td>Standard deviation correlation</td>
<td>0.537</td>
<td>1.413</td>
<td>2.344</td>
<td>3.221</td>
<td>11.29</td>
</tr>
<tr>
<td>Sample size</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>

*Medical & Biological Engineering & Computing* March 1986
modulus difference and correlation as a function of peak SNR. From these figures it can be seen that there is a negligible difference between the two methods above 10 dB. Below 10 dB correlation gives slightly better results than modulus difference.

5 Application to foetal heart rate determination
Continuous wave Doppler ultrasonic backscatter from a foetal heart was used to compare correlator and modulus difference processor outputs. The audiofrequency Doppler foetal heart signals were recorded on a standard cassette recorder from the tape output of a small hand-portable

Fig. 4 (a) Upper trace: audiofrequency Doppler returns from two successive foetal heartbeats; lower trace: signal of upper trace envelope detected and low-pass filtered; (b) same recording as (a) but a few seconds later, showing how the signal has changed shape

Medical & Biological Engineering & Computing March 1986
foetal pulse detector. The upper trace of Fig. 4a shows the audiofrequency Doppler-shifted backscatter from two successive foetal heart beats. These signals typically consist of up to four bursts of sinusoid at frequencies up to 600 Hz. These bursts of sinusoid are caused by heart valves opening and closing with components of motion in a radial direction relative to the transmit receive transducers. The signal changes slowly with time as the foetal heart turns relative to the insonating waveform. These changes are mainly in the amplitudes of the peaks and the number of distinct peaks associated with a single cardiac cycle. The shape of the waveform and the underlying physical mechanism is discussed by Colsin (1980). Provided a quasistationary assumption can be made on a beat-to-beat basis then the detailed shape of the waveform is not important for the purposes of extracting FHR information. Fig. 4b shows how the signal of Fig. 4a changed in the space of a few heartbeats.

The recordings were replayed through a full-wave rectifier followed by a low-pass filter of 50 Hz bandwidth. The resulting signal is shown in the lower trace of Figs. 4a and 4b. This signal was input to a PDP-11 60 computer via an 8-bit analogue-to-digital convertor running at 300 samples s

Fig. 5 shows the result of applying the two processing methods to two successive foetal heartbeats. The number of samples taken in each individual heartbeat was standardized to 200 by padding from the end of the signal with zeros. For the example shown in Fig. 5a this involved padding from sample 129 to 200 inclusive. Figs. 5b and 5d show the cross-correlation and cross-modulus difference functions of the signals of Figs. 5a and 5c. In this particular example the maximum of the correlation and the minimum of the modulus difference function both lie at a shift of 12. This means that the best possible match of these waveforms occurs if one of the waveforms is shifted by 12 sample intervals in the appropriate direction. The resulting overlay is shown in Fig. 6a.

Knowing the value of shift for the optimum match it is possible to calculate the foetal heart rate. For example, consider the case where the two signals align perfectly for shift = 0. This means that the period of the heart beat is exactly the length of the first of the two signals. If this first signal consists of samples S

It may be observed from Fig. 5 that the subsidiary dips (or local minima) of the modulus difference function are less prominent than the corresponding local maxima of the cross-correlation function. If this proves to be a characteristic difference between the two techniques it could enhance the reliability of global minimum detection in the case of the modulus difference method. Furthermore, the modulus difference method is less sensitive to variations in the power level of the signal than the cross-correlation technique (see Appendixes).

6 Real-time realisation on a general-purpose microprocessor

The previous section presented results obtained offline using a PDP-11 60 computer. Both correlation and modulus difference processing have also been implemented on a Motorola 68000 single-board computer in "Prol kit" evaluation boards. Both algorithms were written in "C" language and cross-compiled to 68000 machine code. This code was then downloaded into the 68000 board and run in real time. The 68000 board was equipped with an 8-bit analogue-to-digital convertor (ADC) and 8-bit digital-to-analogue convertor (DAC) to provide analogue input and output.

Inspection of the 68000 code at assembly level revealed it to be surprisingly efficient provided that the register declaration facility was used. The code would not run significantly faster even if critical parts were written directly in assembler. As a result such actions were deemed unnecessary, particularly since only the relative speeds of the two algorithms was being determined. 16-bit integer arithmetic was used as this is generally found to be necessary to avoid unacceptable arithmetic quantisation effects, although 8-bit inputs and outputs were found to be acceptable.

The sample rate was reduced to 100 samples s

The major aim of this project is to provide signal-processing facilities to permit reliable foetal heart rate data collection on a routine basis from early in pregnancy. Early pregnancy recordings are often characterised by a poor signal-to-noise ratio. As a result the technique of insonating individual beats and correlation or modulus difference processing on a beat-to-beat basis was abandoned in favour of processing over longer blocks of data. These longer blocks contained some seven or eight heartbeats and lasted 3.75 s ± 1.6 of a minute. This block length is longer than that used in any of the commercially available machines but is hopefully beneficial at poor SNR. There is also a decrease in ability to reflect true beat-to-beat variability, but this was not important for this particular data-collection exercise. From the signal-processing viewpoint the important advantage of this approach is that it obviates the necessity of finding the gaps between suc-
cessive heartbeats. Finding these gaps reliably is often impossible, particularly at low SNR.

A further advantage is that the analogue preprocessing bandwidth can be reduced to the minimum commensurate with passing the fundamental Fourier component of the envelope detected (baseband) Doppler signal of Figs. 4a and b. This minimum width passband is from 1 Hz to 4 Hz and is determined by the minimum and maximum foetal heart rates which are to be processed. 1 Hz corresponds to 60 beats min$^{-1}$ and 4 Hz corresponds to 240 beats min$^{-1}$. The foetal heart rate would normally be expected to lie in the 120–150 beats min$^{-1}$ region, giving a fundamental Fourier component in the 2–2.5 Hz region.

A typical signal obtained with this preprocessing bandwidth is shown in Fig. 7. This is essentially a sine wave with some second harmonic and noise. The autocorrelation and modulus difference functions for this signal are shown in Fig. 8. The expressions for the autocorrelation and modulus difference functions of a sine wave are derived in the Appendix. The autocorrelation and modulus difference functions for the signal in Fig. 7. These two algorithms both implemented linear rather than circular processing to avoid inevitable discontinuities between the two ends of the data block.

7 Conclusion

It has been shown that the modulus difference algorithm proposed in this paper yields comparable results to those obtained by the standard correlation algorithm. The modulus difference algorithm is however more attractive for real-time signal processing since it involves addition/subtraction operations rather than multiplication. On a Motorola 68000 based system the modulus difference method was three times as fast as the corresponding correlation method. For the case of a foetal ultrasonic Doppler signal with minimum bandwidth preprocessing, the outputs obtained with both techniques have been shown to agree closely with the theoretical outputs as derived in Appendix I.

8 Further work

Frequency domain techniques including analogue tracking (switched capacitor filters), zero-padded FFTs, baseband analyser and a number of autoregressive spectral analysis algorithms are also being investigated. As is the frequency-domain computation of the autocorrelation function commonly referred to as ‘fast convolution’. When all approaches are available for real-time use comparative tests will be performed using more extensive foetal ultrasonic Doppler recordings.

References


Appendix 1
Derivation of the correlation function for a sinusoid

Let
\[ r(t) = \cos(wt + \theta) \tag{11} \]
where \( w \) is angular frequency and \( \theta \) is the phase of the sinusoid.
The correlation function is by definition
\[ \phi_{r,r}(t) = E[r(t)r(t + t)] \tag{12} \]
which in this case becomes
\[ \phi_{r,r}(t) = E[\cos(wt + \theta)\cos(w(t + t) + \theta)] \tag{13} \]
where \( E[ ] \) denotes expectation. Using the trigonometric identity
\[ 2\cos x\cos y = \cos(x-y) + \cos(x+y) \tag{14} \]
\[ 2\phi_{r,r}(t) = E[\cos(2\omega + 2t) + \cos(2\omega)] \tag{15} \]
Hence
\[ 2\phi_{r,r}(t) = \frac{1}{T} \int_{-T/2}^{T/2} \cos(2\omega + 2t) + \cos(2\omega) \, dt \tag{16} \]
Therefore
\[ \phi_{r,r}(t) = \frac{1}{T} \left[ \frac{1}{2} \sin(2\omega t + 2\theta) + \cos(2\omega) \right] \tag{17} \]
This function is shown in Fig. 9a.

Appendix 2
Derivation of the modulus difference function for a sinusoid

Let
\[ x(t) = \cos(\omega t) \tag{18} \]
In this derivation the phase of the signal may be assumed to be zero without any loss in generality. This assumption simplifies the mathematics.
The modulus difference function is by definition
\[ \Delta_{r,r}(t) = E[|r(t)| - |r(t + t)|] \tag{19} \]
\[ \Delta_{r,r}(t) = E[|\cos(\omega t - \omega t)| - \cos(\omega t)|] \tag{20} \]
Hence
\[ \Delta_{r,r}(t) = \frac{1}{T} \int_{-T/2}^{T/2} \cos(\omega t - \omega t) - \cos(\omega t) \, dt \tag{21} \]
From Fig. 9b by symmetry
\[ \Delta_{r,r}(t) = \frac{1}{T} \int_{0}^{T/2} \cos(\omega t - \omega t) - \cos(\omega t) \, dt \tag{22} \]
For $a < r < h$
\[
\cos \phi = \frac{r}{h}
\]

for $h < r < T - a$
\[
\cos \phi > \cos (T - a)
\]

$u$ and $h$ can be determined from
\[
\cos \phi = \cos (T - a)
\]

for $t = a, t = b$

Using
\[
\cos \phi = \frac{\cos \phi \cos \beta - \sin \phi \sin \beta}{\cos \phi \cos \alpha - \sin \phi \sin \alpha}
\]

\[
\sin \phi \sin \alpha = \cos \phi \cos \beta - \sin \phi \sin \beta
\]

\[
\tan \phi = \frac{\cos \phi - 1}{\sin \phi}
\]

Therefore
\[
u = \arctan \left( \frac{\cos \phi - 1}{\sin \phi} \right)
\]

and
\[
u = \phi + \nu
\]

\[
\Delta_{1,2} = \frac{1}{2} \left[ \sin \varphi \cos \alpha - \sin \varphi \cos \beta \right]
\]

\[
\Delta_{1,2} = \frac{1}{2} \left[ \sin \varphi \cos \alpha - \sin \varphi \cos \beta \right]
\]

\[
\Delta_{1,2} = \frac{1}{2} \left[ \sin \varphi \cos \alpha - \sin \varphi \cos \beta \right]
\]

\[
\Delta_{1,2} = \frac{1}{2} \left[ \sin \varphi \cos \alpha - \sin \varphi \cos \beta \right]
\]

From eqn. 26 and using Pythagoras's theorem
\[
\cos \varphi = \frac{\sin \varphi}{\sqrt{2}}
\]

\[
\sin \varphi = \frac{\cos \varphi}{\sqrt{2}}
\]

Substituting eqns. 34 and 35 into eqn. 33
\[
\Delta_{1,2} = \frac{1}{2} \left[ \sin \varphi \cos \alpha - \sin \varphi \cos \beta \right]
\]

This function is shown in Fig. 9a.

