Adaptive adjustment filter techniques applied to digital data receivers for telephone channels, HF & mobile radio links

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfillment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/10537](https://dspace.lboro.ac.uk/2134/10537)

Publisher: © N. I. Tsabieris

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
ADAPTIVE ADJUSTMENT FILTER TECHNIQUES APPLIED TO DIGITAL DATA RECEIVERS FOR TELEPHONE CHANNELS HF & MOBILE RADIO LINKS

by

Nikos I.Φ. Tsabieris

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy of the Loughborough University

July 1996

Supervisor: Mr. D.W. Hoare

Department of Electronic and Electrical Engineering

© by N. I. Tsabieris, 1996
To my parents

Yianni & Fotini

their love kept me going

Στους λατρεύτους μου γονείς

ΓΙΑΝΝΗ & ΦΩΤΕΙΝΗ

Η αγάπη τους με κρατήσε όρθο
Invictus

Out of the night that covers me,
Black as the pit from pole to pole,
I thank whatever gods may be
For my unconquerable soul.

In the fell clutch of circumstance
I have not winced nor cried aloud.
Under the bludgeoning of chance
My head is bloody, but unbowed.

Beyond this place of wrath and tears
Looms but the Horror of the shade,
And yet the menace of the years
Finds, and shall find, me unafraid.

It matters not how strait the gate,
How charged with punishments the scroll,
I am the master of my fate:
I am the captain of my soul.

W. E. Henley


Contents

ACKNOWLEDGEMENTS ................................................................. vii
LIST OF ILLUSTRATIONS .......................................................... viii
NOTATION .................................................................................. xii
ABSTRACT .................................................................................... xiii

1 CHAPTER ONE INTRODUCTION .................................................. 1

1.1 Background of the Research .................................................... 1
1.2 Outline of the Thesis .............................................................. 7

References .................................................................................. 9

2 CHAPTER TWO MODEL OF DATA TRANSMISSION SYSTEM ....... 13

2.1 Introduction ............................................................................. 13
2.2 Model of the Digital Transmission System ................................... 13
2.3 Linear Feedforward Estimator .................................................... 24
2.4 Maximum Likelihood Detection .................................................. 26
2.4.1 The Viterbi Algorithm Detection Process ................................. 33
2.5 Near Maximum Likelihood Detectors .............................................. 36
2.5.1 Reduced State Viterbi-Algorithm Detector ................................. 36
2.5.2 Simple Near Maximum Likelihood Detector ............................... 38
2.6 Adaptive Linear Feedforward Transversal Filter ......................... 41
2.7 Adaptive Adjustment Criteria ................................................... 43
3 CHAPTER THREE SOME ROOT-FINDING ALGORITHMS AND SPECTRAL FACTORIZATION TECHNIQUES FOR THE ADJUSTMENT OF THE PREFILTER

3.1 Introduction ........................................................................................................................................ 55

3.2.1 QR Method .................................................................................................................................... 56

3.2.i Newton-Raphson Method .......................................................................................................... 59
3.2.i.a Variant of Newton-Raphson (Truncated input) ........................................................................... 60
3.2.i.b Newton-Raphson combined with QR Method (QRN) ............................................................ 61
3.2.i.i Newton Raphson (Novel Application) ...................................................................................... 63

3.2.3.i Laguerre’s Method .................................................................................................................... 64
3.2.3.a Laguerre’s Method combined with QR (QRL) ......................................................................... 65
3.2.3.ii Laguerre’s Method (Novel Application) .................................................................................. 66

3.2.4 Adjustment scheme for the root finding algorithms 2 and 3 .......................................................... 66
3.2.4.i Adjustment scheme .................................................................................................................... 67

3.2.5 Clark-Hau algorithm ..................................................................................................................... 71

3.3 Spectral Factorisation Techniques .................................................................................................... 76
3.3.1 The Toeplitz method ................................................................................................................... 76
3.3.2 Bauer’s Method ........................................................................................................................... 79
3.3.3 Gram-Schmidt orthogonalisation Method ................................................................................... 80

3.3.4 N.I.T Method .............................................................................................................................. 83
3.3.4.a Modified N.I.T.a algorithm ..................................................................................................... 87
3.3.4.b Modified N.I.T.b algorithm ..................................................................................................... 88

3.3.5 Cepstrum ....................................................................................................................................... 90
3.3.6 The Feje’r- Riesz theorem ......................................................................................................... 92

References ............................................................................................................................................... 103
CONTENTS

4 CHAPTER FOUR  ADJUSTMENT ALGORITHMS OVER THE TELPHONE NETWORK........................................................................................................ 105

4.1 Introduction......................................................................................................................................... 105
4.2 Transmission of digital data over the telephone network: Telephone circuits......................... 105
4.2.1 Attenuation and delay distortions over telephone circuits
(Transmission impairments).................................................................................................................. 109
4.3 Model of Data Transmission system over telephone channels.................................................. 111

4.4 Performance of Adjustment Algorithms over the Telephone Network:
Computer Simulation Tests and Results.................................................................................................. 113
4.4.1 Computer Simulation Tests............................................................................................................ 113
4.4.1.1 QR Method Results.................................................................................................................. 117
4.4.1.2 Newton-Raphson Results...................................................................................................... 117
4.4.1.2a QRN (Newton-Raphson combined with QR) Results......................................................... 118
4.4.1.2b Variant of Newton-Raphson (Truncated input) Results.......................................................... 119
4.4.1.3 NRm (Newton-Raphson -Novel Application) Results............................................................... 120
4.4.1.4 Laguerre method Results........................................................................................................ 120
4.4.1.4a QRL (Newton-Raphson combined with Laguerre’s Method) Results................................. 122
4.4.1.5 Clark-Hau Results.................................................................................................................. 123
4.4.1.6 N.I.T Results.......................................................................................................................... 124
4.4.1.6a Modified N.I.T.b Results........................................................................................................ 125
4.4.1.6b Modified N.I.T.a Results........................................................................................................ 125
4.4.1.7 Gram-Schmidt Results............................................................................................................ 125
4.4.1.8 Bauer Results.......................................................................................................................... 125
4.4.1.9 Levinson Results...................................................................................................................... 126
4.4.1.10 Cepstrum Results.................................................................................................................. 126
4.4.1.11 N.I.T- The Time Varying Case (N.I.T over HF radio links).................................................. 127

4.4.2 Summary of Results..................................................................................................................... 131

References............................................................................................................................................... 231
CONTENTS

CHAPTER FIVE   FILTER IMPLEMENTATION................................................... 233

5.1 Introduction ........................................................................................................... 233
5.2 Fixed point - Floating point Arithmetic .............................................................. 235
5.3 Algorithm Implementation ................................................................................... 235
  5.3.1 Implementation considerations for the root finding algorithms
  using 16-bit fixed point arithmetic ........................................................................... 235
  5.3.2 Cepstrum ........................................................................................................... 236
  5.3.3 Implementation on a TMS320C25 .................................................................. 240
     5.3.3.1 First part of the algorithm FFT block ......................................................... 240
     5.3.3.2 Second part of Cepstrum algorithm ............................................................. 249
  5.3.4 TMS320C30 Implementation ......................................................................... 250
5.4 Computer Simulation Tests and Results ............................................................. 250
  5.4.1 Conclusions ..................................................................................................... 255

References .................................................................................................................. 280

CHAPTER SIX   CONCLUSIONS AND FURTHER WORK.................................... 282

6.1 Conclusions ......................................................................................................... 282
6.2 Suggestions for further work .............................................................................. 287

References .................................................................................................................. 288
CONTENTS

Volume 2

APPENDICES

APPENDIX 2 (A2) .................................................................................................... 289

A2.1 Differential Encoding and the Mapping of the uncoded 16 (or 64) level QAM signal ................................................................. 289

A2.2 Equalisers ................................................................................................. 290

References ........................................................................................................... 299

APPENDIX 3 (A3) .................................................................................................... 301

A3. Clark-Hau equivalence to Newton-Raphson ..................................................... 301
A3.1 The Levinson Recursion ............................................................................... 302
A3.2 Durbin's Algorithm .................................................................................... 309
A3.3 Whittle’s Exponential-Logarithmic Method ................................................ 315
A3.4 Proof of the Fejer-Riesz theorem .................................................................. 317
A3.4.1 Hilbert Transform Relations ................................................................... 320
A3.5 LU Factorisation Decomposition .................................................................. 321
A3.6 Extended Newton’s Method for the complex case ...................................... 322
A3.6.1 Real coefficients-complex roots ............................................................. 323
A3.6.2 Complex coefficients-complex roots ....................................................... 326
A3.7 Schur Decomposition .................................................................................. 328
A3.8 Companion Matrix ...................................................................................... 329

References ........................................................................................................... 330
ACKNOWLEDGEMENTS

I would like to thank Mr. D.W. Hoare, my supervisor, for his help and guidance throughout the course of the research.

I would also like to thank my Director of Research, Dr. M. Woodward for his friendly approach.

The financial support of the Onassis Foundation is gratefully acknowledged.

I would like to express my thanks to Professor C.F.N. Cowan for his comments and suggestions, concerning some very fine points indeed.

My most sincere gratitude and appreciation to the late Professor A.P. Clark, initially my Director of Research, who has been a source of inspiration to me. His immense knowledge and deep understanding generated a wealth of research that laid the foundation for this work.

Above all, I would like to thank my parents for everything.....
LIST OF ILLUSTRATIONS

Tables

4.1 Sampled Impulse Response of Channels 1 to 4 ................................................. 132
4.2 Sampled Impulse Response of Channels 5 to 8 .................................................. 133
4.3 Minimum Phase Sample Impulse of Channels 1 to 4 .......................................... 134
4.4 Minimum Phase Sample Impulse of Channels 5 to 8 ........................................ 135
4.5 Root Positions of the Sampled Impulse Response of Channels 1 and 2 .......... 136
4.6 Root Positions of the Sampled Impulse Response of Channels 3 and 4 .......... 136
4.7 Root Positions of the Sampled Impulse Response of Channels 5 and 6 .......... 136
4.8 Root Positions of the Sampled Impulse Response of Channels 7 and 8 .......... 137
4.9 Sampled Impulse Responses of Truncated Channels 1 to 4 ............................. 137
4.10 Sampled Impulse Responses of Truncated Channels 5 to 8 ........................... 138
4.11 Minimum Phase Sampled Impulse Responses of Truncated Channels 1 to 4 ... 138
4.12 Minimum Phase Sampled Impulse Responses of Truncated Channels 5 to 8 ... 139
4.13 Root Positions of the Sampled Impulse responses of Truncated Channels 1 and 2 ........................................................................................................... 139
4.14 Root Positions of the Sampled Impulse responses of Truncated Channels 3 and 4 ........................................................................................................... 140
4.15 Root Positions of the Sampled Impulse responses of Truncated Channels 5 and 6 ........................................................................................................... 140
4.16 Root Positions of the Sampled Impulse responses of Truncated Channels 7 and 8 ........................................................................................................... 140
5.3.1a Bit- Reversal Algorithm for an 8-Point Radix-2 Decimation-in-time FFT ....... 277
5.3.1b Radix-4 Digit Reversal for N=6 ................................................................... 277
5.3.2 Range of numbers for Cepstrum algorithm when running the Algorithm with full precision ....................................................................................... 278
5.3.3 Range of numbers (after scaling) for Cepstrum algorithm when using 16-bit fixed point arithmetic .............................................................................. 278

A2.1 Differential Encoding of the 16 or 64-Level QAM signal ................................... 295

A4.4.1 U.S.A.E.L Classification of HF Channels ......................................................... 355
A4.4.2 C.C.I.R Classification of HF Channels ............................................................ 355
A4.4.3 Filter Tap values to obtain the required Frequency spreads ....................... 370
A4.4.4 Sampled Impulse Response of the Minimum Phase Transmitter and Receiver Filters sampled at 4800 samples/second ...................................... 375
A4.4.5 Characteristics of the HF channel platform used for computer Simulation Tests ............................................................................................................. 376

Figures

2.2.1 Model of Digital Data Transmission System ..................................................... 47
2.2.2a Amplitude Response of Filter A ................................................................... 48
2.2.2b Frequency Spectrum of z(t) .......................................................................... 48
CONTENTS

2.2.2c  Amplitude Response of filter C ......................................................... 48
2.2.3  Equivalent Baseband Model of Data Transmission System ................. 49
2.2.4  Simplified Baseband Model of Data Transmission System used in the 
       Computer Simulation .............................................................................. 50
2.5.1    Model of Data Transmission System with 'Desired' Impulse Response .... 50

3.1  The QR Method......................................................................................... 95
3.2  One Tap Feedback Filter........................................................................... 96
3.3  Two Tap Feedforward Filter..................................................................... 96
3.4  Adjustment of the Receiver Prefilter......................................................... 97
3.5  Flow Diagram of the Clark-Hau Algorithm............................................... 98
3.6  Leat-error energy filtering........................................................................ 99
3.7  Predetection Filter Implementation employing Gram-Schmidt Orthogonalisation .... 99
3.8  Matrix Y for the Gram-Schmidt Method.................................................. 100
3.9  The N.I.T algorithm.................................................................................. 101
3.10  Cepstrum Algorithm............................................................................... 102

4.4.1.a  Attenuation and Group Delay Characteristics of Telephone Circuit 1 .... 226
4.4.1.b  Group Delay Characteristics............................................................... 226
4.4.2.a  Attenuation and Group Delay Characteristics of Telephone Circuit 2 .... 227
4.4.2.b  Group Delay Characteristics............................................................... 227
4.4.3.a  Attenuation and Group Delay Characteristics of Telephone Circuit 3 .... 228
4.4.3.b  Group Delay Characteristics............................................................... 228
4.4.4.a  Attenuation and Group Delay Characteristics of Telephone Circuit 4 .... 229
4.4.4.b  Group Delay Characteristics............................................................... 229

5.2.1a  Floating-point number.......................................................................... 269
5.2.1b  Fixed-point (integer) number................................................................. 269
5.2.1c  Scaled fixed-point number.................................................................... 269
5.3.1a  Butterfly for decimation-in-time FFT.................................................... 269
5.3.1b  Alternative form of Butterfly............................................................... 270
5.3.2  Statistical model for fixed-point roundoff noise in a decimation-in-time butterfly computation ................................................................. 270
5.3.3  Decimation-in-time FFT algorithm.......................................................... 271
5.3.4  Functional block diagram of a Sequential FFT processor ...................... 272
5.3.5  The cascade FFT processor.................................................................... 272
5.3.6  The parallel Iterative Processor.............................................................. 273
5.3.7  The Array Processor................................................................................ 273
5.3.8  Minimal Processing System with external data RAM and PROM/EPROM. 274
5.3.9  Butterfly for radix-4, Decimation-in-time FFT........................................ 275
5.3.10  A 16-point, radix-4, Decimation-in-time FFT....................................... 276
5.3.12  Input/Output Buffering.......................................................................... 278

A2.1  Block Diagram of the Differential Encoder............................................. 295
A2.2  Signal Constellation for a 16-Level Differentially Encoded QAM signal .... 296
A2.3  Signal Constellation for a 16-Level Differentially Encoded QAM signal .... 297
A2.2.1  Linear Feedforward Transversal Equaliser........................................ 298
A2.2.2  Pure non-linear Equaliser and Detector.............................................. 298
A4.1  Typical attenuation-frequency characteristics........................................ 335
A4.2  Attenuation-frequency characteristics of a poor circuit......................... 335
A4.3 Typical group-delay frequency characteristics................................. 336
A4.4 Group-delay characteristics of a poor circuit........................................ 336
A4.3.1 Attenuation and Group Delay Characteristics of filter set 1................. 337
A4.3.2 Attenuation and Group Delay Characteristics of filter set 2.................. 338
A4.4.1 Single Hop Propagation ................................................................... 351
A4.4.2 Refractive Bending ........................................................................... 351
A4.4.2a Reflection from different Layers....................................................... 352
A4.4.2b Different Number of Hops................................................................. 352
A4.4.3 Typical Response of a Multipath Channel........................................... 353
A4.4.5 Rayleigh Probability Density Function............................................... 354
A4.4.6 Block Diagram of HF Ionospheric Channel Model............................... 367
A4.4.7 Rayleigh Fading introduced by one skywave......................................... 367
A4.4.8 Power Spectrum of q(t) and q(t) ......................................................... 368
A4.4.9 Characteristics of the Signals q(t) ......................................................... 369
A4.4.10 Block Diagram of the filter used for generation of Fading Samples........ 370
A4.4.11 The linear interpolation process ......................................................... 377
A4.4.12 Model of the Digital Data Transmission System............................... 371
A4.4.13 Model of a two Skywave HF Radio link............................................. 372
A4.4.14 Relationship between {Y} and {Z} ...................................................... 373
A4.4.15 Frequency Characteristics of the Practical Equipment and Radio Filters 374
A4.4.16 Timing Relationship between the Transmitter Filter Impulse Response
(real and imaginary) and its delayed version .................................................. 375
A5.1a Number wheel for unsigned Integers ................................................... 388
A5.1b Number wheel for signed Integers ......................................................... 388
A5.1c Number wheel for Fixed-Point Representation ....................................... 388

Graphs

GRO.1 Root Positions of Channel 1......................................................... 141
GRO.2 Root Positions of Channel 2 ......................................................... 142
GRO.3 Root Positions of Channel 3.......................................................... 143
GRO.4 Root Positions of Channel 4.......................................................... 144
GRO.5 Root Positions of Channel 5.......................................................... 145
GRO.6 Root Positions of Channel 6.......................................................... 146
GRO.7 Root Positions of Channel 7.......................................................... 147
GRO.8 Root Positions of Channel 8.......................................................... 148
GNR.1 Performance of Newton-Raphson over Channel 1 ......................... 149
GNR.2 Performance of Newton-Raphson over Channel 2 ......................... 149
GRN.3 Performance of Newton-Raphson over Channel 3 ......................... 149
GRN.4a Performance of Newton-Raphson over Channel 4 ....................... 149
GRN.4b Performance of Newton-Raphson over Channel 4 ....................... 150
GRN.5 Performance of Newton-Raphson over Channel 5 ......................... 150
GRN.6 Performance of Newton-Raphson over Channel 6 ......................... 150
GRN.7 Performance of Newton-Raphson over Channel 7 ......................... 150
GRN.8 Performance of Newton-Raphson over Channel 8 ......................... 151
GL.1a Performance of Laguerre method over Channel 1............................ 152
GL.1b Performance of Laguerre method over Channel 1............................ 152
GL.2 Performance of Laguerre method over Channel 2............................ 152
GL.3 Performance of Laguerre method over Channel 3............................ 152
CONTENTS

GL.4  Performance of Laguerre method over Channel 4 ........................................ 153
GL.5  Performance of Laguerre method over Channel 5 ........................................ 153
GL.6  Performance of Laguerre method over Channel 6 ........................................ 153
GL.7  Performance of Laguerre method over Channel 7 ........................................ 153
GL.8  Performance of Laguerre method over Channel 8 ........................................ 154
G.NT1  Performance of Cepstrum and N.I.T algorithms over HF radio links (HF2) ... 213
G.NT2  Performance of Cepstrum and N.I.T algorithms over HF radio links (HF5) 
       [1st Transmission instant] ................................................................................ 213
G.NT3  ‘Raw’ Impulse Response, Minimum Phase and Pseudominimum Phase
       Impulse Responses ......................................................................................... 214
G.NT4  Performance of Cepstrum and N.I.T algorithms over HF radio links (HF5) 
       [2nd Transmission instant] .............................................................................. 214
G.NT5  Performance of Root Finding (any), Cepstrum and N.I.T algorithms over
       HF radio links (HF4) ..................................................................................... 215
G.NT6  Performance of Root finding Methods (any) over an HF radio link (HF4) ... 215
G.NT7  Performance of Cepstrum and N.I.T Methods over HF radio links (HF4) ... 216
G.NT8  Performance of Root finding Methods (any) over an HF radio link (HF4) ... 216
G.NT9  Performance of Cepstrum and N.I.T Methods over HF radio links (HF4) ... 217
G.1  Performance of Selected Adjustment Techniques over Channel 1 .................. 218
G.2  Performance of Selected Adjustment Techniques over Channel 2 ................. 219
G.3  Performance of Selected Adjustment Algorithms over Channel 3 ................. 220
G.4  Performance of Selected Adjustment Algorithms over Channel 4 ................. 221
G.5  Performance of Selected Adjustment Algorithms over Channel 5 ................. 222
G.6  Performance of Selected Adjustment Algorithms over Channel 6 ................. 223
G.7  Performance of Selected Adjustment Algorithms over Channel 7 ................. 224
G.8  Performance of Selected Adjustment Algorithms over Channel 8 ................. 225

NOTATION

s_i  ==  Data Symbol (Chapter 2) (see p. 13)
m  ==  Number of Levels for the QAM signal (Chapter 2) (see p. 13)
a(t)  ==  Impulse response of a filter (Chapter 2) (see p. 14)
A(f)  ==  Frequency response of a filter (Chapter 2) (see p. 14)
b(t)  ==  Impulse response of a filter (Chapter 2) (see p. 14)
B(f)  ==  Frequency response of bandpass filter (Chapter 2) (see p. 14)
w(t)  ==  White Gaussian noise with zero mean and two-sided power spectral
       density n_0/2 (Chapter 2) (see p. 15)
n_0/2  ==  Power spectral density of w(t)
j  ==  When used as subscript is an integer, otherwise it is \sqrt{-1} (Chapter 2)
       (see p.15)
*  ==  Convolution operator (Chapter 2) (see p. 16), or as a superscript denotes
       Complex Conjugate (Chapter 2) (see p. 19)
r(t)  ==  Received signal (Chapter 2) (see p. 17)
y(t)  ==  Impulse response of linear baseband channel (Chapter 2) (see p. 19)
Y_i  ==  Sampled impulse response of linear baseband channel at time t = iT
       (Chapter 2) (see p. 22)
CONTENTS

\[ s_i \] = Detected Data Symbol (Chapter 2) (see p. 23, 24)
\[ E[.] \] = Expectation operator (Chapter 2) (see p. 23)
\[ e_i \] = Error in the estimated value of the received signal \( r_i \) (Chapter 2)
\[ (\text{see p. 25}) \]
\[ r_i \] = Complex valued sample of \( r(t) \) at time \( t = iT \) (Chapter 2) (see p. 25)
\[ Y_i' \] = Estimate of \( Y_i \) at time \( t = iT \) (Chapter 2) (see p. 25)
\[ \psi_1 \] = Error measurement given by equation 2.7.9 (Chapter 2) (see p. 45)
\[ \psi_2 \] = Error measurement given by equation 2.7.10 (Chapter 2) (see p. 45)
\[ \psi_3 \] = Error measurement given by equation 2.7.11 (Chapter 2) (see p. 45)
\[ \theta_i \] = Error measurement given by equation 2.7.11 (Chapter 2) (see p. 45)
\[ Q \] = Householder Matrix (Chapter 3) (see p. 56)
\[ R \] = Householder Matrix (Chapter 3) (see p. 57)
\[ \|w\|_2 \] = Matrix norm (Chapter 3) (see p. 57)
\[ D(z) \] = \( z \)-transform of linear pre-detection filter (Chapter 3) (see p. 68)
\[ \lambda_i \] = An estimate of the negative reciprocal of a root of \( Y(z) \) that lie
outside the unit circle (Chapter 3) (see p. 71)
\[ \delta \] = Threshold level for determining convergence of an iterative process
(Chapter 3) (see p. 59)
\[ d \] = Threshold level for determining convergence of an iterative process
(Chapter 3) (see p. 73)
\[ \psi \] = Represents both \( \psi_2 \) and \( \psi_3 \) in the NI T method (Chapter 4) (see p. 116)
\[ \varepsilon \] = Threshold level for determining convergence of an iterative process
(Chapter 4) (see p. 117)
\[ \text{eps} \] = Fractional roundoff error in Laguerre method (Chapter 4) (see p. 121)
\[ \text{Re}[.] \] = Real part of a complex number (Chapter 5) (see p. 237)
\[ \text{Im}[.] \] = Imaginary part of a complex number (Chapter 5) (p. 237)
ABSTRACT

The thesis investigates techniques for the adjustment of pre-detection filters employed by high speed serial digital modems operating over both time-invariant and time-varying channels.

Various techniques for the adjustment of these pre-detection filters have been considered in previous investigations. The more recent ones enable such filters to be adjusted in a simple and accurate way using an estimate of the sampled impulse response of the linear baseband channel, together with a prior knowledge of the roots that lie outside the unit circle in the z-plane. A root finding algorithm is an integral part of these techniques.

Algorithms for the location of the required roots are presented here, and compared with previous ones in an attempt to optimise the operational speed and accuracy of the adjustment of these filters.

Alternative algorithms have also been considered, that operate directly on the sampled impulse response of the linear baseband channel, without the need for locating any roots, thus enabling a faster and more accurate adjustment of the pre-detection filter.

The relative performances of all the algorithms are then evaluated over different telephone channels, at transmission rates of 9600 and 19200 bits/sec. The algorithms are also tested over fading channels (such as HF radio links), so as to select the one which offers the best compromise between performance and complexity for hardware implementation.

Finally, various aspects of the hardware implementation of the selected algorithm have been considered.
1.1 Background of the Research

The thesis investigates techniques for the adaptive adjustment of a digital receiver in a digital data-transmission system, where the latter operates in the presence of additive noise and severe intersymbol interference in the received signal, which is the principal reason for the poor performance of higher data rate transmission/reception methods over telephone channels [1,2].

In serial data transmission (considered in this thesis) the data are extracted from the received signal using a sophisticated detection process which overcomes the intersymbol interference problem mentioned earlier. In the detection of serially transmitted digital data, the detector may adopt one of several strategies which are broadly classified into two groups. In the first group, the detector employs a device known as an equaliser to remove the intersymbol interference from the received signal [3-20]. The process of removing the intersymbol interference by the equaliser often results in using only part of the transmitted signal energy in the detection of a data symbol, with a consequent reduction in tolerance to additive noise. Some serial modems use equalisation techniques in the detection process [21-23].

In the second group, the detector makes use of the fact that if the detection process is to have the best tolerance to additive white Gaussian noise, i.e. the detector minimises the probability of an error in the detection of the complete message, then the whole of the received signal must be used by the detection process. This leads to two definitions of the optimum detector [16], the first type minimises the probability of error in the detection of the whole of the received message, and the second type minimises the error probability in the detection of an individual data symbol and hence
minimises the average bit error rate over all received data symbols, provided that they are statistically independent. The optimum detection process is known as the maximum likelihood detection and in many cases its performance is as good as if there were no intersymbol interference [18].

The major drawback of the maximum likelihood detector, compared with the equaliser, is its high complexity, which in most cases prevents its use in practice. A recursive technique known as the Viterbi algorithm [24] greatly simplifies the maximum likelihood detection process. Unfortunately, when the sampled impulse response of the channel contains a large number of components (non-zero samples), which is often the case in practice, the Viterbi algorithm involves both an excessive amount of storage and an excessive number of operations per received data symbol [2]. When a Viterbi algorithm detector is used at the receiver and the data signal is transmitted over the channel at below the Nyquist rate, a 'noise-whitened matched filter' must be used ahead of the detector if the latter is to minimise the probability of error in the detection of the received message [18]. This is also the filter that forms the first part of the non-linear equaliser. In the case of the non-linear equaliser (containing both a linear and non-linear filter), the channel is equalised by the linear filter and partly the non-linear filter, the two filters together achieving the accurate equalisation of the channel [24-27].

If the data symbol rate is increased up to the Nyquist rate, the noise-whitened matched filter degenerates into the linear feedforward transversal filter just described for the non-linear equaliser [28].

A compromise between the equalisers and maximum likelihood detectors is the near maximum likelihood detector which combines the simplicity of the equaliser and the optimality of the maximum likelihood detector. It has been shown, by computer simulation tests that over linear channels that introduce different levels of amplitude distortion (ranging from low to very severe, but with little or no phase distortion), the
Viterbi algorithm can indeed be replaced, with no significant loss, by the simpler detection processes (Chapter 2 - Section 2.5) which implement near maximum likelihood detection [33].

Computer simulation tests on various QAM data transmission systems operating at 9600 bit/s and 19200 bit/s over models of various telephone circuits [28-30,32], have shown that, if the linear feedforward transversal filter (previously described as forming the first part of a non-linear equaliser) is used ahead of the detector, a significant improvement in tolerance to additive white Gaussian noise (typically 1-6 dB) can be achieved by replacing the conventional non-linear equaliser by any one of various near maximum likelihood detection processes [28-30].

The transversal filter is an allpass network with ideally an infinite number of taps, without however changing any amplitude distortion introduced by the channel and without affecting the data signal and noise statistics; which means that the absolute value of the frequency response of the channel and filter is the same as that of the channel itself [28-31]. It has been shown [34], that without the linear transversal filter ahead of the detector, the severe phase distortion introduced by the poorer telephone circuits prevents the satisfactory operation of the near maximum likelihood detectors (even when modified to handle a very small value of the first component of the sampled impulse response of the channel [34]), unless the detectors are further modified into much more complex systems [35].

It has become clear by now that for the most reliable operation of any near-maximum likelihood detector or non-linear equaliser, in a 9600 or 19200 bit/s data-transmission system operating over a telephone circuit, the linear feedforward transversal filter ahead of the detector must be correctly adjusted for the given telephone circuit, to act as an allpass network that makes the resultant response minimum phase [28-32,35].
The transversal filter can, for convenience, be considered to operate in two stages. First, it equalises all phase distortion to give a sampled impulse response that is linear phase and secondly it converts this linear phase impulse response into the corresponding minimum phase response, having the same absolute value of the frequency response [31]. The particular virtue of the operation just described is that it concentrates the energy of the sampled impulse response of the channel and filter towards the earliest samples, without changing the signal/noise ratio at the output of the filter.

Simple iterative processes based on the gradient algorithms and on various developments of this, can be used both to adjust the linear feedforward transversal filter and to estimate the sampled impulse response of the channel and filter [2,25,28,36,37]. When correct convergence of the iterative process has been reached, in the presence of additive white Gaussian noise at high signal/noise ratios, the resulting linear filter approximates reasonably well to the ideal transversal filter of the non-linear equaliser. However when operating over the poorer telephone circuits, a sufficiently accurate adjustment of the filter is not always obtained, even with a more sophisticated algorithm [37]. Furthermore, the inaccurate adjustment of the linear filter can lead to a significant degradation in tolerance to noise, whether a near-maximum likelihood detector or non-linear equaliser is used, and an unduly long training signal may now have to be employed at the start of the transmission, for the initial adjustment of the filter [37].

To alleviate the problems just described, alternative techniques for the adjustment of the prefilter have been developed [38,39]. These techniques make full use of the prior knowledge that the linear filter must be an allpass network such that the sampled impulse response of the channel and filter is minimum phase, and secondly operate solely and directly on the estimate of the sampled impulse response of the channel. This
estimate can be obtained much more quickly and accurately than the corresponding adjustment of the adaptive linear filter by conventional means [2,37,40-43]. The techniques rely on the fact that the linear filter ahead of the detector operates by removing all roots (zeros) of the z-transform of the sampled impulse response of the channel that lie outside the unit circle in the z-plane, and replacing them by the complex conjugates of their reciprocals, while leaving the remaining roots unchanged. The determination of the required roots that lie outside the unit circle, with no restrictions in time or complexity could be achieved by any conventional root finding algorithm. The speed of the present day microprocessors does, however, require that the root finding algorithm be as fast as possible, and at the same time, accurate such that it may be implemented in real time. The techniques mentioned above [38-39] are in this sense suited to real time applications.

However, they occasionally fail especially over the more severely distorted channels. This failure results in a low accuracy in the adjustment of the linear filter which is the single most important reason that limits the highest possible transmission rate over a channel (given of course that highly accurate estimates are readily available) [44]. Note also that although near maximum likelihood detectors have been researched and used widely at L.U.T [28-31,33-35], this is not the norm outside. Perhaps the simplest and most widely used techniques for maximising the transmission rate over any given channel are those of linear and decision-feedback (non-linear) equalisation despite their suboptimum performance when compared with the near maximum likelihood detectors.

Although an occasional failure of an adjustment algorithm to locate say one or more of the required roots (provided that their moduli does not exceed a predetermined small value, say 1.1) does not result in a serious degradation in the performance of a near maximum likelihood detector, this is not the case for the non-linear equaliser. A single failure to locate a root of magnitude, say $a$, results in a degradation of $10\log_{10}(1/a)$ dB in the performance of the equaliser. This could represent an unacceptable loss
especially if the $a$ is a big number, say $a > 1.5$, and more than one root have been missed by the algorithm.

Moreover, improvement as far as the root locating capability and therefore the accuracy as well as the speed of the above methods [38,39] are concerned, is desirable, if the latter were to operate effectively over 'difficult' HF radio links.

Considering the above it is evident that despite the wealth of research on techniques for the adaptive adjustment of the linear filter, there is scope for further improvement, modifications and innovations on the subject.

In this thesis methods described in Ref. [38,39] are re-examined with a view to further improving them. Additional Root finding algorithms have been suggested and investigated together with the already available ones, in the hope of devising a 'global' adjustment strategy.

It is shown here that this is not possible with a root finding algorithm. Alternative techniques, known collectively as Spectral Factorisation Techniques, that operate solely on the estimate of the sample impulse response of the channel, without the need to locate any roots have been considered. Some of these techniques offer no advantage over the root finding algorithms mentioned earlier. However, a particular Spectral Factorisation technique (Cepstrum) seems to offer an improved performance (as far as accuracy and speed of operation are concerned) as well as a global technique for the adjustment of the prefilter as it can now be applied over telephone channels and HF radio links.
Finally, along the lines of spectral factorisation a novel algorithm has been designed (NIT), especially for application over the most severely distorted HF channels. Results for the HF case would justify a separate volume [45]. However, a few preliminary results are presented here (in an attempt to demonstrate the global application of the last two i.e. Cepstrum - NIT techniques) and also published in Ref. 46.

1.2 Outline of the Thesis

The thesis consists of six Chapters. Chapter 2 develops the model of a typical digital communications system. It shows the equivalence between the bandpass and baseband models, where the data is transmitted over a linear baseband channel. The receiver is analysed down to its building blocks, i.e. the estimator, the detector and the feedforward linear adaptive filter which lies ahead of the detector. The maximum likelihood detection process is the optimum detection process and is discussed in some detail together with some near maximum likelihood detectors which implement a simplified version of the maximum likelihood detection process. The alternative, suboptimum but nevertheless widely used equalisers (and in particular non-linear equalisers) are discussed in the appendix of this chapter. Finally, the operation of the adaptive filter is discussed and adjustment criteria are given, against which the various adjustment techniques will be judged for effectiveness i.e. speed of operation, accuracy and simplicity. This chapter really acts as an introduction to the following one which describes the various adjustment algorithms and techniques.

Chapter 3 presents various root finding algorithms and spectral factorisation techniques for the prefilter adjustment. Conventional algorithms are reconsidered and new techniques and algorithms are presented in order to develop a 'global' algorithm
which would successfully operate over time invariant channels, such as telephone
channels (but also at the highest possible data transmission rates) as well as time
varying channels, such as HF radio links (even under the most severe cases of fading),
given the important role that the adaptive prefilter plays as a part of a complete digital
communications system with either a near maximum likelihood detector or a non-linear
equaliser at the receiver end. Two original applications for the two well known root
finding algorithms Newton-Raphson and Laguerre (algorithms 3.2.2.i and 3.2.3.i in
Chapter 3) are presented in the form of algorithms 3.2.2.ii and 3.2.3.ii, respectively.
Some arguments are also presented to theoretically justify the use of the Cepstrum
technique (algorithm 3.3.5) with complex polynomials (since the sampled impulse
response of a digital QAM data transmission system is represented by a complex
polynomial). Finally, a novel algorithm has been designed primarily to operate over
HF radio links but also exhibits desirable properties in the case of telephone channels
thus giving along with the Cepstrum technique two quite powerful techniques for the
adjustment of the linear filter. This Chapter really builds the infrastructure for the rest
of the thesis and together with next Chapter form the backbone of this investigation.

Chapter 4 effectively tests all root finding and spectral factorisation techniques
considered in Chapter 3 (some 14 algorithms altogether), using the criteria laid down
in Chapter 2. The methods are tested over eight telephone channels which represent
varying degrees of distortion ranging from mild to very severe and their performance is
evaluated to determine which is the best candidate for real time implementation.
Considerations such as speed of operation (given here in number of arithmetic
operations) and accuracy are taken into account when deciding which algorithm is best
for inclusion in a real time system.
The real time implementation of the 'best' algorithm is the theme of Chapter 5.
Chapter 5 considers various aspects for implementing the chosen algorithm from the previous chapter with limited precision arithmetic on a TMS fixed point arithmetic digital signal processor (DSP). Speed (and of course, accuracy) of the adjustment algorithm is a critical issue, when the latter operates in real time. This is especially true for time varying environments, such as telephone circuits operating at the higher data transmission rates and even more so HF radio links. Thus, the different blocks of the algorithm are implemented using readily available routines and then assembled together to assess the time taken by the algorithm when it is running with limited precision. The time for the implementation on the TMS DSP, is then compared with the time that was calculated 'theoretically', when the algorithm was simulated with limited precision using Matlab software routines. The 'theoretical' time is in close agreement with the 'real' time. Finally, some issues relating to a real time implementation on a TMS DSP with floating point arithmetic capability are considered briefly; time considerations and accuracy for this implementation have been studied thoroughly in Chapter 4.

REFERENCES


CHAPTER TWO
MODEL OF A DIGITAL DATA TRANSMISSION SYSTEM

2.1 Introduction

This chapter describes a general model of a digital data transmission system, where the transmission of binary data occurs over a voice-band channel. The most suitable type of modulation scheme for voice-band channels is suppressed carrier quadrature amplitude modulation (QAM) and has been used since the late 1960's to achieve reliable transmission of data at rates up to 9600 bit/s or more [1]. The derivation of the baseband model of QAM systems, employing linear coherent demodulation at the receiver, is presented in some detail. Special emphasis is being placed on the estimator, detector and adaptive filter that form the receiver part of this digital modem.

2.2. Model of the Digital Transmission System

The digital data stream in figure 2.2.1 is first encoded to produce two streams of bipolar impulses, $\Sigma s_{0,i} \delta(t-iT)$ and $\Sigma s_{1,i} \delta(t-iT)$, where $1/T$ is the rate at which the impulses are generated. For an $m$-level QAM signal $s_{0,i}$ and $s_{1,i}$ have one of their possible values given by

$$s_{1,0}, s_{1,1} = 2l - \sqrt{m} + 1,$$

2.2.1

$$l = 0, 1, \ldots \sqrt{m} - 1$$
The symbols $s_{i,o}$ and $s_{i,1}$, together, may thus have one of $m$ possible combinations. Thus, when the data transmission system operates at 9600 bit/s (the transmitted signal being a serial stream of 16-level QAM signal-elements with a carrier frequency of 1800 Hz and a signal-element rate of 2400 bauds), the possible values of $s_{i,o}$ and $s_{i,1}$ are $\pm 1$ and $\pm 3$ (Equation 2.2.1). When the bit rate is 19200 bit/s (the transmitted signal being a serial stream of 64-level QAM signal-elements with a carrier frequency of 1800 Hz and a signal-element rate of 2400 bauds), the possible values of $s_{i,o}$ and $s_{i,1}$ are $\pm 1$, $\pm 3$, $\pm 5$ and $\pm 7$. Each of the two stream of impulses are fed separately into one of two lowpass filters at the transmitter, to be shaped to the appropriate bandwidth. These two lowpass filters have identical real valued impulse responses, $a(t)$, with transfer functions $A(f)$. The output signals of these two filters are then modulated (multiplied) by two carriers, in phase quadrature, but with the same carrier frequency $f_c$. The factor $\sqrt{2}$ in $\sqrt{2} \cos(2\pi f_c t)$ and $\sqrt{2} \sin(2\pi f_c t)$ gives each of these signals a mean square value of unity [2,3,4,5].

The outputs of the two linear modulators are added together to form a QAM signal, which is then fed to the bandpass filter $B$ at the transmitter. Filter $B$ has a real valued response $b(t)$ and transfer function $B(f)$. This filter prevents any unwanted signals that lie outside the band of frequencies, occupied by the QAM signal, entering the transmission path. The carrier frequency $f_c$ is chosen such that the amplitude spectrum $Z(f)$ of $z(t)$, which is input to the transmission path will fit within the frequency characteristics of the channel, whose real valued impulse response is $h(t)$ (and its transfer function $H(f)$). The amplitude response of filter $A$ is shown in figure 2.2.2.a and figure 2.2.2.b shows the amplitude spectrum of $z(t)$. It can be seen (figure 2.2.2.b) that for $|Z(f)|$ to have a bandpass shape, the carrier frequency, $f_c$, must not be less than $1/2T$, where $|A(f)|$ is assumed to be bandlimited to $[-1/2T, 1/2T]$ Hz, that is the system is assumed to be operating at Nyquist rate [6]. The only noise signal assumed here,
white Gaussian noise (AWGN), $w$ with zero mean and a two sided power spectral density $n_0/2$. At the receiver, the receiver filter $C$ which has a real valued impulse response $c(t)$ (and transfer function $C(f)$), removes any spectral components of $w(t)$ that lie outside the frequency band of the QAM signal, without however, causing excessive distortion. The amplitude response of filter $C$, $|C(f)|$ is shown in figure 2.2.2.c.

The output signal from the bandpass filter is now coherently demodulated by two reference carriers which have the same frequency but are in phase quadrature. The two lowpass filters after the demodulator suppress the high frequency components generated by the demodulation process, so that only the baseband signals are retained, which are then fed to the detector (Section 2.4 describes the detection process and Section 2.5 describes near maximum likelihood detector structures which in turn implement the maximum likelihood detection process).

These two lowpass filters $D$ are as those in the transmitter with impulse response $d(t)$ and transfer function $D(f)$. It is assumed that the transmitter and receiver lowpass filters are such that:

$$A(f) = B(f) = 0 \quad |f| > f_c$$  \hspace{1cm}  \textit{2.2.2}$$

and that $f_c$ is such that

$$f_c \geq \frac{1}{2T}$$  \hspace{1cm}  \textit{2.2.3}$$

where $1/T$ is the signal-element rate. The complex data symbol is defined as

$$s_i = s_{i,0} + js_{i,1}$$  \hspace{1cm}  \textit{2.2.4}$$
\[ z(t) = \sqrt{2} \sum l s_{l,0} \alpha(t - iT) \cos 2\pi f_c t \]  
\[ - \sqrt{2} \sum l s_{l,1} \alpha(t - iT) \sin 2\pi f_c t \]  

2.2.5

Now, Equation 2.2.5 may be written as

\[ z(t) = \sqrt{2} \Re \left[ \sum I s_i \alpha(t - iT) e^{j2\pi f_c t} \right] \]  

2.2.6

or equivalently

\[ z(t) = \frac{1}{\sqrt{2}} \sum l \left( s_i e^{j2\pi f_c t} + s_i^* e^{-j2\pi f_c t} \right) \alpha(t - iT) \]  

2.2.7

where \( s_i^* e^{-j2\pi f_c t} \) is the complex conjugate of \( s_i e^{j2\pi f_c t} \). The input signal to the coherent demodulator is given by

\[ m(t) = z(t) \ast b(t) \ast h(t) \ast c(t) + w(t) \ast c(t) \]  

2.2.8

where \( \ast \) indicates the convolution operation. In the demodulator, the multiplication process causes the transfer function of \( m(t) \) to be shifted in a negative direction by \( f_c \) Hz. The lowpass filters \( d(t) \), then remove the high frequency components leaving only the baseband components. The signal at the output of the two lowpass filters, \( d(t) \) is given by
\[ r_1(t) = \left( \sqrt{2} m(t) \cos(2\pi f_c t + \theta) \right) \ast d(t) \quad 2.2.9 \]

and

\[ r_2(t) = \left( -\sqrt{2} m(t) \sin(2\pi f_c t + \theta) \right) \ast d(t) \quad 2.2.10 \]

where \( \theta \) is relative phase error between the modulator and demodulator carriers.

It is assumed that the frequency of the two reference carriers is equal to the frequency of the signal carrier, \( f_c \), and that the phase difference between the reference carriers and the signal carrier is zero, i.e. \( \theta = 0 \).

The phase difference may be adjusted to be zero using phase locked loop techniques [7], therefore, it is assumed zero throughout the rest of this analysis.

Combining the two signals \( r_1(t) \) and \( r_2(t) \) in a complex form will yield the complex signal \( r(t) \)

\[ r(t) = r_1(t) + j r_2(t) \quad 2.2.11 \]

then

\[ r(t) = \sqrt{2} (m(t) e^{-j2\pi f_c t}) \ast d(t) \quad 2.2.12 \]

substituting 2.2.7 and 2.2.8 into 2.2.12 yields the following expression for the received signal
\[ r(t) = \sum_i s_i \left( a(t - iT) \ast \left( (b(t) \ast h(t) \ast c(t)) e^{-j2\pi f_c t} \right) \right) \ast d(t) \]
\
\[ + \sum_i s_i' \left( (a(t - iT) e^{-j2\pi f_c t}) \ast \left( (b(t) \ast h(t) \ast c(t)) e^{-j2\pi f_c t} \right) \right) \ast d(t) \quad 2.2.13 \]
\
\[ + \sqrt{2} \left( (u(t) \ast c(t)) e^{-j2\pi f_c t} \right) \ast d(t) \]

The term \( a(t - iT) e^{j\omega t} \) in the second summation in equation 2.2.13 represents a bandpass signal whose frequency lies outside the frequency band of the low pass filter \( D \), therefore the second summation can be ignored. Thus equation 2.2.13 reduces to

\[ r(t) = \sum_i s_i y(t - iT) + w(t) \quad 2.2.14 \]

where

\[ y(t) = \{ a(t) \ast \left( \left( b(t) \ast h(t) \ast c(t) \right) e^{-j2\pi f_c t} \right) \} \ast d(t) \quad 2.2.15 \]

and

\[ w(t) = \sqrt{2} \left( \left( u(t) \ast c(t) \right) e^{-j2\pi f_c t} \right) \ast d(t) \quad 2.2.16 \]

The terms in the square brackets in equation 2.2.15 can be expressed alternatively as

\[ (b(t) e^{-j2\pi f_c t}) \ast (h(t) e^{-j2\pi f_c t}) \ast (c(t) e^{-j2\pi f_c t}) \quad 2.2.17 \]
therefore the Fourier transform of equation 2.2.15 is

\[ Y(f) = A(f)B(f+f_c)H(f+f_c)C(f+f_c)D(f) \]  \hspace{1cm} 2.2.18

Consider \( B(f+f_c) \), since \( b(t) \) is a real valued band pass function then [8]

\[ B(f) = B^*(-f) \]  \hspace{1cm} 2.2.19

Defining \( B_0(f-f_c) \) as

\[ B_0(f-f_c) = \begin{cases} B(f) & f > 0 \\ 0 & f < 0 \end{cases} \]  \hspace{1cm} 2.2.20

then

\[ B_0^*(f-f_c) = \begin{cases} 0 & f > 0 \\ B^*(-f) & f < 0 \end{cases} \]  \hspace{1cm} 2.2.21

thus from equation 2.2.19

\[ B(f) = B_0(f-f_c) + B_0^*(f-f_c) \]  \hspace{1cm} 2.2.22

Taking the inverse Fourier transform of equation 2.2.22 yields [8]
where the function \( b_0(t) \) is said to be the baseband equivalent of \( b(t) \). Multiplying \( b(t) \) in 2.2.23 by \( e^{j2\pi f_c t} \) gives

\[
b(t)e^{j2\pi f_c t} = b_0(t) + b_0^*(t)e^{-j2\pi f_c t}
\]

The second term on the right hand side of equation 2.2.25 has spectral components that lie outside the frequency response of the low pass filters \( A \) and \( D \), therefore, it can be ignored. From equation 2.2.25, and from a similar analysis for \( h(t) \) and \( c(t) \), the impulse response of the channel, \( y(t) \), given by equation 2.2.15 can be written as

\[
y(t) = a(t) * b_0(t) * h_0(t) * c_0(t) * d(t)
\]

which represents the linear baseband channel formed by the two transmitter filters \( A \), the linear modulators, filter \( B \), the bandpass transmission path, filter \( C \), the linear demodulators and the two receiver filters \( D \). Equation 2.2.25 is in general a complex valued quantity.

The Fourier transform of 2.2.25 is

\[
Y(f) = A(f)B_0(f)H_0(f)C_0(f)D(f)
\]

where \( B_0(f) \), \( H_0(f) \), and \( C_0(f) \) are the baseband equivalent transfer functions of filter \( B \) the transmission path and filter \( D \), respectively. Let
represent the overall filtering carried out at the transmitter, and

\[ R_x(f) = C_0(f)D(f) \]  \hspace{1cm} (2.2.28)

represent the overall filtering carried out at the receiver, then (2.2.26) becomes

\[ Y(f) = T_x(f)H_0(f)R_x(f) \]  \hspace{1cm} (2.2.29)

which is the transfer function of the linear baseband channel. The transfer functions \( T_x(f) \) and \( R_x(f) \) are adjusted so that \( |T_x(f)| = |R_x(f)| \) such that if the transmission path introduced no distortion then the receiver filter would be matched to the transmitter filter and the signal-to-noise ratio at the output of filter \( R_x \) would be maximised [6,8].

It may be shown that if \( |R_x(f)| \) is an even function, the real and imaginary part of \( w(t) \) in equation 2.2.16, must be statistically independent Gaussian random variable with zero mean and fixed variance [3,5]. Furthermore, the real and imaginary part of \( w(t) \) are uncorrelated and statistically orthogonal Gaussian random variables. However, at higher transmission rates, the receiver filter will in practice cause the output noise sample to be correlated to a greater or lesser extent. This affects the tolerance of the detection process to noise which can be neglected by increasing the bandwidth of the receiver filter.

A model of the linear baseband channel represented by equation 2.2.29 is shown in figure 2.2.3. The continuous baseband signal \( r(t) \) is then sampled at the baud rate (once every \( T \) seconds) to give the received sample values \( \{r_i\} \), where
and $y_h=(hT)$, $r_i=r(iT)$, $w_i=w(iT)$ and $y_h$ is the $(h+1)^{th}$ component of the sampled impulse response of the baseband channel when sampled at the baud rate.

To ensure that the information carried by the received waveform $r(t)$ is contained also in the sample $\{r_i\}$, the sampling rate must be close to the Nyquist rate \[9,10,11\]. It is assumed that there is no delay in transmission, thus $y_0 \neq 0$. Also $y_h=0$ for $h<0$ and $h>g$, where $g$ is a positive integer. Since the amplitude response of the baseband channel, $|Y(f)|$, is approximately zero for $|f|>1/2T$, the sampling rate of the sampler approximately satisfies the Nyquist sampling theorem, and the samples $\{r_i\}$ contain all the necessary information needed by the detection process.

Thus, the sampled impulse response of the linear baseband channel in figure 2.2.3 is given by the $(g+1)$ component row vector

$$Y=[y_0 \ y_1 \ y_2 \ ... \ y_g]$$ \[2.2.31\]

The vector $Y$ is a measure of the signal distortion introduced by the channel. For a distortionless channel the sampled impulse response of an ideal channel is given by

$$Y=[1 \ 0 \ 0 \ ... \ 0]$$ \[2.2.32\]

and the received sample $\{r_i\}$ at the output of the receiver filter will therefore given by
The signal processor and detector in figure 2.2.3 uses the received signal samples, \{r_i\}, together with the sampled impulse response of the linear baseband channel (or its estimate) to give the detected data symbol \( s'_i \). The signal detector, of course has prior knowledge of the possible values of data symbols \{s_i\}.

From equation 2.3.33, a simpler model of the system can be derived, suitable for use in computer simulation. The simplified model is shown in figure 2.2.4, where the noise components in \( r(t) \) are generated separately by feeding the complex valued noise signal \( u(t) \) into a separate filter identical to the receiver filter \( R_x \).

Using equation 2.2.14, the average energy per signal element in \( r(t) \) is

\[
\varepsilon_s = E \left[ |S_i|^2 \int_{-\infty}^{\infty} |y(t)|^2 \, dt \right]
\]

where \( E[.] \) is the expected value. If the \( \{s_i\} \) are statistically independent and with zero mean

\[
\varepsilon_s = \overline{s_i^2} \int_{-\infty}^{\infty} |y(t)|^2 \, dt
\]

where \( \overline{s_i^2} \) is given by

\[
\overline{s_i^2} = E[|s_i|^2]
\]
The impulse response of the channel \( y(t) \) is scaled such that the integral in equation 2.2.35 has a value equal to unity, therefore

\[
\mathcal{E}_s = \overline{s_i^2} \tag{2.2.37}
\]

The representation of the QAM system by a linear channel model, as shown in figure 2.2.4, gives a simplified view of the data transmission system. Because of its simplicity, the linear baseband model is very useful for computer simulation tests that measure the performance of the system under different conditions.

2.3 Linear feedforward estimator [8,12]

The linear feedforward estimator is the simplest of all channel estimators and is implemented as a linear feedforward transversal filter [8] (Figure 2.3.1). The filter is adaptive in nature, and it was first proposed [12] for use with a maximum-likelihood detector employing the Viterbi algorithm. It makes use of the received samples \( r_i \) and the detected data \( s_i \) in such a way as to minimise the mean square error between the received samples \( \{r_i\} \) and the corresponding estimates of the received samples \( \{r_i'\} \).

The estimator has \( g+1 \) taps which is equal to the number of components in the sampled impulse response of the channel. It is assumed that the data symbols \( \{s_i'\} \) are detected correctly and therefore, \( s_i' = s_i \) for all \( i \). Each of the squares marked T in figure 2.3.1 represent a store that introduces a delay of one sampling period equal to \( T \) seconds. Each store holds the corresponding detected data-symbol \( s_{i-h} \) (Equation 2.2.30 gives the transmitted data-symbol \( s_{i-h} \)). Every time that a new sample \( r_i \) is received, the stores are triggered and the stored values are shifted one place to the right. Following
the detection of $s_i$ the detected data-symbols $s_i', s_{i-h}', ..., s_{i-g}'$ are held in the estimator. Each of the stored values, $\{s_{i-b}\}$, is then multiplied by the corresponding components $\{y_{i+1,b}\}$, where $y_{i+1,b}'$ is an estimate of the components $y_{i,b}$ of the sample impulse response at time $t=(i-1)T$ to give $r_i$.

$$r_i' = \sum_{h=0}^{g} s_{i-h}' y_{i-1,h}'$$  \hspace{1cm} 2.3.1$$

where $y_{i,b} = y_{i,b}(hT)$ and $y_{i,b} = 0$ for $h < 0$ and $h > g$ for practical purposes [see also Equations A4.5.2-10 and Figure A4.4.14 in Appendix 4 - Section A4.5.2].

Next, $r_i'$ is compared with the actual received sample $r_i$ to give an error $e_i$.

$$e_i = r_i - r_i'$$  \hspace{1cm} 2.3.2$$

The error $e_i$ is then multiplied by a small positive quantity $\Delta$. The resulting signal $\Delta e_i$ is multiplied by $(s_{i-h})^*$, complex conjugate of $s_{i-h}$ and the products are added to the corresponding components of $Y_{i+1}$ to give the new stored estimate of the impulse response $Y_i'$. Thus the $(h+1)^{th}$ component of $Y_i'$ is given by

$$y_i' = y_{i-1,h}' + \Delta e_i (s_{i-h}')^*$$  \hspace{1cm} 2.3.3$$

for $h=0,1,...,g$. Equation 2.3.3 is usually known as the stochastic gradient algorithm. The smaller the value of $\Delta$, the smaller the effects of additive noise on $Y_i'$, but the slower the rate of response of $Y_i'$ to changes in $Y_i$. Initially, the vector $Y_i'$ can be set to zero, and a sequence of data symbols known to the receiver, transmitted such that the estimator converges to an accurate estimate of the sampled impulse response before any actual data is sent.
2.4 Maximum Likelihood Detection

In high speed digital transmission systems, where a signal is transmitted over a channel that introduces severe distortion, such as telephone channels, HF links, or even land mobile radio links, a simple threshold detector is usually ineffective as a means of detecting the data from the received signal. Therefore, more sophisticated techniques for detecting the signal in the presence of interference and noise have been developed.

Detection processes for distorted digital signals may be classified into two separate groups. In the first of these, the received sampled digital signal is fed through an equaliser that corrects the distortion introduced by the channel and restores the received signal into a copy of the transmitted signal, neglecting for the moment the effects of the noise. The resultant received signal is then detected in the conventional manner, as normally applied to a serial digital signal in the absence of intersymbol interference. In other words, the equaliser acts as the inverse of the channel, so that the channel and equaliser together introduce no signal distortion, and each data symbol is detected as it arrives, independently of the others, by comparing the corresponding sample value with the appropriate threshold level [3].

There are three main types of equaliser, known as, the linear equaliser, the pure non-linear equaliser and the decision feedback equaliser (DFE). The DFE is, in fact, a combination of the linear and the pure non-linear equaliser and can sometimes exhibit a much better tolerance, to additive noise, than linear equalisers[6,9] (See also Appendix 2 - Section A2.2).
The main weakness of channel equalisation employing the optimum non-linear equaliser is that only a portion of the received signal element is used in the detection of that element, the remaining part of that element being removed by cancellation and not involved in the detection process itself.

The detection process that has the best tolerance to additive white Gaussian noise in the sense that minimises the probability of error in the detection of the complete message, must involve at least the whole of the received signal in a detection process for the complete message. [9,13,14,15,16].

This is the case, in the second group of detection processes, where the decision process itself is modified to take account of the signal distortion that has been introduced by the channel, and no attempt need, in fact, be made to reduce the signal distortion prior to the actual decision process.

This group includes maximum-likelihood detectors, which for our purposes implement the optimum detection process. Assuming that the white Gaussian noise samples at the maximum-likelihood detector input are statistically independent, and bearing in mind that the entire transmitted energy is used in the detection process the maximum-likelihood detector selects as the detected message the possible sequence of the transmitted data symbols for which there is a minimum mean square difference between the samples of the corresponding received data signal, for the given distortion but in the absence of noise, and the samples of the signal actually received. When the transmitted data symbols are statistically independent and equally likely to have any of their possible values, the maximum-likelihood detector minimises the probability of error in the detection of the received message, and in this sense offers the best
performance possible [6,9]. If a sequence of data symbols \( \{s_i\} \) are transmitted, as shown in Figure 2.2, then the received samples at time \( t=iT \), will be given by

\[
\mathbf{r}_i = \mathbf{v}_i + \mathbf{w}_i
\]

where

\[
\mathbf{v}_i = \sum_{h=0}^{g} s_{i-h} y_h
\]

It is assumed that transmission starts at time \( t=iT \) and \( s_i=0 \) for \( i \leq 0 \). It is also assumed that \( \{w_i\} \) are statistically independent Gaussian random variables with zero mean and variance \( \sigma^2 \), that represent the noise components in the received samples \( \{r_i\} \). The quantities \( \{r_i\}, \{v_i\}, \{w_i\}, \) and \( \{y_i\} \) are, in general, complex valued quantities. Let the total number of transmitted symbols be \( N \), i.e. \( i=1,2,...,N \), then the samples can be represented by the \( N \)-component row vectors \( \mathbf{R}_N, \mathbf{V}_N, \mathbf{W}_N, \) and \( \mathbf{S}_N \), where

\[
\mathbf{R}_N = \begin{bmatrix} r_1 & r_2 & \cdots & r_N \end{bmatrix}
\]

\[
\mathbf{V}_N = \begin{bmatrix} v_1 & v_2 & \cdots & v_N \end{bmatrix}
\]

\[
\mathbf{W}_N = \begin{bmatrix} w_1 & w_2 & \cdots & w_N \end{bmatrix}
\]

\[
\mathbf{S}_N = \begin{bmatrix} s_1 & s_2 & \cdots & s_N \end{bmatrix}
\]
From Equations 2.4.1-2.4.6

\[ R_N = V_N + W_N \]  

2.4.7

The detector is assumed to have prior knowledge of the m possible vectors \( \{S_N\} \), and it also has prior knowledge of their a priori probabilities \( \{P(S_N)\} \). The whole of this prior knowledge is used to optimise the detection process, that is, to minimise the probability of error in the detection of the element value.

In order to minimise the probability of error in a detection process [9] it is necessary to maximise the probability of a correct decision, \( P(C) \)

\[ P(C) = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} P(C \mid R_N) p(r_1, r_2, \ldots, r_N) \, dr_1 \ldots dr_N \]  

2.4.8

where \( P(C \mid R_N) \) is the conditional probability of a correct decision given the received vector \( R_N \) and \( p(r_1, r_2, \ldots, r_n) \) is the value of the joint probability density function of the \( n \) random variables, corresponding to the \( n \) components of \( R \), at the given value of \( R \).

The vector \( R \) may have any of its possible values and is not confined to the particular value received. Equation 2.4.8 can be written more simply as

\[ P(C) = \int P(C \mid R) p(R) \, dR \]  

2.4.9

Since \( P(R_N) \) is non-negative, it can be seen that \( P(C) \) is maximised by maximising \( P(C \mid R_N) \) for every possible value of the vector \( R_N \). For any given received vector \( R \), \( P(C \mid R_N) \) is maximised by selecting as the detected vector \( S_N \), the possible vector \( S_N \), for
which the value \((P(S_N/R_N))\) is maximum, where \(P(S_N/R_N)\) is the conditional probability of \(S_N\) given \(R_N\) and is thus a posteriori probability of \(S_N\). The optimum detection process is now reduced to the maximum a posteriori probability (MAP) detection process [17].

Now, since there is a one to one mapping between \(S_N\) and \(V_N\) (Equation 2.4.2) then

\[
P(S_N | R_N) = P(V_N | R_N) \tag{2.4.10}
\]

Moreover, by Bayes’ theorem [9]

\[
P(V_N | R_N) = \frac{P(V_N)}{p(R_N)} p(R_N | V_N) \tag{2.4.11}
\]

where \(P(V_N)\) is the a priori probability of \(V_N\), \(p(R_N)\) is the probability density of \(R_N\), and \(p(R_N | V_N)\) is the conditional probability density function of \(R_N\) given \(V_N\). For a given received vector \(R_N\), \(p(R_N)\) is independent of \(V_N\).

Therefore, from Equations 2.4.10 and 2.4.11, maximising \(P(S_N|R_N)\) is equivalent to maximising the product \(P(V_N)p(R_N|V_N)\). When all of possible values of \(s_i\) are equally likely to occur and hence all \(m^N\) values of \(S_N\) are equally likely to be transmitted

\[
P(S_N) = P(V_N) = m^{-N} \tag{2.4.12}
\]

and the optimum detection process now reduces to the one that selects as the detected vector, the possible \(S_N\) for which the corresponding value of \(p(R_N|V_N)\) is maximised.
process. Since \( p(R_n|V_N) \) is the conditional joint probability density function of the random variable with sample values \( r_1, r_2, ..., r_N \) (Equation 2.4.3) given the values \( v_1, v_2, ..., v_N \) (Equation 2.4.4) it can be written

\[
p(R_K | V_K) = p(r_1, r_2, ..., r_N | v_1, v_2, ..., v_N) \tag{2.4.13}
\]

Let the complex values \( \{r_i\} \) and \( \{v_i\} \) be

\[
r_i = r_{i,1} + j r_{i,2} \tag{2.4.14}
\]

for \( i = 1, 2, ..., N \) and \( j = \sqrt{-1} \). Since \( \{w_i\} \) are statistically independent Gaussian random variables with zero mean and variance \( \sigma_w^2 \), then for a given transmitted vector \( S_N \) and hence for a given vector \( V_N \), the received samples, \( \{r_i\} \), are statistically independent random variables with mean \( v_i \) and variance \( \sigma_w^2 \), thus [17]

\[
p(R_N | V_N) = p(r_{i,1} | v_{i,1}) p(r_{i,2} | v_{i,2}) \cdots \]

\[
\cdots p(r_{N,1} | v_{N,1}) p(r_{N,2} | v_{N,2}) \tag{2.4.16}
\]

so for each \( r_{i,h} \)

\[
p(r_{i,h} | v_{i,h}) = \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(\frac{-(r_{i,h} - v_{i,h})^2}{2\sigma_w^2}\right) \tag{2.4.17}
\]

for \( h = 1, 2, ..., N \). Substituting 2.4.17 into 2.4.16 gives
\[
p(R_N | V_N) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( \frac{-(r_{i,1} - v_{i,1})^2}{2\sigma_w^2} \right)
\]
\[
\times \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( \frac{-(r_{i,2} - v_{i,2})^2}{2\sigma_w^2} \right)
\]
\[
= \frac{1}{(2\pi\sigma^2)^N} \exp \left( \frac{-1}{2\sigma_w^2} \sum_{i=1}^{N} (r_{i,1} - v_{i,1})^2 + (r_{i,2} - v_{i,2})^2 \right)
\]
\[
= \frac{1}{(2\pi\sigma^2)^N} \exp \left( \frac{-(R_N - V_N)^2}{2\sigma_w^2} \right)
\]

where

\[
|R_N - V_N|^2 = \sum_{i=1}^{N} |r_i - v_i|^2
\]
\[
= \sum_{i=1}^{N} \left( (r_{i,1} - v_{i,1})^2 + (r_{i,2} - v_{i,2})^2 \right)
\]

and \(|R_N - V_N|\) is the length of the vector \((R_N - V_N)\) and so it is the unitary distance in the \(N\)-dimensional vector space between the vectors \(R_N\) and \(V_N\) [9]. It can be seen from equation 2.4.19 that the value of the likelihood function \(p(R_N | V_N)\) is maximum when the unitary distance, \(|R_N - V_N|\) is minimum. Thus, for the conditions assumed, the maximum likelihood detector selects as the detected vector the possible vector \(S_N\) for which the
Thus, for the conditions assumed, the maximum likelihood detector selects as the detected vector the possible vector $S_N$ for which the corresponding signal vector $V_N$ is closest (in terms of the unitary distance) to the received vector $R_N$. The computation of the $m^N$ a-posteriori probabilities suggested by equation 2.4.11 can therefore be replaced by the evaluation of the $m^N$ possible values of $|R_N - V_N|^2$ given by equation 2.4.19, which are known as 'costs'. The vector $V_N$ giving the vector $|R_N - V_N|$ with the smallest cost will be the selected vector.

The above process does not minimise the probability of error in the detection of individual data-symbols, but at high signal-to-noise ratios and with white Gaussian noise, the described process will approximately minimise the average probability of an error in the detection of an individual symbol [3]. Since $m$ and $N$ will, in general, be large the maximum likelihood detection process described above will be far too complex to implement since it requires the storage of $m^N$ possible data-symbol vectors $\{S_N\}$ and the same number of unitary distance measurements. Fortunately, the maximum likelihood detection process can be realised by the Viterbi algorithm which not only saves storage but also a very large number of unnecessary calculations [18].

2.4.1 The Viterbi Algorithm Detection Process [19]

The Viterbi algorithm is a recursive method of obtaining the maximum likelihood estimate of the transmitted sequence of data symbols, $S_N$, without the need for computing all of the $m^N$ costs given by equation 2.4.19. Its operation will now be described.

Let $X_N$ be a possible value of the transmitted sequence of data symbols given by the components of the vector $S_N$ and let the vector $X_i$ (known as survivor) be an initial
segment of the vector $X_N$. Also let the $g$-component vector $Z_i$ be a vector whose components are equal to the last $g$ components of $X_i$, i.e.

$$X_N = \begin{bmatrix} x_1 & x_2 & \ldots & x_N \end{bmatrix} \quad 2.4.1.1$$

and

$$X_1 = \begin{bmatrix} x_1 & x_2 & \ldots & x_1 \end{bmatrix} \quad 2.4.1.2$$

and

$$Z_1 = \begin{bmatrix} x_{1-g+1} & x_{1-g+2} & \ldots & x_1 \end{bmatrix} \quad 2.4.1.3$$

Now, for any time, say $t=iT$, the detector can only hold in memory $m^f$ vectors (survivors) $\{X_i\}$. Associated with each of these stored vectors is a cost given by

$$|R_i - X_i|^2 = \sum_{j=1}^{i} |r_j - x_j|^2 \quad 2.4.1.4$$

$$= \sum_{j=1}^{i} \left( (r_{j,1} - x_{j,1})^2 + (r_{j,2} - x_{j,2})^2 \right)$$

where the vector $R_i$ is the $i$ component vector
\[ R_1 = [r_1, r_2, \ldots, r_l] \]

The \( m^g \) stored vectors have the \( m^g \) smallest possible costs associated with them. For all of the survivors their last \( g \) components which form the vectors \( \{Z_i\} \) take on one of the \( m^g \) possible combinations of data symbols.

The detector does not know which of the vectors \( Z_i \) will form part of the maximum likelihood vector \( X_N \) giving the smallest cost, but it definitely knows that \( Z_i \) can only have \( m^g \) different values. Therefore, for any value of \( i \), it can be decided that there are only \( m^g \) possible vectors \( \{X_i\} \) which may be candidates to form part of the vector \( X_N \) giving the smallest cost.

Now, at time \( t=(i+1)T \), the detector, in turn, takes each of the vectors \( \{X_i\} \) and adds on an extra component, \( x_i+1 \), to the end, where \( x_i+1 \) takes on all \( m \) possible values of \( s_i \). Thus from each stored vector \( X_i \), \( m \) expanded vectors \( \{X_i+1\} \) are created. Expanding all \( m^g \) stored vectors will give \( m^{g+1} \) expanded vectors \( \{X_i+1\} \) and their associated costs. But the detector can only hold in store \( m^g \) vectors, and so it selects, from the \( m^{g+1} \) expanded vectors, the \( m^g \) vectors whose last \( g \) components take on all \( m^g \) possible combinations of component values and have the smallest costs. This recursive process continues until \( t=NT \), when the detector has in store the \( m^g \) vectors \( \{X_N\} \) along with their associated costs. It then selects from these vectors the vector with the smallest cost associated with it and gives this as the maximum likelihood estimate of the transmitted sequence \( S_N \) and the detection process will have been completed.
2.5 Near Maximum-Likelihood Detectors

The Viterbi algorithm becomes impractical for channels with long sampled impulse responses. The reason for this is that for each detection instant the Viterbi algorithm computes $m^* + 1$ costs (Equation 2.4.18), so for a complete message of $N$ data symbols, it computes $Nm^* + 1$ costs. A considerable amount of storage is required by the detector (which increases linearly with time), because as $i$ increases the lengths of the survivors $\{X_i\}$ (Equation 2.4.2), will increase.

Any practical derivatives must therefore employ a much smaller number of vectors say $k << m^*$. Clark and colleagues [7,20-25] have described many practical detection schemes.

These simplified detectors, known as near maximum likelihood detectors, drastically reduce the equipment complexity required by the Viterbi algorithm. They utilise the fact that a large number of the stored vectors described in Section 2.4, do not play an important role in the selection process and can be ignored. The following detectors to be described have in store $k$ vectors ($k << g$).

2.5.1. Reduced State Viterbi Algorithm Detector [3,5,21,24]

The data transmission system is as shown in Figure 2.2.3. Just prior to the receipt of the sample $r_i$ at time $t = iT$, the detector holds in store $k$-component vectors $\{Q_i\}$, where

$$Q_{t-1} = [x_{t-n}, x_{t-n+1} \ldots x_{t-1}]$$
and $x_{i-h}$, for $h=1,2,...,n$ takes on one of the possible values of $s_{i-h}$, the $k$ vectors \{$Q_{i-h}$\} being all different. Each vector $Q_{i-1}$ represents a possible sequence of values of the data symbols $s_{i-n}$, $s_{i-n+1}$, ..., $s_{i-1}$. On the receipt of the signal $r_i$ at $t=iT$, each vector is expanded (by adding an extra component to the end of each vector) into $mk(n+1)$-component vectors \{$P_i$\}, where

$$P_i = [x_{i-n} \ x_{i-n+1} ... x_i]$$  \hspace{1cm} (2.5.2)

and $x_i$ has one of the $m$ possible values of $s_i$. In each group of the $m$ vectors \{$P_i$\}, derived from any one vector $Q_{i-1}$, the first $n$ components are as in the original $Q_{i-1}$ and the last component $x_i$ takes on one of the $m$ possible values of $s_i$. Each of the resulting vectors \{$P_i$\} has the cost

$$|C_i|^2 = |C_{i-1}|^2 + \left| r_i - \sum_{h=0}^{n} x_{i-h}Y_h \right|^2$$  \hspace{1cm} (2.5.3)

where $|C_{i-1}|^2$ is the cost of the vector $Q_{i-1}$ from which $P_i$ was derived, such that

$$|C_{i-1}|^2 = \sum_{j=0}^{n} \left| r_j - \sum_{h=0}^{n} x_{j-h}Y_h \right|^2$$  \hspace{1cm} (2.5.4)

and $x_i = 0$ for $i < 0$ and $i > m-1$. The detected $s_{i-h}$ of the data symbol $s_{i-h}$ is now taken as the value of $x_{i-h}$ in the vector $P_i$ with the smallest cost. Any vector $P_i$ whose first component $x_{i-h}$ differs from $s_{i-h}$ is then discarded, and from the remaining vectors \{$P_i$\}
(including that from which $s_{\text{th}}$ was detected) are selected the $k$ vectors having the smallest costs $\{C_i\}$. The first component of each of the $k$ selected vectors $\{P_i\}$ is now omitted (without changing its cost) to give the corresponding vectors $\{Q_i\}$, which are then stored, together with the associated costs ready for the next detection process. The discarding of the vectors $\{P_i\}$ just mentioned is a convenient method of ensuring that the $k$ stored vectors $\{Q_i\}$ are always different, provided only that they were different at the first detection process which can be easily arranged [20].

The main weakness of the detection process is that it involves $k$ searches through $mk$ costs. When $m$ is large this may require an excessive number of operations. A promising technique for reducing the number of operations under these conditions is that employed in pseudobinary and pseudoquaternary systems [21,23,25].

The detector here considers only two or four of the expanded vectors $\{P_i\}$, originating from any one $Q_{i-1}$ that have the smallest costs, the remaining vectors not being used. This is achieved by means of a simple threshold-level comparison technique that not itself involve the evaluation of any costs [21,23,25]. The complexity of the process is therefore reduced to that for the corresponding binary or quaternary signal.

2.5.2. Simple Near Maximum Likelihood Detector [5]

With this system, which is a development of the reduced state Viterbi algorithm detector previously described, a further significant reduction in the number of operations is achieved. Two different versions of the simple near maximum likelihood detector, the pseudobinary and the pseudoquaternary arrangements, have been studied and the pseudobinary version will be considered first.
Just prior to the receipt of the signal \( r_t \) the detector holds in store \( k \) vectors \( \{Q_{t-1}\} \) together with their costs \( \{C_{t-1}\} \) as before. However, \( k \) must be even and the vectors \( \{Q_{t-1}\} \) are arranged in pairs, where the two vectors \( \{Q_{t-1}\} \) in any one pair differ only in the values of the last component \( x_{i-1} \). For the given values of \( x_{i-1}, x_{i-1+1}, \ldots, x_{i-2} \) in a pair of vectors \( \{Q_{t-1}\} \), the value of \( x_{i-1} \) in the first of the pair is the one of its possible values giving the smallest cost \( C_{i-1} \), and the value of \( x_{i-1} \) in the second of the pair is the one of its possible values giving the second smallest cost. On the receipt of \( r_t \), each vector \( Q_{t-1} \) is expanded into the corresponding vector \( P_t \) having the smallest cost. The selection of \( P_t \) is achieved through the use of simple threshold-level comparisons and not involve the computation of any costs [23,25]. The detector next evaluates the costs \( \{|C_i|^2\} \) of the \( k \) vectors \( \{P_t\} \) and selects the vector with the smallest cost, taking the value of its component \( x_{i-n} \) as the detected value \( s_{i-n}^* \) of the data symbol \( s_{i-n} \). All vectors \( \{P_t\} \) for which \( x_{i-n} \neq s_{i-n}^* \) are now discarded, and the first components of all remaining vectors \( \{P_t\} \) are omitted (without changing their costs), to give the corresponding \( n \)-component vectors \( \{Q_t\} \).