Authors' biographies

Keith Manning received a B.Sc. degree in Electrical & Electronic Engineering from Queen's University of Belfast in June 1982. He is currently studying for his Ph.D. in signal processing for ultrasonic fetal monitoring at Edinburgh University. This research concerns the application of time- and frequency-domain algorithms to the determination of short-term fetal heart rate.

Jimm Dripps received a B.Sc. degree in Electrical Engineering from Queen’s University of Belfast in 1970 and a Ph.D. degree from Strathclyde University, Glasgow, in 1977, studying modulation methods for digital data transmission on HF ionospheric links. From 1970 to 1972 he worked on circuit design with Short Bros. (Aircraft and Missiles) Belfast and from 1977 to 1980 on FFT processor design for radar at Ferranti, Edinburgh. He is currently a lecturer in the Department of Electrical Engineering at Edinburgh University. His research interests are in the application of signal processing techniques to communication and medical electronics systems.
PROCESSING AND DISPLAY OF FETAL HEART RATE PLOTS FROM ULTRASONIC DOPPLER SIGNALS

G. K. Manning* and J. H. Dripps*

1. Introduction
Fetal heart rate monitoring using non-invasive Doppler Ultrasound is an established technique in obstetric practice to aid decisions on fetal development and well-being. Before the introduction of the latest generation of fetal monitors, commercial monitors used demodulation and fixed analogue filtering of the ultrasonic returns from the fetal heart valves, followed by thresholding and interval measurement to obtain estimates of fetal heart rate (FHR). More recently time domain digital signal processing algorithms such as correlation [1] and Magnitude Differencing [2] have been used to process the demodulated returns in an approximate bandwidth of 150 Hz over typically 1 to 2 second intervals. By estimating the position of the maxima/minima of these functions a better estimate of the FHR can be obtained particularly at poor signal to noise ratios (SNR's).

This paper presents some frequency domain algorithms which may be used as an alternative to the time domain algorithms currently in use. When combined with a 3-D plotting routine these frequency domain techniques have the advantage of producing a display of FHR that conveys signal quality information visually. This should give the clinician more confidence in the interpretation of the displayed heart rate data than is currently available. Results obtained on actual data from a patient are presented. Real-time implementation of the techniques on a 68000 single board computer (SBC) and on a specially developed TMS32010 VMEbus signal processing board discussed.

2. Frequency Domain Fetal Heart Rate Determination
The time domain approach of correlation processing as a means of determining FHR requires that the sampling rate be as high as possible in order to finely quantise the time domain and hence accurately determine the period of the fetal heart beats. For example in order to obtain a 1 BPM accuracy in FHR determination a sampling rate of 1 kHz is required assuming that the 1 BPM specification is to be met at a FHR of 240 BPM (4 Hz). By processing the signal in the frequency domain the sampling rate can be reduced by two orders of magnitude to 10 Hz and the accuracy still maintained. Reduction to 10 Hz sampling rate is only possible if attention is concentrated on the fundamental Fourier component of the demodulated ultrasonic returns from the fetal heart. This may be achieved by low-pass filtering of the demodulated signal to reject the harmonic terms. Although all the results presented have been obtained adopting this approach it should be noted that significant power does exist at times in mainly the odd harmonics up to the seventh. Better processing techniques would attempt to use all the signal power in these harmonics on which to base the estimate of FHR. Further work is being carried out on methods of analysis which will use the harmonic structure of the signal to obtain better estimates of FHR however in justification of the present processing the fundamental component is always guaranteed to be present in the returns from the fetal heart and is often among the strongest power components.

2.1. Fourier Transform Processors
The conventional method of transforming N data points from the time to frequency domain is the Discrete Fourier Transform (DFT) defined as

\[ X(n) = \sum_{k=0}^{N-1} x(k)w_n^k \quad n=0,1...N-1 \]  

\[ w_N = \exp\left(\frac{2\pi i}{N}\right) \]

The separation between successive frequency bins \( f_q \) is given as

\[ f_q = \frac{f_s}{N} \]

where

\( f_s \) is the sampling frequency.

\( N \) is the Transform size.

*University of Edinburgh.
Dept. of Electrical Engineering.
Signal Processing Group.
It is clear that in order to more finely quantize the frequency domain it is necessary to either decrease the sampling rate while still meeting the Nyquist criterion or increase the Transform size N. The fetal signal in this case was sampled at 10 Hz and processed in blocks of 10 samples overlapped by 50% with the preceding block. A 256 point Radix-2 Fast Fourier Transform (FFT) was used to transform the data which was first padded with zeros to the 256 points. Zero padding is a convenient means of achieving interpolation in the frequency domain and hence finely quantizing the frequency domain. It does not increase resolution which is quite distinct from quantization. Use was also made of the fact that the data is entirely real and by reordering samples according to the doubling algorithm [3] the 256 point transform was actually used to calculate the power spectrum equivalent to a 512 point transform. Thus the spacing between frequency bins was
\[ 0.028 \text{ Hz} = 1.2 \text{ BPM} \]
All processing was performed using 16 bit integer arithmetic on a Motorola 68000 based SBC using interrupt driven data collection.

1. Sequential Processing

The FFT is normally the most computationally efficient method of computing the DFT requiring \( O(N \log N) \) complex multiplications to transform a block of N time domain samples to their corresponding N frequency domain data samples. However the efficiency of the FFT is reduced compared to the direct evaluation of the DFT when

1. The data is highly zero padded.
2. Only a limited number of frequency bins are of interest.
3. Successive data blocks are highly overlapped.

In PPH estimation using the extreme case of sequential processing where the power spectrum is updated after each new input sample for example

1. Only 38 data samples are used being padded by 218 zeros.
2. Only the range 1 Hz to 4 Hz is of interest.
3. Blocks are fully overlapped to provide a smooth display on a VDU or digital storage oscilloscope.

Thus these points lead to the use of a recursive formulation for the DFT based on equation 1.

1.1. Halberstein's Method

Halberstein [4] developed the following recursive equation for evaluation of the DFT of N time domain data points to N frequency domain points taken from the infinite time series \( x(t) \). Let \( x(k) \) be the data set consisting of \( x(k), x(k+1), \ldots, x(N(k-1)) \).

\[
T_{k+1}(n) = \frac{1}{N} \left[ T_k(n) + N x(N+k) - x(k) \right]
\]
and introducing a new variable

\[
T_k(n) = w_{N}^{nk} x_k(n)
\]
to give

\[
T_{k+1}(n) = T_k(n) + w_{N}^{nk} \left[ x(N+k) - x(k) \right]
\]
Equation 3 can be modified to take advantage of the zero padding case for N data samples zero padded to \( N \) samples

\[
T_{k+1}(n) = T_k(n) + w_{N}^{nk} x(N+k) - w_{N}^{nk} x(k)
\]

1.2. Exponential Time Window

Rather than using the sliding rectangular window on the time domain data we have introduced an exponential window given by

\[
W(z) = \frac{a}{1 - a z^{-1}}
\]

\[
W_{k}(z) = a \beta^{k-n} \quad n=k,k+1,\ldots
\]

\[
a = 1 - \beta \quad 0 < \beta < 1
\]

This results in the older samples being gradually weighted out rather than, as in Ting's case, being subtracted out once and for all as the window moves forward in time. Mathematically a sliding block estimator has been replaced by a first order recursive estimator [6]. This modification changes equation 3 to
and has the structure shown in Fig. 1(a). This is equivalent to a baseband analyzer in which a
local oscillator in this case \( W \) demodulates
the incoming signal to baseband following which a
low-pass filter rejects out-band product
terms. In this algorithm the incoming signal
sample regarded as a positive frequency phasor
is rotated by different amounts by each of a
set of negative frequency phasors. One of
these will de-rotate (or render stationary)
the input phasor. This de-rotated phasor will
add coherently to previous de-rotated inputs
forming a global peak in the spectrum at the
corresponding frequency.

An alternative algorithm may be formulated by
similarly multiplying equation 1 by the phase
rotation \( W \) and applying the recursive exponen-
tial window. This leads to equation 5

\[
T(n+1) = \beta T(n) + \alpha x(n)
\]

where this time the particular phasor is
rotated to align with the next incoming sample
in the bin containing the global peak. This
algorithm Fig. 1(b) is very similar to the
Goertzel first order algorithm [7] which can be
used to compute a DFT. Both equations 4
and 5 yield the same results however equation 5
has simpler rotational factors and is easier to
implement.

These recursive algorithms essentially compute
the full DFT however they have like the DFT
an advantage over the FFT that the Fourier coeffi-
cients are independently calculated and thus
the individual frequency channels can be
divided to take account of the frequency range
of interest and/or to implement a parallel pro-
cessing structure. Also the effective block
length \( L \) of the data can be easily changed by
altering the factor \( \beta \) which for this simple
first order case is given by [8].

\[
L = \frac{1}{1 - \beta}
\]

Examples of the power spectra obtained using
this technique are shown in Fig. 2, and
Fig. 6(a) which will be discussed later. These
algorithms can easily run in real time using 16
bit integer arithmetic on the 68000 SBC computing
the 154 bins of a 512 point DFT correspond-
ting to the 1 Hz to 4 Hz frequency range of
interest.


The Fourier Transform assumes a model for the
observed signal \( x(k) \) as being composed of a sum
of harmonically related sinusoids of appropri-
ate amplitudes and phases. An alternative
autoregressive (AR) model can be used which
gives increased spectral resolution over that
of the Fourier analysis for any given observa-
tion period. This model assumes that any sam-
ple of the signal \( x(k) \) is composed of a
weighted sum of the \( P \) previous samples \( x(k-
P) \ldots x(k-1) \) and a noise term \( w(k) \) where \( P \)

\[
x(k) = \sum_{n=1}^{P} a_n x(k-n) + w(k)
\]

Using this model the observed signal is assumed
to have been produced by the coloration of a
white noise input by an infinite impulse
response (IIR) filter as shown in Fig. 3. By
determining the AR coefficients \( a_1, a_2, \ldots, a_P \),
the spectrum of the signal can be calculated.

301
where $\sigma^2$ is the variance of the white noise $w(k)$

Various algorithms [9] can be used to determine the AR coefficients from the input signal and for selection of the model order $P$. The technique which we have elected to use with the FHR data is that of Adaptive Line Enhancement [10] which uses a finite impulse response (FIR) adaptive filter with the Widrow Least Mean Squares (LMS) algorithm, for system identification. The FIR filter is the inverse filter to the IIR model and the LMS algorithm adjusts the coefficients in order that the error $e(k)$ is white thus it acts to whiten the input signal. (Appendix A). The structure used is shown in Fig. 4. and equations 7 summarise the method [12].

$$P(x(k) = \sum_{n=1}^{P} a_n x(k-n)$$

(7a)

$$e(k) = x(k) - \hat{x}(k)$$

(7b)

$$u(k) = \frac{\alpha}{\hat{r}_x(0)} \hat{r}_x(0) \hat{r}_x(k)$$

(7c)

$$a_n(k+1) = a_n(k) + u(k)e(k)x(k-n)$$

(7d)

$n = 1, 2, 3, \ldots, P$

where $\hat{x}(k)$ is the predicted input sample.

$e(k)$ is the error between the actual input sample and the predicted sample.

$P$ is the model order.

$\hat{r}_x(0)$ is an estimate of the signal power.

The factor $\alpha$ governs the convergence and hence the tracking properties of the LMS algorithm and in this case has been chosen according to equation 8 to give guaranteed convergence within 40 samples ($\hat{r}_x(0)$).

$$\alpha = P \left(1 - \exp\left(-\frac{1}{\hat{r}_x(0)}\right)\right)$$

(8)

The spectrum is computed every 20 samples using equation 6 with $\sigma^2$ set equal to 1. Fig. 6(b) shows a sequence of AR spectra computed offline from data from a patient at 40 weeks gestation using the plotting technique discussed later.

It should be noted that the coefficients $a_1, \ldots, a_P$ are fixed at zero and not adapted in the structure Fig. 4. This is necessary to provide delay elements which are used to de-correlate any noise in the signal. The length of this delay is a compromise between a long delay to de-correlate a wide range of coloured noise and a short delay which enables fast recovery from periods of non-stationarity in the signal.

The spectrum produced by the AR method gives a direct indication of the frequency components present in the signal however unlike the Fourier power spectrum the power of the com-
components is given by the area beneath the peak rather than the height of the peak itself.

An alternative algorithm that of the Gradient Lattice [11] has also been used but was seen to produce noisier estimates of the position of the spectral peak although this algorithm has been shown to have better tracking performance than the LMS algorithm where it is desired to track multiple signals of widely different power levels. Fig. 5 demonstrates the LMS and Lattice algorithms performance in attempting to track a single ramped frequency modulated sinusoid.

Fig. 6(a) and 6(b) show the three dimensional plots of power spectra produced by the recursive DFT and AR analysis of the same section of data from a patient at 40 weeks gestation. Each spectrum is the result of the analysis of 40 data samples with a plot being produced every 20 samples (2 secs). The X-axis represents FHR from 1.5 Hz (90 BPM) to 3.5 Hz (210 BPM). The Y-axis represents time in increments of 2 seconds and the Z-axis is that of normalised power.

After calculating the power spectrum by one of the techniques discussed, the global peak is determined and its value used to normalise the spectrum to the range 0 to 1. In the case of the Fourier transform method this normalisation does represent a loss of information since the height of the peak gives a measure of the signal power and could be used as a confidence measure. For the AR spectrum it is the bandwidth of the peak that can be visually interpreted as a measure of confidence and this is preserved despite the normalisation. The normalised spectrum is scaled and plotted to an appropriate size on a low cost ink pen plotter (Tandy CGP-115 or Micro-Peripherals MCP40). The axis is then moved down by a fixed amount to prevent overlap of successive plots and the origin re-initialised. Because of the mechanism of paper feed on this type of plotter there is no significant limitation on the length of the plot.

Fig. 7 shows the conventional 2-D plot of FHR obtained from the recording of the peak position from the power spectrum for this section.
Fig. 6(a) Normalised spectra of baseband ultrasonic heart valve returns computed using recursive DFT over effective 4 sec block lengths, spectra plotted every 2 secs.

Fig. 6(b) Normalised spectra computed by Autoregressive technique for same data as in Fig. 6(a).

Fig. 7 Conventional 2-D plot of fetal heart rate derived from peak positions of spectra in Fig. 6(a).
of data. The distinct advantage of the 3-D plot over the conventional 2-D plot of FHR is the extra information retained which can be used to further aid medical decisions based on FHR plots. By observing the spectrum of the fetal data during periods where decisions of peak position are being made on poor SWF data, doubtful decisions can be identified and the resulting SWF information can be disregarded. The spectra of the plots during such periods can be seen to be more noisy while those obtained from the AR technique have a wider bandwidth. Both these features can be seen in Fig. 6.

These plots have been produced off-line in the case of the AR spectra as insufficient processing time was available using the 68000 SBC to determine the AR coefficients and compute the spectrum, however the main limitation that applies to both techniques is that of plotter speed. With advances in plotter technology using thermal plotters however it should be feasible to produce the 3-D real-time display.

6. VNDBUS TMS32010 Processor Board

The VNDBus has been developed to meet the need of a multi-processor environment. Recently signal processing boards [13],[14] have begun to appear to enable fast digital signal processing (DSP) of block data in a VME system. These use the Texas Instruments digital signal processing microprocessor TMS32010 which has an instruction cycle of 200 ns for most of its instruction set. In order to facilitate implementation of the AR spectral analysis in real-time a board has been developed by our signal processing group to permit fast sequential processing using the TMS32010 as part of a VME system with a 68000 master processor board.

In this arrangement the DSP board has its own 8-bit ADC DAC and direct access to a 16-bit input and a 16-bit output port. Bank switching has been implemented to overcome some of the limitations of the 4K word program memory space of the TMS32010 and external static data memory of 8K words has also been included to extend the TMS320’s on-board data memory. For stand-alone use the board can also work from EPROM ROM program memory.

The philosophy behind this design is to use the master board to download the TMS120 machine code to a particular DSP board and then allow the much faster CPU to work independently with access to its own input/output data streams. The inclusion of the fully handshaked parallel ports enables up to 8 such slave DSP boards at different address locations in the VME memory map to be cascaded serially as in one-dimensional array processor with the results from one board being passed to the next for further processing.

Alternatively one board can be dedicated to one input channel allowing up to 8 parallel channels to be processed. A TMS320 macro-cross assembler on a VAX 11/750 and on a Torch Unicorn system both running the UNIX operating system provides some software support for the system together with a C compiler for the 68000.

Using this system it is hoped that the results shown for the AR analysis may be obtained in real-time. The system has been designed to meet the requirements of higher bandwidth signal processing across a wide range of applications.

7. Conclusion

It has been shown that the frequency domain approaches of Fourier and Autoregressive analysis can be used to determine fetal heart rate and combined with the 3-D plotting procedures these techniques provide a better confidence indicator than currently available monitoring methods.

Work is continuing on the application of the AR technique for real-time measurement and on improving the signal processing to include the harmonic components present in the signal.

In above diagram Fig. 1(a) shows the AR generation model for a single sinusoid of frequency \( f \) sampled at a sample period \( T \). This model is an IIR filter with a transfer function given by:

\[
X(z) = H_{\text{model}}(z) = \frac{1}{z^2 - 2 \cos \omega T z^{-1} + 1}
\]

driven by the white noise input \( w(k) \).

This function places a complex conjugate pair of poles on the unit circle in the Z-plane as shown and thus acts as an oscillator. \( x(k) \) is the observed signal which consists of the sinusoid and the white noise term \( w(k) \).

The task of the AR estimator is to determine the coefficients of the inverse model which will whiten the observed signal \( x(k) \). This inverse model Fig. 1(b) is a FIR filter in which the coefficients are iteratively adjusted by some adaptive algorithm (eq. 1.115). After convergence, this filter places zeros in the Z-plane over the poles introduced by the AR generation model. In this case the inverse filter will have the transfer function:

\[
X(z) = H_{\text{inverse}} = z^{-2} - 2 \cos \omega T z^{-1} + 1
\]

The combined transfer function is thus:

\[
E(z) = W(z) = 1
\]

ie.

\[
e(k) = w(k) \quad \text{white noise}
\]
Appendix B

Signal Processing Software

This appendix contains the following software written in 'C' on a VAX11/750 running UNIX.

1. Median Filtering

2. Radix-2 Fast Fourier Transform


4. Autoregressive Analysis using Gradient LMS and Lattice Algorithms

Manual Pages describing the operation of the software are given in the cases of the FFT and AR programs. The remaining software includes a brief help facility.
NAME
fft - "Radix-2 Fast Fourier Transform" Spectral Analysis

SYNTAX
fft -n SIZE [ options ] ....

OPTIONS

-n number
Specifies the size of Transform to be performed. SIZE should be a power of 2 if not it defaults to the highest power of 2 above the number specified. Zero-padding is performed automatically if the transform size is greater than the data record length.

-s number
Specifies the sample rate of the data used to scale the frequency scale of the output spectrum

-r
Specifies input data to be REAL if not specified input data is taken as pairs of complex numbers real/imaginary

-h
Specifies that only the spectrum from D.C. to Half the sample rate should be output - useful when real data is used.

-w
Suppress windowing by the Hamming window function on the data

-c
Output Complex Fourier coefficients - defaults to dB log power spectrum otherwise

-l
Output linear power spectrum - defaults to dB log power spectrum otherwise

-N
Normalises the output linear/log power spectrum so that maximum value is 1.0 if linear or 0 dB if log.

-i
Calculate the Inverse Fourier transform - defaults to the forward transform.