In addition to the vector \( Q_t \) with the smallest cost, which has already been selected, the detector now selects, from the remaining vectors \( \{Q_t\} \), the \( 0.5k-1 \) vectors with smallest costs, to give a total of \( 0.5k \) selected vectors \( \{Q_t\} \). To each of these vectors is then added an additional vector \( Q_i \), whose first \( n-1 \) components are as in the original vector \( Q_i \) and whose last component \( x_i \) takes on its possible value for which the cost of \( Q_i \) has its second smallest value. The value of \( x_i \) giving the smallest cost is, of course, that in the original vector \( Q_i \). The simple algorithm previously mentioned [23,25] is employed here, so that no actual costs need be evaluated. The detector now holds in store \( k \) vectors \( \{Q_t\} \), in the form of \( 0.5k \) pairs of vectors, the two vectors of any pair having the same values of \( x_{i-n+1}, x_{i-n+2}, \ldots, x_{i-1} \).
The costs \(|C_i|^2\) of the 0.5k vectors \(Q_i\) not yet determined are next evaluated, and associated with each vector \(Q_i\) is then stored the corresponding cost.

The pseudoquaternary version of system of the simple near maximum likelihood detector, is basically the same to the pseudobinary version.

Just prior to the receipt of \(r_i\), the detector holds in store \(k\) vectors \(Q_{i1}\) together with their costs \(|C_{i1}|^2\), as before. However \(k\) is now a multiple of four and the vectors \(Q_{i1}\) are arranged in groups of four, where the four vectors in any one of group have the same values of \(x_{i0}, x_{i+n+1}, \ldots, x_{i+2}\) and four different values of \(x_{i+1}\). The latter four values are determined by simple threshold-level comparisons and without requiring the evaluation of any costs [21].

On the receipt of \(r_i\) the detection process proceeds exactly as for the pseudobinary version of the simple near maximum likelihood detector, until the detector has \(k\) vectors \(Q_i\) together with the associated costs \(C_i\), every vector \(Q_i\) having been derived from the corresponding vector \(P_i\) by omitting the first component. In addition to the vector \(Q_i\) with the smallest cost, which has been already selected, the detector now selects from the remaining vectors \(Q_i\) the 0.25k-1 vectors with the smallest costs, to give a total of 0.25k selected vectors \(Q_i\). Each of these vectors is then expanded into four vectors \(Q_i\), through the addition of three vectors to the original. The four vectors in any one group have the same values of \(x_{i+n+1}, x_{i+n+2}, \ldots, x_{i+1}\) as the original \(Q_i\) and four different values of \(x_i\). The latter four values generally give the four smallest costs for the corresponding \(Q_i\) and are determined in the same way as the four values of \(x_{i+1}\) in any one group of four vectors \(Q_i\) at the start of the detection process as before, using simple threshold level comparisons and not requiring
the evaluation of any costs [21]. The costs \( |C_i|^2 \) of the 0.25k vectors \( \{Q_i\} \) not yet determined are now evaluated, and the detector then stores k vectors \( \{Q_i\} \) together with the associated costs, ready for the next detection process. The particular virtue of this system is that, for a given number of stored vectors, it involves fewer operations per detection process than does the corresponding arrangement of the reduced-state Viterbi algorithm detector. Thus, with k stored vectors, the pseudobinary version of the simple near maximum likelihood detector evaluates 1.5k costs and carries out 0.5k searches through k costs. On the other hand, the pseudobinary version of the reduced state Viterbi algorithm detector evaluates 2k costs and carries out k searches through 2k costs, per detection process, and the pseudoquaternary version evaluates 4k costs and carries out k searches through 4k costs.

2.6 Adaptive linear feedforward transversal filter

Equation 2.2.30 can be rewritten as

\[
r_i = s_iy_0 + \sum_{h=1}^{\xi} s_{i-h}y_h + w_i
\]

where \( s_iy_0 \) is the wanted component and \( \sum s_{i-h}y_h \) is the intersymbol interference. Now, the selection of the k maximum likelihood estimates of the transmitted sequence, involves the computation of the costs of each of the expanded vectors \( \{P_i\} \), using

\[
C_i = C_{i-1} + \left| r_i - \sum_{h=0}^{\xi} x_{i-h}y_h \right|^2
\]
If however, the first few values of the sampled impulse response of the channel have a small magnitude, and the sample with the largest magnitude is $y_r$, where $0 < f \leq g$, then the value that $x_i$ takes on during the expansion of the vectors $\{Q_{i-1}\}$ will have little effect upon the costs of the expanded vectors $\{P_i\}$. This means that the vectors are selected almost arbitrarily. To prevent this occurring the detection of the signal element $x_{i-n}$ is delayed by $f$ sampling intervals [26].

The problem that this method presents is that the value of $f$ must be determined and, if the channel varies considerably with time, the value of $f$ must be continuously updated. Also since the complexity of the near maximum likelihood detector increases as the number of components in the sampled impulse response of the channel such that there are fewer components and such that its first (or one of the earliest components) has a significant magnitude relative to the other component magnitudes.

To obtain this 'desired' impulse response a filter is placed ahead of the detector, as shown in figure 2.3.2, such that the impulse response of the channel and filter is minimum phase [3, 27, 28].

A minimum phase impulse response has the important property that most of its energy is concentrated towards its first few components, with the latter components rapidly decaying in magnitude, therefore the magnitude of its first component should be relatively large.

The filter needed to make the sampled impulse response minimum phase is an adaptive linear feedforward transversal filter and is an allpass network with ideally an infinite number of taps and a flat amplitude impulse response.
2.7 Adaptive Adjustment -criteria

The ideal adjustment of the prefilter is as follows.

Let the estimate of the channel sampled impulse response be the \((g+1)\)-component vector \(Y\),

\[
Y = [y_0 \ y_1 \ \ldots \ y_g]
\]

which has a z-transform

\[
Y(z) = y_0 + y_1 z^{-1} + \ldots + y_g z^{-g}
\]

Equation 2.7.2 can be written as the product of two factors \(Y_1(z)\) and \(Y_2(z)\)

\[
Y(z) = Y_1(z)Y_2(z)
\]

where \(Y_1(z)\) and \(Y_2(z)\) are complex polynomials with roots (zeros) inside and outside the unit circle in the complex z-plane respectively and are given by

\[
Y_1(z) = n(1 + a_1 z^{-1})(1 + a_2 z^{-1})\ldots
\]

\[
\ldots (1 + a_{g-n} z^{-g})
\]

and

\[
Y_2(z) = z^{-m} (1 + \beta_1 z)(1 + \beta_2 z)\ldots(1 + \beta_m z)
\]
where \( |a_i| < 1 \) and \( |\beta_i| < 1 \), and \( a_i \) is the negative of a root of \( Y(z) \); \( \beta_i \) is the negative of the reciprocal of one of the roots of \( Y(z) \). The quantity \( n \) in Equation 2.7.4 is a complex valued constant needed to satisfy Equations 2.7.2 and 2.7.3.

The quantity \( m \) in Equation 2.7.5 is the number of roots of \( Y(z) \) that lie outside the unit circle in the complex \( z \)-plane. To achieve a minimum phase channel the roots of \( Y(z) \) that lie outside the unit circle \((-1/\beta_i)\) must be replaced by the complex conjugates of their reciprocal values. The \( z \)-transform of the sampled impulse response of the prefilter is now approximately

\[
D(z) = z^{-n} Y_2^{-1}(z) Y_3(z)
\]

\[
= d_0 + d_1 z^{-1} + \ldots + d_q z^{-q}
\]

where

\[
Y_3(z) = (1 + \beta_1^* z^{-1})(1 + \beta_2^* z^{-1})\ldots(1 + \beta_m^* z^{-1})
\]

and \( \beta_i^* \) is the complex conjugate of \( \beta_i \). The \( z \)-transform of the channel and linear filter in cascade will approximately be

\[
F(z) = Y(z)D(z)
\]

\[
= f_0 + f_1 z^{-1} + \ldots + f_{q+g} z^{-q-g}
\]

where \( f_h \approx 0 \) for \( h=0,1,\ldots(q-1) \). The estimate of the sampled impulse response of the combined channel and adaptive filter is the sequence \( F_m \), with \( z \)-transform

\[
F_m(z) = f_{m,0} + f_{m,1} z^{-1} + \ldots + f_{m,g} z^{-g} = z^g F(z) = Y_1(z)Y_3(z)
\]
which is needed by the near maximum likelihood detector. The delay of q sampling intervals introduced by the adaptive filter is, for convenience ignored here, but must obviously be taken into account when comparing $Y(z)D(z)$ and $F_m(z)$.

Practical algorithms that achieve the accurate adjustment of the prefilter are the subject of the following chapter. Their performance can be measured using three parameters $\psi_1$, $\psi_2$ and $\psi_3$ given by

\[ \psi_1 = 10 \log_{10} \left( \sum_{h=0}^{q-1} |f_{bh}|^2 \right) \quad 2.7.10 \]

\[ \psi_2 = 10 \log_{10} \left( \sum_{h=0}^{q+1} |f_{m,h} - f_{b+q}|^2 \right) \quad 2.7.11 \]

\[ \psi_3 = 10 \log_{10} \left( \sum_{h=0}^{q} |f_{b+h} - f_{m,h}|^2 \right) \quad 2.7.12 \]

The parameters $\psi_1$, $\psi_2$ and $\psi_3$ give the discrepancy between the actual sampled impulse response of the channel and adaptive filter and that assumed by the detector.

In particular the quantity $\psi_1$ (Equation 2.7.10) gives a measure of how close to zero the first q components of the sampled impulse response of the channel and filter in cascade are (Equation 2.7.8). The quantity $\psi_2$ (equation 2.7.11) represents the discrepancy between the minimum phase sequence (Equation 2.7.9) and the last $q+1$ components of the minimum phase sequence with a z-transform given by Equation 2.7.8. The last parameter, $\psi_3$, is a measure of the difference between the minimum
phase sequence \( \{f_{m,h}\} \), and the ideal minimum phase sequence \( \{f_{0,m}\} \) found using the EISPACK/MATLAB library.

Finally, another parameter \( \theta_i \) is used to assess the performance of the various root finding algorithms and is given by

\[
\theta_i = 10 \log_{10} \left( |\beta_h - \beta'_h|^2 \right)
\]  

where \( \beta_h \) is the negative of the reciprocal of the \( h^{th} \) root of \( Y(z) \) outside the unit circle (found using the EISPACK/MATLAB root-finding routines) and \( \beta'_h \) is the negative of the reciprocal of the same \( h^{th} \) root of \( Y(z) \) found by any of the root-finding algorithms of Chapter 3.

It is important to find very accurate adjustment methods since the factor which limits the highest achievable transmission rate over time invariant as well as time varying channels is most likely to be the accuracy to which the receiver can adjust the adaptive linear filter, which becomes quite critical particularly at the higher transmission rates [3] and this is the theme of the next Chapter.
Figure 2.2.1 Model of Digital Data Transmission System
Figure 2.2.2a Amplitude Response of Filter $A$.

$|A(t)|$

Figure 2.2.2b Frequency Spectrum of $z(t)$.

$Z(f)$

Figure 2.2.2c Amplitude Response of Filter $C$.

$|C(f)|$
Figure 2.2.3 Equivalent Baseband Model of Data Transmission System

Figure 2.2.4 Simplified Baseband Model of Data Transmission System used in the Computer Simulation
Figure 2.3.1 Block Diagram of the Linear Feedforward Channel Estimator

Figure 2.3.2 Model of Data Transmission System with 'Desired' Impulse Response
REFERENCES


CHAPTER THREE

SOME ROOT-FINDING ALGORITHMS AND SPECTRAL FACTORISATION TECHNIQUES FOR THE ADJUSTMENT OF THE PREFILTER

3.1 Introduction

This Chapter is concerned with the adaptive adjustment of a digital data receiver, when operating in the presence of signal distortion and additive noise. In particular, the chapter addresses the problem of adjusting the tap coefficients of the predetection filter, described in Chapter two, a process that involves computing the complex roots of complex-valued polynomials. The adjustment technique requires an estimate of the sampled impulse response of the channel over which data is to be transmitted, and involves the location of the roots of the z-transform of the sampled impulse response of the channel, that lie outside the unit circle in the z-plane.

One of the main requirements set for the adjustment algorithm is the high accuracy with which it should be able to adjust the predetection filter, since the factor which limits the highest achievable transmission rate over time invariant as well as time varying channels is most likely to be the accuracy to which the receiver can adjust the adaptive linear filter [1].

The second requirement is that the algorithm should be able to adaptively adjust the filter in a very short time, if the algorithm would be of use to a great many diverse environments (such as severely distorted telephone channels, HF radio links and perhaps land mobile radio links). A third very important requirement would be that the algorithm should very rarely fail to adjust the filter even if this is a partial failure (in the sense that the algorithm provides the receiver with a near minimum phase waveform instead of a true minimum phase waveform). The reason for this being that if the prefilter were to be used together with a near maximum likelihood detector (see Chapter 2), then this slight deviation from the ideal adjustment could be tolerated, however this is not the case when an adaptive equaliser is used instead.
Taking all these considerations into account, a collection of Algorithms for the adjustment of the prefilter is presented here with the view to investigating the relative merits of each and their potential to provide a 'global' solution. They fall broadly in two categories: Root finding algorithms and Spectral Factorisation techniques and are the subject of Sections 3.2 and 3.3 respectively.

3.2 ROOT FINDING METHODS

3.2.1 QR Method

The QR method is perhaps the best currently known iterative method for the calculation of all the eigenvalues. At the present time this is the most efficient (extremely stable numerically) and widely used general method for the calculation of all of the eigenvalues of a matrix [see also Appendix 3 - Section A3.8]. The QR method has been a subject of intense investigation since it was first published. It is quite complex in both its theory and application and only an introduction is given here to the theory of the method. (The method is mentioned for completeness as well as because it is used as the standard method/routine for calculating eigenvalues for comparisons throughout this thesis and also although expensive for the case of short impulse responses becomes attractive. Alternatively, the Schur method [see Appendix 3 - Section A3.7], can be employed for benchmark tests). The QR method is based on the use of orthogonal transformations.

For an arbitrary matrix $A$, there exist an orthogonal matrix $Q$ and an upper triangular matrix $R$ such that

$$ A = QR $$  \hspace{1cm} \text{(3.2.1.1)}

Here $R$ is upper triangular and $Q$ is orthogonal that is

$$ Q^TQ = 1 $$  \hspace{1cm} \text{(3.2.1.2)}

where $Q^T$ is the transpose matrix of $Q$. 
The standard algorithm for the QR decomposition involves successive Householder transformations. The general form of a Householder matrix is

\[ U = I - 2ww' \]  \hspace{1cm} (3.2.1.3)

with

\[ \|w\|_2^2 = w'w = 1 \]

An appropriate Householder matrix applied to a given matrix can zero all elements in a column of the matrix situated below a chosen element. Thus the first Householder matrix \( Q_1 \) can be chosen to zero all elements in the first column of \( A \) below the first element. Similarly \( Q_2 \) zeroes all elements in the second column below the second element, and so on up to \( Q_{n-1} \). Therefore

\[ R = Q_{n-1} \ldots Q_1 A \]  \hspace{1cm} (3.2.1.4)

Since the Householder matrices are orthogonal,

\[ Q = (Q_{n-1} \ldots Q_1)^{-1} = Q_1 \ldots Q_{n-1} \]  \hspace{1cm} (3.2.1.5)

Beginning with \( A_1 = A \), the QR method generates matrices \( Q_i \), \( R_i \), and \( A_i \) according to the following rules:

\[ A_1 = Q_1 R_1 \]

\[ Q_i^T Q_i = I \]  \hspace{1cm} (3.2.1.6)

\[ R_i = \begin{bmatrix} * & \cdots & * \\ \vdots & \ddots & \vdots \\ 0 & \cdots & * \end{bmatrix} \]

\[ A_{i+1} = R_i Q_i \]
In this way the matrices have been factored into a product of a unitary matrix \( Q \) and an upper triangular matrix \( R \). A QR decomposition always exists and can be computed in a numerically stable way by applying, for instance, Householder transformations. There are \( n-1 \) Householder matrices \( H_j^{(0)}, j=1, \ldots, n-1 \), that satisfy the following Equation

\[
H_n^{(0)} \ldots H_1^{(0)} A_1 = R_1
\]

where \( R_i \) is upper triangular. Then, since the Householder matrices are unitary \( (H_j^{(0)})^*=(H_j^{(0)})^{-1} \) the matrix

\[
Q_i = H_1^{(0)} \ldots H_{n-1}^{(0)}
\]

is unitary and satisfies \( A_i=Q_i R_i \), so that \( A_{i+1} \) can be computed as \( A_{i+1}=R_i Q_i \). Then \( \lambda_{i+1} \) can be computed

\[
A_{i+1} = R_i Q_i = R_i H_n^{(0)} \ldots H_1^{(0)}
\]

If \( A \) satisfies any one number of conditions on its form, then the matrices \( \{A_i\} \) converge to an upper triangular matrix \( A_\infty \) with the eigenvalues \( \lambda_j \) of \( A \) as diagonal elements, \( \lambda_j=(A_\infty)_{jj} \), or to a near-triangular matrix from which the eigenvalues can be easily calculated.

The QR method can be relatively expensive because the QR factorisation is time consuming when repeated many times: a complete step \( A_i \rightarrow A_{i+1} \) for a dense \( nxn \) matrix \( A \) requires \( O(n^3) \) operations. To avoid this drawback the QR method should be applied to matrix \( A \), only when the latter has been reduced to a simpler form. In the general case, if matrix \( A \) is a nonsymmetric matrix must first be reduced to a similar Hessenberg matrix. A matrix \( B \) is Hessenberg if \( B_{ij}=0 \) for all \( i > j+1 \). Also in its original form the convergence is usually slow. To remedy this disadvantage a technique known as shifting is used to accelerate convergence. The QR method is generally applied with a shift of origin for the eigenvalues in order to increase the speed of convergence. For a sequence of constants \( \{c_i\} \), define \( A_1=A \) and

\[
A_1 - c_1 I = Q_1 R_1
\]

\[
A_{i+1} = c_i I + R_i Q_i
\]
It can be shown that the matrices $A_i$ are similar to $A_{i+1}$ and as a consequence the
eigenvalues of $A_{i+1}$ are the same as those of $A_i$, and thus the same as those of the original matrix
A. In this way, the QR method becomes a rapid general purpose method, faster and more
accurate than any other general method at the present time. A practical implementation of the
QR algorithm is shown in Figure 3.1. This method however accurate is far too computationally
involved for inclusion in a real time system, therefore other methods should be considered, that
are less complex, and at the same time exhibit the same (or as close as possible) desired behaviour
as far as accuracy is concerned, to the QR method. In this context the next methods are
presented.

3.2.2.i Newton-Raphson method

Perhaps the most celebrated of all root-finding routines is Newton's method, also called the
Newton-Raphson method. Here, it is extended to cope with complex-valued polynomials $Y(z)$
(See Appendix A3.6). This method requires the evaluation of both a function, say $f(x)$ and its
derivative at arbitrary points, say x. Algebraically, the method derives from the familiar Taylor
series expansion of the function $f(x)$ in the neighbourhood of a point,

$$f(x + \delta) \approx f(x) + f'(x)\delta + \frac{f''(x)}{2!}\delta^2 + \ldots$$  3.2.2.1

where $f'(x)$ and $f''(x)$ represent the first and second derivatives of the function $f(x)$.

When $x$ is close to the actual root, or in other words when $\delta$ becomes fairly small, then Equation
3.2.2.1 reduces to

$$\delta = -\frac{f(x)}{f'(x)}$$  3.2.2.2

since the terms beyond linear become unimportant. Equation 3.2.2.1 then can be solved for
Therefore, the Newton-Raphson method approximates the roots through an iterative process. The iterative algorithm used is given by Equation 3.2.2.3 and applied to the polynomial \( Y(z) \) which represents the sampled impulse responses of telephone channels assumed in this study is,

\[
x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{3.2.2.3}
\]

where \( x_{i+1} \) is the estimate of the value of the \( i \)th root after \( n \) iterations of the algorithm, \( Y(\gamma_{ln}) \) is the value polynomial under consideration evaluated at \( \gamma_{ln} \) and \( Y'(\gamma_{ln}) \) is the value of the first derivative evaluated at \( \gamma_{ln} \). The iterative process suggested by equation 3.2.2.3-4 is repeated until \( |\gamma_{ln+1} - \gamma_{ln}| < \delta \), where \( \delta \) is some small, real-valued quantity. When this is achieved, the process has converged to a root of the polynomial, within the accuracy set by the value of \( \delta \). Once located, the first root of the polynomial must be removed in order for the algorithm to be reapplied to locate a second root without fear of the same first root being located. This process, achieved by polynomial deflation [1] forms the new reduced-order polynomial, \( Y(z)/(z-\gamma_{ln}) \). The iterative process described by equation 3.2.2.3-4 is then applied to the deflated polynomial to locate the next root of \( Y(z) \).

\[
\gamma_{ln+1} = \gamma_{ln} - \frac{Y(\gamma_{ln})}{Y'(\gamma_{ln})} \tag{3.2.2.4}
\]

a. The Newton-Raphson method in its basic form was used to locate all the roots of \( Y(z) \), whereas the roots required for the adjustment of the linear predetection filter are only those lying outside the unit circle in the z-plane. Since the position of the roots of the polynomial equation \( Y(z) \) is completely unknown many initial starting points may have to be tried, to ensure convergence to all roots of the polynomial. A knowledge of the approximate root positions will obviously aid the selection of an initial starting value \( \gamma_0 \). However the algorithm does not necessarily find roots in any given order (for example, in order of decreasing moduli). Therefore, all \( g \) roots must be located before the roots, (say \( k \) roots) which lie outside the unit circle can be identified. This algorithm is a modification of the basic Newton-Raphson technique, so that it attempts to locate only the roots that lie outside the unit circle in the z-plane. Clearly, this
influences greatly the time taken for the adjustment of the predetection filter, especially when a hardware implementation of the algorithm is concerned. The algorithm operates as follows:

**Step 1.** Using the basic Newton method it locates and removes the roots of \( Y(z) \) until a root with modules less than unity is encountered.

**Step 2.** Having located such a root, the input polynomial \( Y(z) \) is truncated to yield \( Y_T(z) \) by dropping all the lower-degree coefficients of \( Y(z) \), such that \( Y_T(z) \) has \( g' \) components. The basic techniques used in the original Newton-Raphson are then employed to locate all the roots of \( Y_T(z) \).

**Step 3.** The basic Newton method (Step 1) is then applied to the original polynomial \( Y(z) \). The starting point for the algorithm, is the negative of the reciprocal of the largest root (that is the one with the greatest modulus) of \( Y_T(z) \) in step 2. This process results in a more accurate value for the particular root. If the chosen starting point results in a root of modulus less than unity the algorithm terminates with the decision that no more roots lie outside the unit circle, otherwise steps 1-3 will be repeated until the degree of the reduced polynomial reaches one. The Newton-Raphson method in this new form is still quite involved and furthermore it misses quite a few roots most of the time. It heavily depends on the right choice of the initial guesses (points). Since this represents a quite heavy computational overhead for this method, it is one of the most serious pitfalls of this method in general and for the application considered (i.e. the prefilter adjustment) in particular. It would be highly desirable to pinpoint more accurate starting points for the algorithm as this would lead to better root approximations (increased accuracy) and reduced number of arithmetic operations. In this context the next algorithm is presented.

**b.** Newton method is combined with the QR method to give a new algorithm of reduced computational complexity and increased accuracy. For reference purposes this hybrid scheme will be called QRN method. In QRN the input polynomial \( Y(z) \) is truncated by dropping the lower order coefficients to yield \( Y_T(z) \). The truncated polynomial \( Y_T(z) \) is of order say \( n \), where \( n < g \) where \( g \) is the order of the original polynomial \( Y(z) \). Now, the QR method is applied to locate the \((n-1)\) roots of the truncated polynomial \( Y_T(z) \). These are subsequently used as starting points for the Newton method in order to locate the \( g-1 \) roots of the polynomial \( Y(z) \). Since the QR method is an extremely accurate method, the truncated polynomial \( Y_T(z) \), (on which the
method is applied) can be very short, especially in the case of the milder channels i.e. for channel 1 it could be of length 4 and for channels 2 and 3 it could be of length 5).

The roots of the truncated polynomial are quite close to the roots of the original polynomial and can be located by the QR method without further bias being introduced. This fact makes the application of the QR method attractive since it is applied on a short polynomial (or more accurately on a short matrix).

Although the starting points are better for most of the channels assumed here, still the algorithm occasionally fails to operate satisfactorily (i.e. fails to provide valid initial guesses for the more severely distorted channels) and the arithmetic complexity is still not sufficiently reduced. On some instances, contrary to what was expected, the number of arithmetic operations increases and furthermore the algorithm collapses, (fails to find any roots) as it becomes evident from the simulation in Chapter 4. In view of the difficulties that the QRN algorithm faces, another way for providing those initial points has to be found.

c. The previous two algorithms occasionally fail to converge to all the roots of the sampled impulse response of some of the channels tested (where there is severe distortion). This version of the Newton-Raphson algorithm has been proposed [2] in order to alleviate the problems experienced by the previous algorithms. The algorithm operates in the same iterative way as algorithms 3.2.2.i, a and b, but the starting and terminating procedures are different.

Furthermore, the new algorithm provides the detector with the estimate of the sampled impulse response of the channel and predetection filter. The algorithm operates as follows:

**Step 1.** The algorithm locates the roots of the z-transform of the sampled impulse response of the channel using the basic Newton method.

**Step 2.** When a located root has a modulus less than unity, the input polynomial is truncated (as in step 2 of algorithm 3.2.2.i.a) to yield $Y_r(z)$, otherwise the located root must be removed by a process of polynomial deflation to give a reduced degree polynomial. In the latter case the complex conjugate of the reciprocal of the located root is inserted to give a new modified polynomial $Y_m(z)$ which is then used in place of $Y(z)$ to locate the next root using the value of the most recently located root as a starting value instead of zero.
Step 3. Next, $Y_m(z)$ is truncated to yield $Y_r(z)$ with $(g'+1)$ components by dropping all the lower degree components of $Y_m(z)$. The $g'$ roots of $Y_r(z)$ are then located using the basic Newton method and stored in an array.

Step 4. The $g'$ stored roots are then used in turn as starting values in the basic Newton method (3.2.2.i) to give the corresponding values of roots in $Y_m(z)$. If this results in a root with modulus greater than unity, the root is removed from $Y_m(z)$ and replaced by the complex conjugate of its reciprocal, otherwise the next starting value will be used. When all the $g'$ values in the array have been used the algorithm is terminated and $Y_m(z)$ represents the sampled impulse response of the channel and predetection filter.

Although this algorithm was expected in principle to have an enhanced performance, it still presented the initial difficulties of missing roots on many occasions and that of increased computational overhead. In view of all these modifications leading to suboptimum results the approach of using some other algorithm (or the algorithm itself) to provide rough initial approximations to the desired roots is abandoned altogether and the Newton-Raphson algorithm is applied in a whole new (and simple) manner to give the best so far results in terms of accuracy and computational cost.

3.2.2.ii. An imaginative way to use the Newton's method to locate only the desired roots of a polynomial (i.e. the roots that are outside the unit circle) is to reverse the original polynomial $Y(z)$ and use it as input to the process. Now, the required roots of the original polynomial are inside the unit circle and close to zero. Since the number of these roots is small the algorithm has no difficulty in locating them. The algorithm uses $0+i*0$ (or any point close to the origin for that matter) and locates the required roots (i.e. the roots that now lie inside the unit circle). The algorithm terminates when it converges to a root with modulus greater than one, assuming that it has located all the wanted roots. The Newton-Raphson algorithm has been applied in a different way and has yielded the best results so far as accuracy and complexity are concerned. Limitations of the method, such as the need for accurate and many initial guesses has been eliminated, thus giving the best algorithm so far. However, some of the limitations of the original method such as the inability to converge to wanted roots over the more distorted channels remain (See Chapter 4). Therefore, bearing in mind that a very robust algorithm has to be found, the quest for that 'perfect algorithm' continues.
3.2.3.i. Laguerre's method [3]

This is one of a very small number of methods that will converge to all types of roots: real, complex, single or multiple. It does require complex arithmetic, even while converging to real roots. Here, the complex arithmetic is no disadvantage, since the polynomial $Y(z)$ which represents the z-transform of the sampled impulse response of the channel, has complex coefficients. Although this method is guaranteed to be always convergent for a polynomial with real roots, this is not always the case with polynomials with at least some complex roots. However much empirical experience suggests that this nonconvergence is extremely unusual. Furthermore, it can almost always be fixed by a simple scheme to break a nonconverging limit cycle.

To derive the Laguerre formulae, the following relations between the polynomial and its roots and derivatives are noted

$$Y(z) = (z-z_1)(z-z_2)\ldots(z-z_n) \quad 3.2.3.1$$

$$\ln|Y_n(z)| = \ln|z-z_1| + \ln|z-z_2| + \ldots + \ln|z-z_n| \quad 3.2.3.2$$

$$\frac{d\ln|Y(z)|}{dz} = \frac{1}{z-z_1} + \frac{1}{z-z_2} + \ldots + \frac{1}{z-z_n} \quad 3.2.3.3$$

$$\frac{Y''}{Y} = G$$

$$-\frac{d^2 \ln|Y(z)|}{dz^2} = \frac{1}{(z-z_1)^2} + \frac{1}{(z-z_2)^2} + \ldots + \frac{1}{(z-z_n)^2} \quad 3.2.3.4$$

$$+ \frac{1}{(z-z_n)^2} = \left[ \frac{P_n'}{P_n} \right]^2 - \frac{P_n''}{P_n} \equiv H$$

Now, if the wanted root $z_1$ lies at some distance $a$ from the current guess $z$, while all other roots are assumed to be located at a distance $b$
Finding Methods

\[ z_1 - z_i = a; \quad z_4 = b \quad i = 2, 3, ..., g \]

Equations 3.2.3.3, 3.2.3.4 can be expressed as

\[ \frac{1}{a} + \frac{g - 1}{b} = G \]  

\[ \frac{1}{a^2} + \frac{g - 1}{b^2} = H \]

which yields as the solution for \( a \)

\[ a = \frac{g}{G \pm \sqrt{(g - 1)(gH - G^2)}} \]

where the sign should be taken to yield the largest magnitude for the denominator. Since the factor inside the square root can be negative, \( a \) can be complex.

The method operates in an iterative fashion: For a trial value \( z \), \( a \) is calculated by equation 3.2.3.8. The difference \((z - a)\) becomes the next trial value. This continues until \( a \) is sufficiently small. This method is a very powerful method and exhibits better convergence properties, but is more computationally involved than the basic Newton-Raphson. Moreover, although nonconvergence for this method is rare, it is still there. Therefore, it is evident that there is scope for improvement and further modifications.

Some of these modifications are next presented.

\( a \). The Laguerre method is combined here with QR method to give an algorithm of reduced computational complexity and great accuracy, similar to the QRN structure. In this version of Laguerre method, call it QRL, the input polynomial \( Y(z) \) is truncated by dropping the lowest order coefficients to yield \( Y_{\tau}(z) \). The truncated polynomial \( Y_{\tau}(z) \) is of order say \( n \), where \( n < g \) where \( g \) is the order of the original polynomial \( Y(z) \). Now, the QR method is applied to locate the
(n-1) roots of the truncated polynomial $Y_T(z)$. These are subsequently used as starting points for the Laguerre method in order to locate the $g-1$ roots of the polynomial $Y(z)$. This scheme proved to be very computationally involved and showed no improved convergence rate when compared with any of the previous methods. This disappointing behaviour led to the next scheme.

b. The algorithm operates as follows:

**Step 1.** Using the basic Laguerre method it locates and removes the roots of $Y(z)$ until a root with modulus less than unity is encountered.

**Step 2.** Having located such a root, the input polynomial $Y(z)$ is truncated to yield $Y_T(z)$ by dropping all the lower-degree coefficients of $Y(z)$, such that $Y_T(z)$ has $g'$ components. The basic techniques used in the original Laguerre are then employed to locate all the roots of $Y_T(z)$.

**Step 3.** The basic Laguerre method (Step 1) is then applied to the original polynomial $Y(z)$. The starting point for the algorithm, is the negative of the reciprocal of the largest root (that is the one with the greatest modulus) of $Y_T(z)$ in step 2. This process results in a more accurate value for the particular root. If the chosen starting point results in a root of modulus less than unity the algorithm terminates with the decision that no more roots lie outside the unit circle, otherwise steps 1-3 will be repeated until the degree of the reduced polynomial reaches one.

3.2.3.ii As in the case of Newton's method an imaginative way to use Laguerre's method to locate only the desired roots of a polynomial (i.e. the roots that are outside the unit circle) is to reverse the original polynomial $Y(z)$ and use it as input to the process. Now, the required roots of the original polynomial are inside the unit circle and close to zero. Since the number of these roots is small the algorithm has no difficulty to locate them using as starting point the obvious $0+i*0$. The algorithm terminates when it encounters a root with modulus greater than one. This algorithm is conclusively the best of the root finding methods tested (See Chapter 4). It is very accurate, converges over all channels tested but although it is far less complex arithmetically than all the previous algorithms, it would be desirable from a real time systems point of view for a faster algorithm to be found.

3.2.4 Adjustment scheme for the root finding algorithms

The algorithms of Section 3.2 form part of a wider adaptive process for the adjustment of digital data receivers. An adjustment scheme is presented that make use of the knowledge of the roots
found by the root finding algorithm (algorithm) and adjust the prefilter according to some
adjustment criterion. In this case the adjustment criterion is that of maximising the signal to noise
ratio at the detector input when the tap coefficients are given by the last \((n+1)\) coefficients of the
infinite set of values used in the ideal filter (where \(n\) is an appropriate integer).

i.-Adjustment scheme

Let \(Y(z)\) be the \(z\)-transform of the sampled impulse response of the linear baseband channel,
given by

$$Y(z) = y_0 + y_1 z^{-1} + \cdots + y_g z^{-g}$$  \hspace{1cm} 3.2.4.1

$$= y_0 (1 + a_1 z^{-1})(1 + a_2 z^{-1}) \cdots (1 + a_g z^{-1})$$  \hspace{1cm} 3.2.4.2

where \(\{a_i\}\) are the negative of the \(g\) roots (zeros) of \(Y(z)\), such that \(Y(z = -a_i) = 0\).

\(Y(z)\) can also be written as

$$Y(z) = Y_1(z)Y_2(z)$$  \hspace{1cm} 3.2.4.3

where

$$Y_1(z) = y_0(1+a_{1,1}z^{-1})(1+a_{1,2}z^{-1}) \cdots (1+a_{1,g}z^{-1})$$  \hspace{1cm} 3.2.4.4

with all zeros lying within the unit circle in the \(z\)-plane, and

$$Y_2(z) = (1+a_{2,1}z^{-1})(1+a_{2,2}z^{-1}) \cdots (1+a_{2,k}z^{-1})$$  \hspace{1cm} 3.2.4.5

with all its zeros lying outside the unit circle.
The predetection filter $D$, in figure 3.2.3 is a linear feedforward transversal filter whose tap coefficients are given by the $(n+1)$-component vector

$$D = [d_0 \ d_1 \ ... \ d_n] \tag{3.2.4.5}$$

with $z$-transform

$$D(z) = d_0 + d_1 z^{-1} + \ldots + d_n z^{-n} \tag{3.2.4.6}$$

The resultant sampled impulse response of the channel and filter is therefore the $(n+g+1)$-component vector,

$$E = [e_0 \ e_1 \ ... \ e_{ng}] \tag{3.2.4.7}$$

whose $z$-transform is,

$$E(z) = Y(z) D(z) \tag{3.2.4.8}$$

$$= e_0 + e_1 z^{-1} + \ldots + e_{ng} z^{-(n+g)}$$

Since $E(z)$ is required to be the $z$-transform of the minimum phase version of $Y(z)$, say $F(z)$, (which is achieved by replacing those zeros of $Y(z)$ with a modulus greater than one by the complex-conjugate of their reciprocals, then, ideally, $E(z)$ will have $(g+1)$ components as,

$$E(z) = F(z) = Y_1(z) Y_3(z) \tag{3.2.4.9}$$

$$= f_0 + f_1 z^{-1} + \ldots + f_g z^{-g}$$

where
\[ Y_3(z) = \left( a_{2,1}^* + z^{-1} \right) \left( a_{2,2}^* + z^{-1} \right) \cdots \left( a_{2,k}^* + z^{-1} \right) \]  \hspace{1cm} 3.2.4.10

and where \( a_{2,i}^* \) are the complex conjugates of the \( a_{2,i} \).

The \( z \)-transform of the filter (equation 3.4.8) is

\[ D(z) = E(z)Y^{-1}(z) = \frac{e_0 + e_1z^{-1} + \ldots e_{n+e}z^{-(n+e)}}{y_0(1+a_1z^{-1})(1+a_2z^{-2}) \ldots (1+a_ez^{-e})} \] \hspace{1cm} 3.2.4.11

which can be expressed as

\[ D(z) = z^{-n}Y_3(z)Y_2^{-1}(z) \] \hspace{1cm} 3.2.4.12

where \( z^n \) is included to ensure a realisable transfer function.

The filter \( D \), therefore introduces a delay of \( n \) sample periods. That is, the output signal from the filter, when fed with the sampled impulse response of the channel, will consist of \( n \) samples whose values should ideally be zero (called the pre-cursor components), followed by \( (g+1) \) samples whose \( z \)-transform is the value of \( E(z) \), which itself is ideally the \( z \)-transform of the minimum phase version of \( Y(z) \), \( F(z) \).

If it is assumed that the root finding algorithm has successfully determined the \( k \) roots (zeros) of \( Y(z) \) that lie outside the unit circle in the \( z \)-plane, then both \( Y_2(z) \) and \( Y_3(z) \) can be formed using equations 3.2.4.4 and 3.2.4.10, respectively, and so the problem of determining the tap coefficients of the filter \( D \) now reduces to the evaluation of \( Y_2^{-1}(z) \). Since all of the factors of \( Y_2(z) \) are of the form \((1+a_2iz^{-1})\) where the \(|a_{2,i}| > 1\), then \( Y_2^{-1}(z) \) cannot be determined by the long division of one by \( Y_2(z) \), nor by the application of the Binomial expansion, since the resultant infinite series in both cases will be divergent. Consequently, \( Y_2^{-1}(z) \) must be determined by some other approach.
Consider the sequence with z-transform

\[ N(z) = \prod_{i=1}^{k} (a_{2i} + z^{-1}) \] 3.2.4.13

which is obtained from \( Y_2(z) \) by reversing the order of its coefficients. The roots of \( N(z) \) are the negative of the reciprocal of those of \( Y_2(z) \) and will lie inside the unit circle in the z-plane. Thus, long division of one by \( N(z) \) will result in a convergent series. The resulting infinite series in \( z^{-1} \) representing \( N^{-1}(z) \) is then truncated to the first \( J \) terms, such that

\[ N^{-1}(z) = n_0 + n_1 z^{-1} + \ldots + n_J z^{-J+1} \] 3.2.4.14

then \( C(z) \) can be formed by reversing the order of the coefficients of \( N^{-1}(z) \),

\[ C(z) = n_J + n_{J-1} z^{-1} + \ldots + n_0 z^{-J+1} \] 3.2.4.15

and so

\[ C(z) Y_2(z) = z^{-J} \] 3.2.4.16

\[ D(z) \approx Y_3(z) C(z) \] 3.2.4.17

where the number of tap coefficient of the linear filter \( D \) is \((n+1)=(J+1)\), and where \( J \) is the delay introduced by the filter. Therefore, \( D(z) \) performs the required filtering process but also introduces a delay of \( n \) sample periods. The approximation given in equation 3.2.4.17 gets better as \( J \to \infty \). However, for the practical implementation of the predetection filter, \( J \) must be obviously reduced to some finite number, the larger the value of \( J \), the more accurate is the filtering process but the filter will introduce a larger delay.
3.2.5 Clark-Hau algorithm

With this algorithm the adjustment of the adaptive pre-filter proceeds as follows. The algorithm first forms a filter, with z-transform

\[ A_1(z) = \frac{1}{1 + \lambda_1 z} \]  

for \( i=0,1,\ldots,k \) in turn, where \( \lambda_i \) is an estimate of one of the roots of the input polynomial \( Y(z) \) that lie outside the unit circle in the z-plane. The sequence \( Y \) is now fed into this filter, and an iterative process used to adjust \( \lambda_i \) such that \( \lambda_i \rightarrow \beta_i \), where \( \beta_i \) is the negative of the reciprocal of the first root of \( Y(z) \) to be processed by the algorithm and, of course, \( |\beta_i|<1 \). The filter with z-transform \( A_i(z) \), does not operate on the received signal in real time therefore its z-transform is not limited to zero and negative powers of \( z \). To achieve the effect of passing the sequence \( Y \) through a filter with z-transform \( A_i(z) \), the sequence \( Y \) is first reversed in order, starting with the last component \( y_0 \) at time \( t=0 \) and ending with the first component \( y_n \) at time \( t=gT \), to give the sequence

\[ Y_R = [y_n, y_{n-1}, \ldots, y_0] \]  

which is then fed into a filter with z-transform

\[ A'_1(z) = \frac{1}{1 + \lambda_1 z^{-1}} \]  

which can be realised as a one tap feedback filter, as shown in figure 3.2. The reversed sequence \( Y_R \), will appear to be moving backwards in time, starting with \( y_n \) at time \( t=0 \). Thus, a delay of one sampling interval, \( T \), in the feedback filter becomes an advance of \( T \) seconds, with z-transform \( z \). The effective z-transform of the filter in figure 3.2.1, when used in the way described, will be given by equation 3.2.5.3 and the output of the filter will be the sequence \( \{e_{ih}\} \), for \( h=0,1,\ldots,g \) which has a z-transform \( Y(z)A_i(z) \), given by
\[ A_i(z) = \ldots + e_{i,1}z + e_{i,0} + e_{i,1}z^{-1} + \ldots + e_{i,g}z^{-g} \] 3.2.5.4

Since, as previously mentioned, the operation just described is not carried out in real time but on the stored sequence \( Y_i \), there is nothing to prevent the components \( e_{i-1}, e_{i-2}, \ldots \) being non-zero. However, as it happens, these components are never used or even generated in the iterative process, which operates on the sequence of \( g+1 \) components \( e_{i,0}, e_{i,1}, \ldots, e_{i,g} \).

Now, it can be shown [4]

\[ e_{i,0} = (\beta_1 - \lambda_i)(u_0 - u_1 \lambda_i + u_2 \lambda_i^2 - \ldots + u_{g+1}(-\lambda_i)^{g+1}) \] 3.2.5.5

If \( \lambda_i \) is close to \( \beta_1 \) then \( e_{i,h} \approx u_{h-1} \), for \( h=1,2,\ldots,g \) and so equation 3.2.5.5 becomes [4]

\[ e_{i,0} \approx (\beta_1 - \lambda_i)(e_{i,1} - e_{i,2} \lambda_i + e_{i,3} \lambda_i^2 - \ldots + e_{i,g}(-\lambda_i)^{g+1}) \]

\[ \approx (\beta_1 - \lambda_i) \epsilon_i = 0 \] 3.2.5.6

where

\[ \epsilon_i = e_{i,1} - e_{i,2} \lambda_i + \ldots + e_{i,g}(-\lambda_i)^{g+1} \] 3.2.5.7

From 3.2.6 it can be seen that

\[ \beta_1 \approx \lambda_i + \frac{e_{i,0}}{\epsilon_i} \] 3.2.5.8

which suggests the following algorithm for finding \( \beta \),

\[ \lambda_{i+1} = \lambda_i + \frac{ce_{i,0}}{\epsilon_i} \] 3.2.5.9
where $\lambda_i$ is a better estimate of $\beta_i$ and $c$ is a small positive constant in the range 0 to 1.

Furthermore, as the value of $\lambda_i$ approaches that of $\beta_i$, i.e. $\lambda_i \rightarrow \beta_i$, the quantity $\frac{e_{i,0}}{e_i}$ becomes smaller and in the limit, when $\lambda_i = \beta_i$, this term becomes zero, which suggests the following stopping criterion for convergence

$$\left| \frac{e_{i,0}}{e_i} \right|^2 < d \quad 3.2.5.10$$

where $d$ is some suitably small threshold. At the end of the iterative process, when 3.2.5.10 is satisfied, i.e. $i = k$, the z-transform of the one-tap feedback filter is

$$A_k(z) = \frac{1}{1 + \beta_1 z} \quad 3.2.5.11$$

and one of the roots of $Y(z)$ that lies outside the unit circle in the z-plane will have been found. Since $|\beta_i| < 1$, the search for the values of $\beta_i$ is limited to values that lie within the unit circle with $|\lambda_i| < 1$.

The receiver now forms a two tap linear feedforward filter, as shown in figure 3.3, with z-transform

$$B_k(z) = 1 + \lambda_k z^{-1} = 1 + \beta_1 z^{-1} \quad 3.2.5.12$$

The output of the one tap feedback filter, $\{e_{h}\}$, for $h=0,1,...,g$, is now fed into the feedforward filter, in the correct order, starting with the component $e_{k,0}$ (which is taken to occur at time $t=0$) to give the $(g+2)$-component sequence, with z-transform

$$F'(z) = f_{1,0} + f_{1,0}z^{-1} + f_{1,1}z^{-2} + ... + f_{1,g}z^{-g+1} \quad 3.2.5.13$$
which is approximately equal to \( Y_i(z)A_k(z)B_k(z) \), and where \( f_{i-1} = 0 \). The overall effect of passing the sequence \( Y \) through the one tap feedback filter \( A \), and the two tap feedforward filter \( B \), is equivalent to passing it through a single filter with \( z \)-transform

\[
C_1(z) = \frac{1 + \beta_1^*z^{-1}}{1 + \beta_1 z} \tag{3.2.5.14}
\]

Finally, the output sequence of the \( \{f_{iB}\} \) is advanced by one place and the first component, \( f_{i-1} \), discarded, to give the sequence \( F_1 \), with \( z \)-transform

\[
F_1(z) = f_{1,0} + f_{1,1}z^{-1} + \ldots f_{1,q}z^q \approx zC_1(z)Y_i(z) \tag{3.2.5.15}
\]

For practical purposes, the linear factor \( (1+\beta_1 z) \) of \( Y(z) \), in equation 2.7.5 (Chapter 2) is replaced in \( F_1(z) \) by the linear factor \( (1+\beta_1^* z^{-1}) \). Thus the root \( -\frac{1}{\beta_1} \) of \( Y(z) \) is replaced by the root \(-\beta_1^*\), which is the complex conjugate of the reciprocal and lies inside the unit circle. In addition \( F_1(z) \) contains an advance of one sampling interval. The sequence \( F_1 \), with \( z \)-transform \( F_1(z) \), is an estimate of the sampled impulse response of the channel and adaptive linear transversal filter (Figure 3.4), when the \( z \)-transform of the latter is

\[
D_1(z) = zC_1(z) \tag{3.2.5.16}
\]

Initially, before the algorithm is started, all the taps of the prefilter are set to zero, except the \((q+1)^{th}\) tap, \( d_0 \), whose value is set to unity. Thus, the initial transform of the filter will be

\[
D_0(z) = z^q \tag{3.2.5.17}
\]

which is a delayed unit impulse and the initial \( z \)-transform of the channel and filter will be \( z^q Y(z) \). After the algorithm has converged, to give a value for \( \beta_1 \), the sequence \( D_0 \) is reversed in order and fed into the one tap feedback filter \( A \), the first \( q+1 \) components at its output are then fed into
the two-tap feedforward filter B. The first q+2 components from the output of filter B are then advanced by one place to give the sequence \( D_1 \), with z-transform

\[ D_1(z) = zC_1(z)D_0(z) = z^{-1}C_1(z) \]

The coefficients of \( D_1(z) \) are the required tap weights of the prefilter.

The whole of the above procedure is now repeated, but with \( F_1(z) \) in place of \( Y(z) \) and \( D_1(z) \) in place of \( D_0(z) \). At the end of the second iterative process, the value of \( \beta_2 \) will have been found, and the values of \( \beta_2 \), the coefficients of \( F_1(z) \) and \( D_1(z) \) determine \( F_2(z) \) and \( D_2(z) \) which are then used in place of \( F_1(z) \) and \( D_1(z) \) for the location of \( \beta_3 \). When all of the values of \( \beta_r \) for \( r = 1, 2, \ldots, m \) have been determined, the z-transform of the adaptive filter, \( D_m(z) \), will be approximately be

\[ D_m(z) = z^{-q m} C_1(z) C_2(z) \ldots C_n(z) = D(z) \]

and the z-transform of the channel and adaptive filter is given by

\[ F(z) = Y(z)D(z) \approx f_0 + f_1 z^{-1} + \ldots + f_q z^{-q-2} \]

where \( f_h \approx 0 \) for \( h=0,1,\ldots,(q-1) \). The estimate of the sampled impulse response of the combined channel and adaptive filter is the sequence \( F_m \), with z-transform

\[ F_m(z) = f_{m,0} + f_{m,1} z^{-1} + \ldots + f_{m,q} z^{-q} = z^q F(z) = Y_1(z)Y_3(z) \]

which is needed by the near maximum likelihood detector. The delay of q sampling intervals introduced by the adaptive filter is, for convenience ignored here, but must obviously be taken into account when comparing \( Y(z)D(z) = F(z) \) and \( F_m(z) \).
To start the algorithm off, some initial guess of the value of $\lambda_0$, (equation 3.2.5.1), is needed. Two possible sets of starting points that were used in the tests[4] were in order of selection: first set {0, 0.5, -0.5i, 0.5i}, and second set {0.5-i0.5, 0.5+0.5, -0.5-i0.5, -0.5+i0.5}. Whilst the algorithm is running, every time $\lambda_{i+1}$ in equation 3.2.5.9, is calculated a check is made to see if $|\lambda_{i+1}|$ is greater than unity, that is the algorithm starts to converge towards a root inside the unit circle. If $|\lambda_{i+1}|$ is greater than unity then the algorithm is restarted with a different starting point. Another check that is made (whilst the algorithm is running) is to see if the number of iterations (the value of $i$) exceeds an upper limit, say, $i>40$. The maximum value of 40 set for $i$ is a compromise between the need for an adequate number of iterations to determine a root and the need to minimise the time wasted when the iterative process has diverged. If more than forty iterations are required to locate a root then the algorithm is stopped and another starting point tried. If all of the starting points have been tried and the algorithm still fails to locate a root then is assumed that all of the roots of $Y(z)$ that lie outside the unit circle in the z-plane have been found. Figure 3.5 shows a flow diagram of the Clark-Hau algorithm.

3.3 SPECTRAL FACTORISATION TECHNIQUES

3.3.1. The Toeplitz method

The method described here is based on the well known least-error-energy (LEE) deconvolution filtering-a time domain approach.

Starting with the autocorrelation sequence $R(z)$, given by Equation

$$R(z) = Y(z)Y(1/z) \tag{3.3.1.1}$$

The Least energy error (LEE) finite length inverse filter $g$ of the filter with impulse response $y$ is determined. To achieve this, only the autocorrelation $r$ need be known and not the impulse response $y$ of the filter. The multiplication in the z domain implied by Equation 3.3.1.1 corresponds to convolution in the time domain given by
\[ r_k = y_k * y_{-k} = \sum_{m=0}^{g} y_m y_{-(k-m)} = \]
\[ = \sum_{m=0}^{g} y_m y_{m-k} \quad \text{3.3.1.1.a} \]

where \( k=0,1,\ldots,g \). This is equivalent to the autocorrelation \( r = \{r_n, r_{n-1}, \ldots, r_1, r_0, r_1, \ldots, r_n\} \) of the sequence \( y \) (with z transform, \( Y(z) \)).

A unit sample \( \delta_k \) is applied to the input of the discrete-time system of Figure 3.6. By minimising the error energy at the output the following set of Equations can be derived:

\[ \sum_{q=0}^{N} g_q r_{m-q} = \sum_{k=0}^{N} \delta_k y_{k-m} \quad \text{3.3.1.2} \]

for \( m=0,1,\ldots,N \); where \( N \) is the order of the inverse filter \( g \). Since \( H(z) \) is required to be minimum phase all \( h_i = 0 \) for \( i < 0 \). Therefore, the following matrix equation can be set up:

\[
\begin{bmatrix}
 r_0 & r_1 & \cdots & r_n \\
 r_1 & r_0 & \cdots & r_n \\
 \vdots & \vdots & \ddots & \vdots \\
 r_n & \cdots & \vdots & r_0 \\
 0 & r_n & \cdots & r_1 \\
\end{bmatrix}
\begin{bmatrix}
 g_0 \\
 g_1 \\
 \vdots \\
 g_{n-1} \\
 0 \\
\end{bmatrix}
= 
\begin{bmatrix}
 1 \\
 0 \\
 \vdots \\
 0 \\
 0 \\
\end{bmatrix}
\rightarrow 
\begin{bmatrix}
 y_0 \\
 y_1 \\
 \vdots \\
 y_{n-1} \\
 y_n \\
\end{bmatrix}
\quad \text{3.3.1.3}
\]

for \( N>n \).
The correlation matrix (Equation 3.3.1.3) has Toeplitz structure, so that $g$ can be evaluated very efficiently by the Levinson algorithm [5]. Levinson's recursion formulae for solving Equation 3.10.3 are given below (See also appendix A3).

$$u = \frac{k-1}{\sum_{q=0}^{k} g_q f_{k-q}}$$

$$\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_k \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 - u^2 \\ 0 \end{bmatrix} \begin{bmatrix} \frac{g_0}{1 - u^2} \\ \vdots \\ \frac{g_{k-1}}{1 - u^2} \\ 0 \end{bmatrix}$$

$$k = k + 1 \text{ return unless } k \geq N$$

with initial conditions $k=1$ and $g_0 = 1/r_0$. A modification of the Levinson recursion, known as Durbin's recursion, exploits the simplified form of the right-hand side of Equation 3.3.1.3 and can be used to solve the set even more efficiently. (Appendix 3, A3.1-A3.2). Now, if the LEE inverse filter $g$ of order $N$ is determined, only one additional recursion step has to be performed to obtain the LEE inverse filter of order $N+1$. From Figure 3.6 it can be seen that the error energy is zero if $g$ is the inverse of $h$, i.e. $G(z)=1/Y(z)$. With increasing order $N$, $g$ approximates the inverse of $h$ more and more accurately. In Equation 3.3.1.4 the first $n+1$ terms of the division

$$\frac{1}{g_0 + g_1 z^{-1} + \ldots} = y'_0 + y'_1 z^{-1} + \ldots + y_n z^{-n} + \varepsilon(z)$$

are computed.
The resulting coefficients \( y_i^*, i=0,1,...,n \) converge (within a scale factor) to the coefficients \( y_i \), \( i=0,1,...,n \), of the minimum phase spectral factor.

The final scaling is then performed by dividing \( h^* \) by \( (y^*y^*/r_n)^{1/2} \).

### 3.3.2 Bauer's Method

In this method, the infinite Toeplitz matrix \([R]\) is first formed by the given coefficients of the input polynomial \( Y(z) \), by a process of linear multiplication of \( Y(z) \) by \( Y(1/z) \). A LU decomposition (see Appendix A3.5) is then performed on the Toeplitz matrix \([R]\).

\[
R = \begin{bmatrix}
r_0 & r_1 & \cdots & r_n & 0 \\
r_1 & r_0 & \cdots & r_n & \\
\vdots & \vdots & \ddots & \vdots & \\
r_n & \vdots & \ddots & \vdots & \\
0 & \vdots & \ddots & \vdots & \\
\end{bmatrix}
\]
If the resulting triangular matrices are chosen to have identical diagonal elements, the decomposition is unique and the nonzero elements $y_i^{(k)}$, $i=0,1,...,n$ of the $k$th row of the upper triangular matrix, or the $k$th column of the lower triangular matrix respectively, can be calculated recursively:

Set:

$$y_0^{(k)} = \left( r_0 - \sum_{m=1}^{n} (y_m^{k-m})^2 \right)^{1/2} \quad 3.3.2.2$$

$$y_i^{(k)} = \left( r_i - \sum_{m=1}^{n-1} y_m^{(k-m)} \right) / y_0^{(k)} \quad 3.3.2.3$$

with initial conditions $y_i^{(m)} = 0$ for $m=0,-1,...,(n-1)$. With increasing $k$ the elements $y_i^{(m)}$, $i=0,1,...,n$ converge to the coefficients of the minimum phase factor $Y(z)$.

3.3.3 Gram-Schmidt orthogonalization method

This algorithm makes use of the orthogonalization process [6,7] in order to adjust the prefilter and at the same time estimates the sampled impulse response of the channel and filter. The main advantage of this scheme, is that there is not a need to evaluate the roots of the channel, thereby
eliminating the need for a separate root finding algorithm. The algorithm can be implemented as a
filter with n+1 taps, shown in Figure 3.7. The signals shown are those present at the time instant
t = iT. Each square marked T is here a storage element that holds the corresponding sample
value. The extreme left hand sample value at the input to the maximum likelihood detector is
that received at time t=iT, the next sample value is that received at time t=(i-1)T and so on,
each storage element introducing a delay of T seconds. A brief explanation of the filter
implemented will now be presented. A matrix Y is formed from the channel sampled impulse
response as follows: Firstly, a row vector is formed by preceding the channel sampled impulse
response by a set of zeros, such that the total number of components of the row vector is n+1,
where n+1 is the number of taps of the filter,

\[
[0 \ldots 0 \ y_{i,g} \ y_{i,g-1} \ldots y_{i,1} \ y_{i,0}]
\]

The matrix Y is then formed by shifting the row vector to the left until the first non-zero
component in the row is the first component in the row, and then continue shifting to the left so
that the non-zero components fall off the edge until there is only one non-zero component in the
row. Similarly the row vector is shifted to the right until there is only one non-zero component
left in the row.
This matrix can be shown diagramatically in Figure 3.8. The matrix is an n+g+1 by n+1
component matrix, in other words it has n+g+1 rows and n+1 columns. The matrix can also be
considered as consisting of n+g+1 row vectors. If the row vectors are considered,

\[
[y_{i,g} \ y_{i,g-1} \ldots y_{i,1} \ y_{i,0} \ 0 \ldots 0]
\]

\[
[0 \ldots 0 \ y_{i,g} \ y_{i,g-1} \ldots y_{i,1} \ y_{i,0}]
\]

\[
[0 \ldots 0 \ y_{i,g} \ y_{i,g-1} \ldots y_{i,1} \ y_{i,0}]
\]
Let the last row vector $Y_0$, and the first as $Y_{n-g}$, then $Y_0, Y_1, ..., Y_{n-g}$ are a set of vectors which are orthogonal to each other. The idea is to obtain a set of orthogonal vectors $Z_0, Z_1, ..., Z_{n-g}$. These are obtained by using the Gram-Schmidt orthogonalization process. The procedure is as follows:

$$Z_{n-1} = Y_{n-1}$$

$$Z_{n-2} = Y_{n-2} - \frac{Y_{n-2} Z_{n-g} Z_{n-g}}{Z_{n-g} Z_{n-g}}$$

$$Z_0 = Y_0 \frac{Z_1 Z_1 - ... - Y_0 Z_{n-g} Z_{n-g}}{Z_{n-g} Z_{n-g}}$$

$Z_0$ being orthogonal to $Z_1, ..., Z_{n-g}$ hence $Z_0$ is orthogonal to the space spanned by $Z_1, Z_2, ..., Z_{n-g}$ and so to every vector in the space, therefore $Z_0$ is orthogonal to $Y_1, Y_2, ..., Y_{n-g}$ and their products are zero.

The tap gains of the filter are given by the complex conjugate of $Z_0$. So the signals $Y_1, Y_2, ..., Y_{n-g}$ are lying on the filter taps, the output would be a maximum. From the properties of the Gram-Schmidt orthogonalization process, the magnitude of the first non-zero component at the output is maximised subject to the exact cancellation of the precursors, i.e. previous symbols.

$$YD_1^T = YZ_0 = YZ_0$$

$YZ_0$ in equation 3.3.3.5 gives the sampled impulse of the channel and filter.
3.3.4 N.I.T method

This novel iterative technique developed by the author is different from the algorithms presented so far because it operates directly on the sampled impulse response of the channel (or more accurately on an estimate of it), without the need for locating any roots, thus enabling the prefilter to be adjusted in a fast and accurate way. This algorithm adjusts the prefilter and at the same time forces the sampled impulse response of the channel and filter to be minimum phase.

3.3.4. The basic NIT algorithm

Let \( Y(z) \) be the transform of the sampled impulse response of the linear baseband channel, given by

\[
Y(z) = y_0 + y_1 z^{-1} + ... + y_g z^{-g} \quad 3.3.4.1
\]

Step 1 First form a vector \( F = [a_0, a_1, ..., a_g] \) where its elements are the elements of the original sampled impulse response \( Y(z) \), but now sorted in an ascending order. Let \( F(z) \) be the z-transform of that vector \( F \), given by

\[
F(z) = a_0 + a_1 z^{-1} + ... + a_g z^{-g} \quad 3.3.4.2
\]

Step 2 Now obtain a new polynomial \( F_m(z) \), by reversing the order of the coefficients of the polynomial \( F(z) \)

* In fact the method bears the author's initials, N I T.
\[ F_m(z) = a_g + a_{g-1}z^{-1} + \ldots + a_0z^{-r} \]  

Step 3 Next perform the long division of \( Y(z) \) by \( F_m(z) \) which results in a convergent series (since \( F_m(z) \) is set to be minimum phase) given by

\[ D(z) = \frac{Y(z)}{F_m(z)} = d_0 + d_1z^{-1} + \ldots \]  

This resulting infinite series in \( z^{-1} \) is then truncated to its first \( k \) terms, where \( k = g \) and \( g \) equals the length of the original input sequence \( Y(z) \) yielding \( D_1(z) \)

\[ D_1(z) = d_0 + d_1z^{-1} + \ldots + d_kz^{-k} \]  

Now by reversing the order of the coefficients of the polynomial \( D_1(z) \) and taking the conjugate the polynomial \( D_m(z) \) is formed given by

\[ D_m(z) = d_k + d_{k-1}z^{-1} + \ldots + d_0z^{-k} \]  

Step 5 Finally form the polynomial \( C(z) \) by convolving the polynomials \( D_m(z) \) and \( Y(z) \)

\[ C(z) = D_m(z)Y(z) = c_0 + c_1z^{-1} + \ldots + c_{k+g}(k+g-1) \]  

\[ = c_0 + c_1z^{-1} + c_{(2g)}z^{-(2g-1)} \]
By ignoring the first \(g-1\) coefficients, (since their magnitude is very small when compared to the magnitude of the \(g^{th}\) coefficient), the polynomial \(C_m(z)\) is obtained given by

\[C_m(z) = c_g z^g + c_{g+1} z^{g+1} + \ldots + c_{2g} z^{2g-1}\] 3.3.4.8

which is a first approximation to the ideal minimum phase \(Y_{\text{min}}(z)\) of the input polynomial \(Y(z)\). Now, if \(C_m(z)\) is considered (equation 3.3.4.8) in place of \(F_m(z)\) (equation 3.3.4.3) and steps 3-5 are repeated a sufficient number of times (depending on the length of the input polynomial \(Y(z)\)), a very accurate minimum phase impulse response \(C_m(z)\) can be obtained approaching the ideal one \(Y_m(z)\). The algorithm is best shown in Figure 3.9.

The starting vector for the algorithm, \(F_m(z)\) in Equation 3.3.4.3 'has been set to be minimum phase'. This is based on the following conjecture:

**NIT conjecture** A sequence of real numbers can be set to be minimum phase by reordering the numbers in a series of decreasing magnitudes. In the case of complex numbers this holds true only if the magnitude is considered.

Other vectors could have served as the starting sequence for the algorithm as long as they form minimum phase sequences, so that the division implied by Equation 3.3.4.4 results in a convergent series. A lot of starting vectors were tested. A pretty obvious choice (and a quite successful one) could be the vector \([1 \ 0 \ 0 \ 0 \ldots]\). This vector clearly is a minimum phase sequence and results to quite fast and accurate operation for the NIT algorithm. However, a lot of information relating to the original sequence is absent from this vector. An intuitive approach dictates that the original sequence should be used in some form as starting vector, since all the information lies within that sequence. This was proved to be right since the second best starting

---

* Professor S.Y. Kung of Princeton University offered to fetch a proof.
vector for the algorithm (apart from the one in Equation 3.3.4.3) was the vector formed when the numbers in the original sequence were ordered in a series of decreasing magnitudes. Although in some cases this resulted in sequences that they were not true minimum phase sequences, it always gave better results than for example, the vector $[1 \ 0 \ 0 \ldots]$ (which incidentally was the 'best' vector from all the other minimum phase sequences not related to the original $Y(z)$). Note, that the sequences resulting by rearranging the original ones in a series of numbers with decreasing magnitudes were always 'near minimum phase' sequences (See Appendix 4).

In order to increase the operational speed of the NIT method for the cases where the channel introduces severe group delay distortion, some form of prediction must be incorporated into the algorithm. Least-squares fading memory prediction was used here, employing polynomial filters which led to a modified version of the basic NIT algorithm.

### a. Modified NIT.a algorithm

**Step 1** Consider the basic algorithm (steps 1-5). If this is iterated a number of times, say $N$, at the end of this process a polynomial $D_m(z)$ (step 4—equation 3.3.4.6) is obtained. Also obtained is a polynomial $C_m(z)$ (step 5 equation 3.3.4.8) which is the $N$th approximation to the ideal minimum phase $Y_{\text{min}}(z)$ of the input polynomial $Y(z)$. Also rename polynomial $D_m(z)$ as $D_{\text{old}}(z)$.

**Step 2** Now use $C_m(z)$ in place of $F_m(z)$ and repeat steps 3-5 once i.e. $N=1$. This time a polynomial $D_m(z)$ (step 4-equation 3.3.4.6) is obtained; call this polynomial $D_{\text{new}}(z)$.

**Step 3** Next incorporate a degree zero polynomial predictor filter [8,9,10] by first forming the error

$$E(z) = D_{\text{new}}(z) - D_{\text{old}}(z)$$

and then the one step prediction

$$D_{\text{pred}}(z) = D_{\text{old}}(z) + kE(z)$$

where $k$ is a constant.
**Step 5** Finally the polynomial $Y(z)$ is convolved with $D_{pred}(z)$ (in place of $D_m(z)$ of the basic algorithm -step 5) thus obtaining an accurate $(N+1)\text{th}$ approximation to the ideal minimum phase $Y_{\text{min}}(z)$. In this way a very significant reduction in the number of iterations $N$ can be achieved without affecting the accuracy of the method.

**b. Modified NIT.b algorithm**

Consider the basic NIT algorithm (section 3.3.4). The new element in this version of the original algorithm is that the polynomial $F_m(z)$ given by Equation 3.3.4.3 now truncated by dropping the lower degree coefficients to yield the new polynomial $F_{MT}(z)$.

$$F_{MT}(z) = a_k + a_{k-1}z^{-1} + \ldots + a_0z^{-k} \quad 3.3.4.11$$

The new truncated polynomial $F_{MT}(z)$ has say $k$ components where $K < g$ ($g$ being the number of the components of the original input polynomial $Y(z)$).

**Step 3** Next perform the long division of $Y(z)$ by $F_{MT}(z)$ which results in a convergent series (since $F_{MT}(z)$ is set to be minimum phase) given by

$$D_{MT} = \frac{Y(z)}{F_{MT}(z)} = d_0 + d_1z^{-1} + \ldots \quad 3.3.4.12$$

This resulting infinite series in $z^{-1}$ is then truncated to its first $k$ terms, where $k=g$ and $g$ equals the length of the original input sequence $Y(z)$ yielding $D_t(z)$

$$D_t(z) = d_0 + d_1z^{-1} + \ldots + d_kz^{-k} \quad 3.3.4.13$$
Now by reversing the order of the coefficients of the polynomial \( D_t(z) \) and taking the conjugate the polynomial \( D_m(z) \) is formed given by

\[
D_m(z) = \bar{d}_k + \bar{d}_{k-1} z^{-1} + \ldots + \bar{d}_1 z^{-k}
\]  \hspace{1cm} 3.3.4.14

Step 5 Finally form the polynomial \( C(z) \) by convolving the polynomials \( D_m(z) \) and \( Y(z) \)

\[
C(z) = D_m(z) Y(z) = c_0 + c_1 z^{-1} + \ldots + c_{k+g} z^{-(k+g-1)}
\]  \hspace{1cm} 3.3.4.15

\[
= c_0 + c_1 z^{-1} + c_{(2g)} z^{-(2g-1)}
\]

By ignoring the first \( g-1 \) coefficients, (since their magnitude is very small when compared to the magnitude of the \( g^{th} \) coefficient), the polynomial \( C_m(z) \) is obtained given by

\[
C_m(z) = c_g z^g + c_{g+1} z^{g+1} + \ldots + c_{2g} z^{(2g-1)}
\]  \hspace{1cm} 3.3.4.16

which is a first approximation to the ideal minimum phase \( Y_{\text{min}}(z) \) of the input polynomial \( Y(z) \). Now, if \( C_m(z) \) is considered (equation 3.3.4.16) in place of \( F_m(z) \) (equation 3.3.4.11) and steps 3-5 are repeated a sufficient number of times (depending on the length of the input polynomial \( Y(z) \)), a very accurate minimum phase impulse response \( C_m(z) \) can be obtained approaching the ideal one \( Y_m(z) \).
3.3.5 CEPSTRUM

Cepstrum is a paraphrasis of the word spectrum since “we find ourselves operating on the frequency side in ways customary on the time side [11]”. The cepstrum method is a non-linear analysis technique which has found application in a number of fields, including echo detection [12], speech analysis [13], geophysical data processing and others[14]. This method relies on the computation of the cepstrum of a signal. The cepstrum is defined as the inverse Fourier transform of the logarithm of the Fourier transform of the input sequence, or else the log-power spectrum of the input sequence. The cepstrum obtained in this manner requires the use of the complex logarithm and complex Fourier transforms and is referred to as the complex cepstrum. An alternative form of the cepstrum is based on the use of the logarithm of the magnitude rather than the complex logarithm of the Fourier transform. This definition of cepstrum avoids ambiguity problems, such as uniqueness and continuity, in the definition of the complex logarithm. This form of cepstrum is often referred to as the real cepstrum (as opposed to the complex cepstrum). Note however that the complex cepstrum is real for a real input. In this analysis the term cepstrum will be used to denote the real cepstrum. Computation of the cepstrum using only the magnitude of the Fourier transform followed by appropriate windowing in the cepstral domain and the inverse cepstral transformation provides a means of transforming a signal to its minimum phase equivalent [15]. Assume that Y(z) is the z-transform of a finite-length sequence y(n).

The method in an algorithmic form is as follows:

**Step 1** Compute the magnitude of the Discrete Fourier Transform (DFT) of the input sequence y(n).

\[
A = |\text{DFT}(y(n))|
\]
Step 2 Take the logarithm of the quantity A in Step 1

\[ B = \log(A) \] 3.3.5.2

Step 3 Calculate the Inverse Discrete Fourier Transform (IDFT) of B in Step 2 to obtain the cepstrum.

\[ \hat{y}[n] = \text{real}\{\text{IDFT}(B)\} \] 3.3.5.3

In the second part of the algorithm a minimum phase reconstruction \( y_m(n) \) of the input sequence \( y(n) \) is obtained. The real cepstrum is windowed by

\[ w[n] = \begin{cases} 0 & n < 0 \\ 1 & n = 0 \\ 2 & n > 0 \end{cases} \] 3.3.5.4

The windowing operation results in mapping the maximum-phase zeros and poles (i.e. the zeros and poles that lie outside the unit circle), to their conjugate symmetric counterparts within the unit circle of the z-plane.

Step 4 The inverse cepstral mapping is computed as

\[ \hat{z}[n] = \hat{y}[n]w[n] \] 3.3.5.5

The mapping in Equation 3.3.5.5 can be evaluated as follows

Step 5 The Discrete Fourier Transform (DFT) of the quantity \( \hat{z}[n] \) is now computed

\[ \hat{A} = \text{DFT}(\hat{z}[n]) \] 3.3.5.6

Step 6 This step involves the complex exponentiation of \( \hat{A} \) in equation 3.12.6
\( \hat{E} = \exp(\hat{A}) \) \hspace{1cm} 3.3.5.7

**Step 7** By taking the Inverse Discrete Fourier Transform of \( \hat{E} \) the minimum-phase reconstructed sequence \( y_m \) can be obtained

\[ y_m = \text{real}(\text{IDFT}(\hat{E})) \] \hspace{1cm} 3.3.5.8

The above described process when implemented in software can be seen as operating in two stages; Stage one is the forward mapping which involves a Fast Fourier Transform (FFT), the determination of the log-magnitude spectrum and an Inverse Fast Fourier Transform (IFFT). This is described in Steps 1-3 (Equivalently Equations 3.3.5.1-3). Equations 3.3.5.3-3.3.5.6 constitute a Hilbert Transform pair. (See also Appendix A3.4.1)

Stage two is the inverse mapping which involves a Fast Fourier Transform, a complex exponentiation, and an Inverse Fast Fourier Transform. The whole process can be better understood with the aid of a flow diagram (Figure 3.10).

In order to gain further insight into this particular process the following section is presented (See also Appendix A3.3).

**The Fejér-Riesz theorem [15,16]**

Given a function with z transform

\[ W(z) = \sum_{n=-m}^{m} w_n z^{-n} \] \hspace{1cm} 3.3.5.9

\[ W(e^{j\omega T}) \geq 0 \]

then a function \( Y(z) \) can be found
such that

$$W(e^{j\omega T}) = |Y(e^{j\omega T})|^2$$ \hspace{1cm} 3.3.5.11

The function $Y(z)$ is unique only if all its roots are in the circle $|z| \leq 1$ i.e. the function $Y(z)$ is minimum phase. (For the proof of the theorem see Appendix 3 section A3.4).

The cepstrum method has been extended to accommodate complex signals, since the sampled impulse responses assumed throughout this thesis are complex signals.

Equations 3.3.5.3 and 3.3.5.8 now have imaginary components, given by

$$\hat{y}[n] = \text{IDFT}(B)$$ \hspace{1cm} 3.3.5.12

$$y_m = \text{IDFT}(\hat{E})$$ \hspace{1cm} 3.3.5.13

Thus resulting in a complex minimum phase sequence.

The quantity $y_m$ in Equation 3.3.5.13 does not represent the cepstrum given by equation 3.3.5.3, since the cepstrum of a real or complex sequence is by definition a sequence of real numbers. To allow for the complex result $y_m$ in equation 3.3.5.13 to hold true, equations 3.3.5.3-6 are now replaced by the equivalent Hilbert Transform relations for complex sequences (See Appendix A3.4.1).

The second stage of the algorithm is to provide the set of coefficients necessary for the adaptive adjustment of the predetection filter. This can be accomplished by employing techniques similar to the one described in Section 3.2.4 or Section 3.3.4. To derive the required set of coefficients with z transform $D(z)$ the original sequence with z transform $Y(z)$ is divided by the minimum phase equivalent sequence $y_m$ with z transform $Y_m(z)$ in equation 3.3.5.14.
\[ D(z) = \frac{Y(z)}{Y_m(z)} = d_0 + d_1 z^{-1} + \ldots \] 3.3.5.14

Thus Equation 3.3.5.14 yields an infinite series of numbers which forms a maximum phase sequence, since they are ordered in ascending order in terms of their magnitudes. The division in equation 3.3.5.14 is valid (i.e. results in a convergent series, since the denominator \( y_m \) is a minimum phase sequence). The maximum phase sequence with z transform \( D(z) \) is next reversed and then truncated to its first \( k \) terms, where \( k \) equals the number of filter taps (this number depends on the distortion introduced by each particular channel). The resulting sequence is given by

\[ D_T(z) = d_0 + d_1 z^{-1} + \ldots + d_k z^{-k} \] 3.3.5.15

The conjugate of the sequence in Equation 3.12.15 given by

\[ \tilde{D}_m(z) = \tilde{d}_k + \tilde{d}_{k-1} z^{-1} + \ldots + \tilde{d}_0 z^{-k} \] 3.3.5.16

is the wanted set of filter coefficients. Now, having set up the prefilter, the impulse response required by the detector i.e. the combined impulse response of the channel and prefilter can be found by convolving \( Y(z) \) with \( D_m(z) \). Thus,

\[ C(z) = D_m(z)Y(z) = c_0 + c_1 z^{-1} + \ldots = +c_{k+g} z^{-(k+g-1)} \] 3.3.5.17
The first terms of the polynomial \( C(z) \) are small in terms of their magnitude, so if the number of taps \( D_m(z) \) is \( k \), then the first \( k-1 \) terms of \( C(z) \) can be ignored thus yielding

\[
C_m(z) = c_0 z^{-z} + c_{g+1} z^{-z+1} + \ldots + c_{k+2} z^{-(k+2-1)}
\tag{3.3.5.18}
\]

which is the combined sampled impulse response of the channel and filter ignoring the delay introduced by the filter. Note that Equations 3.3.5.14-18 are same with the Equations 3.3.4.4-8 (Section 3.3.4), with the exception of replacing \( F_m(z) \) (Equation 3.3.4.4) with \( Y_m(z) \) in equation 3.3.5.14.

The Cepstrum algorithm concludes Chapter 3, which should be viewed in conjunction with Chapter 4, where the proposed algorithms are simulated and their performance evaluated over different practical communication channels in order to select the one which fulfils the three requirements described earlier in Section 3.1 of this Chapter.
Figure 3.1 The QR method

INPUT: GENERAL MATRIX

HOUSEHOLDER TRANSFORMATION

HESSENBerg TRANSFORMATION

APPLY QR METHOD
Figure 3.2 One Tap Feedback Filter

\[ \{y_g \ y_{g-1} \ldots \ y_0\} \rightarrow T \rightarrow \{e_{i,h}\} \]

Figure 3.3 Two Tap Feedforward Filter

\[ \sum_i s_i \delta(t - iT) \rightarrow \Sigma \rightarrow T \rightarrow \{e_{i,h}\} \rightarrow x \rightarrow -\lambda_i \]

\[ \beta^*_1 \rightarrow x \]

\[ F \]
Figure 3.4 Adjustment of the Receiver Prefilter

Linear Baseband Channel

Transmitter
Lowpass Filter

Telephone circuit/HF/mobile links

Receiver Lowpass Filter

AWGN

Near Maximum Likelihood Detector

prefilter

Sampler t=\(iT\)

\(r_i\)

\(r(t)\)

Estimate of the Minimum Phase Sampled Impulse Response

Adjustment Algorithm

Prefilter Taps

Estimate of Channel

Training Sequence

Linear Feedforward Channel Estimator
Figure 3.5 Flow Diagram of the Clark-Hau Algorithm

Start

First Starting Point

\( i = 0 \)

Pass \( y \) through the Feedback Filter

Calculate \( e_i \)

Calculate \( \left| \frac{e_i}{e_{i-1}} \right| \)

\( \text{is} \ | \frac{e_i}{e_{i-1}} | < d \)

Yes

No

Calculate \( \lambda_{i+1} \)

\( \text{is} \ | \lambda_i | > 1 \)

Yes

No

\( i = i+1 \)

\( i = 40 \)

Next Starting Point

All Points Used?

End
Figure 3.6. Least-error energy filtering

Figure 3.7 Predetection Filter Implementation employing GRAM-SCHMIDT Orthogonalisation
Figure 3.8 Matrix $Y$ for the Gram-Schmidt method

$$
\begin{array}{cccccccccc}
y_{1,0} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
0 & 0 & y_{i,g} & \cdots & y_{i,2} & y_{i,1} & y_{i,0} & \cdots & 0 \\
0 & 0 & 0 & y_{i,g} & \cdots & y_{i,2} & y_{i,1} & y_{i,0} & 0 & 0 \\
0 & 0 & 0 & 0 & y_{i,g} & \cdots & y_{i,2} & y_{i,1} & y_{i,0} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & y_{i,g} & \cdots & y_{i,2} & y_{i,1} & y_{i,0} & n+1 \\
\end{array}
$$

Precursors 2

Precursors 1

Postcursors

Taps
Figure 3.9 The NIT algorithm

Initialisation

\[ \frac{Y}{F_m} = D \quad \rightarrow \quad Y = Y \cdot D \quad F_m = Y \]

Infinite Recursion

\[ \frac{Y}{Y \cdot D} = \frac{Y}{Y \cdot D} = \frac{Y}{Y \cdot D} = \cdots \]
Figure 3.10 Cepstrum Algorithm

Forward Mapping

1. FFT input sequence
2. LOG-MAG Computation; Phase set to zero
3. IFFT

Inverse Mapping

1. FFT cepstral sequence
2. Exponentiate LOG SPECTRUM
3. IFFT
REFERENCES


CHAPTER 4

ADJUSTMENT ALGORITHMS OVER THE TELEPHONE NETWORK

4.1 Introduction

Chapter 4 in a way concludes the previous Chapter since it contains the results of the tests devised to evaluate the algorithms presented in the previous chapter. These adjustment algorithms are tested over the simulated telephone network. It is only natural therefore, that the topic of transmitting data over the telephone circuits together with the transmission impairments experienced by the network is presented first in Section 4.2. Taking the above concepts into consideration a simulation model of a data transmission system is then developed in Section 4.3. This forms the platform over which eight different channels (representing various degrees of distortion) are modelled. These actual channels are subsequently used as a means of testing the adjustment algorithms proposed in Chapter 3 and results of these tests are presented in Section 4.4. Finally, a summary of the most important results is given in graphic form in this final Section as an aid to distinguish, through the plethora of the methods and results, which is the best method for real time implementation.

4.2 Transmission of digital data over the telephone network: Telephone circuits

Telephone circuits are an arrangement with interconnection whereby the communication of speech or data can be carried between any two points. They are voice band channels, with a nominal bandwidth of 300-3400 Hz [1-19]. It may be privately or publicly owned. Private circuits can be considered as point to point communication which are permanently or on a part time basis rented by one or more subscribers [20-24].
They are not connected through any of the switches in the exchange nor to the exchange or repeater station's battery supplies. Large organisations such as banks and railway authorities may have their own network of lines to meet their own demands [21]. The public switched telephone network (PSTN) is necessary to connect any given subscriber to another at a telephone exchange (which may be manually or automatically operated). The choice between using private circuits or using the PSTN for data transmission must be made by careful consideration of factors, such as cost, availability, speed of working and transmission performance.

A PSTN connection is established by dialling the telephone number of the distant data terminal and the route used for a particular call is a random choice from a large number of different possible routing, including various combinations of audio-frequency cables and multi-channel systems [21-24]. For data transmission, private-leased circuits have the following advantages over the PSTN [20-24].

1. The bandwidth of private circuits tends to be around 300-3000 Hz, which can be adjusted to give a good performance over this bandwidth, whereas the bandwidth of the PSTN is more restricted.
2. Private circuits are less noisy than PSTN channels because of the switching equipment used in the telephone exchanges.
3. Efficient performance due to exclusive use of the circuits is obtained.
4. The link can be adjusted to have the optimum performance, where high speed and more reliable transmission are possible.
5. Full-duplex operation is available at higher bit rates.
Finally, the cost of the permanently private circuit may be relatively high especially if
there are insufficient data traffic on the line. In practice, most private data networks
consist of a combination of both leased and PSTN link.

Voice-frequency data circuits can also be of any length. A long telephone circuit of
several miles in length, may contain the following arrangement of individual links:
unloaded audio, loaded audio, PCM and carrier links.

Unloaded audio links are generally very short (of the order of 3 miles) [25]. They
comprise a pair of wires with impedance 600 Ω. Because of their short length, they
have a good frequency response, with some attenuation distortion and negligible delay
distortion over the voice-frequency band. It is the (high) attenuation, (which increases
as the length of the link increases), that prevents the use of long unloaded audio links.
In many situations it is desirable to extend the length of the links beyond the limit of 3
miles. Common methods to attain longer links without exceeding loss limits are the
following [20-24].

1. Increase the conductor diameter
2. Use amplifiers
3. Use inductive loading.

Loaded audio links may be very much longer (up to about a hundred miles). Loading a
particular voice pair link consists of inserting series inductance (often 44 or 88 mH)
into the links at fixed intervals (typically 2000 yards). Adding load coils tends to
increase the velocity of propagation and increase the impedance. Over the centre of
the voice-frequency band the attenuation distortion decreases due to the presence of
the loading coils. In return the attenuation and group delay distortion will increase
over the higher frequencies. This increases as the length of the loaded audio link
increases, especially at the high frequency end of the voiceband [20-24].
The telephone network throughout the world is currently undergoing rapid changes, with the replacement of analogue links by digital links, such as pulse code modulation (PCM) and adaptive differential pulse code modulation (ADPCM) links [20-24]. The fourth type of link is the carrier link which can be very much longer than loaded audio links.

A process of single sideband suppressed carrier amplitude modulation is used here to shift the signal frequency band to higher frequencies for transmission over a wideband channel. The modulation process at the transmitter first multiplies the voice-frequency signal by a sine wave with the appropriate frequency, to give a double sideband signal centred on this frequency. The sideband required for transmission is then isolated by means of a filter. The process of linear demodulation first multiplies the received single-sideband signal by a sine wave with ideally the same frequency as that used at the transmitter, and the required voice-frequency signal is then isolated by means of a filter.

The filters involved in the modulation and demodulation in a carrier link have an effect on the demodulated voice-frequency signal, at the output of the link, equivalent to that of a high-pass filter with a cut-off frequency in the range 100-300 Hz. They also restrict the high-frequency components of the demodulated signal, but to a much less significant degree [25].

Because of the high cost of line plant it is desirable to use a line to carry more than one data link (multi-channel) by using a multiplexer. With frequency division multiplexing, (FDM) each data channel is frequency translated to a different part of the available frequency spectrum. This combination of modulation and frequency division multiplexing is the basis of the multi-channel carrier telephony systems which operate over carrier pairs, coaxial cable, microwave radio relay systems, satellite systems and HF radio links.
The correct operation of a multi-channel system, a carrier needs to be re-inserted at the receiver, at the correct frequency. If the sine wave carriers, used for modulating and demodulating the signal in a carrier link, do not have exactly the same frequency, the frequency spectrum of the signal at the output of the carrier link is shifted by an amount equal to the difference between the two carrier frequencies, but the shape of the spectrum remains unchanged. This resultant frequency offset over carrier links may have a value up to $\pm 5$ Hz, but usually lies within the range $\pm 1$ Hz. This leads to signal impairments rather than pure signal distortion.

4.2.1 Attenuation and delay distortions over telephone circuits (Transmission Impairments)

An important characteristic of telephone circuits is the wide range of different types of distortion and noise that may be experienced, together with the widely varying severities of the different effects. This is caused by the fact that a telephone circuit is normally made up of several different links connected in cascade, each link having its own particular properties and being itself selected from possibly a large number of links of the same basic type [25]. The resulting characteristics of any given telephone circuit are not, therefore, accurately predictable, and, from time to time, a combination of severe attenuation distortion and severe group-delay distortion occur in the frequency characteristic of a circuit [26].

The attenuation and group delay characteristics are the most widely studied characteristics of telephone connections and can be defined as the variation of the attenuation and group delay with frequency [26]. A significant source of these distortions in a telephone circuit is multipath propagation. This results from the reflections at two or more points of mismatch in the circuit. Multipath propagation causes time dispersion of the received signal elements. Each element now constitutes a main pulse together with several echoes of this pulse that are dispersed in time.
The received signal elements therefore overlap with each other, and so, any sample of the received waveform is a function of several elements (and hence of the data symbol carried by these elements).

Figures A4.1 and A4.3 (Appendix 4) show the attenuation-frequency and group-delay characteristic respectively, of a typical voice-frequency channel. Since frequencies below 300 Hz and above 3000 Hz are not required for the intelligible reception of speech, the attenuation of the channel may increase rapidly at frequencies below 300 Hz and above 3000 Hz. There is a very wide spread in the frequency characteristics of different telephone circuits. [26]. Channels introducing severe attenuation and group delay distortion may have sharper responses (Appendix 4, Figs A4.1-4).

Telephone circuits introduce different types of noise. The noise obtained on telephone circuits may be classified into two distinct groups: additive noise in which a waveform is added to the transmitted signal, and multiplicative noise in which the transmitted signal is modulated by the interfering waveform. The most important type of additive noise on telephone circuits is impulsive noise, and, over the public telephone network, impulsive noise tends to predominate over other types of noise (both additive and multiplicative). Impulsive noise is noncontinuous, consisting of irregular pulses or noise spikes of short duration and of relatively high amplitude. It is generated by variety of causes including the electrical/mechanical switches in the exchanges [43] and external electromagnetic disturbances, such as lightning. Because the shape of the impulsive noise varies widely from one telephone circuit to another, with duration extended over many adjacent signal elements, it makes the simulation of this noise by computer a difficult task. Recently, electrical/mechanical switches have been replaced by electronic switches, which greatly reduce the impulsive noise. Another type of additive noise is crosstalk. Crosstalk, an unwanted coupling between signal paths, can occur by electrical coupling between nearly twisted pair or, rarely, coaxial cable lines carrying multiple signals. Whereas over switched lines the majority of the noise is
additive, over private lines amplitude and frequency modulation effects can
predominate, and these must therefore also be considered. Amplitude and frequency
modulation noise are two forms of multiplicative noise, together with transient
interruptions, sudden level changes, frequency offsets, sudden phase changes and pure
phase jitters. Finally white noise, a steady background noise of low level and
relatively wide bandwidth, produces errors only at very low signal levels and is not
normally a significant source of errors [26].

The best examples of white noise are thermal noise and shot noise which have a
Gaussian amplitude distribution and are known as Gaussian white noise. Passing this
noise through bandlimited telephone circuits will produce bandlimited white or
coloured noise [26]. Strictly speaking, white noise has an infinite average power and,
as such, it is not physically realisable [26, 43-44]. However white noise lends itself
well to theoretical calculations, due to its convenient mathematical properties, and is
also easily produced in the laboratory over the required frequency range. Therefore is
a useful tool in system analysis [20,26,43-45]. Any two different samples of white
noise are uncorrelated and statistically independent if the white noise is Gaussian
distributed [26, 43-46].

4.3 Model of Data Transmission system over telephone channels

The model of the baseband system incorporating a telephone circuit is shown in figure
2.2.3 (Chapter 2). The data transmission system is a synchronous serial system and
may operate either with an 16-level QAM signal or else with a 64-level QAM signal.
In either case the QAM signal has a carrier frequency of 1800 Hz. Two possible baud
rates have been considered here. These are 2400 baud for the 16-level, giving a useful
transmission rate of 9600 and 3200 bauds for the 64-level QAM signal giving a useful
transmission rate of 19200 bit/s.
In the model, the transmitted signal consists of a stream of impulses which are modulated by the values of the data symbols \( \{s_i\} \) [see Chapter 2 (2.2)]. The impulses are a mathematical tool, used to make the analysis of the model easier. In reality a stream of square pulses would be used and the transfer function of the transmitter filter (Fig. 2.2.3) appropriately modified with \( \sin x/x \) correction. The lowpass filter and linear modulator at the transmitter, the transmission path and the linear demodulator at the receiver together form a linear baseband channel as shown in figure 2.2.3. The channel has a complex-valued impulse response \( y(t) \), which for practical purposes, has a finite duration and is time invariant, such that \( y(t-iT) \) is a time shifted version of \( y(t-jT) \), for any integers \( i \neq j \). There is the modulation interval (1/2400 or 1/3200 for the 16-level or the 64-level QAM signals respectively). The relationship between the resultant linear baseband channel and the bandpass transmission path is considered in great detail in Chapter 2. Although the telephone circuits introduce various types of additive and multiplicative noise, it is assumed here that the only noise introduced by the channel is stationary white Gaussian noise. The additive white Gaussian noise in Figure 2.2.3 is complex valued with zero mean and a constant two-sided power spectral density of \( 1/2 N_0 \) for each of its real and imaginary parts. The demodulated and filtered waveform is a complex-valued baseband signal \( r(t) \), with a bandwidth extending from about -1200 to 1200 Hz, where

\[
r(t) = \sum_i y(t-iT) + w(t)
\]

and \( y(t) \) is the resultant impulse response of the filters and telephone circuit, and \( w(t) \) is the noise waveform in \( r(t) \). The sampler in Figure 2.2.3 samples the received waveform \( r(t) \) once per data symbol, at time instants \( \{iT\} \), to give the received samples

\[
r_i = \sum_{h=0}^{\ell} s_i y_{i-h} + w_i
\]
where \( \{y_i\} \) are the components of the vector

\[
Y = [y_0, y_1, \ldots, y_n]
\]

which represents the resultant sampled impulse response of the linear baseband channel (Fig. 4.3.1) formed by the filters and the telephone circuit.

4.4 Performance of Adjustment Algorithms over the Telephone Network: Computer Simulation Tests and Results

4.4.1 Computer Simulation Tests

Computer Simulation tests were carried out to evaluate the performance of the adjustment algorithms (Chapter three) over different models of telephone circuits. Four different models of telephone circuits have been suggested to represent a wide range of telephone lines in the British PSTN. The sampled impulse responses of the linear baseband channel (Figure 4.3.1) have been derived for both the 16-point QAM signal operating at 9600 bit/s and the 64-point QAM signals operating at 19200 bit/s; these are designated as channels 1-4 and 5-8 respectively. Figures 4.4.1-4.4.4 represent the attenuation and group delay characteristics of the four telephone circuits. Telephone circuits 1 and 2 introduce typical levels of attenuation and group delay distortion. The telephone circuits in channel 3 and 4 are close to the typical worst circuits (in the PSTN) normally considered for the transmission of data at 9600 bit/s.

The telephone circuit in channel 3 is close to the standard network N6 and introduces severe group delay distortions as well as considerable attenuation distortions. The telephone circuit in channel 4 is close to the standard network N3 and introduces extremely severe group delay distortion [27-29]. Two different sets of equipment
ADJUSTMENT ALGORITHMS OVER THE TELEPHONE NETWORK

filters (Appendix 4- Section 3) have been suggested to use with the telephone circuits 1-4, to form channels 1-8. In the computation of the sampled impulse response of the linear baseband channel, the transfer function of a practical channel is often given in terms of its attenuation and group delay characteristics [30,31]. The attenuation and group delay characteristics of the channel are given by those of telephone circuit (Figures 4.4.1-4.4.4) and the equipment filter (Appendix 4 -Figures A4.3.1 and A4.3.2) in cascade. The sampled impulse responses of the linear baseband channels 1-4 can be computed according to Equations A4.2.1-A4.2.8 (Appendix 4- Section 2). Tables 4.1 and 4.2 show the sampled impulse responses of the linear baseband channels 1-4 and 5-8, respectively. Tables 4.3 and 4.4 show the minimum phase versions of the sampled impulse responses of the linear baseband channels by reflecting the roots that lie outside the unit circle in their conjugate reciprocal positions. These roots have been found ideally in this case by the QR method.

Extensive computer simulation tests have been carried out on eight different models of telephone channels designated as channels 1-8. The tests were designed to investigate the speed of operation and the eventual accuracy of the various adjustment algorithms and techniques presented in Chapter 3. A very important property for the root finding algorithms of Chapter 3 (Section 3.2) is the accurate location of the roots of \( Y(z) \) that lie outside the unit circle in the \( z \) plane. These roots, along with their absolute values are given in Tables T4.5-T4.8. The performance of the various algorithms has been measured using the parameters \( \psi_1, \psi_2, \psi_3 \) and \( \theta \) (Equations 2.7.9-2.7.12 in Chapter two). The parameters \( \psi_1, \psi_2, \psi_3 \) give the discrepancy between the actual sampled impulse response of the channel and adaptive filter and that assumed by the detector and together with the number of floating point operations have been used to evaluate both Root finding Algorithms and Spectral Factorisation techniques. Since the measurements \( \psi_1, \psi_2, \psi_3 \) appear throughout the Simulation tests it is now appropriate to attach some physical meaning to the Equations associated with them.
Thus $\psi_3$ shows how close is the minimum phase sequence that the algorithm achieves to the 'ideal' one (found by the QR method). This in the author's view, is the most significant measurement, since this indicates if the sequence (output from the algorithm) is minimum phase or not. The second parameter, $\psi_2$, shows how close is the combined sampled impulse response of the prefilter and channel to the one that is output from the algorithm. It is the second most important measurement, since it shows how well the algorithm adjusts the predetection filter. Finally, $\psi_1$, is the first part of the combined sampled impulse response of the filter and channel and represents the delay introduced by the filter. It should therefore, be ideally zero.

In the case of the Root finding Algorithms the parameter $\theta$ (Equation 2.7.12) has also been introduced. The parameter $\theta$ is a measure of the error in the estimate of the negative of the reciprocal of the roots found by any of the root finding algorithms, and those found ideally by the QR method. In the case of Spectral Factorisation techniques this parameter is meaningless, since no knowledge of roots is now necessary.

In addition the operation count (floating point operations) for the various algorithms has been taken into account, since the number of arithmetic operations required by any algorithm gives a good measure of its efficiency. Therefore, throughout the various simulation tests the number of real arithmetic operations required (additions multiplications and divisions) were computed and recorded in tabular form. Some conventions, symbols and comments relating to the Tables of this Section are given below.

1. To facilitate the reader to travel through the maze of Tables, the columns of the Tables for each method that are of special importance have been grey shaded, thus forming a 'path' that can lead to an easy way of comparing the different methods and drawing useful conclusions about their performance.

2. The Methods compared, are the root finding methods of Section 3.2 and the Spectral Factorisation Techniques of Section 3.3 in Chapter 3.

3. Parameters used for the comparison of the various methods are: $\psi_1$, $\psi_2$, $\psi_3$ (and $\theta$) and the number of floating point arithmetic operations (Flops).
4. Tables fall broadly in three categories:

a. Tables that indicate the Root accuracy (in dB) of the root finding methods, versus the number of iterations necessary to achieve that accuracy.

Note that the symbol F appearing on some of these tables signifies failure of the algorithm to converge to one or more wanted roots. Note also that all the starting points are used before failure of the algorithm is admitted. This is designated by the word any appearing in the relevant Tables. Finally, the symbol \( \psi_i \) used in the NIT method represents collectively \( \psi_2, \psi_3 \) due to the way that these two parameters are calculated in the NIT method.

b. Tables that show the performance of the various algorithms (in dB) making use of the parameters \( \psi_1, \psi_2 \) and \( \psi_3 \).

c. Tables that show the arithmetic complexity of each algorithm in terms of number of arithmetic operations. The numbers of addition/subtraction, multiplication and division indicate complex operations, while the numbers that show the total operational count signify real operations.

To conclude about the usefulness of an algorithm the three different categories of Tables should be taken into account.

Note that the Tables are named after the method which describe followed by the actual number of the Table. A list of the symbols and representations used to describe the Tables of this Section (and their relation to the Algorithms of Chapter 3) is given next

<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Chapter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR.</td>
<td>Newton-Raphson</td>
</tr>
<tr>
<td>NRm.</td>
<td>Newton-Raphson</td>
</tr>
<tr>
<td>L.</td>
<td>Laguerre method</td>
</tr>
<tr>
<td>CH.</td>
<td>Clark-Hau</td>
</tr>
<tr>
<td>LE.</td>
<td>Levinson</td>
</tr>
<tr>
<td>B.</td>
<td>Bauer</td>
</tr>
<tr>
<td>GS.</td>
<td>Gram-Schmidt</td>
</tr>
<tr>
<td>NIT.</td>
<td>NIT</td>
</tr>
<tr>
<td>NITT.</td>
<td>NIT.b</td>
</tr>
<tr>
<td>CP.</td>
<td>Cepstrum</td>
</tr>
</tbody>
</table>
All the above methods followed by the letter s indicate that the algorithms operate on a shortened sampled impulse response over the eight different telephone channels assumed in the simulation. The various methods and results of the tests are discussed next in some detail, in order to reveal their relative strengths and weaknesses.

1. **QR** This method sets the standard by which the rest of the methods of this Section have been evaluated. Since it is the most accurate and robust algorithm for the evaluation of the eigenvalues of a matrix it is used to find the roots of the Sampled Impulse Responses of the Channels 1-8, 'ideally' (very high accuracy) and thus provide the benchmark tests needed for analysing the performance of both the Root Finding methods and Spectral Factorisation Techniques.

2. **Newton-Raphson** This algorithm (3.2.2.i, 3.2.2.i.a, in Chapter 3) requires the evaluation of a function, say \( f(x) \), as well as its derivative \( f'(x) \). The requirement of an explicit representation of the derivative constitutes a major disadvantage of the method. Note, however that if the function \( f(x) \) is a polynomial, this disadvantage vanishes and the method has some practical value. Tables NR.1a, 1, 4, 8 show the accuracy of the roots found over Channels 1-8. Two stopping criteria, given by the parameters \( \delta \) and \( \epsilon \) have been used to terminate the algorithm. The threshold \( \delta \) (Equations 3.2.2.3 and 3.2.2.4) which signifies the closeness of the initial starting point to an actual root to ensure convergence to that root, is set to \( 10^{-4} \). The second parameter \( \epsilon \) shows the tolerance for the output polynomial evaluated at the newly found root (See Appendix 6 - Programs:Newton.m ). The parameter \( \epsilon \) was set to \( \epsilon = 10^{-8} \) (Relevant Tables showing that the root locating capability and accuracy of the algorithm does not improve with \( \delta < 10^{-4} \) and \( \epsilon < 10^{-8} \) without the inevitable increase in the arithmetic operations [are not included here]). In the case of channel 1, all three roots were located with very high accuracy and 15 iterations in total. The same observation holds true for channels 2 and 3 where all four roots (for each channel) were located successfully (with 29 and 30 iterations respectively, and high accuracy). The number of filter taps was decided, from Tables NR.3 and 5, to be 30, a high enough number to exploit the
accuracy achieved by the algorithm (and therefore ensure the accurate adjustment of the prefilter).

In the case of channel 4, the algorithm failed to locate two roots (Table NR.4), thus resulting in the unacceptably low value of $\psi_3$ shown in Table NR.5. In the case of channel 5 all roots were found but in the next channel (channel 6) the algorithm failed to locate one root giving a very low value of $\psi_3$ (Table NR.6), thus resulting in a poor adjustment of the prefilter (poor value of $\psi_2$ - Table NR.6). Channel 7 presents an even worse case for the Newton-Raphson method which now fails to find 4 roots (Table NR.8). This results in the lowest value of $\psi_3$ so far, although the value of $\psi_2$ is high (Table NR.9). Channel 8 shows that the algorithm managed to locate all thirteen roots with high accuracy. Tables NR.3, 7, 10 show the number of arithmetic iterations when Newton-Raphson is applied over channels 1-8. Note that a great many starting points are assumed, in the simulation, before failure of the algorithm to converge is admitted. This highlights a second very important point. The Tables indicating the number of iterations show only the iterations that led to a successful location of a root. For almost every case of valid results there are many unsuccessful attempts resulting in a great number of extra iterations. The exact number depends on the particular channel and as a rule of thumb, the higher the number of roots is, the higher the number of extra iterations. Of course, this overhead is not taken into account when the number of operations is calculated. Bear also in mind that the algorithm converges to all roots and therefore the overhead increases still further.

However, when it converges to the required roots, it does so with very high accuracy and the rate of convergence is quadratic as can be seen from Graphs GNR 1-8. To exploit the very good convergence property and to avoid the increased computational load associated with the location of all roots (wanted or not), a new composite algorithm based on the Newton-Raphson method was devised. This composite structure will be called QRN method and is a combination of the Newton-Raphson and QR methods.

2a. QRN In order to reduce the failure rate of Newton-Raphson (and perhaps also reduce the number of operations required by this algorithm), the algorithm has been combined with the QR method to give a new structure the QR/Newton-Raphson (QRN) algorithm. Applying the QR on a truncated input polynomial, provides the Newton-Raphson with good initial guesses which in
ADJUSTMENT ALGORITHMS OVER THE TELEPHONE NETWORK

turn are used to find the required roots in a more accurate fashion. Note that the Tables and results relating to the discussion of this method are listed in Appendix A4-Section A4.6.1. In the case of channel 1 the minimum length to which the input Y(z) should be truncated is specified by the failure of the QRN to find one root of the three roots of channel 1 (Table QRN.1). Therefore, the length is now set to L=6 and the three roots are now found. Note that compared with the accuracy achieved by the ‘ordinary’ Newton-Raphson over the same channel (Table NR.1) QRN achieves higher accuracy although at the expense of one more iteration. Table QRN.2 shows that the minimum truncation length is L=9 for the algorithm to accurately find the four roots that lie outside the unit circle, for channel 2. The number of iterations needed to find the roots was 18 (Table QRN.2), compared with 29 (Table NR.1) for the ‘ordinary’ Newton-Raphson. For Channel 3 the length required is 10 and the number of iterations 23 (Table QRN.3) as opposed to 30 iterations (Table NR.1). Over channel 4 the algorithm fails to locate two roots (Table QRN.4) while in channel 5 locates successfully all eight roots with 44 iterations (Table QRN.5) as opposed to 53 (Table NR.4). QRN method operated extremely well as far as accuracy is concerned, over channels 1, 2, 3 and 5. However, this was offset by the greater numbers of arithmetic operations (Table QRN.6 ) required by this algorithm. Although the second part of the algorithm (Newton-Raphson) required in general, fewer iterations than the Newton Raphson (on its own), the first part the QR method, in order to provide good initial guesses, imposed a heavy penalty on the total computational load of the algorithm. Finally, the method fails over channels 6, 7 and 8, in the sense that the QR method should do most of the work (i.e. the polynomial Y(z) is now truncated at such a high point that it becomes meaningless to use the QR method combined with Newton-Raphson).

2b. In this version (3.2.2.i.c in Chapter 3) the original input polynomial is truncated at certain lengths, depending on the distortion introduced by the particular channel (more severe distortion usually means longer polynomial in this context), to provide initial approximations to the wanted roots, which in turn serve as starting points for the original polynomial. This algorithm was expected to exhibit better behaviour than the ‘ordinary’ Newton as far as accuracy (and perhaps arithmetic complexity) is concerned or the QRN structure (fewer arithmetic operations), since the
initial approximations would be more accurate (and fewer) than randomly chosen starting points. Results that are not included here clearly showed that the algorithm exhibited a surprisingly high overhead especially over channels 5, 7 and 8 without increased accuracy. This disappointing behaviour was due to the unpredictability of the method when the initial points are not very good approximations. In view of the rather poor performance of the algorithms presented so far, the Newton-Raphson algorithm is applied next in an original way, thus giving enhanced performance as far as complexity and operations are concerned.

3. NRm Now the algorithm is searching for the roots that lie inside the unit circle. They are the minority and they lie within a well specified area; therefore it is much easier for the algorithm to locate them. Tables NRm.1, 4, 7, 11, 13 and 20 give the accuracy with which the roots are located when the algorithm is applied over the eight channels employed in the simulation tests. The parameter $\delta$ has been set to $\delta = 10^{-4}$ permanently for all channels and the parameter $\varepsilon$ is varied ($\varepsilon = 10^{-4}, 10^{-6}, 10^{-8}$). Contrary to what was expected increasing the value of $\varepsilon$ resulted in a decrease in the number of iterations for channels 1 and 2. Therefore the highest accuracy was achieved with the fewest arithmetic operations as it is shown in Tables NRm.3 and 6. However this does not hold true for the rest of the channels. Tables NRm.9, 12, 15, 17, 20 and 22 indicate the number of arithmetic operations required by the algorithm when applied over the channels 1-8. Finally, Tables NRm.2, 5, 8, 10, 14, 17 and 23 show the performance of the algorithm in terms of the values of $\psi_1, \psi_2$ and $\psi_3$ with different numbers of prefilter taps. It is fairly obvious from these Tables that 30 taps represent an adequate choice for the accurate adjustment of the prefilter. The algorithm fails only in the case of channel 7 where it fails to locate one root (Table NRm.19). Overall, the performance of this algorithm is very good and is the best algorithm considered up to now. However, the number of operations is still high especially for channels 7 and 8. This fact, and secondly the occasional failure over channel 7 lead to the consideration of the next algorithm.

4. Laguerre method (3.2.3.i) Laguerre method is globally convergent for real roots only. If the input polynomial has complex roots (as is the case here), the method is no longer globally convergent. However empirical evidence indicates that nonconvergence is exceptional. Turning
to efficiency considerations Laguerre’s method is cubically convergent for simple roots, which is exceedingly fast. Laguerre’s method has been applied on channels 1-8 and has successfully located all roots in all cases. However, this approach presents two problems. The first one, is that the method finds all the roots of the input polynomial irrespective of their position to the unit circle at the expense of an increased computational load. The choice of the initial starting points plays no particular role in the order that the method locates the roots. A new way of applying this method was found so that Laguerre method (3.2.3.ii, in Chapter 3) converges only on the roots that lie outside the unit circle. This development combined with the accuracy of the method, makes Laguerre method a very attractive algorithm for the adjustment of the prefilter. The input polynomial is now reversed and therefore all the roots that lie inside the unit circle are now the wanted roots; The algorithm always starts from the same point $0+i*0$ (alternatively the last located root) and searches for the roots inside the unit circle (a much simpler problem since these inside roots form the minority of the polynomial roots and secondly because the algorithm now searches through a smaller and well defined area). Tables L.1, 6, 10 and Graphs GL. 1-8 show the accuracy with which the roots are found when applying Laguerre method over Channels 1-8. Tables L.2, 3, 4, 7, 8 and 11 show the performance of the algorithm over Channels 1-8 for different lengths of the prefilter (20-100 taps), given by the values of $\psi_1$, $\psi_2$ and $\psi_3$. The parameter $\text{eps}$ signifies the fractional roundoff error in evaluating the polynomial $Y(z)$ and has been set to $\text{eps}=10^{-10}$. Finally, Tables L.5, 9, 12 show the number of arithmetic operations required by the algorithm to locate the roots over each of the eight channels.

Turning to efficiency considerations, Laguerre’s method is cubically convergent for simple roots, (See Graphs GL 1-8) which is exceedingly fast. It is evident, from the relevant tables that Laguerre method converges in all roots on all cases with a reasonable number of iterations. On average it took three iterations for Channels 1-3 and 4 iterations for channels 5-8. In summary, Laguerre’s method is very good. Its main advantage is global convergence which, incidentally, is not offset by relatively poor efficiency. In general, it is lack of understanding of its behaviour in the case of the complex roots that has prevented it from gaining more widespread use.
4.a QRL  In an attempt to reduce the computational load implied by the QR method, while at the same time maintaining the accuracy implicit in it, QR method was combined with Laguerre algorithm. In this combined QR/Laguerre (QRL) method, the original input polynomial $Y(z)$ is truncated up to a degree dictated by the distortion of the particular channel. The root of the truncated polynomial then are located with very high accuracy by the QR method. Of course, the located roots are different from the corresponding ones of the original polynomial, but they serve as initial starting points for the second stage of the combined method, i.e. the Laguerre method. The Laguerre method, in this way, serves as a means of polishing the roots initially found by the QR method. Therefore the Laguerre can in some cases actually locate the required roots (roots outside the unit circle), with high accuracy and in just a few iterations. Note that the Tables and results relating to the discussion of the QRL method are listed in Appendix A4.

Tables QRL.1, QRL.3-4, QRL.6, QRL.8 and QRL.10 (see Appendix A4- Section A4.6.2) show the accuracy with which the roots are found, and the numbers of iterations required by the algorithm to locate the roots, for channels 1-8. The input polynomial $Y(z)$ is truncated to a length of 6 components for the case of channel 1. The Laguerre method is set to operate with $\text{eps}=10^{-4}$, $10^{-6}$ and finally $10^{-8}$. The number of iterations (Table QRL.1) change slightly with eps (less iterations with $10^{-4}$), and the accuracy ($\psi_2$, $\psi_3$) remains the same with a small change in the value of $\psi_1$. Therefore from now on the value of eps is permanently set to $10^{-4}$ and only the length of the input polynomial $Y(z)$ changes according to the distortion of the channels. In table QRL.3 two different truncation values ($L=6$ and 7) have been applied to the original polynomial. It can be seen that the method fails to locate the fourth roots that lies outside the unit circle for these two truncation lengths. This is due to the fact that the truncated polynomial is too sort for the fourth root to survive, while the other three roots have survived with enough accuracy to serve as good initial guesses for the Laguerre method. Therefore $Y(z)$ is next truncated up to 8 components and all four roots located successfully (Table QRL.4). In the case of channel 4 the method failed to locate the required roots. For channel 5 the QRL managed to find all eight roots but the original polynomial had to be truncated up to 20 components (the original polynomial had 31 components). This caused a considerable increase in the operations required by the algorithm,
shown in Table QRL.9. Table QRL.10 shows the root accuracy of the algorithm for channel 6. The algorithm has found all nine roots and the input polynomial was truncated up to eight components which represents a pretty reasonable number of arithmetic operations (Table QRL.11), especially when compared with channel 6 (Table QRL.9). Tables not included show that the algorithm fails over channels 7 and 8. This algorithm although extremely accurate on some instances is neither fast nor globally convergent. This fact reinforces the belief that the best algorithm (of all the root finding algorithms examined up to now) is the Laguerre algorithm (version 3.2.ii).

In order to use the root finding methods in the adaptive adjustment of the prefilter, they should be able to provide a set of coefficients necessary to set and adjust the prefilter. Two adjustment schemes have been proposed (Chapter three). These have been evaluated and the results represented in Tables AD.1-AD.8 for the first general adjustment scheme, and in Tables CAD.1-8 for the second adjustment scheme which forms part of the Clark-Hau root finding algorithm to be presented next. Note that only results showing their arithmetic complexity are presented, since their accuracy is shown in the Tables concerning the accuracy of the various methods. The accuracy of the adjustment schemes is given by the parameters $\psi_1$ and $\psi_2$.

5. CLARK-HAU Tables CH.1-1.a show the root accuracy of this algorithm over channel 1 for the case of two different sets of starting points when the threshold value $d$ used for terminating this algorithm has been set to $d = 10^{-6}$ and $10^{-8}$. Table CH.2 shows the performance of the root finding algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 1, for different numbers of prefilter taps. Tables CH.3 and CH.4 show the root accuracy of the algorithm and the performance of the Clark-Hau algorithm (for different numbers of prefilter taps), over Channel 2. Note that the set of starting points selected is the one giving the greatest accuracy. Tables CH.6 and CH.7 clearly show that the accuracy achieved by the algorithm, very much depends on the choice of the set of the starting points. Obviously, the set of the starting points in Table CH.7 achieves the highest accuracy for Channel 3. Table CH.8, indicates the performance of the
algorithm over channel 3, for different numbers of prefilter taps (when the root accuracy is the one shown in Table CH.7).

Table CH.9 shows again the vulnerability of the Clark-Hau algorithm, when the algorithm fails to locate one of the wanted roots, over Channel 4. The point that become clear from the above tables is that the choice of the starting points as well as the order in which they are tried makes a great difference to the root locating ability of the algorithm. Next table, Table CH.10 shows how the choice of \( d \), affects the root accuracy of the algorithm. In channel 5 the algorithm has failed to identify one root. Tables CH.13-14 indicate three (of many) different set of starting points used by the algorithm in an attempt to locate the missing root. Table CH.15 shows the performance of the Clark-Hau algorithm over channel 5, given by the values of \( \psi_1 \), \( \psi_2 \) and \( \psi_3 \). These values are unacceptably low, which was expected since one root is not accounted for, when the algorithm converts \( Y(z) \) into its minimum phase equivalent version. Tables CH.16-19 show that the algorithm fails once more to locate at least one root, when operating over channel 6, despite the different set of starting points tried. The situation gets worse when the algorithm operates over channel 7. Four roots are now not found and the result of this failure is evident in the low values of \( \psi_1 \), \( \psi_2 \) and \( \psi_3 \) in table CH.23. Over channel 8 things are better, since only one root is not located (Table CH.24), but still this is an unacceptable situation (Table CH.25).

A suit of spectral factorisation techniques (section 3.3 in Chapter 3) that provide an alternative approach to the problem of minimum phasing is the theme of the rest of this Section. Although the spectral factorisation techniques, assume no knowledge of roots, since they are dealing with the polynomial \( Y(z) \) itself, some tables indicating the root accuracy of the techniques have been included for comparison purposes.

6. NIT algorithm (3.3.4) Table NIT.1 shows the accuracy of the NIT algorithm, given by \( \psi \) over channels 1-4 for different numbers of iterations. Similarly, Table NIT.2 gives the accuracy of the algorithm over channels 5-8. As it can be seen from Table NIT.1 the algorithm is extremely accurate over channels 1 and 2. This become clear when the number of iterations becomes
greater than 10. This changes for the rest of the channels although the achieved accuracy is high enough for the adjustment of the prefilter. Tables NIT.3-8 give the arithmetic operations required by the algorithm over channels 1-8. Although the algorithm is quite accurate it requires a great number of arithmetic operations.

6a.NIT (3.3.4.b) The algorithm has now been modified in an attempt to reduce the number of required arithmetic operations, while at the same time to maintain the accuracy of the algorithm. Tables NIT.1, 3, 4, 6, 8, 11 and 13 show the accuracy (value of \( \psi \)) of the modified algorithm for channels 1-8, for different lengths \( L \) of the filter and different numbers of iterations. Note that there is a loss of accuracy for channels 1 and 2 (compared with Table NIT.1), especially for a number of iterations greater than 15, and a smaller loss of accuracy for channels 3-8. However, this is accompanied by a reduced number of arithmetic operations (Tables NIT.2, 5, 7, 9, 10, 12 and 14), thus making this version of the NIT algorithm more attractive for real time implementation.

6b.NIT (3.3.4.a) This is a very fast algorithm indeed. Tables that have not been included here show that this version of NIT is slightly suboptimum when compared with 3.3.4. or 3.3.4.b versions. However the increased operational speed more than compensates for the slight loss in accuracy. The real problem experienced by the algorithm is the need for an effective (global) stopping criterion. Further results and details are given in [32].

7.GRAM-SCHMIDT Tables GS.1-8 give the accuracy of the Gram-Schmidt algorithm over channels 1-8, for different numbers of filter taps, while Tables GS.9-15 indicate the cost of this algorithm, (as far as arithmetic complexity is concerned) over the same channels. Gram-Schmidt method is very accurate over all eight channels but unfortunately requires a great number of flops and therefore, is not suitable for real time implementation.

8. BAUER METHOD Tables B.1 show the accuracy of this method given by \( \psi_1, \psi_2 \) and \( \psi_3 \), over channels 1-6. Table B.2 shows the enhanced performance of the Bauer method over
channels 1 and 2 when the original input is injected with a number of zeros. This lengthening of the input polynomial though results also in an increased number of arithmetic operations. Notice also that the performance of the method remains unacceptably low over channel 3. Bear also in mind that it also fails for the rest of the channels.

9. LEVINSON This algorithm implements the Toeplitz method (3.3.1-Chapter 3).
Tables LE 1-11 show collectively the behaviour of the algorithm over the eight telephone channels. The algorithm exhibits quite acceptable performance both in terms of accuracy and complexity. However, much better performance was expected and definitely a lot of other more powerful algorithms are available such as the Cepstrum algorithm next.

10. CEPSTRUM Table CP.1 gives the performance of the Cepstrum algorithm over Channel 1 (values of $\psi_1$, $\psi_2$ and $\psi_3$) for different lengths of the Fourier transform, and Table CP.2 the number of arithmetic operations over the same channel. Since the accuracy that the Fourier transform of length 64 yields is high enough (Table CP.1), combined with a reasonable number of arithmetic operations that corresponds to that length (Table CP.2), the length of the Fourier transform has been set to 64 in Table CP.3 while the number of filter taps varies, and the performance of the algorithm is then given by the values of $\psi_1$, $\psi_2$ and $\psi_3$. Table CP.4 show the number of arithmetic operations required by the Cepstrum algorithm for a 64-point Fourier transform over channel 1 for different numbers of filter taps. The same procedure is followed for the rest of the channels. Tables CP.5-CP.24 collectively, show the 'worth' of the algorithm over all channels. The Cepstrum algorithm is the most accurate algorithm and also the fastest of all algorithms (root finding methods and spectral factorisation techniques) investigated so far and therefore is the most promising for hardware implementation which is the theme of next chapter. The last Tables of this Section (Tables NITs 1-2, CLs 1-11, LEs 1-6) show collectively the performance of the NIT, Clark-Hau and Levinson when they operate on truncated sampled impulse responses. Now the Sampled impulse responses of Channels 1-3 have been truncated to 10 components. The sampled impulse response of Channel 4 has been truncated to 15 components and the sampled impulse responses of channels 5-8 to 20 components. They are
given in Tables 4.9-4.11 along with the corresponding roots (given in Tables 4.12-4.15). The algorithms have been selected to represent best and worst cases and used as to get a 'feel' if things change radically when the sampled impulse responses of the channels are shortened (as it would be the case in a real time implementation). It is evident that the situation remains unchanged and the same algorithms exhibit the same behaviour with the truncated channels now as with the full channels before.

11. NIT -The Time Varying case (N I T over HF radio Links)

The Root Finding algorithms and Spectral Factorisation techniques presented so far have been tested by the author over HF radio links (and land mobile radio links) which represent a time varying environment. A number of ten (10) different HF channels have been computer-simulated with a view to forming a platform over which HF modems can be tested and the highest possible data transmission rates can be determined. The channels simulate conditions which range from very mild conditions to extremely disturbed conditions. The channels are assumed to comprise of up to three (3) and in some cases four (4) independent and totally uncorrelated skywaves each undergoing Rayleigh fading with frequency spread ranging from 0.5 Hz to 2 Hz (fading of 22 to 88 fades per minute) and a multipath spread of up to 3 ms. The channels together represent a wide range of operating conditions that are more severe than the C.C.I.R or U.S.A.E.L classification. For the purposes of this work, it is necessary to accept the sampled impulse responses in Tables T.1, T.4, T.6, T.8, T.11 and T.13, for what they are -sequences of numbers over which several methods have been tested. When viewed in this light, it is not necessary to show the way they were derived. However, it is equally important to stress that they are valid sampled impulse responses representing real channels (and not merely artificial number structures) for finding the limiting conditions for each method considered. To justify their validity see the discussion in Appendix 4 (Sections A4.4 -A4.5) and Ref. 33. The results of the tests over the derived sampled impulse responses are presented in Graphs G.NTI-5 and Tables T.2-3, T.5, T.7, T.9, T.12 and T.14, for different transmission instants each time.
In particular, Graph G.NT1 shows the performance of Cepstrum and N I T algorithms over HF radio link 1 (HF1), which represents an HF radio channel with two skywaves present with relative (to the first) time delay 3 ms and frequency spread 0.5 Hz. The sampled impulse response of this HF radio channel is shown in Table T.1, along with the root positions of the sampled impulse response. Table T.2 indicates the accuracy of the two methods over HF1 in terms of \( \psi_2 \) and \( \psi_3 \) parameters. It is pretty obvious that the N I T method performs better than the Cepstrum method, especially when the latter is applied with an FFT length of 64. This results in the introduction of a new root of 1.03 magnitude in the sampled impulse response which is available to the detector (associated with \( \psi_3 \)), and another root of magnitude 1.06 in the combined sampled impulse response of the channel and filter (associated with \( \psi_2 \)), also readily available to the detector. These new roots result in a 'pseudo minimum' phase sequence as an input to the detector. It is clear that this is not a failure of the Cepstrum algorithm, since it has successfully shifted the original two roots (with magnitudes 1.12 and 1.12) of the 'raw' sampled impulse response very close to the unit circle. In fact, their magnitudes are acceptable when compared with the 1.05 established limit [Hau Clark et al]. Bear also in mind that by increasing the FFT length (FFT length transform is now 128) the roots move even closer to the unit circle [see Table T.3], (new magnitudes 1.012 and 1.016 respectively), although the algorithm is now slower. In the case of the N I T algorithm there are no roots outside the unit circle, thus a better performance can be achieved (Table T.2). Any of the root finding algorithms encountered problems with this particular sampled impulse response. Although there are only two roots outside the unit circle, they caused either total failure of some root finding algorithms or in the best case severe overhead in terms of iterations, thus in essence precluding the use of root finding algorithms over HF1 channel and for this particular transmission instant.

Graph G.NT2 shows the performance of Cepstrum and N I T algorithms over HF2, which represents an HF radio channel with three skywaves present with relative time delays 1 and 3 ms and frequency spread 2 Hz. There are now four roots outside the unit circle as can be seen from Table T.4 which shows the 'raw' sampled impulse response' of HF2 together with the root positions associated with this sampled impulse response. Table T.5 shows the accuracy of the
Cepstrum and N I T and any of the root finding algorithms. Assuming perfect location of the root positions by any of the root finding algorithms (and not processing roots with magnitudes less than the limit 1.05), N I T fares better than the rest since it achieves (in a few iterations) accuracy which is as high as the one that could be obtained by a root finding method assuming successful and very accurate location of all roots outside the unit circle (in this case two) but without processing those roots that lie outside the unit circle but have magnitude less than 1.05. The Cepstrum method achieves comparable overall performance (slightly worse $\psi_2$ but fares equally well when $\psi_3$ is considered) to that of the N I T method.

Graphs G.NT3-4 show the performance of the Cepstrum, N I T and any of the root finding algorithms over HF2 radio channel but during another transmission instant. In particular, graph G.NT3 shows the 'raw' sampled impulse response (indicated by the dotted line in the graph), the ideal minimum phase sampled impulse response (found by an EISPACK routine and indicated with a continuous thick line) and finally the minimum phase sequence achieved by the root finding algorithm (any), assuming perfect location of the roots outside the unit circle (but at the same time non-processing of those roots that lie outside the unit circle but their magnitude is less than 1.05). It can be seen from Table T.6 that there are sixteen (16) roots outside the unit circle, and the fifteen of these roots lie in a distance less than 1.05 away from the unit circle, thus posing an impossible situation for any root finding algorithm if the latter were to process every single one of these roots. Furthermore, these roots form rings just outside the unit circle and appear in clusters. Accepting the 1.05 limit (established by Clark-Hau et al) only one root (the one with modulus 1.5 is processed), but this results in the low accuracy indicated by $\psi_2$ and $\psi_3$ in Table T.7 and the 'pseudominimum' sampled impulse response, shown in Graph G.NT3 by the continuous thin line. As can be seen, it is quite different from the true minimum phase sampled impulse response shown by the continuous thick line in the same Graph. The Cepstrum and N I T methods show comparable performance (when the number of iterations is set to 10 for the N I T and the length of the Fourier Transform is set to 64 for the Cepstrum method). Graph G.NT4 clearly shows how similar the two algorithms behave. It can also be seen from Table T.7 that the number
of iterations can be reduced down to 6 (or even further, although not shown) and still the N I T algorithm achieves better performance than any of the root finding algorithms.

The next Graph, G.NT5 shows the performance of Cepstrum and N I T algorithms over HF radio channel 3 (HF3). This channel incorporates three skywaves with relative time delays 1 and 2 ms and frequency spread 2 Hz. Table T.9 shows the accuracy of the two methods in terms of $\psi_2$ and $\psi_3$. It can be seen that the N I T has a much better performance, than the Cepstrum method when the number of iterations is set to 10 (while at the same time the length of the Fourier transform is set to 64 for the Cepstrum method). The number of iterations can be reduced down to only two (Itrs=2), and even then its performance is superior to that of the Cepstrum method. On this occasion the Cepstrum method introduces two roots (associated with $\psi_2$) outside the unit circle, with magnitudes 1.03 and 1.04 respectively. It also introduces two roots (associated with $\psi_3$), with magnitudes 1.08 and 1.059 respectively. However, this changes when the length of the Fourier transform length increases to 128. Then the roots are shifted so close to the unit circle that for all purposes they are considered to be on the unit circle (see Table T.10). One reason that explains the behaviour of the Cepstrum algorithm over this channel and at this particular transmission instant, is the existence in the ‘raw’ sampled impulse response of roots very close to the unit circle (magnitudes 1.0045 0.998 0.98 1.001, as can be seen from Table T.8) and this poses a very difficult situation for the Cepstrum algorithm.

Graph G.NT6 shows the ‘raw’ sampled impulse response (indicated by the dotted line) of HF3 at another transmission instant, along with the ideal minimum phase sampled impulse response (indicated by the thick continuous line). The ‘pseudominimum’ phase sampled impulse response (achieved by any root finding algorithm) is also shown (thin continuous line) and differs quite a lot especially over the first few samples. This comes as no surprise since there are sixteen roots outside the unit circle. These roots form a ring at a small distance from the unit circle. Only two out of these sixteen roots have to be processed since only two have magnitudes greater than 1.05 (Table T.11). Table T.12 shows the accuracy (in terms of $\psi_2$ and $\psi_3$) for the Cepstrum and N I T
methods as well as any of the root finding methods. N I T and Cepstrum methods are both accurate and the ‘closeness’ of their performance is clearly visible in Graph G.NT7.

Graph G.NT8 shows the performance of any root finding method over the same as previously HF channel (HF3) at yet another transmission instant, in terms of the ‘pseudominimum’ sampled impulse response (indicated by the thin continuous line). This is different to the the ideal (indicated by the thick continuous line) since there are again sixteen roots (forming a ring outside the unit circle) and only one is processed since it has magnitude greater than 1.05 (see Table T.13). This results in the low accuracy in $\psi_2$ and $\psi_3$ (Table T.14), while the Cepstrum and N I T exhibit accurate (and very similar) performance (same Table). This is also shown graphically in Graph G.NT9.

4.4.2 Summary of Results

In this part of the Chapter only the principal adjustment algorithms have been considered. These are the Newton-Raphson (algorithm 3.2.2.ii), the Laguerre method (algorithm 3.2.3.ii), the Clark-Hall algorithm and finally the Cepstrum technique. Since it is well known that a picture can be as good a thousand words the characteristics of the above methods are displayed in a set of graphs G1-8 (p. 218-225). On each of these graphs each method is represented by four bars. The top bar for each set reads according to the top axis of the graph which is the number of flops (or else the execution time). The rest of the bars in each set represent the three familiar by now parameters $\psi_1$, $\psi_2$ and $\psi_3$ and should be read with the lower axis of the graphs which measures accuracy in dB.

It is crystal clear from these graphs that although no single best method exists for each measurement separately and for all the eight channels, the method that is best when all the parameters are considered together, is the Cepstrum method.

Finally, note that, given the closeness of the performance of the Cepstrum and N I T methods (over the considered HF channels), the N I T method (not shown in the graphs G1-8, for simplicity) constitutes a very powerful (albeit slower) alternative to the Cepstrum algorithm.
<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.0291</td>
<td>-0.0373</td>
<td>0.0145</td>
<td>-0.0006</td>
</tr>
<tr>
<td>0.2290</td>
<td>-0.2244</td>
<td>0.0750</td>
<td>0.0176</td>
</tr>
<tr>
<td>0.7612</td>
<td>0.1817</td>
<td>0.3951</td>
<td>0.0033</td>
</tr>
<tr>
<td>0.2988</td>
<td>0.3050</td>
<td>0.7491</td>
<td>-0.1718</td>
</tr>
<tr>
<td>-0.0338</td>
<td>-0.2915</td>
<td>0.1951</td>
<td>0.0972</td>
</tr>
<tr>
<td>-0.0789</td>
<td>0.0516</td>
<td>-0.2856</td>
<td>0.1894</td>
</tr>
<tr>
<td>0.0291</td>
<td>0.0287</td>
<td>0.0575</td>
<td>-0.2096</td>
</tr>
<tr>
<td>-0.0137</td>
<td>-0.0352</td>
<td>0.0655</td>
<td>0.1139</td>
</tr>
<tr>
<td>0.0020</td>
<td>0.0204</td>
<td>-0.0825</td>
<td>-0.0424</td>
</tr>
<tr>
<td>0.0004</td>
<td>-0.0108</td>
<td>0.0623</td>
<td>0.0085</td>
</tr>
<tr>
<td>0.0028</td>
<td>0.0065</td>
<td>-0.0438</td>
<td>0.0034</td>
</tr>
<tr>
<td>-0.0027</td>
<td>-0.0014</td>
<td>0.0294</td>
<td>-0.0049</td>
</tr>
<tr>
<td>0.0000</td>
<td>-0.0013</td>
<td>-0.0181</td>
<td>0.0032</td>
</tr>
<tr>
<td>0.0003</td>
<td>0.0006</td>
<td>0.0091</td>
<td>0.0003</td>
</tr>
<tr>
<td>-0.0002</td>
<td>0.0001</td>
<td>-0.0038</td>
<td>-0.0023</td>
</tr>
<tr>
<td>-0.0009</td>
<td>-0.0006</td>
<td>0.0019</td>
<td>0.0027</td>
</tr>
<tr>
<td>-0.0018</td>
<td>-0.0014</td>
<td>0.0005</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0003</td>
<td>0.0004</td>
<td>0.0006</td>
<td>0.0003</td>
</tr>
<tr>
<td>0.0001</td>
<td>-0.0011</td>
<td>0.0005</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0004</td>
<td>0.0001</td>
<td>-0.0008</td>
<td>-0.0001</td>
</tr>
<tr>
<td>-0.0005</td>
<td>-0.0001</td>
<td>0.0000</td>
<td>-0.0002</td>
</tr>
<tr>
<td>0.0006</td>
<td>0.0004</td>
<td>0.0001</td>
<td>0.0006</td>
</tr>
<tr>
<td>-0.0004</td>
<td>0.0001</td>
<td>0.0010</td>
<td>0.0007</td>
</tr>
<tr>
<td>-0.0001</td>
<td>0.0000</td>
<td>-0.0007</td>
<td>0.0007</td>
</tr>
<tr>
<td>0.0004</td>
<td>0.0008</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 4.1  Sampled Impulse Responses of Channels 1 to 4
<table>
<thead>
<tr>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.0048</td>
<td>0.0020</td>
<td>0.0085</td>
<td>0.0012</td>
</tr>
<tr>
<td>-0.0233</td>
<td>-0.0056</td>
<td>0.0167</td>
<td>0.0039</td>
</tr>
<tr>
<td>-0.0040</td>
<td>-0.0952</td>
<td>0.0673</td>
<td>0.0082</td>
</tr>
<tr>
<td>0.2497</td>
<td>-0.1696</td>
<td>0.2217</td>
<td>0.0002</td>
</tr>
<tr>
<td>0.5415</td>
<td>0.1196</td>
<td>0.5021</td>
<td>-0.0566</td>
</tr>
<tr>
<td>0.2978</td>
<td>0.3424</td>
<td>0.5481</td>
<td>-0.1103</td>
</tr>
<tr>
<td>0.0116</td>
<td>-0.0933</td>
<td>0.0464</td>
<td>0.0641</td>
</tr>
<tr>
<td>0.0699</td>
<td>-0.2273</td>
<td>-0.2943</td>
<td>0.2100</td>
</tr>
<tr>
<td>-0.1166</td>
<td>0.1071</td>
<td>0.0065</td>
<td>-0.1019</td>
</tr>
<tr>
<td>-0.0515</td>
<td>0.0550</td>
<td>0.1460</td>
<td>-0.1284</td>
</tr>
<tr>
<td>0.1206</td>
<td>-0.0515</td>
<td>-0.0797</td>
<td>0.1840</td>
</tr>
<tr>
<td>-0.0806</td>
<td>-0.0175</td>
<td>-0.0513</td>
<td>-0.0552</td>
</tr>
<tr>
<td>-0.0013</td>
<td>0.0481</td>
<td>0.0846</td>
<td>-0.0871</td>
</tr>
<tr>
<td>0.0547</td>
<td>-0.0171</td>
<td>-0.0174</td>
<td>0.1275</td>
</tr>
<tr>
<td>-0.0422</td>
<td>-0.0252</td>
<td>-0.0488</td>
<td>-0.0708</td>
</tr>
<tr>
<td>0.0076</td>
<td>0.0428</td>
<td>0.0592</td>
<td>-0.0081</td>
</tr>
<tr>
<td>0.0167</td>
<td>-0.0342</td>
<td>-0.0296</td>
<td>0.0558</td>
</tr>
<tr>
<td>-0.0237</td>
<td>0.0167</td>
<td>-0.0076</td>
<td>-0.0585</td>
</tr>
<tr>
<td>0.0136</td>
<td>-0.0033</td>
<td>0.0283</td>
<td>0.0350</td>
</tr>
<tr>
<td>0.0018</td>
<td>-0.0050</td>
<td>-0.0318</td>
<td>-0.0078</td>
</tr>
<tr>
<td>-0.0142</td>
<td>0.0074</td>
<td>0.0254</td>
<td>-0.0105</td>
</tr>
<tr>
<td>0.0178</td>
<td>-0.0087</td>
<td>-0.0148</td>
<td>0.0185</td>
</tr>
<tr>
<td>-0.0145</td>
<td>0.0103</td>
<td>0.0044</td>
<td>-0.0188</td>
</tr>
<tr>
<td>0.0070</td>
<td>-0.0120</td>
<td>0.0026</td>
<td>0.0149</td>
</tr>
<tr>
<td>0.0014</td>
<td>0.0109</td>
<td>-0.0065</td>
<td>-0.0102</td>
</tr>
<tr>
<td>-0.0072</td>
<td>-0.0071</td>
<td>0.0079</td>
<td>0.0058</td>
</tr>
<tr>
<td>0.0095</td>
<td>0.0007</td>
<td>-0.0077</td>
<td>-0.0022</td>
</tr>
<tr>
<td>-0.0069</td>
<td>0.0040</td>
<td>0.0063</td>
<td>-0.0004</td>
</tr>
<tr>
<td>0.0027</td>
<td>-0.0063</td>
<td>-0.0048</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0016</td>
<td>0.0060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0041</td>
<td>-0.0034</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0016</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0027</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0033</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0034</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0034</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0034</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0033</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0031</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0027</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2 Sampled Impulse Response of Channels 5 to 8
<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.5178</td>
<td>-0.6638</td>
<td>0.8709</td>
<td>-0.0360</td>
</tr>
<tr>
<td>-0.0368</td>
<td>-0.5166</td>
<td>0.4455</td>
<td>0.1567</td>
</tr>
<tr>
<td>0.1053</td>
<td>0.0977</td>
<td>-0.1264</td>
<td>-0.0019</td>
</tr>
<tr>
<td>-0.0392</td>
<td>0.0151</td>
<td>0.0258</td>
<td>-0.0969</td>
</tr>
<tr>
<td>0.0149</td>
<td>-0.0083</td>
<td>0.0083</td>
<td>0.0065</td>
</tr>
<tr>
<td>-0.0012</td>
<td>0.0162</td>
<td>-0.0118</td>
<td>-0.0031</td>
</tr>
<tr>
<td>-0.0033</td>
<td>-0.0060</td>
<td>0.0087</td>
<td>0.0004</td>
</tr>
<tr>
<td>0.0022</td>
<td>0.0033</td>
<td>-0.0093</td>
<td>-0.0011</td>
</tr>
<tr>
<td>-0.0032</td>
<td>-0.0039</td>
<td>0.0071</td>
<td>0.0023</td>
</tr>
<tr>
<td>0.0033</td>
<td>0.0002</td>
<td>-0.0038</td>
<td>-0.0023</td>
</tr>
<tr>
<td>-0.0008</td>
<td>0.0026</td>
<td>0.0002</td>
<td>0.0029</td>
</tr>
<tr>
<td>-0.0003</td>
<td>-0.0013</td>
<td>0.0009</td>
<td>-0.0024</td>
</tr>
<tr>
<td>-0.0007</td>
<td>0.0003</td>
<td>-0.0004</td>
<td>0.0013</td>
</tr>
<tr>
<td>0.0007</td>
<td>0.0010</td>
<td>0.0006</td>
<td>0.0008</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.0019</td>
<td>0.0009</td>
<td>0.0004</td>
</tr>
<tr>
<td>-0.0005</td>
<td>-0.0001</td>
<td>0.0006</td>
<td>0.0005</td>
</tr>
<tr>
<td>-0.0008</td>
<td>0.0006</td>
<td>-0.0002</td>
<td>-0.0003</td>
</tr>
<tr>
<td>-0.0003</td>
<td>-0.0002</td>
<td>-0.0002</td>
<td>-0.0003</td>
</tr>
<tr>
<td>0.0003</td>
<td>0.0000</td>
<td>-0.0001</td>
<td>0.0003</td>
</tr>
<tr>
<td>-0.0002</td>
<td>-0.0003</td>
<td>0.0000</td>
<td>0.0002</td>
</tr>
<tr>
<td>0.0001</td>
<td>-0.0002</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 4.3 Minimum Phase Sampled Impulse Responses of Channels 1 to 4
Table 4.4 Minimum Phase Sampled Impulse Responses of Channels 5 to 8

<table>
<thead>
<tr>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.5188</td>
<td>0.2131</td>
<td>0.5913</td>
<td>0.1636</td>
</tr>
<tr>
<td>-0.7716</td>
<td>0.0991</td>
<td>0.6865</td>
<td>0.3239</td>
</tr>
<tr>
<td>0.0082</td>
<td>-0.2023</td>
<td>-0.1136</td>
<td>0.0399</td>
</tr>
<tr>
<td>0.1322</td>
<td>-0.0013</td>
<td>-0.0823</td>
<td>-0.0553</td>
</tr>
<tr>
<td>-0.1033</td>
<td>0.0409</td>
<td>0.1182</td>
<td>0.0514</td>
</tr>
<tr>
<td>0.0645</td>
<td>-0.0481</td>
<td>-0.0706</td>
<td>-0.0151</td>
</tr>
<tr>
<td>0.0190</td>
<td>0.0039</td>
<td>-0.0035</td>
<td>-0.0065</td>
</tr>
<tr>
<td>-0.0183</td>
<td>0.0145</td>
<td>0.0266</td>
<td>0.0113</td>
</tr>
<tr>
<td>-0.0011</td>
<td>-0.0118</td>
<td>-0.0127</td>
<td>-0.0025</td>
</tr>
<tr>
<td>0.0053</td>
<td>0.0042</td>
<td>0.0059</td>
<td>0.0037</td>
</tr>
<tr>
<td>-0.0106</td>
<td>0.0073</td>
<td>-0.0013</td>
<td>-0.0037</td>
</tr>
<tr>
<td>0.0065</td>
<td>-0.0140</td>
<td>-0.0007</td>
<td>0.0022</td>
</tr>
<tr>
<td>-0.0009</td>
<td>0.0134</td>
<td>-0.0003</td>
<td>0.0016</td>
</tr>
<tr>
<td>-0.0046</td>
<td>-0.0117</td>
<td>-0.0002</td>
<td>-0.0036</td>
</tr>
<tr>
<td>0.0119</td>
<td>0.0074</td>
<td>-0.0005</td>
<td>0.0028</td>
</tr>
<tr>
<td>-0.0135</td>
<td>0.0006</td>
<td>0.0004</td>
<td>-0.0011</td>
</tr>
<tr>
<td>0.0117</td>
<td>-0.0069</td>
<td>0.0002</td>
<td>0.0003</td>
</tr>
<tr>
<td>-0.0060</td>
<td>0.0108</td>
<td>-0.0003</td>
<td>0.0009</td>
</tr>
<tr>
<td>-0.0009</td>
<td>-0.0105</td>
<td>-0.0001</td>
<td>-0.0016</td>
</tr>
<tr>
<td>0.0070</td>
<td>0.0069</td>
<td>0.0003</td>
<td>0.0011</td>
</tr>
<tr>
<td>-0.0096</td>
<td>-0.0011</td>
<td>0.0001</td>
<td>-0.0003</td>
</tr>
<tr>
<td>0.0071</td>
<td>-0.0031</td>
<td>-0.0009</td>
<td>-0.0006</td>
</tr>
<tr>
<td>-0.0020</td>
<td>0.0058</td>
<td>0.0009</td>
<td>0.0010</td>
</tr>
<tr>
<td>-0.0027</td>
<td>-0.0047</td>
<td>0.0005</td>
<td>-0.0005</td>
</tr>
<tr>
<td>0.0034</td>
<td>0.0020</td>
<td>-0.0011</td>
<td>-0.0010</td>
</tr>
<tr>
<td>-0.0023</td>
<td>0.0019</td>
<td>-0.0009</td>
<td>-0.0004</td>
</tr>
<tr>
<td>0.0010</td>
<td>-0.0029</td>
<td>-0.0003</td>
<td>-0.0001</td>
</tr>
<tr>
<td>0.0027</td>
<td>0.0004</td>
<td>-0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.0011</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>-0.0002</td>
<td>0.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.0001</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4 Minimum Phase Sampled Impulse Responses of Channels 5 to 8
<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 1</th>
<th>Channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>Root 1</td>
<td>1.8111</td>
<td>-4.7911</td>
</tr>
<tr>
<td>Root 2</td>
<td>-1.5808</td>
<td>-1.8673</td>
</tr>
<tr>
<td>Root 3</td>
<td>-1.3452</td>
<td>-0.4516</td>
</tr>
<tr>
<td>Root 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5 Root Positions of the Sampled Impulse Responses of Channels 1 and 2

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>Root 1</td>
<td>-4.0645</td>
<td>1.5207</td>
</tr>
<tr>
<td>Root 2</td>
<td>-2.1505</td>
<td>0.4796</td>
</tr>
<tr>
<td>Root 3</td>
<td>-1.2298</td>
<td>-1.3614</td>
</tr>
<tr>
<td>Root 4</td>
<td>-1.3431</td>
<td>-0.2367</td>
</tr>
<tr>
<td>Root 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6 Root Positions of the Sampled Impulse Responses of Channels 3 and 4

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>Root 1</td>
<td>-3.4199</td>
<td>1.2997</td>
</tr>
<tr>
<td>Root 2</td>
<td>3.1775</td>
<td>-1.2290</td>
</tr>
<tr>
<td>Root 3</td>
<td>0.1372</td>
<td>-2.0145</td>
</tr>
<tr>
<td>Root 4</td>
<td>-0.9299</td>
<td>-1.2328</td>
</tr>
<tr>
<td>Root 5</td>
<td>-1.4458</td>
<td>0.2662</td>
</tr>
<tr>
<td>Root 6</td>
<td>-1.1967</td>
<td>-0.6103</td>
</tr>
<tr>
<td>Root 7</td>
<td>-1.1067</td>
<td>-0.2457</td>
</tr>
<tr>
<td>Root 8</td>
<td>-1.0731</td>
<td>0.0081</td>
</tr>
<tr>
<td>Root 9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7 Root Positions of the Sampled Impulse Responses of Channels 5 and 6
Table 4.8 Root Positions of the Sampled Impulse Responses of Channels 7 and 8

| Channels | Channel 7 | | | Channel 8 | | |
|----------|-----------|-----------|-----------|-----------|-----------|
| Root Number | Real | Imaginary | Modulus | Real | Imaginary | Modulus |
| Root 1 | -6.2917 | 2.1451 | 6.6473 | -1.7503 | 2.2279 | 2.8332 |
| Root 2 | -1.3661 | 1.8652 | 2.3120 | 2.4175 | 0.5847 | 2.4872 |
| Root 3 | -0.2567 | -1.7115 | 1.7306 | -1.6854 | 0.6374 | 1.8019 |
| Root 4 | -1.1909 | 1.0488 | 1.5869 | 1.3611 | -0.8126 | 1.5852 |
| Root 5 | -1.0345 | -1.1205 | 1.5250 | -0.7932 | -1.6344 | 1.8168 |
| Root 6 | -1.1734 | 0.5057 | 1.2777 | 0.6898 | -1.1178 | 1.3135 |
| Root 7 | -1.1367 | 0.2359 | 1.1610 | 0.1999 | -1.1609 | 1.1779 |
| Root 8 | -1.0132 | -0.5504 | 1.1530 | -1.3118 | 0.1038 | 1.3159 |
| Root 9 | -1.1019 | -0.0787 | 1.1043 | -0.1589 | -1.0760 | 1.0877 |
| Root 10 | -1.0730 | -0.0787 | 1.0759 | -0.4420 | -0.9724 | 1.0682 |
| Root 11 | -1.0481 | -0.2404 | 1.0753 | -1.1464 | -0.1367 | 1.1545 |
| Root 12 | -0.9686 | -0.3279 | 1.0226 | -1.0795 | -0.1889 | 1.0959 |

Table 4.9 Sampled Impulse Responses of Truncated Channels 1 to 4

<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.0291</td>
<td>-0.0373</td>
<td>0.0145</td>
<td>-0.0006</td>
</tr>
<tr>
<td>0.2290</td>
<td>-0.2244</td>
<td>0.0750</td>
<td>0.0176</td>
</tr>
<tr>
<td>0.7612</td>
<td>0.1817</td>
<td>0.3951</td>
<td>0.0033</td>
</tr>
<tr>
<td>0.2988</td>
<td>0.3050</td>
<td>0.7491</td>
<td>-0.1718</td>
</tr>
<tr>
<td>-0.0338</td>
<td>-0.2915</td>
<td>0.1951</td>
<td>0.0972</td>
</tr>
<tr>
<td>-0.0789</td>
<td>0.0516</td>
<td>-0.2856</td>
<td>0.1894</td>
</tr>
<tr>
<td>0.0291</td>
<td>0.0287</td>
<td>0.0575</td>
<td>-0.2096</td>
</tr>
<tr>
<td>-0.0137</td>
<td>-0.0332</td>
<td>0.0655</td>
<td>0.1139</td>
</tr>
<tr>
<td>0.0020</td>
<td>0.0204</td>
<td>-0.0825</td>
<td>-0.0424</td>
</tr>
<tr>
<td>0.0004</td>
<td>-0.0108</td>
<td>0.0623</td>
<td>0.0085</td>
</tr>
<tr>
<td>0.0335</td>
<td>-0.0519</td>
<td>-0.0323</td>
<td>0.0170</td>
</tr>
<tr>
<td>0.0008</td>
<td>-0.0051</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4.10 Sampled Impulse Responses of Truncated Channels 5 to 8

<table>
<thead>
<tr>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.0048</td>
<td>0.2550</td>
<td>0.0020</td>
<td>0.1744</td>
</tr>
<tr>
<td>-0.0233</td>
<td>0.0437</td>
<td>0.0167</td>
<td>0.3046</td>
</tr>
<tr>
<td>-0.0040</td>
<td>-0.1928</td>
<td>0.0673</td>
<td>0.0371</td>
</tr>
<tr>
<td>0.2497</td>
<td>0.0229</td>
<td>0.2217</td>
<td>-0.0464</td>
</tr>
<tr>
<td>0.5415</td>
<td>0.0219</td>
<td>0.5021</td>
<td>0.0389</td>
</tr>
<tr>
<td>0.2978</td>
<td>-0.0437</td>
<td>0.5481</td>
<td>-0.0027</td>
</tr>
<tr>
<td>0.0116</td>
<td>0.0091</td>
<td>0.0464</td>
<td>-0.0156</td>
</tr>
<tr>
<td>0.0699</td>
<td>0.0088</td>
<td>-0.2943</td>
<td>0.0141</td>
</tr>
<tr>
<td>-0.1166</td>
<td>-0.0088</td>
<td>0.0065</td>
<td>0.0028</td>
</tr>
<tr>
<td>-0.0515</td>
<td>0.0053</td>
<td>0.1460</td>
<td>-0.0079</td>
</tr>
<tr>
<td>0.1206</td>
<td>0.0006</td>
<td>-0.0701</td>
<td>0.0075</td>
</tr>
<tr>
<td>-0.0806</td>
<td>-0.0020</td>
<td>-0.0513</td>
<td>-0.0007</td>
</tr>
<tr>
<td>-0.0013</td>
<td>0.0005</td>
<td>0.0846</td>
<td>-0.0059</td>
</tr>
<tr>
<td>0.0547</td>
<td>-0.0039</td>
<td>-0.0174</td>
<td>0.0060</td>
</tr>
<tr>
<td>-0.0422</td>
<td>0.0022</td>
<td>-0.0488</td>
<td>0.0020</td>
</tr>
<tr>
<td>0.0076</td>
<td>0.0054</td>
<td>0.0592</td>
<td>-0.0043</td>
</tr>
<tr>
<td>0.0167</td>
<td>0.0013</td>
<td>-0.0296</td>
<td>-0.0029</td>
</tr>
<tr>
<td>-0.0237</td>
<td>-0.0001</td>
<td>-0.0076</td>
<td>-0.0006</td>
</tr>
<tr>
<td>0.0156</td>
<td>-0.0002</td>
<td>0.0283</td>
<td>-0.0001</td>
</tr>
<tr>
<td>0.0018</td>
<td>-0.0001</td>
<td>-0.0318</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

### Table 4.11 Minimum Phase Sampled Impulse Responses of Truncated Channels 1 to 4

<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.5174</td>
<td>-0.6632</td>
<td>0.8667</td>
<td>-0.0359</td>
</tr>
<tr>
<td>-0.0374</td>
<td>-0.5177</td>
<td>0.4535</td>
<td>0.1453</td>
</tr>
<tr>
<td>0.1055</td>
<td>0.0987</td>
<td>-0.1263</td>
<td>0.0159</td>
</tr>
<tr>
<td>-0.0389</td>
<td>0.0138</td>
<td>0.0196</td>
<td>-0.0309</td>
</tr>
<tr>
<td>0.0137</td>
<td>-0.0068</td>
<td>0.0208</td>
<td>0.0294</td>
</tr>
<tr>
<td>0.0011</td>
<td>0.0152</td>
<td>-0.0280</td>
<td>-0.0219</td>
</tr>
<tr>
<td>-0.0059</td>
<td>-0.0065</td>
<td>0.0131</td>
<td>0.0063</td>
</tr>
<tr>
<td>0.0021</td>
<td>0.0039</td>
<td>0.0202</td>
<td>0.0041</td>
</tr>
<tr>
<td>-0.0038</td>
<td>0.0009</td>
<td>0.0036</td>
<td>-0.0014</td>
</tr>
<tr>
<td>0.0000</td>
<td>-0.0006</td>
<td>0.0010</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4.12 Minimum Phase Sampled Impulse Responses of Truncated Channels 5 to 8