-a
Calculate the Autoregressive power spectrum. Effectively sets no windowing and real input data options expecting data input to be real valued A.R. coefficients

-e number
Uses the following number as the prediction error power when calculating the autoregressive spectrum.

-f string
Specifies the input data file which contains data to be transformed.

-o string
Specifies the output file to contain the results of the transform.

DESCRIPTION
The command calculates the radix-2 D.I.F. Fast Fourier Transform of data from the input data file specified or from data entered from the keyboard if no file is specified. Results are placed in the output file if specified on appear on the terminal by default. Choice of transform size, Hamming Windowing, real or complex input data may all be selected by the user together with the type of output linear/log spectrum, or complex Fourier coefficients.
EXAMPLE

fft -n 512 -i infile -o outfile -l -r -h -s 120

This command will take real input data from file "infile" and calculate the linear Fourier power spectrum of size 512 points zero-padding if necessary for the region 0-60 Hz outputing this to the file "outfile".

fft -n 8 -c -r -w

Calculates the complex Fourier coefficients for 8 (or less) data points entered from the terminal, no hamming windowing and outputing the result to the terminal – Try this for yourself!

SPECIAL CONSIDERATIONS

When calculating the power spectrum in dB for certain data input it is possible to produce zero values in the frequency bins when this occurs 10*log(0) is taken as -200 dB rather than letting the program crash by attempting to take the log of zero.

FILES

On line documentation may also be obtained by typing the command by itself with no parameters which will also occur if errors are made in entering any of the parameters.

AUTHOR

Keith Manning - EE department kman@edee

SEE ALSO

alc(1)
NAME
ale - Adaptive Line Enhancement Spectral Analysis

SYNTAX
ale -a[yoeb] [ options ] ...

OPTIONS
-f input-file
  Specifies the file containing data to be analysed - default stdin
-o output-file
  Specifies the file in which the results will be placed - default stdout
-n number
  Selects the model order to be used in the analysis
-a[yoeb]
  Selects the algorithm to be used in determining the Autoregressive coefficients and hence the spectrum. Choose from
  y  -  Yule-Walker Algorithm
  b  -  Burg Algorithm
  o  -  Optimum Tapered Burg Algorithm
  e  -  Energy Weighted Burg Algorithm
-t number
  This selects the size of the final Fourier Transform to be performed to calculate the Spectrum - default 512.
-l  Selects a log spectrum output - defaults to linear
-s number
  Selects the sample rate to be used to scale the frequency axis.
-p  Selects that the autoregressive coefficients and error will be output rather than the default A.R. Spectrum.

DESCRIPTION
The program enable the user to determine the A.R. coefficients/Spectrum for the input data using one of the specified algorithms and the given model order.

EXAMPLES
   ale -f data -o spectrum -ab -n 10 -t 256 -s 10
This will calculate the A.R. Spectrum for data in the file "data", outputing the linear spectrum of size 256 and scaled to a sample rate of 10 Hz, to the file "spectrum", using the Burg Algorithm to calculate the coefficients on which the spectrum is based.

SPECIAL CONSIDERATIONS
If the algorithms become unstable due to too high a model order being selected for the given data then a spectrum or coefficients will be output for the last stable model order.

AUTHOR
Keith Manning EE department - mail kman@edee

SEE ALSO
   fft (1)
Median Filtering

/*
* Median Filtering Program - File median.c
* Outputs samples which are the median of N input Samples.
* Initial Output Samples = Input samples until block N is formed
* Final Output Samples = Input samples
*/

#include <stdio.h>

main(argc, argv)
int argc;
char **argv;
{
    char *help = "\n\tNumber of points in median filter\n\tFile Input file\n\tOutput file\n\nMedian Filtering Program.\n"
;
    int N,i;
    char *prog;
    float *x,*y;
    float *emalloc(),median();
    FILE *fpin,*fpout,*efopen();

    fpin = stdin;
    fpout = stdout;
    prog = argv[0];
    if(argc < 3 ){
        fprintf(stderr,help,prog);
        exit(1);
    }
    /* Process the command line options */
    while(--argc && (*++argv)[0] == '-'){
        switch(*((argv[0] + 1))){
            case 'n':
                N = atoi(*++argv);
                argc--;
                break;
            case 'f':
                fpin = efopen(*++argv,"r");
/* fetch the required memory for the arrays */

x = emalloc(N);
y = emalloc(N);

/* Input the first data block */

for (i = 0; i < N; i++)
    fscanf(fpin, "%f", x+i);

/* Output the initial samples */

for (i = 0; i < N-1; i++)
    fprintf(fpout, "%f\n", x[i]);

do{
    /* For each new sample until end */
    /* determine the median and shift the data block */

    fprintf(fpout, "%f\n", median(x, y, N));
    for (i = 1; i < N; i++)
        x[i] = x[i-1];

    /* Fetch a new sample */

    while (fscanf(fpin, "%f", x) != EOF);

    /* Write out the remaining samples of the block */

    for (i = 0; i < N-1; i++)
        fprintf(fpout, "%f\n", x[i+1]);
}

/* Determines the Median of the given data */

float median(float *x, float *y, int N);
for(i=0; i<n; i++)
    *(y+i) = *x++;
qusort(y, n, sizeof(*y), tst);
return(y[n>>1]);
}
tst(x, y)
float *x, *y;
{
    if(*x < *y ) return(-1);
    else if( *x == *y ) return(0);
    else return(1);
}

Fast Fourier Transform

/*
 * Main FFT Program - determines command line options etc.
 * File - fftmain.c
 */
#include <stdio.h>
#include <math.h>
#define DIRECT 0
#define INVERSE 1
#define ON 1
#define OFF 0
char *progname;
char *help="man fft";
extern char *optarg;
extern int optind;
extern int getopt();

main(argc, argv)
int argc;
char *argv[];
{
    extern void hamming();
    extern void fft();
    extern int atoi();
    extern double log10(), atof();
    extern char *calloc();

    FILE *efopen(), *fpin, *fcout;
struct {
    unsigned real_ip:1; /* real/Complex input data */
    unsigned window:1; /* hamming window on/off */
    unsigned complex_op:1; /* Complex/Power output */
    unsigned log_op:1; /* log/linear output power */
    unsigned araspect:1; /* A.R. Spectra */
    unsigned error_ip:1; /* Error power present for A.R. */
    unsigned half_op:1; /* Output only 1st half of spectrum */
    unsigned normalise:1; /* Normalise Output */
} flags;

int i;
int M;
int N = 512; /* Size of Transform 512 default */
int size; /* Actual No. of Data points */
int inv = DIRECT; /* Time -> Frequency Domain */
char ch;

float *real;
float *imag;
float *PSD;
float fs = 1.0; /* Sampling frequency=1.0 default */
float e = 1.0; /* A.R. error power = 1.0 default */
float scale,max;

progname = argv[0];
fpin = stdin; /* default input from stdin */
fpout = stdout; /* output to stdout */

/*
Default settings:
Complex i/p data
Window on
Log Power o/p
*/
flags.real_ip = OFF;
flags.window = ON;
flags.log_op = ON;
flags.complex_op = OFF;
flags.arspect = OFF;
flags.error_ip = OFF;
flags.half_op = OFF;
flags.normalise = OFF;

if(argc < 3){
    mistake:
    system(help);
    exit(1);
}
while((ch = getopt(argc,argv,"acef:hiln:No:s:rw"))!=EOF){
    switch(ch){
    /*
Transform Size
*/
case 'n':
    N = atoi(optarg);
    break;

/*
Normalise O/P
*/

case 'N':
    flags.normalise = ON;
    break;

/*
Window off
*/

case 'w':
    case 'W':
    flags.window = OFF;
    break;

/*
Inverse transform
*/

case 'i':
    case 'I':
    inv = INVERSE;
    flags.complex_op = ON;
    break;

/*
Linear scale
*/

case 'l':
    flags.log_op = OFF;
    break;

/*
Complex o/p
*/

case 'c':
    flags.complex_op = ON;
    break;

/*
Autoregressive data
*/

case 'a':
    flags.arspect = ON;
    flags.window = OFF;
    flags.real_ip = ON;
    break;

/*
Error power i/p
*/

case 'e':
    flags.error_ip = ON;
    break;

/*
Input sampling frequency

```c
    case 's':
        fs = (float)atof(optarg);
        break;
```

Output file

```c
    case 'o':
        fpout = efopen(optarg,"w");
        break;
```

Input file

```c
    case 'f':
        fpin = efopen(optarg,"r");
        break;
```

Output Half points

```c
    case 'h':
        flags.half_op = ON;
        break;
```

real i/p data

```c
    case 'r':
        flags.real_ip = ON;
        break;
        case '?':
        default:
            goto mistake;
            break;
    }

if(argc== -1) goto mistake;

Check TSIZE within range

```c
    if(N <= 0 ){
        printf(stderr,"\nInvalid Transform size !!\n", N);
        exit(1);
    }
```

Make sure N is a power of 2