<table>
<thead>
<tr>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>-0.6208</td>
<td>0.2550</td>
<td>0.6303</td>
<td>0.1744</td>
</tr>
<tr>
<td>-0.6968</td>
<td>0.0437</td>
<td>0.6575</td>
<td>0.3046</td>
</tr>
<tr>
<td>0.0709</td>
<td>-0.1928</td>
<td>-0.1275</td>
<td>0.0371</td>
</tr>
<tr>
<td>0.0764</td>
<td>0.0229</td>
<td>-0.0690</td>
<td>-0.0464</td>
</tr>
<tr>
<td>-0.0774</td>
<td>0.0219</td>
<td>0.1159</td>
<td>0.0389</td>
</tr>
<tr>
<td>0.0617</td>
<td>-0.0437</td>
<td>-0.0751</td>
<td>-0.0027</td>
</tr>
<tr>
<td>0.0164</td>
<td>0.0091</td>
<td>-0.0007</td>
<td>-0.0156</td>
</tr>
<tr>
<td>-0.0238</td>
<td>0.0088</td>
<td>0.0297</td>
<td>0.0141</td>
</tr>
<tr>
<td>0.0080</td>
<td>-0.0088</td>
<td>-0.0180</td>
<td>0.0028</td>
</tr>
<tr>
<td>-0.0014</td>
<td>0.0053</td>
<td>0.0078</td>
<td>-0.0079</td>
</tr>
<tr>
<td>-0.0084</td>
<td>0.0006</td>
<td>0.0026</td>
<td>0.0075</td>
</tr>
<tr>
<td>0.0066</td>
<td>-0.0020</td>
<td>-0.0070</td>
<td>-0.0007</td>
</tr>
<tr>
<td>0.0003</td>
<td>0.0005</td>
<td>0.0005</td>
<td>-0.0059</td>
</tr>
<tr>
<td>-0.0033</td>
<td>-0.0039</td>
<td>0.0064</td>
<td>0.0060</td>
</tr>
<tr>
<td>0.0014</td>
<td>0.0022</td>
<td>-0.0015</td>
<td>0.0020</td>
</tr>
<tr>
<td>-0.0007</td>
<td>0.0054</td>
<td>-0.0096</td>
<td>-0.0043</td>
</tr>
<tr>
<td>-0.0025</td>
<td>0.0013</td>
<td>-0.0062</td>
<td>-0.0029</td>
</tr>
<tr>
<td>-0.0008</td>
<td>-0.0001</td>
<td>-0.0020</td>
<td>-0.0006</td>
</tr>
<tr>
<td>0.0001</td>
<td>-0.0002</td>
<td>-0.0007</td>
<td>-0.0001</td>
</tr>
<tr>
<td>0.0000</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 4.13 Root Positions of the Sampled Impulse Responses of Truncated Channels 1 and 2

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 1</th>
<th>Channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>1.8111</td>
<td>-1.7858</td>
</tr>
<tr>
<td></td>
<td>-4.7911</td>
<td>-5.1965</td>
</tr>
<tr>
<td></td>
<td>5.1220</td>
<td>5.4948</td>
</tr>
<tr>
<td>Root 2</td>
<td>-1.5808</td>
<td>-0.6153</td>
</tr>
<tr>
<td></td>
<td>-1.8673</td>
<td>3.5564</td>
</tr>
<tr>
<td></td>
<td>2.4466</td>
<td>3.6092</td>
</tr>
<tr>
<td>Root 3</td>
<td>-1.3439</td>
<td>-2.0639</td>
</tr>
<tr>
<td></td>
<td>-0.4513</td>
<td>0.5668</td>
</tr>
<tr>
<td></td>
<td>1.4176</td>
<td>2.1403</td>
</tr>
<tr>
<td>Root 4</td>
<td>-1.3991</td>
<td>0.1249</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.4047</td>
</tr>
</tbody>
</table>
### Table 4.14 Root Positions of the Sampled Impulse Responses of Truncated Channels 3 and 4

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Number</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>Root 1</td>
<td>-4.0645</td>
<td>1.5207</td>
</tr>
<tr>
<td>Root 2</td>
<td>-2.1546</td>
<td>-0.4801</td>
</tr>
<tr>
<td>Root 3</td>
<td>-1.2353</td>
<td>-1.3702</td>
</tr>
<tr>
<td>Root 4</td>
<td>-1.3651</td>
<td>-0.3261</td>
</tr>
<tr>
<td>Root 5</td>
<td>0.6599</td>
<td></td>
</tr>
<tr>
<td>Root 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.15 Root Positions of the Sampled Impulse Responses of Truncated Channels 5 and 6

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Number</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>Root 1</td>
<td>-3.4199</td>
<td>1.2997</td>
</tr>
<tr>
<td>Root 2</td>
<td>3.1775</td>
<td>-1.2290</td>
</tr>
<tr>
<td>Root 3</td>
<td>0.1372</td>
<td>-2.0145</td>
</tr>
<tr>
<td>Root 4</td>
<td>-0.9299</td>
<td>-1.2326</td>
</tr>
<tr>
<td>Root 5</td>
<td>-1.4442</td>
<td>0.2677</td>
</tr>
<tr>
<td>Root 6</td>
<td>-1.1996</td>
<td>-0.6160</td>
</tr>
<tr>
<td>Root 7</td>
<td>-1.1852</td>
<td>-0.2733</td>
</tr>
<tr>
<td>Root 8</td>
<td>-1.1911</td>
<td>0.0489</td>
</tr>
<tr>
<td>Root 9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.16 Root Positions of the Sampled Impulse Responses of Truncated Channels 7 and 8

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Number</td>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>Root 1</td>
<td>-6.2917</td>
<td>2.1451</td>
</tr>
<tr>
<td>Root 2</td>
<td>-1.3661</td>
<td>1.8652</td>
</tr>
<tr>
<td>Root 3</td>
<td>-0.2567</td>
<td>-1.7116</td>
</tr>
<tr>
<td>Root 4</td>
<td>-1.1922</td>
<td>1.0492</td>
</tr>
<tr>
<td>Root 5</td>
<td>-1.0367</td>
<td>-1.1210</td>
</tr>
<tr>
<td>Root 6</td>
<td>-1.2212</td>
<td>0.5084</td>
</tr>
<tr>
<td>Root 7</td>
<td>-1.2861</td>
<td>0.1385</td>
</tr>
<tr>
<td>Root 8</td>
<td>-1.2488</td>
<td>-0.2438</td>
</tr>
<tr>
<td>Root 9</td>
<td>-1.1011</td>
<td>-0.5937</td>
</tr>
<tr>
<td>Root 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.16 Root Positions of the Sampled Impulse Responses of Truncated Channels 7 and 8
Graph GRO.1 Root Positions of Channel 1

- Root inside the unit circle
- Root reflected inside
- Root outside the unit circle
Graph GRO.2 Root Positions of Channel 2
Graph GRO.3 Root Positions of Channel 3

- Root inside the unit circle
- Root reflected inside
- Root outside the unit circle
Graph GRO.4 Root Positions of Channel 4
Graph GRO.5 Root Positions of Channel 5

- Root inside the unit circle
- Root reflected inside
- Root outside the unit circle
Graph GRO.6 Root Positions of Channel 6
Graph GRO.7 Root Positions of Channel 7
Graph GRO.8 Root Positions of Channel 8
Graph GNR 1. Performance of Newton-Raphson over channel 1

Graph GNR 2. Root Accuracy of Newton-Raphson over channel 2

Graph GNR 3. Performance of Newton-Raphson over channel 3

Graph GNR 4a. Performance of Newton-Raphson over channel 4
ADJUSTMENT ALGORITHMS / COMPUTER SIMULATION RESULTS

Graph GNR 4b. Performance of Newton-Raphson over channel 4

Graph GNR 5. Performance of Newton-Raphson over channel 5

Graph GNR 6. Root Accuracy of Newton-Raphson over channel 6

Graph GNR 7. Root Accuracy of Newton-Raphson over channel 7
Graph GNR8. Root Accuracy of Newton-Raphson over channel 8
Graph GL.1a Performance of Laguerre method over Channel 1

Graph GL.1b Performance of Laguerre method over Channel 1

Graph GL.2 Performance of Laguerre method over Channel 2

Graph GL.3 Performance of Laguerre method over Channel 3
ADJUSTMENT ALGORITHMS/COMPUTER SIMULATION-RESULTS

Graph GL.4 Performance of Laguerre method over Channel 4

Graph GL.5 Performance of Laguerre method over Channel 5

Graph GL.6 Performance of Laguerre method over Channel 6

Graph GL.7 Performance of Laguerre method over Channel 7
Graph GL.8 Performance of Laguerre method over Channel 8

Number of Iterations
### Table L.1 Root Accuracy when applying Laguerre method over Channel 1 (\( \varepsilon \) set to \( 10^{-10} \))

| Root Number | Channel 1 | | Channel 2 | | Channel 3 | |
|-------------|-----------|-------------|-----------|-------------|-----------|
| Root 1      | -301.30   | 3           | -284.13   | 3           | -205.28   | 3         |
| Root 2      | -199.12   | 3           | -297.51   | 3           | -193.97   | 3         |
| Root 3      | -101.15   | 2           | -295.45   | 3           | -204.38   | 3         |
| Root 4      | -301.30   | -           | -301.30   | 3           | -219.18   | 3         |

### Table L.2 Performance of Laguerre method over Channel 1 (\( \psi_1 \), \( \psi_2 \) and \( \psi_3 \) for different numbers of prefilter taps)

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_1 )</td>
<td>-56.88</td>
<td>-117.67</td>
<td>-148.71</td>
<td>-179.46</td>
</tr>
<tr>
<td>( \psi_2 )</td>
<td>-75.55</td>
<td>-105.61</td>
<td>-136.14</td>
<td>-166.68</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>-266.30</td>
<td>-266.30</td>
<td>-266.30</td>
<td>-266.30</td>
</tr>
</tbody>
</table>

### Table L.3 Performance of Laguerre method over Channel 2

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_1 )</td>
<td>-49.26</td>
<td>-109.57</td>
<td>-139.72</td>
<td>-139.72</td>
</tr>
<tr>
<td>( \psi_2 )</td>
<td>-114.81</td>
<td>-303.83</td>
<td>-303.83</td>
<td>-303.83</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>-272.49</td>
<td>-272.49</td>
<td>-272.49</td>
<td>-272.49</td>
</tr>
</tbody>
</table>

### Table L.4 Performance of Laguerre method over Channel 3

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_1 )</td>
<td>-44.11</td>
<td>-71.90</td>
<td>-98.10</td>
<td>-125.00</td>
</tr>
<tr>
<td>( \psi_2 )</td>
<td>-95.32</td>
<td>-220.26</td>
<td>-220.26</td>
<td>-220.26</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>-204.43</td>
<td>-204.43</td>
<td>-204.43</td>
<td>-204.43</td>
</tr>
</tbody>
</table>

### Table L.5 Number of Arithmetic Operations when applying Laguerre method over Channels 1, 2 and 3 (\( \varepsilon \) set to \( 10^{-10} \), Number of Prefilter Taps set to 30)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>801</td>
<td>1170</td>
<td>1386</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1560</td>
<td>2306</td>
<td>2714</td>
</tr>
<tr>
<td>Division</td>
<td>24</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>11106</td>
<td>16392</td>
<td>19272</td>
</tr>
</tbody>
</table>
### Channels

<table>
<thead>
<tr>
<th>Root Number</th>
<th>Root Accuracy</th>
<th>Iterations</th>
<th>Root Accuracy</th>
<th>Iterations</th>
<th>Root Accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-283.53</td>
<td>4</td>
<td>-307.50</td>
<td>3</td>
<td>-301.00</td>
<td>3</td>
</tr>
<tr>
<td>Root 2</td>
<td>-288.98</td>
<td>3</td>
<td>-289.43</td>
<td>3</td>
<td>-282.97</td>
<td>4</td>
</tr>
<tr>
<td>Root 3</td>
<td>-283.53</td>
<td>4</td>
<td>-306.25</td>
<td>3</td>
<td>-284.77</td>
<td>4</td>
</tr>
<tr>
<td>Root 4</td>
<td>-288.98</td>
<td>4</td>
<td>-295.00</td>
<td>4</td>
<td>-289.55</td>
<td>4</td>
</tr>
<tr>
<td>Root 5</td>
<td>-313.10</td>
<td>4</td>
<td>-301.30</td>
<td>4</td>
<td>-313.70</td>
<td>4</td>
</tr>
<tr>
<td>Root 6</td>
<td>-290.15</td>
<td>5</td>
<td>-290.15</td>
<td>4</td>
<td>-303.53</td>
<td>3</td>
</tr>
<tr>
<td>Root 7</td>
<td>-299.92</td>
<td>5</td>
<td>-303.52</td>
<td>4</td>
<td>-299.91</td>
<td>5</td>
</tr>
<tr>
<td>Root 8</td>
<td>-293.71</td>
<td>4</td>
<td>-306.56</td>
<td>4</td>
<td>-296.17</td>
<td>7</td>
</tr>
<tr>
<td>Root 9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-292.24</td>
<td>3</td>
</tr>
</tbody>
</table>

Table L.6 Root Accuracy when applying Laguerre method over Channel 4, 5 and 6 (eps set to $10^{-10}$)

### Channel 4

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>$\psi_1$</td>
</tr>
</tbody>
</table>

Table L.7 Performance of Laguerre method over Channel 4 ($\psi_1$, $\psi_2$ and $\psi_3$ for different numbers of prefilter taps)

### Channel 5

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>$\psi_1$</td>
</tr>
<tr>
<td>$\psi_3$</td>
</tr>
</tbody>
</table>

Table L.8 Performance of Laguerre method over Channel 5 ($\psi_1$, $\psi_2$ and $\psi_3$ for different numbers of prefilter taps)
### Table L.9 Performance of Laguerre method over Channel 6 ($\psi_1$, $\psi_2$ and $\psi_3$ for different numbers of prefilter taps)

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-23.70</td>
<td>-23.78</td>
<td>-38.14</td>
<td>-50.49</td>
<td>-69.82</td>
<td>-92.30</td>
<td>-93.35</td>
<td>-103.31</td>
<td>-117.49</td>
</tr>
</tbody>
</table>

### Table L.10 Number of Arithmetic Operations when applying Laguerre method over Channels 4, 5 and 6 (eps set to $10^{-10}$, Number of Prefilter Taps set to 30)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channel 4</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3791</td>
<td>3867</td>
<td>4563</td>
</tr>
<tr>
<td>Multiplication</td>
<td>7565</td>
<td>7649</td>
<td>9090</td>
</tr>
<tr>
<td>Division</td>
<td>99</td>
<td>87</td>
<td>111</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>53566</td>
<td>54150</td>
<td>64332</td>
</tr>
</tbody>
</table>

### Table L.11 Root Accuracy when applying Laguerre method over Channels 7 and 8 (eps set to $10^{-10}$)

<table>
<thead>
<tr>
<th>Root Number</th>
<th>Channel 7 Root Accuracy</th>
<th>Channel 8 Root Accuracy</th>
<th>Channel 7 Iterations</th>
<th>Channel 8 Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-281.95</td>
<td>-279.45</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Root 2</td>
<td>-301.30</td>
<td>-278.75</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Root 3</td>
<td>-219.62</td>
<td>-281.30</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Root 4</td>
<td>-213.84</td>
<td>-291.49</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Root 5</td>
<td>-188.94</td>
<td>-267.94</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Root 6</td>
<td>-209.79</td>
<td>-281.45</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Root 7</td>
<td>-216.18</td>
<td>-280.20</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Root 8</td>
<td>-199.29</td>
<td>-277.84</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Root 9</td>
<td>-293.71</td>
<td>-295.00</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Root 10</td>
<td>-203.77</td>
<td>-278.91</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Root 11</td>
<td>-206.87</td>
<td>-273.51</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Root 12</td>
<td>-210.84</td>
<td>-277.80</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Root 13</td>
<td>-</td>
<td>-296.17</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>
### Table L.12 Performance of Laguerre method over Channel 7 ($\psi_1$, $\psi_2$, and $\psi_3$ for different numbers of prefilter taps)

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-24.42</td>
<td>-32.14</td>
<td>-32.81</td>
<td>-32.81</td>
<td>-43.50</td>
<td>-44.98</td>
<td>-46.53</td>
<td>-52.16</td>
<td>-51.66</td>
</tr>
</tbody>
</table>

### Table L.13 Performance of Laguerre method over Channel 8 ($\psi_1$, $\psi_2$, and $\psi_3$ for different numbers of prefilter taps)

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-29.19</td>
<td>-32.28</td>
<td>-41.70</td>
<td>-47.90</td>
<td>-52.78</td>
<td>-70.10</td>
<td>-66.51</td>
<td>-86.61</td>
<td>-80.20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-50.69</td>
<td>-264.74</td>
<td>-264.74</td>
<td>-264.74</td>
<td>-264.74</td>
<td>-264.74</td>
<td>-264.74</td>
<td>-264.74</td>
<td>-264.74</td>
</tr>
</tbody>
</table>

### Table L.14 Number of Arithmetic Operations when applying Laguerre method over Channels 7 and 8 (eps set to $10^{-10}$, Number of Prefilter Taps set to 50 and 30)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>9981</td>
<td>6610</td>
</tr>
<tr>
<td>Multiplication</td>
<td>18213</td>
<td>12800</td>
</tr>
<tr>
<td>Division</td>
<td>141</td>
<td>162</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>130086</td>
<td>90992</td>
</tr>
</tbody>
</table>

### Table NR.1a Root Accuracy when applying Newton-Raphson method over Channels 1, 2 and 3 ($\delta = 10^{-4}$, eps = $10^{-4}$)

<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta=10^{-4}$</td>
<td>$\varepsilon=10^{-4}$</td>
</tr>
<tr>
<td>S.P=0</td>
<td>S.P=0</td>
</tr>
<tr>
<td>Iterations</td>
<td>Iterations</td>
</tr>
<tr>
<td>Root 1</td>
<td>-184.95</td>
</tr>
<tr>
<td>Root 2</td>
<td>-178.21</td>
</tr>
<tr>
<td>Root 3</td>
<td>-150.73</td>
</tr>
<tr>
<td>Root 4</td>
<td>-297.51</td>
</tr>
</tbody>
</table>
### Table NR.1 Root Accuracy when applying Newton-Raphson method over Channels 1, 2 and 3 ($\delta = 10^{-5}$, $\epsilon = 10^{-5}$)

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Number</td>
<td>Root Accuracy</td>
<td>Iterations</td>
<td>Root Accuracy</td>
</tr>
<tr>
<td>Root 1</td>
<td>-224.66</td>
<td>6</td>
<td>-203.10</td>
</tr>
<tr>
<td>Root 2</td>
<td>-176.00</td>
<td>5</td>
<td>-234.98</td>
</tr>
<tr>
<td>Root 3</td>
<td>-150.67</td>
<td>5</td>
<td>-158.75</td>
</tr>
<tr>
<td>Root 4</td>
<td></td>
<td></td>
<td>-133.93</td>
</tr>
</tbody>
</table>

### Table NR.2 Number of Arithmetic Operations when applying Newton-Raphson method over Channels 1, 2 and 3

<table>
<thead>
<tr>
<th></th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>16</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>730</td>
<td>2100</td>
<td>2569</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1537</td>
<td>2879</td>
<td>3481</td>
</tr>
<tr>
<td>Division</td>
<td>32</td>
<td>58</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>10874</td>
<td>21822</td>
<td>26384</td>
</tr>
</tbody>
</table>

### Table NR.3 Accuracy when applying Newton-Raphson method over Channels 1 and 2 given by $\psi_1$, $\psi_2$ and $\psi_3$ for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 1</th>
<th>Channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Taps in the Prefilter</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>$\psi_1$</td>
<td>-56.88</td>
<td>-117.67</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-122.14</td>
<td>-163.26</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-158.94</td>
<td>-158.93</td>
</tr>
</tbody>
</table>

### Table NR.4 Root Accuracy when applying Newton-Raphson method over Channels 4, 5 and 6

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 4</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Number</td>
<td>Root Accuracy</td>
<td>Iterations</td>
<td>Root Accuracy</td>
</tr>
<tr>
<td>Root 1</td>
<td>-216.41</td>
<td>15</td>
<td>-137.69</td>
</tr>
<tr>
<td>Root 2</td>
<td>-128.93</td>
<td>3</td>
<td>-117.26</td>
</tr>
<tr>
<td>Root 3</td>
<td>-131.63</td>
<td>9</td>
<td>-162.59</td>
</tr>
<tr>
<td>Root 4</td>
<td>-196.54</td>
<td>8</td>
<td>-132.17</td>
</tr>
<tr>
<td>Root 5</td>
<td>-229.74</td>
<td>4</td>
<td>-127.14</td>
</tr>
<tr>
<td>Root 6</td>
<td>-247.89</td>
<td>34</td>
<td>-139.99</td>
</tr>
<tr>
<td>Root 7</td>
<td>F</td>
<td>-</td>
<td>-183.37</td>
</tr>
<tr>
<td>Root 8</td>
<td>F</td>
<td>-</td>
<td>-139.45</td>
</tr>
<tr>
<td>Root 9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table NR.5 Accuracy when applying Newton-Raphson method over Channels 1 and 2 given by $\psi_1$, $\psi_2$ and $\psi_3$ for different numbers of prefilter taps.

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-44.11</td>
<td>-124.98</td>
</tr>
<tr>
<td>30</td>
<td>-98.10</td>
<td>-43.00</td>
</tr>
<tr>
<td>40</td>
<td>-152.19</td>
<td>-152.19</td>
</tr>
<tr>
<td>50</td>
<td>-152.81</td>
<td>-152.81</td>
</tr>
</tbody>
</table>

Table NR.6. Accuracy when applying Newton-Raphson method over Channels 5 and 6 given by $\psi_1$, $\psi_2$ and $\psi_3$ for different numbers of prefilter taps.

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-32.74</td>
<td>-23.95</td>
</tr>
<tr>
<td>30</td>
<td>-28.24</td>
<td>-28.35</td>
</tr>
<tr>
<td>40</td>
<td>-38.76</td>
<td>-39.35</td>
</tr>
<tr>
<td>50</td>
<td>-45.38</td>
<td>-53.43</td>
</tr>
</tbody>
</table>

Table NR.7. Number of Arithmetic Operations when applying Newton-Raphson method over Channels 4, 5 and 6.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channel 4</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>73</td>
<td>53</td>
<td>49</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>6270</td>
<td>5382</td>
<td>4721</td>
</tr>
<tr>
<td>Multiplication</td>
<td>8618</td>
<td>7230</td>
<td>6325</td>
</tr>
<tr>
<td>Division</td>
<td>146</td>
<td>106</td>
<td>98</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>65124</td>
<td>54780</td>
<td>47980</td>
</tr>
</tbody>
</table>
### Channels

<table>
<thead>
<tr>
<th>Root Number</th>
<th>Root Accuracy</th>
<th>Iterations</th>
<th>Root Accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-147.93</td>
<td>9</td>
<td>-93.39</td>
<td>8</td>
</tr>
<tr>
<td>Root 2</td>
<td>-112.79</td>
<td>4</td>
<td>-122.24</td>
<td>9</td>
</tr>
<tr>
<td>Root 3</td>
<td>-117.25</td>
<td>5</td>
<td>-102.49</td>
<td>30</td>
</tr>
<tr>
<td>Root 4</td>
<td>-114.81</td>
<td>5</td>
<td>-101.78</td>
<td>20</td>
</tr>
<tr>
<td>Root 5</td>
<td>-112.82</td>
<td>6</td>
<td>-164.12</td>
<td>8</td>
</tr>
<tr>
<td>Root 6</td>
<td>-125.82</td>
<td>5</td>
<td>-115.00</td>
<td>8</td>
</tr>
<tr>
<td>Root 7</td>
<td>F</td>
<td>-</td>
<td>-126.65</td>
<td>13</td>
</tr>
<tr>
<td>Root 8</td>
<td>-117.64</td>
<td>5</td>
<td>-139.57</td>
<td>7</td>
</tr>
<tr>
<td>Root 9</td>
<td>-180.89</td>
<td>10</td>
<td>-121.82</td>
<td>8</td>
</tr>
<tr>
<td>Root 10</td>
<td>F</td>
<td>-</td>
<td>-126.17</td>
<td>11</td>
</tr>
<tr>
<td>Root 11</td>
<td>F</td>
<td>-</td>
<td>-140.10</td>
<td>15</td>
</tr>
<tr>
<td>Root 12</td>
<td>F</td>
<td>-</td>
<td>-131.41</td>
<td>9</td>
</tr>
<tr>
<td>Root 13</td>
<td></td>
<td>-</td>
<td>-126.00</td>
<td>9</td>
</tr>
</tbody>
</table>

Table NR.8 Root Accuracy when applying Newton-Raphson method over Channels 5 and 6

### Channels

<table>
<thead>
<tr>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Taps in the Prefilter</td>
<td>Number of Taps in the Prefilter</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>$\psi_1$</td>
<td>-27.36</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-33.78</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-14.70</td>
</tr>
</tbody>
</table>

Table NR.9 Accuracy when applying Newton-Raphson method over Channels 7 and 8 given by $\psi_1$, $\psi_2$ and $\psi_3$ for different numbers of prefilter taps

### Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>49</td>
<td>164</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>7088</td>
<td>13705</td>
</tr>
<tr>
<td>Multiplication</td>
<td>9155</td>
<td>19127</td>
</tr>
<tr>
<td>Division</td>
<td>98</td>
<td>328</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>69694</td>
<td>144140</td>
</tr>
</tbody>
</table>

Table NR.10 Number of Arithmetic Operations when applying Newton-Raphson method over Channels 7 and 8
### Channel 1

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\delta=10^{-4}$</td>
</tr>
<tr>
<td>Root 1</td>
<td>-105.55</td>
</tr>
<tr>
<td>Root 2</td>
<td>-57.66</td>
</tr>
<tr>
<td>Root 3</td>
<td>-67.35</td>
</tr>
</tbody>
</table>

Table NRm.1 Root accuracy obtained using Newton-Raphson algorithm over channel 1

### Channel 1

<table>
<thead>
<tr>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-56.88</td>
<td>-75.55</td>
<td>-248.81</td>
</tr>
<tr>
<td>-117.67</td>
<td>-136.14</td>
<td>-248.81</td>
</tr>
<tr>
<td>-148.71</td>
<td>-166.68</td>
<td>-248.81</td>
</tr>
<tr>
<td>-178.47</td>
<td>-197.22</td>
<td>-248.81</td>
</tr>
<tr>
<td>-208.86</td>
<td>-227.78</td>
<td>-248.81</td>
</tr>
<tr>
<td>-238.63</td>
<td>-255.97</td>
<td>-248.81</td>
</tr>
<tr>
<td>-247.32</td>
<td>-261.72</td>
<td>-248.81</td>
</tr>
<tr>
<td>-247.35</td>
<td>-261.64</td>
<td>-248.81</td>
</tr>
</tbody>
</table>

Table NRm.2 Performance of Newton-Raphson, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over channel 1

### Channel 1

<table>
<thead>
<tr>
<th>Operation</th>
<th>$10^{-4}$</th>
<th>$10^{-3}$</th>
<th>$10^{-2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1480</td>
<td>1285</td>
<td>1155</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2011</td>
<td>1729</td>
<td>1541</td>
</tr>
<tr>
<td>Division</td>
<td>40</td>
<td>34</td>
<td>30</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>15266</td>
<td>13148</td>
<td>11736</td>
</tr>
</tbody>
</table>

Table NRm.3 Number of arithmetic operations required by Newton-Raphson over channel 1

### Channel 2

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Root 1</td>
<td>-36.58</td>
</tr>
<tr>
<td>Root 2</td>
<td>-75.16</td>
</tr>
<tr>
<td>Root 3</td>
<td>-54.61</td>
</tr>
<tr>
<td>Root 4</td>
<td>-86.99</td>
</tr>
</tbody>
</table>

Table NRm.4 Root accuracy obtained using Newton-Raphson algorithm over channel 2
Table NRm.5 Performance of Newton-Raphson, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over channel 2

<table>
<thead>
<tr>
<th>Channel 2</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-40.20</td>
<td>-69.42</td>
<td>-109.57</td>
<td>-139.69</td>
<td>-161.62</td>
<td>-162.32</td>
<td>-162.32</td>
<td>-162.32</td>
<td>-162.32</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-86.36</td>
<td>-99.94</td>
<td>-127.62</td>
<td>-158.87</td>
<td>-173.34</td>
<td>-173.32</td>
<td>-173.32</td>
<td>-173.32</td>
<td>-173.32</td>
</tr>
</tbody>
</table>

Table NRm.6 Number of arithmetic operations required by Newton-Raphson algorithm over channel 2

<table>
<thead>
<tr>
<th>Operation</th>
<th>$10^4$</th>
<th>$10^6$</th>
<th>$10^8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2165</td>
<td>1970</td>
<td>1710</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2973</td>
<td>2691</td>
<td>2315</td>
</tr>
<tr>
<td>Division</td>
<td>60</td>
<td>54</td>
<td>46</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>22528</td>
<td>20410</td>
<td>17586</td>
</tr>
</tbody>
</table>

Table NRm.7 Root accuracy obtained using Newton-Raphson algorithm over channel 3

<table>
<thead>
<tr>
<th>Root number</th>
<th>10$^4$</th>
<th>Iterations</th>
<th>10$^6$</th>
<th>Iterations</th>
<th>10$^8$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-35.90</td>
<td>9</td>
<td>-87.33</td>
<td>10</td>
<td>-173.82</td>
<td>12</td>
</tr>
<tr>
<td>Root 2</td>
<td>-90.16</td>
<td>6</td>
<td>-161.87</td>
<td>7</td>
<td>-161.87</td>
<td>7</td>
</tr>
<tr>
<td>Root 3</td>
<td>-75.73</td>
<td>6</td>
<td>-172.19</td>
<td>7</td>
<td>-172.19</td>
<td>7</td>
</tr>
<tr>
<td>Root 4</td>
<td>-96.32</td>
<td>6</td>
<td>-96.32</td>
<td>6</td>
<td>-170.98</td>
<td>7</td>
</tr>
</tbody>
</table>

Table NRm.8 Performance of Newton-Raphson, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over channel 3

<table>
<thead>
<tr>
<th>Channel 3</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-44.11</td>
<td>-60.31</td>
<td>-98.45</td>
<td>-124.99</td>
<td>-151.94</td>
<td>-176.98</td>
<td>-181.47</td>
<td>-181.48</td>
<td>-181.48</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-66.83</td>
<td>-83.73</td>
<td>-120.33</td>
<td>-149.57</td>
<td>-174.30</td>
<td>-191.64</td>
<td>-192.39</td>
<td>-192.36</td>
<td>-192.36</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-182.92</td>
<td>-182.92</td>
<td>-182.92</td>
<td>-182.92</td>
<td>-182.92</td>
<td>-182.92</td>
<td>-182.92</td>
<td>-182.92</td>
<td>-182.92</td>
</tr>
</tbody>
</table>
Channel 3

<table>
<thead>
<tr>
<th>Operation</th>
<th>$10^4$</th>
<th>$10^4$</th>
<th>$10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2338</td>
<td>2569</td>
<td>2800</td>
</tr>
<tr>
<td>Multiplication</td>
<td>3151</td>
<td>3481</td>
<td>3811</td>
</tr>
<tr>
<td>Division</td>
<td>54</td>
<td>60</td>
<td>66</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>23906</td>
<td>26384</td>
<td>28862</td>
</tr>
</tbody>
</table>

Table NRm.9 Number of arithmetic operations required by Newton-Raphson over channel 3

Channel 4

<table>
<thead>
<tr>
<th>$\psi_1$</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>-22.27</td>
<td>-39.24</td>
<td>-51.00</td>
<td>-63.15</td>
<td>-76.29</td>
<td>-82.87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\psi_2$</th>
<th>46.55</th>
<th>66.83</th>
<th>76.92</th>
<th>88.29</th>
<th>101.75</th>
<th>108.49</th>
</tr>
</thead>
<tbody>
<tr>
<td>-136.30</td>
<td>-136.30</td>
<td>-136.30</td>
<td>-136.30</td>
<td>-136.30</td>
<td>-136.30</td>
<td></td>
</tr>
</tbody>
</table>

Table NRm.10 Performance of Newton-Raphson, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over channel 4

Channel 4

<table>
<thead>
<tr>
<th>Root number</th>
<th>$10^4$</th>
<th>$10^4$</th>
<th>$10^4$</th>
<th>$10^4$</th>
<th>$10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-57.48</td>
<td>-130.38</td>
<td>7</td>
<td>-130.38</td>
<td>7</td>
</tr>
<tr>
<td>Root 2</td>
<td>-59.42</td>
<td>-137.99</td>
<td>9</td>
<td>-227.81</td>
<td>11</td>
</tr>
<tr>
<td>Root 3</td>
<td>-60.31</td>
<td>-75.93</td>
<td>4</td>
<td>-130.15</td>
<td>6</td>
</tr>
<tr>
<td>Root 4</td>
<td>-45.71</td>
<td>-102.57</td>
<td>4</td>
<td>-120.95</td>
<td>7</td>
</tr>
<tr>
<td>Root 5</td>
<td>-62.43</td>
<td>-90.21</td>
<td>4</td>
<td>-143.77</td>
<td>13</td>
</tr>
<tr>
<td>Root 6</td>
<td>-44.93</td>
<td>-120.31</td>
<td>6</td>
<td>-120.35</td>
<td>8</td>
</tr>
<tr>
<td>Root 7</td>
<td>-47.74</td>
<td>-95.61</td>
<td>5</td>
<td>-124.59</td>
<td>7</td>
</tr>
<tr>
<td>Root 8</td>
<td>-54.39</td>
<td>-120.31</td>
<td>8</td>
<td>-124.59</td>
<td>7</td>
</tr>
</tbody>
</table>

Table NRm.11 Root accuracy obtained using Newton-Raphson over channel 4

Channel 4

<table>
<thead>
<tr>
<th>Operation</th>
<th>$10^4$</th>
<th>$10^4$</th>
<th>$10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4830</td>
<td>5630</td>
<td>6110</td>
</tr>
<tr>
<td>Multiplication</td>
<td>6566</td>
<td>7706</td>
<td>8390</td>
</tr>
<tr>
<td>Division</td>
<td>110</td>
<td>130</td>
<td>142</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>49716</td>
<td>58276</td>
<td>63412</td>
</tr>
</tbody>
</table>

Table NRm.12 Number of arithmetic operations required by Newton-Raphson over channel 4
### Channel 5

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>Root accuracy</th>
<th>Root accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$10^4$</td>
<td>Iterations</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Root 1</td>
<td>-48.89</td>
<td>6</td>
<td>-98.74</td>
</tr>
<tr>
<td>Root 2</td>
<td>-42.96</td>
<td>5</td>
<td>-110.38</td>
</tr>
<tr>
<td>Root 3</td>
<td>-76.36</td>
<td>6</td>
<td>-145.19</td>
</tr>
<tr>
<td>Root 4</td>
<td>-48.62</td>
<td>7</td>
<td>-86.53</td>
</tr>
<tr>
<td>Root 5</td>
<td>-25.78</td>
<td>5</td>
<td>-120.39</td>
</tr>
<tr>
<td>Root 6</td>
<td>-95.29</td>
<td>7</td>
<td>-118.64</td>
</tr>
<tr>
<td>Root 7</td>
<td>-38.64</td>
<td>4</td>
<td>-93.23</td>
</tr>
<tr>
<td>Root 8</td>
<td>-44.94</td>
<td>5</td>
<td>-109.90</td>
</tr>
</tbody>
</table>

Table NRm.13 Root accuracy obtained using Newton-Raphson over channel 5

### Channel 5

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-28.95</td>
<td>-58.77</td>
<td>-45.57</td>
<td>-49.70</td>
<td>-57.22</td>
<td>-62.94</td>
<td>-68.89</td>
<td>-75.28</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-44.28</td>
<td>-54.99</td>
<td>-60.97</td>
<td>-65.33</td>
<td>-72.89</td>
<td>-78.46</td>
<td>-84.35</td>
<td>-90.71</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-124.70</td>
<td>-124.70</td>
<td>-124.70</td>
<td>-124.70</td>
<td>-124.70</td>
<td>-124.70</td>
<td>-124.70</td>
<td>-124.70</td>
</tr>
</tbody>
</table>

Table NRm.14 Performance of Newton-Raphson, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over channel 5

### Channel 5

<table>
<thead>
<tr>
<th>Operation</th>
<th>$10^4$</th>
<th>$10^4$</th>
<th>$10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4646</td>
<td>5842</td>
<td>6118</td>
</tr>
<tr>
<td>Multiplication</td>
<td>6190</td>
<td>7880</td>
<td>8270</td>
</tr>
<tr>
<td>Division</td>
<td>90</td>
<td>116</td>
<td>122</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>46972</td>
<td>59660</td>
<td>62588</td>
</tr>
</tbody>
</table>

Table NRm.15 Number of arithmetic operations required by Newton-Raphson over channel 5
<table>
<thead>
<tr>
<th>Root number</th>
<th>10⁻⁴</th>
<th>Iterations</th>
<th>Root accuracy</th>
<th>10⁻⁴</th>
<th>Iterations</th>
<th>10⁻⁸</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-25.21</td>
<td>4</td>
<td>-73.93</td>
<td>5</td>
<td>-137.86</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-52.00</td>
<td>6</td>
<td>-93.67</td>
<td>7</td>
<td>-129.24</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-47.83</td>
<td>4</td>
<td>-105.14</td>
<td>5</td>
<td>-105.14</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>-44.19</td>
<td>5</td>
<td>-85.59</td>
<td>6</td>
<td>-185.45</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Root 5</td>
<td>-60.50</td>
<td>6</td>
<td>-141.92</td>
<td>7</td>
<td>-111.96</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Root 6</td>
<td>-33.11</td>
<td>7</td>
<td>-106.84</td>
<td>9</td>
<td>-174.15</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Root 7</td>
<td>F</td>
<td>-</td>
<td>F</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root 8</td>
<td>-31.50</td>
<td>3</td>
<td>-65.10</td>
<td>5</td>
<td>-116.11</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Root 9</td>
<td>-63.82</td>
<td>4</td>
<td>-106.10</td>
<td>5</td>
<td>-109.97</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Table NRm.16 Root accuracy obtained using Newton-Raphson over channel 6

<table>
<thead>
<tr>
<th>Taps</th>
<th>10⁻⁴</th>
<th>10⁻⁸</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-23.96</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>-31.58</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>-39.74</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>-51.69</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>-39.84</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>-22.13</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>-4.53</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>13.10</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>30.64</td>
<td></td>
</tr>
</tbody>
</table>

Table NRm.17 Performance of Newton-Raphson, given by the values of \( \psi_1 \), \( \psi_2 \), and \( \psi_3 \), over channel 6

<table>
<thead>
<tr>
<th>Operation</th>
<th>10⁻⁴</th>
<th>10⁻⁸</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>39</td>
<td>49</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3997</td>
<td>4721</td>
</tr>
<tr>
<td>Multiplication</td>
<td>5221</td>
<td>6325</td>
</tr>
<tr>
<td>Division</td>
<td>78</td>
<td>98</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>39788</td>
<td>47980</td>
</tr>
</tbody>
</table>

Table NRm.18 Number of arithmetic operations required by Newton-Raphson over channel 6
Table NRm.19 Root accuracy obtained using Newton-Raphson algorithm over channel 7

<table>
<thead>
<tr>
<th>Root number</th>
<th>10^4</th>
<th>Iterations</th>
<th>Root accuracy</th>
<th>10^4</th>
<th>Iterations</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-31.47</td>
<td>4</td>
<td>-69.10</td>
<td>5</td>
<td>-147.93</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-39.65</td>
<td>5</td>
<td>-80.11</td>
<td>6</td>
<td>-131.85</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-25.66</td>
<td>4</td>
<td>-75.32</td>
<td>5</td>
<td>-121.25</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>-43.27</td>
<td>5</td>
<td>-109.45</td>
<td>6</td>
<td>-120.28</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Root 5</td>
<td>-36.17</td>
<td>7</td>
<td>-66.15</td>
<td>8</td>
<td>-118.35</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Root 6</td>
<td>F</td>
<td>-</td>
<td>F</td>
<td>-</td>
<td>F</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Root 7</td>
<td>-43.87</td>
<td>15</td>
<td>-110.72</td>
<td>19</td>
<td>-126.85</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Root 8</td>
<td>-59.85</td>
<td>14</td>
<td>-119.40</td>
<td>15</td>
<td>-122.89</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Root 9</td>
<td>-37.31</td>
<td>6</td>
<td>-98.56</td>
<td>8</td>
<td>-125.71</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Root 10</td>
<td>-30.20</td>
<td>3</td>
<td>-77.10</td>
<td>7</td>
<td>-132.38</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Root 11</td>
<td>F</td>
<td>-</td>
<td>F</td>
<td>-</td>
<td>F</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Root 12</td>
<td>-32.71</td>
<td>4</td>
<td>-82.35</td>
<td>7</td>
<td>-141.52</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Table NRm.20 Number of arithmetic operations required by Newton-Raphson over channel 7

<table>
<thead>
<tr>
<th>Operation</th>
<th>10^4</th>
<th>Iterations</th>
<th>10^4</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>67</td>
<td>86</td>
<td>397</td>
<td>13088</td>
</tr>
<tr>
<td>Multiplication</td>
<td>9338</td>
<td>11713</td>
<td>17507</td>
<td>132382</td>
</tr>
<tr>
<td>Division</td>
<td>12287</td>
<td>15593</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>93202</td>
<td>118016</td>
<td>194</td>
<td></td>
</tr>
</tbody>
</table>

Table NRm.21 Root accuracy obtained using Newton-Raphson over channel 8

<table>
<thead>
<tr>
<th>Root number</th>
<th>10^4</th>
<th>Iterations</th>
<th>Root accuracy</th>
<th>10^4</th>
<th>Iterations</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-23.84</td>
<td>4</td>
<td>-118.74</td>
<td>6</td>
<td>-118.74</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-28.92</td>
<td>5</td>
<td>-122.71</td>
<td>7</td>
<td>-122.71</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-30.54</td>
<td>5</td>
<td>-80.96</td>
<td>6</td>
<td>-115.72</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>-27.93</td>
<td>4</td>
<td>-109.33</td>
<td>6</td>
<td>-109.33</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Root 5</td>
<td>-14.28</td>
<td>4</td>
<td>-79.58</td>
<td>6</td>
<td>-164.12</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Root 6</td>
<td>-36.53</td>
<td>6</td>
<td>-104.23</td>
<td>8</td>
<td>-134.98</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Root 7</td>
<td>-38.12</td>
<td>6</td>
<td>-98.23</td>
<td>7</td>
<td>-141.42</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Root 8</td>
<td>-32.91</td>
<td>3</td>
<td>-59.59</td>
<td>5</td>
<td>-139.57</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Root 9</td>
<td>-40.21</td>
<td>5</td>
<td>-90.45</td>
<td>6</td>
<td>-113.31</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Root 10</td>
<td>-35.68</td>
<td>6</td>
<td>-90.12</td>
<td>7</td>
<td>-111.97</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Root 11</td>
<td>-38.99</td>
<td>4</td>
<td>-81.54</td>
<td>8</td>
<td>-118.77</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Root 12</td>
<td>-48.21</td>
<td>11</td>
<td>-110.23</td>
<td>13</td>
<td>-123.73</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Root 13</td>
<td>-47.15</td>
<td>6</td>
<td>-118.32</td>
<td>7</td>
<td>-120.10</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
ADJUSTMENT ALGORITHMS / COMPUTER SIMULATION-RESULTS

### Table NRm.22
Number of arithmetic operations required by Newton-Raphson over Channel 8

<table>
<thead>
<tr>
<th>Operation</th>
<th>10^4</th>
<th>10^4</th>
<th>10^4</th>
<th>10^4</th>
<th>10^4</th>
<th>10^4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>69</td>
<td>92</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>6105</td>
<td>7945</td>
<td>8745</td>
<td>8745</td>
<td>8745</td>
<td>8745</td>
</tr>
<tr>
<td>Multiplication</td>
<td>8297</td>
<td>10919</td>
<td>12059</td>
<td>12059</td>
<td>12059</td>
<td>12059</td>
</tr>
<tr>
<td>Division</td>
<td>138</td>
<td>184</td>
<td>204</td>
<td>204</td>
<td>204</td>
<td>204</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>62820</td>
<td>82508</td>
<td>91068</td>
<td>91068</td>
<td>91068</td>
<td>91068</td>
</tr>
</tbody>
</table>

Table NRm.23
Performance of Newton-Raphson, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$ over channels 7 and 8

<table>
<thead>
<tr>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>-29.61</td>
<td>-36.73</td>
<td>-32.12</td>
<td>-42.71</td>
<td>-29.19</td>
<td>-32.28</td>
<td>-41.70</td>
<td>-47.91</td>
<td></td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-33.87</td>
<td>-43.89</td>
<td>-43.51</td>
<td>-43.57</td>
<td>-50.69</td>
<td>-110.70</td>
<td>-110.70</td>
<td>-110.70</td>
<td></td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-11.78</td>
<td>-11.77</td>
<td>-11.77</td>
<td>-11.77</td>
<td>-11.77</td>
<td>-112.68</td>
<td>-112.70</td>
<td>-112.70</td>
<td></td>
</tr>
</tbody>
</table>

Table AD.1
Number of arithmetic operations required by the adjustment scheme over Channel 1, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>20 30 40 50 60 70 80 90 100</td>
</tr>
<tr>
<td>Multiplication</td>
<td>500 750 1000 1250 1500 1750 2000 2250 2500</td>
</tr>
<tr>
<td>Division</td>
<td>20 30 40 50 60 70 80 90 100</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>3250 4850 6450 8050 9650 11250 12850 14450 16050</td>
</tr>
</tbody>
</table>

Table CAD.1
Number of arithmetic operations required by Clark-Hau adjustment scheme over Channel 1, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>20 30 40 50 60 70 80 90 100</td>
</tr>
<tr>
<td>Multiplication</td>
<td>171 221 271 321 371 421 471 521 571</td>
</tr>
<tr>
<td>Division</td>
<td>20 30 40 50 60 70 80 90 100</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>1398 1818 2238 2658 3078 3498 3918 4338 4758</td>
</tr>
</tbody>
</table>
### Table AD.2
Number of arithmetic operations required by the adjustment scheme over Channel 2, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td></td>
<td>66</td>
<td>86</td>
<td>106</td>
<td>126</td>
<td>146</td>
<td>166</td>
<td>186</td>
<td>206</td>
<td>226</td>
</tr>
<tr>
<td>Multiplication</td>
<td></td>
<td>520</td>
<td>780</td>
<td>1040</td>
<td>1300</td>
<td>1560</td>
<td>1820</td>
<td>2080</td>
<td>2340</td>
<td>2600</td>
</tr>
<tr>
<td>Division</td>
<td></td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td></td>
<td>3372</td>
<td>5032</td>
<td>6692</td>
<td>8352</td>
<td>10012</td>
<td>11672</td>
<td>13332</td>
<td>14992</td>
<td>16652</td>
</tr>
</tbody>
</table>

### Table CAD.2
Number of arithmetic operations required by Clark-Hau adjustment scheme over Channel 1, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td></td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>Multiplication</td>
<td></td>
<td>184</td>
<td>224</td>
<td>264</td>
<td>304</td>
<td>344</td>
<td>384</td>
<td>424</td>
<td>464</td>
<td>504</td>
</tr>
<tr>
<td>Division</td>
<td></td>
<td>251</td>
<td>321</td>
<td>391</td>
<td>461</td>
<td>531</td>
<td>601</td>
<td>671</td>
<td>741</td>
<td>811</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td></td>
<td>3194</td>
<td>3654</td>
<td>4114</td>
<td>4574</td>
<td>5034</td>
<td>5494</td>
<td>5954</td>
<td>6414</td>
<td>6874</td>
</tr>
</tbody>
</table>

### Table AD.3
Number of arithmetic operations required by the adjustment scheme over Channel 3, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td></td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>Multiplication</td>
<td></td>
<td>184</td>
<td>224</td>
<td>264</td>
<td>304</td>
<td>344</td>
<td>384</td>
<td>424</td>
<td>464</td>
<td>504</td>
</tr>
<tr>
<td>Division</td>
<td></td>
<td>251</td>
<td>321</td>
<td>391</td>
<td>461</td>
<td>531</td>
<td>601</td>
<td>671</td>
<td>741</td>
<td>811</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td></td>
<td>3194</td>
<td>3654</td>
<td>4114</td>
<td>4574</td>
<td>5034</td>
<td>5494</td>
<td>5954</td>
<td>6414</td>
<td>6874</td>
</tr>
</tbody>
</table>
Table AD.4 Number of arithmetic operations required by the adjustment scheme over Channel 4, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Addition</td>
<td>75</td>
</tr>
<tr>
<td>Multiplication</td>
<td>700</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>4470</td>
</tr>
</tbody>
</table>

Table CAD.4 Number of arithmetic operations required by Clark-Hau adjustment scheme over Channel 4, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Addition</td>
<td>79</td>
</tr>
<tr>
<td>Multiplication</td>
<td>780</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>4958</td>
</tr>
</tbody>
</table>

Table AD.5 Number of arithmetic operations required by the adjustment scheme over Channel 5, for different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Addition</td>
<td>408</td>
</tr>
<tr>
<td>Multiplication</td>
<td>563</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>4314</td>
</tr>
</tbody>
</table>

Table CAD.5 Number of arithmetic operations required by Clark-Hau adjustment scheme over Channel 5, for different numbers of prefilter taps
### Channel 6

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Addition</td>
<td>78</td>
</tr>
<tr>
<td>Multiplication</td>
<td>760</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>4836</td>
</tr>
</tbody>
</table>

Table AD.6 Number of arithmetic operations required by the adjustment scheme over Channel 6, for different numbers of prefilter taps.

### Channel 7

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Addition</td>
<td>744</td>
</tr>
<tr>
<td>Multiplication</td>
<td>987</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>7530</td>
</tr>
</tbody>
</table>

Table CAD.6 Number of arithmetic operations required by Clark-Hau adjustment scheme over Channel 6, for different numbers of prefilter taps.

### Channel 7

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Addition</td>
<td>94</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1080</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>6788</td>
</tr>
</tbody>
</table>

Table AD.7 Number of arithmetic operations required by the adjustment scheme over Channel 7, for different numbers of prefilter taps.

### Channel 7

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Taps in the Prefilter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Addition</td>
<td>744</td>
</tr>
<tr>
<td>Multiplication</td>
<td>987</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>7530</td>
</tr>
</tbody>
</table>

Table CAD.7 Number of arithmetic operations required by Clark-Hau adjustment scheme over Channel 7, for different numbers of prefilter taps.
Table AD.8 Number of arithmetic operations required by the adjustment scheme over Channel 8, for different numbers of prefitter taps

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Addition</td>
<td>80</td>
<td>100</td>
<td>120</td>
<td>140</td>
<td>160</td>
<td>180</td>
<td>200</td>
<td>220</td>
<td>240</td>
</tr>
<tr>
<td></td>
<td>Multiplication</td>
<td>800</td>
<td>1200</td>
<td>1600</td>
<td>2000</td>
<td>2400</td>
<td>2800</td>
<td>3200</td>
<td>3600</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td>Division</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Total Operational Count</td>
<td>5080</td>
<td>7580</td>
<td>10080</td>
<td>12580</td>
<td>15080</td>
<td>17580</td>
<td>20080</td>
<td>22580</td>
<td>25080</td>
</tr>
</tbody>
</table>

Table CAD.8 Number of arithmetic operations required by Clark-Hau adjustment scheme over Channel 8, for different numbers of prefitter taps

<table>
<thead>
<tr>
<th>Number of Taps in the Prefilter</th>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Addition</td>
<td>611</td>
<td>741</td>
<td>871</td>
<td>1001</td>
<td>1131</td>
<td>1261</td>
<td>1391</td>
<td>1521</td>
<td>1651</td>
</tr>
<tr>
<td></td>
<td>Multiplication</td>
<td>876</td>
<td>1126</td>
<td>1376</td>
<td>1626</td>
<td>1876</td>
<td>2126</td>
<td>2376</td>
<td>2626</td>
<td>2876</td>
</tr>
<tr>
<td></td>
<td>Division</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Total Operational Count</td>
<td>6598</td>
<td>8418</td>
<td>10238</td>
<td>12058</td>
<td>13878</td>
<td>15698</td>
<td>17518</td>
<td>19338</td>
<td>21158</td>
</tr>
</tbody>
</table>

Table CH.1a Root accuracy obtained using Clark-Hau algorithm over channel 1

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-129.25</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-210.34</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-218.22</td>
<td>6</td>
<td>0.5</td>
<td>-218.22</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-210.14</td>
<td>6</td>
<td>0.5</td>
<td>-220.23</td>
<td>4</td>
</tr>
</tbody>
</table>

Table CH.1 Root accuracy obtained using Clark-Hau algorithm over channel 1
Table CH.2  Performance of algorithm 1, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over channel 1

<table>
<thead>
<tr>
<th>Root Startin Point</th>
<th>Root accuracy (d=10^-6)</th>
<th>Iterations</th>
<th>Root Accuracy (d=10^-6)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-56.88</td>
<td>30</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-117.67</td>
<td>-148.71</td>
<td>-178.46</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-75.55</td>
<td>105.61</td>
<td>-136.14</td>
<td>-166.68</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-228.16</td>
<td>228.16</td>
<td>-228.16</td>
<td>-228.16</td>
</tr>
</tbody>
</table>

Table CH.3  Root accuracy obtained using Clark-Hau algorithm over channel 2

<table>
<thead>
<tr>
<th>Channel 2</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-61.33</td>
<td>-79.42</td>
<td>-109.57</td>
<td>-139.61</td>
<td>-155.64</td>
<td>-155.64</td>
<td>-155.80</td>
<td>-155.80</td>
<td>-155.80</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-81.49</td>
<td>-99.95</td>
<td>-127.59</td>
<td>-155.47</td>
<td>-159.24</td>
<td>-159.26</td>
<td>-159.26</td>
<td>-159.26</td>
<td>-159.26</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-158.99</td>
<td>-158.99</td>
<td>-158.99</td>
<td>-158.99</td>
<td>-158.64</td>
<td>-158.80</td>
<td>-158.80</td>
<td>-158.80</td>
<td>-158.80</td>
</tr>
</tbody>
</table>

Table CH.4  Performance of Clark-Hau algorithm, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over Channel 2

<table>
<thead>
<tr>
<th>Channels</th>
<th>1</th>
<th>40</th>
<th>2</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>d=10^-4</td>
<td>c=1.0</td>
<td>d=10^-4</td>
<td>c=1.0</td>
<td>d=10^-4</td>
<td>c=1.0</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>14</td>
<td>16</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2884</td>
<td>2886</td>
<td>2902</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiplication</td>
<td>1767</td>
<td>1865</td>
<td>2663</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Division</td>
<td>54</td>
<td>56</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>16694</td>
<td>17238</td>
<td>22214</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table CH.5  Number of arithmetic operations required by Clark-Hau algorithm over channels 1 and 2
### Table CH.6 Root accuracy obtained using Clark-Hau algorithm over channel 3

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy $10^{-6}$</th>
<th>Iterations</th>
<th>Root accuracy $10^{-6}$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-87.33</td>
<td>9</td>
<td>-173.82</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-101.56</td>
<td>8</td>
<td>-153.75</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-104.68</td>
<td>6</td>
<td>-193.17</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>-133.49</td>
<td>5</td>
<td>-164.84</td>
<td>6</td>
</tr>
</tbody>
</table>

### Table CH.7 Root accuracy obtained using Clark-Hau algorithm over channel 3

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy $d=10^{-6}$</th>
<th>Iterations</th>
<th>Root accuracy $d=10^{-6}$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5+i*0.5</td>
<td>-119.20</td>
<td>12</td>
<td>-247.69</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-153.59</td>
<td>8</td>
<td>-153.59</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>0.5+i*0.5</td>
<td>-122.60</td>
<td>5</td>
<td>-239.93</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>-164.72</td>
<td>6</td>
<td>-164.72</td>
<td>6</td>
</tr>
</tbody>
</table>

### Table CH.8 Performance of Clark-Hau, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over channel 3

<table>
<thead>
<tr>
<th>$\Psi_1$</th>
<th>$\Psi_2$</th>
<th>$\Psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-44.11</td>
<td>-85.14</td>
<td>-173.81</td>
</tr>
<tr>
<td>-55.44</td>
<td>-95.44</td>
<td>-173.81</td>
</tr>
<tr>
<td>-173.81</td>
<td>-173.81</td>
<td>-173.81</td>
</tr>
</tbody>
</table>

### Table CH.9 Root accuracy obtained using Clark-Hau algorithm over channel 4

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-143.34</td>
<td>12</td>
<td>0.5+i*0.5</td>
<td>-161.88</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-157.33</td>
<td>11</td>
<td>0.5+i*0.5</td>
<td>-183.84</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-227.80</td>
<td>10</td>
<td>0.5+i*0.5</td>
<td>-164.91</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>-158.86</td>
<td>5</td>
<td>-0.5</td>
<td>-176.29</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>-169.87</td>
<td>5</td>
<td>-0.5+i*0.5</td>
<td>-184.28</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>-143.77</td>
<td>7</td>
<td>-0.5+i*0.5</td>
<td>-174.96</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>-0.5</td>
<td>-157.32</td>
<td>6</td>
<td>-0.5+i*0.5</td>
<td>-175.65</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>-0.5</td>
<td>-157.32</td>
<td>6</td>
<td>-0.5+i*0.5</td>
<td>-175.65</td>
<td>5</td>
</tr>
</tbody>
</table>
### Table CH.10 Root accuracy obtained using Clark-Hau algorithm over channel 4

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy (d=10^6)</th>
<th>Iterations</th>
<th>Root accuracy (d=10^6)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5+i*0.5</td>
<td>-120.62</td>
<td>9</td>
<td>-161.88</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>-0.5-i*0.5</td>
<td>-183.83</td>
<td>7</td>
<td>-183.84</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0.5-i*0.5</td>
<td>-133.45</td>
<td>7</td>
<td>-164.91</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>-110.25</td>
<td>6</td>
<td>-176.29</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>-0.5+i*0.5</td>
<td>-165.57</td>
<td>5</td>
<td>-184.28</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>-0.5-i*0.5</td>
<td>-174.96</td>
<td>11</td>
<td>-174.96</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>-0.5+i*0.5</td>
<td>-171.79</td>
<td>9</td>
<td>-171.79</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>-0.5+i*0.5</td>
<td>-100.43</td>
<td>3</td>
<td>-175.65</td>
<td>5</td>
</tr>
</tbody>
</table>

### Table CH.11 Performance of Clark-Hau algorithm, given by the values of \(\psi_1\), \(\psi_2\), and \(\psi_3\), over channel 4

<table>
<thead>
<tr>
<th>(\psi)</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\psi_1)</td>
<td>-22.27</td>
<td>-40.29</td>
<td>-39.25</td>
<td>-51.00</td>
<td>-56.27</td>
<td>-64.39</td>
<td>-71.44</td>
<td>-78.00</td>
<td>-85.88</td>
</tr>
<tr>
<td>(\psi_2)</td>
<td>-73.68</td>
<td>-171.70</td>
<td>-171.70</td>
<td>-171.70</td>
<td>-171.70</td>
<td>-171.70</td>
<td>-171.70</td>
<td>-171.70</td>
<td>-171.70</td>
</tr>
<tr>
<td>(\psi_3)</td>
<td>-170.96</td>
<td>-170.96</td>
<td>-170.96</td>
<td>-170.96</td>
<td>-170.96</td>
<td>-170.96</td>
<td>-170.96</td>
<td>-170.96</td>
<td>-170.96</td>
</tr>
</tbody>
</table>

### Table CH.12 Number of arithmetic operations required by Clark-Hau algorithm over channels 3 and 4

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channels</th>
<th>3 (c=1.0)</th>
<th>4 (c=1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(d=10^4)</td>
<td>(d=10^4)</td>
<td>(d=10^4)</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>31</td>
<td>33</td>
<td>57</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4211</td>
<td>4237</td>
<td>4245</td>
</tr>
<tr>
<td>Multiplication</td>
<td>3373</td>
<td>5237</td>
<td>5653</td>
</tr>
<tr>
<td>Division</td>
<td>81</td>
<td>107</td>
<td>115</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>29146</td>
<td>40538</td>
<td>43098</td>
</tr>
</tbody>
</table>
### Channel 5

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
<th>Starting Point</th>
<th>Root Accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5+i*0.5</td>
<td>-148.91</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-91.61</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-187.94</td>
<td>6</td>
<td>0.5+i*0.5</td>
<td>-106.51</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0.5+i*0.5</td>
<td>-199.23</td>
<td>8</td>
<td>0.5+i*0.5</td>
<td>-112.50</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>-0.5+i*0.5</td>
<td>-174.22</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-114.23</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>-0.5+i*0.5</td>
<td>-158.76</td>
<td>6</td>
<td>-0.5+i*0.5</td>
<td>-111.75</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>0.5+i*0.5</td>
<td>-192.34</td>
<td>9</td>
<td>-0.5+i*0.5</td>
<td>-110.65</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td>any</td>
<td>F</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>-175.65</td>
<td>11</td>
<td>0.5+i*0.5</td>
<td>-211.37</td>
<td>9</td>
</tr>
</tbody>
</table>

Table CH.13  Root accuracy obtained using Clark-Hau algorithm over channel 5

### Channel 5

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy d=10^-4</th>
<th>Iterations</th>
<th>Root Accuracy d=10^-9</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5+i*0.5</td>
<td>-108.61</td>
<td>14</td>
<td>-189.54</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>0.5+i*0.5</td>
<td>-135.55</td>
<td>5</td>
<td>-135.55</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0.5+i*0.5</td>
<td>-108.84</td>
<td>9</td>
<td>-213.58</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>-0.5+i*0.5</td>
<td>-151.33</td>
<td>5</td>
<td>-151.33</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>-0.5+i*0.5</td>
<td>-148.75</td>
<td>5</td>
<td>-148.75</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0.5+i*0.5</td>
<td>-115.67</td>
<td>9</td>
<td>-146.90</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td>any</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.5+i*0.5</td>
<td>-127.96</td>
<td>8</td>
<td>-211.37</td>
<td>9</td>
</tr>
</tbody>
</table>

Table CH.14  Root accuracy obtained using Clark-Hau algorithm over channel 5

### Channel 5

<table>
<thead>
<tr>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>-32.83</td>
<td>-9.58</td>
</tr>
<tr>
<td>40</td>
<td>-37.81</td>
<td>-9.58</td>
</tr>
<tr>
<td>50</td>
<td>-49.46</td>
<td>-9.58</td>
</tr>
<tr>
<td>60</td>
<td>-7.77</td>
<td>-9.58</td>
</tr>
<tr>
<td>70</td>
<td>25.68</td>
<td>-9.58</td>
</tr>
</tbody>
</table>

Table CH.15  Performance of Clark-Hau, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over channel 5
<table>
<thead>
<tr>
<th>Channel 6</th>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
<th>Starting Point</th>
<th>Root Accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-157.83</td>
<td>9</td>
<td>0.5+i*0.5</td>
<td>-127.60</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-139.44</td>
<td>7</td>
<td>0.5+i*0.5</td>
<td>-132.89</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-149.61</td>
<td>10</td>
<td>0.5+i*0.5</td>
<td>-134.63</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.5-i*0.5</td>
<td>-141.18</td>
<td>10</td>
<td>-0.5+i*0.5</td>
<td>-126.76</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.5-i*0.5</td>
<td>-165.74</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-143.48</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>-153.52</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-134.65</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>-164.21</td>
<td>10</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>-156.83</td>
<td>14</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td>0.5+i*0.5</td>
<td>-143.26</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Table CH.16  Root accuracy obtained using Clark-Hau algorithm over channel 6

<table>
<thead>
<tr>
<th>Channel 6</th>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
<th>Starting Point</th>
<th>Root Accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-157.83</td>
<td>9</td>
<td>0.5+i*0.5</td>
<td>-127.60</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-139.44</td>
<td>7</td>
<td>0.5+i*0.5</td>
<td>-132.89</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-149.61</td>
<td>10</td>
<td>0.5+i*0.5</td>
<td>-134.63</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.5-i*0.5</td>
<td>-141.18</td>
<td>10</td>
<td>-0.5+i*0.5</td>
<td>-126.76</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.5-i*0.5</td>
<td>-165.74</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-143.48</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>-153.52</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-134.65</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>-164.21</td>
<td>10</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>-156.83</td>
<td>14</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td>0.5+i*0.5</td>
<td>-143.26</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Table CH.17  Root accuracy obtained using Clark-Hau algorithm over channel 6

<table>
<thead>
<tr>
<th>Channel 6</th>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy [d=10^-4]</th>
<th>Iteration s</th>
<th>Root Accuracy [d=10^-4]</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-157.83</td>
<td>9</td>
<td>-157.83</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-139.44</td>
<td>7</td>
<td>-139.44</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-149.61</td>
<td>5</td>
<td>-149.61</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.5-i*0.5</td>
<td>-141.18</td>
<td>10</td>
<td>-141.18</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.5-i*0.5</td>
<td>-165.74</td>
<td>5</td>
<td>-165.74</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>-153.52</td>
<td>5</td>
<td>-153.52</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>-164.21</td>
<td>10</td>
<td>-164.21</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>-156.83</td>
<td>14</td>
<td>-156.83</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td>F</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Table CH.18  Root accuracy obtained using Clark-Hau algorithm over channel 6
## Channel 6

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5-i*0.5</td>
<td>-197.28</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>0.5+i*0.5</td>
<td>-222.68</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>0.5-i*0.5</td>
<td>-242.61</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>0.5+i*0.5</td>
<td>-198.99</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>0.5-i*0.5</td>
<td>-163.60</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>0.5+i*0.5</td>
<td>-193.91</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>any</td>
<td>F</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.5-i*0.5</td>
<td>-141.76</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>any</td>
<td>F</td>
<td>-</td>
</tr>
</tbody>
</table>

Table CH.19 Root accuracy obtained using Clark-Hau algorithm over channel 6

## Channel 6

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-20.29</td>
<td>-33.17</td>
<td>-50.84</td>
<td>-62.79</td>
<td>-79.38</td>
<td>-93.85</td>
<td>-82.19</td>
<td>-69.31</td>
<td>-56.33</td>
</tr>
</tbody>
</table>

Table CH.20 Performance of Clark-Hau, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over channel 6

## Number of arithmetic operations required by Clark-Hau algorithm over channels 5 and 6

<table>
<thead>
<tr>
<th>Channels</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>d=10^4 c=1.0</td>
<td>d=10^8 c=1.0</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>55</td>
<td>59</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>5010</td>
<td>5014</td>
</tr>
<tr>
<td>Multiplication</td>
<td>5973</td>
<td>6221</td>
</tr>
<tr>
<td>Division</td>
<td>105</td>
<td>109</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>46488</td>
<td>48008</td>
</tr>
</tbody>
</table>

Table CH.21 Number of arithmetic operations required by Clark-Hau algorithm over channels 5 and 6
### Table CH.22 Root accuracy obtained using Clark-Hau algorithm over channel 7

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy ( \times 10^4 )</th>
<th>Iterations</th>
<th>Root Accuracy ( \times 10^4 )</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-89.32</td>
<td>8</td>
<td>-186.32</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-120.34</td>
<td>13</td>
<td>-167.39</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>0+0.5</td>
<td>-160.15</td>
<td>4</td>
<td>-177.12</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>-0.5+0.5</td>
<td>-163.88</td>
<td>5</td>
<td>-163.88</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>-102.31</td>
<td>7</td>
<td>-167.37</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>-110.67</td>
<td>9</td>
<td>-187.15</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>-0.5</td>
<td>-151.41</td>
<td>6</td>
<td>-151.41</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>-0.5+0.5</td>
<td>-98.78</td>
<td>10</td>
<td>-157.52</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Table CH.23 Performance of Clark-Hau, given by the values of \( \psi_1, \psi_2 \) and \( \psi_3 \), over channel 7

<table>
<thead>
<tr>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
<th>( \psi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-33.97</td>
<td>-38.98</td>
</tr>
<tr>
<td>30</td>
<td>-47.10</td>
<td>-51.46</td>
</tr>
<tr>
<td>40</td>
<td>-58.84</td>
<td>-64.70</td>
</tr>
<tr>
<td>50</td>
<td>-73.67</td>
<td>-64.10</td>
</tr>
<tr>
<td>60</td>
<td>-52.61</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>-13.69</td>
<td>-13.69</td>
</tr>
<tr>
<td>80</td>
<td>-13.69</td>
<td>-13.69</td>
</tr>
<tr>
<td>90</td>
<td>-13.69</td>
<td>-13.69</td>
</tr>
<tr>
<td>100</td>
<td>-13.69</td>
<td>-13.69</td>
</tr>
</tbody>
</table>

### Table CH.24 Root accuracy obtained using Clark-Hau algorithm over channel 8

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy ( \times 10^4 )</th>
<th>Iterations</th>
<th>Root Accuracy ( \times 10^4 )</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>-111.70</td>
<td>10</td>
<td>-154.29</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-96.12</td>
<td>10</td>
<td>-204.32</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>-0.5</td>
<td>-161.37</td>
<td>7</td>
<td>-161.37</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>-164.61</td>
<td>7</td>
<td>-164.61</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>-173.28</td>
<td>8</td>
<td>-173.28</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>0.5+0.5</td>
<td>-165.29</td>
<td>7</td>
<td>-170.34</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>0.5+0.5</td>
<td>-180.93</td>
<td>8</td>
<td>-180.48</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>-0.5</td>
<td>-105.72</td>
<td>5</td>
<td>-190.93</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>all</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-0.5</td>
<td>-155.43</td>
<td>6</td>
<td>-170.58</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>-0.8</td>
<td>-174.87</td>
<td>5</td>
<td>-174.87</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>0.5</td>
<td>-162.38</td>
<td>4</td>
<td>-175.96</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>-0.5</td>
<td>-119.22</td>
<td>11</td>
<td>-167.90</td>
<td>14</td>
</tr>
<tr>
<td>Channels</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>$d=10^{-6}$</td>
<td>$c=1.0$</td>
<td>$d=10^{-6}$</td>
<td>$c=1.0$</td>
<td>$d=10^{-6}$</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>62</td>
<td>78</td>
<td>62</td>
<td>78</td>
<td>62</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>6722</td>
<td>4413</td>
<td>6722</td>
<td>4413</td>
<td>6722</td>
</tr>
<tr>
<td>Multiplication</td>
<td>8985</td>
<td>7188</td>
<td>8985</td>
<td>7188</td>
<td>8985</td>
</tr>
<tr>
<td>Division</td>
<td>112</td>
<td>128</td>
<td>112</td>
<td>128</td>
<td>112</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>68026</td>
<td>52722</td>
<td>68026</td>
<td>52722</td>
<td>68026</td>
</tr>
</tbody>
</table>

Table CH.26 Number of arithmetic operations required by Clark-Hau algorithm over channels 7 and 8

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-53.68</td>
<td>-58.78</td>
<td>-58.78</td>
</tr>
<tr>
<td>128</td>
<td>-62.95</td>
<td>-309.85</td>
<td>-264.89</td>
</tr>
<tr>
<td>256</td>
<td>-62.95</td>
<td>-312.82</td>
<td>-264.89</td>
</tr>
<tr>
<td>512</td>
<td>-62.95</td>
<td>-312.73</td>
<td>-264.90</td>
</tr>
<tr>
<td>1024</td>
<td>-62.95</td>
<td>-312.71</td>
<td>-264.90</td>
</tr>
</tbody>
</table>

Table CP.1 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$ over channel 1

<table>
<thead>
<tr>
<th>Operation</th>
<th>CHANNEL 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>32</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1927</td>
</tr>
<tr>
<td>Division</td>
<td>1280</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>6068</td>
</tr>
</tbody>
</table>

Table CP.2 Number of arithmetic operations required by Cepstrum algorithm over Channel 1
Table CP.3 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$ over Channel 1, for $L=64$ and different numbers of prefilter taps

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-56.87</td>
<td>-87.27</td>
<td>-117.65</td>
<td>-141.77</td>
<td>-142.94</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-120.32</td>
<td>-147.61</td>
<td>-147.61</td>
<td>-147.61</td>
<td>-147.61</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-147.61</td>
<td>-147.61</td>
<td>-147.61</td>
<td>-147.61</td>
<td>-147.61</td>
</tr>
</tbody>
</table>

Table CP.4 Number of arithmetic operations required by Cepstrum algorithm over Channel 1 for $L=64$ and different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3875</td>
<td>4085</td>
<td>4295</td>
<td>4505</td>
<td>4715</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2074</td>
<td>2294</td>
<td>2514</td>
<td>2734</td>
<td>2954</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>8548</td>
<td>10288</td>
<td>12028</td>
<td>13768</td>
<td>15508</td>
</tr>
</tbody>
</table>

Table CP.5 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 1, for $L=128$ and different numbers of prefilter taps

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-56.87</td>
<td>-87.27</td>
<td>-117.66</td>
<td>-148.05</td>
<td>-178.45</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-120.33</td>
<td>-309.85</td>
<td>-309.85</td>
<td>-309.85</td>
<td>-309.85</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-264.89</td>
<td>-264.89</td>
<td>-264.89</td>
<td>-264.89</td>
<td>-264.89</td>
</tr>
</tbody>
</table>

Table CP.6 Number of arithmetic operations required by Cepstrum algorithm over Channel 1 for $L=128$ and different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>8951</td>
<td>9431</td>
<td>10111</td>
<td>10991</td>
<td>12071</td>
</tr>
<tr>
<td>Multiplication</td>
<td>5278</td>
<td>5778</td>
<td>6478</td>
<td>7378</td>
<td>8478</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>16590</td>
<td>20550</td>
<td>26110</td>
<td>33270</td>
<td>42030</td>
</tr>
</tbody>
</table>
### Table CP.7 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over Channel 2

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-53.22</td>
<td>-56.39</td>
<td>-56.38</td>
</tr>
<tr>
<td>128</td>
<td>-55.29</td>
<td>-244.14</td>
<td>-244.17</td>
</tr>
<tr>
<td>256</td>
<td>-55.29</td>
<td>-309.45</td>
<td>-266.41</td>
</tr>
<tr>
<td>512</td>
<td>-55.28</td>
<td>-315.67</td>
<td>-266.41</td>
</tr>
<tr>
<td>1024</td>
<td>-55.28</td>
<td>-315.75</td>
<td>-266.41</td>
</tr>
</tbody>
</table>

Table CP.8 Number of arithmetic operations required by Cepstrum algorithm over Channel 2 for $L=64$ and different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3875</td>
<td>4085</td>
<td>4295</td>
<td>4505</td>
<td>4715</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2074</td>
<td>2294</td>
<td>2514</td>
<td>2734</td>
<td>2954</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>8548</td>
<td>10288</td>
<td>12028</td>
<td>13768</td>
<td>15508</td>
</tr>
</tbody>
</table>

### Table CP.9 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over Channel 3

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-36.10</td>
<td>-36.37</td>
<td>-36.38</td>
</tr>
<tr>
<td>64</td>
<td>-63.40</td>
<td>-67.24</td>
<td>-66.13</td>
</tr>
<tr>
<td>128</td>
<td>-60.26</td>
<td>-109.60</td>
<td>-108.27</td>
</tr>
<tr>
<td>256</td>
<td>-60.28</td>
<td>-186.19</td>
<td>-185.85</td>
</tr>
<tr>
<td>512</td>
<td>-60.28</td>
<td>-310.11</td>
<td>-252.86</td>
</tr>
<tr>
<td>1024</td>
<td>-60.28</td>
<td>-313.41</td>
<td>-252.86</td>
</tr>
</tbody>
</table>

Table CP.10 Number of arithmetic operations required by Cepstrum algorithm over Channel 3 for $L=64$ and different numbers of prefilter taps
ADJUSTMENT ALGORITHMS / COMPUTER SIMULATION - RESULTS

<table>
<thead>
<tr>
<th>Operation</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2111</td>
<td>4101</td>
<td>9215</td>
<td>19429</td>
<td>45695</td>
<td>95525</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1472</td>
<td>2310</td>
<td>5554</td>
<td>9830</td>
<td>27328</td>
<td>47974</td>
</tr>
<tr>
<td>Division</td>
<td>26</td>
<td>-26</td>
<td>-26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>7588</td>
<td>10416</td>
<td>18774</td>
<td>33264</td>
<td>77028</td>
<td>147504</td>
</tr>
</tbody>
</table>

Table CP.10 Number of arithmetic operations required by Cepstrum algorithm over Channel 3 for different lengths of Fourier Transform

<table>
<thead>
<tr>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>-43.42</td>
<td>-58.61</td>
<td>-59.12</td>
<td>-62.61</td>
<td>-64.26</td>
</tr>
</tbody>
</table>

Table CP.11 Performance of Cepstrum algorithm over Channel 3, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, for $L=64$ and different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3951</td>
<td>4201</td>
<td>4451</td>
<td>4701</td>
<td>4951</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2154</td>
<td>2414</td>
<td>2674</td>
<td>2934</td>
<td>3194</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>9180</td>
<td>11240</td>
<td>13300</td>
<td>15360</td>
<td>17420</td>
</tr>
</tbody>
</table>

Table CP.12 Number of arithmetic operations required by Cepstrum algorithm over Channel 3 for $L=128$ and different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-30.55</td>
<td>-44.11</td>
<td>-43.96</td>
</tr>
<tr>
<td>64</td>
<td>-32.61</td>
<td>-84.13</td>
<td>-84.17</td>
</tr>
<tr>
<td>128</td>
<td>-32.62</td>
<td>-137.87</td>
<td>-134.16</td>
</tr>
<tr>
<td>256</td>
<td>-32.62</td>
<td>-232.89</td>
<td>-228.57</td>
</tr>
<tr>
<td>512</td>
<td>-32.62</td>
<td>-307.42</td>
<td>-266.61</td>
</tr>
<tr>
<td>1024</td>
<td>-32.62</td>
<td>-305.71</td>
<td>-266.61</td>
</tr>
</tbody>
</table>

Table CP.13 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 4 for different lengths of Fourier Transform
Table CP.14 Number of arithmetic operations required by Cepstrum algorithm over Channel 4 for 30 taps and different lengths L of Fourier transform

<table>
<thead>
<tr>
<th>Operation</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2162</td>
<td>4152</td>
<td>9266</td>
<td>19480</td>
<td>45746</td>
<td>95576</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1525</td>
<td>2363</td>
<td>5607</td>
<td>9883</td>
<td>27381</td>
<td>48027</td>
</tr>
<tr>
<td>Division</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>8008</td>
<td>10836</td>
<td>19194</td>
<td>33684</td>
<td>77448</td>
<td>147924</td>
</tr>
</tbody>
</table>

Table CP.15 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$ for $L=64$ and different numbers of prefilter taps, over Channel 4
### Table CP.16 Number of arithmetic operations required by Cepstrum algorithm over Channel 4 for L=64 and different numbers of prefilter taps

<table>
<thead>
<tr>
<th>Operation</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3970</td>
<td>4230</td>
<td>4490</td>
<td>4750</td>
<td>5010</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2174</td>
<td>2444</td>
<td>2714</td>
<td>2754</td>
<td>3024</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>9338</td>
<td>11478</td>
<td>13618</td>
<td>15758</td>
<td>17898</td>
</tr>
</tbody>
</table>

### Table CP.17 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 5

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-26.10</td>
<td>-34.52</td>
<td>-33.21</td>
</tr>
<tr>
<td>64</td>
<td>-27.81</td>
<td>-56.93</td>
<td>-45.23</td>
</tr>
<tr>
<td>128</td>
<td>-28.79</td>
<td>-85.17</td>
<td>-61.58</td>
</tr>
<tr>
<td>256</td>
<td>-28.94</td>
<td>-115.27</td>
<td>-87.72</td>
</tr>
<tr>
<td>512</td>
<td>-28.94</td>
<td>-157.58</td>
<td>-137.14</td>
</tr>
<tr>
<td>1024</td>
<td>-28.94</td>
<td>-231.13</td>
<td>-196.13</td>
</tr>
</tbody>
</table>

### Table CP.18 Number of arithmetic operations required by Cepstrum algorithm over Channel 5 for 30 taps and different lengths L of Fourier transforms

<table>
<thead>
<tr>
<th>Operation</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2386</td>
<td>4376</td>
<td>9490</td>
<td>19704</td>
<td>45970</td>
<td>95800</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1757</td>
<td>2595</td>
<td>35839</td>
<td>70115</td>
<td>27613</td>
<td>48259</td>
</tr>
<tr>
<td>Division</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>9848</td>
<td>12676</td>
<td>21034</td>
<td>35524</td>
<td>79288</td>
<td>149764</td>
</tr>
</tbody>
</table>

### Table CP.19 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 6

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-21.49</td>
<td>-32.69</td>
<td>-32.66</td>
</tr>
<tr>
<td>64</td>
<td>-22.10</td>
<td>-70.89</td>
<td>-61.93</td>
</tr>
<tr>
<td>128</td>
<td>-21.99</td>
<td>-102.25</td>
<td>-102.10</td>
</tr>
<tr>
<td>256</td>
<td>-21.99</td>
<td>-156.82</td>
<td>-140.48</td>
</tr>
<tr>
<td>512</td>
<td>-21.99</td>
<td>-255.41</td>
<td>-238.00</td>
</tr>
<tr>
<td>1024</td>
<td>-21.99</td>
<td>-307.10</td>
<td>-245.84</td>
</tr>
</tbody>
</table>
### Table CP.20 Number of arithmetic operations required by Cepstrum algorithm over Channel 6 for 30 taps and different lengths L of Fourier transforms

<table>
<thead>
<tr>
<th>Operation</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2270</td>
<td>4260</td>
<td>9374</td>
<td>19588</td>
<td>45854</td>
<td>95684</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1637</td>
<td>2475</td>
<td>5719</td>
<td>9995</td>
<td>27493</td>
<td>48139</td>
</tr>
<tr>
<td>Division</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>8896</td>
<td>11724</td>
<td>20082</td>
<td>34572</td>
<td>78336</td>
<td>148812</td>
</tr>
</tbody>
</table>

### Table CP.21 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 7

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>-26.73</td>
<td>-47.68</td>
<td>-47.50</td>
</tr>
<tr>
<td>128</td>
<td>26.16</td>
<td>89.13</td>
<td>77.57</td>
</tr>
<tr>
<td>256</td>
<td>-26.13</td>
<td>-121.31</td>
<td>-114.34</td>
</tr>
<tr>
<td>512</td>
<td>-26.13</td>
<td>-178.78</td>
<td>-161.67</td>
</tr>
<tr>
<td>1024</td>
<td>-26.13</td>
<td>-284.43</td>
<td>-173.71</td>
</tr>
</tbody>
</table>

### Table CP.22 Number of arithmetic operations required by Cepstrum algorithm over Channel 7 for 50 taps and different lengths L of Fourier transforms

<table>
<thead>
<tr>
<th>Operation</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>5157</td>
<td>10271</td>
<td>20485</td>
<td>46751</td>
<td>96581</td>
</tr>
<tr>
<td>Multiplication</td>
<td>3398</td>
<td>6642</td>
<td>10918</td>
<td>28416</td>
<td>49062</td>
</tr>
<tr>
<td>Division</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>19056</td>
<td>27414</td>
<td>41904</td>
<td>85668</td>
<td>156114</td>
</tr>
</tbody>
</table>

### Table CP.23 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 8

<table>
<thead>
<tr>
<th>Length of Fourier Transform</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-19.33</td>
<td>-35.18</td>
<td>-35.17</td>
</tr>
<tr>
<td>64</td>
<td>21.16</td>
<td>-87.86</td>
<td>-85.62</td>
</tr>
<tr>
<td>128</td>
<td>21.13</td>
<td>-139.20</td>
<td>-129.63</td>
</tr>
<tr>
<td>256</td>
<td>-21.13</td>
<td>-218.87</td>
<td>-211.51</td>
</tr>
<tr>
<td>512</td>
<td>-21.13</td>
<td>-301.74</td>
<td>-267.83</td>
</tr>
<tr>
<td>1024</td>
<td>-21.13</td>
<td>-313.51</td>
<td>-267.82</td>
</tr>
</tbody>
</table>

Table CP.23 Performance of Cepstrum algorithm, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 8
Table CP.24 Number of arithmetic operations required by Cepstrum algorithm over Channel 8 for 50 taps and different lengths L of Fourier transforms.

<table>
<thead>
<tr>
<th>Operation</th>
<th>50 taps</th>
<th>96 taps</th>
<th>128 taps</th>
<th>200 taps</th>
<th>1050 taps</th>
<th>21456 taps</th>
<th>22636 taps</th>
<th>9286 taps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>6136</td>
<td>8604</td>
<td>10478</td>
<td>13766</td>
<td>24132</td>
<td>25118</td>
<td>10478</td>
<td>9286</td>
</tr>
<tr>
<td>Multiplication</td>
<td>6998</td>
<td>9726</td>
<td>10478</td>
<td>13766</td>
<td>24132</td>
<td>25118</td>
<td>10478</td>
<td>9286</td>
</tr>
<tr>
<td>Division</td>
<td>176</td>
<td>208</td>
<td>216</td>
<td>248</td>
<td>464</td>
<td>336</td>
<td>216</td>
<td></td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>55316</td>
<td>76812</td>
<td>82736</td>
<td>108632</td>
<td>197996</td>
<td>82736</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tables LE.1 Number of arithmetic operations required by the Levinson algorithm over channels 1-8.

Table LE.2 Root accuracy of Levinson algorithm, over channels 1-3.

Table LE.3 Root accuracy of Levinson algorithm, over channels 4, 5 and 6.
### Table LE.4 Root accuracy of Levinson algorithm, over channel 6

<table>
<thead>
<tr>
<th>Root Number</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
<th>3 Iteration</th>
<th>4 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-3.91</td>
<td>-4.77</td>
<td>-4.28</td>
<td>-4.89</td>
</tr>
<tr>
<td>2</td>
<td>-4.76</td>
<td>-5.76</td>
<td>-5.38</td>
<td>-5.91</td>
</tr>
<tr>
<td>3</td>
<td>-5.31</td>
<td>-6.28</td>
<td>-10.80</td>
<td>-6.18</td>
</tr>
<tr>
<td>4</td>
<td>-10.25</td>
<td>-11.96</td>
<td>-12.11</td>
<td>-12.10</td>
</tr>
<tr>
<td>5</td>
<td>-10.53</td>
<td>-12.69</td>
<td>-15.74</td>
<td>-12.71</td>
</tr>
<tr>
<td>6</td>
<td>-14.12</td>
<td>-17.89</td>
<td>-24.59</td>
<td>-17.53</td>
</tr>
<tr>
<td>7</td>
<td>-24.80</td>
<td>-32.41</td>
<td>-30.54</td>
<td>-41.91</td>
</tr>
<tr>
<td>8</td>
<td>-25.19</td>
<td>-31.39</td>
<td>-30.61</td>
<td>-30.76</td>
</tr>
<tr>
<td>9</td>
<td>-38.11</td>
<td>-31.34</td>
<td>-44.80</td>
<td>-28.36</td>
</tr>
</tbody>
</table>

### Table LE.5 Root accuracy of Levinson algorithm, over channels 7 and 8

<table>
<thead>
<tr>
<th>Root</th>
<th>Channel 7 1 Iteration</th>
<th>Channel 7 2 Iteration</th>
<th>Channel 8 1 Iteration</th>
<th>Channel 8 2 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2.38</td>
<td>-2.48</td>
<td>-5.86</td>
<td>-10.88</td>
</tr>
<tr>
<td>2</td>
<td>-6.46</td>
<td>-6.48</td>
<td>-7.48</td>
<td>-6.85</td>
</tr>
<tr>
<td>3</td>
<td>-10.21</td>
<td>-10.24</td>
<td>-10.70</td>
<td>-10.69</td>
</tr>
<tr>
<td>4</td>
<td>-11.17</td>
<td>-10.60</td>
<td>-14.41</td>
<td>-13.35</td>
</tr>
<tr>
<td>5</td>
<td>-12.32</td>
<td>-13.92</td>
<td>-10.96</td>
<td>-11.63</td>
</tr>
<tr>
<td>6</td>
<td>-17.51</td>
<td>-17.95</td>
<td>-22.64</td>
<td>-26.48</td>
</tr>
<tr>
<td>7</td>
<td>-25.16</td>
<td>-40.16</td>
<td>-37.86</td>
<td>-22.14</td>
</tr>
<tr>
<td>8</td>
<td>-30.71</td>
<td>-29.73</td>
<td>-21.97</td>
<td>-21.36</td>
</tr>
<tr>
<td>9</td>
<td>-47.58</td>
<td>-41.22</td>
<td>-38.88</td>
<td>-17.11</td>
</tr>
<tr>
<td>10</td>
<td>-40.51</td>
<td>-37.70</td>
<td>-25.71</td>
<td>-16.95</td>
</tr>
<tr>
<td>11</td>
<td>-48.00</td>
<td>-33.57</td>
<td>-25.86</td>
<td>-25.26</td>
</tr>
<tr>
<td>12</td>
<td>-21.40</td>
<td>-18.55</td>
<td>-18.84</td>
<td>-20.72</td>
</tr>
</tbody>
</table>

### Table LE.6 Accuracy of Levinson algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channels 1, 2 and 3

<table>
<thead>
<tr>
<th>Channel 1 1 Iteration</th>
<th>Channel 2 1 Iteration</th>
<th>Channel 3 1 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$ -33.39</td>
<td>-14.43</td>
<td>-14.28</td>
</tr>
<tr>
<td>$\psi_2$ -6.82</td>
<td>-8.50</td>
<td>-3.35</td>
</tr>
<tr>
<td>$\psi_3$ -9.24</td>
<td>-8.35</td>
<td>-3.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel 1 2 Iteration</th>
<th>Channel 2 2 Iteration</th>
<th>Channel 3 2 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$ -62.81</td>
<td>-60.19</td>
<td>-41.10</td>
</tr>
<tr>
<td>$\psi_2$ -80.25</td>
<td>-60.34</td>
<td>-42.42</td>
</tr>
<tr>
<td>$\psi_3$ -60.63</td>
<td>-60.34</td>
<td>-42.37</td>
</tr>
</tbody>
</table>
Table LE.7 Accuracy of Levinson algorithm, given by $\psi_1$, $\psi_2$, and $\psi_3$, over Channels 4 and 5

<table>
<thead>
<tr>
<th>Operation</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>138.24</td>
<td>-38.23</td>
<td>33.53</td>
<td>-30.00</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>1.10</td>
<td>-50.34</td>
<td>-6.34</td>
<td>-32.74</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-2.43</td>
<td>-50.30</td>
<td>-6.66</td>
<td>-32.89</td>
</tr>
</tbody>
</table>

Table LE.8 Accuracy of Levinson algorithm, given by $\psi_1$, $\psi_2$, and $\psi_3$, over Channels 6

<table>
<thead>
<tr>
<th>Operation</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
<th>3 Iteration</th>
<th>4 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-12.27</td>
<td>-23.71</td>
<td>-11.51</td>
<td>-23.72</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-1.51</td>
<td>-40.37</td>
<td>1.46</td>
<td>-41.49</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-1.53</td>
<td>-40.84</td>
<td>1.49</td>
<td>-41.45</td>
</tr>
</tbody>
</table>

Table LE.9 Accuracy of Levinson algorithm, given by $\psi_1$, $\psi_2$, and $\psi_3$, over Channels 7 and 8

<table>
<thead>
<tr>
<th>Operation</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>10.97</td>
<td>32.64</td>
<td>103.17</td>
<td>-31.40</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>0.46</td>
<td>51.00</td>
<td>1.52</td>
<td>48.36</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>0.30</td>
<td>51.24</td>
<td>-2.76</td>
<td>48.47</td>
</tr>
</tbody>
</table>

Table LE.10 Number of operations required by the Levinson algorithm over channels 1, 2, 3 and 4

<table>
<thead>
<tr>
<th>Operation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2440</td>
<td>2440</td>
<td>3404</td>
<td>3670</td>
</tr>
<tr>
<td>Multiplication</td>
<td>3218</td>
<td>3218</td>
<td>4426</td>
<td>4578</td>
</tr>
<tr>
<td>Division</td>
<td>92</td>
<td>92</td>
<td>108</td>
<td>112</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>24740</td>
<td>24740</td>
<td>34012</td>
<td>36560</td>
</tr>
</tbody>
</table>

Table LE.11 Number of operations required by the Levinson algorithm over channels 5, 6, 7 and 8

<table>
<thead>
<tr>
<th>Operation</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4834</td>
<td>8464</td>
<td>8860</td>
<td>3670</td>
</tr>
<tr>
<td>Multiplication</td>
<td>6206</td>
<td>10916</td>
<td>11178</td>
<td>4758</td>
</tr>
<tr>
<td>Division</td>
<td>128</td>
<td>240</td>
<td>172</td>
<td>112</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>47672</td>
<td>83864</td>
<td>85820</td>
<td>36560</td>
</tr>
<tr>
<td>Iterations</td>
<td>Channel 1</td>
<td>Channel 2</td>
<td>Channel 3</td>
<td>Channel 4</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>5</td>
<td>-97.16</td>
<td>-103.17</td>
<td>-71.92</td>
<td>-66.93</td>
</tr>
<tr>
<td>10</td>
<td>-132.49</td>
<td>-133.52</td>
<td>-81.24</td>
<td>-68.39</td>
</tr>
<tr>
<td>15</td>
<td>-162.36</td>
<td>-166.90</td>
<td>-88.77</td>
<td>-69.92</td>
</tr>
<tr>
<td>20</td>
<td>-197.69</td>
<td>-197.26</td>
<td>-95.88</td>
<td>-71.72</td>
</tr>
<tr>
<td>22</td>
<td>-210.73</td>
<td>-210.75</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>-227.57</td>
<td>-230.64</td>
<td>-102.27</td>
<td>-73.28</td>
</tr>
<tr>
<td>26</td>
<td>-</td>
<td>-103.63</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-73.88</td>
</tr>
<tr>
<td>50</td>
<td>-302.53</td>
<td>-307.17</td>
<td>-134.56</td>
<td>-80.13</td>
</tr>
<tr>
<td>70</td>
<td>-301.74</td>
<td>-309.29</td>
<td>-160.75</td>
<td>-84.58</td>
</tr>
<tr>
<td>100</td>
<td>-302.32</td>
<td>-307.82</td>
<td>-200.21</td>
<td>-90.83</td>
</tr>
</tbody>
</table>

Table NIT.1 Accuracy of NIT algorithm given by $\psi$, over channels 1-4

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-52.12</td>
<td>-62.88</td>
<td>-62.88</td>
<td>-53.48</td>
</tr>
<tr>
<td>10</td>
<td>-57.10</td>
<td>-65.68</td>
<td>-65.61</td>
<td>-55.14</td>
</tr>
<tr>
<td>15</td>
<td>-56.37</td>
<td>-67.89</td>
<td>-67.87</td>
<td>-56.36</td>
</tr>
<tr>
<td>20</td>
<td>-58.93</td>
<td>-67.92</td>
<td>-68.59</td>
<td>-57.49</td>
</tr>
<tr>
<td>25</td>
<td>-58.15</td>
<td>-69.96</td>
<td>-69.65</td>
<td>-58.35</td>
</tr>
<tr>
<td>27</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-93.65</td>
</tr>
<tr>
<td>31</td>
<td>-58.94</td>
<td>-70.47</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>42</td>
<td>-</td>
<td>-</td>
<td>-92.68</td>
<td>-</td>
</tr>
<tr>
<td>50</td>
<td>-60.32</td>
<td>-71.72</td>
<td>-73.85</td>
<td>-60.65</td>
</tr>
<tr>
<td>70</td>
<td>-62.36</td>
<td>-74.21</td>
<td>-73.98</td>
<td>-61.88</td>
</tr>
<tr>
<td>100</td>
<td>-63.63</td>
<td>-77.31</td>
<td>-79.85</td>
<td>-69.23</td>
</tr>
</tbody>
</table>

Table NIT.2 Accuracy of NIT algorithm given by $\psi$, over channels 5-8

<table>
<thead>
<tr>
<th>Operation</th>
<th>Itrs=5</th>
<th>Itrs=10</th>
<th>Itrs=15</th>
<th>Itrs=22</th>
<th>Itrs=25</th>
<th>Itrs=50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4410</td>
<td>8820</td>
<td>6930</td>
<td>10164</td>
<td>11550</td>
<td>23100</td>
</tr>
<tr>
<td>Multiplication</td>
<td>4620</td>
<td>9240</td>
<td>13860</td>
<td>20328</td>
<td>23100</td>
<td>46200</td>
</tr>
<tr>
<td>Division</td>
<td>110</td>
<td>220</td>
<td>330</td>
<td>484</td>
<td>550</td>
<td>1100</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>36540</td>
<td>73080</td>
<td>99000</td>
<td>145200</td>
<td>165000</td>
<td>330000</td>
</tr>
</tbody>
</table>

Table NIT.3 Number of arithmetic operations required by the NIT algorithm, over channel 1, for different numbers of iterations Itrs
### Table NIT.4 Number of arithmetic operations required by the NIT algorithm, over channel 3, for different numbers of iterations \( Itrs \)

<table>
<thead>
<tr>
<th>Operation</th>
<th>( Itrs=5 )</th>
<th>( Itrs=10 )</th>
<th>( Itrs=15 )</th>
<th>( Itrs=20 )</th>
<th>( Itrs=26 )</th>
<th>( Itrs=50 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3250</td>
<td>6500</td>
<td>9750</td>
<td>13000</td>
<td>16900</td>
<td>32500</td>
</tr>
<tr>
<td>Multiplication</td>
<td>6500</td>
<td>13000</td>
<td>19500</td>
<td>26000</td>
<td>33800</td>
<td>65000</td>
</tr>
<tr>
<td>Division</td>
<td>130</td>
<td>260</td>
<td>390</td>
<td>520</td>
<td>676</td>
<td>1300</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>46280</td>
<td>92560</td>
<td>138840</td>
<td>185120</td>
<td>240656</td>
<td>462800</td>
</tr>
</tbody>
</table>

### Table NIT.5 Number of arithmetic operations required by the NIT algorithm, over channel 4, for different numbers of iterations \( Itrs \)

<table>
<thead>
<tr>
<th>Operation</th>
<th>( Itrs=5 )</th>
<th>( Itrs=10 )</th>
<th>( Itrs=15 )</th>
<th>( Itrs=20 )</th>
<th>( Itrs=27 )</th>
<th>( Itrs=50 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>18954</td>
<td>7020</td>
<td>10530</td>
<td>14040</td>
<td>18954</td>
<td>35100</td>
</tr>
<tr>
<td>Multiplication</td>
<td>37908</td>
<td>14040</td>
<td>21060</td>
<td>28080</td>
<td>37908</td>
<td>70200</td>
</tr>
<tr>
<td>Division</td>
<td>729</td>
<td>270</td>
<td>405</td>
<td>540</td>
<td>729</td>
<td>1350</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>269730</td>
<td>99900</td>
<td>149850</td>
<td>199800</td>
<td>269730</td>
<td>499500</td>
</tr>
</tbody>
</table>

### Table NIT.6 Number of arithmetic operations required by the NIT algorithm, over channel 5, for different numbers of iterations \( Itrs \)

<table>
<thead>
<tr>
<th>Operation</th>
<th>( Itrs=5 )</th>
<th>( Itrs=10 )</th>
<th>( Itrs=15 )</th>
<th>( Itrs=20 )</th>
<th>( Itrs=31 )</th>
<th>( Itrs=50 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4650</td>
<td>9300</td>
<td>13950</td>
<td>18600</td>
<td>28830</td>
<td>46500</td>
</tr>
<tr>
<td>Multiplication</td>
<td>9300</td>
<td>18600</td>
<td>27900</td>
<td>37200</td>
<td>57660</td>
<td>93000</td>
</tr>
<tr>
<td>Division</td>
<td>155</td>
<td>310</td>
<td>465</td>
<td>620</td>
<td>961</td>
<td>1550</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>66030</td>
<td>132060</td>
<td>198090</td>
<td>264120</td>
<td>409386</td>
<td>660300</td>
</tr>
</tbody>
</table>

### Table NIT.7 Number of arithmetic operations required by the NIT algorithm, over channel 6, for different numbers of iterations \( Itrs \)

<table>
<thead>
<tr>
<th>Operation</th>
<th>( Itrs=5 )</th>
<th>( Itrs=10 )</th>
<th>( Itrs=15 )</th>
<th>( Itrs=20 )</th>
<th>( Itrs=29 )</th>
<th>( Itrs=50 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4060</td>
<td>8120</td>
<td>12180</td>
<td>16240</td>
<td>23548</td>
<td>40600</td>
</tr>
<tr>
<td>Multiplication</td>
<td>8120</td>
<td>16240</td>
<td>24360</td>
<td>32480</td>
<td>47096</td>
<td>81200</td>
</tr>
<tr>
<td>Division</td>
<td>145</td>
<td>8120</td>
<td>435</td>
<td>580</td>
<td>841</td>
<td>1450</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>57710</td>
<td>115420</td>
<td>173130</td>
<td>230840</td>
<td>334718</td>
<td>577100</td>
</tr>
</tbody>
</table>
## Table NIT.8 Number of arithmetic operations required by the NIT algorithm, over channel 7, for different numbers of iterations Itrs

<table>
<thead>
<tr>
<th>Operation</th>
<th>Itrs=5</th>
<th>Itrs=10</th>
<th>Itrs=15</th>
<th>Itrs=20</th>
<th>Itrs=42</th>
<th>Itrs=50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>8610</td>
<td>17220</td>
<td>25830</td>
<td>34440</td>
<td>72324</td>
<td>86100</td>
</tr>
<tr>
<td>Multiplication</td>
<td>17220</td>
<td>34440</td>
<td>51660</td>
<td>68880</td>
<td>144648</td>
<td>172200</td>
</tr>
<tr>
<td>Division</td>
<td>210</td>
<td>420</td>
<td>630</td>
<td>840</td>
<td>1764</td>
<td>2100</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>121800</td>
<td>243600</td>
<td>365400</td>
<td>487200</td>
<td>1023120</td>
<td>1218000</td>
</tr>
</tbody>
</table>

## Table NITT.1 Accuracy, of NIT algorithm given by \( \psi \), over Channels 1 and 2 (for different numbers of taps, L, and iterations, Itrs)

<table>
<thead>
<tr>
<th>L</th>
<th>Channel 1</th>
<th>Channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Itrs=5</td>
<td>Itrs=10</td>
</tr>
<tr>
<td>2</td>
<td>-74.48</td>
<td>-74.67</td>
</tr>
<tr>
<td>5</td>
<td>-94.13</td>
<td>-101.28</td>
</tr>
<tr>
<td>8</td>
<td>-123.24</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-96.80</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>-96.78</td>
<td>-128.38</td>
</tr>
<tr>
<td>15</td>
<td>-95.67</td>
<td>-90.55</td>
</tr>
</tbody>
</table>

## Table NITT.2 Accuracy, of NIT algorithm given by \( \psi \), over Channel 1 (number or iterations, set to Itrs = 5 and different numbers of filter taps, L)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Itrs=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>190</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2300</td>
</tr>
<tr>
<td>Division</td>
<td>100</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>14780</td>
</tr>
</tbody>
</table>

## Table NITT.3 Accuracy, of NIT algorithm given by \( \psi \), over Channel 2 (number or iterations, set to Itrs = 10 and different numbers of filter taps, L)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Itrs=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>380</td>
</tr>
<tr>
<td>Multiplication</td>
<td>200</td>
</tr>
<tr>
<td>Division</td>
<td>200</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>29560</td>
</tr>
</tbody>
</table>
### Table NITT.4 Accuracy of NIT algorithm given by $\eta$, over Channel 3 (number of iterations, set to Itrs = 5, 10 and different numbers of filter taps, $L$)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Itrs=5</th>
<th>Itrs=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L=2$</td>
<td>$L=5$</td>
<td>$L=8$</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>6480</td>
<td>6300</td>
</tr>
<tr>
<td>Multiplication</td>
<td>460</td>
<td>1000</td>
</tr>
<tr>
<td>Division</td>
<td>240</td>
<td>210</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>41240</td>
<td>41060</td>
</tr>
</tbody>
</table>

### Table NITT.5 Number of arithmetic operations, required by NIT algorithm, over channel 3 (number of iterations set to Itrs = 10 and different numbers of filter taps, $L$)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Itrs=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L=2$</td>
<td>$L=5$</td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>480</td>
</tr>
<tr>
<td>Multiplication</td>
<td>7000</td>
</tr>
<tr>
<td>Division</td>
<td>250</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>44460</td>
</tr>
</tbody>
</table>

### Table NITT.6 Accuracy of NIT algorithm, given by $\eta$, over Channel 4 (different number of iterations and different numbers of filter taps $L$)

Table NITT.7, Number of arithmetic operations, required by NIT algorithm, over Channel 4 (Number of Iterations set to Itrs =10 and different numbers of filter taps $L$)
### Table NITT.8
Number of arithmetic operations, required by NIT algorithm, over Channels 5 and 6
(Number of Iterations set to \( \text{Itrs} = 5, 10 \) and different numbers of filter taps, \( L \))

<table>
<thead>
<tr>
<th>Lengths</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-51.58</td>
<td>-53.81</td>
</tr>
<tr>
<td>8</td>
<td>-51.67</td>
<td>-55.60</td>
</tr>
<tr>
<td>9</td>
<td>-51.93</td>
<td>-55.88</td>
</tr>
<tr>
<td>10</td>
<td>-52.15</td>
<td>-56.11</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>-52.36</td>
<td>-57.77</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>-52.30</td>
<td>-53.84</td>
</tr>
<tr>
<td>25</td>
<td>-51.75</td>
<td>-51.88</td>
</tr>
</tbody>
</table>

### Table NITT.9
Number of arithmetic operations required by NIT algorithm, over Channel 5 (Number of Iterations set to \( \text{Itrs} = 10 \) and different numbers of filter taps \( L \))

<table>
<thead>
<tr>
<th>Operation</th>
<th>( L = 2 )</th>
<th>( L = 5 )</th>
<th>( L = 8 )</th>
<th>( L = 9 )</th>
<th>( L = 12 )</th>
<th>( L = 15 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>560</td>
<td>1250</td>
<td>1760</td>
<td>1890</td>
<td>2160</td>
<td>22250</td>
</tr>
<tr>
<td>Multiplication</td>
<td>9280</td>
<td>9100</td>
<td>8740</td>
<td>8580</td>
<td>7980</td>
<td>7200</td>
</tr>
<tr>
<td>Division</td>
<td>290</td>
<td>260</td>
<td>230</td>
<td>220</td>
<td>190</td>
<td>160</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>58540</td>
<td>58660</td>
<td>57340</td>
<td>56580</td>
<td>53340</td>
<td>48660</td>
</tr>
</tbody>
</table>

### Table NITT.10
Number of arithmetic operations required by NIT algorithm, over Channel 6 (Number of Iterations set to \( \text{Itrs} = 10 \) and different numbers of filter taps \( L \))

<table>
<thead>
<tr>
<th>Operation</th>
<th>( L = 2 )</th>
<th>( L = 5 )</th>
<th>( L = 8 )</th>
<th>( L = 9 )</th>
<th>( L = 12 )</th>
<th>( L = 15 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>260</td>
<td>575</td>
<td>800</td>
<td>855</td>
<td>960</td>
<td>975</td>
</tr>
<tr>
<td>Multiplication</td>
<td>4050</td>
<td>3960</td>
<td>3780</td>
<td>3700</td>
<td>3400</td>
<td>3010</td>
</tr>
<tr>
<td>Division</td>
<td>135</td>
<td>120</td>
<td>105</td>
<td>100</td>
<td>85</td>
<td>70</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>25630</td>
<td>25630</td>
<td>24910</td>
<td>24510</td>
<td>22830</td>
<td>20430</td>
</tr>
</tbody>
</table>

### Table NITT.11
Accuracy of NIT algorithm, given by \( \psi \), over Channel 7 (different numbers of Iterations \( \text{Itrs} \) and different numbers of filter taps, \( L \))

<table>
<thead>
<tr>
<th>Lengths</th>
<th>Channel 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-60.73</td>
</tr>
<tr>
<td>8</td>
<td>-62.91</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Table NITT.11 Accuracy of NIT algorithm, given by \( \psi \), over Channel 7 (different numbers of Iterations \( \text{Itrs} \) and different numbers of filter taps, \( L \))
### Table NIT.12 Number of arithmetic operations required by NIT algorithm, over Channel 7 (Number of Iterations set to Itrs = 15 and different numbers of filter taps, L)

<table>
<thead>
<tr>
<th>Operation</th>
<th>L=2</th>
<th>L=5</th>
<th>L=8</th>
<th>L=9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1170</td>
<td>2700</td>
<td>3960</td>
<td>4320</td>
</tr>
<tr>
<td>Multiplication</td>
<td>25800</td>
<td>25530</td>
<td>24990</td>
<td>24750</td>
</tr>
<tr>
<td>Division</td>
<td>600</td>
<td>555</td>
<td>510</td>
<td>495</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>160740</td>
<td>161910</td>
<td>160920</td>
<td>160110</td>
</tr>
</tbody>
</table>

### Table NIT.13 Accuracy of NIT algorithm, given by $\psi$, over Channel 8 (different numbers of Iterations Itrs and filter taps, L)

<table>
<thead>
<tr>
<th>Lengths</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-53.79</td>
</tr>
<tr>
<td>8</td>
<td>-53.45</td>
</tr>
<tr>
<td>9</td>
<td>-53.46</td>
</tr>
</tbody>
</table>

### Table NIT.14 Number of arithmetic operations required by NIT algorithm, over Channel 8 (Number of Iterations set to Itrs = 15 and different numbers of filter taps, L)

<table>
<thead>
<tr>
<th>Operation</th>
<th>L=2</th>
<th>L=5</th>
<th>L=8</th>
<th>L=9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1170</td>
<td>2700</td>
<td>3960</td>
<td>4320</td>
</tr>
<tr>
<td>Multiplication</td>
<td>25800</td>
<td>25530</td>
<td>24990</td>
<td>24750</td>
</tr>
<tr>
<td>Division</td>
<td>600</td>
<td>555</td>
<td>510</td>
<td>495</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>160740</td>
<td>161910</td>
<td>160920</td>
<td>160110</td>
</tr>
</tbody>
</table>

### CHANNEL 1

<table>
<thead>
<tr>
<th>Prefilter Taps</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-117.67</td>
<td>-148.70</td>
<td>-178.46</td>
<td>-208.86</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-195.28</td>
<td>-246.31</td>
<td>-262.28</td>
<td>-318.79</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-195.10</td>
<td>-247.43</td>
<td>-266.00</td>
<td>-266.00</td>
</tr>
</tbody>
</table>

Table GS.1 Accuracy of Gram-Schmidt algorithm, given by $\psi_1$, $\psi_2$, and $\psi_3$, over Channel 1 (different numbers of filter taps)
### CHANNEL 2

<table>
<thead>
<tr>
<th></th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-79.42</td>
<td>-109.57</td>
<td>-139.72</td>
<td>-169.87</td>
<td>-200.00</td>
<td>-230.17</td>
<td>-260.32</td>
<td>-290.46</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-126.45</td>
<td>-163.00</td>
<td>-199.47</td>
<td>-235.99</td>
<td>-272.50</td>
<td>-308.39</td>
<td>-317.19</td>
<td>-317.00</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-125.13</td>
<td>-161.69</td>
<td>-198.18</td>
<td>-234.65</td>
<td>-267.27</td>
<td>-282.30</td>
<td>-272.49</td>
<td>-272.50</td>
</tr>
</tbody>
</table>

Table GS.2 Accuracy of Gram-Schmidt algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 2 (different numbers of filter taps)

### CHANNEL 3

<table>
<thead>
<tr>
<th></th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-61.92</td>
<td>-74.49</td>
<td>-86.00</td>
<td>-96.95</td>
<td>-107.65</td>
<td>-118.27</td>
<td>-128.86</td>
<td>-139.46</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-73.32</td>
<td>-84.27</td>
<td>-94.93</td>
<td>-105.56</td>
<td>-116.15</td>
<td>-126.74</td>
<td>-137.33</td>
<td>-147.91</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-61.38</td>
<td>-72.00</td>
<td>-82.68</td>
<td>-93.27</td>
<td>-103.86</td>
<td>-114.45</td>
<td>-125.00</td>
<td>-135.63</td>
</tr>
</tbody>
</table>

Table GS.3 Accuracy of Gram-Schmidt algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 3 (different numbers of filter taps)

### CHANNEL 4

<table>
<thead>
<tr>
<th></th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-40.52</td>
<td>-39.32</td>
<td>-51.10</td>
<td>-54.38</td>
<td>-63.15</td>
<td>-68.85</td>
<td>-76.30</td>
<td>-82.88</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-79.39</td>
<td>-94.22</td>
<td>-105.27</td>
<td>-121.94</td>
<td>-133.81</td>
<td>-148.10</td>
<td>-175.36</td>
<td>-175.36</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-72.85</td>
<td>-84.55</td>
<td>-100.18</td>
<td>-113.27</td>
<td>-127.15</td>
<td>-140.72</td>
<td>-168.11</td>
<td>-168.11</td>
</tr>
</tbody>
</table>

Table GS.4 Accuracy of Gram-Schmidt algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 4 (different numbers of filter taps)

### CHANNEL 5

<table>
<thead>
<tr>
<th></th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-38.10</td>
<td>-40.79</td>
<td>-44.72</td>
<td>-46.34</td>
<td>-61.81</td>
<td>-76.98</td>
<td>-76.98</td>
<td></td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-50.80</td>
<td>-61.30</td>
<td>-69.47</td>
<td>-74.00</td>
<td>-77.21</td>
<td>-79.50</td>
<td>-68.95</td>
<td></td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-41.30</td>
<td>-47.45</td>
<td>-50.96</td>
<td>-53.88</td>
<td>-56.61</td>
<td>-59.41</td>
<td>-62.31</td>
<td></td>
</tr>
</tbody>
</table>

Table GS.5 Accuracy of Gram-Schmidt algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channel 5 (different numbers of filter taps)
### Table GS.6 Accuracy of Gram-Schmidt algorithm, given by \( \psi_1 \), \( \psi_2 \) and \( \psi_3 \), over Channel 6 (different numbers of filter taps)

<table>
<thead>
<tr>
<th>Operation</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1611</td>
<td>2541</td>
<td>3600</td>
<td>5001</td>
<td>6531</td>
<td>8261</td>
<td>10191</td>
<td>12321</td>
</tr>
<tr>
<td>Multiplication</td>
<td>79560</td>
<td>151280</td>
<td>255600</td>
<td>398520</td>
<td>586040</td>
<td>824160</td>
<td>1118880</td>
<td>1476200</td>
</tr>
<tr>
<td>Division</td>
<td>1560</td>
<td>2480</td>
<td>3671</td>
<td>4920</td>
<td>6440</td>
<td>8160</td>
<td>10080</td>
<td>1476200</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>510750</td>
<td>957410</td>
<td>1602870</td>
<td>2483130</td>
<td>3634190</td>
<td>5092050</td>
<td>6892710</td>
<td>9072170</td>
</tr>
</tbody>
</table>

### Table GS.7 Accuracy of Gram-Schmidt algorithm, given by \( \psi_1 \), \( \psi_2 \) and \( \psi_3 \), over Channel 7 (different numbers of filter taps)

<table>
<thead>
<tr>
<th>Operation</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1611</td>
<td>2541</td>
<td>3600</td>
<td>5001</td>
<td>6531</td>
<td>8261</td>
<td>10191</td>
<td>12321</td>
</tr>
<tr>
<td>Multiplication</td>
<td>79560</td>
<td>151280</td>
<td>255600</td>
<td>398520</td>
<td>586040</td>
<td>824160</td>
<td>1118880</td>
<td>1476200</td>
</tr>
<tr>
<td>Division</td>
<td>1560</td>
<td>2480</td>
<td>3671</td>
<td>4920</td>
<td>6440</td>
<td>8160</td>
<td>10080</td>
<td>1476200</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>510750</td>
<td>957410</td>
<td>1602870</td>
<td>2483130</td>
<td>3634190</td>
<td>5092050</td>
<td>6892710</td>
<td>9072170</td>
</tr>
</tbody>
</table>

### Table GS.8 Accuracy of Gram-Schmidt algorithm, given by \( \psi_1 \), \( \psi_2 \) and \( \psi_3 \), over Channel 8 (different numbers of filter taps)

<table>
<thead>
<tr>
<th>Operation</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1611</td>
<td>2541</td>
<td>3600</td>
<td>5001</td>
<td>6531</td>
<td>8261</td>
<td>10191</td>
<td>12321</td>
</tr>
<tr>
<td>Multiplication</td>
<td>79560</td>
<td>151280</td>
<td>255600</td>
<td>398520</td>
<td>586040</td>
<td>824160</td>
<td>1118880</td>
<td>1476200</td>
</tr>
<tr>
<td>Division</td>
<td>1560</td>
<td>2480</td>
<td>3671</td>
<td>4920</td>
<td>6440</td>
<td>8160</td>
<td>10080</td>
<td>1476200</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>510750</td>
<td>957410</td>
<td>1602870</td>
<td>2483130</td>
<td>3634190</td>
<td>5092050</td>
<td>6892710</td>
<td>9072170</td>
</tr>
</tbody>
</table>

### Table GS.9 Number of arithmetic operations required by Gram-Schmidt algorithm over Channel 1 (different numbers of filter taps)
### Channel 3

<table>
<thead>
<tr>
<th>Operation</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1735</td>
<td>2705</td>
<td>3875</td>
<td>5245</td>
<td>6815</td>
<td>8585</td>
<td>10556</td>
<td>12725</td>
</tr>
<tr>
<td>Multiplication</td>
<td>92400</td>
<td>171600</td>
<td>285000</td>
<td>438600</td>
<td>638400</td>
<td>890400</td>
<td>1200600</td>
<td>1575000</td>
</tr>
<tr>
<td>Division</td>
<td>1680</td>
<td>2640</td>
<td>3800</td>
<td>5160</td>
<td>6720</td>
<td>8480</td>
<td>10440</td>
<td>12600</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>592150</td>
<td>1084650</td>
<td>1785550</td>
<td>2730850</td>
<td>3956550</td>
<td>5498650</td>
<td>7393150</td>
<td>9676050</td>
</tr>
</tbody>
</table>

Table GS.10 Number of arithmetic operations required by Gram-Schmidt algorithm over Channel 3 (different numbers of filter taps)

### Channel 4

<table>
<thead>
<tr>
<th>Operation</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1766</td>
<td>2746</td>
<td>3926</td>
<td>5306</td>
<td>6886</td>
<td>8666</td>
<td>10646</td>
<td>12826</td>
</tr>
<tr>
<td>Multiplication</td>
<td>95760</td>
<td>176880</td>
<td>292600</td>
<td>448920</td>
<td>651840</td>
<td>907360</td>
<td>1221480</td>
<td>1600200</td>
</tr>
<tr>
<td>Division</td>
<td>1710</td>
<td>2680</td>
<td>3850</td>
<td>5220</td>
<td>6790</td>
<td>8560</td>
<td>10530</td>
<td>12700</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>613440</td>
<td>1117700</td>
<td>1832760</td>
<td>2794620</td>
<td>4039280</td>
<td>5602740</td>
<td>7521000</td>
<td>9830060</td>
</tr>
</tbody>
</table>

Table GS.11 Number of arithmetic operations required by Gram-Schmidt algorithm over Channel 4 (different numbers of filter taps)

### Channel 5

<table>
<thead>
<tr>
<th>Operation</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>2910</td>
<td>4130</td>
<td>5550</td>
<td>7170</td>
<td>8990</td>
<td>11010</td>
<td>13230</td>
</tr>
<tr>
<td>Multiplication</td>
<td>198800</td>
<td>324000</td>
<td>491400</td>
<td>707000</td>
<td>976800</td>
<td>1306800</td>
<td>1703000</td>
</tr>
<tr>
<td>Division</td>
<td>2840</td>
<td>4050</td>
<td>5460</td>
<td>7070</td>
<td>8880</td>
<td>10890</td>
<td>13100</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>1254860</td>
<td>2027760</td>
<td>3057060</td>
<td>4378760</td>
<td>6028860</td>
<td>8043360</td>
<td>10458260</td>
</tr>
</tbody>
</table>

Table GS.12 Number of arithmetic operations required by Gram-Schmidt algorithm over Channel 5 (different numbers of filter taps)

### Channel 6

<table>
<thead>
<tr>
<th>Operation</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1828</td>
<td>2828</td>
<td>4028</td>
<td>5428</td>
<td>7028</td>
<td>8828</td>
<td>10828</td>
<td>13028</td>
</tr>
<tr>
<td>Multiplication</td>
<td>102660</td>
<td>187680</td>
<td>308100</td>
<td>469920</td>
<td>679140</td>
<td>941760</td>
<td>1263780</td>
<td>1651200</td>
</tr>
<tr>
<td>Division</td>
<td>1770</td>
<td>2760</td>
<td>3950</td>
<td>5340</td>
<td>6930</td>
<td>8720</td>
<td>10710</td>
<td>12900</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>657148</td>
<td>1185288</td>
<td>1929028</td>
<td>2924368</td>
<td>42027308</td>
<td>5813848</td>
<td>7779988</td>
<td>10141728</td>
</tr>
</tbody>
</table>

Table GS.13 Number of arithmetic operations required by Gram-Schmidt algorithm over Channel 6 (different numbers of filter taps)
### Channel 7

<table>
<thead>
<tr>
<th>Operation</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4640</td>
<td>6221</td>
<td>7951</td>
<td>9881</td>
<td>12011</td>
<td>14341</td>
</tr>
<tr>
<td>Multiplication</td>
<td>40950</td>
<td>618120</td>
<td>870240</td>
<td>1180960</td>
<td>1556280</td>
<td>2002200</td>
</tr>
<tr>
<td>Division</td>
<td>4550</td>
<td>6120</td>
<td>7840</td>
<td>9760</td>
<td>11880</td>
<td>14200</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>2558380</td>
<td>3839490</td>
<td>5382950</td>
<td>7281210</td>
<td>9570270</td>
<td>12286130</td>
</tr>
</tbody>
</table>

Table GS.14 Number of arithmetic operations required by Gram-Schmidt algorithm over Channel 7 (different numbers of filter taps)

### Channel 8

<table>
<thead>
<tr>
<th>Operation</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1766</td>
<td>2787</td>
<td>3977</td>
<td>5367</td>
<td>7028</td>
<td>8666</td>
<td>10646</td>
<td>12725</td>
</tr>
<tr>
<td>Multiplication</td>
<td>95760</td>
<td>182240</td>
<td>300300</td>
<td>459360</td>
<td>679140</td>
<td>907360</td>
<td>1221480</td>
<td>1575000</td>
</tr>
<tr>
<td>Division</td>
<td>1710</td>
<td>2720</td>
<td>3900</td>
<td>5280</td>
<td>6930</td>
<td>8560</td>
<td>10530</td>
<td>12600</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>613440</td>
<td>1151246</td>
<td>1880586</td>
<td>2859126</td>
<td>4207308</td>
<td>5602740</td>
<td>7521000</td>
<td>9676050</td>
</tr>
</tbody>
</table>

Table GS.15 Number of arithmetic operations required by Gram-Schmidt algorithm over Channel 8 (different numbers of filter taps)

### Table B.1a

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-56.88</td>
<td>-49.26</td>
<td>-14.69</td>
<td>-4.48</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-92.93</td>
<td>-95.11</td>
<td>-8.47</td>
<td>-11.23</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-87.65L</td>
<td>-92.67U</td>
<td>-95.38U</td>
<td>-8.32</td>
</tr>
</tbody>
</table>

Table B.1a Accuracy of Bauer method, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over Channels 1-5

### Table B.1

<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-87.28</td>
<td>-55.29</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-266.13</td>
<td>-126.10</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-132.65</td>
<td>-115.19</td>
</tr>
</tbody>
</table>

Table B.1 Accuracy of Bauer method, given by the values of $\psi_1$, $\psi_2$, and $\psi_3$, over Channels 1, 2 and 3 when adding a number of zeros
### Table B.2 Number of Arithmetic Operations required by the Bauer method over Channels 1-4

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
<th>Total Operational Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>5576</td>
<td>5576</td>
<td>9101</td>
<td>10172</td>
<td>57826</td>
</tr>
<tr>
<td>Multiplication</td>
<td>7716</td>
<td>7716</td>
<td>11850</td>
<td>13082</td>
<td>10656</td>
</tr>
<tr>
<td>Division</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>57826</td>
<td>57826</td>
<td>89742</td>
<td>99291</td>
<td></td>
</tr>
</tbody>
</table>

### Table B.3 Number of Arithmetic Operations required by the Bauer method over Channels 5-8

<table>
<thead>
<tr>
<th>Operation</th>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
<th>Total Operational Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>15317</td>
<td>12559</td>
<td>37722</td>
<td>10172</td>
<td>148287</td>
</tr>
<tr>
<td>Multiplication</td>
<td>19515</td>
<td>15805</td>
<td>45182</td>
<td>13082</td>
<td>169767</td>
</tr>
<tr>
<td>Division</td>
<td>40</td>
<td>30</td>
<td>50</td>
<td>27</td>
<td>162</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>148287</td>
<td>120441</td>
<td>347246</td>
<td>99291</td>
<td></td>
</tr>
</tbody>
</table>

### Table NITTs.1 Accuracy of NIT algorithm, given by \( \psi \), over Channels 1-4 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
<th>Channel 4 [20]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-61.79</td>
<td>-66.10</td>
<td>-41.73</td>
<td>-26.24</td>
<td>-28.44</td>
</tr>
<tr>
<td>6</td>
<td>-72.73</td>
<td>-76.11</td>
<td>-46.56</td>
<td>-27.48</td>
<td>-29.83</td>
</tr>
<tr>
<td>10</td>
<td>-97.73</td>
<td>-103.10</td>
<td>-59.53</td>
<td>-30.22</td>
<td>-32.97</td>
</tr>
<tr>
<td>15</td>
<td>-125.65</td>
<td>-133.50</td>
<td>-74.18</td>
<td>-32.44</td>
<td>-35.61</td>
</tr>
<tr>
<td>20</td>
<td>-161.44</td>
<td>-170.54</td>
<td>-90.92</td>
<td>-34.40</td>
<td>-38.00</td>
</tr>
<tr>
<td>22</td>
<td>-174.32</td>
<td>-184.10</td>
<td>-97.17</td>
<td>-35.20</td>
<td>-38.78</td>
</tr>
<tr>
<td>25</td>
<td>-189.48</td>
<td>-200.95</td>
<td>-105.45</td>
<td>-35.77</td>
<td>-39.74</td>
</tr>
<tr>
<td>27</td>
<td>-202.14</td>
<td>-214.44</td>
<td>-111.70</td>
<td>-36.31</td>
<td>-40.43</td>
</tr>
<tr>
<td>50</td>
<td>-314.10</td>
<td>-315.55</td>
<td>-184.65</td>
<td>-41.17</td>
<td>-46.91</td>
</tr>
<tr>
<td>70</td>
<td>-315.41</td>
<td>-315.66</td>
<td>-247.15</td>
<td>-44.50</td>
<td>-51.33</td>
</tr>
<tr>
<td>100</td>
<td>-315.41</td>
<td>-314.47</td>
<td>-311.98</td>
<td>-48.35</td>
<td>-57.16</td>
</tr>
</tbody>
</table>
### Table NITs.2 Accuracy of NIT algorithm, given by $\psi$, over Channels 5-8 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-33.31</td>
<td>-31.19</td>
<td>-34.95</td>
<td>-24.30</td>
</tr>
<tr>
<td>10</td>
<td>-40.75</td>
<td>-35.45</td>
<td>-45.50</td>
<td>-27.54</td>
</tr>
<tr>
<td>15</td>
<td>-44.81</td>
<td>-37.93</td>
<td>-52.35</td>
<td>-29.34</td>
</tr>
<tr>
<td>20</td>
<td>-50.29</td>
<td>-40.85</td>
<td>-60.75</td>
<td>-30.91</td>
</tr>
<tr>
<td>27</td>
<td>-55.53</td>
<td>-43.89</td>
<td>-56.91</td>
<td>-32.86</td>
</tr>
<tr>
<td>29</td>
<td>-57.21</td>
<td>-44.77</td>
<td>-69.92</td>
<td>-32.86</td>
</tr>
<tr>
<td>31</td>
<td>-58.87</td>
<td>-45.63</td>
<td>-75.63</td>
<td>-33.25</td>
</tr>
<tr>
<td>42</td>
<td>-68.61</td>
<td>-50.16</td>
<td>-92.15</td>
<td>-35.10</td>
</tr>
<tr>
<td>50</td>
<td>-75.50</td>
<td>-53.15</td>
<td>-103.48</td>
<td>-36.33</td>
</tr>
<tr>
<td>70</td>
<td>-91.41</td>
<td>-60.25</td>
<td>-131.81</td>
<td>-38.77</td>
</tr>
<tr>
<td>100</td>
<td>-114.97</td>
<td>-70.41</td>
<td>-174.28</td>
<td>-28.77</td>
</tr>
</tbody>
</table>

### Table CLs.1 Root Accuracy of Clark-Hau algorithm, over Channel 1 (Sampled Impulse Response set to 10)

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root Accuracy [d=10^4]</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-210.13</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-159.55</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>-0.5</td>
<td>-115.20</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table CLs.2 Root Accuracy of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$, and $\psi_3$ over Channel 2 (Sampled Impulse Response set to 10)

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy (d=10^4)</th>
<th>Iterations</th>
<th>Starting point</th>
<th>Root Accuracy (d=10^4)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5+i*0.5</td>
<td>-162.82</td>
<td>9</td>
<td>0.5</td>
<td>-129.57</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>0.5+i*0.5</td>
<td>-156.50</td>
<td>6</td>
<td>0.5</td>
<td>-142.43</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-156.73</td>
<td>6</td>
<td>-0.5</td>
<td>-194.18</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>-82.58</td>
<td>6</td>
<td>-0.5</td>
<td>-171.13</td>
<td>4</td>
</tr>
</tbody>
</table>
### Table CLs.3 Root Accuracy of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$ over Channel 3 (Sampled Impulse Response set to 10)

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-172.69</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-116.52</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-191.74</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>-155.79</td>
<td>8</td>
</tr>
</tbody>
</table>

### Table CLs.4 Root Accuracy of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$ over Channel 4 (Sampled Impulse Response set to 15)

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-121.83</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-112.68</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-132.75</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>-119.69</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>-125.64</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>0.5+i*0.5</td>
<td>-118.62</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>-0.5</td>
<td>-117.65</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>all</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table CLs.5 Root Accuracy of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$ over Channel 5 (Sampled Impulse Response set to 20)

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5+i*0.5</td>
<td>-149.63</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-187.94</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>0.5+i*0.5</td>
<td>-193.46</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>-0.5+i*0.5</td>
<td>-159.62</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-140.34</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>0.5+i*0.5</td>
<td>-146.18</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>0.5+i*0.5</td>
<td>-154.75</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Channel 6

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-157.83</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-147.55</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>0.5+i*0.5</td>
<td>-162.10</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>-0.5+i*0.5</td>
<td>-152.70</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>-0.5</td>
<td>-161.90</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>-149.10</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>-140.30</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>-157.15</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>0.5+i*0.5</td>
<td>-159.32</td>
<td>6</td>
</tr>
</tbody>
</table>

Table CLs.6 Root Accuracy of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$ over Channel 6 (Sampled Impulse Response set to 20)

### Channel 7

<table>
<thead>
<tr>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>-186.32</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>-0.5<em>i</em>0.5</td>
<td>-146.33</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>-148.89</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>-0.5<em>i</em>0.5</td>
<td>-145.84</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>-140.30</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>-0.5</td>
<td>-150.24</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>any</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>-0.5</td>
<td>-157.15</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>-0.5</td>
<td>-159.32</td>
<td>6</td>
</tr>
</tbody>
</table>

Table CLs.7 Root Accuracy of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$ over Channel 7 (Sampled Impulse Response set to 20)
<table>
<thead>
<tr>
<th>Channel 8</th>
<th>Root</th>
<th>Starting Point</th>
<th>Root accuracy</th>
<th>Iterations</th>
<th>Root order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>-130.76</td>
<td>5</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-139.43</td>
<td>6</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.5</td>
<td>-136.84</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>-141.34</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-198.28</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>-137.58</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-0.8</td>
<td>-135.56</td>
<td>4</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>-145.81</td>
<td>9</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.5+i*0.5</td>
<td>-207.55</td>
<td>12</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-0.5</td>
<td>-147.40</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>any</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>-0.5</td>
<td>-144.82</td>
<td>11</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Table CLs.8 Root Accuracy of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$, and $\psi_3$ over Channel 8 (Sampled Impulse Response set to 20)

<table>
<thead>
<tr>
<th>Channels</th>
<th>Operation</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of Iterations</td>
<td>60</td>
<td>59</td>
<td>58</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Addition &amp; Subtraction</td>
<td>4285</td>
<td>4306</td>
<td>4245</td>
<td>6348</td>
</tr>
<tr>
<td></td>
<td>Multiplication</td>
<td>3025</td>
<td>3067</td>
<td>3027</td>
<td>4605</td>
</tr>
<tr>
<td></td>
<td>Division</td>
<td>60</td>
<td>59</td>
<td>58</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Total Operational Count</td>
<td>27600</td>
<td>27888</td>
<td>27520</td>
<td>41344</td>
</tr>
</tbody>
</table>

Table CLs.9 Number of arithmetic operations over Channels 1-4 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>Channels</th>
<th>Operation</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of Iterations</td>
<td>60</td>
<td>59</td>
<td>58</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Addition &amp; Subtraction</td>
<td>4285</td>
<td>4306</td>
<td>4245</td>
<td>6348</td>
</tr>
<tr>
<td></td>
<td>Multiplication</td>
<td>3025</td>
<td>3067</td>
<td>3027</td>
<td>4605</td>
</tr>
<tr>
<td></td>
<td>Division</td>
<td>60</td>
<td>59</td>
<td>58</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Total Operational Count</td>
<td>27600</td>
<td>27888</td>
<td>27520</td>
<td>41344</td>
</tr>
</tbody>
</table>

Table CLs.10 Number of arithmetic operations over Channels 5-8 (Shortened Sampled Impulse Responses)
<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
<th>Channel 5</th>
<th>Channel 6</th>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
<td>-128.35</td>
<td>-146.91</td>
<td>-144.83</td>
<td>-32.77</td>
<td>-15.99</td>
<td>-11.89</td>
<td>-15.33</td>
<td>-22.94</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-127.88</td>
<td>-145.20</td>
<td>-140.68</td>
<td>-33.00</td>
<td>-16.73</td>
<td>-11.95</td>
<td>-15.51</td>
<td>-23.25</td>
</tr>
</tbody>
</table>

Table CLs.11 Performance of Clark-Hau algorithm, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channels 1-8

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Iteration</td>
<td>2 Iteration</td>
<td>1 Iteration</td>
</tr>
<tr>
<td>$\psi_1$</td>
<td>-21.27</td>
<td>-20.10</td>
<td>-14.10</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-9.92</td>
<td>-48.10</td>
<td>-9.82</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-9.10</td>
<td>-47.35</td>
<td>-9.84</td>
</tr>
</tbody>
</table>

Table LEs.1 Accuracy of Levinson, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channels 1, 2 and 3 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Iteration</td>
</tr>
<tr>
<td>$\psi_1$</td>
<td>81.44</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>0.96</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-1.77</td>
</tr>
</tbody>
</table>

Table LEs.2 Accuracy of Levinson, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channels 1, 2 and 3 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>Channels</th>
<th>Channel 5</th>
<th>Channel 6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Iteration</td>
<td>2 Iteration</td>
</tr>
<tr>
<td>$\psi_1$</td>
<td>17.66</td>
<td>-22.46</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-1.13</td>
<td>-56.77</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-1.30</td>
<td>-56.95</td>
</tr>
</tbody>
</table>

Table LEs.3 Accuracy of Levinson, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channels 5 and 6 (Shortened Sampled Impulse Responses)
### Table L:E.4 Accuracy of Levinson, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channels 7 and 8 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>Channel 7</th>
<th>Channel 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>$\psi_1$</td>
</tr>
<tr>
<td>Iteration</td>
<td>Iteration</td>
</tr>
<tr>
<td>-25.90</td>
<td>-32.31</td>
</tr>
<tr>
<td>-25.42</td>
<td>-29.77</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>$\psi_2$</td>
</tr>
<tr>
<td>Iteration</td>
<td>Iteration</td>
</tr>
<tr>
<td>-10.82</td>
<td>-10.77</td>
</tr>
<tr>
<td>-43.29</td>
<td>-58.25</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>$\psi_3$</td>
</tr>
<tr>
<td>Iteration</td>
<td>Iteration</td>
</tr>
<tr>
<td>-10.98</td>
<td>-9.10</td>
</tr>
<tr>
<td>-43.30</td>
<td>-58.41</td>
</tr>
</tbody>
</table>

### Table L:E.5 Accuracy of Levinson, given by $\psi_1$, $\psi_2$ and $\psi_3$, over Channels 1, 2 and 3 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1228</td>
<td>1228</td>
<td>1228</td>
<td>5636</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1502</td>
<td>1502</td>
<td>1502</td>
<td>6604</td>
</tr>
<tr>
<td>Division</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>240</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>11948</td>
<td>11948</td>
<td>11948</td>
<td>52336</td>
</tr>
</tbody>
</table>

### Table L:E.6 Number of arithmetic operations, required by Levinson algorithm, over Channels 5, 6, 7 and 8 (Shortened Sampled Impulse Responses)

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>5058</td>
<td>5058</td>
<td>5058</td>
<td>5058</td>
</tr>
<tr>
<td>Multiplication</td>
<td>5802</td>
<td>5802</td>
<td>5802</td>
<td>5802</td>
</tr>
<tr>
<td>Division</td>
<td>160</td>
<td>160</td>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>45888</td>
<td>45888</td>
<td>45888</td>
<td>45888</td>
</tr>
</tbody>
</table>
### Table T.4 Sampled Impulse Response and associated Roots for Channel 2 (HF5)

<table>
<thead>
<tr>
<th>Real</th>
<th>Imaginary</th>
<th>Modulus</th>
<th>Real</th>
<th>Imaginary</th>
<th>Modulus</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1445</td>
<td>0.1096</td>
<td>0.1814</td>
<td>0.8513</td>
<td>0.7364</td>
<td>1.1256</td>
</tr>
<tr>
<td>0.1535</td>
<td>0.1500</td>
<td>0.2147</td>
<td>0.9520</td>
<td>0.2455</td>
<td>0.9831</td>
</tr>
<tr>
<td>0.1690</td>
<td>0.1812</td>
<td>0.2477</td>
<td>0.4545</td>
<td>0.8866</td>
<td>0.9963</td>
</tr>
<tr>
<td>0.1908</td>
<td>0.1990</td>
<td>0.2756</td>
<td>0.1813</td>
<td>0.9536</td>
<td>0.9707</td>
</tr>
<tr>
<td>0.2283</td>
<td>0.2049</td>
<td>0.3068</td>
<td>-0.1056</td>
<td>0.9582</td>
<td>0.9640</td>
</tr>
<tr>
<td>0.2566</td>
<td>0.1922</td>
<td>0.3206</td>
<td>-0.3854</td>
<td>0.8867</td>
<td>0.9669</td>
</tr>
<tr>
<td>0.2575</td>
<td>0.1589</td>
<td>0.3026</td>
<td>-0.6687</td>
<td>0.7440</td>
<td>1.0003</td>
</tr>
<tr>
<td>0.2220</td>
<td>0.1079</td>
<td>0.2468</td>
<td>0.9077</td>
<td>-0.2567</td>
<td>0.9433</td>
</tr>
<tr>
<td>0.1239</td>
<td>0.0353</td>
<td>0.1288</td>
<td>0.8198</td>
<td>-0.8270</td>
<td>1.1645</td>
</tr>
<tr>
<td>0.0120</td>
<td>-0.0338</td>
<td>0.0359</td>
<td>-0.8393</td>
<td>0.4677</td>
<td>0.9608</td>
</tr>
<tr>
<td>-0.0727</td>
<td>-0.0796</td>
<td>0.1078</td>
<td>-0.9430</td>
<td>0.1994</td>
<td>0.9638</td>
</tr>
<tr>
<td>-0.1014</td>
<td>-0.0893</td>
<td>0.1351</td>
<td>-0.9634</td>
<td>-0.0985</td>
<td>0.9685</td>
</tr>
<tr>
<td>-0.0195</td>
<td>-0.0406</td>
<td>0.0451</td>
<td>-0.8942</td>
<td>-0.3935</td>
<td>0.9769</td>
</tr>
<tr>
<td>0.1134</td>
<td>0.0359</td>
<td>0.1189</td>
<td>-0.7182</td>
<td>-0.6877</td>
<td>0.9944</td>
</tr>
<tr>
<td>0.2436</td>
<td>0.1125</td>
<td>0.2683</td>
<td>-0.4702</td>
<td>-0.8275</td>
<td>0.9518</td>
</tr>
<tr>
<td>0.3270</td>
<td>0.1662</td>
<td>0.3668</td>
<td>0.4214</td>
<td>-0.9284</td>
<td>1.0195</td>
</tr>
<tr>
<td>0.2839</td>
<td>0.1605</td>
<td>0.3261</td>
<td>-0.2000</td>
<td>-0.9526</td>
<td>0.9733</td>
</tr>
<tr>
<td>0.1733</td>
<td>0.1218</td>
<td>0.2118</td>
<td>0.1092</td>
<td>-0.9834</td>
<td>0.9894</td>
</tr>
<tr>
<td>0.0477</td>
<td>0.0758</td>
<td>0.0896</td>
<td>0.3168</td>
<td>-0.2704</td>
<td>0.4165</td>
</tr>
<tr>
<td>-0.0488</td>
<td>0.0465</td>
<td>0.0674</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table T.5 Accuracy of Root Finding (any), Cepstrum and N I T methods given by parameters \( \psi_2 \) and \( \psi_3 \) for Channel 2 (HF5)

<table>
<thead>
<tr>
<th>Root Finding Methods</th>
<th>Cepstrum</th>
<th>N I T</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_2 )</td>
<td>-30.55</td>
<td>-26.61</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>-30.55</td>
<td>-29.73</td>
</tr>
</tbody>
</table>

**ADJUSTMENT ALGORITHMS / COMPUTER SIMULATION-RESULTS**
### CHANNEL 2 (HF5) [2nd Transmission Instant]

<table>
<thead>
<tr>
<th>Raw'Sampled Impulse Response</th>
<th>Roots of Raw' Sampled Impulse Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>0.0769</td>
<td>0.1205</td>
</tr>
<tr>
<td>0.1048</td>
<td>0.0374</td>
</tr>
<tr>
<td>0.1279</td>
<td>-0.0555</td>
</tr>
<tr>
<td>0.1456</td>
<td>-0.1297</td>
</tr>
<tr>
<td>0.1583</td>
<td>-0.1570</td>
</tr>
<tr>
<td>0.1691</td>
<td>-0.1120</td>
</tr>
<tr>
<td>0.1611</td>
<td>-0.0230</td>
</tr>
<tr>
<td>0.1280</td>
<td>0.0874</td>
</tr>
<tr>
<td>0.0660</td>
<td>0.1896</td>
</tr>
<tr>
<td>-0.0274</td>
<td>0.2522</td>
</tr>
<tr>
<td>-0.1216</td>
<td>0.2666</td>
</tr>
<tr>
<td>-0.1951</td>
<td>0.2275</td>
</tr>
<tr>
<td>-0.2270</td>
<td>0.1398</td>
</tr>
<tr>
<td>-0.1993</td>
<td>0.0146</td>
</tr>
<tr>
<td>-0.1285</td>
<td>-0.1111</td>
</tr>
<tr>
<td>-0.0290</td>
<td>-0.2085</td>
</tr>
<tr>
<td>0.0783</td>
<td>-0.2524</td>
</tr>
<tr>
<td>0.1689</td>
<td>-0.2230</td>
</tr>
<tr>
<td>0.2358</td>
<td>-0.1518</td>
</tr>
<tr>
<td>0.2703</td>
<td>-0.0644</td>
</tr>
</tbody>
</table>

Table T.6 Sampled Impulse Response and associated Roots for Channel 2 (HF5)

<table>
<thead>
<tr>
<th>Root Finding Methods</th>
<th>Cepstrum</th>
<th>N I T</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Itrs = 10</td>
<td>Itrs = 7</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-15.24</td>
<td>-29.49</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-15.24</td>
<td>-33.59</td>
</tr>
</tbody>
</table>

Table T.7 Accuracy of Root Finding (any), Cepstrum and N I T methods given by parameters $\psi_2$ and $\psi_3$ for Channel 2 (HF5)
### CHANNEL 3 (HF4) (1st Transmission Instant)

<table>
<thead>
<tr>
<th>'Raw' Sampled Impulse Response</th>
<th>Roots of 'Raw' Sampled Impulse Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>0.2629</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.2327</td>
<td>-0.1637</td>
</tr>
<tr>
<td>0.1617</td>
<td>-0.2605</td>
</tr>
<tr>
<td>0.0428</td>
<td>-0.2029</td>
</tr>
<tr>
<td>-0.0836</td>
<td>-0.0551</td>
</tr>
<tr>
<td>-0.1829</td>
<td>0.1167</td>
</tr>
<tr>
<td>-0.2316</td>
<td>0.2507</td>
</tr>
<tr>
<td>-0.1900</td>
<td>0.2477</td>
</tr>
<tr>
<td>-0.1153</td>
<td>0.1684</td>
</tr>
<tr>
<td>-0.0431</td>
<td>0.0459</td>
</tr>
<tr>
<td>-0.0128</td>
<td>-0.0696</td>
</tr>
<tr>
<td>-0.0650</td>
<td>-0.1187</td>
</tr>
<tr>
<td>-0.1459</td>
<td>-0.1172</td>
</tr>
<tr>
<td>-0.2235</td>
<td>-0.0693</td>
</tr>
<tr>
<td>-0.2534</td>
<td>0.0064</td>
</tr>
<tr>
<td>-0.2117</td>
<td>0.0865</td>
</tr>
<tr>
<td>-0.1424</td>
<td>0.1562</td>
</tr>
<tr>
<td>-0.0731</td>
<td>0.1996</td>
</tr>
<tr>
<td>-0.0471</td>
<td>0.2025</td>
</tr>
<tr>
<td>-0.0728</td>
<td>0.1683</td>
</tr>
</tbody>
</table>

Table T.8 Sampled Impulse Response and associated Roots for Channel 3 (HF4)

### Table T.9 Accuracy of Cepstrum and N I T methods given by parameters \( \psi_2 \) and \( \psi_3 \) for Channel 3 (HF4)

<table>
<thead>
<tr>
<th></th>
<th>L = 64</th>
<th>L = 128</th>
<th>Itr = 10</th>
<th>Itr = 5</th>
<th>Itr = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_2 )</td>
<td>-16.28</td>
<td>-27.30</td>
<td>-28.70</td>
<td>-20.26</td>
<td>-16.64</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>-12.24</td>
<td>-25.98</td>
<td>-28.70</td>
<td>-20.26</td>
<td>-20.26</td>
</tr>
</tbody>
</table>

Table T.9 Accuracy of Cepstrum and N I T methods given by parameters \( \psi_2 \) and \( \psi_3 \) for Channel 3 (HF4)

### Table T.10 Roots outside the unit circle for Cepstrum method over Channel 3 (HF4)

<table>
<thead>
<tr>
<th></th>
<th>L = 64</th>
<th>L = 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_2 )</td>
<td>1.036</td>
<td>1.040</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>1.083</td>
<td>1.059</td>
</tr>
</tbody>
</table>

Table T.10 Roots outside the unit circle for Cepstrum method over Channel 3 (HF4)
<table>
<thead>
<tr>
<th>Real</th>
<th>Imaginary</th>
<th>Modulus</th>
<th>Real</th>
<th>Imaginary</th>
<th>Modulus</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2469</td>
<td>0.1196</td>
<td>0.2743</td>
<td>1.0646</td>
<td>0.3779</td>
<td>1.1297</td>
</tr>
<tr>
<td>0.2388</td>
<td>0.0602</td>
<td>0.2462</td>
<td>0.8411</td>
<td>0.5354</td>
<td>0.9970</td>
</tr>
<tr>
<td>0.1837</td>
<td>-0.0061</td>
<td>0.1838</td>
<td>0.5354</td>
<td>0.9171</td>
<td>1.0619</td>
</tr>
<tr>
<td>0.0834</td>
<td>-0.0707</td>
<td>0.1093</td>
<td>0.2436</td>
<td>1.0180</td>
<td>1.0467</td>
</tr>
<tr>
<td>-0.0076</td>
<td>-0.1115</td>
<td>0.1117</td>
<td>-0.0816</td>
<td>1.0365</td>
<td>1.0397</td>
</tr>
<tr>
<td>-0.0566</td>
<td>-0.1183</td>
<td>0.1312</td>
<td>-0.3977</td>
<td>0.9575</td>
<td>1.0368</td>
</tr>
<tr>
<td>-0.0193</td>
<td>-0.0805</td>
<td>0.0828</td>
<td>-0.6763</td>
<td>0.7789</td>
<td>1.0315</td>
</tr>
<tr>
<td>0.0918</td>
<td>-0.0097</td>
<td>0.0923</td>
<td>-0.8666</td>
<td>0.5395</td>
<td>1.0208</td>
</tr>
<tr>
<td>0.2106</td>
<td>0.0633</td>
<td>0.2199</td>
<td>-0.9946</td>
<td>0.2399</td>
<td>1.0231</td>
</tr>
<tr>
<td>0.2919</td>
<td>0.1167</td>
<td>0.3144</td>
<td>-1.0179</td>
<td>-0.0800</td>
<td>1.0211</td>
</tr>
<tr>
<td>0.2644</td>
<td>0.1217</td>
<td>0.2911</td>
<td>-0.9405</td>
<td>-0.3888</td>
<td>1.0177</td>
</tr>
<tr>
<td>0.1587</td>
<td>0.0920</td>
<td>0.1835</td>
<td>-0.7820</td>
<td>-0.6711</td>
<td>1.0305</td>
</tr>
<tr>
<td>0.0367</td>
<td>0.0558</td>
<td>0.0668</td>
<td>1.0048</td>
<td>-0.1471</td>
<td>1.0155</td>
</tr>
<tr>
<td>-0.0561</td>
<td>0.0367</td>
<td>0.0670</td>
<td>-0.5242</td>
<td>-0.8698</td>
<td>1.0156</td>
</tr>
<tr>
<td>-0.0379</td>
<td>0.0690</td>
<td>0.0788</td>
<td>0.9123</td>
<td>-0.4435</td>
<td>1.0144</td>
</tr>
<tr>
<td>0.0321</td>
<td>0.1384</td>
<td>0.1420</td>
<td>-0.2295</td>
<td>-0.9843</td>
<td>1.0107</td>
</tr>
<tr>
<td>0.0996</td>
<td>0.2268</td>
<td>0.2477</td>
<td>0.0876</td>
<td>-1.0015</td>
<td>1.0054</td>
</tr>
<tr>
<td>0.1194</td>
<td>0.3128</td>
<td>0.3348</td>
<td>0.3994</td>
<td>-0.9143</td>
<td>0.9977</td>
</tr>
<tr>
<td>0.0118</td>
<td>0.3642</td>
<td>0.3644</td>
<td>0.5430</td>
<td>-0.7181</td>
<td>0.9003</td>
</tr>
<tr>
<td>-0.1495</td>
<td>0.3794</td>
<td>0.4078</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table T.11 Sampled Impulse Response and associated Roots for Channel 3 (HF4)

<table>
<thead>
<tr>
<th>Root Finding Methods</th>
<th>Cepstrum</th>
<th>N I T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
<td>-12.80</td>
<td>-23.77</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-12.80</td>
<td>-23.77</td>
</tr>
</tbody>
</table>

Table T.12 Accuracy of Root Finding (any), Cepstrum and N I T methods given by parameters $\psi_2$ and $\psi_3$ for Channel 3 (HF4)
### CHANNEL 3 (HF4) [3rd Transmission Instant]

<table>
<thead>
<tr>
<th>'Raw' Sampled Impulse Response</th>
<th>Roots of 'Raw' Sampled Impulse Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Imaginary</td>
</tr>
<tr>
<td>0.0201</td>
<td>0.2539</td>
</tr>
<tr>
<td>-0.0017</td>
<td>0.3334</td>
</tr>
<tr>
<td>-0.0201</td>
<td>0.3568</td>
</tr>
<tr>
<td>-0.0250</td>
<td>0.3282</td>
</tr>
<tr>
<td>-0.0119</td>
<td>0.2596</td>
</tr>
<tr>
<td>0.0327</td>
<td>0.1756</td>
</tr>
<tr>
<td>0.0736</td>
<td>0.0870</td>
</tr>
<tr>
<td>0.0910</td>
<td>0.0909</td>
</tr>
<tr>
<td>0.0656</td>
<td>-0.0457</td>
</tr>
<tr>
<td>-0.0217</td>
<td>-0.0688</td>
</tr>
<tr>
<td>-0.1172</td>
<td>-0.0714</td>
</tr>
<tr>
<td>-0.1895</td>
<td>-0.0619</td>
</tr>
<tr>
<td>-0.1995</td>
<td>-0.0547</td>
</tr>
<tr>
<td>-0.1331</td>
<td>-0.0564</td>
</tr>
<tr>
<td>-0.0513</td>
<td>-0.0433</td>
</tr>
<tr>
<td>0.0088</td>
<td>-0.0061</td>
</tr>
<tr>
<td>-0.0069</td>
<td>0.0742</td>
</tr>
<tr>
<td>-0.0952</td>
<td>0.1817</td>
</tr>
<tr>
<td>-0.1969</td>
<td>0.2695</td>
</tr>
<tr>
<td>-0.2708</td>
<td>0.3103</td>
</tr>
</tbody>
</table>

Table T.13 Sampled Impulse Response and associated Roots for Channel 3 (HF4)

### Root Finding Methods

<table>
<thead>
<tr>
<th>Root Finding Methods</th>
<th>Cepstrum</th>
<th>N I T</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_2 )</td>
<td>-9.21</td>
<td>-29.19</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>-9.21</td>
<td>-29.19</td>
</tr>
</tbody>
</table>

Table T.14 Accuracy of Root Finding (any), Cepstrum and N I T methods given by parameters \( \psi_2 \) and \( \psi_3 \) for Channel 3 (HF4)
Graph G.NT1 Performance of Cepstrum and NIT algorithms over HF radio links (HF2)

Graph G.NT2 Performance of Cepstrum and NIT algorithms over HF radio links (HF5)
Graph G.NT3 Raw Impulse Response, Minimum Phase and Pseudo Minimum Phase Impulse Responses

Graph G.NT4 Performance of Cepstrum and NIT algorithms over HF radio links (HF5)
Graph G.NT5 Performance of Root Finding (any), Cepstrum and NIT algorithms over HF radio links (HF4)

Graph G.NT6 Performance of Root Finding Methods (any), over an HF radio link (HF4)
Graph G.NT7 Performance of Cepstrum and N I T methods over an HF radio link (HF4)

Graph G.NT8 Performance of Root Finding Methods (any), over an HF radio link (HF4)
Graph G.NT9 Performance of Cepstrum and N I T methods over an HF radio link (HF4)
Graph G.1 Performance of Selected Adjustment Techniques over Channel 1
Graph G.2 Performance of Selected Adjustment Techniques over Channel 2
NUMBER OF ARITHMETIC OPERATIONS

Graph G.3 Performance of Selected Adjustment Techniques over Channel 3
NUMBER OF ARITHMETIC OPERATIONS

ACCURACY (dB)

Graph G.4 Performance of Selected Adjustment Techniques over Channel 4
Graph G.5 Performance of Selected Adjustment Techniques over Channel 5
Graph G.6 Performance of Selected Adjustment Techniques over Channel 6
Graph G.7 Performance of Selected Adjustment Techniques over Channel 7
NUMBER OF ARITHMETIC OPERATIONS

CHANNEL 8

![Graph showing the number of arithmetic operations for different adjustment algorithms over Channel 8.](image)

ACCURACY (dB)

Graph G.8 Performance of Selected Adjustment Techniques over Channel 8
Figure 4.4.1 Attenuation and Group Delay Characteristics of Telephone Circuit 1

a. Attenuation Characteristics

b. Group Delay Characteristics
Figure 4.4.2 Attenuation and Group Delay Characteristics of Telephone Circuit 2

a. Attenuation Characteristics

b. Group Delay Characteristics
Figure 4.4.3 Attenuation and Group Delay Characteristics of Telephone Circuit 3

a. Attenuation Characteristics

b. Group Delay Characteristics
Figure 4.4.4 Attenuation and Group Delay Characteristics of Telephone Circuit 4

a. Attenuation Characteristics

b. Group Delay Characteristics
References


Mathematical Modelling and Applications, Arab School of Science and Technology,
Damascus, Syria (1986).