```c
    for(M=0,i=N;i;M++)i = i >> 1;
    N = 1 << ( M-1 );
```

Fetch memory for arrays
real = (float *)ecalloc(N,sizeof(float));
imag = (float *)ecalloc(N,sizeof(float));

/* Check for A.R. spectrum with error power and read error */
if( flags.error_pip && flags.arspect )
    fscanf(fpin,"%e",&e);

/* Real/Complex input data */
if(flags.real_ip)
    for(i=0;
i<N &&
       (fscanf(fpin,"%f",real+i))!=EOF;
i++)imag[i]=0.0;
else
    for(i=0;
i<N &&
       (fscanf(fpin,"%f%f",real+i,imag+i))!=EOF;
i++)
/* Pad with zeros to Transform length */
size=i;
for(;i<N;i++)real[i]=imag[i]=0.0;
/* Perform fft */
fft_2(real,imag,N,inv);
if(size < N){
scale = (float)N/(float)size;
    for(i=0;i<N;i++){
       real[i] *= scale;
       imag[i] *= scale;
    }
}
M = N;
if(flags.real_op){
    N=N/2+1;
    N++;
}
/* Window the Spectrum in the Frequency Domain */
if(flags.window){
    hamming(real,N);
    hamming(imag,N);
}
PSD = (float *)ecalloc(N,sizeof(float));
for(i=0;i<N;i++)
```c
PSD[i] = real[i] * real[i] + imag[i] * imag[i];

if (flags.aspect) {
    if (e < 0.0) e = -1.0 * e;
    for (i = 0; i <= N; i++) {
        if (PSD[i] <= 0.0) PSD[i] = 1e-20;
        else PSD[i] = e / PSD[i];
    }
}

if (flags.normalise) {
    for (max = -1.0, i = 0; i < N; i++) if (PSD[i] > max) max = PSD[i];
    for (i = 0; i < N; i++) PSD[i] /= max;
}
else if (flags.half_op) for (i = 0; i < N; i++) PSD[i] *= 2.0;
fs = fs / M;
if (flags.complex_op) {
    for (i = 0; i < M; i++)
        fprintf(fpout, "%-8.8g \t% -8.8g\n", real[i], imag[i]);
}
else if (flags.log_op) {
    for (i = 0; i < N; i++)
        if (PSD[i] > 1e-20)
            fprintf(fpout, "%-8.8g \t% -8.8g\n", fs * i, 10.0 * log10((double)PSD[i]));
        else
            fprintf(fpout, "%-8.8g \t% -8.8g\n", fs * i, -200.0);
}
if (!flags.half_op) {
    if (PSD[0] > 1e-20)
        fprintf(fpout, "%-8.8g \t% 8.8g\n", fs * N, 10.0 * log10((double)PSD[0]));
    else
        fprintf(fpout, "%-8.8g \t% -8.8g\n", fs * N, -200.0);
}
else {
    for (i = 0; i < N; i++)
        fprintf(fpout, "%-8.8g \t% -8.8g\n", fs * i, PSD[i]);
    if (!flags.half_op)
        fprintf(fpout, "%-8.8g \t% -8.8g\n", fs * N, PSD[0]);
}
```
Parameters:

- **REAL** - base address of REAL data
- **IMAG** - base address of IMAG data
- **N** - Transform Size (power of 2)
- **inv** - Boolean variable
- **0** Forward Transform
- **1** Inverse Transform

```c
#include <stdio.h>
#define reg register

void fft_2(REAL, IMAG, N, inv)
float REAL[], IMAG[];
int N, inv;
{
    extern double cos(), sin(), atan();

    int N2,
        N1,
        pass,
        step,
        span;

    reg int i,
        j,
        k,
        m;    /* Counters */

    reg float *rp,
            *ip,
            *rpl,
            *ipl,
            *finish;

    /* Twiddle Factor recursion variables */
    float wr,
            wi,
            ur,
            ur,
            ul,
            vr,
            vl,
            tr,
            ti;    /* Temporary variables */
```
double PI;

/*
 * Determine the Number of Passes
 */

PI = 4.0*atan(1.0);
i = N;
for(m= -1;i;m++) i = i>>1;
/*
 * Bit-reversal shuffle
 */
N2 = N>>1;
N1 = N-1;
for(j=i=0;i<N1;i++){
    if(i<j){
        tr = REAL[j];
        REAL[j] = REAL[i];
        REAL[i] = tr;
        ti = IMG[j];
        IMG[j] = IMG[i];
        IMG[i] = ti;
    }
    k = N2;
    j++;
    while(k<j){
        j -= k;
        k = k >>1;
    }
    j = j + k -1;
}
/*
 * m passes
 */
finish = &REAL[N];
for(pass = 1 ; pass <= m; pass++ ){
    step = (1 << pass);
    span = step>>1;
    ur = 1.0;
    ui = 0.0;
    wr = (float)cos((double)PI/span);
    if(inv)
        wi = (float)sin((double)PI/span);
    else
        wi = -(float)sin((double)PI/span);
    /*
    N/2 butterflies
    */
    for( j=0 ; j < span; j++){
        rp = &REAL[j];
        ip = &IMG[j];
        while( rp < finish ){
\[
\text{rpl} = \text{rp} + \text{span}; \\
\text{ipl} = \text{ip} + \text{span};
\]

\[
\text{/*} \\
\text{Complex multiply} \\
\text{*/}
\]

\[
\text{tr} = *\text{rpl} \ast \text{ur} - *\text{ipl} \ast \text{ui}; \\
\text{ti} = *\text{ipl} \ast \text{ur} + *\text{rpl} \ast \text{ui};
\]

\[
\text{Radix-2 Butterfly} \\
\text{*/}
\]

\[
*\text{rpl} = *\text{rp} - \text{tr}; \\
*\text{ipl} = *\text{ip} - \text{ti}; \\
*\text{rp} += \text{tr}; \\
*\text{ip} += \text{ti}; \\
\text{rp} += \text{step}; \\
\text{ip} += \text{step};
\]

\[
\text{Next Twiddle factor by recursion} \\
\text{*/}
\]

\[
\text{vr} = \text{ur} \ast \text{wr} - \text{ui} \ast \text{wi}; \\
\text{vi} = \text{ur} \ast \text{wi} + \text{ui} \ast \text{wr}; \\
\text{ur} = \text{vr}; \\
\text{ui} = \text{vi};
\]

\[
\text{if(! inv)}
\]

\[
\text{rp} = \text{REAL} ; \\
\text{ip} = \text{IMAG} ; \\
\text{while}(\text{rp} < \text{finish})
\]

\[
*\text{rp}++ /= \text{N}; \\
*\text{ip}++ /= \text{N};
\]

\]

\[
\text{Hamming window for FFT and DFT Analysis. -- File hamm.c}
\]

\[
\text{Analysis of Time series, Chatfield} \\
\text{NY. 1975 Chapman & Hall.}
\]

\[
(-0.23F(n-1) , 0.54F(n) , -0.23F(n+1) )
\]

\[
\text{void hamming(array\[\]}, \text{float array\[\]);} \\
\text{int n;} \\
\{ \\
\]
float *tmp,
    *start,
    *end,
    *x0,
    *x1,
    *x2;

extern char *ecalloc();
/*
 * Allocate Temporary Memory
*/
tmp = (float *)ecalloc(n,sizeof(float));
start = tmp;
end = &tmp[n-1];
x0 = array;
x1 = x0 + 1;
x2 = x1 + 1;
/*
 * Calculate Beginning and end points
*/
*start++ = -0.23*(array[n-1] + *x1 )+0.54* *x0;
*end = -0.23*( *x0 + array[n-2] )+0.54* array[n-1];
while(start < end)
    *start++ = -0.23 * ( *x0++ + *x2++ ) + 0.54 * *x1++;
/*
 * Copy Windowed points back to original array
*/
start = tmp;
end = tmp+n;
xll = array;
while(start < end)
    *x0++ = *start++;
/*
 * free up allocated memory
*/
free(tmp);
}

Autoregressive Spectral Analysis

/*
 * Shell A.R. routine. - processes command line options etc.
*/
#include <stdio.h>
#include <math.h>
#define BURG 0
#define OTBURG 1
#define BMBURG 2
#define YW 3
#define ON 1
#define OFF 0
main(argc, argv)
int argc;
char *argv[];
{
    FILE *fpin, *fpout, *fptmp, *fopen();

    int order,
    poles,
    N,
    i,
    select,
    type,
    log,
    TSIZE;

    float *X, /* Data */
    *R, /* ACF Matrix */
    **A, /* A.R. coefficients */
    *E, /* Error Power */
    *PSD, /* A.R Power Spectrum */
    *ap;

    float x,
    fs;

    char *prog = argv[0],
    *tmpfile = "TMP.AR",
    *help = ".

Usage: %s

    -f infile (stdin)
    -o outfile (stdout)
    -n # Model Order
    -a[yboe] Algorithm selection
    -t # Transform Size (512)
    -l Log Spectrum Linear
    -s # Sampling Rate (1.0)
    -p Output A.R. coefficients & Error Power

\n\nDetermine the A.R. coefficients/Spectrum for the input data using one of the specified algorithms and the given model order.

extern char *ecalloc();

extern int yw(),
extern double atof(),
log10();

fpin = stdin;
fpout = stdout;
log=OFF;
TSIZE = 512;
fs = 1.0;
type = 0;
if(argc < 3) goto error;
while(--argc && (*++argv)[0]=='-'){
    switch(*(*(argv[0]+1))){
        case 'f':
            fpin = efopen(*++argv,"r");
            argc--;
            break;
        case 'o':
            fpout = efopen(*++argv,"w");
            argc--;
            break;
        case 'l':
            log=ON;
            break;
        case 'a':
            switch(*(*(argv[0]+2))){
                case 'b':
                    select=BURG;
                    break;
                case 'o':
                    select=OTBURG;
                    break;
                case 'e':
                    select=EBBURG;
                    break;
                case 'y':
                    select=YW;
                    break;
            }
            break;
        case 's':
            fs = (float)atof(*++argv);
            argc--;
            break;
        case 't':
            TSIZE = atoi(*++argv);
```
argc--;  
break;

case 'p':
    type = 1;
    break;

case 'n':
    order = atoi(*++argv);
    argc--;
    break;

default:
    error:
        fprintf(stderr,help,prog);
        exit(1);
        break;
}
}
if(fpin == stdin){
    ftmp = fopen(tmpfile,"w");
    N=0;
    while(fscanf(fpin,"%f", &x)!EOF){
        fprintf(ftmp,"%f\n", x);
        N++;
    }
    fclose(ftmp);
    fpin = fopen(tmpfile,"r");
}
else {
    N=0;
    while(fscanf(fpin,"%f", &x)!EOF)N++;  
}
/*
 * Allocate Memory
 */
X = (float *) calloc(N,sizeof(float));
E = (float *) calloc(order+1,sizeof(float));
A = (float **) calloc((order+1)*(order+1),sizeof(float));
PSD = (float *) calloc((TSIZE>>1)+1,sizeof(float));
/*
 * Read Data
 */
rewind(fpin);
for(i=0;fscanf(fpin,"%f",X+i)!EOF;i++){
    /*
    * Determine A.R Coefficients
    */
    switch(select){
    case BURG:
        poles = burg(X,N,A,order+1,order,E);
        break;
    case EWBURG:
        poles = ewburg(X,N,A,order+1,order,E);
        break;
    ```
case OTBURG:
    poles = otburg(X,N,A,order+1,order,E);
    break;

case VW:
    R = (float *)ecalloc(order+2,sizeof(float));
    acf(X,N,R,order+1);
    poles = yw(R,A,order+1,order,E);
    break;

if(poles!=order)
    fprintf(stderr, "Model order used = %d\n", poles);
    ap = (float *)A + (order+1)*poles;
if(type){
    fprintf(fpout,"Model order used = %d\n",poles);
    fprintf(fpout,"\tA.R coefficients \n");
    for(i=0;i<=poles;i++)
        fprintf(fpout,"\t%f\n",*ap++);
    fprintf(fpout,"\tEstimation Error = %f\n", E[poles]);
}
else {
    arfft(ap,poles,PSD,TSIZE);
    if(log) for(i=0;i<(TSIZE)>>1;i++)
        PSD[i] = (float)10.0*log10((double)PSD[i]);
    fs /= TSIZE;
    for(i=0;i<(TSIZE)>>1;i++)
        fprintf(fpout,'%f %g\n",fs,PSD[i]);
}
unlink(tmpfile);

Yule Walker

/*

*YULE-WALKER METHOD TO CALCULATE AUTOREGRESSIVE SPECTRAL COEFFICIENTS USING LEVINSON RECURSION.