33. Tsabieris, N.I., 'Computer Modelling of High Frequency Channels', IEE
International Colloquium on Frequency Selection and Management techniques for
CHAPTER FIVE

FILTER IMPLEMENTATION

5.1 Introduction

This Chapter discusses the hardware implementation of the algorithms of the previous Chapter that performed best (i.e. adjusted the digital receiver in a fast and accurate fashion). Since hardware implementation involves usually fixed-point arithmetic Section 5.2 serves as a brief introduction to this subject. The next part of the Chapter (5.3) discusses issues relating to the hardware implementation of the selected for implementation algorithms. Extensive computer simulation is the theme of part 5.4. Different tests show in a most definite way the best algorithm and its performance under various operating conditions. Finally Section 5.5 summarises the findings of the previous part and stresses the most important results generated in this Chapter.

5.2 Fixed point -floating point arithmetic

Digital signal processing algorithms such as linear filtering and discrete Fourier transforms are realised either with special-purpose digital hardware or as programs for general-purpose digital computer. In both cases sequence values and coefficients are stored in a binary format with finite-length registers. (When implementing digital signal processing systems, coefficients and signals should be represented in some digital number system that must be of finite precision. Most digital computers or special-purpose hardware use a binary number system).

Numbers can, in general, be stored or represented as floating-point, fixed-point or, scaled fixed-point numbers using one word (simple precision) in a computer as shown in Figure 5.2.1.
An arbitrary real number would require an infinite number of bits for its exact binary representation. Since the word-length of the computer is fixed, any number that exceeds this length must be quantified before it can be stored in the memory. The operation of quantizing a number to B bits can be implemented by rounding or truncation, but in either case quantization is a non-linear memoryless operation.

In addition (or subtraction) of two fixed-point numbers, each of B bits long, the result will be a B-bit number. If the result of the addition exceeds the largest number that can be represented with B bits, an overflow occurs. The multiplication of two fixed-point numbers each of B bits in length results in a product of 2B bits long. In fixed-point arithmetic, the product is either truncated or rounded back to B bits. This results in a truncation or round-off error in the B least significant bits.

Thus, very often, in a system, small errors in the input values will give results which are off by a significantly large amount. The value of the roundoff or truncation error, depends on the number of bits in the original number to the number of bits after quantization. The characteristics of the errors also depend on the particular form of number representation. Fixed-point arithmetic supports three number representations (i.e. Sign-Magnitude, Ones Complement and Twos Complement). They are all referred to as fixed point because the radix point (binary point) is fixed and assumed to be to the right of the rightmost digit (Figure 5.2.1). Note that, since rounding involves only the magnitude of the number, the round off error is independent of the type of fixed-point representation. Since the TMS320C25 supports two's complement arithmetic a brief reference to the two's complement number system is presented in Appendix A5.

The immediate question is whether a 16-bit digital signal processor working in integer arithmetic will be able to give the necessary accuracy which is required by the selected algorithm(s) for implementation. The immediate question is whether a 16-bit digital signal processor working in integer arithmetic will be able to give the necessary accuracy which is required by the selected algorithms for implementation on the TMS320. To find out the answer to that question, it is, first of all, necessary to know
the largest number that can occur in the algorithm, including those resulted from all arithmetic operations within an arithmetic expression. A word-length of 16 bits gives a total of 65536 levels and, therefore, to implement the algorithm using 16-bit fixed-point arithmetic means that the numbers can only take 65536 levels. Given the largest number which has to be handled by the processor, it places a limit on the smallest number and, therefore, the accuracy and resolution of the numbers.

For example, if the largest number is 1 then the smallest value which can be represented is $1.5259 \times 10^{-4}$, assuming that all numbers are positive, and the range is $1.5259 \times 10^{-4}$ to 1. Numbers smaller than $1.5259 \times 10^{-4}$ might be too insignificant for additions and subtractions and, thus, can be ignored without affecting the results. However, these same numbers, when appearing in multiplications, divisions or as threshold values in conditional statements, can be absolutely vital and must not be allowed to set to zero. Thus, prior knowledge of the range of numbers that occur for all operational conditions is necessary for the algorithm implementation.

Once the range of numbers is determined, it is a matter of simply reducing this range by as much as possible by modifying the algorithm. Having done that, the algorithm can be tested using computer simulation and, depending on the outcome, the algorithm might have to be further modified until successful operation can be achieved which might not be possible, of course.

5.3 Algorithm Implementation

5.3.1 Implementation considerations for the root finding algorithms using 16-bit fixed point arithmetic.

Root finding techniques such as the Clark-Hau or the Laguerre methods, presented in the previous chapter cannot be implemented using fixed-point or integer arithmetic without further modifications because of the round-off errors or quantization noise present in the calculations, at various stages of the algorithms. One standard method of simulating the effect of this round off error is by assuming that the various
calculations are carried out with infinite precision, then simply adding a round-off noise in the results [1]. The round-off noises are the sample values of a uniform random variable with zero mean and a variance $2^{-2b}/12$, where $b+1$ is the word-length (number of bits) of each integer number. The necessary changes for the proper operation of the Clark-Hau algorithm are described elsewhere [2] and have been incorporated in the computer simulation programs in this thesis. Changes in Laguerre method included scaling at various stages and a change in the fractional resolution of the method to bring the numbers occurring when running the algorithm with full precision, within the range allowed by the 16-bit fixed point arithmetic.

5.3.2 CEPSTRUM Algorithm

The algorithm as used in this thesis can be divided into two parts. The first part of the algorithm makes possible the conversion of the original signal to its minimum phase version (Chapter three, Equations 3.12.1-3.12.8). The second part provides the coefficients necessary for the adjustment of the prefilter and performs the actual adjustment in the form of a convolution (Chapter three, Equations 3.12.9-3.12.13). Since the discrete Fourier Transform is a basic building block in this method it is important to understand the effects of finite register length in the calculations involved. However, a precise analysis of the effects is difficult and often a simplified analysis is sufficient for the purpose of choosing the required length register for computing the discrete Fourier transform. In particular an analysis of arithmetic roundoff is carried out, by means of an additive noise source at each point in the computation algorithm where roundoff occurs.

The discrete Fourier transform is defined by the equation

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0,1,\ldots,N-1$$

$$W_N^{kn} = e^{-j2\pi kn/N} = \cos(2\pi kn/N) - j\sin(2\pi kn/N) \quad 5.3.1a$$
The inverse transform is given by

\[ x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-kn} \]  \hspace{1cm} 5.3.1b

\[ n = 0, 1, \ldots, N \]

Equation 5.3.1 is generally evaluated by one of the algorithms collectively known as fast Fourier transform.

A particular class of fast Fourier Transforms is considered here. In particular the most commonly used FFT algorithm is the radix 2 decimation-in-time frequency form.

In general \( x(n) \) is a sequence of complex numbers. The inner loop of the power of two FFT algorithm operates on two complex numbers from the sequence. It takes these two numbers and produces two new complex numbers which replace the original ones in the sequence. Let \( X_m(p) \) and \( X_m(q) \) be the original complex numbers. Then, the new pair \( X_{m+1}(p), X_{m+1}(q) \) are given by

\[ X_{m+1}(p) = X_m(p) + X_m(q)W_N \]
\[ X_{m+1}(q) = X_m(p) - X_m(q)W_N \]  \hspace{1cm} 5.3.2

where \( m \) and \( m+1 \) refer to the \( m \)th array and \( p \) and \( q \) denote the location of the numbers in the array.

This basic numerical computation, called butterfly, is shown diagramatically in Figure 5.3.1. If Equations 5.3.1 are written out in terms of their real and imaginary parts, then

\[ \text{Re}\{X_{m+1}(p)\} = \text{Re}\{X_m(p)\} + \text{Re}\{X_m(p)\text{Re}\{W_N\}} \]
\[ -\text{Im}\{X_m(p)\}\text{Im}\{W_N\} \]

\[ \text{Im}\{X_{m+1}(q)\} = \text{Im}\{X_m(p)\} - \text{Re}\{X_m(p)\text{Im}\{W_N\}} \]
\[ -\text{Im}\{X_m(q)\}\text{Re}\{W_N\} \]  \hspace{1cm} 5.3.3
At each stage the algorithm goes through the entire sequence of N numbers in this fashion, two at a time. The discrete Fourier transform is then computed in \( v = \log_2 N \) stages. At each stage, \( N/2 \) separate butterfly computations are carried out to produce the next array. Moving from stage to stage through the calculation, the magnitudes of the numbers in the sequence generally increase which means that it can be kept properly scaled by right shifts. A number of strategies are available to keep the array properly scaled.

a) Shifting right one bit at every iteration: If the initial sequence, \( X_0(p) \), is scaled so that \( |X_0(p)| < 1/2 \) for all \( k \) and if there is a right shift of one bit after every iteration (excluding the last one) then there will be no overflows.

b) Controlling the sequence so that \( |X_m(p)| < 1/2 \): Again assume that the initial sequence is scaled so that \( |X_0(p)| < 1/2 \) for all \( k \). Then \( |X_m(p)| \) is checked at each iteration and if it is greater than one half for any \( k \) a one bit right shift is performed before each calculation throughout the next iteration.

c) Testing for an overflow: In this case the initial sequence is scaled so that \( \text{Re}(X_m(p)) < 1 \) and \( \text{Im}(X_m(p)) < 1 \). Whenever an overflow occurs in an iteration the entire sequence is shifted right by one bit and the iteration is continued at the point at which the overflow occurred. In this case there could be two overflows during an operation.

The first alternative is the simplest, but the least accurate. Since it is not generally necessary to rescale the sequence at each iteration, there is an unnecessary loss in accuracy. The second alternative is also not as accurate as possible because one less than the total number of bits available is being used for the representation of the sequence. This alternative also requires the computation of the modulus of every member of the sequence at each iteration. The third alternative is the most accurate. It has the disadvantage that one must process through the sequence an additional time whenever there is an overflow. The indexing for this processing is, however straightforward. It would not be the complex indexing required for the algorithm. The roundoff noise is modelled by associating an additive noise generator with each fixed-point multiplication.
The butterfly of Figure 5.3.1 is now replaced by that of Figure 5.3.2 for analysing the roundoff noise effects. The quantity \( e(m,q) \) in Figure 5.3.2 represents a complex error introduced in computing the \((m+1)\)st array from the \(m\)th array and specifically in multiplication of the \(q\)th element of the \(m\)th array by a complex coefficient. The roundoff noise due to each real multiplication is uniformly distributed in amplitude between \( \pm 1/2 \cdot 2^{-b} \), where \( b+1 \) is the wordlength in bits, and thus has a variance of \( \sigma_e^2 = 1/12 \cdot 2^{-2b} \). Since each of the quantization errors has a variance \( \sigma_x^2 \) and assuming a complex input (i.e. four multiplications) sequence \( x(n) \), the variance from the \(4N\) multiplication is

\[
\sigma_q^2 = 4N\sigma_e^2 = \frac{N}{3} \cdot 2^{-2b}
\]

where \( N \) is the length of the FFT. Hence the variance of the quantization error is proportional to the size of the FFT.

If the input is \( x(n) \) is initially such that \( |x(n)| < 1 \) for all \( n \), then each point in the sequence can be divided by \( N \) to ensure that

\[
\sum_{n=0}^{N-1} |x(n)| < 1
\]

is satisfied.

Given the assumptions of independent roundoff errors and white input signal the variance at the output of the output FFT coefficients is

\[
\sigma_x^2 = N\sigma_x^2 = \frac{1}{3N}
\]

where \( \sigma_x^2 \) is the variance of the input signal is equal to

\[
\sigma_x^2 = \frac{1}{3N^2}
\]

Thus the signal to noise power ratio is
\[ \frac{\sigma_q^2}{\sigma_x^2} = \frac{1}{N^2} \cdot 2^{2b} \]

The scaling implied by the above Equation is extremely severe. However, using alternative scaling i.e. by distributing the total scaling of 1/N into each of the stages of the FFT algorithm by applying a scaling factor of 1/2 each time, the total variance of the quantization errors at the output of the FFT algorithm is

\[ \sigma_q^2 \approx \frac{2}{3} \cdot 2^{2b} \]

For the same input signal with variance \( \sigma_x^2 = \frac{1}{3N} \) the SNR is

\[ \frac{\sigma_x^2}{\sigma_q^2} = \frac{1}{2N} \cdot 2^{2b} \]

5.3.3 IMPLEMENTATION ON A TMS320C25

The Cepstrum algorithm can be viewed as having two parts. The first part, performs the conversion of a signal to its equivalent minimum phase version, while the second part provides the coefficients for the setting-up of the prefilter and performs the actual adjustment of the digital receiver. The first part of the algorithm consists, for the purpose of implementation of basic blocks, such as: a. FFT transform (and IFFT), b. the logarithm function and c. the exponential function. The second part of the algorithm involves the process of convolution (and deconvolution) and can be viewed and implemented as an Finite Impulse Response (FIR) filter.

5.3.3.1 First part of the algorithm FFT BLOCK
i. FFT  In recent years Efficient algorithms have been developed to increase the execution speed of FFTs while keeping requirements for memory size low. Single-chip programmable DSP devices, currently available or under development, can realise FFTs with speeds that allow the implementation of very complex systems in real-time.
Since 'the number of variations of the FFT algorithm appears to be directly proportional to the number of people using it' [3] it is apparent that there exist a great number of implementations in software[4-9] or hardware. Most applications for special-purpose FFT processors result from signal processing problems which have an inherent real-time constraint, such as digital filtering, or off-line processing where the volume of data makes processing impractical unless a dedicated machine is used. In these examples, a processing rate slower than real-time would overload the system with input data or lead to worthless results. A special purpose processor is a programmable processor which is a cost effective solution to high speed realisation of a class of computationally intensive algorithms such as the FFT algorithm.

Some hardware realisations of the FFT transform through special-purpose hardware are briefly described.

a. The Sequential Processor: This involves implementing the basic operation depicted in Figure 5.3.1[10] This basic operation (i.e. one complex multiplication followed by an addition and a subtraction) can be applied sequentially to the 12 sets of data shown in Figure 5.3.3. The same memory can be used to store the input data, the intermediate results, and the resulting Fourier coefficients. Since only one basic operation is involved and the accessing pattern is very regular, the amount of hardware involved can be relatively small. A simplified block diagram of the resulting processor is shown in Figure 5.3.4. The sequential processor will be characterised as having:

1. one arithmetic unit,
2. \((N/2)log_2N\) operations performed sequentially,
3. an execution time of \(B(N/2)log_2N\) us, where \(B\) is the time requires for performing one basic operation and \(u\) is a time unit. The reordering of the Fourier coefficients can be done either in place or during I/O.

b. The cascade processor: Improved processor performance can be achieved by introducing parallelism into the flow diagram shown in Figure 5.3.4 [11]. By using a separate arithmetic unit for each iteration, the throughput can be increased by a factor of \(log_2N\). In terms of Figure 5.3.3. this means that the first arithmetic unit performs the operations labelled 1 through 4, the second performs operations 5 through 8, and...
the third performs operations 9 through 12. A simplified block diagram of the resulting processor is shown Figure 5.3.5. The cascade processor can be characterised as having:
1. m arithmetic units,
2. m iterations performed in parallel,
3. \(\frac{N}{2}\) operations performed sequentially
4. an execution time of \((B\cdot N)\) us per record, and
5. buffering incorporated within the processor in the form of time delays.

b. The Parallel Iterative Processor: A third alternative for improving performance involves parallelism within each iteration. By using four arithmetic units, the operations labelled 1 through 4 can be performed in parallel before performing operations 5 through 8 in parallel, etc. The processor performs the iterations sequentially, but performs all of the operations within each iteration in parallel. A simplified diagram of the resulting processor is shown in Figure 5.3.6. This processor is characterised as having
1. \(\frac{N}{2}\) arithmetic units,
2. \(\frac{N}{2}\) operations performed in parallel
3. m iterations performed sequentially
4. an execution time of \(B(\log_2 N)\)us.

c. The Array Analyser: Now, all 12 of the operations of Figure 5.3.3 are performed in parallel [12]. By pipelining three different sets of data through this processor simultaneously, the effective execution time is simply the time required for performing on basic operation. A simplified block diagram is shown in Figure 5.3.7 for \(N=8\). The logic analyser can be characterised as having
1. \((\frac{N}{2})(\log_2 N)\) arithmetic units,
2. \((\frac{N}{2})(\log_2 N)\) operations performed in parallel, and
3. an execution time of \(B\) us.

It has become apparent by now, that special-purpose processors are designed to efficiently carry out a class of operations. One of the most popular special-purpose family of processors is the TMS320 series. In particular the TMS30C25 (member of
the second generation of the TMS family) was chosen for the implementation of the CEPSTRUM algorithm. The TMS320C25, manufactured by Texas Instruments, is a very powerful digital signal processing (DSP) chip, capable of performing a 16 by 16-bit multiplication on 100 ns. Some other key features are 4K words of on-chip masked Read Only Memory (ROM), 544 words of on-chip Random Access Memory (RAM) and 128K words of total program data memory space. A minimal processing configuration assuming the TMS320C25 as a stand-alone system is shown in Figure 5.3.8. The software required to implement the various algorithmic steps must be highly optimised for speed in order to make full use of the processing power. The design of the TMS320C25 has several interesting features that are responsible for its high computation rate. The main features are the modified Harvard architecture, special DSP instructions, extensive pipelining, and the implementation of functions within hardware instead of implementing functions through software or microcode, for example the inclusion of hardware dedicated multiplier. A modification of the Harvard architecture is adopted to enhance speed and flexibility. In a Harvard architecture, the program and data memory spaces are isolated to allow a full overlap of instruction fetch and execution. The modification of the Harvard architecture in the TMS320C25 also permits transfers between program and data spaces. This leads to an increase in the flexibility of the device and in the processing rate by maintaining two separate bus structures for full overlapping of different phases of instruction. Making use of the modified Harvard architecture, extensive pipelining is employed in the TMS processor. In an n-level pipelining n instructions are executed in an overlapped parallel fashion in which each instruction is in a different stage of its execution. The TMS320C25 uses a three-stage pipeline.

The FFT algorithm selected for implementation on the TMS320C25 is a 64-point complex FFT optimised for execution speed. The algorithm is a three butterfly radix-4 FFT with straight-line code. The radix-2 butterfly shown in Figure 5.3.1 (or 5.3.2) is the smallest element in a radix-2 FFT. The radix of the FFT represents the number of inputs that are combined in each butterfly. The Fast Fourier Transform is usually
explained around the radix-2 algorithm (Figure 5.3.3) for conceptual simplicity. If, however, higher-order radices are used, more computational savings can be achieved. These savings increase with the radix, but there is very little improvement above radix 4. That is why the radix-2 and radix-4 FFTs are the most commonly used FFT algorithms. In radix-4 FFT, each butterfly has 4 inputs and 4 outputs, essentially combining two stages of a radix-2 algorithm in one. This combination is shown graphically in Figure 5.3.9. Although four radix-2 butterflies are combined into one radix-4 butterfly, the computational load of the latter is less than four times the load of a radix-2 butterfly. An example of a radix-4, 16-point FFT is shown in Figure 5.3.10.

The program implementing the 64-point radix-4 FFT uses the macro capability of the assembler to generate the code, which is almost 2800 lines when expanded. Three macros are used, one for each type of butterfly. All data begins and ends on page 0 of data RAM, and page 1 contains two temporary locations which are addressed by auxiliary registers. The 13-bit immediate coefficients are used as part of the MPYK (multiply immediate) instruction. A time-consuming and unnecessary part of the execution of the FFT program discussed so far is the generation of the sine and cosine terms which are the real and imaginary parts of $W_N$ (Equation 5.3.1). There are basically three approaches to obtaining these sines and cosines. The first is to generate or calculate them as needed. The second is to generate some sine and cosine values and then update them as needed. The fastest way is to precalculate the sine and cosine values and fetch them from a table as needed. A table lookup program is given in Appendix 5. Next the bit-reversal problem is addressed. It is evident from Figure 5.3.3 (and Figure 5.3.9) that the inputs (or outputs) of an FFT are not in sequential order, i.e., they are scrambled. The scrambling of the data addressing is a direct result of the radix-2 derivation (or radix-4) derivation. Actually, this scrambling occurs in a very systematic way, called bit-reversed order. If you express the indices of a scrambled sequence in binary and you reverse this number, the result is the order that this particular point occupies. For instance, in Figure 5.3.3, $X(3)$ occupies the sixth position in the output (when counting from the zero position). In binary form, $3_{10}=011_2$, and if bit-reversed, then $110_2=6_{10}$, which is the position that $X(3)$ occupies. Figure 5.3.3 shows that the index of each input is actually bit reversed, as shown in Table 5.3.1a.
In the case of the 16-point, Radix-4, Decimation in Frequency FFT (Figure 5.3.10), the inputs are in order, but the order of the output sequence is scrambled. If the indices of the outputs are represented in base 4 (which uses digits 0, 1, 2), they can be unscrambled using digit reversal. For example, X(8) would be swapped with X(2), since (8)_D in decimal represents (20)_4 in base 4. Digits 0 and 1 would be reversed to yield (02)_4, which is (02)_D in decimal. Similarly, X(4) would be swapped with X(1), and so on, as shown in Table 5.3.1b.

In general, bit reversal or data scrambling must be performed either at the input stage on the time samples x(n) or at the output samples X(k). Bit reversal can be performed in place and such a process generally requires the use of one temporary data memory location. Because of its double-precision and its versatile instruction set the TMS320C25 processor can perform in-place bit reversal or data scrambling without the use of a temporary data memory location. Although bit-reversal can be regarded as a separate task performed either in at the input or output stage of an FFT implementation, some FFT algorithms exist with bit reversal as an integral part [14]. Such algorithms are said to be in-place and in-order, and they tend to have higher execution speeds than that of the FFT and bit-reversal algorithms executed separately.

Unlike non real-time FFT applications where data samples to be transformed are assumed to already be in data memory, real-time applications demand careful considerations of data input/output and system memory utilisation. The TMS320C25 has 544 words of on-chip data RAM, organised into two 256-word blocks and one 32-word block that can be used as scratch-pad locations. In non real-time applications, input/output data buffering is generally required. For small transforms sizes, up to 128-point complex FFTs, the double-buffering technique, shown in Figure 5.3.12, can be used for real-time applications without the need of any external data memory. The two on-chip RAM blocks are organised into Buffer A and Buffer B respectively. Now, let us consider the case of a 64-point complex FFT. Real-time input data to be transformed can be grouped into 'frames' of 128 words (128 complex inputs) read into either Buffer A or Buffer B, depending on which one is not currently being used by the FFT program. The idea is to use the two on-chip RAM blocks alternatively as I/O and transform buffers. Assuming that the frame of data in buffer A, the current transform
buffer, is being transformed in-place, a software flag is then set to indicate that Buffer B can now be used as the current I/O buffer. This means that while time-domain data is read into Buffer B, the current I/O buffer, previous transformed data in Buffer B must be transferred out at the same time to make room for the incoming data. This can be accomplished efficiently if the I/O transfers are sequential and organised in a back-to-back manner (i.e., an output operation followed by an input operation). Resetting the flag indicates that the roles of Buffer A and Buffer B are now reversed. In this case, Buffer B now has a full frame of input data ready to be transformed, while Buffer A has a full frame of transformed frame (spectral samples) ready to be transferred out to make room for more incoming time-domain data.

ii. IFFT: The procedure used to develop the FFT algorithm can be extended to find the inverse discrete Fourier transform (IDFT) defined in Equation 5.3.1b. It can be seen by comparing Equations 5.3.1a and 5.3.1b, that the same FFT algorithm can be used to compute the IDFT by noting two differences: a scaling factor of $1/N$, and $W_N^{-nk}$ instead of $W_n^{-nk}$. Thus, if the forward FFT is implemented with scaling, the resulting values in the frequency domain are $(1/N)X(k)$ and not $X(k)$, and therefore, for the inverse FFT, no scaling should be applied in order to get the original signal back. On the other hand, if the forward FFT has not been scaled, the inverse FFT must be scaled. The negative exponent of the twiddle factors implies that the values of $\sin(X)$ will have the opposite sign from that in the forward FFT. One way to implement the inverse FFT is to have an additional table with the negatives of $\sin(X)$. Hence the same flow graphs for the FFT can be used for the IDFT, if the twiddle factor $W_N$ is replaced by its complex conjugate $W_N^*$ and appropriate scaling by $1/N$ is then applied. However a more practical way of performing the IFFT will now be explained. Multiplying both sides of Equation 5.3.1b by $N$ and taking the complex conjugate, then the IDFT can be calculated by using the FFT algorithm. Equations 5.3.4 and 5.3.5 describe this procedure:

$$ N x^*(n) = \sum_{n=0}^{N-1} X^*(k) W^{-nk} $$

5.3.4
writing \( W_N \) as \( W \). Solving Equation 5.3.4 results in a rule for performing the IDFT.

\[
X(n) = \frac{1}{N} \left[ \sum_{n=0}^{N-1} X^*(k) W^{nk} \right]^* 
\]

5.3.5

The rule that Equation 5.3.5 implies is: Conjugate the input \( X(k) \), take the DFT, then take the conjugate of the summation and divide it by \( N \). In this form, there is no need to have an additional table for \( \sin(X) \). As Equation 5.3.5 demonstrates only one FFT program is necessary to do both the FFT and IFFT transforms.

Another function that is needed by the algorithm is the log function. There are many ways to calculate an approximation to a log function; using a polynomial approximation, a look-up table, linear interpolation to name but a few. The implementation of the 'log' function required careful consideration, since there is no such function on the processor. A look-up table would provide a good solution as speed is concerned, but it was ruled out since it requires an excessive amount of memory, to achieve high accuracy. It was felt that the best solution for this application, as far as accuracy (but unfortunately not speed) is concerned was to calculate the required logarithms using polynomial approximation (See A5.1. Appendix 5). The expansion is of the form[15]

\[
\ln(1+z) = z - \frac{1}{2} z^2 + \frac{1}{3} z^3 - \ldots 
\]

5.3.6

\(|z| \leq 1, \ z \neq -1\)

or equivalently,

\[
\ln z = (z - 1) - \frac{1}{2} (z - 1)^2 + \frac{1}{3} (z - 1)^3 + \ldots 
\]

5.3.7

\(|z - 1| \leq 1, \ z \neq 0\)

where \( z \) is a complex number.
\[ \ln z = (z - 1) - \frac{1}{2}(z - 1)^2 + \frac{1}{3}(z - 1)^3 + \ldots \quad 5.3.7 \]

\[ |z - 1| \leq 1, \quad z \neq 0 \]

where \( z \) is a complex number.

The approximation series that implements the series expansions, denoted by Equations 5.3.6 or 5.3.7, on the TMS processor is given by

\[ \ln(1 + z) = \sum_{i=1}^{8} a_i z^i + e(x) \]

\[ 0 \leq z < 1, \quad |e(x)| < 3 \times 10^{-8} \quad 5.3.8 \]

The expansion for \( \ln z \) (Equation 5.3.7) can be used if the transformation \( z \rightarrow z - 1 \) is applied.

\[ \ln(z) = \sum_{i=1}^{\infty} a_i (z - 1)^i + e(x) \]

\[ 0 \leq z < 1, \quad |e(x)| < 3 \times 10^{-8} \quad 5.3.9 \]

The program routine (Appendix 5) that implements Equation 5.3.9 (Appendix 5) performs the approximation with seven decimal digits accuracy, which was the accuracy sought for the proper operation of the Cepstrum algorithm.

iii. EXP: The exponential function has been approximated by the following series expansion [15]

\[ e^t = \exp(z) = 1 + \frac{z}{1!} + \frac{z^2}{2!} + \frac{z^3}{3!} + \ldots, \quad 5.3.10 \]

where \( e \) is the real number such that \( \int_1^e \frac{dt}{t} = 1 \).

The approximation that implements the series in Equation 5.3.10 on the TMS320 processor is given by [15]

\[ \exp(z) = \sum_{i=0}^{7} a_i z^i + e(x), \]

\[ 0 \leq z < \ln(2), \quad |e(x)| < 2 \times 10^{-19} \quad 5.3.11 \]
5.3.3.2 Second part of Cepstrum algorithm: The second part of the Cepstrum algorithm requires a deconvolution (to derive a set of coefficients necessary for the adjustment of the prefilter) and a convolution (to calculate the minimum phase impulse response of the channel and filter combined).

The convolution can be implemented on a TMS320 as a finite impulse response (FIR) filter. The coefficients of the filter may be stored anywhere in program memory (reconfigurable on-chip RAM, on-chip ROM or external memories). When the coefficients are stored in on-chip ROM or externally, the entire on-chip data RAM may be used to store the sample sequence. Ultimately, this allows filters of up to 512 taps to be implemented on the TMS320 processor. Execution of the filter will be at full speed, of say 100 ns per tap for the TMS320C25, as long as the memory supports full-speed execution.

The deconvolution can be implemented as a series of divisions. Unfortunately, the TMS320C25 does not have a hardware divider and so, division has to be performed in software. Integer division can be performed using repeated subtraction by making use of the assembly language command ‘SUBC’. A simple integer division program, that will run on the TMS320C25, will now be described. It assumes that both the numerator and the denominator are positive 16-bit integers and that the numerator is greater than the denominator. The complete division process takes fifteen machine cycles to perform, excluding any extra operations needed to check for divide by zero, calculation of the sign of the division and a check to see if the numerator is greater than the denominator. The high byte of the accumulator is first loaded with the numerator and the low byte is zeroed. The denominator is then multiplied by $2^{15}$ and conditionally subtracted from the contents of the accumulator (using the SUBC command). If the subtraction gives a negative result then the contents of the accumulator is left as it was before the subtraction and then shifted one place to the left with the least significant bit filled (LSB) with a ‘0’. If the result of the subtraction is positive then the accumulator is loaded with the difference, it is then shifted to the left one place and a ‘1’ loaded into the LSB. This subtraction process is repeated fifteen times and, after the last subtraction, the low byte of the accumulator will hold the result of the division scaled up by $2^{15}$ and its high byte will hold the remainder scaled up by $2^{15}$. 
and, after the last subtraction, the low byte of the accumulator will hold the result of the division scaled up by $2^{15}$ and its high byte will hold the remainder scaled up by $2^{15}$.

### 5.3.4 TMS320C30 IMPLEMENTATION

The TMS320C30 [35,36] is Texas Instruments third-generation member of the TMS320 family of compatible digital signal processors. It shares all the characteristics of the TMS320C25 with the additional advantages of even higher computational rate (at a level of performance that, at onetime, was the exclusive domain of supercomputers), and of greater dynamic range since it supports floating-point arithmetic. The implementation of the Cepstrum algorithm on the TMS320C30 follows closely the one on the TMS320C25. However, some of the operations now, are implemented in such a way as to exploit the floating-point capability of the processor. For example one major difference is the way in which divisions are performed; In the case of the TMS320C30, a floating-point divide routine implements the divide operation, $a = b/c$. This is done by calculating the reciprocal or inverse of the divisor $c$. Then

$$a = b \left(1 / c \right) \quad 5.3.12$$

can be computed (See A5.2 in Appendix 5). A special function for the computation of the inverse is approximated by iteratively solving a particular non-linear equation say, $g(x) = 0$. (A5.2 Appendix 5).

### 5.4 COMPUTER SIMULATION TESTS AND RESULTS

The sampled impulse responses of eight telephone channels (derived in Appendix A4.2–Chapter 4) were used in the computer simulations. In the derivation of the sampled impulse responses, the actual transmission of the data over the various channels was simulated using 64-bit floating point arithmetic. To test for limited precision the complex valued analogue signals (represented by 64-point floating variables) were converted into digital signals, by first scaling it, and then converting
its real and imaginary parts into 16-bit integers. In the computer simulation programs this is done by adding a round-off noise to the number sequence (see Section 5.3.1).

However an ‘exact’ simulation of the round-off error would be highly desirable. Two functions were available in Matlab that implement rounding and truncation (chopping) respectively. The function ‘CHOP(x,n) rounds the elements of X to n significant figures. Another implementation of the same function implements truncation. The function \( \text{CHOP}(P) \) now truncates the last \( 4*P \) bits of each floating-point number for \( 0 < P < 13 \). This two forms of the function CHOP are direct implementations of the definition of rounding and truncation. Suppose a number \( v \) which has been normalised so that \( |v| \leq 1 \) has the binary expansion

\[
v = -v_0 + \sum_{k=1}^{\infty} v_k 2^{-k}
\]

where \( v_k = 1 \) or \( 0 \). To approximate \( v \) by a ‘word’ of only \( t \) bits by rounding, a 1 or 0 is first added to the \( t \)th bit \( v_{t+1} \) according to whether the \( (t+1) \)th bit \( v_t \) is 1 or 0. Then, only the first bits of the result are kept. In chopping, those bits beyond the most significant \( t \) bits are simply dropped.

In the ‘actual’ implementation on the TMS320C25 the LACT can be used to convert a floating-point number to a fixed-point number. Now the effect of limited precision was first simulated by adding a round off noise to the output signal at various stages of the algorithm. This was sort of ‘theoretical approach and gave a useful indication of the ‘worth’ of the adjustment filter techniques when running with limited precision. However, two other more ‘realistic’ and certainly more accurate ways for simulating the limited precision arithmetic were employed in the simulation programs. These were represented by the Matlab function chop which has two forms one implementing rounding and the second truncation (chopping). Both were used and the results were close (although slightly worse for the case of truncation) between them and when compared with the ‘theoretical’ ones. The discrepancy between the results of truncation and rounding was to be expected, since the error introduced by chopping is more serious than that introduced by rounding due to bias; that is why, chopping arithmetic is not commonly used. The reason that was used here in addition to rounding, was to provide a lower bound for the simulation tests.

However, since the results of both rounding and truncation are not that dissimilar, only the results when truncation was used are shown here.
Running the Cepstrum algorithm with 64-bit floating point arithmetic provides the range of numbers occurring during the various stages of the algorithm, when operating over eight different telephone channels. For example, for channel 1 the maximum number is 1.3870 and the minimum number is 3.7091x10^7. The minimum number for the 16-bit fixed-point implementation (accepting as maximum number the number 1.3870), is 2.1165x10^-5. The maximum (and minimum allowed) numbers occurring when running the Cepstrum algorithm for channels 2 to 8 are shown in Table 5.3.2. After appropriate scaling the maximum number for the 16-bit implementation is 3.3649 and the minimum number occurring is 6.0087x10^-5. The minimum number (allowed for the maximum number 2.6364) is 5.1345x10^-5. The effect of scaling for channels 2 to 8, in order to successfully implement the Cepstrum method with 16-bit fixed point arithmetic can be seen in Table 5.3.3.

Tables 5.4.1-5.37 show the overall simulation for a 16-bit implementation of the selected algorithm(s), for the adjustment of the prefilter. The simulation was designed as follows. It has become evident that the Cepstrum algorithm is the best possible candidate for inclusion in a real time system. This is justified by previous simulation (Chapter 4) that clearly showed that with floating-point arithmetic the Cepstrum algorithm overall is more accurate and requires fewer arithmetic operations, than any of the other root finding algorithms or spectral factorisation techniques. However, two very powerful root finding techniques (i.e. the Clark-Hau algorithm and the Laguerre method) were employed in the simulation of Channels 1-8 anew, with 16-point fixed arithmetic. This was deemed to be necessary in case their accuracy was not that much affected, while on the other hand, because of the limited precision, the number of iterations required would be smaller. Tables 5.4.1, 3, 5, 7, 9, 11 and 12 show the Root Accuracy with limited precision arithmetic of these two methods over Channels 1-8, and at the same time the number of iterations required to locate the roots with the indicated accuracy. Tables 5.4.2,4,6,8,10 and 13 show the values ψ1, ψ2 and ψ3 for limited precision arithmetic for Channels 1-8. Note that the number of filter taps is 30 for Channels 1-4, 40 for Channel 5, 30 for Channel 6, for Channel 7 is 50 and for Channel 8 is 30 in all tests.
FILTER IMPLEMENTATION

Compared with their equivalent Tables (Chapter 4), it is clear that although their accuracy is reduced (in some cases significantly) with limited precision, the number of iterations remains almost the same, as when they were running with 64-point floating point arithmetic.

In particular, Tables 5.4.1-6 collectively show the ‘worth’ of the Clark-Hau algorithm over channels 1-3. The Clark-Hau method was used in the simulation, although it has become evident from the relevant Tables of the previous Chapter that it is less accurate and requires more iterations than the Laguerre method, when the latter is applied as suggested by the author, over Channels 1-8. However, the Clark-Hau method was used to give a lower bound of the performance of the root finding methods investigated in this work. For Channel 4, the Clark-Hau method failed to give any valid results and inevitably Laguerre came into play. However, although Laguerre is an unquestionably better method than the Clark-Hau algorithm, it also failed when applied over Channel 4 with 16-bit limited precision as implied by Table 5.4.8 (value of $\psi_3$ is 2.97). Over Channel 5, the Laguerre method failed to locate one root (Table 5.4.9), thus yielding the low accuracy indicated by the values of $\psi_1$, $\psi_2$ and $\psi_3$ in Table 5.4.10. For Channel 6, the algorithm fails in the most definite way since it manages to find only one root (starting point 0.5, root accuracy -40.16, 5 iterations). It appears that is performing better over the next Channel (Channel 7), since it manages to locate 9 roots out of the wanted 12 (Table 5.4.11), but the accuracy with which adjusts the prefilter is extremely low (given by $\psi_3 = -1.84$). Similar observations apply to Channel 8 (Table 5.4.13) where the Laguerre method missed seven roots (indicated by F in Table 5.4.12), thus giving the disastrous $\psi_3 = -1.24$ (Table 5.4.13). It has become perfectly clear by now, that no root finding algorithm can satisfactorily adjust channels 4-8 (which are the most severely distorted channels).

Tables 5.4.14-5.4.21 show the accuracy of the Cepstrum technique (indicated by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with limited precision arithmetic. It is evident from the simulation tables that this method is capable of properly adjusting the prefilter, even when operating over the poorer quality telephone channels (Channels 4-8). Therefore, up to this point it is undisputably the best candidate for fixed point implementation. However, it is interesting to see how the algorithm behaves when exposed to noise and ignoring for the moment the reduced precision, in other words full precision is assumed once more.
FILTER IMPLEMENTATION

Tables 5.4.22-5.4.29 show the behaviour of the algorithm under different Signal to Noise Ratio (SNR) conditions. Again it is obvious that the algorithm survives even the worst SNR of 20 dB over all eight channels. The combined effect of the limited precision and noise (which is the situation that the algorithm deals with when in a real time system) is shown in Tables 5.4.30-5.4.32. The Cepstrum algorithm adjusts the prefilter over channels 1-3 even at a SNR of 20 dB. It fails over Channels 4-8 but this is to be expected given the severe distortion experienced over these channels as well as the extremely unfavourable SNR condition of 20 dB.

The actual implementation of the Cepstrum algorithm on the TMS320C25 was made easier because of the availability of excellent code for the FFT and IFFT blocks which form the 'core' of the Cepstrum algorithm. In addition, routines implementing other functions such as exponentiation, division, logarithms, etc. were also widely available. Therefore, the task of developing the algorithm in a complete program for the TMS320C25 reduces really in modifying these routines, implement some extra vector multiplications and assembling them in a complete program. Thus, it was possible to 'really' estimate the time that the algorithm takes when implemented on a TMS320 chip. The time that the algorithm takes when running with full precision can be easily calculated by utilising Tables 5.4.33-34 which show the number of floating point operations each time the algorithm operates over a particular channel. Table 5.4.35 shows the 'calculated' timing of the algorithm over Channels 1-8, when running with limited precision (16 bits). Tables 5.4.36-37 show the 'actual' timing of the algorithm when implemented on a TMS320C25.

Finally, Table 5.4.38-39 show the actual timing of the algorithm when implemented on a TMS320C30 floating-point Digital Signal Processor. Note that Tables 5.4.36-39 have two columns labelled as BEST and WORST. They signify the best (smallest number of cycles possible) and worst (greatest number of cycles) times achieved by the Cepstrum algorithm when it is 'actually' implemented on the TMS320C25/30 chips. Note also the close agreement between simulated and actual times indicated in the relevant Tables.
5.4.1 Conclusions Three different adjustment techniques were proposed for hardware implementation on the TMS320C25/30 digital signal processors. The first two (i.e. the Clark-Hau and Laguerre root finding methods) were not able to adjust the digital receiver, when they were considered over the more severely distorted channels. The remaining third method the Cepstrum algorithm *, performed well over all eight telephone channels (considered in the simulation) when simulated with limited precision. Perhaps, the most surprising result was the ability of the algorithm to adjust the prefilter at very low SNR. Also, very encouraging was the fact that it performed well for the first three channels even when both limited precision and a very low SNR (as low as 20 dB) were assumed.

* Simulation results for the N I T algorithm over HF channels are given in Appendix 5-Sections A5.4.3-4.
### CHANNEL 1

<table>
<thead>
<tr>
<th>Root</th>
<th>Iterations</th>
<th>Number of Bits used</th>
<th>S.point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>64, 32</td>
<td>-184.67, -142.76, -70.76, -64.12, -45.89, -32.00, 0.5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>32, 16</td>
<td>-205.38, -146.28, -62.11, -51.87, -41.32, -46.35, 0.5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>16, 14</td>
<td>-198.44, -153.00, -75.17, -63.32, -43.23, -60.67, -0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Root</th>
<th>Iterations</th>
<th>Bits used=8</th>
<th>S.Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>-17.00</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>-25.44</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>-30.15</td>
<td>-0.5</td>
</tr>
</tbody>
</table>

Table 5.4.1 Root Accuracy of Clark-Hau, with Limited Precision Arithmetic, over Channel 1

### CHANNEL 1

<table>
<thead>
<tr>
<th>Taps=40</th>
<th>Number of bits used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
</tr>
<tr>
<td>(\psi_1)</td>
<td>-117.67</td>
</tr>
<tr>
<td>(\psi_2)</td>
<td>-152.98</td>
</tr>
<tr>
<td>(\psi_3)</td>
<td>-151.65</td>
</tr>
</tbody>
</table>

Table 5.4.2 Accuracy of Clark-Hau (given by the values of \(\psi_1\), \(\psi_2\) and \(\psi_3\)) with Limited Precision Arithmetic over Channel 1

### CHANNEL 2

<table>
<thead>
<tr>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>0.5</td>
<td>-147.48</td>
<td>3</td>
<td>14</td>
<td>0.5</td>
<td>-74.62</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0.5</td>
<td>-129.13</td>
<td>1</td>
<td>7</td>
<td>0.5</td>
<td>-51.23</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.5</td>
<td>-169.88</td>
<td>2</td>
<td>5</td>
<td>0.5</td>
<td>-60.96</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-0.5</td>
<td>-150.42</td>
<td>4</td>
<td>5</td>
<td>0.5</td>
<td>-68.95</td>
</tr>
</tbody>
</table>
### CHANNEL 2

<table>
<thead>
<tr>
<th>Taps= 40</th>
<th>Number of bits used</th>
<th>32</th>
<th>16</th>
<th>14</th>
<th>12</th>
<th>10</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi_1 )</td>
<td>(-149.11)</td>
<td>(-68.51)</td>
<td>(-66.63)</td>
<td>(-43.91)</td>
<td>(-41.92)</td>
<td>(-31.64)</td>
<td></td>
</tr>
<tr>
<td>( \psi_2 )</td>
<td>(-151.18)</td>
<td>(-80.83)</td>
<td>(-70.15)</td>
<td>(-47.77)</td>
<td>(-47.71)</td>
<td>(-30.51)</td>
<td></td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>(-109.57)</td>
<td>(-74.75)</td>
<td>(-68.70)</td>
<td>(-46.82)</td>
<td>(-42.90)</td>
<td>(-31.64)</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4.4 Accuracy of Clark-Hau (given by the values of \( \psi_1 \), \( \psi_2 \) and \( \psi_3 \)) with Limited Precision Arithmetic over Channel 2

### CHANNEL 3

<table>
<thead>
<tr>
<th>Number of bits used</th>
<th>64</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Itrs</td>
<td>S.point</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>(-0.5)</td>
</tr>
</tbody>
</table>

Table 5.4.3 Root Accuracy of Clark-Hau, with Limited Precision Arithmetic, over Channel 2
<table>
<thead>
<tr>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0.5</td>
<td>-58.10</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0.5</td>
<td>-71.89</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>0.5</td>
<td>-55.18</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>-0.5</td>
<td>-73.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>-0.5</td>
<td>-45.14</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>0.5</td>
<td>-43.10</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0.5</td>
<td>-41.74</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0.5</td>
<td>-67.77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>16</td>
<td>-0.5+i*0.5</td>
<td>-43.47</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>0.5</td>
<td>-42.72</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>0.5</td>
<td>-42.21</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.5</td>
<td>-45.84</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.5</td>
<td>18</td>
<td>-17.10</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>20</td>
<td>-2.21</td>
</tr>
<tr>
<td>2</td>
<td>-0.5</td>
<td>16</td>
<td>-1.32</td>
</tr>
</tbody>
</table>

Table 5.4.5 Root Accuracy of Clark-Hau, with Limited Precision Arithmetic, over Channel 3
Table 5.4.6 Accuracy of Clark-Hau (given by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with Limited Precision Arithmetic over Channel 3

<table>
<thead>
<tr>
<th>Number of bits used</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>14</th>
<th>12</th>
<th>10</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>-98.45</td>
<td>-98.00</td>
<td>-73.95</td>
<td>-52.76</td>
<td>-44.21</td>
<td>-38.65</td>
<td>-24.23</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-172.35</td>
<td>-137.10</td>
<td>-76.57</td>
<td>-53.79</td>
<td>-46.74</td>
<td>-41.64</td>
<td>-25.76</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-173.82</td>
<td>-138.40</td>
<td>-71.36</td>
<td>-54.13</td>
<td>-46.93</td>
<td>-41.61</td>
<td>-25.92</td>
</tr>
</tbody>
</table>

Table 5.4.6 Accuracy of Clark-Hau (given by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with Limited Precision Arithmetic over Channel 3

Table 5.4.7 Root Accuracy of Laguerre, with Limited Precision Arithmetic, over Channel 4

| Channel 4 |
|-----------------|-------|----------|-------|----------|-------|----------|-------|----------|
| Root | Itrs | S.point | Accuracy | Root | Itrs | S.point | Accuracy |
| 3 | 4 | 0.5 | -283.82 | 3 | 4 | 0.5 | -149.54 |
| 4 | 5 | 0 | -291.48 | 4 | 5 | 0 | -141.72 |
| 1 | 4 | 0.5 | -301.00 | 1 | 4 | 0.5 | -143.60 |
| 5 | 6 | 0.5 | -296.16 | 5 | 6 | 0.5 | -130.52 |
| 6 | 7 | 0.5 | -285.84 | 6 | 7 | 0.5 | -155.10 |
| 2 | 5 | 0.5 | -285.46 | 2 | 5 | 0.5 | -140.88 |
| 8 | 7 | 0.5 | -297.51 | 8 | 7 | 0.5 | -145.86 |
| 7 | 6 | -0.5 | -291.48 | 7 | 6 | -0.5 | -134.93 |

Table 5.4.7 Root Accuracy of Laguerre, with Limited Precision Arithmetic, over Channel 4
<table>
<thead>
<tr>
<th>CHANNEL 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Taps=40</strong></td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>$\psi_1$</td>
</tr>
<tr>
<td>$\psi_2$</td>
</tr>
<tr>
<td>$\psi_3$</td>
</tr>
</tbody>
</table>

Table 5.4.8 Accuracy of Laguerre (given by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with Limited Precision Arithmetic over Channel 4

<table>
<thead>
<tr>
<th>CHANNEL 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Root Itrs S.point Accuracy</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

Table 5.4.9 Root Accuracy of Laguerre, with Limited Precision Arithmetic, over Channel 5

<table>
<thead>
<tr>
<th>CHANNEL 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Taps=50</strong></td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>$\psi_1$</td>
</tr>
<tr>
<td>$\psi_2$</td>
</tr>
<tr>
<td>$\psi_3$</td>
</tr>
</tbody>
</table>

Table 5.4.10 Accuracy of Laguerre (given by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with Limited Precision Arithmetic over Channel 5
### CHANNEL 7

<table>
<thead>
<tr>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>0.5+i*0.5</td>
<td>-42.00</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>-0.5</td>
<td>-34.12</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>-0.5+i*0.5</td>
<td>-26.11</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>-0.5+i*0.5</td>
<td>-31.50</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>-0.5+i*0.5</td>
<td>-33.38</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>4*</td>
<td>-64.74</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1* Root</td>
<td>-19.44</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>2* Root</td>
<td>-28.39</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-0.5-i*0.5</td>
<td>-20.16</td>
</tr>
</tbody>
</table>

Table 5.4.11 Root Accuracy of Laguerre, with Limited Precision Arithmetic, over Channel 7

### CHANNEL 8

<table>
<thead>
<tr>
<th>Root</th>
<th>Itrs</th>
<th>S.point</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>-123.93</td>
<td>-52.78</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-123.43</td>
<td>-79.26</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>-90.89</td>
<td>-79.69</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>-97.75</td>
<td>-1.24</td>
</tr>
</tbody>
</table>

Table 5.4.12 Root Accuracy of Laguerre, with Limited Precision Arithmetic, over Channel 8

### CHANNEL 8

<table>
<thead>
<tr>
<th>Taps=60</th>
<th>Number of bits used</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-52.78</td>
</tr>
<tr>
<td>16</td>
<td>-79.26</td>
</tr>
<tr>
<td>-96.69</td>
<td>-1.24</td>
</tr>
</tbody>
</table>

Table 5.4.13 Accuracy of Laguerre (given by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with Limited Precision Arithmetic over Channel 8
### CHANNEL 1

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>$\Psi_1$</th>
<th>$\Psi_2$</th>
<th>$\Psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-62.95</td>
<td>-146.99</td>
<td>-147.00</td>
</tr>
<tr>
<td>16</td>
<td>-62.22</td>
<td>-72.19</td>
<td>-72.00</td>
</tr>
<tr>
<td>14</td>
<td>-51.75</td>
<td>-51.88</td>
<td>-51.72</td>
</tr>
<tr>
<td>12</td>
<td>-47.46</td>
<td>-47.07</td>
<td>-46.92</td>
</tr>
<tr>
<td>10</td>
<td>-35.65</td>
<td>-34.91</td>
<td>-34.76</td>
</tr>
<tr>
<td>8</td>
<td>-15.18</td>
<td>-14.21</td>
<td>-14.12</td>
</tr>
</tbody>
</table>

Table 5.4.14 Accuracy of Cepstrum (given by the values of $\psi_1$, $\psi_2$, and $\psi_3$) with Limited Precision Arithmetic over Channel 1

### CHANNEL 2

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>$\Psi_1$</th>
<th>$\Psi_2$</th>
<th>$\Psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-55.28</td>
<td>-117.33</td>
<td>-117.32</td>
</tr>
<tr>
<td>16</td>
<td>-54.45</td>
<td>-62.12</td>
<td>-61.84</td>
</tr>
<tr>
<td>14</td>
<td>-53.46</td>
<td>-57.91</td>
<td>-57.64</td>
</tr>
<tr>
<td>12</td>
<td>-48.15</td>
<td>-48.00</td>
<td>-47.78</td>
</tr>
<tr>
<td>10</td>
<td>-36.00</td>
<td>-34.21</td>
<td>-34.00</td>
</tr>
<tr>
<td>8</td>
<td>-19.15</td>
<td>-17.74</td>
<td>-17.40</td>
</tr>
</tbody>
</table>

Table 5.4.15 Accuracy of Cepstrum (given by the values of $\psi_1$, $\psi_2$, and $\psi_3$) with Limited Precision Arithmetic over Channel 2

### CHANNEL 3

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>$\Psi_1$</th>
<th>$\Psi_2$</th>
<th>$\Psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-63.48</td>
<td>-67.24</td>
<td>-66.13</td>
</tr>
<tr>
<td>16</td>
<td>-54.78</td>
<td>-65.57</td>
<td>-65.40</td>
</tr>
<tr>
<td>14</td>
<td>-32.10</td>
<td>-36.75</td>
<td>-36.40</td>
</tr>
<tr>
<td>12</td>
<td>-43.69</td>
<td>-49.63</td>
<td>-43.69</td>
</tr>
<tr>
<td>10</td>
<td>-20.93</td>
<td>-23.65</td>
<td>-23.12</td>
</tr>
<tr>
<td>8</td>
<td>-10.88</td>
<td>-11.00</td>
<td>-10.43</td>
</tr>
</tbody>
</table>

Table 5.4.16 Accuracy of Cepstrum (given by the values of $\psi_1$, $\psi_2$, and $\psi_3$) with Limited Precision Arithmetic over Channel 3
### CHANNEL 4

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
<th>( \psi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-32.62</td>
<td>-84.15</td>
<td>-84.17</td>
</tr>
<tr>
<td>16</td>
<td>-32.88</td>
<td>-61.00</td>
<td>-59.98</td>
</tr>
<tr>
<td>14</td>
<td>-33.17</td>
<td>-48.90</td>
<td>-48.00</td>
</tr>
<tr>
<td>12</td>
<td>-29.43</td>
<td>-36.16</td>
<td>-35.36</td>
</tr>
<tr>
<td>10</td>
<td>-16.92</td>
<td>-23.76</td>
<td>-22.85</td>
</tr>
<tr>
<td>8</td>
<td>-8.26</td>
<td>-11.23</td>
<td>-10.32</td>
</tr>
</tbody>
</table>

Table 5.4.17 Accuracy of Cepstrum (given by the values of \( \psi_1 \), \( \psi_2 \), and \( \psi_3 \)) with Limited Precision Arithmetic over Channel 4

### CHANNEL 5

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
<th>( \psi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-25.31</td>
<td>-56.93</td>
<td>-45.23</td>
</tr>
<tr>
<td>16</td>
<td>-25.35</td>
<td>-55.13</td>
<td>-45.25</td>
</tr>
<tr>
<td>14</td>
<td>-25.41</td>
<td>-48.29</td>
<td>-44.56</td>
</tr>
<tr>
<td>12</td>
<td>-25.13</td>
<td>-37.50</td>
<td>-37.32</td>
</tr>
<tr>
<td>10</td>
<td>-21.00</td>
<td>-25.00</td>
<td>-24.76</td>
</tr>
<tr>
<td>8</td>
<td>-10.91</td>
<td>-11.83</td>
<td>-10.83</td>
</tr>
</tbody>
</table>

Table 5.4.18 Accuracy of Cepstrum (given by the values of \( \psi_1 \), \( \psi_2 \), and \( \psi_3 \)) with Limited Precision Arithmetic over Channel 5

### CHANNEL 6

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
<th>( \psi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-22.13</td>
<td>-61.92</td>
<td>-61.92</td>
</tr>
<tr>
<td>16</td>
<td>-22.14</td>
<td>-60.56</td>
<td>-57.39</td>
</tr>
<tr>
<td>14</td>
<td>-22.18</td>
<td>-48.61</td>
<td>-47.87</td>
</tr>
<tr>
<td>12</td>
<td>-19.12</td>
<td>-30.00</td>
<td>-35.48</td>
</tr>
<tr>
<td>10</td>
<td>-21.00</td>
<td>-23.49</td>
<td>-22.94</td>
</tr>
<tr>
<td>8</td>
<td>-11.76</td>
<td>-10.20</td>
<td>-10.52</td>
</tr>
</tbody>
</table>

Table 5.4.19 Accuracy of Cepstrum (given by the values of \( \psi_1 \), \( \psi_2 \), and \( \psi_3 \)) with Limited Precision Arithmetic over Channel 6
<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>$\Psi_1$</th>
<th>$\Psi_2$</th>
<th>$\Psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-26.74</td>
<td>-47.68</td>
<td>-47.50</td>
</tr>
<tr>
<td>16</td>
<td>-26.65</td>
<td>-47.48</td>
<td>-47.15</td>
</tr>
<tr>
<td>14</td>
<td>-26.30</td>
<td>-44.85</td>
<td>-44.24</td>
</tr>
<tr>
<td>12</td>
<td>-24.58</td>
<td>-35.34</td>
<td>-34.64</td>
</tr>
<tr>
<td>10</td>
<td>-16.87</td>
<td>-22.16</td>
<td>-21.52</td>
</tr>
<tr>
<td>8</td>
<td>-9.00</td>
<td>-10.16</td>
<td>-9.18</td>
</tr>
</tbody>
</table>

Table 5.4.20 Accuracy of Cepstrum (given by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with Limited Precision Arithmetic over Channel 7

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>$\Psi_1$</th>
<th>$\Psi_2$</th>
<th>$\Psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-21.19</td>
<td>-87.88</td>
<td>-85.63</td>
</tr>
<tr>
<td>16</td>
<td>-21.63</td>
<td>-60.41</td>
<td>-59.88</td>
</tr>
<tr>
<td>14</td>
<td>-22.44</td>
<td>-48.31</td>
<td>-47.79</td>
</tr>
<tr>
<td>12</td>
<td>-19.36</td>
<td>-35.79</td>
<td>-35.35</td>
</tr>
<tr>
<td>10</td>
<td>-9.37</td>
<td>-23.32</td>
<td>-22.87</td>
</tr>
<tr>
<td>8</td>
<td>-0.07</td>
<td>-10.23</td>
<td>-10.56</td>
</tr>
</tbody>
</table>

Table 5.4.21 Accuracy of Cepstrum (given by the values of $\psi_1$, $\psi_2$ and $\psi_3$) with Limited Precision Arithmetic over Channel 8
### CHANNEL 1

<table>
<thead>
<tr>
<th>S/N</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-28.39</td>
<td>-22.23</td>
<td>-20.88</td>
</tr>
<tr>
<td>40</td>
<td>-47.85</td>
<td>-45.00</td>
<td>-42.48</td>
</tr>
<tr>
<td>60</td>
<td>-61.86</td>
<td>-64.26</td>
<td>-61.90</td>
</tr>
<tr>
<td>80</td>
<td>-62.95</td>
<td>-83.87</td>
<td>-81.79</td>
</tr>
</tbody>
</table>

Table 5.4.22 Accuracy of Cepstrum, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios over Channel 1

### CHANNEL 2

<table>
<thead>
<tr>
<th>S/N</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-26.34</td>
<td>-21.89</td>
<td>-21.76</td>
</tr>
<tr>
<td>40</td>
<td>-46.31</td>
<td>-42.57</td>
<td>-41.95</td>
</tr>
<tr>
<td>60</td>
<td>-55.00</td>
<td>-63.54</td>
<td>-62.35</td>
</tr>
<tr>
<td>80</td>
<td>-55.28</td>
<td>-83.94</td>
<td>-82.38</td>
</tr>
</tbody>
</table>

Table 5.4.23 Accuracy of Cepstrum, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios over Channel 2

### CHANNEL 3

<table>
<thead>
<tr>
<th>S/N</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-29.41</td>
<td>-22.65</td>
<td>-20.81</td>
</tr>
<tr>
<td>40</td>
<td>-49.16</td>
<td>-42.79</td>
<td>-40.76</td>
</tr>
<tr>
<td>60</td>
<td>-62.47</td>
<td>-61.17</td>
<td>-58.74</td>
</tr>
<tr>
<td>80</td>
<td>-63.24</td>
<td>-66.79</td>
<td>-65.42</td>
</tr>
</tbody>
</table>

Table 5.4.24 Accuracy of Cepstrum, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios over Channel 3
### CHANNEL 4

<table>
<thead>
<tr>
<th>S/N</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
<th>( \psi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-24.11</td>
<td>-24.37</td>
<td>-17.94</td>
</tr>
<tr>
<td>40</td>
<td>-30.48</td>
<td>-43.66</td>
<td>-42.85</td>
</tr>
<tr>
<td>60</td>
<td>-32.34</td>
<td>-63.37</td>
<td>-61.94</td>
</tr>
<tr>
<td>80</td>
<td>-32.58</td>
<td>-80.70</td>
<td>-79.75</td>
</tr>
</tbody>
</table>

Table 5.4.25 Accuracy of Cepstrum, given by the values of \( \psi_1, \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios over Channel 4

### CHANNEL 5

<table>
<thead>
<tr>
<th>S/N</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
<th>( \psi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-24.13</td>
<td>-24.86</td>
<td>-17.33</td>
</tr>
<tr>
<td>40</td>
<td>-30.88</td>
<td>-44.11</td>
<td>-31.86</td>
</tr>
<tr>
<td>60</td>
<td>-25.94</td>
<td>-56.64</td>
<td>-55.39</td>
</tr>
<tr>
<td>80</td>
<td>-26.61</td>
<td>-57.61</td>
<td>-52.35</td>
</tr>
</tbody>
</table>

Table 5.4.26 Accuracy of Cepstrum, given by the values of \( \psi_1, \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios over Channel 5

### CHANNEL 6

<table>
<thead>
<tr>
<th>S/N</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
<th>( \psi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-24.21</td>
<td>-24.86</td>
<td>-17.33</td>
</tr>
<tr>
<td>40</td>
<td>-30.88</td>
<td>-44.11</td>
<td>-31.86</td>
</tr>
<tr>
<td>60</td>
<td>-25.94</td>
<td>-56.64</td>
<td>-55.40</td>
</tr>
<tr>
<td>80</td>
<td>-26.61</td>
<td>-57.61</td>
<td>-52.36</td>
</tr>
</tbody>
</table>

Table 5.4.27 Accuracy of Cepstrum, given by the values of \( \psi_1, \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios over Channel 6
### CHANNEL 7

<table>
<thead>
<tr>
<th>S/N</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-19.64</td>
<td>-21.33</td>
<td>-17.78</td>
</tr>
<tr>
<td>40</td>
<td>-24.32</td>
<td>-30.58</td>
<td>-30.21</td>
</tr>
<tr>
<td>60</td>
<td>-25.27</td>
<td>-31.70</td>
<td>-31.60</td>
</tr>
<tr>
<td>80</td>
<td>-25.31</td>
<td>-31.77</td>
<td>-31.66</td>
</tr>
</tbody>
</table>

Table 5.4.28 Accuracy of Cepstrum, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios over Channel 7

### CHANNEL 8

<table>
<thead>
<tr>
<th>S/N</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>-22.56</td>
<td>-23.98</td>
<td>-12.69</td>
</tr>
<tr>
<td>40</td>
<td>-31.24</td>
<td>-43.93</td>
<td>-34.71</td>
</tr>
<tr>
<td>60</td>
<td>-21.25</td>
<td>-66.32</td>
<td>-50.12</td>
</tr>
<tr>
<td>80</td>
<td>-21.18</td>
<td>-84.34</td>
<td>-71.36</td>
</tr>
</tbody>
</table>

Table 5.4.29 Accuracy of Cepstrum, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios over Channel 8

### CHANNEL 1

<table>
<thead>
<tr>
<th>S/N=20</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
<th>$\psi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>-27.12</td>
<td>-22.61</td>
<td>-21.83</td>
</tr>
<tr>
<td>14</td>
<td>-27.19</td>
<td>-22.56</td>
<td>-21.75</td>
</tr>
<tr>
<td>12</td>
<td>-26.37</td>
<td>-22.31</td>
<td>-21.38</td>
</tr>
<tr>
<td>10</td>
<td>-22.34</td>
<td>-20.93</td>
<td>-19.56</td>
</tr>
<tr>
<td>8</td>
<td>-13.24</td>
<td>-11.35</td>
<td>-10.83</td>
</tr>
</tbody>
</table>

Table 5.4.30 Accuracy of Cepstrum, given by the values of $\psi_1$, $\psi_2$ and $\psi_3$, with Limited Precision Arithmetic and Signal-to-Noise Ratio set to 20 dB over Channel 1
### CHANNEL 2

<table>
<thead>
<tr>
<th>S/N=20</th>
<th>(\psi_1)</th>
<th>(\psi_2)</th>
<th>(\psi_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>-25.88</td>
<td>-23.10</td>
<td>-22.26</td>
</tr>
<tr>
<td>16</td>
<td>-25.87</td>
<td>-23.00</td>
<td>-22.23</td>
</tr>
<tr>
<td>14</td>
<td>-25.79</td>
<td>-22.98</td>
<td>-22.14</td>
</tr>
<tr>
<td>12</td>
<td>-25.33</td>
<td>-22.51</td>
<td>-21.68</td>
</tr>
<tr>
<td>10</td>
<td>-22.10</td>
<td>-19.61</td>
<td>-19.23</td>
</tr>
<tr>
<td>8</td>
<td>-13.43</td>
<td>-10.87</td>
<td>-10.70</td>
</tr>
</tbody>
</table>

Table 5.4.31 Accuracy of Cepstrum, given by the values of \(\psi_1\), \(\psi_2\) and \(\psi_3\), with Limited Precision Arithmetic and Signal-to-Noise Ratio set to 20 dB over Channel 2

### CHANNEL 3

<table>
<thead>
<tr>
<th>S/N=20</th>
<th>(\psi_1)</th>
<th>(\psi_2)</th>
<th>(\psi_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>-28.34</td>
<td>-21.68</td>
<td>-19.95</td>
</tr>
<tr>
<td>14</td>
<td>-28.00</td>
<td>-21.73</td>
<td>-19.92</td>
</tr>
<tr>
<td>12</td>
<td>-26.65</td>
<td>-21.75</td>
<td>-19.73</td>
</tr>
<tr>
<td>10</td>
<td>-20.68</td>
<td>-20.75</td>
<td>-17.95</td>
</tr>
<tr>
<td>8</td>
<td>-4.43</td>
<td>-9.76</td>
<td>-10.00</td>
</tr>
</tbody>
</table>

Table 5.4.32 Accuracy of Cepstrum, given by the values of \(\psi_1\), \(\psi_2\) and \(\psi_3\), with Limited Precision Arithmetic and Signal-to-Noise Ratio set to 20 dB over Channel 3
Sign Bit  
\[ \downarrow \]
S | ... | S | ... 
\[ \text{MANTISSA} \quad \text{EXPONENT} \]

Figure 5.2.1a Floating-point number

Position of binary or decimal point

Figure 5.2.1b Fixed-point (integer) number

Position of binary or decimal point

Figure 5.2.1c Scaled fixed-point number
Figure 5.3.1a Butterfly computation for decimation-in-time

Figure 5.3.2 Statistical model for fixed-point roundoff noise in a decimation-in-time butterfly computation.
Figure 5.3.3 Decimation-in-time FFT algorithm.
Figure 5.3.4 Functional block diagram of a Sequential FFT processor.

Figure 5.3.5. The cascade FFT processor
Figure 5.3.6 The parallel Iterative Processor

Figure 5.3.7 The Array Processor
Figure 5.3.8 Minimal Processing System with external data RAM and PROM/EPROM
Figure 5.3.9 Butterfly for radix-4, Decimation-in-Time FFT
Figure 5.3.10 A 16-Point, Radix-4, Decimation-in-Time FFT
### Filter Implementation

#### Table 5.3.1a Bit-Reversal Algorithm for an 8-Point Radix-2 Decimation-in-Time FFT

<table>
<thead>
<tr>
<th>Index</th>
<th>Bit Pattern</th>
<th>Bit-Reversed Pattern</th>
<th>Bit-Reversed Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

#### Table 5.3.1b Radix-4 Digit Reversal for N=16

<table>
<thead>
<tr>
<th>Scrambled</th>
<th>Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(0)</td>
<td>X(0)</td>
</tr>
<tr>
<td>X(4)</td>
<td>X(1)</td>
</tr>
<tr>
<td>X(8)</td>
<td>X(2)</td>
</tr>
<tr>
<td>X(12)</td>
<td>X(3)</td>
</tr>
<tr>
<td>X(1)</td>
<td>X(4)</td>
</tr>
<tr>
<td>X(5)</td>
<td>X(5)</td>
</tr>
<tr>
<td>X(9)</td>
<td>X(6)</td>
</tr>
<tr>
<td>X(13)</td>
<td>X(7)</td>
</tr>
<tr>
<td>X(2)</td>
<td>X(8)</td>
</tr>
<tr>
<td>X(6)</td>
<td>X(9)</td>
</tr>
<tr>
<td>X(10)</td>
<td>X(10)</td>
</tr>
<tr>
<td>X(14)</td>
<td>X(11)</td>
</tr>
<tr>
<td>X(3)</td>
<td>X(12)</td>
</tr>
<tr>
<td>X(7)</td>
<td>X(13)</td>
</tr>
<tr>
<td>X(11)</td>
<td>X(14)</td>
</tr>
<tr>
<td>X(15)</td>
<td>X(15)</td>
</tr>
</tbody>
</table>
FILTER IMPLEMENTATION

BUFFER A:
INPUT : x(n)
OUTPUT: X(k)

BUFFER B:
INPUT : x(n)
OUTPUT: X(k)

INPUT/OUTPUT
DATA
CHANNELS

Figure 5.3.12 Input/Output Double Buffering

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th>MAXIMUM</th>
<th>MINIMUM</th>
<th>MINIMUM (ALLOWED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.2795</td>
<td>2.2185x10^9</td>
<td>1.9523x10^5</td>
</tr>
<tr>
<td>3</td>
<td>3.1950</td>
<td>1.4192x10^-6</td>
<td>4.8752x10^-5</td>
</tr>
<tr>
<td>4</td>
<td>4.5764</td>
<td>4.7474x10^-7</td>
<td>6.9831x10^-5</td>
</tr>
<tr>
<td>5</td>
<td>6.9339</td>
<td>4.9518x10^-4</td>
<td>1.0580x10^-4</td>
</tr>
<tr>
<td>6</td>
<td>4.8041</td>
<td>1.0719x10^-6</td>
<td>7.3304x10^-4</td>
</tr>
<tr>
<td>7</td>
<td>5.7399</td>
<td>6.6319x10^-7</td>
<td>8.7584x10^-5</td>
</tr>
<tr>
<td>8</td>
<td>6.9480</td>
<td>1.4034x10^-7</td>
<td>1.0602x10^-4</td>
</tr>
</tbody>
</table>

Table 5.3.2 Range of Numbers for Cepstrum algorithm when running the Algorithm with full precision

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th>MAXIMUM</th>
<th>MINIMUM</th>
<th>MINIMUM (ALLOWED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.5374</td>
<td>0.0013</td>
<td>3.8721x10^-5</td>
</tr>
<tr>
<td>3</td>
<td>2.9607</td>
<td>1.4259x10^-4</td>
<td>4.5177x10^-3</td>
</tr>
<tr>
<td>4</td>
<td>4.3573</td>
<td>1.0378x10^-4</td>
<td>6.6487x10^-5</td>
</tr>
<tr>
<td>5</td>
<td>6.9012</td>
<td>1.5536x10^-4</td>
<td>1.0530x10^-4</td>
</tr>
<tr>
<td>6</td>
<td>4.7726</td>
<td>5.5195x10^-4</td>
<td>7.2824x10^-4</td>
</tr>
<tr>
<td>7</td>
<td>5.7242</td>
<td>1.6401x10^-4</td>
<td>8.7344x10^-4</td>
</tr>
<tr>
<td>8</td>
<td>6.8992</td>
<td>1.3800x10^-4</td>
<td>1.0527x10^-4</td>
</tr>
</tbody>
</table>

Table 5.3.3 Range of Numbers (after scaling) for Cepstrum algorithm when using 16-bit fixed point arithmetic
### FILTER IMPLEMENTATION

#### Table 5.4.33 Number of Arithmetic Operations, required by the Cepstrum Algorithm, over Channels 1-4

<table>
<thead>
<tr>
<th>Operation</th>
<th>CHANNELS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4085</td>
<td>4085</td>
<td>4201</td>
<td>4230</td>
<td>4646</td>
<td>4288</td>
<td>10599</td>
<td>4230</td>
<td></td>
</tr>
<tr>
<td>Multiplication</td>
<td>2294</td>
<td>2294</td>
<td>2414</td>
<td>2444</td>
<td>2874</td>
<td>2504</td>
<td>6978</td>
<td>2444</td>
<td></td>
</tr>
<tr>
<td>Division</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>50</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>10288</td>
<td>10288</td>
<td>11240</td>
<td>11478</td>
<td>14890</td>
<td>11954</td>
<td>30086</td>
<td>11478</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 5.4.34 Number of Arithmetic Operations, required by the Cepstrum Algorithm, over Channels 5-8

<table>
<thead>
<tr>
<th>Operation</th>
<th>CHANNELS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Addition &amp; Subtraction</td>
<td>4646</td>
<td>4288</td>
<td>10599</td>
<td>4230</td>
<td>2874</td>
<td>2504</td>
<td>6978</td>
<td>2444</td>
<td></td>
</tr>
<tr>
<td>Multiplication</td>
<td>40</td>
<td>30</td>
<td>50</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>50</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>14890</td>
<td>11954</td>
<td>30086</td>
<td>11478</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 5.4.35 Calculated Speed (ms) of the Cepstrum algorithm for TMS320C25 Implementation

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BEST</td>
<td>1.5339</td>
<td>1.7003</td>
<td>1.1240</td>
<td>1.1478</td>
<td>1.4890</td>
<td>1.1954</td>
<td>3.0086</td>
<td>1.1478</td>
</tr>
</tbody>
</table>

#### Table 5.4.36 Actual Time taken by Cepstrum on the TMS320C25 (Channels 1-4)

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BEST</td>
<td>1.5339</td>
<td>1.7003</td>
<td>1.1240</td>
<td>1.1478</td>
<td>1.4890</td>
<td>1.1954</td>
<td>3.0086</td>
<td>1.1478</td>
</tr>
</tbody>
</table>

#### Table 5.4.37 Actual Time taken by Cepstrum on the TMS320C25(Channels 5-8)

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th></th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BEST</td>
<td>1.5525</td>
<td>1.7189</td>
<td>1.5367</td>
<td>1.7031</td>
</tr>
</tbody>
</table>

---

**TMS320C25 IMPLEMENTATION**
FILTER IMPLEMENTATION

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th>BEST</th>
<th>WORST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1899</td>
<td>1.3563</td>
</tr>
<tr>
<td>2</td>
<td>1.1899</td>
<td>1.3563</td>
</tr>
<tr>
<td>3</td>
<td>1.1915</td>
<td>1.3579</td>
</tr>
<tr>
<td>4</td>
<td>1.1919</td>
<td>1.3583</td>
</tr>
</tbody>
</table>

Table 5.4.38 Actual Time taken by Cepstrum on the TMS320C30 (Channels 1-4)

<table>
<thead>
<tr>
<th>CHANNELS</th>
<th>BEST</th>
<th>WORST</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.2365</td>
<td>1.4029</td>
</tr>
<tr>
<td>6</td>
<td>1.1927</td>
<td>1.3591</td>
</tr>
<tr>
<td>7</td>
<td>2.2599</td>
<td>2.4263</td>
</tr>
<tr>
<td>8</td>
<td>1.1919</td>
<td>1.3583</td>
</tr>
</tbody>
</table>

Table 5.4.39 Actual Time taken by Cepstrum on the TMS320C30(Channels 5-8)

REFERENCES


6.1 Conclusions

The techniques investigated here, are concerned with the adaptive adjustment of the receiver in a digital data transmission system operating with additive noise and severe intersymbol interference in the received signal. The digital data receiver employs a linear predetection filter that can be used as part of the decision feedback equaliser or ahead of a near maximum likelihood detector. This predetection filter must be an all-pass network, such that the impulse response of the channel and filter is minimum phase. Many adaptive adjustment techniques have been studied [1-4], including ‘on line’ and ‘off-line’ systems. The former have been rejected on account of their low tolerance to the additive noise in the signal, this noise being effectively eliminated in the off-line system. All techniques developed so far [2-4], implement off-line systems and are basically root-finding algorithms that determine, in sequence, the roots (zeros) of the z-transform of the sampled impulse response of the channel, whose absolute values (moduli) is greater than one (or greater than a predetermined limit [2,3,5]. It then uses a knowledge of these roots to determine the tap gains of the linear feedforward transversal filter, and to form an estimate of the sampled impulse response of the channel and filter.

Extensive computer simulation tests of the existing root finding adjustment techniques proved satisfactory, for the 9600 bit/s QAM data-transmission system and over different telephone channels. However when the transmission of data increased to 19200 bit/s (a 64-point QAM signal constellation is now employed) the very same tests revealed a very different picture.
Since 19200 bit/s is over 80% of the Shannon limit, it is inevitable that the channels are now much more difficult to handle, and that conventional techniques are unlikely to operate satisfactorily. Indeed the number of roots of the sampled impulse response has now increased from 8 (which was the worst case considered for the 9600 bit/s) to 13 (worst case for the 19200 bit/s system). Furthermore, there are more roots lying close to the unit circle. However, even when modifying the starting-up procedure as suggested in Ref. 2, as a means to improving the root finding algorithm, the algorithm still fails as indicated in Chapter 4. The same result has been obtained in Ref. 4. This was to be expected, since the Clark-Hau algorithm is basically the Newton-Raphson algorithm (with a different convergence-stopping criterion and modified to provide the prefilter coefficients by means of an appropriate adjustment scheme). All root finding algorithms, are very much dependent on the initial starting points and a great many must be tried before a satisfactory result (suitable set of starting points) can be obtained. This also presupposes prior knowledge of the channel (which might not be available as is the case for the time-varying HF channels). Newton-Raphson has been tested by the author over the same telephone channels and exhibits similar behaviour to the Clark-Hau algorithm. It misses roots as indicated in Chapter 4 (in particular Tables NR.4 and 8). This could be acceptable in the case of a near-maximum likelihood detector where it is in fact only necessary to process those roots that lie further away from the unit circle [6-7]. However in the case of the widely used adaptive non-linear equaliser (see Appendix A2-Section A2.2), it is obvious that as many as possible of the roots of the sampled impulse response should be processed by the system, since otherwise there will be a needless reduction in tolerance to additive noise [3]. Bearing this in mind other root finding algorithms, that are known to possess better convergence properties
CONCLUSIONS

(such as Laguerre method) were tested by the author over a platform of eight telephone channels representing a wide range of operating conditions. However occasional failure led to an imaginative and novel way of applying both Laguerre and Newton methods (Chapter 3 - Algorithms 3.2.2.ii and 3.2.3.ii) over the same telephone channels, with impressive results (Tables L.1,3-8,11-13 and NRm.1-2,4-5,7-8,10-11,13-14), with the modified Newton method failing only over two of the most distorted channels (Channels 6 and 7 - Tables NRm.16 and 19). The modified Laguerre method operated successfully over all eight Channels. The results obtained using these modified root finding methods (and Laguerre in particular) are definitely much better (in terms of accuracy and speed) than those obtained when using other root finding methods, such as the Clark-Hau method.

Failure to locate roots (in the case of modified Newton method), their speed (number of arithmetic operations required by both Newton and Laguerre), and implementation considerations together with the failure of both these methods to operate with limited precision over the most severely distorted Channels (see Chapter 5 -Tables 5.4.10-13) makes perfectly clear that no root finding method can satisfactorily adjust the prefilter over Channels 4-8 (A most significant result obtained in Chapter 5). This forced the author to seek methods where there is no need of locating roots and therefore dropping the requirement of prior channel knowledge. This could be particularly good news for the HF channels with the added advantage of reducing the operational time for the adjustment of the prefilter, thus in turn making the adjustment of the prefilter possible within fewer sampling intervals, situation particularly welcome for a time varying environment. A suit of Spectral factorisation techniques has been investigated here and the Cepstrum method proved to behave extremely satisfactorily over all eight telephone channels adjusting the prefilter in an extremely fast and accurate manner (see relevant tables in Chapters 4 and 5).

The algorithm here is used with a complex input signal and has been modified to provide a set of coefficients necessary for the adjustment of the prefilter. This modification consists of including one of the considered adjustment schemes studied in Chapter 3 and tested in Chapter 4.
A novel algorithm (NIT) that exhibits very similar behaviour to that of the Cepstrum has also been designed by the author and tested over both telephone channels and HF channels. These algorithms behave extremely well as proved by the Simulation results in Chapter 4. The algorithms have been also tested with limited precision arithmetic in Chapter 5.

The following very important results were derived from this thesis:

1. The Clark-Hau method (and variants [4]) can adjust successfully the first four telephone channels (9600 bit/s) but cannot adjust the next four (19200), even when using improved start up procedure and previously located roots. Also the computational overhead associated with this algorithm makes it impossible to adjust the prefilter fast enough for it to be effective in a time varying environment. It has been estimated that the algorithm can process four roots (and update the prefilter taps) approximately every four sampling periods [8], where the algorithm has been implemented on a TMS320C25.

Since the number of roots for even the milder HF channels can be as large as 12 (or more), during transmission, a root finding algorithm, such as the Clark-Hau algorithm can adjust the prefilter every 12 sampling periods, which could lead to an unacceptable degradation in the performance of the system as a whole. Therefore, no root finding algorithm can be used to adjust the prefilter over the HF channels considered here (even the milder ones for a 9600 bit/s system).

Finally, the algorithm is not suitable for operation with near maximum likelihood detectors, for the case of HF channels because it processes only the roots with magnitude greater than 1.05.

* The Clark-Hau algorithm is chosen here, as an example of a typical root finding algorithm.
This limit has been set \([3,5]\) clearly in an attempt to reduce the number of processed roots (and therefore the number of iterations) and also to reduce the number of taps required by the adaptive filter. It has been considered high enough to avoid any significant degradation in the performance of the detector. However, in the case of the HF channels, frequently situations arose where there was a number of roots with magnitudes smaller than \(1.05\) (say around 1.03) and this number (of roots) was big enough to cause the sampled impulse response to deviate significantly from the ideal minimum phase, after the algorithm had operated on it (leaving these roots unchanged since their magnitude was less than the pre-set limit of 1.05). The resulting 'pseudominimum' sample impulse response deviates enough from the ideal (see for example, Graph GNT.8 in Chapter 4) to defeat the purpose of setting this limit in the first place, i.e. simplifies the adaptive system but increases the complexity of the detector (which is forced now to operate on a non minimum phase waveform). All the above apply to the Laguerre algorithm (a slightly more accurate but somewhat slower than the Clark-Hau algorithm), and to any other root finding algorithm presented in this thesis. Therefore, no root finding algorithm can be used to adjust the prefilter over the HF channels considered here (even the milder ones for a 9600 bit/s system).

2. Since the accurate adjustment of the prefilter (given correct estimates) is the single most limiting factor in the transmission of data over the more severely distorted channels and at the higher transmission rates \([3]\), the importance of an accurate, robust and fast adjustment technique cannot be underestimated. The Cepstrum algorithm operates satisfactorily over telephone channels and HF radio links even under extremely unfavourable conditions (low SNR 16 bits). When implemented on either a TMS320C25 or 30 even with 16 bits can adjust the prefilter in a satisfactory manner (as shown in the relevant Tables in Chapters 4 and 5).
3. The N I T algorithm follows closely, however it can run fast enough only on a TMS320C30 for the HF channels. The case of 10 iterations has been assumed although sometimes a smaller number of iterations is needed.

4. Perhaps the single most important result is that by employing the Cepstrum algorithm, adjustment of the predetection filter is now possible (except when the estimator fails [9]), over even the most severely distorted HF channels considered here.

6.2 Suggestions for further work

1. A complete modem receiver, employing the Cepstrum technique for the adjustment of the prefilter, can be simulated in Matlab with limited precision arithmetic using one of the recently available Matlab tools for dynamic system simulation with limited precision. The resulting code can then be optimised for Implementation using TMS320C25/30 processors.

2. More research can be directed towards finding an effective stopping criterion for the modified N I T algorithm (N I T.a and N I T.b in Chapter 3), since this would result in an extremely fast (even faster than the basic N I T method simulated in the previous two Chapters) and accurate adjustment technique, particularly well suited for HF applications.

3. Finally, given the need for high speed and accuracy required on land mobile applications, and considering the lack of nonproprietary solutions, fast enough to satisfy the GSM requirements [10], where adaptive equalisation is concerned, some research into the usefulness of the discussed adjustment techniques, would in the author’s view, yield particularly good results. In fact, the author has already given considerable thought to the above suggested points (2 and 3 in particular).
References


## Contents

### Volume 2

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Illustrations</td>
<td>iii</td>
</tr>
<tr>
<td>Notation</td>
<td>vi</td>
</tr>
<tr>
<td>Appendices</td>
<td>289</td>
</tr>
</tbody>
</table>

#### Appendices

**Appendix 2 (A2)**  
A2.1 Differential Encoding and the Mapping of the uncoded 16 (or 64) level QAM signal | 289
A2.2 Equalisers | 290

**References** | 299

**Appendix 3 (A3)**  
A3. Clark-Hau equivalence to Newton-Raphson | 301
A3.1 The Levinson Recursion | 302
A3.2 Durbin's Algorithm | 309
A3.3 Whittle's Exponential-Logarithmic Method | 315
A3.4 Proof of the Fejer-Riesz theorem | 317
A3.4.1 Hilbert Transform Relations | 320
A3.5 LU Factorisation Decomposition | 321
A3.6 Extended Newton's Method for the complex case | 322
A3.6.1 Real coefficients-complex roots | 323
CONTENTS

A3.6.2 Complex coefficients-complex roots .......................... 326
A3.7 Schur Decomposition...................................................... 328
A3.8 Companion Matrix.......................................................... 329

References.................................................................................. 330

APPENDIX 4 (A4).............................................................................. 331
A4.1 Attenuation and Delay Distortions over Telephone Circuits... 331
A4.2 Derivation of the Sampled Impulse Responses From the
Attenuation and Group-Delay Characteristics.............................. 332
A4.3 The Equipment Filters.......................................................... 334
A4.4 HF radio Links................................................................. 339
A4.4.1 Ionospheric Radio Propagation......................................... 339
A4.4.2 Theory of Ionospheric Radio Propagation.............................. 340
A4.4.3 Distortion on HF Channels............................................... 341
A4.4.4 Signal Statistics.............................................................. 345
A4.4.5 HF Channel Classification................................................. 350
A4.5 Model and Simulation of the HF channel.............................. 356
A4.5.1 Model of the HF Channel.................................................. 356
A4.5.2 Data Transmission over a Model of an HF Channel using QAM 359
A4.5.3 Model of the HF Data Transmission System used in Tests... 364
A4.6.1 QRN method................................................................. 376
A4.6.2 QRL method................................................................. 378

References.................................................................................. 382

APPENDIX 5 (A5).............................................................................. 385
A5 Fixed-Point Arithmetic............................................................. 385
A5.1 Polynomial Approximation.................................................. 386
A5.2 Approximation of the Inverse............................................... 386
A5.3 N.I.T (with Limited Precision Arithmetic).............................. 389
A5.4 N.I.T implementation on a TMS320C25-30............................ 396
CONTENTS

Figures

2.2.1 Model of Digital Data Transmission System ........................................ 47
2.2.2a Amplitude Response of Filter A .................................................. 48
2.2.2b Frequency Spectrum of z(t) ....................................................... 48
2.2.2c Amplitude Response of filter C .................................................. 48
2.2.3 Equivalent Baseband Model of Data Transmission System .............. 49
2.2.4 Simplified Baseband Model of Data Transmission System used in the Computer Simulation .................................................. 50
2.5.1 Model of Data Transmission System with 'Desired' Impulse Response .... 50

3.1 The QR Method..................................................................................... 95
3.2 One Tap Feedback Filter................................................................. 96
3.3 Two Tap Feedforward Filter............................................................ 96
3.4 Adjustment of the Receiver Prefilter.................................................. 97
3.5 Flow Diagram of the Clark-Hau Algorithm......................................... 98
3.6 Least-error energy filtering............................................................... 99
3.7 Predetection Filter Implementation employing Gram-Schmidt Orthogonalisation ................................................................. 99
3.8 Matrix Y for the Gram-Schmidt Method............................................. 100
3.9 The N.I.T algorithm........................................................................... 101
3.10 Cepstrum Algorithm........................................................................ 102

4.4.1.a Attenuation and Group Delay Characteristics of Telephone Circuit 1........ 226
4.4.1.b Group Delay Characteristics.................................................... 226
4.4.2.a Attenuation and Group Delay Characteristics of Telephone Circuit 2........ 227
4.4.2.b Group Delay Characteristics.................................................... 227
4.4.3.a Attenuation and Group Delay Characteristics of Telephone Circuit 3........ 228
4.4.3.b Group Delay Characteristics.................................................... 228
4.4.4.a Attenuation and Group Delay Characteristics of Telephone Circuit 4........ 229
4.4.4.b Group Delay Characteristics.................................................... 229

5.2.1a Floating-point number.................................................................. 269
5.2.1b Fixed-point (integer) number....................................................... 269
5.2.1c Scaled fixed-point number.......................................................... 269
5.3.1a Butterfly for decimation-in-time FFT .......................................... 269
5.3.1b Alternative form of Butterfly..................................................... 270
5.3.2 Statistical model for fixed-point roundoff noise in a decimation-in-time butterfly computation .................................................. 270
5.3.3 Decimation-in-time FFT algorithm................................................ 271
5.3.4 Functional block diagram of a Sequential FFT processor ............... 272
5.3.5 The cascade FFT processor.......................................................... 272
5.3.6 The parallel Iterative Processor..................................................... 273
5.3.7 The Array Processor...................................................................... 273
5.3.8 Minimal Processing System with external data RAM and PROM/EPROM.... 274
5.3.9 Butterfly for radix-4, Decimation-in-time FFT................................. 275
5.3.10 A 16-point, radix-4, Decimation-in-time FFT............................... 276
5.3.12 Input/Output Buffering............................................................... 278

A2.1 Block Diagram of the Differential Encoder........................................... 295
**CONTENTS**

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN.7 Performance of Newton-Raphson over Channel 7</td>
</tr>
<tr>
<td>GRN.8 Performance of Newton-Raphson over Channel 8</td>
</tr>
<tr>
<td>GL.1a Performance of Laguerre method over Channel 1</td>
</tr>
<tr>
<td>GL.1b Performance of Laguerre method over Channel 1</td>
</tr>
<tr>
<td>GL.2 Performance of Laguerre method over Channel 2</td>
</tr>
<tr>
<td>GL.3 Performance of Laguerre method over Channel 3</td>
</tr>
<tr>
<td>GL.4 Performance of Laguerre method over Channel 4</td>
</tr>
<tr>
<td>GL.5 Performance of Laguerre method over Channel 5</td>
</tr>
<tr>
<td>GL.6 Performance of Laguerre method over Channel 6</td>
</tr>
<tr>
<td>GL.7 Performance of Laguerre method over Channel 7</td>
</tr>
<tr>
<td>GL.8 Performance of Laguerre method over Channel 8</td>
</tr>
<tr>
<td>G.NT1 Performance of Cepstrum and N.I.T algorithms over HF radio links (HF2)</td>
</tr>
<tr>
<td>G.NT2 Performance of Cepstrum and N.I.T algorithms over HF radio links (HF5)</td>
</tr>
<tr>
<td>G.NT3 ‘Raw’ Impulse Response, Minimum Phase and Pseudominimum Phase Impulse Responses</td>
</tr>
<tr>
<td>G.NT4 Performance of Cepstrum and N.I.T algorithms over HF radio links (HF5)</td>
</tr>
<tr>
<td>G.NT5 Performance of Root Finding (any), Cepstrum and N.I.T algorithms over HF radio links (HF4)</td>
</tr>
<tr>
<td>G.NT6 Performance of Root finding Methods (any) over an HF radio link (HF4)</td>
</tr>
<tr>
<td>G.NT7 Performance of Cepstrum and N.I.T Methods over HF radio links (HF4)</td>
</tr>
<tr>
<td>G.NT8 Performance of Root finding Methods (any) over an HF radio link (HF4)</td>
</tr>
<tr>
<td>G.NT9 Performance of Cepstrum and N.I.T Methods over HF radio links (HF4)</td>
</tr>
<tr>
<td>G.1 Performance of Selected Adjustment Techniques over Channel 1</td>
</tr>
<tr>
<td>G.2 Performance of Selected Adjustment Techniques over Channel 2</td>
</tr>
<tr>
<td>G.3 Performance of Selected Adjustment Algorithms over Channel 3</td>
</tr>
<tr>
<td>G.4 Performance of Selected Adjustment Algorithms over Channel 4</td>
</tr>
<tr>
<td>G.5 Performance of Selected Adjustment Algorithms over Channel 5</td>
</tr>
<tr>
<td>G.6 Performance of Selected Adjustment Algorithms over Channel 6</td>
</tr>
<tr>
<td>G.7 Performance of Selected Adjustment Algorithms over Channel 7</td>
</tr>
<tr>
<td>G.8 Performance of Selected Adjustment Algorithms over Channel 8</td>
</tr>
</tbody>
</table>

**NOTATION**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_i$</td>
<td>Data Symbol (Chapter 2) (see p. 13)</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of Levels for the QAM signal (Chapter 2) (see p. 13)</td>
</tr>
<tr>
<td>$a(t)$</td>
<td>Impulse response of a filter (Chapter 2) (see p. 14)</td>
</tr>
<tr>
<td>$A(f)$</td>
<td>Frequency response of a filter (Chapter 2) (see p. 14)</td>
</tr>
<tr>
<td>$b(t)$</td>
<td>Impulse response of a filter (Chapter 2) (see p. 14)</td>
</tr>
<tr>
<td>$B(f)$</td>
<td>Frequency response of bandpass filter (Chapter 2) (see p. 14)</td>
</tr>
<tr>
<td>$w(t)$</td>
<td>White Gaussian noise with zero mean and two-sided power spectral density $n_0/2$ (Chapter 2) (see p. 15)</td>
</tr>
<tr>
<td>$n_0/2$</td>
<td>Power spectral density of $w(t)$</td>
</tr>
<tr>
<td>$j$</td>
<td>When used as subscript is an integer, otherwise it is $\sqrt{-1}$ (Chapter 2) (see p. 15)</td>
</tr>
<tr>
<td>$*$</td>
<td>Convolution operator (Chapter 2) (see p. 16), or as a superscript denotes</td>
</tr>
</tbody>
</table>
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2.2</td>
<td>Signal Constellation for a 16-Level Differentially Encoded QAM signal</td>
<td>296</td>
</tr>
<tr>
<td>A2.3</td>
<td>Signal Constellation for a 16-Level Differentially Encoded QAM signal</td>
<td>297</td>
</tr>
<tr>
<td>A2.2.1</td>
<td>Linear Feedforward Transversal Equaliser</td>
<td>298</td>
</tr>
<tr>
<td>A2.2.2</td>
<td>Pure non-linear Equaliser and Det&lt;sub&gt;x&lt;/sub&gt;</td>
<td>298</td>
</tr>
<tr>
<td>A4.1</td>
<td>Typical attenuation-frequency characteristics</td>
<td>335</td>
</tr>
<tr>
<td>A4.2</td>
<td>Attenuation-frequency characteristics of a poor circuit</td>
<td>335</td>
</tr>
<tr>
<td>A4.3</td>
<td>Typical group-delay frequency characteristics</td>
<td>336</td>
</tr>
<tr>
<td>A4.4</td>
<td>Group-delay characteristics of a poor circuit</td>
<td>336</td>
</tr>
<tr>
<td>A4.3.1</td>
<td>Attenuation and Group Delay Characteristics of filter set 1</td>
<td>337</td>
</tr>
<tr>
<td>A4.3.2</td>
<td>Attenuation and Group Delay Characteristics of filter set 2</td>
<td>338</td>
</tr>
<tr>
<td>A4.4.1</td>
<td>Single Hop Propagation</td>
<td>351</td>
</tr>
<tr>
<td>A4.4.2</td>
<td>Refractive Bending</td>
<td>351</td>
</tr>
<tr>
<td>A4.4.2a</td>
<td>Reflection from different Layers</td>
<td>352</td>
</tr>
<tr>
<td>A4.4.2b</td>
<td>Different Number of Hops</td>
<td>352</td>
</tr>
<tr>
<td>A4.4.3</td>
<td>Typical Response of a Multipath Channel</td>
<td>353</td>
</tr>
<tr>
<td>A4.4.4</td>
<td>Rayleigh Probability Density Function</td>
<td>354</td>
</tr>
<tr>
<td>A4.4.5</td>
<td>Block Diagram of HF Ionospheric Channel Model</td>
<td>367</td>
</tr>
<tr>
<td>A4.4.7</td>
<td>Rayleigh Fading introduced by one skywave</td>
<td>367</td>
</tr>
<tr>
<td>A4.4.8</td>
<td>Power Spectrum of q&lt;sub&gt;t&lt;/sub&gt;(t) and q&lt;sub&gt;r&lt;/sub&gt;(t)</td>
<td>368</td>
</tr>
<tr>
<td>A4.4.9</td>
<td>Characteristics of the Signals q&lt;sub&gt;t&lt;/sub&gt;(t)</td>
<td>369</td>
</tr>
<tr>
<td>A4.4.10</td>
<td>Block Diagram of the filter used for generation of Fading Samples</td>
<td>370</td>
</tr>
<tr>
<td>A4.4.11</td>
<td>The linear Interpolation process</td>
<td>371</td>
</tr>
<tr>
<td>A4.4.12</td>
<td>Model of the Digital Data Transmission System</td>
<td>372</td>
</tr>
<tr>
<td>A4.4.13</td>
<td>Model of a two Skywave HF Radio link</td>
<td>373</td>
</tr>
<tr>
<td>A4.4.14</td>
<td>Relationship between {Y&lt;sub&gt;i&lt;/sub&gt;} and {Z&lt;sub&gt;i&lt;/sub&gt;}</td>
<td>374</td>
</tr>
<tr>
<td>A5.1a</td>
<td>Number wheel for unsigned Integers</td>
<td>388</td>
</tr>
<tr>
<td>A5.1b</td>
<td>Number wheel for signed Integers</td>
<td>388</td>
</tr>
<tr>
<td>A5.1c</td>
<td>Number wheel for Fixed-Point Representation</td>
<td>388</td>
</tr>
</tbody>
</table>

## Graphs

<table>
<thead>
<tr>
<th>Graph</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRO.1</td>
<td>Root Positions of Channel 1</td>
<td>141</td>
</tr>
<tr>
<td>GRO.2</td>
<td>Root Positions of Channel 2</td>
<td>142</td>
</tr>
<tr>
<td>GRO.3</td>
<td>Root Positions of Channel 3</td>
<td>143</td>
</tr>
<tr>
<td>GRO.4</td>
<td>Root Positions of Channel 4</td>
<td>144</td>
</tr>
<tr>
<td>GRO.5</td>
<td>Root Positions of Channel 5</td>
<td>145</td>
</tr>
<tr>
<td>GRO.6</td>
<td>Root Positions of Channel 6</td>
<td>146</td>
</tr>
<tr>
<td>GRO.7</td>
<td>Root Positions of Channel 7</td>
<td>147</td>
</tr>
<tr>
<td>GRO.8</td>
<td>Root Positions of Channel 8</td>
<td>148</td>
</tr>
<tr>
<td>GNR.1</td>
<td>Performance of Newton-Raphson over Channel 1</td>
<td>149</td>
</tr>
<tr>
<td>GNR.2</td>
<td>Performance of Newton-Raphson over Channel 2</td>
<td>149</td>
</tr>
<tr>
<td>GRN.3</td>
<td>Performance of Newton-Raphson over Channel 3</td>
<td>149</td>
</tr>
<tr>
<td>GRN.4a</td>
<td>Performance of Newton-Raphson over Channel 4</td>
<td>149</td>
</tr>
<tr>
<td>GRN.4b</td>
<td>Performance of Newton-Raphson over Channel 4</td>
<td>150</td>
</tr>
<tr>
<td>GRN.5</td>
<td>Performance of Newton-Raphson over Channel 5</td>
<td>150</td>
</tr>
<tr>
<td>GRN.6</td>
<td>Performance of Newton-Raphson over Channel 6</td>
<td>150</td>
</tr>
</tbody>
</table>
GRN.7  Performance of Newton-Raphson over Channel 7 ........................................... 150
GRN.8  Performance of Newton-Raphson over Channel 8 ........................................... 151
GL.1a  Performance of Laguerre method over Channel 1 ........................................... 152
GL.1b  Performance of Laguerre method over Channel 1 ........................................... 152
GL.2  Performance of Laguerre method over Channel 2 ........................................... 152
GL.3  Performance of Laguerre method over Channel 3 ........................................... 152
GL.4  Performance of Laguerre method over Channel 4 ........................................... 153
GL.5  Performance of Laguerre method over Channel 5 ........................................... 153
GL.6  Performance of Laguerre method over Channel 6 ........................................... 153
GL.7  Performance of Laguerre method over Channel 7 ........................................... 153
GL.8  Performance of Laguerre method over Channel 8 ........................................... 154
G.NT1 Performance of Cepstrum and N.I.T algorithms over HF radio links (HF2) .... 213
G.NT2 Performance of Cepstrum and N.I.T algorithms over HF radio links (HF5) [1st Transmission instant] .............................................................................. 213
G.NT3 'Raw' Impulse Response, Minimum Phase and Pseudominimum Phase Impulse Responses ................................................................................... 214
G.NT4 Performance of Cepstrum and N.I.T algorithms over HF radio links (HF5) [2nd Transmission instant] .............................................................................. 214
G.NT5 Performance of Root Finding (any), Cepstrum and N.I.T algorithms over HF radio links (HF4) .............................................................................. 215
G.NT6 Performance of Root finding Methods (any) over an HF radio link (HF4) ...... 215
G.NT7 Performance of Cepstrum and N.I.T Methods over HF radio links (HF4) .... 216
G.NT8 Performance of Root finding Methods (any) over an HF radio link (HF4) .... 216
G.NT9 Performance of Cepstrum and N.I.T Methods over HF radio links (HF4) .... 217
G.1  Performance of Selected Adjustment Techniques over Channel 1 .................. 218
G.2  Performance of Selected Adjustment Techniques over Channel 2 .................. 219
G.3  Performance of Selected Adjustment Algorithms over Channel 3 ............... 220
G.4  Performance of Selected Adjustment Algorithms over Channel 4 ............... 221
G.5  Performance of Selected Adjustment Algorithms over Channel 5 ............... 222
G.6  Performance of Selected Adjustment Algorithms over Channel 6 ............... 223
G.7  Performance of Selected Adjustment Algorithms over Channel 7 ............... 224
G.8  Performance of Selected Adjustment Algorithms over Channel 8 ............... 225

NOTATION

\( s_i \) = Data Symbol (Chapter 2) (see p. 13)
\( m \) = Number of Levels for the QAM signal (Chapter 2) (see p. 13)
\( a(t) \) = Impulse response of a filter (Chapter 2) (see p. 14)
\( A(f) \) = Frequency response of a filter (Chapter 2) (see p. 14)
\( b(t) \) = Impulse response of a filter (Chapter 2) (see p. 14)
\( B(f) \) = Frequency response of bandpass filter (Chapter 2) (see p. 14)
\( w(t) \) = White Gaussian noise with zero mean and two-sided power spectral density \( n_J2 \) (Chapter 2) (see p. 15)
\( n_{j/2} \) = Power spectral density of \( w(t) \)
\( j \) = When used as subscript is an integer, otherwise it is \( \sqrt{-1} \) (Chapter 2) (see p. 15)
\( * \) = Convolution operator (Chapter 2) (see p. 16), or as a superscript denotes
CONTENTS

Complex Conjugate (Chapter 2) (see p. 19)

$\mathbf{r}(t) = \mathbf{y}(t)$ Received signal (Chapter 2) (see p. 17)

Impulse response of linear baseband channel (Chapter 2) (see p. 19)

Sampled impulse response of linear baseband channel at time $t = iT$ (Chapter 2) (see p. 22)

$\mathbf{s}_i = \mathbf{E} [ \mathbf{e} ]$ Detected Data Symbol (Chapter 2) (see p. 23, 24)

$\mathbf{E} [ \mathbf{.} ]$ Expectation operator (Chapter 2) (see p. 23)

$e_i$ Error in the estimated value of the received signal $r_i$ (Chapter 2)

(see p. 25)

$\mathbf{r}_i$ Complex valued sample of $\mathbf{r}(t)$ at time $t = iT$ (Chapter 2) (see p. 25)

$\mathbf{Y}_i \mathbf{= \text{ Estimate}} \text{ of} \mathbf{Y}_i \text{ at time } t = iT \text{ (Chapter 2) (see p. 25)}$

Error measurement given by equation 2.7.9 (Chapter 2) (see p. 45)

Error measurement given by equation 2.7.10 (Chapter 2) (see p. 45)

Error measurement given by equation 2.7.11 (Chapter 2) (see p. 45)

Error measurement given by equation 2.7.11 (Chapter 2) (see p. 45)

Householder Matrix (Chapter 3) (see p. 56)

Householder Matrix (Chapter 3) (see p. 57)

Matrix norm (Chapter 3) (see p. 57)

$\mathbf{D}(z)$ $z$-transform of linear pre-detection filter (Chapter 3) (see p. 68)

$\lambda_4$ An estimate of the negative reciprocal of a root of $Y(z)$ that lie outside the unit circle (Chapter 3) (see p. 71)

$\delta$ Threshold level for determining convergence of an iterative process (Chapter 3) (see p. 59)

$\theta$ Threshold level for determining convergence of an iterative process (Chapter 3) (see p. 73)

$\psi$ Represents both $\psi_2$ and $\psi_3$ in the NI T method (Chapter 4) (see p. 116)

$\varepsilon$ Threshold level for determining convergence of an iterative process (Chapter 4) (see p. 117)

$\mathbf{e}_\text{ps}$ Fractional roundoff error in Laguerre method (Chapter 4) (see p. 121)

$\mathbf{R}_\text{e} [ \mathbf{.} ]$ Real part of a complex number (Chapter 5) (see p. 237)

$\mathbf{I} \text{m} [ \mathbf{.} ]$ Imaginary part of a complex number (Chapter 5) (p. 237)
VOLUME 2
APPENDIX 2 [A2]

A2.1 DIFFERENTIAL ENCODING AND THE MAPPING OF THE UNCODED 16 (OR 64) LEVEL QAM SIGNAL

At the transmitter (Chapter 2), the binary data stream is encoded using differential encoding. Differential encoding reduces the number of errors occurring in the detected data due to sudden phase changes in the signal carrier relative to the reference carriers at the receiver. Figure A2.1 shows a block diagram of the differential encoder. It is assumed that the uncoded (not convolutionally encoded) binary digits entering the encoder are statistically independent and equally likely to have the values 0 or 1. As the binary digits enter the encoder they are split up into groups of four or six digits, \( \{a_h\} \), for \( h=1,2,...,b \), where, depending on whether 16 or 64-level QAM is used, \( b=4 \) or \( 6 \), respectively. The first two binary digits \( a_{i,1} \) and \( a_{i,2} \) in any one group are now recoded to give the corresponding two binary digits \( \beta_{i,1} \) and \( \beta_{i,2} \) according to Table A2.1, whereas

\[
\beta_{i,3} = a_{i,3} \quad \text{A2.1}
\]

and

\[
\beta_{i,4} = a_{i,4} \quad \text{A2.2}
\]

where in Table A2.1, \( \beta_{i-1,1} \) and \( \beta_{i-1,2} \) are the two previously differentially encoded digits. The remaining binary digits, \( a_{i,3} \) to \( a_{i,b} \) are left as they are, giving the digits \( \beta_{i,3} \) to \( \beta_{i,b} \) at the encoder output. Initially, when the first group are encoded (when \( i=0 \)), \( \beta_{-1,1} \) and \( \beta_{-1,2} \) are fixed to some arbitrary value, say 0.

The resulting group of \( b \) digits now determine the corresponding coded data symbol \( s_i \). Figures A2.2 and A2.3 show the signal constellations of the coded symbols for a 16 or 64-level QAM signal, respectively, where the binary numbers below each point are the values of the encoded binary digits, \( \beta_{i,1}, \beta_{i,2},...,\beta_{i,b} \). The first two binary digits, \( \beta_{i,1}, \beta_{i,2},...,\beta_{i,b} \). The first two binary digits, \( \beta_{i,1} \) and \( \beta_{i,2} \) in any coded number determine which quadrant in figure A2.2 or A2.3 \( s_i \) lies in and the remaining digits,
\(\beta_{1,3}\) to \(\beta_{1,b}\) determine the position of \(s_i\) within a particular quadrant. To further help reduce the probability of errors occurring, due to noise, in the detected data, Gray coding has been used within each of the quadrants as can be seen in Figures A2.2 and A2.3. Exact Gray coding is not possible over the entire signal constellation.

At the receiver, after the data symbol \(s_i\) has been detected, the corresponding values of the encoded binary digits, \(\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,b}\), are found using Figure A2.2 or A2.3. The values of \(a_{i,1}\) and \(a_{i,2}\) are then determined using Table A2.1, where \(\beta_{i,1}\) and \(\beta_{i,2}\) are now the detected coded digits, and \(\beta_{i-1,1}\) and \(\beta_{i-1,2}\) are the previously detected coded digits.

It can be seen from Table A2.1 and Figures A2.2 or A2.3 that a phase shift of \(\pi/2\) radians (or any multiple) in the phase relationship between the reference carriers in the coherent demodulators and the received signal carrier, giving the corresponding rotation in the phase angle of a received sample \(r_i\), does not change the detected values of \(a_{i,3}\) to \(a_{i,b}\), corresponding to any given value of \(s_i\), nor can it lead to any prolonged burst of errors in the detected values of \(a_{i,1}\) and \(a_{i,2}\).

### A2.2 EQUALISERS

The equaliser structure is used to correct the linear distortion introduced into a digital signal when the latter is transmitted at a relatively high rate over a linear baseband channel [1-3]. Recently, transversal filters have emerged as a convenient and flexible type of equaliser. The transversal structure is the most widely studied linear equaliser [4-7].

The linear feedforward transversal filter in Figure A2.2.1 implements a linear equaliser.

The equaliser operates on the sample values \(r_i\), \(i=1\ldots m\), present at the time instant \(t=iT\). Each square marked T is here a storage element that holds the corresponding sample value \(r_i\). On the reception of a sample, the stored signals are shifted one place to the right. Thus, the extreme left-hand signal at the input to the equaliser is that received at time \(t=iT\), the next signal is that received at time \(t=(i-1)T\), and so on, each storage introducing a delay of \(T\) seconds. Each circle in Figure A2.2.1 represents a
multiplier \( d_i \), that multiplies the signal \( r_i \) by \( d_i \). Therefore, the output signal at time \( t = iT \), is the sum of the outputs from the multipliers at that instant, which is given by:

\[
x_i = \sum_{j=0}^{m} r_{i-j} d_j
\]

The sampled impulse response of the filter is given by the sequence of its tap gains \( \{d_i\} \),

\[
D = [d_0 \ d_1 \ldots d_m]
\]

which also form the coefficients in the corresponding z-transform

\[
D(z) = d_0 + d_1 z^{-1} + \cdots + d_m z^{-m}
\]

Let the samples-impulse response of the linear baseband channel (Chapter 2) be given by the \((g+1)\)-component row vector

\[
Y = [y_0 \ y_1 \ldots y_g]
\]

with the corresponding z-transform

\[
Y(z) = y_0 + y_1 z^{-1} + \cdots + y_g z^{-g}
\]

For achieving the accurate equalisation of the channel, the z-transform of the channel and equaliser is

\[
Y(z)D(z) \approx z^{-h}
\]
where $h$ is the non-negative integer in the range $0$ to $m+g$. In general the sampled-impulse response of the channel and the equaliser is given by

$$Y(z)D(z) = E(z)$$  \hspace{1cm} A2.2.7

where $E(z)$ is the $(m+g+1)$-component vector

$$E = [e_0 \ e_1 \cdots e_{m+g}]$$  \hspace{1cm} A2.2.8

From Equation A2.2.6 and for equalising the channel Equation A2.2.7 becomes

$$Y(z)D(z) = z^{-h}E_h$$  \hspace{1cm} A2.2.9

where $E_h$ is the ideal or desired sampled-impulse response of the equalised channel given by

$$E_h = [0 \ 0 \ 1 \ 0 \cdots 0]$$  \hspace{1cm} A2.2.10

Note that the vector $E_h$ in Equation A2.2.10 is the vector $E$ of Equation A2.2.8, where in the latter $e_h \approx 1$ and $e_i \approx 0$, for $i=0,1,...,m+g$ and $i \neq h$.

For the exact equalisation of the channel, $e_i = 0$. This condition is satisfied when the number of taps is infinite[2,8]. With a finite feedforward transversal filter Equation A2.2.10 will not in general be satisfied exactly. However as the length of the equaliser increases, intersymbol interference can be made arbitrarily small [9]. Thus an approximation to Equation A2.2.10 may be achieved.

Several design techniques are readily available for evaluating the equaliser coefficients $\{d_j\}$. Since the average probability of error is the most meaningful performance measure in a digital communication system, it is desirable to optimise the coefficients of the equaliser to minimise the probability of error [10]. However, the probability of error is a highly non-linear function of $\{d_j\}$. Therefore, alternative techniques have been developed to adjust the Equaliser coefficients. In the early work on linear
equalisers these were generally designed to minimise the peak distortion, where the peak distortion is defined to be [11,12]

\[ D_p = \frac{1}{|e_m|} \sum_{i=0}^{m+2} |e_i| \]  \hspace{1cm} A2.2.11

This, i.e. the minimisation of the peak distortion can be achieved, under certain conditions, by a technique known as zero forcing [4,5,8]. In the zero forcing technique, the equaliser coefficients are adjusted to set \( e_m \) to unity and the remaining \( \{e_i\} \) in Equation A4.2.8 to zero. Not only does the technique of zero forcing minimise the peak distortion in the equalised signal,

\[ \frac{1}{|y_j|} \sum_{i=0}^{n} |y_i| \]  \hspace{1cm} A2.2.12

when less than unity, but it also leads to some simple iterative processes for holding the equaliser correctly adjusted for a slowly time-varying channel [13,14].

The weakness of this technique is that it cannot be used in its simple form for equalising severe time-varying signal distortion. The second technique, i.e. the minimum mean-square error technique (MMSE), minimises the mean-square difference between the actual and the desired sample values at the equaliser output, for a given number of taps, when a continuous stream of data elements is being received in the presence of noise. It is now recognised that a linear transversal equaliser that minimises the mean square error at its output signal generally gives a more useful or effective degree of equalisation, for a given number of taps, than an equaliser that minimises the peak distortion [11,15]. It can be shown [16] that a linear MMSE equaliser also minimises the mean-square distortion in the equalised signal given by

\[ D_m = \frac{1}{\sum_{i=0}^{m+2} e_i^2} \]  \hspace{1cm} A2.2.13
The former equaliser is in fact a special case of the latter. Further details on these two techniques and other techniques for adjusting the tap gains of the linear equaliser are given elsewhere [2,17]. When a channel introduces pure phase distortion, it can be shown [16] that the linear feedforward transversal equaliser as shown in Figure A4.2.1 gives the best tolerance to additive white Gaussian noise. This equaliser when correctly adjusted for pure phase distortion, acts as a filter that is matched to the channel and to each received signal-element, and in the presence of Gaussian noise, maximises the signal to noise ratio at its output, and introduces no change in the level or other statistical properties of the noise samples, nor does it change the level of the data signal. However, in cases where the channel introduces both amplitude and phase distortions, the linear feedforward transversal equaliser no longer gives the best tolerance to Gaussian noise [16]. Indeed, when the z-transform of the sampled-impulse response of the channel Y(z), has all its roots inside the unit circle in the z-plane, at high signal to noise ratios when error extension effects can be neglected, the pure non-linear equaliser (Figure A2.2.2) will always give a better tolerance to Gaussian noise than the linear equaliser. In general, when Y(z) has roots both inside and outside the unit circle, equalisation of the channel is best carried out by the conventional non-linear equaliser [16].
### Table A2.1 Differential Encoding of the 16 or 64-Level QAM signal

<table>
<thead>
<tr>
<th>$a_{i,1}$</th>
<th>$a_{i,2}$</th>
<th>$\beta_{i-1,1}$</th>
<th>$\beta_{i-1,2}$</th>
<th>$\beta_{i,1}$</th>
<th>$\beta_{i,2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: $b=4$ for 16 or 64-Level QAM respectively

### Figure A2.1 Block Diagram of the Differential Encoder
Figure A2.2 Signal Constellation for a 16-Level Differentially Encoded QAM Signal
Figure A2.3 Signal Constellation for a 64-Level Differentially Encoded Signal
Figure A2.2.1 Linear feedforward Transversal Equaliser

Figure A2.2.2 Pure non-linear Equaliser and Detector
References


APPENDIX THREE [A3]

A3. The Clark-Hau algorithm as described is in fact equivalent to the Newton-Raphson method for locating the zeros of a polynomial. Consider the polynomial $Q(x)$ where

$$Q(x) = A_n x^n + A_{n-1} x^{n-1} + \ldots + A_0 = 0 \quad 3.2.5.22$$

Let the zeros $Q(x)$ be $x_0, x_1, \ldots, x_n$ such that $Q(x_0) = Q(x_1) = \ldots = Q(x_n) = 0$. The Newton-Raphson method enables the roots (zeros) of $Q(x)$ to be computed as follows

$$a_{i+1} = a_i - \frac{Q(a_i)}{Q'(a_i)} \quad 3.2.5.23$$

where $a_i$ is an estimate of one of the roots at the $i^{th}$ iteration. $Q(a_i)$ is the value of the polynomial $Q(x)$ at $x=a_i$ and as $i \to \infty$, $a_i \to \{x_i\}$. Suppose now that the z-transform of the sampled impulse response of the channel is

$$Y(z) = y_0 + y_1 z^{-1} + \ldots + y_k z^{-k} = \quad 3.2.5.24$$

$$= (1 + \beta_1 z)(u_0 z^{-1} + u_1 z^{-2} + \ldots + u_{k-1} z^{-k})$$

where $-\frac{1}{b_1}$ is a root (zero) of $Y(z)$ lying outside the unit circle in the $z$-plane. Now let

$$x = -z^{-1} \quad 3.2.5.25$$

and

$$Q(x) = Y(-z^{-1}) = y_0 - y_1 x + y_2 x^2 - y_3 x^3 + \ldots + y_k (-x)^{k-1} \quad 3.2.5.26$$
where it can be seen that $Q(x)$ has a root $\beta_1$ which lies inside the unit circle in the x-plane. The first derivative of $Q(x)$ is

$$Q'(x) = -y_1 + 2y_2x - 3y_3x^2 + \ldots + gy_x(-x)^{x-1} \quad 3.2.5.27$$

Substituting $\lambda_i$ for $x$ in Equations 3.2.5.26-27, where $\lambda_i$ is an estimate of $\beta_1$,

$$Q(\lambda_i) = y_0 - \lambda_i y_1 + \lambda_i^2 y_2 - \lambda_i^3 y_3 + \ldots + (-\lambda_i)^2 y_x \quad 3.2.5.28$$

$$Q'(\lambda_i) = -y_1 + 2y_2\lambda_i - 3y_3\lambda_i^2 + \ldots + g y_x(-\lambda_i)^{x-1} \quad 3.2.5.29$$

According to Newton-Raphson method (Equation 3.2.5.23)

$$\lambda_{i+1} = \lambda_i - \frac{Q(\lambda_i)}{Q'(\lambda_i)} = \lambda_i + \frac{y_0 - \lambda_i y_1 + \lambda_i^2 y_2 - \lambda_i^3 y_3 + \ldots + (-\lambda_i)^2 y_x}{y_1 - 2\lambda_i y_2 + 3\lambda_i^2 y_3 + \ldots + g(-\lambda_i)^{x-1} y_x} \quad 3.2.5.30$$

It can be seen that the evaluation of $\lambda_{i+1}$ (Equation 3.2.5.30) is identical to that of Clark-Hau algorithm when $c=1$; (Equations 3.2.5.5, 3.2.5.7 and 3.2.5.9).

A3.1 The Levinson Recursion [1]

The determination of the L point least-squares optimal filter is obtained from the solution of the linear algebraic equations

$$\sum_{i=0}^{L-1} r(i-j)f(i) = g(j) \quad \text{where } j=0,1,\ldots,L-1,$$

or

$$Rf = g \quad 3.1.1$$
where \( f=[f(0), f(1), \ldots, f(L-1)]' \), \( g=[g(0), g(1), \ldots, g(L-1)]' \), and where \( R \) has elements \( R(i, j)=r(i-j) \). A significant reduction in the computational requirements for the solution of this set can be obtained by exploiting the Toeplitz structure of \( R \). Levinson [2] has produced an algorithm which solves equations 3.1.1 in approximately \( 2L^2 \) operations and with a reduced storage requirement as compared to that of classical algorithms. Note that classical techniques for the solution of such set require \( O(L^3/2) \) operations and \( O(L^2) \) for storage.

The algorithm begins with the (trivial) solution to the scalar \((1\times 1)\) set:

\[
\begin{align*}
\begin{bmatrix}
    r(0) & r(1) & r(2)
  \end{bmatrix}
\begin{bmatrix}
    f_0^{(2)} \\
    f_1^{(2)} \\
    f_2^{(2)}
  \end{bmatrix}
  &=
  \begin{bmatrix}
    g(0) \\
    g(1) \\
    g(2)
  \end{bmatrix} 
\end{align*}
\]

3.1.2

and iteratively increases the order of the equations up to the full \((L\times L)\) system. Rather than giving a formal statement of the iteration equations, we will illustrate the algorithm by examining the iteration at a single step. Consider the iteration from \( i=2 \) to \( 3 \). That is, assume that we have computed from the previous step the filter coefficients \( f_0^{(2)}, f_1^{(2)}, f_2^{(2)} \), where

\[
\begin{align*}
\begin{bmatrix}
    r(0) & r(1) & r(2)
  \end{bmatrix}
\begin{bmatrix}
    f_0^{(2)} \\
    f_1^{(2)} \\
    f_2^{(2)}
  \end{bmatrix}
  &=
  \begin{bmatrix}
    g(0) \\
    g(1) \\
    g(2)
  \end{bmatrix} 
\end{align*}
\]

3.1.3

Also, assume that as a by product of the computation we have calculated auxiliary coefficients \( a_0^{(2)}, a_1^{(2)}, a_2^{(2)} \) and \( a_3 \), where
where $\gamma_3$ is computed directly from Eq. 3.1.13 as

$$\gamma_3 = r(3)f_0^{(2)} + r(2)f_1^{(2)} + r(1)f_2^{(2)}$$

We then extend the auxiliary system 3.1.4 as

$$\begin{bmatrix}
    r(0) & r(1) & r(2) & r(3) \\
    r(1) & r(0) & r(1) & r(2) \\
    r(2) & r(1) & r(0) & r(1) \\
    r(3) & r(2) & r(1) & r(0)
\end{bmatrix}
\begin{bmatrix}
    a_0^{(2)} \\
    a_1^{(2)} \\
    a_2^{(2)} \\
    0
\end{bmatrix} =
\begin{bmatrix}
    a_3 \\
    0 \\
    0 \\
    \beta_3
\end{bmatrix}$$

where $\beta_3$ may be computed directly from 3.1.17 as in

$$\beta_3 = r(3)a_0^{(2)} + r(2)a_1^{(2)} + r(1)a_2^{(2)}$$

Now, the Toeplitz structure of the coefficient matrix allows us to reverse the equations 3.1.7 without changing the coefficient matrix $R$. That is
We then subtract some constant multiple $k_3$, say, of the equations 3.1.9 from 3.1.7 as

$$
\begin{bmatrix}
  r(0) & r(1) & r(2) & r(3) \\
  r(1) & r(0) & r(1) & r(2) \\
  r(2) & r(1) & r(0) & r(1) \\
  r(3) & r(2) & r(1) & r(0)
\end{bmatrix}
\begin{bmatrix}
  a_0^{(2)} \\
  a_1^{(2)} \\
  a_2^{(2)} \\
  a_0^{(2)}
\end{bmatrix}
= \begin{bmatrix}
  0 \\
  0 \\
  0 \\
  a_3
\end{bmatrix}
$$

3.1.9

We choose $k_3$ so that the resulting system will have only one non-zero element on the right-hand side. This may be achieved by selecting

$$
\begin{bmatrix}
  r(0) & r(1) & r(2) & r(3) \\
  r(1) & r(0) & r(1) & r(2) \\
  r(2) & r(1) & r(0) & r(1) \\
  r(3) & r(2) & r(1) & r(0)
\end{bmatrix}
\begin{bmatrix}
  a_0^{(2)} \\
  a_1^{(2)} \\
  a_2^{(2)} \\
  0
\end{bmatrix}
- k_3
\begin{bmatrix}
  0 \\
  a_2^{(2)} \\
  a_1^{(2)} \\
  a_0^{(2)}
\end{bmatrix}
= \begin{bmatrix}
  a_3 \\
  0 \\
  0 \\
  -k_3 a_3
\end{bmatrix}
$$

3.1.10

so that from Eq.3.1.20 we have

$$
k_3 = \frac{\beta_3}{a_3}
$$

3.1.11
\[
\begin{bmatrix}
    r(0) & r(1) & r(2) & r(3)
  \end{bmatrix}
\begin{bmatrix}
    a_0^{(3)} \\
    a_1^{(3)} \\
    a_2^{(3)} \\
    a_3^{(3)}
  \end{bmatrix}
= \begin{bmatrix}
    0 \\
    0 \\
    0 \\
    0
  \end{bmatrix}
\] 3.1.12

where

\[a_4 = a_3 - k_3 \beta_3\] 3.1.13

\[a_0^{(3)} = a_0^{(2)}\]
\[a_1^{(3)} = a_1^{(2)} - k_3 a_2^{(2)}\]
\[a_2^{(3)} = a_2^{(2)} - k_3 a_1^{(2)}\]
\[a_3^{(3)} = -k_3 a_0^{(2)}\] 3.1.14

We can also reverse the system 3.1.14 because of the autocorrelation matrix structure. This yields

\[
\begin{bmatrix}
    r(0) & r(1) & r(2) & r(3)
  \end{bmatrix}
\begin{bmatrix}
    a_0^{(3)} \\
    a_1^{(3)} \\
    a_2^{(3)} \\
    a_3^{(3)}
  \end{bmatrix}
= \begin{bmatrix}
    0 \\
    0 \\
    0 \\
    0
  \end{bmatrix}
\] 3.1.15

We now subtract some multiple, c_3, say of 3.1.15 from 3.1.5
Finally, we select $c_3$ so that

$$
\gamma_3 - c_3 a_4 = g(3)
$$

$$
c_3 = \frac{\gamma_3 - g(3)}{a_4}
$$

and we obtain

$$
\begin{bmatrix}
\{ r(0) & r(1) & r(2) & r(3) \} & \begin{bmatrix}
\begin{bmatrix} f_0^{(3)} \\ f_1^{(3)} \\ f_2^{(3)} \\ f_3^{(3)} \end{bmatrix} \\
\begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ a_2^{(3)} \\ a_3^{(3)} \end{bmatrix} \end{bmatrix} \\
\begin{bmatrix} \left[ \begin{bmatrix} g(0) \\ g(1) \\ g(2) \end{bmatrix} - c_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
\begin{bmatrix} \left[ \begin{bmatrix} g(0) \\ g(1) \\ g(2) \end{bmatrix} - c_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
\begin{bmatrix} \left[ \begin{bmatrix} g(0) \\ g(1) \\ g(2) \end{bmatrix} - c_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
\end{bmatrix}
$$
Hence, we have computed the filter coefficients, auxiliary coefficients, and constant $a_4$ required for the next step of the iteration. At this point the process is repeated, starting with the augmentation as in 3.1.5 to produce the coefficients for the next iteration and so on.

Finally we have to specify the initial conditions for the algorithm.

\[ r(0)f_0^{(0)} = g(0) \]  

from which

\[ f_0^{(0)} = g(0) / r(0) \]  

For the auxiliary vector we have

\[ r(0)a_9^{(0)} = a_1 \]  

We may select $a_9^{(0)}$ as an arbitrary non-zero value; $a_0^{(0)}=1$ is a common choice, and this gives

\[ a_1 = r(0) \]
A3.2 Durbin's Algorithm [1]

Consider the simplified normal equations arising in prediction error filtering:

\[
\begin{bmatrix}
  r(0) & r(1) & r(2) & \cdots & r(L-1) \\
  r(1) & r(0) & \ddots & \vdots & \\
  \vdots & \ddots & \ddots & \ddots & \\
  r(L-1) & r(L-2) & \cdots & r(0)
\end{bmatrix}
\begin{bmatrix}
f(0) \\
f(1) \\
\vdots \\
f(L-1)
\end{bmatrix} = \begin{bmatrix}
J_{\text{nn}} \\
0 \\
\vdots \\
0
\end{bmatrix}
\]

\[f = [f(0), f(1), \ldots, f(L-1)^t] = [1, -c_1, -c_2, \ldots, -c_p]^t \]

where \(c_i\) is the \(i^{th}\) least-squares predictor. A modification exists for the solution of these equations, known as Durbin's algorithm [3]. The algorithm is similar to the Levinson recursion but exploits the simplified form of the right-hand side in Eq.3.2.1 and, for systems of the same order, provides a further computational reduction compared to Levinson, by about a factor of two.
As with the Levinson recursion, Durbin’s algorithm begins with the (trivial) solution to the scalar (1x1) set:

$$r(0)f(0)^{(0)} = J^{(0)}_{\min}$$  \hspace{1cm} 3.2.3

and iteratively obtains the solution for orders 2, 3, ..., up to the full (LxL) system. We begin by considering a single step in the solution from $i=2$ to 3. That is, we assume we have computed from the previous iteration the filter coefficients $f_0^{(2)}, f_1^{(2)}, f_2^{(2)}$, where

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} f_0^{(2)} \\ f_1^{(2)} \\ f_2^{(2)} \end{bmatrix} = \begin{bmatrix} J^{(2)}_{\min} \\ 0 \\ 0 \end{bmatrix}$$  \hspace{1cm} 3.2.4

The approach to the solution is similar to that for the Levinson algorithm, but the result is even simpler in this case because no auxiliary vector is required. We begin by augmenting equations 3.2.4 as

$$\begin{bmatrix} r(0) & r(1) & r(2) & r(3) \\ r(1) & r(0) & r(1) & r(2) \\ r(2) & r(1) & r(0) & r(1) \\ r(3) & r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} f_0^{(2)} \\ f_1^{(2)} \\ f_2^{(2)} \\ a_3 \end{bmatrix} = \begin{bmatrix} J^{(2)}_{\min} \\ 0 \\ 0 \\ a_3 \end{bmatrix}$$  \hspace{1cm} 3.2.5

where the term $a_3$ is computed directly from 3.2.5 as
\[ a_3 = r(3) f_0^{(2)} + r(2) f_1^{(2)} + r(1) f_2^{(2)} \]  

Now, the Toeplitz structure of the coefficient matrix allows us to reverse the equations 3.2.5 without changing the coefficient matrix R. That is

\[
\begin{bmatrix}
    r(0) & r(1) & r(2) & r(3) \\
    r(1) & r(0) & r(1) & r(2) \\
    r(2) & r(1) & r(0) & r(1) \\
    r(3) & r(2) & r(1) & r(0)
\end{bmatrix}
\begin{bmatrix}
    0 \\
    f_0^{(2)} \\
    f_1^{(2)} \\
    f_2^{(2)}
\end{bmatrix} =
\begin{bmatrix}
    a_3 \\
    0 \\
    0 \\
    J_{\text{min}}^{(2)}
\end{bmatrix}
\] \hspace{1cm} 3.2.7

We may subtract some multiple \( k_3 \) of 3.2.7 from 3.2.5 as

\[
\begin{bmatrix}
    r(0) & r(1) & r(2) & r(3) \\
    r(1) & r(0) & r(1) & r(2) \\
    r(2) & r(1) & r(0) & r(1) \\
    r(3) & r(2) & r(1) & r(0)
\end{bmatrix}
\begin{bmatrix}
    f_0^{(2)} \\
    f_1^{(2)} \\
    f_2^{(2)} \\
    0
\end{bmatrix} -
\begin{bmatrix}
    k_3 \\
    0 \\
    0 \\
    0
\end{bmatrix} =
\begin{bmatrix}
    J_{\text{min}}^{(2)} \\
    0 \\
    0 \\
    a_3
\end{bmatrix}
\] \hspace{1cm} 3.2.8

where we select \( k_3 \) so that
Hence we obtain

\[
\begin{bmatrix}
    r(0) & r(1) & r(2) & r(3) & f_0^{(3)} \\
    r(1) & r(0) & r(2) & r(3) & f_1^{(3)} \\
    r(2) & r(1) & r(0) & r(3) & f_2^{(3)} \\
    r(3) & r(2) & r(1) & r(0) & f_3^{(3)}
\end{bmatrix}
\begin{bmatrix}
    J_{mn}^{(3)} \\
    J_{mn}^{(2)} \\
    0 \\
    0
\end{bmatrix}
= 0
\]

where

\[
\begin{align*}
f_0^{(3)} &= f_0^{(2)} \\
f_1^{(3)} &= f_1^{(2)} - k_3 f_2^{(2)} \\
f_2^{(3)} &= f_2^{(2)} - k_3 f_0^{(2)} \\
f_3^{(3)} &= -k_3 f_0^{(2)}
\end{align*}
\]

and

\[
J_{mn}^{(3)} = J_{mn}^{(2)} - k_3 a_3
\]

Now from equation 3.2.9
Hence, we have computed the filter coefficients, and minimum mean-squared error required for the next step. Since we have completely specified the iterative update, it only remains to specify the initial conditions for the algorithm. We have

\[ r(0)f_0^{(0)} = J_{\text{min}}^{(0)} \]  

from which

\[ f_0^{(0)} = J_{\text{min}}^{0} / r(0) \]  

Now, the equations for the complete algorithm follow:

i. Initialization
\[ r(0)f_0^{(0)} = J_{\text{min}}^{(0)} = E\{x^2(n)\} \]  

3.2.17

For \( j = 1, 2, \ldots, L \)

\[
k_j = \frac{a_j}{J_{\text{min}}^{(j-1)}} = \frac{r(j)f_0^{j-1} + r(j-1)f_1^{j-1} + \ldots + r(1)f_{j-1}^{j-1}}{J_{\text{min}}^{(j-1)}}
\]  

3.2.18

\[ J_{\text{min}}^{(j)} = (1 - k_j^2)J_{\text{min}}^{(j-1)} \]  

3.2.18

\[ f_i^{(j)} = f_i^{(j-1)} - k_j f_{j-1}^{(j-1)} \]  

3.2.19

where \( i = 1, 2, \ldots, j \) and

\[ f_j^{(j-1)} = 0, \quad f_0^{(j)} = 1 \]  

3.2.20

Equation 3.2.18 is of particular interest since it shows \( J_{\text{min}} \) is directly computed at each step of the iteration. Also note that as a consequence of the monotonic property for the error energy with increasing filter length, from 3.2.18

\[ 0 \leq k \leq 1 \]  

3.2.21
In practice $k_j < 1$, since the prediction will not be perfect. The values $k_j$ are called reflection coefficients during computation of the least-squares prediction error filter provides a ready check on stability during the computation, since values of $k_j$ outside the range of the inequality can only result from numerical problems in the iteration.

### A3.3 Whittle’s Exponential-Logarithmic method

In this method of spectral factorisation a power series is substituted into other power series. The spectrum of the input polynomial $Y(z)$ is described by the autocorrelation function $R(z)$ where

$$R(z) = r_1 z^{-1} + r_2 z^{-2} + \ldots + r_n z^{-n} \quad 3.3.1$$

If $|R| > 2$ on the unit circle then a scale factor should be divided out. Insert this power series into the power series for logarithms

$$U(z) = \ln R(z) = (R - 1) - \frac{(R - 1)^2}{2} + \frac{(R - 1)^3}{3} - \ldots \quad 3.3.2$$

$$= \ldots + u_{-1} z^{-1} + u_1 + u_2 z + u_3 z^2 + \ldots$$

where $0 < R \leq 2$. By dropping negative powers of $z$ from $U(z)$ in Equation 3.3.2, $U^*(z)$ is defined:

$$U^*(z) = \frac{u_1}{2} + u_2 z + u_3 z^2 + \ldots \quad 3.3.3$$
The minimum phase factor $Y_{\min}(z)$ can be obtained by inserting $U^*(z)$ (Equation 3.3.3) into the power series for the exponential

$$Y_{\min}(z) = e^{u^*(z)} =$$

$$= 1 + U^* + \frac{(U^*)^2}{2!} + \frac{(U^*)^3}{3!} + ...$$

The spectrum of $Y_{\min}(z)$ is $R(z)$. This becomes evident by considering the following identities

$$R(z) = e^{iR(z)} =$$

$$= \exp\left(\frac{u^*}{2} + \sum_{k=1}^{\infty} \frac{u_k^*}{k} z^k \right)$$

$$= \exp\left(\frac{u^*}{2} + \sum_{k=1}^{\infty} \frac{u_k^*}{k} z^k \right) \exp\left(\frac{u_n}{2} + \sum_{k=1}^{\infty} u_k z^k \right)$$

$$= \exp\left[ U^*(\frac{1}{z}) \right] \exp[U^*(z)]$$

$$= \bar{Y}\left(\frac{1}{z}\right) Y(z) = Y_{\min}(z) Y(z)$$

$R(z)$ has been factored into the desired conjugate parts. The proof that $Y_{\min}(z) = e^{U^*(z)}$ is minimum phase is as follows. Note that $U^*(z)$ is finite and does not contain powers of $z^{-1}$. By observing that the imaginary part of $U(z)$ on the unit circle is the phase angle of $Y_{\min}(z)$. To be minimum phase, the phase of $Y_{\min}(z)$ must not be augmented by multiples of $2\pi$ as $z$ goes around the unit circle.

For minimum phase, the phase should be periodic with period $2\pi$. The phase $u_1 \sin \omega + u_2 \sin 2 \omega + ...$ obviously satisfies the condition. In a more abstract fashion note that the only way for $Y_{\min}(z_0) = e^{U^*(z_0)}$ to be zero for some $z_0$ would be if $U^*(z)$ were equal to $-\infty$; in other words, if $U^*(z)$ were nonconvergent. This is impossible inside the unit circle because the log series for $U(z)$
was taken to be absolutely convergent (finite) inside the circle. Since \( Y_{min}(z) \) cannot have zeros inside the unit circle, it must be minimum phase.

**A.3.4 Proof of the Fejér-Riesz theorem**

Since it has been assumed that \( W(e^{j\omega_T}) \) is real; \( w_{\omega} = w_{\omega}^* \). Thus,

\[
W'(z) = W\left(\frac{1}{z}\right)
\]

**A.3.4.1**

Let \( z_i \) is a root of \( W(z) \). Then

\[
W(z_i) = 0
= W'(z_i) = W\left(\frac{1}{z^*_i}\right)
\]

**A.3.4.2**

Therefore, \( 1/z_i^* \) is also a root. (If \( z_i = e^{j\phi} \) is a root on the unit circle, then, since \( W(e^{j\phi}) \geq 0 \) must have even multiplicity. Let \( z_i \) be all the roots of \( W(z) \) that are outside the unit circle or on its boundary. There are \( m \) such roots if we count the roots on the boundary with half their multiplicity. The remaining \( m \) roots \( 1/z_i^* \), are inside the unit circle or on its boundary. This leads to the following factorisation

\[
W(z) = A z^{-m} \prod_{i=1}^{m} \left(z - z_i\right) \prod_{i=1}^{m} \left(zz_i - 1\right)
\]

**A.3.4.3**

where \( A \) is some constant.
Equation A3.4.3 can be rewritten as

\[ W(z) = Y(z) \prod_{i=1}^{m} (z - z_i) \]  

where the function \( Y(z) \) in Equation A3.4.4 is of the form

\[ Y(z) = \sqrt{A} z^{-m} \prod_{i=1}^{m} (z z_i - 1) \]  

The function \( Y(z) \) in Equation A3.4.5 is of the form of Equation 3.3.5.10 and satisfies Equation 3.3.5.11 (Chapter 4, Section 3.3.5) because

\[ |e^{j\omega T} z_i - 1| = |e^{j\omega T} - z_i| \]  

To determine \( Y(z) \) from Equation A3.4.5 the polynomial \( W(z) \) in Equation A3.4.4 must be factored. The following technique avoids factorisation:

It can be shown that the values of \( Y(z) \) at the points \( z_k = e^{j\omega T} \), \( \omega_k = \frac{2\pi k}{m+1} \), and its coefficients \( y_n \) form a Discrete Fourier Series (DFS) pair. With

\[ a(\omega) + j\phi(\omega) = \ln Y(e^{j\omega T}) \]  

Now, the function \( a(\omega) \) is known (Equations 3.3.5.11 and A3.4.7) and is given

\[ a(\omega) = \frac{1}{2} \ln W(e^{j\omega T}) \]  

It is obvious from Equation A3.4.7 that in order to determine \( Y(z) \), it suffices, to compute \( \phi(\omega_k) \). If \( W(z) \) has no roots on the unit circle, then has no roots in the region \(|z| \geq 1\).
Therefore the function \( \ln Y(z) \) is analytic in this region and can be expanded into a convergent power series

\[
\ln Y(z) = \sum_{n=0}^{\infty} a_n z^{-n}
\]  

A3.4.9

From the above Equation (A3.4.9) it follows that if the coefficients \( a_n \) are real then

\[
a(\omega) = \sum_{n=0}^{\infty} a_n \cos nT\omega
\]

\[
\varphi(\omega) = -\sum_{n=1}^{\infty} a_n \sin nT\omega
\]

A3.4.10

That is \( \varphi(\omega) \) is the Hilbert Transform of \( a(\omega) \). Therefore to determine the coefficients \( Y_n \) of the function \( Y(z) \), \( \varphi(\omega) \) is first computed in terms of the known \( a(\omega) \), using two DFS of order \( N \) where \( N \) is sufficiently high to avoid aliasing errors. The DFS of order \( m+1 \) of the resulting numbers \( a(\omega_k) + j\varphi(\omega_k) \) yields \( Y_n \). Equations A3.4.1-10 together with Equations 3.3.5.9-11 are completely identical to the Equations 3.3.5.1-9; They serve the purpose to view the same problem through a different angle, and thus to facilitate, perhaps, a deeper understanding of the particular algorithm.

Note that it is possible to determine the phase function by

\[
\varphi(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} a(n) \cot \frac{n-\omega}{2} \, dn
\]

A3.4.11

However, at least for filters of higher degree, this integral relation is not appropriate for a numerical evaluation.
A3.4.1 Hilbert Transform Relations

There exist relationships between the real and imaginary parts of the magnitude and phase of the Fourier transform. In the mathematics literature these relations are known as Hilbert Transform relations.

If a real or complex function \( f(t) \) is causal then the real and imaginary parts of its Fourier transform are related.

If \( a_n=0 \) for \( n \leq 0 \), that is, if

\[
 f(t) = r(t) + ix(t) = \sum_{n=1}^{\infty} a_n e^{j\omega_n t} \quad \text{A3.4.1.1}
\]

then

\[
 r(t) = -\frac{1}{T} \int_{-T/2}^{T/2} x(t - \tau) \cot \frac{\omega \tau}{2} d\tau \quad \text{A3.4.1.2}
\]

\[
 x(t) = \frac{1}{T} \int_{-T/2}^{T/2} r(t - \tau) \cot \frac{\omega \tau}{2} d\tau
\]

Note that if the coefficients \( a_n \) are real, then A3.4.1.1 yields

\[
 r(t) = \sum_{n=1}^{\infty} a_n \cos n\omega_n t \quad \text{A3.4.3}
\]

\[
 x(t) = \sum_{n=1}^{\infty} a_n \sin n\omega_n t
\]

Thus, the cosine and sine series satisfy Equations A3.4.1.2
A3.5 LU FACTORISATION_DECOMPOSITION [4, 5]

The most basic factorisation expresses any square matrix as the product of two essentially triangular matrices, one of them a permutation of a lower triangular matrix and the other an upper triangular matrix. The factorisation is often called the LU (or sometimes LR) factorisation. Most of the algorithms for computing it are variants of the Gaussian elimination. This factorisation can be used to obtain the inverse and the determinant of a matrix. It is also the basis for the linear equation solution or “matrix division”.

Let matrix $A$ be the product of two matrices

$$LU = A$$ \hspace{1cm} A3.5.1

where $L$ is lower triangular (has elements only on the diagonal and below) and $U$ is upper triangular (elements only on the diagonal and above).

Equation A4.3.1 can be used to solve a linear set

$$Ax = b$$ \hspace{1cm} A3.5.2

$$Ax = (LU)x = L(Ux) = b$$

by first solving for the vector $y$ such that

$$Ly = b$$ \hspace{1cm} A3.5.4

$$Ux = y$$ \hspace{1cm} A3.5.5
A3.6 Extended Newton's method for the complex case[6]

Newton's method may be modified to allow for the possibility of complex roots. Consider the polynomial

$$p_{*}(z) = a_{n}z^{n} + a_{n-1}z^{n-1} + \ldots + a_{*}$$  \hspace{1cm} \text{A3.6.1}$$

where \(\{a_{j}\}_{j=0,1,\ldots,n}\) are real or complex coefficients and \(z\) may be real or complex. The polynomial in Equation A3.6.1 may be written as

$$p_{*}(z) = R(x,y) + i*S(x,y)$$  \hspace{1cm} \text{A3.6.2}$$

where \(R\) and \(S\) are real or complex functions and \(z=x+iy\). To locate a root of the polynomial \(p_{*}(z)\) (Equation A3.6.1) the pair of nonlinear simultaneous equations given by

\[
\begin{align*}
R(x,y) &= 0 \\
S(x,y) &= 0
\end{align*}
\]  \hspace{1cm} \text{A3.6.3}$$

must be solved. The extension of Newton's method to solve the system of \(m\) equations in \(m\) unknowns

\[
\begin{align*}
f(x) &= 0 \\
f^{T}(x) &= (f_{1}(x), f_{2}(x), \ldots, f_{m}(x)) \\
x^{T} &= (x_{1}, x_{2}, \ldots, x_{m}) \\
x_{k+1} &= x_{k} - J_{k}^{-1}f(x_{k})
\end{align*}
\]  \hspace{1cm} \text{A3.6.4}$$
where $J$, the Jacobian matrix associated with $f$, is given by

$$ J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_m} \end{pmatrix} $$  \[A3.6.5\]

now for the system $A3.6.3$ the iteration $A3.6.4$ gives

$$ \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} (R_x)_k & (R_y)_k \\ (S_x)_k & (S_y)_k \end{pmatrix}^{-1} \begin{pmatrix} R_k \\ S_k \end{pmatrix} \quad A3.6.6$$

Using the Cauchy-Rieman Equations [7] the following set of Equations is obtained

$$ (R_x)_k = (S_y)_k $$
$$ (R_y)_k = -(S_x)_k \quad A3.6.7 $$

The system $A3.3.6$, given Equation $A3.6.7$, now becomes

$$ \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \frac{1}{(R_x)_k^4 + (S_x)_k^4} \begin{pmatrix} R_k(R_x)_k + S_k(S_x)_k \\ S_k(R_x)_k - R_k(S_x)_k \end{pmatrix} \quad A3.6.8 $$

1. Real coefficients-complex roots

Assume that the coefficients in Equation $A3.6.1$ are real. If $x + iy$ is a root then so is $x - iy$. Next consider a factor of the form...
\[(z-(x+iy))(z-(x-iy)) = \]
\[z^2 - uz - v\]  \hspace{1cm} A3.6.9

where it is assumed that \(y \neq 0\) and \(u = 2x\) and \(v = x^2 + y^2\) are real. For arbitrary \(z\)

\[p_\ast(z) = (z^2 - uz - v) q_{\ast -1}(z) + b_{\ast -1}z + b_{\ast}\]

where

\[q_{\ast -1}(z) = b_{\ast}z^{n-2} + b_1z^{n-3} + \cdots + b_{n-1}z + b_{n-2}\]  \hspace{1cm} A3.6.10

and the \(\{b_j| j = 0, 1, \ldots, n\}\) are real. Hence

\[p_\ast(x + iy) = (b_{\ast -1}x + b_{\ast}) + ib_{\ast -1}y\]

and if \(x + iy\) is a root of \(p_\ast(z)\) then,

\[p_\ast(z) = (z^2 - uz - v) q_{\ast -1}(z)\]

and \(q_{\ast -1}(z)\) is the deflated polynomial.