* Parameters:
* A[][ASIZE] - A.R. Coefficient Matrix
* E[ ] - Prediction Error Matrix
* R[ ] - ACF data array,
* ORDER - Model Order
* ASIZE - Number of columns in A[][]
*/

#define reg register

yw(R,A,ASIZE,ORDER,E)
float **A,*E,*R;
int ORDER,ASIZE;
{
```c
/* Pointer to ACF array */
*ep, /* Error Power Array */
*past, /* Previous row of A[][] */
*row, /* Current row of A[][] */
*finish;

float K, sum;

int order; /* Model Order */

/* INITIALISATION */
ep = E;
ep = *rp;
row = (float *)A + ASIZE; /* &A[1][0] */
K = -rp[1] / rp[0];
row[1] = K;
K = K * K;
++ep = (1.0 - K) * rp[0];

/* MAIN LOOP */
while (past < finish)

/* Increment the Order of the System */
/* move to next row in A[][] */
past = row;
finish = row + order;
row += ASIZE;
/* Calculate the High order Coefficient */
*row = 1.0;
sun = 0.0;
while (past < finish)
    sum += *past++ * *rp--;
K = -sum / ep[0];
row[order] = K;
/* Perform Levinson Recursion to determine */
/* the lower order coefficients */
Levinson(A, ASIZE, K, order);
/* Calculate the prediction error */
```
```c
/*
K = K*K;
if( K > 1.0)
    return(--order);
ep[1] = (1.0 - K) * ep[0];
ep++;
}
return(--order);
*/

Burg

/*
BURG ALGORITHM TO EVALUATE AUTOREGRESSIVE SPECTRAL COEFFICIENTS.
*
* Parameters:
*  X[] - data array
*  N   - No. of data points.
*  A[][]ASIZE - autoregressive coefficients.
*  ASIZE - No. of columns in A[][]
*  ORDER - Model ORDER to be used
*  E    - noise power.
*/

#define reg register
burg(X,N,A,ASIZE,ORDER,E)
float X[],*A[],E[];
int N,ASIZE,ORDER;
{
    int order; /* Model Order */
    float q,     /* Error Power */
         den,   /* Denominator */
         num,   /* Numerator */
         K,     /* Reflection Coeff. */
         *F,    /* Forward Prediction Errors (FPE) */
         *B,    /* Backward Prediction Errors (BPE) */
    temp;
    reg float *fp,    /* FPE Array pointer */
            *bp,    /* BPE */
            *xp,    /* data */
            *ep,    /* Error power array pointer */
            *finish,/* general pointer */
            *row,   /* Ptr to current order row of A[][] */
    char *ecalloc();
    /*
    INITIALISATION
    */
```
Dynamic Fetch of memory for arrays

```c
F = (float *)ecalloc(N+1,sizeof(float));
B = (float *)ecalloc(N+1,sizeof(float));
```

Set row pointer to &A[0][0]
and other pointers to the begin of
there appropriate arrays

```c
row = (float *)A;
fp = F;
bp = B;
xp = X;
ep = E;
```

Initialise FPE & BPE arrays for order 0
and calculate total forward & backward
prediction error energies.

```c
finish = &F[N];
num = 0.0;
while( fp < finish ){
    num += *xp * *xp;
    *fp++ = *bp++ = *xp++;
}
```

den = 2.0*num;
ep = num/N;
rowq = 1.0;

```c
for(order = 1; order <= ORDER; order++){
```

Increment row to point at A[order][0]

```c
row += ASIZE;
```

```c
row = 1.0;
```

```c
Calculate the first cross-correlation term
of the forward and backward prediction errors
```

```c
bp = &B[order-1];
fp = &F[order];
finish = &F[N];
num = 0.0;
temp = 0.0;
while( fp < finish )
    num += *bp++ * *fp++;
```
Calculate the total sum of the forward & backward prediction error energies recursively

```
```

PARCOR coefficient calculation

```
row[order] = K = -2.0*num/den;
q = 1.0 - K*K;
ep[1] = ep[0] * q;
```

Check for numerical stability PARCOR <= 1.0

```
if(q < 0) {
    free((char*)F);
    free((char*)B);
    return(--order);
}
```

LEVINSON RECURSION

Determines the A.R. coefficients from the PARCOR coefficients.

```
if(order != 1) Levinson(A,A SIZE,K,order);
```

UPDATE PREDICTION ERRORS FOR NEXT MODEL ORDER

```
fp = &F[N-1];
bp = &B[N-1];
finish = &F[order-1];
while(fp > finish ){
    temp = *fp;
    *fp = *fp + K * bp[-1];
    *bp = bp[-1] + K * temp;
    fp++, bp--;
}
ep++;
```

Free allocated memory and return

```
order--;
free((char*)F);
free((char*)B);
return(order);  /* Returns number of poles required. */
```
Energy Weighted Burg

/**
* ENERGY WEIGHTED BURG ALGORITHM
* TO EVALUATE AUTOREGRESSIVE SPECTRAL COEFFICIENTS.
*
* Parameters:
* X[] - data array
* N - No. of data points.
* A[][] - autoregressive coefficients.
* ASIZE - No. of cols in A[][]
* ORDER - No. of poles to be used
* E[] - noise power.
*/
#define reg register

ewburg(X,N,A,ASIZE,ORDER,E)
float X[],A[],E[];
int N,ASIZE,ORDER;
{
    int order, /* Counters */
    m,k;
    float *F, /* Forward Prediction Errors */
        *B, /* Backward Prediction Errors */
        numf, /* Forward Numerator */
        numb, /* Backward Numerator */
        denf, /* Forward Denominator */
        denb, /* Backward Denominator */
        Ef, /* Forward window weight */
        Eb, /* Backward window weight */
        Kf, /* Forward PARCOR coefficient */
        Kb, /* Backward PARCOR coefficient */
        K, /* Harmonic Mean PARCOR coefficient */
        temp,
        q;
    reg float *fp, /* FPE Array ptr */
            *bp, /* BPE Array ptr */
            *ep, /* Noise Power Array ptr */
            *row, /* Ptr to current row of A[][] */
            *xp, /* Ptr to data Array */
            *finish;

    extern char *ecalloc();

    /*
    * INITIALISATION
    */
    /*
    * Dynamically fetch memory for Arrays
    */
}


```c
/*
 * Initialise Array pointers to their associated arrays
 */
fp = F;
xp = X;
bp = B;
ep = E;
/*
 * Initialise FP Errors & BP Errors for order 0 and calculate total power
 */
*ep = 0.0;
finish = F + N;
while(fp < finish){
   *ep += *xp * *xp;
   *fp = *bp = *xp;
   fp++, bp++, xp++;
}
*ep = *ep/N;
row = (float *)A;
*row = q =1.0;
/*
 * COMPUTE REFLECTION COEFFICIENT
 */
for( order = 1; order <= ORDER; order++ ){
   /*
   * row points to row of A[order][]
   */
   row += ASIZE;
   *row = 1.0;
   /*
   * Calculate Energy Weighted Forward & Backward Prediction error cross-correlation and weighted total error energies.
   */
   bp = &B[order - 1];
   fp = &F[order];
   denf = denb = numf = numb = 0.0;
   for( k = order; k < N; k++ ){
      /*
      * Calculate the forward & backward energy weights for time k
      */
      Ef = Eb = 0.0;
      for( m = 1; m <= order; m++ ){
         Ef += X[k-m] * X[k-m];
         Eb += X[k+m-order] * X[k+m-order];
      }
   }
}
*/
```
/ Calculate the cross-correlation
* between F[k] & B[k-1]
*/

temp = *bp * fp;

/*
* Weight and add into correlation sum
*/

numf += temp * Ef;
numb += temp * Eb;

/*
* Add in the weighted error energies
*/

denf += *bp * bp * Ef;
denb += *fp * fp * Eb;

/*
* Increment ptrs for next time instant k
*/

bp++; fp++;

} /*
* Determine Forward, Backward and thence
* Harmonic (Burg) PARCOR Coefficients
* for current model order.
*/

Kf = numf/denf;
Kb = numb/denb;
row[order] = K = 2.0*Kf*Kb/(Kf+Kb);
q = 1 - K*K;

/*
* Check for Stability
*/

if( q < 0.0 ){

/*
* If Unstable free allocated memory
* and return last stable model order
*/

free((char*)B);
free((char*)F);
return(--order);

} /*
* Determine noise power for current model order
*/

ep[1] = ep[0] * q;
ep++;

/*
* LEVINSON RECURSION
* Calculates the A.R. coefficients from the
* PARCOR coefficient for current model order
*/
if(order != l) Levinsoo(A, ASIZE, k, order);
/*
 * UPDATE PREDICTION ERRORS
 */
fp = &F[N-1];
bp = &B[N-1];
finish = &F[order-1];
while(fp > finish){
temp = *fp;
    *fp = *fp + K * bp[-1];
    *bp = bp[-1] + K * temp;
fp--; bp--;
}
/*
 * Free allocated memory and return
 * model order.
 */
free((char*)B);
free((char*)F);
return(--order);

Optimum Tapered Burg

/*
 *
 * OPTIMUM TAPERED BURG ALGORITHM
 * TO EVALUATE AUTOREGRESSIVE SPECTRAL COEFFICIENTS.
 *
 *
 * Parameters:
 * X[] - Data Array
 * N - Size of X[]
 * A[] - A.R Coefficient Array
 * ASIZE - Number of columns in A[]
 * ORDER - Model ORDER to use
 * E[] - Error Power Matrix
 */
#define reg register

otburg(X,N,A,ASIZE,ORDER,E)
float X[],*A[],E[];
int N,ASIZE,ORDER;
{
    extern char *ecalloc();

    int order;

    float q,
        den, /* Numerator */
num. /* Denominator */
K. /* PARCOR coeff. */
lambda,
w_0. /* Window weights */
w_1.
w_2.
"F. /* Forward & Backward Prediction */
"B. /* error arrays */
temp;

/* Pointers to Arrays */
reg float "bp. /* Backward Prediction errors */
"fp. /* Forward */
"xp. /* Input Data Array */
"row. /* A.R. coefficient Array */
"ep. /* Noise Power Array */
"finish;

/*
* INITIALISATION
*/

/* Dynamically Allocate memory for Arrays */
F = (float *)calloc(N,sizeof(float));
B = (float *)calloc(N,sizeof(float));

/* Initialise ptrs to arrays */
fp = F;
bp = B;
xp = X;
ep = E;

/* Initialise Forward & Backward error arrays */
and calculate noise power for order 0 */
num = 0.0;
finish = F + N;
while(fp < finish ){
    num += "ip " 'fp+-4- = "bp+-4- =
    *fp++ = "bp++ = "xp++;
}
*ep = num/N;

/* row points to A[0][0] */
row = (float *)A;
*row = q = 1.0;

/*
* COMPUTE REFLECTION COEFFICIENTS */
for(order = 1; order <= ORDER; order++) {
  /*
   * row points to A[order][0]
   */
  row += ASIZE;
  row = 1.0;
  /*
   * Time k = 0 :
   * Calculate window value and
   * partial Forward/Backward cross-correlation.
   */
  bp = &B[order-1];
  fp = &F[order];
  lamda = 12.0/((N-order)*(N-order+1)*(N-order+2));
  w_2 = (N-order)*lamda/2.0;
  num = w_2 * *bp * *fp;
  den = w_2 * (*bp * *bp + *fp * *fp);
  bp++;
  fp++;
  /*
   * Time = 1 :
   * Calculate window value,
   * weighted partial Forward/Backward cross-correlation.
   * weighted forward/backward error energies.
   */
  w_1 = 2*w_2 - lamda;
  num += w_1 * *bp * *fp;
  den += w_1 * (*bp * *bp + *fp * *fp);
  bp++;
  fp++;
  /*
   * Time = 2,3,...N :
   * Calculate window weights recursively.
   * weighted partial Forward/Backward cross-correlation.
   * weighted forward/backward error energies.
   */
  finish = &F[N];
  while( fp < finish ){
    w_0 = 2.0*w_1 - w_2 - lamda;
    num += w_0 * *bp * *fp;
    den += w_0 * (*bp * *bp + *fp * *fp);
    bp++;
    fp++;
    w_2 = w_1;
    w_1 = w_0;
  }
  /*
   * Calculate PARCOR Coefficient
   */
  row[order] = K = -2.0*num/den;
  q = 1.0 - K*K;
  /*
* Check for stability
*/
if(q < 0.0){
    free((char*)B);
    free((char*)F);
    return(--order);
}
/*
* Update noise power
*/
ep[1] = ep[0] * q;
/*
* LEVINSON RECURSION
*/
if(order != 1)Levinson(A,ASIZE,K,order);
/*
* PREDICTION ERRORS UPDATE
*/
fp = &F[N-1];
bp = &B[N-1];
finish = &F[order-1];
while( fp > finish ){
    temp = *fp;
    *fp = *fp + K * bp[-1];
    *bp = bp[-1] + K * temp;
    bp--;fp--;
}
ep++;
/*
* Free allocated memory
*/
free((char*)B);
free((char*)F);
return(--order); /* Return number of poles */

Support routines

/*
* Levinson recursion to determine Inverse of a Toeplitz Matrix A[][]
*/

Levinson(A,ASIZE,K,order)
float * ** A;
float K;
int ASIZE, order;
{

register float *pres,
    *past,
    *last,
    *finish;

past = (float *)A + (order-1)*ASIZE + 1;
pres = past + ASIZE;
last = past + order - 2;
finish = pres + order - 1;

while(pres < finish)
{
    *pres++ = *past++ + K * *last--;
}

/*
* Final Stage to produce a A.R Spectrum using a FFT
*
* Parameters:
*    A[] - A.R coefficients
*    P - Number of Poles
*    PSD - Final A.R Spectrum
*    TSIZE - Transform Size
*/
#include <math.h>
#define reg register
arfft(A,P,PSD,TSIZE)
float A[], PSD[];
int P, TSIZE;
{
    extern char *ecalloc();
    extern int free();

    float *REAL, *IMAG;
    reg float *rp, *ip, *pwr, *finish;
    int i;

    REAL = (float *)ecalloc(TSIZE, sizeof(float));
    IMAG = (float *)ecalloc(TSIZE, sizeof(float));
    /*
    * Copy A.R coefficients to Transform Array
    */
    rp = REAL;
    ip = IMAG;
    for(i=0; i<=P; i++)
    {
        *rp++ = A[i];
        *ip++ = 0.0;
    }
/ * Zero Pad remaining points */
finish = &REAL[TSIZE];
while(rp < finish){
  rp++ = 0.0;
  ip++ = 0.0;
}
/*
 * Fourier Transform */
fft_2(REAL, IMAG, TSIZE, 1);
/*
 * Calculate the A.R Power Spectrum */
rp = REAL;
ip = IMAG;
pwr = PSD;
finish = &REAL[(TSIZE>>1)];
while(rp <= finish ){
pwr = *rp * *rp + *ip * *ip;
  rp++;
ip++;
  if( *pwr > 0.0 )
    *pwr = 1.0/*pwr*/;
  else *pwr = HUGE;
pwr++;
}
free((char *)REAL);
free((char *)IMAG);

/*
 * correlation function
 *
 * Parameters:
 *  X[] - data
 *  N - No. of data points
 *  R[] - Correlation function
 */
acf(X,N,R, lags)
float X[]; float R[];
int N, lags;
{
  float *x, /* Pointers to Data*/
  *y,
  *r, /* Pointer to ACF */
  *finish;
int i; /* Counter */
LMS Adaptive Line Enhancement

#include <stdio.h>
define TSIZE 512
define S_1HZ 50
define S_4HZ 205
define TAPS 0
define PEAK 1
define SPECTRUM 2

define reg register

extern float fir();

char *help= "\n\nUsage %s:\n\n-f $ Input data file\n-o $ Output results file\n-a # Adaption Time \n-n # Number of Coefficients\n-d # Number of delay elements\n-t # Output after ? input samples\n-s # Start bin of spectrum\n-e # End bin of spectrum\n-N # Size of Transform\n-c O/P coefficients\n-p O/P peak\n\nUses the LMS adaptive filter on the input data to:\ncalculate the AR coefficients and thence the Spectrum\n";

char *progsname;
main(argc,argv)
int argc;
char **argv;
{
    extern char *calloc();
    extern int optind;
    extern char *optarg;

float *X, /* Input Data */
    *A, /* Filter coefficients */
    y, /* Predicted Output */
    e, /* Error */
    mu, /* Convergence Factor */
    mu_c, /* Mu error */
    alpha,
    Ta; /* Adaptation time */

FILE *fp,*fo,*efopen();

float *ap,
    *finish,
    sample,
    *PSD;

int N,
    tsize = TSIZE,
    OP_time = 1,
    start = S_1HZ,
    end = S_4HZ,
    delays,
    type = SPECTRUM,
    count = 0,
    i;

proname = argv[0];
fp = stdin;
fo = stdout;
delays = 1;
    != EOF){
    switch(ch){
    case 'f': fp = efopen(optarg,"r"); break;
    case 'o': fo = efopen(optarg,"w"); break;
    case 'a': sscanf(optarg,"%f",&Ta); break;
    case 's': sscanf(optarg,"%d",&start); break;
    case 'e': sscanf(optarg,"%d",&end); break;
    case 'N': sscanf(optarg,"%d",&tsize); break;
    case 't': sscanf(optarg,"%d",&OP_time); break;
    case 'n': sscanf(optarg,"%d",&N); break;
}
break;
case 'd':    sscanf(optarg,"%d",&delays):
    break;
case 'p':    type = PEAK;
    break;
case 'c':    type = TAPS;
    break;
case 'q':
case 'q':
default:  fprintf(stderr,-help,proglme);  
exit(0);
}
if(N <= 0) error("Incorrect Number of taps\n");
X = (float *)calloc(N+delays+1,sizeof(float));
A = (float *)calloc(N+delays+1,sizeof(float));
PSD = (float *)calloc(tsize,sizeof(float));
/*
 * Initialise Shift register and Taps
 */
init(X,A,N,delays,Ta,&alpha);
/*
 * For each new input sample
 */
mu = 0;
count = 0;
while(fscanf(fp,"%f",&sample)!=EOF){
    /*
    * calculate predicted filter output
    */
y=fir(&A[delays],&X[delays],N);
    /*
    * Determine estimate error
    */
e = X[0] + y;  
    /*
    * Adapt the filter weights using LMS algorithm
    */
    mu_e = mu * e;
lms(&A[delays],&X[delays],N,mu_e);
    /*
    * Clock the Shift register
    */
    shift(X,N+delays,sample,alpaha,&mu);
    if( ++count >= OP_time){
    /*
    * Output Results
    */
    switch(type){
case TAPS:
    for (i=0; i<NH; i++)
        if ((i+1)%6 != 0) fprintf(fo, "%f\t", A[i]);
        else fprintf(fo, "%f\t", A[i]);
    fprintf(fo, "\n\n\n");
    break;

case SPECTRUM:
    /*
     * Calculate the AR Spectrum from
     * filter weights
     */
    arfft(A,N+delays,PSD,tsize);
    for (i=start; i<=end; i++)
        fprintf(fo, "%f\n",
                ((float)i)/tsize,PSD[i]);
    break;

case PEAK:
    arfft(A,N+delays,PSD,tsize);
    fprintf((o,
                ((float)peak(PSD, start ,end))/tsize);
    break;

flush(fo);
count = 0;
}

/ *
   * Initialise LMS Adaptive filter
   */
init(X,A,N,D,Ta, alpha)
float A[],X[],Ta;*alpha;
int N,D;
{

extern double exp();
reg float *xp = X,
        *ap = A,
        *finish = A + N + D;

*ap++ = 1.0;
while ( ap < finish ){
    *ap++ = 0.0;
    *xp++ = 0.0;
}
*/
* Compute alpha from adaption time Ta */
*alpha = (1.0 - (float)exp(-1.0/(double)Ta))'N;

fprintf(stderr, 
"Adaption Time = %f samples --> alpha = %g\n", Ta,"alpha);
}

error(s)
char *s;
{
fprintf(stderr, 
%s :	

" ,progname,s);
exit(0);
}

peak(X,start,end)
float *X;
int start,end;
{

register float *xp = X + start;
register int pos,i;
register float *finish = X + end;
float max;

i = start;
max = -1e32;
while(xp < finish){
    if(*xp > max){
        max = *xp;
        pos = i;
    }
    i++;
    xp++;
}
return(pos);
}

/*
 * LMS Adaptive Filter algorithm
 */

lms(A,X,N,mu_e)
float A[], /* Coefficient Array */
X[], /* Input Data Array */
mu_e; /* Step-size = 2 * mu * e */
int N; /* Size of Arrays */
{

register float *xp, /* Pointer to Dat Array */
    *ap, /* Pointer to Coefficient Array */
    *finish; /* Pointer to end of Array */
ap = A;
xp = X;
finish = ap + N;
while(ap < finish)
    *ap++ = *ap - mu_c * *xp++;
}

/*
 * Shift delay line and insert new sample
 */

void shift(X,N,sample,alpha,mu)
float X[],sample,alpha,*mu;
int N;
{
    register float *xp,*yp;
    static float totalpower = 0.0;

    xp = X + N -1;
    yp = X + N -2;
    totalpower = totalpower + sample*sample - *xp * *xp;
    while( yp >= X ) *xp-- = *yp--;
    *xp = sample;
    if(totalpower == 0.0 ) totalpower = 10.0;
    *mu = alpha/totalpower;
}

/*
 * FIR filter subroutine
 */

float fir(A,X,N)
float X[],A[];
int N;
{
    register float  *xp,
                  *ap,
                  *finish;
    float y;

    xp = X;
    ap = A;
    finish = ap + N;
    y = 0.0;
    while( ap < finish )
        y += *ap++ * *xp++;
    return(y);
}
Gradient/Recursive lattice Filter Line Enhancement

/*
   Header file for Lattice Programs - file lattice.h
   *
#define squ (x) ((x)*(x))
#define reg register
#define TRUE 1
#define FALSE 0

#define KVALUES 1
#define ARVALUES 2
#define SPECTRUM 3
#define PEAK 4

typedef struct LATTICE {
    float f; /* Forward Prediction error */
    float b; /* Backward Prediction error */
    float c; /* Delayed Backward error */
    float K; /* Reflection coefficient */
    float Den; /* Denominator */
    float Num; /* Numerator */
} LATTICE;

/*
 *  File - lattice.c
 */

#include <stdio.h>
#include <math.h>
#include "lattice.h"

extern int optind;
extern char *optarg;

LATTICE *lattice; /* The Lattice */
float alpha; /* Recursive Filter Coeffs. */
int P; /* Model order */
char gradient=FALSE; /* Gradient algorithm flag */

FILE *fpin,*fput,*efopen();

main(argc,argv)
int argc;
char *argv[];
{
    extern double atof(),exp();
    extern int forward(),backward(),harmonic();
}
reg int (*update)(); /* Pointer to Lattice Update Algorithm */

float "A, /* A.R. coefficients */
*PSD, /* Power Spectral Density */
T adapting; /* Adaptation Time in # iteration */

int T, /* O/P after N I/P's */
i, count, /* iteration counter */
start=0, /* Starting bin of peak search */
end=0, /* ending bin of peak search */
tsize=512; /* Size of Transform */

char ch;
flag = SPECTRUM,
*programname = argv[0],
*ecalloc(),
*help = "\n\nUsage: %s
-f $ Input data file\n-o $ Output data file\n-b # Effective Block Size\n-p # Number of Poles\n-t # Output after ? samples\n-s # Starting Bin of Spectrum (default 0)\n-e # Last Bin of Spectrum (default 256)\n-n # Size of Spectrum (default 512)\n-a A.R. Coefficients Output\n-k Reflection Coefficients Output\n-g Gradient method\n\nUsing the Lattice filter structure the AR coefficients are calculated recursively over the effective data block length specified, from the computed PARCOR coeffs. Output may be the PARCOR or Direct AR coeffs., the ARPSD or the position of the global maxima in the AR PSD.

*error1 = "\n\nERROR INVALID -p option !!\n",
*error2 = "\n\nERROR INVALID -t option !!\n",
*error3 = "\n\nERROR INVALID -b option !!\n";

fpin = stdin;
fpout = stdout;
count=0;

Sort out various options from the command line

while(
!=EOF)
{
switch(ch){
case 'b': /* Block length */
T adapting = (float)atof(optarg);
}
case 'f':
    break;
    fpin = efopen(optarg,"r");
    break;

case 'p':
    P = atoi(optarg);
    break;

case 'o':
    fpout = efopen(optarg,"w");
    break;

case 't':
    /* Output Time */
    T = atoi(optarg);
    break;

case 'k':
    flag = KVALUES;
    break;

case 'a':
    /* A.R. Coefficients */
    flag = ARVALUES;
    break;

case 'm':
    /* PEAK */
    flag = PEAK;
    break;

case 's':
    /* Starting bin in Spectrum */
    start = atoi(optarg);
    break;

case 'c':
    case 'E':
        /* Last bin in Spectrum */
        end = atoi(optarg);
        break;

case 'n':
    case 'N':
        /* Size of DFT */
        tsize = atoi(optarg);
        break;

case 'g':
    gradient = TRUE;
    break;

case '?':
    case 'q':
        default:
            fprintf(stderr,help,progname);
            exit(0);
            break;

        }

    /*
     * Some Basic Input error checks
     */
    if( P <= 0 ){
fprintf(stderr.error1);
exit(1);
}
if( T <= 0 ){
    fprintf(stderr.error2);
    exit(2);
}
if( T_adapt <= 0 ){
    fprintf(stderr.error3);
    exit(3);
}
if(end == 0) end = tsize/2;
/
* Determine Filter coefficients and gradient stepsize
* factor from the desired block length
*/
beta = (T_adapt - 1)/(T_adapt + 1);
alpha = 1.0 - beta;
/
* Fetch Memory for the arrays
*/
PSD = (float *)calloc(tsize+1,sizeof(float));
A = (float *)calloc(P+1,sizeof(float));
lattice = (LATTICE *)calloc(P+1,sizeof(LATTICE));
/
* Initialise the lattice
*/
for(i=0;i<=P;i++){
    lattice[i].f = lattice[i].b = 0.0;
    lattice[i].c = lattice[i].Num = 0.0;
    lattice[i].K = 0.0;
}
fscanf(fpin,"%f",&lattice[1].c);
count++;
lattice[1].f = lattice[1].c;
for(i=0;i<=P;i++)
    lattice[i].Den = 2.0*sq(lattice[1].c);
/
* Decide on Update Algorithm
*/
if(strcmp(argv[0],"forward")==0)update=forward;
else if(strcmp(argv[0],"backward")==0)update=backward;
else update=harmonic;
/
* Repeat for each input sample until the end of file
*/
while(fscanf(fpin,"%f",&lattice[0].b)!EOF){
count++;
/*
* Time Update
*/
lattice[0].f = lattice[0].b;
update();
if(count >= T){
    count = 0;
    /*
     * Output Results
     */
    output(A, PSD, tsize, start, end, flag);
}
free(PSD);
free(A);
free(lattice);

#include "lattice.h"

/*
 * Burg - Harmonic Mean Algorithm to update the Lattice
 * and calculate the PARCOR coefficients
 */
harmonic()
{
extern LATTICE *lattice;
extern float alpha,
    beta;
extern int P;
extern char gradient;

reg LATTICE *last,
    *pres;

float mu;

last = &lattice[P-1];
pres = &lattice[P];
for(;pres > &lattice[0];pres--, last--){
    /*
     * Lattice Updates
     */
    pres->f = last->f + pres->K * last->c;
pres->b = last->c + pres->K * last->f;
pres->Den = beta*pres->Den +
    alpha*(squ(last->f) + squ(last->c));
pres->Num = beta*pres->Num +
    2.0*alpha*last->c*last->f;
    if(!gradient){
        pres->K = -pres->Num/pres->Den;
    }
```c
#endif
define reg register

/*
 * Uses the Levinson algorithm to convert Lattice PARCOR Coefficients into the Direct A.R. coefficients */

/**
 * lattice - Structure of the lattice
dir - Array to contain direct AR coefficients
* P - Model order / No. of lattice stages
*/
Latt_dir(lattice,dir,P)
LATTICE lattice[];
float dir[];
int P;
{
    int m, /* Counters */
    j;
    reg float **a, /* AR Coefficient Matrix */
    *prev, /* Pointers to the matrix */
    *pres,
    *row,
    K; /* The PARCOR coefficient */

    /* Fetch memory for the matrix */
    a = (float **)calloc((P+1)*(P+1),sizeof(float));
    /*
    Point row at &a[0][0]
    */
    pres = row = (float *)a;
    row[0] = 1.0;
```
for(m=1; m<=P; m++){
    prev = pres;  /* prev = &a[m-1][0] */
    pres = prev+P+1;  /* pres = &a[m][0] */
    pres[0] = 1.0;
    pres[m] = K = lattice[m].K;
    /*
     * The Levinson recursion
     */
    for(j=1; j<m; j++)
        pres[j] = prev[j] + K*prev[m-j];
}

/*
 * Copy the last row of the AR parameter matrix to the array dir[
 */
row += (P+1)*P;
for(j=0; j<=P; j++)
    dir[j] = row[j];

/*
 * Free up allocated memory
 */
free(a);

/*
 * File - peak.c
 */

/*
 * Finds the position of the global maximum in X[] between X[s] and X[e]
 */
peak(X, s, e)
float *X;
int s, e;
{
    register float *xp = X + s,
              *xend = X + e;

    float max;
    int pos,
        - i;

    max = -1e32;
    pos = s;
    i = s;
    while(xp < xend){
        if(*xp > max){
            max = *xp;
            pos = i;
        }
        xp++;
    }
i++;
}
return(pos);