Comparing coefficients in Equations A3.6.1 and A3.6.10 yields

\[a_{\ast} = b_{\ast}\]

\[a_{\ast -1} = b_{\ast -1} - ub_{\ast}\]

\[a_j = b_j - ub_j - vb_{j-2}, \quad j = 2, 3, \ldots, n - 1\]  \hspace{1cm} A3.6.11

\[a_{\ast -1} = b_{\ast -1} - vb_{\ast -2}\]
which suggest the Hornor algorithm

\[ b_0 = a_0 \]

\[ b_1 = a_1 + u b_0 \]

\[ b_j = a_j + u b_{j-1} + v b_{j-1}, \quad j = 2, 3, \ldots, n - 1 \]

\[ b_n = a_n + v b_{n-1} \]

Thus,

\[ p_*(x + iy) = R + iS = (x b_{n-1} + b_n) + iy b_{n-1} \]

yielding a fast algorithm for the solution of Equation A3.6.8.

Now, for the derivatives (by differentiating Equation A3.6.10)

\[ p'_*(z) = (z^2 - uz - v)q_{n-1}'(z) + (2z - u)q_{n-1}(z) + b_{n-1} \]

\[ p'_*(x + iy) = (2x - u + 2iy)q_{n-1}(x + iy) + b_{n-1} = 2iyq_{n-1}(x + iy) + b_{n-1} \]

Let

\[ q_{n-1}(z) = (z^2 - uz - v)T(z) + c_{n-2}z + c_{n-1} \]
where

\[ T(z) = c_{n}z^{n-4} + c_{n-1}z^{n-5} + \ldots + c_{2}z + c_{1} \quad \text{(A3.6.16)} \]

Equating coefficients yields

\[ c_{n} = b_{n} \]
\[ c_{n-1} = b_{n-1} + uc_{n} \]
\[ c_{j} = b_{j} + uc_{j-1} + vc_{j-1}, \quad j = 2, 3, \ldots, n - 3 \quad \text{(A3.6.17)} \]
\[ c_{n-1} = b_{n-1} + vc_{n-2} \]

which gives

\[ q_{n-1}(x+iy) = (c_{n-3}x + c_{n-2}) + iyc_{n-3} \quad \text{(A3.6.18)} \]

Hence

\[ p_{s}(x+iy) = (b_{n-1} - 2c_{n-3}y^2) + i2y(c_{n-3}x + c_{n-2}) = \]
\[ = R_{s} + iS_{s} \quad \text{(A3.6.19)} \]

Since \( p_{s}(x+iy) = R_{s} + iS_{s} \) [7] this enables \( R_{s} \) and \( S_{s} \) to be evaluated efficiently.

2. Complex coefficients-complex roots

Consider the case where the coefficients of the polynomial \( p_{s}(x) \) are complex. Consider also the case where the roots of the polynomial are distinct (not pairs of conjugates). Then
\[ p_\ast(z) = (z - (x + iy)) q_{\ast -1}(z) + b_\ast \]

where

\[ q_{\ast -1}(z) = b_\ast z^{\ast -1} + b_\ast z^\ast + \ldots + b_{\ast -1} z + b_{\ast -1} \]

Now, by equating powers in Equations A3.6.1 and A3.6.20 the following recurrence relations are obtained

\[ a_n = b_{n-1} \]

\[ a_{n-1} = b_{n-2} (x+iy) b_{n-1} \quad \text{(A3.6.21)} \]

\[ a_j = b_{j-1} - (x+iy) b_j \]

\[ a_n = R - (x+iy) b_n \]

So the \( b_j \)'s may be calculated

\[ b_j = a_{j+1} + (x+iy) b_{j+1} \quad j = n-1, \ldots, 0 \quad b_n = 0 \quad \text{(A3.6.22)} \]

From the last Equation in A3.6.21

\[ R = a_0 + (x+iy) b_0 = b_{-1} \]

where \( b_{-1} \) is defined by Equation A3.6.22
If the quotient in Equation A3.6.20 is again divided by \((z-(x+iy))\) then

\[ p_n(z) = (z-(x+iy))^2(c_0z^{n_2} + ... + c_1z^{n_3} + ... + c_n) + (z-(x+iy))R + R \quad \text{A3.6.23} \]

where

\[ R = p'_{n}(x+iy) \]

Now,

\[ c_j = b_{j+1} + (x+iy)c_{j+1}, \quad j = n-2, ... 0 \]

\[ c_{n-1} = 0 \]

\[ R' = b_{n} + (x+iy)c_{n} = c_{n-1} \quad \text{A3.6.25} \]

### A3.7 SCHUR DECOMPOSITION

Let matrix \(A\) have order \(n\) with complex elements. Then there exists a unitary matrix \(U\) such that

\[
T = U^*AU = \begin{bmatrix}
\lambda_1 & * & \cdots & * \\
& \lambda_2 & \cdots & * \\
& & \ddots & \cdots & \* \\
& & & \lambda_n \\
& & & & 0
\end{bmatrix}
\]

is upper triangular, and since \(U^* = U^{-1}\),

\[
f_{A}(\lambda) = f_{T}(\lambda) = (\lambda - t_{11}) \cdots (\lambda - t_{nn})
\]

and thus the eigenvalues of \(A\) are the diagonal elements of \(T\).
A3.8 COMPANION MATRIX

Given a polynomial

\[ p(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_0 \]  

A3.8.1

It can be shown that

\[ p(\lambda) = \det[\lambda I - A] \]

for the matrix

\[
A = \begin{bmatrix}
0 & 1 & \ldots & \ldots & 0 \\
0 & 0 & 1 & 0 & : \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \ldots & 1 & \vdots \\
-a_n & -a_{n-1} & \ldots & \ldots & -a_0
\end{bmatrix}
\]

A3.8.2

The roots of \( p(\lambda) \) are the eigenvalues of \( A \). The matrix \( A \) is called the companion matrix for the polynomial \( p(\lambda) \).
References


APPENDIX 4 [A4]

A4.1. ATTENUATION AND DELAY DISTORTIONS OVER TELEPHONE CIRCUITS

Figure A4.1 shows the typical attenuation-frequency characteristic of a telephone circuit containing both audio and carrier links. The whole of the frequency band from 300 to 3000 Hz is here available for transmission but there is gradually increasing attenuation at frequencies above about 1100 Hz. Figure A4.2 shows the attenuation-frequency characteristic of a poor circuit containing both audio and carrier links. Frequencies above 2500 Hz cannot be used here and there is severe attenuation distortion within the available frequency band.

Figures A4.1 and A4.2 demonstrate the characteristic property of telephone circuits which is rising attenuation with frequency, for frequencies above about 1000 Hz. The attenuation at 1000 Hz may be as much as 30 dB for a switched line, but is unlikely to exceed 15 dB for a private line.

Figure A4.3 shows a typical group-delay frequency characteristic for a circuit containing both loaded-audio and carrier links. The group delay increases towards the lower and upper ends of the frequency band, there being a distortion of about one millisecond in the frequency band 600-2800 Hz. The better circuits tend to have a smaller increase in the group delay at the lower frequencies, say only 1/2 ms at 600 Hz.

Figure A4.4 shows the group-delay frequency characteristic of a poor circuit containing both loaded audio and carrier links. The characteristic shows a delay distortion of 1 ms in the frequency band 1000-2200 Hz. Over the very worst switched telephone lines, the delay distortion in this frequency band can be about 2 ms.
There is a very wide spread in the frequency characteristics of different telephone circuits, so that although the four characteristics considered give some idea of the attenuation and delay characteristics likely to be experienced, the characteristics of any particular circuit may differ appreciably from these.

A4.2. DERIVATION OF THE SAMPLED IMPULSE RESPONSE FROM THE ATTENUATION AND GROUP-DELAY CHARACTERISTICS

The impulse responses are derived here with respect to the carrier frequency 1800 Hz. The values of the attenuation and group-delay of the equipment filters and the telephone circuit are read from the corresponding graphs (Chapter 4) every 50 Hz over the frequency range 50-3750 Hz. The sample values of the attenuation and group-delay for the telephone circuit and the equipment filters are added together for each frequency. The combined response is then normalised so that the attenuation and group delay at the carrier frequency are both zero. The amplitude and phase angle at each frequency are then calculated from the combined impulse response as follows:

The amplitude is given by

\[ A = 10^{\frac{\text{ATT}}{20}} \]  \hspace{1cm} \text{A4.2.1} \]

where \( A \) is the value of the amplitude and \( \text{ATT} \) is the value of the attenuation in dB, which is given by the frequency characteristics of the combined response. The phase angle at each frequency is derived with respect to the phase angle at the carrier frequency, which is taken to be zero here. The relation between the phase angle and the group delay is given by
where MGD is the mean group delay and \( \phi \) is the phase angle and \( \omega = 2\pi f \).

Now Eqn. A4.2.2 can be expressed numerically as

\[
MGD = -\frac{d\phi}{d\omega}
\]  

A4.2.2

where \( MGD \) is the mean group delay and \( \phi \) is the phase angle and \( \omega = 2\pi f \).

Now Eqn. A4.2.2 can be expressed numerically as

\[
MGD = \frac{\phi_{i-1} - \phi_i}{2\pi \Delta f}
\]  

A4.2.3

where \( \Delta f \) is 50 Hz and \( \phi_i \) and \( \phi_{i-1} \) are the phase angle at the frequencies \( f_i \) and \( f_{i-1} \) respectively. Now, Eqn A4.2.3 can be written as

\[
\phi_i = \phi_{i-1} - 2\pi \Delta f \cdot MGD
\]  

A4.2.4

which gives the value of the phase angle \( \phi_i \) at frequency \( f_i \) assuming the phase angle \( \phi_{i-1} \) at \( f_{i-1} \) is known. The definition of the mean group delay is

\[
MGD = \frac{d_i + d_{i-1}}{2}
\]  

A4.2.5

where \( d_i \) and \( d_{i-1} \) are the values of the group delay (in seconds) at the frequencies \( f_i \) and \( f_{i-1} \), respectively. These values are available from the group-delay characteristics. As a result Eqn. A4.2.4 becomes

\[
\phi_i = \phi_{i-1} - \pi \Delta f (d_i + d_{i-1})
\]  

A4.2.6

Thus the phase angles at frequencies above and below the carrier frequency can be derived from the group delay by using Eqn. A4.2.6, where the phase angle at the carrier frequency is zero. The resultant complex valued frequency response of the telephone circuit and the equipment filter are then determined.
by calculating the values of the real and imaginary components of the response at each frequency, as follows

\[ R_i = A_i \cos(\phi_i) \quad \text{A4.2.7} \]

\[ I_i = A_i \sin(\phi_i) \quad \text{A4.2.8} \]

where \( A_i \) and \( \phi_i \) are determined by Eqn A4.2.1 and A4.2.6 respectively, for the given value of frequency \( (f_i) \). In the above Equations \( R_i \) and \( I_i \) are the real and imaginary values of the resultant frequency response, respectively. These values are then formatted for the inverse discrete Fourier transform (IDFT) routine.

**A4.3. The Equipment Filters**

The equipment filters include all filters in the transmission path such as those used in the modulation, demodulation, band limiting and main hum rejection. The equipment filters used here are models of practical filters that have been designed for an actual modem for application to 9600 bit/s modem, when operating with 16 point QAM signal at a symbol rate of 2400 baud. The attenuation and group-delay characteristics of the filters are shown in Fig. A4.3.1. These filters, when considered as operating on the transmitted bandpass signal, introduce an attenuation of about 15 dB at 600 and 3000 Hz, the attenuation increasing rapidly as the frequency is reduced below 600 Hz or increased above 3000 Hz, and being less than 1 dB over the frequency band from 1100 to 2200 Hz. The equipment filters (Fig. A4.3.1) are known here as equipment filter set 1.

The second set of equipment filters, filter set 2, employed here has been designed specially for use in applications over telephone network for a transmission of data at rate of 19200 bit/s, when operating with 64-point QAM signals at a symbol rate of 3200 baud's. The attenuation and group delay characteristics of equipment filter set 2 are shown in Fig. A4.3.2.
Fig A4.1 Typical attenuation-frequency characteristic

Fig A4.2 Attenuation-frequency characteristic of a poor circuit
Fig A4.3  Typical group-delay frequency characteristic

Fig A4.4 Group-delay frequency characteristic of a poor circuit
a. Attenuation characteristics

b. Attenuation and group delay characteristics of filter set 1

Fig. A4.3.1 Attenuation and group delay characteristics of filter set 1
Figure A4.3.2 Attenuation and group delay characteristics of filter set 2
A 4.4 HF RADIO LINKS

A4.4.1 Ionospheric Radio Propagation

The ability of Earth's ionosphere to propagate HF waves is well known [1-3] and has long been used in both civil and military applications. The ionosphere is a layer of ionised air molecules, which include free electrons, lying approximately 50-500 Km above the Earth's surface. This layer of ionised air molecules is formed as a result of the ultraviolet (UV) radiation emitted by the Sun passing through the upper regions of the Earth's atmosphere. The free electrons act as reflectors for HF radio waves, and their density (and hence their ability to reflect radio signals) which varies with height may be divided into three main regions, known as the D, E and F layers. The D layer, at a height of 60 to 90 Km above the surface of the Earth, is the lowest level. Its electron density varies from sunrise to sunset, reaching a maximum just after midday and dropping to almost zero at night. The critical frequency for the D layer, defined as the highest carrier frequency of a vertically incident ray that can be reflected by the layer, is around 100 to 700 kHz. Since HF radio waves have typical frequencies greater than 3 MHz, the D layer acts principally as an attenuator and is of no use for HF propagation. The E layer lies between 90 and 130 km above the surface of the Earth and is the next highest layer. The E layer has a critical frequency of about 4 MHz. As with the D layer, ionisation begins at sunrise and maximum density occurs near noon with the seasonal maximum occurring in summer. After sunset the layer gradually breaks down. Thin ionised layers, known as sporadic E layers, are often found between 90 and 150 km above the Earth. They arise only occasionally and may have a local electron density exceeding that of the rest of the E layer and which varies rapidly with time. Since they have a high critical frequency they are capable of reflecting high frequencies. The next layer, called the F layer, is a thick region that extends from 130 km upwards above the surface of the Earth and is very important for the propagation of short waves. The lower region of this layer shows different variation characteristics compared with the upper part of the layer and thus it has been
subdivided into two layers, named the $F_1$ and $F_2$ layers. The $F_1$ layer, which exists only during daytime, extends from 130 km to 220 km and has similar variations to those of the E layer. During the night time and sometimes during the day time, especially in the winter months, the two layers $F_1$ and $F_2$, merge and are termed, simply, the F layer; the critical frequency then drops to 3 to 5 MHz. The $F_2$ layer is an important part of the ionosphere for HF radio communication both during day and night time. The high electron density in the $F_2$ layer makes it a good reflector of HF radio waves, and because of its considerable height, it can support single hop propagation over a distance as great as 4000 km as shown in Figure A4.4.1.

A4.4.2 Theory of Ionospheric Radio Propagation [4]

The layers that constitute the ionosphere have three main effects on HF radio waves impinging on them, namely, refractive bending, absorption and ray splitting. Refractive bending is the actual process by which HF radio waves are returned to Earth. The refractive index, $\varepsilon$, of the ionospheric layer changes continuously with its height as $n$ is a function of the electron density in the ionised medium. The refractive index varies as

$$\varepsilon = (1 - \frac{81N}{f^2})^2$$  \hspace{1cm} \text{A4.4.2.1}$$

where $f$ is the frequency of the radio wave and $N$ is the ion density (i.e. the number of free electrons per cubic meter. The refractive bending process is clearly illustrated in Figure A4.4.2. Refractive bending occurs in the E and F layers. Absorption of the radio wave occurs mostly in the D layer where molecular density is very high and absorption is proportional to $1/f^2$, so higher frequencies suffer less attenuation due to absorption. The phenomenon of ray splitting occurs when the radio wave is within an ionised layer and it is due to the action of the magnetic field of the Earth.
The combined effect of the magnetic field and the electron motion when radio waves enter the ionised layer is such that the wave is split up into two magneto-ionic components, known as the ordinary and extraordinary rays. These rays have distinctly different characteristics and are reflected at different heights in the layer and therefore they have different propagation delays. For a given angle of incidence of a radio wave meeting a reflecting layer, total internal reflection occurs and a skywave is formed when

\[ \sin \theta_i = (1 - \frac{81N}{f^2})^{\frac{1}{2}} \]  

A4.4.2.2

The critical frequency of a layer is obtained when \( \theta_i = 0^\circ \) (vertical incidence) and represents the highest reflectable frequency of the layer at his incidence. It is given by

\[ f_c = 9\sqrt{N_m} \]  

A4.4.2.3

where \( N_m \) is the maximum ion density of the layer. Higher frequencies can be reflected from this layer at other angles of incidence but for any given angle, \( \theta_i \), there is a maximum frequency at which reflection takes place. This is called the maximum usable frequency (MUF) and is given by

\[ \text{MUF} = f_c \sec \theta_i \]  

A4.4.2.4

Because the maximum usable frequency may vary with time of the day and the season a somewhat lower frequency is used [5,6].

A4.4.3 Distortion on HF Channels

Multipath Propagation and Time Dispersion

A radio wave may be propagated to the receiver along one or more different paths, of unequal lengths. This effect is known as multipath propagation [7]. Many of these
propagation paths (modes) are possible, especially for long haul propagation, however, the number of 'effective' modes is small. For example, the transmitted radio signal may travel from the transmitter to the receiver via two skywaves which are from two different layers, in the ionosphere is shown in Figure A4.4.2a. Alternatively, the transmitted signal may travel from the transmitter to the receiver via both one and two hops, that is being reflected either once or twice from the ionosphere, as shown in Figure A4.4.2b. In general when a short pulse of RF energy is transmitted, the received signal will have an amplitude versus time profile such as that in Figure A4.4.3 [7]. The time between reception of the first and last pulse in Figure A4.4.3 is known as the time spread or time dispersion of the received signal. The time dispersion of a transmitted signal-element is usually less than 3 ms. Measurements have shown that, usually, two to four paths are present for most of the time [8] and that the time dispersion is order of 1 ms [8,9]. Time dispersion gives rise to intersymbol interference, when the symbol period becomes comparable with the relative multipath delay neighbouring symbols will overlap and make correct detection of the transmitted data difficult.

**Frequency Dispersion**

Frequency dispersion arises on a single propagation path by the variations in altitude of the ionospheric layers with time, which introduce a Doppler shift into the received signal. It can be seen from Figure A4.4.4 that there are certain times of the day, notably from 5 am until and between 5 pm to 7 pm when the reflecting layers are moving rapidly in the vertical direction - approximately 50 km/hour for the F2 layer during the evening [4]. The magnitude of the Doppler shift is frequency dependent, for example a typical operating 15 MHz frequency, would give rise to a Doppler shift of around 1 MHz [9,10].
Fading

Fading is another form of distortion occurring on HF links [11-13]. Fading is the variation with time of the received signal strength, and may be caused by several different processes. This phenomenon can be divided into two types, long and short term fading. Long term fading is the daily or seasonal variation in received signal strength due to the night/day seasonal changes in the structure and ionisation of the various layers. However, of much greater importance here is the short-term fading which can be subdivided into two distinct types selective or interference fading and Rayleigh fading. Selective fading happens when the large number of frequency components that lie, within the bandwidth of a modulated carrier, are exposed to randomly varying multipath propagation conditions. This may result in selective blackouts or fading of a small section of the bandwidth. Interference fading can also occur if at the receiver the sky wave signals are also superimposed by the ground wave signals. The time-varying nature of a practical HF channel is due to the phenomenon of Rayleigh fading. There are several effects which produce this type of fading [14], but essentially they are all functions of the short-term variations in ionisation of the reflecting layers and the variation in position of these layers. Rayleigh fading is that fading which occurs on the one or more paths which constitute the received signal. The received signal is in fact made up of the sum of several slightly different paths, all randomly added at the receiver [12]. Short term variations in the ionisation of the layer alter the way these paths add at the receiver. The amplitude distribution of the envelope of the received signal will have a Rayleigh probability density function and its phase will have a uniform probability density function. The occurrence of other fading distributions has been found in practice, such as Nakagami-Rice (or simply Ricean) probability distribution [12,15] which is a Rayleigh distribution with a specular (non-fading component as a result of say, direct ground-wave reception as well as the sky-wave paths. Variations in the positions of the reflecting layers, especially F layer variations, also contribute to this fading. The rapid (and random) variations of the ionospheric layers cause small changes in the frequency of the received signal to occur
due to the Doppler effect. These changes cause the bandwidth of the signal to be slightly increased, typically around 0.1 Hz under mild conditions and 0.5 Hz for more severe links.

**FLUTTER FADEING**

When the radio links are exposed to severe ionospheric disturbances, there is another type of interference called flutter fading. This is associated with the F region. In this the variation in signal strength takes the form of a fast rhythmic beat, as though a low frequency modulation is superimposed on the modulated carrier. Although the fading period is very small (10-100 ms), this type of fading represents a considerable source of disturbance for radio reception.

**POLARISATION FADEING**

This is due to the magnetic field of the Earth, splitting the radio waves into ordinary and extraordinary waves (section A4.4.2). The combination of the phase and amplitude of these waves changes the polarisation of the received signal to be elliptically polarised. The phase and dimensions of the axes of the ellipse rotate as the ordinary and extraordinary waves are subjected to random variations in the propagation conditions.

**ABSORPTION FADEING**

Absorption fading is caused by the variation in the absorption characteristics of the ionosphere with time. The attenuation characteristic of the D layer slowly changes and is usually greatest at sunrise and sunset. The fading period is of the order of one hour and the depth of fading can be as high as 10 dB below the mean value.
**SKIP FADING**

Skip fading is caused by the continuous variations of the MUF (section A4.4.2). The operating frequency, which at one particular instant is definitely below the MUF, may no longer be so at another instant and so penetrates the reflecting layer for a short period of time. During this time, radio communication is interrupted at the receiver. Skip fading can be avoided by working well below the MUF.

**A4.4.4 Signal Statistics**

Consider a transmitted signal that may be represented in general as

\[ s(t) = \text{Re}[u(t)e^{j2\pi f_c t}] \quad A4.4.4.1 \]

where \( f_c \) is the carrier frequency is the frequency of the carrier and \( u(t) \) is the lowpass modulating signal, given by

\[ u(t) = a(t)e^{j\theta(t)} \quad A4.4.4.2 \]

where \( a(t) \) denotes the amplitude (envelope) of \( s(t) \), and \( \theta(t) \) represents the phase of \( s(t) \). It is assumed that there are multiple propagation paths and associated with each path is a propagation delay and an attenuation factor. Both the propagation delays and attenuation factors are time variant. Due to this multipath propagation, the received bandpass signal may be expressed as

\[ x(t) = \sum_n a_n(t)s[t - \tau_n(t)] \quad A4.4.4.3 \]

where \( a_n(t) \) is the amplitude of the signal received via the \( n^{th} \) path and \( \tau_n(t) \) is the propagation delay for the \( n^{th} \) path. Substituting Equation A4.4.4.1 into Equation A4.4.4.3 results in,
The equivalent lowpass received signal is

\[ r(t) = \sum_n a_n(t)e^{-j2\pi f \tau_n^{(t)}}u[t - \tau_n(t)] \]  

A4.4.4.5

As \(r(t)\) is the response of an equivalent lowpass channel to the equivalent lowpass signal \(u(t)\), the equivalent lowpass channel is described by the time variant response

\[ h(t, \tau) = \sum_n a_n(t)e^{-j2\pi f \tau_n^{(t)}}\delta(t - \tau_n(t)) \]  

A4.4.4.6

For some channels it is more appropriate to consider the received signal as consisting of a continuum of multipath components. In such a case, the summation in Equation A4.4.4.3 may be expressed in the integral form

\[ x(t) = \int_{-\infty}^{\infty} a(\tau, t)s(t - \tau)d\tau \]  

A4.4.4.7

where \(a(\tau, t)d\tau\) denotes the attenuation at time \(t\) of all rays arriving with relative delay times in the range \((\tau, \tau+dt)\). Substituting A4.4.4.1 into A4.4.4.7 gives

\[ x(t) = \text{Re}\left\{ \int_{-\infty}^{\infty} a(\tau, t)e^{-j2\pi f \tau^{(t)}}u(t - \tau)d\tau \right\}e^{j2\pi f t} \]  

A4.4.4.8

the equivalent lowpass received signal is,

\[ r(t) = \int_{-\infty}^{\infty} a(\tau, t)e^{-j2\pi f \tau^{(t)}}u(t - \tau)d\tau \]  

A4.4.4.9
In Equation A4.4.4.9, the received signal $r(t)$ is given by the convolution of $u(t)$ with an equivalent lowpass time variant impulse response $c(\tau, t)$

\[
c(\tau, t) = a(\tau, t)e^{-j2\pi f_c \tau}
\]

Consider now, the case when an unmodulated carrier is transmitted, at frequency $f_c$. Then $u(t) = 1$ for all $t$, and the received signal for the discrete case (Equation A4.4.4.6) reduces to

\[
r(t) = \sum_n a_n(t)e^{-j2\pi f_c \tau_n(t)}
\]

\[
= \sum_n a_n(t)e^{-j\theta_n(t)}
\]

where $\theta_n(t) = 2\pi f_c \tau_n(t)$. For the continuous case the signal reduces to

\[
r(t) = \int_{-\infty}^{\infty} a(\tau, t)e^{-j2\pi f_c \tau(t)} d\tau
\]

Only when large there are large dynamic changes in the structure of the reflecting medium $a_n(t)$ changes sufficiently to cause a significant change in the received signal. On the other hand $\theta_n(t)$, will change by $2\pi$ radians whenever $\tau_n$ changes by $1/f_c$. But $1/f_c$ is a small quantity and, hence, $\theta_n$ can change by $2\pi$ radians with relatively small motions of the medium. The received signal can be modelled as a random process. This is implied by the fact that delays $\tau_n(t)$ and therefore $\theta_n(t)$, vary in a random manner. When there are a large number of paths each varying randomly and independently of each other, then according to the central limit theorem, $r(t)$ can be modelled as a complex Gaussian random process.
This means that $c(t;\tau)$ is a complex valued Gaussian random process. If the impulse response $c(t;\tau)$ is modelled as a complex valued Gaussian random process, the envelope of $c(t;\tau)$, that is $|c(t;\tau)|$ will have at any instant $t$ a Rayleigh distribution and the phase of $c(t;\tau)$ will be uniformly distributed between 0 and $2\pi$ radians. Since Rayleigh distributed envelope fading has been observed often on the majority of ionospheric media, it appears that the Rayleigh fading model best describes most HF channels.

The Rayleigh model describes a continuous random variable, derived from two independent Gaussian random variables $X$ and $Y$

$$m_X = m_Y = 0 \quad \sigma_X^2 = \sigma_Y^2 = \sigma^2$$  \hspace{1cm} (A4.4.4.13)

where $m$ and $\sigma^2$ are the mean and variance of $X$ and $Y$. The envelope

$$R = \sqrt{X^2 + Y^2}$$  \hspace{1cm} (A4.4.4.14)

has a Rayleigh distribution and a probability density function given by [16]

$$P_R(r) = \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \quad 0 \leq r \leq \infty$$  \hspace{1cm} (A4.4.4.15)

Since $R$ cannot be negative, by definition, it must have a nonzero mean value, even though $X$ and $Y$ have zero means. Figure A4.4.5 shows the probability density function of the Rayleigh distribution. The curve has a maximum at $1/\sigma \sqrt{e}$ at $r=\sigma$. The cumulative density function of the Rayleigh distribution is given by
for \( r > 0 \). The mean value of \( R \) is given by

\[
\bar{r} = \int_0^\infty r f(r) dr = \sqrt{\frac{\pi}{2} \sigma}
\]

The second moment of \( R \) is given by

\[
\]

where

\[
E[X^2] = \sigma^2 + m_x^2
\]

\[
E[Y^2] = \sigma^2 + m_y^2
\]

As \( m_x \) and \( m_y \) are zero (Equation A4.4.4.13), substituting Equation A4.4.4.19 into Equation A4.4.4.18 gives the mean square value of \( R \) to be

\[
E[R^2] = \bar{r}^2 = 2\sigma^2
\]

The variance of \( R \) is

\[
\sigma_r^2 = E[R^2] - (E[R])^2
\]

\[
= \bar{r}^2 - (\bar{r})^2
\]
From Equations A4.4.17, A4.4.20 and A4.4.21

\[ \sigma_r^2 = \left( 2 - \frac{\pi}{2} \right) \sigma^2 \]  

A4.4.4.22

A4.4.5 HF Channel Classification

Several attempts have been made at HF channel classification [17,18]. The U.S.A E.L. classification [18] is given in Table A4.4.1. Here, fading rate is given as the speed of propagation of a selective fade through a 3 KHz band. Another classification made by C.C.I.R [18] is given in Table A4.4.2. These are suggested parameter values for the testing of HF radio communication equipment on an HF channel simulator. The latter models HF channels as two independent fading paths with equal amplitude in the absence of fading, equal frequency spreads and no frequency shifts. The HF channels used for test purposes in this report are basically modified from the C.C.I.R requirement for an HF channel simulator to give up to three fading paths or skywaves. The fading paths are independent of each other and totally uncorrelated and have a multipath spread (time between the arrivals of signals from the first and the last path) of up to 3 ms. Frequency spread for the different paths is tested up to 2 Hz which corresponds to a fading rate of 88 fades per minute which is about five times worse than the value for moderate operating conditions.
Figure A4.4.1 A Single Hop Propagation

Figure A4.4.2 Refractive Bending
Figure A4.4.2a Reflection from different Layers

Figure A4.4.2b Different Number of Hops
Figure A4.4.3 Typical Response of a Multipath Channel
Figure A4.4.5 Rayleigh Probability Density Function
Multipath Spread

<table>
<thead>
<tr>
<th>Fade Rate through 3 KHz band</th>
<th>Mild 0-0.4 ms Selective Fade</th>
<th>Medium 0.4-1ms Two but not three Selective Fades</th>
<th>Severe ≥ 1 ms Three or more Selective Fades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow: 10 secs or more</td>
<td>A-1</td>
<td>B-1</td>
<td>C-1</td>
</tr>
<tr>
<td>Medium 2-10 secs</td>
<td>A-2</td>
<td>B-2</td>
<td>C-2</td>
</tr>
<tr>
<td>Fast 2 secs or less</td>
<td>A-3</td>
<td>B-3</td>
<td>C-3</td>
</tr>
</tbody>
</table>

Table A4.4.1 U.S.A.E.L. Classification of HF Channels

1) Good conditions
Multipath Spread: 0.5 ms
Frequency Spread: 0.1 Hz

2) Moderate conditions
Multipath Spread: 1 ms
Frequency Spread: 0.5 Hz

3) Poor conditions
Multipath Spread: 2 ms
Frequency Spread: 1 Hz

4) Flutter Fading (if required)
Multipath Spread: 0.5 ms
Frequency Spread: 10 Hz

Table A4.4.2 C.C.I.R. Classification of HF Channels
A4.5 Model and Simulation of the HF channel

A4.5.1 Model of the HF Channel

There are two methods available for testing the performance of a transmission system for use on HF radio channels [17]. Firstly, the constructed equipment can be tested out over an actual HF channel and its performance evaluated by error rate measurements. This method may be costly to implement and very time consuming, since any change required for the adjustments and/or improvements of the equipment could well involve alterations to the hardware. Also when several systems are to be compared, they have to be tested simultaneously because the channels characteristics randomly vary with time and hence the tests cannot be repeated under exactly the same conditions at other times. A second disadvantage of this method, is that it is difficult to ascertain the weaknesses of the equipment from its performance, because a poor performance may have one or several causes such as impulsive noise, fading rate, multipath or Doppler shifts. Finally the equipment should be tested for a sufficiently long time to cover most of the likely channel conditions.

To avoid such problems, the system can be tested over a channel simulator [19-21]. HF channel simulators, whether in hardware or software, are versatile in that a variety of conditions can simply be produced, and if desired, these can be repeated any number of times with consistent results. Also the type and amount of distortion can be controlled so that any particular weakness of the system can be identified and studied in isolation. So, a channel simulator, when it is valid, has the advantage of accuracy, regularity of performance, repeatability, availability of a large range of channel conditions, and lower cost when used in place of an actual HF channel to compare different systems.

Most of the simulator designs, given in literature, are based on the tapped delay line to represent the HF channel, which has been proposed by Watterson et al in Reference 22. This model has been adopted unanimously by the International Radio Consultative
committee (CCIR) of the International Telecommunications union (ITU). This thesis assumes the tapped delay model shown in Figure A4.4.6. It is assumed that a complex valued baseband input signal is used. The input signal is fed to an adjustable tapped delay line and is delivered at several taps, one for each ionospheric propagation path. Rayleigh fading is then imposed on the delayed signals by multiplying each signal by a suitable tap gain function, $Q_h(t)$. The resulting delayed and modulated signals from the different taps are added to form an output of the tapped delay-line. The received signal is the sum of the output of the delay line and an additional noise term $u_n(t)$ which represents the noise on HF channels.

In HF channels various types of noise may be present, such as atmospheric, man-made and thermal noise. It is common practice to represent these types of noise by white Gaussian noise [19,20]. For one propagation path, the Rayleigh fading introduced by the skywave is modelled as shown in Figure A4.4.7, where $q_1(t)$ and $q_2(t)$ are two random processes which must have the following properties [19]:

1. Each random process must be Gaussian with zero mean and the same variance
2. The random processes $q_1(t)$ and $q_2(t)$ must be statistically independent
3. The power spectrum of each random process must be Gaussian in shape, having the same rms frequency, $f_{rms}$.

The power spectra of $q_1(t)$ and $q_2(t)$ are given by

$$Q_1(f)^2 = Q_2(f)^2 = \exp\left(-\frac{f^2}{2f_{rms}^2}\right) \quad A4.5.1.1$$

and are shown in Figure A4.4.8. The frequency spread $f_{sp}$, introduced by $q_1(t)$ and $q_2(t)$ into an unmodulated carrier is defined as the width of the power spectrum and is given [19]

$$f_{sp} = 2f_{rms} \quad A4.5.1.2$$
The rms frequency is related to the fading rate $f_r$, which is defined (for a single carrier) as the number of downward crossings per second of the envelope through the median value \[14,19\]

\[
  f_{\text{rms}} = \frac{f_r}{1.475} \tag{A4.5.1.3}
\]

from Equations A4.5.1.2 and A4.5.1.3, $f_{sp}$ is related to $f_r$ by

\[
  f_{sp} = 1.356f_r \tag{A4.5.1.4}
\]

The random process $q_1(t)$ (or $q_2(t)$) is generated by filtering a zero mean white noise signal $n_1(t)$ as shown in Figure A4.4.9. The power spectra of $q_1(t)$ and $q_2(t)$ is Gaussian, hence the filter should also have a Gaussian frequency response matching the power spectrum of $q_1(t)$ and $q_2(t)$. The power spectrum of each filter is given by Equation A4.5.1.1. The frequency response of the filter is given by

\[
  F(f) = \exp\left(-\frac{f^2}{4f_{\text{rms}}^2}\right) \tag{A4.5.1.5}
\]

A fifth order Bessel filter is used in the channel simulator to provide the necessary shaping to give the random process $q_1(t)$. The frequency and impulse response of the Bessel filter approaches Gaussian, when the order of the filter is sufficiently large \[21\]. This filter is implemented using the arrangement in Figure A4.4.10 and the tap values required to obtain the three frequency spreads of interest, namely, 0.5 Hz, 1 Hz and 2 Hz are given in Table A4.4.3.

In a digital implementation of the HF model it is neither possible nor necessary to represent the fading signal $q_m(t)$ as continuous. Each of these signals must be represented as the corresponding sequence of discrete samples in time so long as they are sampled at or above the Nyquist rate. As the fading signals have Gaussian spectra, they contain all frequencies. For practical purposes a fading signal of (say) 2 Hz frequency spread (highest value assumed in this report) could be adequately
represented at a sampling frequency of 10 samples/second without any significant aliasing occurring. However, the computer simulations used in this chapter are aimed at an HF modem of 9600 bits/sec using a 16-point QAM signal transmitting at 2400 bauds. For testing a 2400 baud digital data modem, it is necessary that the channel samples are obtained at 4800 Hz. This means that 4800 samples of $q_h(t)$ must be provided every second for the simulation which represents a gross oversampling of $q_h(t)$. Alas, at this rate the filter poles are pushed very close to the unit circle in the $z$-plane. These poles must now be specified to a very high degree of accuracy (in order to define the very large tap values) to obtain the required narrowband Gaussian shapes, implied by the position of the poles in the $z$-plane, otherwise instability of the filter can occur [23-25]. This problem can be solved by the use of a much lower sampling frequency, 50 Hz, in the digital filter and employing a process of linear interpolation between the 50 fading samples per second produced by the filters, to obtain the remaining samples, as shown in Figure A4.4.11. The 50 Hz sampling frequency was chosen so that it is clearly high enough to satisfy the Nyquist sampling criterion for $q_h(t)$, yet not so low as to introduce inaccuracies in the fading samples when linear interpolation is applied. As well as obtaining more reliable digital filtering, this 50 Hz sampling frequency has a further advantage, that if a particular fading sequence is required several times in the simulation, then only the 50 Hz samples need be stored and the interpolation process carried out during the simulation, thus significantly reducing the storage requirement for a given fading sequence which normally lasts for several tens of seconds.

A4.5.2. Data Transmission over a Model of an HF channel using QAM

In this section, the transmission of the QAM signals (Chapter 2-Section 2.2) over a model of an HF radio channel (section A4.5.1) is considered. The main difference between a voice band HF radio link and a telephone channel is that the characteristics of the former vary rapidly with time compared with the latter. These rapid changes cause the received signal power to change with time with, sometimes, a complete loss of signal. Therefore, the data transmission model assumed in this Chapter is identical
The model of the data transmission system is shown in Figure A4.4.12. The information to be transmitted is carried by the data symbols \( \{ s_i \} \), where

\[
s_i = s_{0,i} + j s_{1,i}
\]

and \( j = \sqrt{-1} \), \( s_{0,i} = \pm 1 \) or \( \pm 3 \), and \( s_{1,i} = \pm 1 \) or \( \pm 3 \). It is assumed that \( s_i = 0 \) for \( i \leq 0 \), so that the impulse \( s_i \delta(t-iT) \) is the \( i^{th} \) signal-element at the input of the transmitter filter. Furthermore, \( \{s_i\} \) for \( i > 0 \) are statistically independent and equally likely to have any of their 16 different possible values.

The transmission path is a linear baseband channel that includes the HF radio link together with a linear modulator at the transmitter and a linear demodulator at the receiver. The QAM signal is the sum of two 4-level double sideband suppressed carrier AM signals with their carriers in phase quadrature. One of these carries the 'in-phase' data symbols \( \{ s_{0,i} \} \) and the other the 'quadrature' data symbols \( \{ s_{1,i} \} \). Each AM signal is associated with the corresponding modulator and demodulator to give a baseband channel, considerable coupling between the two channels being introduced by the HF radio link. Clearly, the \( \{ s_{0,i} \} \) and \( \{ s_{1,i} \} \) (Equation A4.5.2.1) are fed separately to two input terminals of the linear baseband channel. By allocating real values to the signals transmitted over one of the two channels and imaginary values to the signals over the other, a complex valued baseband signal is obtained at both the input and output of the resultant transmission path in Figure A4.4.12. The 16-level QAM signal is fed to the HF radio link where its spectrum is shifted into the HF band by a process of linear single sideband suppressed carrier amplitude modulation, the resulting signal being then transmitted via one or more independently Rayleigh fading skywaves to the receiver, where its spectrum is returned to the voice band by a process of linear single sideband suppressed carrier amplitude modulation.

The transmitter filter, transmission path and receiver filter together form a linear baseband channel whose sampled-impulse response is the complex-valued \( y(t) \). The various types of additive noise normally introduced by an HF radio link are neglected here, and it is assumed that the only noise is stationary bandlimited Gaussian noise with
zero mean and a flat power spectral density over the whole of the frequency band, which is added to the data signal at the output of the radio link. The bandpass filter at the input of the linear demodulator removes as much of the Gaussian noise outside the frequency band of the received signal as possible without unduly distorting the data signal. The filtered signal is fed to two linear coherent demodulators whose reference carriers are in phase quadrature and have the same frequency, which is constant and equal to the average instantaneous frequency of the received signal carrier, thus eliminating any constant frequency offset in the QAM signal but not tracking the small variations in the signal carrier frequency introduced by the HF radio link. The demodulated signals at the outputs of the 'inphase' and 'quadrature' coherent demodulators are taken to be real-valued and imaginary-valued, respectively, so that the resultant demodulated baseband signal \( r(t) \) is complex-valued and its bandwidth extends from about -1200 Hz to 1200 Hz.

Consider Equation 2.2.14 (Chapter 2), which gives the signal \( r(t) \), at the output of the receiver filter in Figure A4.4.12,

\[
r(t) = \sum_i s_i y_i(t - iT) + w(t) \tag{A4.5.2.2}
\]

where

\[
y_1(t - iT) = h_1(t - iT)*r_x(t) \tag{A4.5.2.3}
\]

where \( r_x(t) \),

\[
r_x(t) = (c_0(t)e^{-j2\pi ft})*d(t) \tag{A4.5.2.4}
\]

represents the overall filtering carried out at the receiver, and \( h_0(t-iT) \) is the impulse response of the transmission path and is given by

\[
h_1(t - iT) = t_1(t - iT)(q_1(t) - jq_2(t)) +
+ t_1(t - \tau - iT)(q_3(t) - jq_4(t)) \tag{A4.5.2.5}
\]
where \( t_s(t) \) is the overall filtering carried out in the transmitter.

\[
t_s(t) = a(t) \ast \left( b_s(t)e^{-j2\pi f_s t} \right)
\]

A4.5.2.6

It is obvious from Equation A4.5.2.5 that \( h(t-iT) \) consists of the time invariant impulse response \( t_s(t) \) and the random components \( q_1(t) \) to \( q_4(t) \). Each of these random components has a Gaussian-shaped power spectral density which extends over all frequencies and has an r.m.s. frequency of only a few Hertz. Also \( q_2(t) \) and \( q_4(t) \) are the Hilbert transforms of \( q_1(t) \) and \( q_3(t) \) respectively. The quantity \( \tau \) represents the relative time delay in transmission between the skywaves. Figure A4.4.13 shows the model of an HF channel which for simplicity assumes only two independent Rayleigh fading skywaves.

From Equation A4.5.2.3

\[
y_1(t-iT) = \{ t_s(t-iT)(q_1(t) - jq_1(t)) +
+ t_s(t-\tau -iT)(q_2(t) - jq_2(t))\} \ast r_s(t) \quad A4.5.2.6
\]

For the general case of \( n \) skywaves

\[
y_1(t-iT) = \{ t_s(t-iT)[q_1(t) - jq_1(t)] +
+ t_s(t-\tau_1 -iT)[q_2(t) - jq_2(t)] +
+ t_s(t-\tau_2 -iT)[q_3(t) - jq_3(t)] +
\}
\ast r_s(t) \quad A4.5.2.6a
\]

which is the time varying impulse response of the linear baseband channel. The waveform \( r(t) \) in Equation A4.5.2.2 is sampled once per data symbol \( s_n \) at the time instants \( t=iT \).
Thus, the sampling rate is 2400 bauds. The complex-valued sample of the demodulated baseband signal $r(t)$, at time $t=iT$ is

$$r_i = \sum_{h=-g}^{g} s_{i-h} y_{i,h} + w_i$$  \hspace{1cm} \text{A4.5.2.7}$$

where $y_{i,h} = y_{i-h}(hT)$ and $y_{i,h} = 0$ for $h<0$ and $h>g$ for practical purposes. The sequence of complex values given by the vector

$$Y_i = [y_{i,0}, y_{i,1}, \ldots, y_{i,g}]$$  \hspace{1cm} \text{A4.5.2.8}$$

is taken to be the 'sampled impulse response' at time $t=iT$ of the resultant linear baseband channel in Figure A4.4.12. It is evident from Equation A4.5.2.7 that the components $\{y_{i,h}\}$ of the 'sampled impulse response' $Y_i$ are associated with $g+1$ different data-symbols $\{s_{i,h}\}$ which implies that $Y_i$ is not in fact obtained by sampling the impulse response of the linear baseband channel in Figure A4.4.12. Nevertheless, $Y_i$ as defined by Equations A4.5.2.7 and A4.5.2.8 can clearly also be considered as a sequence of samples of the corresponding complex-valued waveform $y_i(t)$ where the latter is taken to be the value at the time $iT$ of the response of the channel to a unit impulse $\delta(t-iT+\tau)$ at time $t=iT-\tau$, whilst the observation instant $iT$ is fixed.

Let the actual sampled impulse response of the baseband channel be

$$z_i = [y_{i,0}, y_{i,1}, \ldots, y_{i,g}]$$  \hspace{1cm} \text{A4.5.2.9}$$

whose $z$-transform is

$$z_i(z) = y_{i,0} + y_{i,1}z^{-1} + \ldots + y_{i,g}z^{-g}$$  \hspace{1cm} \text{A4.5.2.10}$$

The $(g+1)$ components of the vector $Z_i$ are obtained by sampling the response of the baseband channel when it is excited by an impulse and are therefore the actual sampled
impulse response of the baseband channel. The relationship between \( \{Y_i\} \) (Equation A4.5.2.8) and \( \{Z_i\} \) (Equation A4.5.2.9) is clearly illustrated in Figure A4.4.14.

**A4.5.3 Model of the HF Data Transmission System used in Tests**

The data-transmission system considered here operates at 9600 bit/s over the HF radio link, the transmitted signal being a serial stream of 16-level QAM signal elements with a carrier frequency of 1800 Hz and a signal rate of 2400 bauds.

The impulse response of the linear baseband channel is given by Equation A4.5.2.6 for the case of two skywaves, and for the general case of \( n \) skywaves by Equation A4.5.2.6a. The characteristics of the overall equipment filters are shown in Figure 6.4.4.15. This consists of the practical filters [26,27] designed for a 9600 b/s modem in cascade with a radio filter Clansman VRC321 type (this being a typical radio filter used in a practical system). The practical filters are composed of the transmitter pre-modulation filter (lowpass), transmitter post-modulation filter (bandpass), receiver pre-modulation filter (bandpass) and receiver post-modulation filter (lowpass). The practical filters have been designed to have short duration and a minimum rise time. However, when the radio filter is included, the response is no longer minimum phase.

In the computer simulation tests, the vector \( Y_i \) (Equation A4.5.2.8) is obtained by sampling the \( \{y(t-iT)\} \) at a rate of 2400 samples/sec. Since the process is performed in the discrete time domain, and to avoid any aliasing likely to occur when any of the \( q_i(t) \) (Equations A4.5.2.6, A4.5.2.6a) is changing rapidly, the convolution in Equation A4.5.2.6a is carried out, with a sampling rate of 4800 samples/sec which is well above the Nyquist rate for filters \( T_x \) and \( R_x \). The functions \( q_i(t) \) are generated at a sampling rate of 4800 samples/sec, as described in Section A4.5.3.

Let the four sequences \( T_x1, T_x2, T_x3 \) and \( R_x \) represent the four impulse responses \( t_x(t), t_x(t-\tau_1), t_x(t-\tau_2) \) and \( r_x(t) \) respectively sampled at 4800 samples/sec.
\[ T \times 1 = [t_{x1,0} \ t_{x1,1} \cdots t_{x1,p}] \]
\[ T \times 2 = [t_{x2,0} \ t_{x2,1} \cdots t_{x2,p}] \quad \text{A 4.5.3.1} \]

and

\[ R \times = [r_{x1} \ r_{x1} \cdots r_{zp}] \quad \text{A 4.5.3.2} \]

where

\[ t_{x1,k} = t_x(kT/2) \]
\[ t_{x2,k} = t_x(kT/2 - \tau) \quad \text{A 4.5.3.3} \]
\[ r_{x,k} = r_x(kT/2) \]

\( p \) is an integer the value of which will become clear later on and \( 1/T \) is the data-symbol rate of 2400 symbols/sec.

For practical purposes (Figure A4.4.16)

\[ t_x(t) \equiv r_x(t) = 0, \]
\[ t < 0 \text{ and } t > (p - p') \frac{T}{2} \quad \text{A 4.5.3.4} \]

\( p' \) being an integer such that

\[ \tau = p' \frac{T}{2} + \tau' \quad \text{A 4.5.3.5} \]

and \( \tau' < T/2 \). This implies that
\[ t_{s1,k} = r_{s,k} = 0, \quad k < 0, \quad k > p - p' \quad \text{A 4.5.3.6} \]

\[ t_{s2,k} = 0, \quad k \leq p', \quad k > p \quad \text{A 4.5.3.7} \]

\( p - p' \) is a fixed quantity. The components \( q_i(t) \), are sampled at 4800 samples/second, and the corresponding samples upto time \( t = iT \) are given by the components of the six row vectors

\[
Q_{1,i} = [q_{1,0} \quad q_{1,1} \quad ... \quad q_{1,2}] \\
Q_{2,i} = [q_{2,0} \quad q_{2,1} \quad ... \quad q_{2,2}] \\
Q_{3,i} = [q_{3,0} \quad q_{3,1} \quad ... \quad q_{3,2}] \\
Q_{4,i} = [q_{4,0} \quad q_{4,1} \quad ... \quad q_{4,2}] 
\]

A4.5.3.8

Now, from Equation A4.5.2.6a it can be seen that \( y_i(t-iT) \) is the result of the discrete convolution between \( t_e(t-iT) \), modified by the time varying HF link with the receiver impulse response \( r_e(t) \), and similarly for \( t_e(t-iT) \). Thus Equation A4.5.3.8 can be written as

\[
y_{i,h} = \left( \frac{T}{2} \right) \sum_{k=0}^{2p} h[t_{s1,k} (q_{1,2(t-k)+h}) + t_{s2,k} (q_{3,2(t-k)+h} - jq_{4,2(t-k)+h})]r_{s2(h-k)} \quad \text{A 4.5.3.9}
\]

for \( h=0,1,...,g \). From Equations 6.4.6 and 6.4.7 it can be shown that [28]

\[
g = \frac{2p - p' + 1}{2} = \frac{2p_r + p' + 1}{2} \quad \text{A 4.5.3.10}
\]

Hence \( g \) is a function of \( p' \) and is therefore also a function of \( \tau \) (Equation A4.5.3.5).

For example for a delay \( \tau \) of 3 msec and \( p_r \) of 15 (Table A4.4.3) from Equation A4.5.3.5 \( p' = 14 \), \( \tau' = 0.4T/2 \) hence \( g = 22 \) (Equation A4.5.3.10).
Figure A4.4.6 Block Diagram of HF Ionospheric Channel Model

Figure A4.4.7. Rayleigh Fading introduced by one Skywave
Figure A4.4.8 Power Spectrum of $q_1(t)$ and $q_2(t)$
Figure A4.4.9 Characteristics of the Signals $q_i(t)$
Figure A4.4.10 Block Diagram of the filter used for generation of Fading samples

Table A4.4.3. Filter Tap values to obtain the required Frequency Spreads

<table>
<thead>
<tr>
<th>Fre. Spread (Hz)</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>-1.9</td>
<td>0.9031350</td>
<td>-1.9276</td>
<td>0.9316561</td>
<td>-0.946</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>0.8155376</td>
<td>-1.8524</td>
<td>0.8680474</td>
<td>-0.895</td>
</tr>
<tr>
<td>2</td>
<td>1.8036</td>
<td>0.6650064</td>
<td>-1.6954</td>
<td>0.7536390</td>
<td>-0.801</td>
</tr>
</tbody>
</table>

Figure A4.4.11 The linear Interpolation process
Figure A4.4.12 Model of the Digital Data Transmission System incorporating a 2 skywave HF radio link
Figure A4.4.13 Model of a Two Skywave HF Radio Link
Figure A4.4.14 Relationship between $\{Y_i\}$ and $\{Z_i\}$
Figure A4.4.15 Frequency Characteristics of the Practical Equipment and Radio Filters
Figure A4.4.16 Timing Relationship between the Transmitter Filter Impulse Response (real and imaginary) and its delayed version

<table>
<thead>
<tr>
<th>Sampled Impulse Response of Transmitter filter Tx1</th>
<th>Sampled Impulse Response of Receiver filter Rx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Part</td>
<td>Imaginary Part</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
</tr>
<tr>
<td>-0.1796</td>
<td>2.3539</td>
</tr>
<tr>
<td>-3.0770</td>
<td>20.7590</td>
</tr>
<tr>
<td>-9.9409</td>
<td>45.5585</td>
</tr>
<tr>
<td>-11.7869</td>
<td>41.4959</td>
</tr>
<tr>
<td>-3.4618</td>
<td>8.7046</td>
</tr>
<tr>
<td>4.4438</td>
<td>-11.7870</td>
</tr>
<tr>
<td>3.0642</td>
<td>-5.5819</td>
</tr>
<tr>
<td>-1.3596</td>
<td>3.1582</td>
</tr>
<tr>
<td>-1.4973</td>
<td>1.7365</td>
</tr>
<tr>
<td>0.2925</td>
<td>-0.7777</td>
</tr>
<tr>
<td>0.5180</td>
<td>-0.1293</td>
</tr>
<tr>
<td>-0.1842</td>
<td>0.2880</td>
</tr>
<tr>
<td>-0.3167</td>
<td>-0.2325</td>
</tr>
<tr>
<td>0.0022</td>
<td>-0.2107</td>
</tr>
<tr>
<td>0.0444</td>
<td>0.0392</td>
</tr>
<tr>
<td>0.0516</td>
<td>0.0098</td>
</tr>
</tbody>
</table>

Table A4.4.4 Sampled Impulse Response of the Minimum Phase Transmitter and Receiver Filters sampled at 4800 samples/second
### Table A4.4.5 Characteristics of the HF channel platform used for computer simulation tests

<table>
<thead>
<tr>
<th>CHANNEL NUMBER</th>
<th>NUMBER OF SKYWAVES</th>
<th>SKYWAVES DELAYS (ms)</th>
<th>FREQUENCY SPREAD (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1.2</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1.2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1.3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1.5,3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2.3</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1.1.5</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>1.5,2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>1.5,2,3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table A4.4.5 Characteristics of the HF channel platform used for computer simulation tests

### A4.6.1 QRN Root Finding Method

QR with Newton

#### Channel 1

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>$e=10^{-8}$ $L=5$</th>
<th>Iterations</th>
<th>$e=10^{-8}$ $L=6$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-301.30</td>
<td>4</td>
<td>-301.30</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-295.94</td>
<td>7</td>
<td>-295.94</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-254.20</td>
<td>-</td>
<td>-254.20</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Table QRN.1 Root Accuracy when applying the combined QR/Newton method over Channel 1

#### Channel 2

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>$e=10^{-8}$ $L=6$</th>
<th>Iterations</th>
<th>$e=10^{-8}$ $L=7$</th>
<th>Iterations</th>
<th>$e=10^{-8}$ $L=9$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-284.13</td>
<td>4</td>
<td>-284.13</td>
<td>4</td>
<td>-284.13</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-297.51</td>
<td>5</td>
<td>-301.30</td>
<td>4</td>
<td>-290.15</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-293.71</td>
<td>8</td>
<td>-291.48</td>
<td>6</td>
<td>-295.94</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>-299.91</td>
<td>-</td>
<td>F</td>
<td>-</td>
<td>-288.98</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Table QRN.2 Root Accuracy when applying the combined QR/Newton method over Channel 2

#### Channel 3

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>$e=10^{-8}$ $L=9$</th>
<th>Iterations</th>
<th>$e=10^{-8}$ $L=10$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-295.94</td>
<td>4</td>
<td>-301.30</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-297.51</td>
<td>7</td>
<td>-293.71</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>F</td>
<td>-</td>
<td>-299.91</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>F</td>
<td>-</td>
<td>-301.30</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Table QRN.3 Root Accuracy when applying the combined QR/Newton method over Channel 3
### Table QRN.4 Root Accuracy when applying the combined QR/Newton method over Channel 4

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>Iterations</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-290.67</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Root 2</td>
<td>-284.13</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Root 3</td>
<td>-283.53</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Root 4</td>
<td>-215.88</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Root 5</td>
<td>-215.75</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Root 6</td>
<td>-207.16</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Root 7</td>
<td>F</td>
<td>-</td>
<td>F</td>
</tr>
<tr>
<td>Root 8</td>
<td>F</td>
<td>-</td>
<td>F</td>
</tr>
</tbody>
</table>

Table QRN.4 Root Accuracy when applying the combined QR/Newton method over Channel 4

### Table QRN.5 Root Accuracy when applying the combined QR/Newton method over Channel 5

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-307.50</td>
<td>2</td>
</tr>
<tr>
<td>Root 2</td>
<td>-290.15</td>
<td>2</td>
</tr>
<tr>
<td>Root 3</td>
<td>-301.30</td>
<td>3</td>
</tr>
<tr>
<td>Root 4</td>
<td>-292.24</td>
<td>4</td>
</tr>
<tr>
<td>Root 5</td>
<td>-307.51</td>
<td>5</td>
</tr>
<tr>
<td>Root 6</td>
<td>-285.83</td>
<td>6</td>
</tr>
<tr>
<td>Root 7</td>
<td>-260.36</td>
<td>14</td>
</tr>
<tr>
<td>Root 8</td>
<td>-305.53</td>
<td>8</td>
</tr>
</tbody>
</table>

Table QRN.5 Root Accuracy when applying the combined QR/Newton method over Channel 5

### Table QRN.6 Number of Arithmetic operations required by the QR/Newton algorithm over Channels 1-5

<table>
<thead>
<tr>
<th>Operation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>1219</td>
<td>1431</td>
<td>2130</td>
<td>2418</td>
<td>4598</td>
</tr>
<tr>
<td>Multiplication</td>
<td>2477</td>
<td>4739</td>
<td>6711</td>
<td>16415</td>
<td>25557</td>
</tr>
<tr>
<td>Division</td>
<td>32</td>
<td>36</td>
<td>46</td>
<td>42</td>
<td>88</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>17492</td>
<td>31512</td>
<td>44802</td>
<td>103578</td>
<td>163066</td>
</tr>
</tbody>
</table>
A4.6.2 QRL Root Finding Method

QR with LAGUERRE

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-259.20</td>
</tr>
<tr>
<td>Root 2</td>
<td>-297.39</td>
</tr>
<tr>
<td>Root 3</td>
<td>-297.51</td>
</tr>
</tbody>
</table>

Table QRL.1 Root Accuracy when applying the combined QR/Laguerre method over Channel 1

<table>
<thead>
<tr>
<th>Operation</th>
<th>Laguerre</th>
<th>QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>627</td>
<td>2628</td>
</tr>
<tr>
<td>Multiplication</td>
<td>1198</td>
<td>2628</td>
</tr>
<tr>
<td>Division</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>8550</td>
<td>21769</td>
</tr>
</tbody>
</table>

Table QRL.2 Number of Arithmetic Operations when applying the combined QR/Laguerre method over Channel 1 (eps set to $10^{-4}$, L set to 6)

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-285.20</td>
</tr>
<tr>
<td>Root 2</td>
<td>-297.48</td>
</tr>
<tr>
<td>Root 3</td>
<td>-290.93</td>
</tr>
<tr>
<td>Root 4</td>
<td>F</td>
</tr>
</tbody>
</table>

Table QRL.3 Root Accuracy when applying the combined QR/Laguerre method over Channel 2 (eps set to $10^{-4}$, L set to 6,7)
### APPENDIX 4

#### Table QRL.4 Root Accuracy when applying the combined QR/Laguerre method over Channel 2
(eps set to $10^{-4}, 10^{-6}, 10^{-8}$, L set to 8)

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>10^{-4}</th>
<th>Iterations</th>
<th>10^{-6}</th>
<th>Iterations</th>
<th>10^{-8}</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-285.20</td>
<td>1</td>
<td>-285.21</td>
<td>2</td>
<td>-285.21</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-246.38</td>
<td>1</td>
<td>-300.96</td>
<td>2</td>
<td>-300.96</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-290.65</td>
<td>2</td>
<td>-290.66</td>
<td>3</td>
<td>-290.66</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>-294.60</td>
<td>3</td>
<td>-295.97</td>
<td>4</td>
<td>-295.97</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

#### Table QRL.5 Number of Arithmetic Operations when applying the combined QR/Laguerre method over Channel 2 (eps set to $10^{-4}, 10^{-6}, 10^{-8}$, L set to 8)

<table>
<thead>
<tr>
<th>Operation</th>
<th>10^{-4}</th>
<th>10^{-6}</th>
<th>10^{-8}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>6964</td>
<td>7225</td>
<td>7312</td>
</tr>
<tr>
<td>Multiplication</td>
<td>7630</td>
<td>8173</td>
<td>8354</td>
</tr>
<tr>
<td>Division</td>
<td>21</td>
<td>30</td>
<td>33</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>61161</td>
<td>64995</td>
<td>66273</td>
</tr>
</tbody>
</table>

#### Table QRL.6 Root Accuracy when applying the combined QR/Laguerre method over Channel 3
(eps set to $10^{-4}, 10^{-6}, 10^{-8}$, L set to 8)

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>10^{-4}</th>
<th>Iterations</th>
<th>10^{-6}</th>
<th>Iterations</th>
<th>10^{-8}</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-291.70</td>
<td>2</td>
<td>-291.70</td>
<td>2</td>
<td>-291.70</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-299.84</td>
<td>2</td>
<td>-299.84</td>
<td>3</td>
<td>-299.84</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-268.61</td>
<td>2</td>
<td>-292.39</td>
<td>3</td>
<td>-292.39</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>-290.68</td>
<td>3</td>
<td>-290.68</td>
<td>4</td>
<td>-290.68</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
### Table QRL.7 Number of Arithmetic Operations when applying the combined QR/Laguerre method over Channel 3 (eps set to $10^{-4}$, $10^{-6}$, $10^{-8}$, L set to 8)

<table>
<thead>
<tr>
<th>Operation</th>
<th>$10^{-4}$</th>
<th>$10^{-6}$</th>
<th>$10^{-8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>7306</td>
<td>7615</td>
<td>7615</td>
</tr>
<tr>
<td>Multiplication</td>
<td>8304</td>
<td>8943</td>
<td>8943</td>
</tr>
<tr>
<td>Division</td>
<td>27</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>65925</td>
<td>70431</td>
<td>70431</td>
</tr>
</tbody>
</table>

### Table QRL.8 Root Accuracy when applying the combined QR/Laguerre method over Channel 5 (eps set to $10^{-4}$, $10^{-6}$, $10^{-8}$, L set to 20)

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>$10^{-4}$</th>
<th>Iterations</th>
<th>$10^{-6}$</th>
<th>Iterations</th>
<th>$10^{-8}$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-299.11</td>
<td>1</td>
<td>-299.11</td>
<td>1</td>
<td>-299.11</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Root 2</td>
<td>-290.13</td>
<td>1</td>
<td>-290.13</td>
<td>1</td>
<td>-290.13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Root 3</td>
<td>-290.13</td>
<td>1</td>
<td>-293.37</td>
<td>1</td>
<td>-293.37</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Root 4</td>
<td>-306.29</td>
<td>2</td>
<td>-306.29</td>
<td>2</td>
<td>-306.29</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Root 5</td>
<td>-288.27</td>
<td>2</td>
<td>-288.27</td>
<td>2</td>
<td>-288.27</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 6</td>
<td>-290.77</td>
<td>2</td>
<td>-290.96</td>
<td>3</td>
<td>-290.76</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Root 7</td>
<td>-272.96</td>
<td>3</td>
<td>-283.95</td>
<td>4</td>
<td>-286.53</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Root 8</td>
<td>-242.22</td>
<td>3</td>
<td>-283.95</td>
<td>4</td>
<td>-283.95</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

### Table QRL.9 Number of Arithmetic Operations when applying the combined QR/Laguerre method over Channel 5 (eps set to $10^{-4}$, $10^{-6}$, $10^{-8}$, L set to 20)

<table>
<thead>
<tr>
<th>Operation</th>
<th>$10^{-4}$</th>
<th>$10^{-6}$</th>
<th>$10^{-8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>99478</td>
<td>99847</td>
<td>100090</td>
</tr>
<tr>
<td>Multiplication</td>
<td>101440</td>
<td>102200</td>
<td>102710</td>
</tr>
<tr>
<td>Division</td>
<td>45</td>
<td>54</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>816150</td>
<td>821490</td>
<td>825060</td>
</tr>
</tbody>
</table>
### Channel 6

<table>
<thead>
<tr>
<th>Root number</th>
<th>Root accuracy</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>-287.30</td>
<td>1</td>
</tr>
<tr>
<td>Root 2</td>
<td>-281.46</td>
<td>1</td>
</tr>
<tr>
<td>Root 3</td>
<td>-294.98</td>
<td>1</td>
</tr>
<tr>
<td>Root 4</td>
<td>-289.43</td>
<td>2</td>
</tr>
<tr>
<td>Root 5</td>
<td>-290.79</td>
<td>2</td>
</tr>
<tr>
<td>Root 6</td>
<td>-270.84</td>
<td>2</td>
</tr>
<tr>
<td>Root 7</td>
<td>-253.69</td>
<td>3</td>
</tr>
<tr>
<td>Root 8</td>
<td>-260.54</td>
<td>4</td>
</tr>
<tr>
<td>Root 9</td>
<td>-258.64</td>
<td>4</td>
</tr>
</tbody>
</table>

Table QRL.10 Root Accuracy when applying the combined QR/Laguerre method over Channel 6 (\(\epsilon\) set to \(10^{-4}\), \(L\) set to 8)

### Operation

<table>
<thead>
<tr>
<th>Operation</th>
<th>(10^{-4})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>8837</td>
</tr>
<tr>
<td>Multiplication</td>
<td>11290</td>
</tr>
<tr>
<td>Division</td>
<td>60</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>87101</td>
</tr>
</tbody>
</table>

Table QRL.11 Number of Arithmetic Operations when applying the combined QR/Laguerre method over Channel 6 (\(\epsilon\) set to \(10^{-4}\), \(L\) set to 8)
REFERENCES


APPENDIX 5 [A5]

A.5 Fixed-Point Arithmetic

In the following description, a 4-bit system will be used rather than a 16-bit word length. A bit word can represent the unsigned numbers 0-15, as shown in table A.5. The 4-bit unsigned numbers represent a modulo (mod) 16 system. If one (1) is added to the largest number (15), the operation wraps around to give 0 as the answer. A number wheel can graphically demonstrate the addition properties of a finite bit system. Figure A5.1.a shows a number wheel with the numbers 0-15 wrapped around the outside. To add any two numbers x and y in the range, find the first number x on the wheel, step off y units in the clockwise direction, thus yielding the answer x+y.

Negative numbers require a different interpretation of the numbers on the wheel (Figure A5.1.b). Now the right half of the circle represents the positive numbers and the left half the negative numbers. This representation is the two’s complement system. The negative numbers are the two’s complement of the positive numbers, and vice versa. A two’s complement binary integer,

\[ B = b_{n-1} \ldots b_1 b_0 \]  

is equivalent to the decimal integer

\[ I(B) = -b_{n-1}2^{n-1} + b_0 2^0 \]  

where \( b \)'s are binary digits. Notice that the sign bit has a negative weight, while all the others have positive weights. Rather than use the integer values just discussed, most DSP applications use a fractional fixed-point number which has values between +0.99... and -1. (Figure A5.1.c). To obtain the fractional n-bit number, the radix point must be moved n-1 places to the left. This leaves one sign bit plus n-1 fractional bits. The expression

\[ F(B) = -b_0 2^0 + b_1 2^{-1} + b_2 2^{-2} + \ldots + b_{n-1} 2^{-(n-1)} \]  

converts a binary fraction to a decimal fraction.
A5.1 Polynomial Approximation [1]

The polynomial approximation method is fundamentally very simple. A limited part of a function is approximated by a polynomial of some order sufficient to obtain the desired accuracy. The polynomial is generally a series of the form

$$P(n, x) = \sum_{i=0}^{n} a_i x^i \quad \text{A5.1.1}$$

where $x$ is the independent variable, $n$ is the polynomial order (a fixed integer), and $a_i$ is a set of $n+1$ fixed coefficients.

The desired function, say $f(x)$, is then approximated by a particular $P(n, x)$ such that

$$f(x) = P(n, x) + e(x), \quad x_1 < x < x_u \quad \text{A5.1.2}$$

where $x_1$ and $x_u$ are the limits of the domain of $x$, and $e(x)$ or $e(x)/f(x)$ is the error function which has been usually minimized in the min-max (equi-ripple) sense. This is done by selecting an appropriate means of calculating the coefficients $a_i$ [2].

A5.2 Approximation of the Inverse [1]

A special function for the computation of the inverse (Equation 5.3.11) is approximated by iteratively solving a particular non-linear equation say, $g(x)=0$. One means for solving this equation is by Newton-Raphson iteration. This can be understood by considering the Taylor series expansion for $g(x)$:

$$g(x + h) = g(x) + hg'(x) + r(x, h), \quad \text{A5.2.1}$$

where $r(x,h)$ is the remainder of the series (which can be assumed to be small), and $g'(x)$ is the derivative of the function $g(x)$. Leaving off the remainder in Equation A5.2.1 the approximation, in terms of incremental values of $x$, is

$$g(x_{i+1}) = g(x_i) + \{x_{i+1} - x_i\}g'(x_i) \quad \text{A5.2.2}$$
Solving for \( x_{i+1} \) in A5.2.2 with \( g(x_{i+1}) = 0 \) yields the approximation:

\[
x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)}
\]

Thus, \( x_{i+1} \) will converge to a solution of \( g(x) = 0 \). Convergence can be shown to be quadratic, i.e. the error in the approximation at each iteration is proportional to the square of the error in the previous iteration. Minimally, this requires a sufficiently close starting value for \( x(0) \) and the condition that \( |g'(x)| > 0 \) for all iterated values of \( x \).

For the computation of good approximations to the inverse \( 1/c \) in Equation 5.3.11 is that both \( g(x) \) and \( g'(x) \) must be derived and then the iteration of Equation ef. A5.2.3 must be used. This is complicated by the restriction that division should be avoided since the TMS320C30 has no divide instructions. The iteration to find the inverse is

\[
g(x_1) = \frac{1}{x_1} - c = 0,
\]

which is solved when \( 1/x = c \) or \( x = 1/c \). Taking the derivative of A5.2.4 and substituting into A5.2.3 and simplifying gives

\[
x_{i+1} = x_i \{2 - cx(i)\}
\]

which needs no division. Thus, A5.2.5 will converge to \( 1/c \) with the accuracy (in digits) for each iteration equal to twice that of the preceding one. Thus, if \( x(0) \) approximates \( 1/c \) to 3 bits of precision, only three iterations of A5.2.5 will yield about 24=3(2^3) bits of accuracy. The initial guess \( x(0) \), for the iterations of \( 1/c \), may be obtained using an interesting approximation. A TMS320C30 floating-point number \( c = (1+m)2^e \), where \( 0 < m < 1 \) and \(-127 < e < 127 \). The extra 1, added to the fractional mantissa \( m \), is the implied bit. Then the inverse of \( c \) can be written as

\[
\frac{1}{c} = \frac{1}{(1+m)2^e}
\]

An excellent approximation for the inverse of the mantissa is

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

which is exact at the end points \( m = 0 \) and \( m = 1 \).
Then the approximation for the reciprocal would be

\[ \frac{1}{c} = (1 - \frac{m}{2})2^{-c} \]
APPENDIX 5

A5.3 N I T (WITH LIMITED PRECISION)

In this Section the performance of the N I T algorithm is evaluated with limited precision over the three HF channels considered in Chapter 4 (Section 4.4.1 Part 11). These are Channel 1 (HF2) which is a two skywave HF channel with the second skywave delayed 3 ms (relatively to the first one) and frequency spread 2 Hz. Channel 2 (HF5) is a three skywave HF channel (with relative delays of 1 and 3 ms for the second and third skywaves respectively), and frequency spread of 2 Hz. Finally, Channel 3 (HF4) is a three skywave HF radio channel (with relative delays of 1 and 2 ms for the second and third skywaves respectively) and frequency spread of 2 Hz. The above considered channels are part of a platform of ten channels shown in Table A4.4.5 (Appendix 4 Section A4.5.3).

The N I T algorithm (along with Cepstrum and Root Finding Algorithms) have been applied over the derived sampled impulse responses of the three HF channels (see also Appendix 4 Sections A4.5.2 and A4.5.3), at different transmission instants (that have been chosen because they would present any adjustment algorithm with extremely unfavorable situations). The most potentially damaging scenario consists of a large number of roots lying very close (magnitude less than 1.05) to the unit circle (which is the case for the chosen transmission instants of the considered sampled impulse responses of the three channels).

The accuracy of N I T algorithm (along with that of Cepstrum and Root Finding methods), given by the values of ψ2 and ψ3, has been calculated first for different Signal-to-Noise Ratios (SNR) and then with limited precision arithmetic (different numbers of bits). Subsequently, the combined effect of different SNRs and limited precision has been calculated for the above considered algorithms and over the same HF channels. The results of these tests are presented in tabular form in Tables A5.3.1-A5.3.29. The number of roots outside the unit circle is indicated in each case by the superscripts in the tables as well as the magnitude of the root furthest away of the unit circle (which is a measure of the severity of the algorithm failure).
It becomes evident from Tables A5.3.1-29 that Cepstrum and N I T algorithms exhibit very close behaviour (a fact first noted in Chapter 4 Section 4.4.1.11).

N I T algorithm (as well as Cepstrum) is quite accurate and adjusts the prefilter even over the worst channels and under the most unfavorable conditions (i.e. low SNR and 16 bit precision) in all cases.

**CEPSTRUM**

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-20.35</td>
</tr>
<tr>
<td>16</td>
<td>-20.35</td>
</tr>
<tr>
<td>80</td>
<td>-20.35</td>
</tr>
<tr>
<td>60</td>
<td>-18.91</td>
</tr>
<tr>
<td>40</td>
<td>-15.73</td>
</tr>
<tr>
<td>20</td>
<td>-15.72</td>
</tr>
</tbody>
</table>

Table A5.3.1 Accuracy of Cepstrum, given by the values of \( \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 1 (HF2)

**N I T**

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-23.30</td>
</tr>
<tr>
<td>16</td>
<td>-23.30</td>
</tr>
<tr>
<td>80</td>
<td>-23.29</td>
</tr>
<tr>
<td>60</td>
<td>-23.16</td>
</tr>
<tr>
<td>40</td>
<td>-22.06</td>
</tr>
<tr>
<td>20</td>
<td>-9.41</td>
</tr>
</tbody>
</table>

Table A5.3.2 Accuracy of Cepstrum, given by the values of \( \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 1 (HF2)

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-23.16</td>
</tr>
<tr>
<td>16</td>
<td>-23.16</td>
</tr>
<tr>
<td>80</td>
<td>-22.06</td>
</tr>
<tr>
<td>60</td>
<td>-17.04</td>
</tr>
<tr>
<td>40</td>
<td>-9.42</td>
</tr>
<tr>
<td>20</td>
<td>-5.85</td>
</tr>
</tbody>
</table>

Table A5.3.3 Accuracy of N I T, given by the values of \( \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 1 (HF2)

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-23.16</td>
</tr>
<tr>
<td>16</td>
<td>-23.16</td>
</tr>
<tr>
<td>80</td>
<td>-22.06</td>
</tr>
<tr>
<td>60</td>
<td>-17.04</td>
</tr>
<tr>
<td>40</td>
<td>-9.42</td>
</tr>
<tr>
<td>20</td>
<td>-5.85</td>
</tr>
</tbody>
</table>

Table A5.3.4 Accuracy of N I T, given by the values of \( \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 1 (HF2)
CEPSTRUM

### CHANNEL 2 (HFS) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-26.61</td>
</tr>
<tr>
<td></td>
<td>-18.37</td>
</tr>
<tr>
<td></td>
<td>-26.34</td>
</tr>
<tr>
<td></td>
<td>-25.19^1 [1.02]</td>
</tr>
<tr>
<td></td>
<td>-24.72</td>
</tr>
<tr>
<td></td>
<td>-5.59</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-29.73</td>
</tr>
<tr>
<td></td>
<td>-18.42</td>
</tr>
<tr>
<td></td>
<td>-29.18</td>
</tr>
<tr>
<td></td>
<td>-26.76</td>
</tr>
<tr>
<td></td>
<td>-25.77</td>
</tr>
<tr>
<td></td>
<td>-5.61</td>
</tr>
</tbody>
</table>

Table A5.3.5 Accuracy of Cepstrum, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 2 (HFS)

### CHANNEL 2 (HFS) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-25.19</td>
</tr>
<tr>
<td></td>
<td>-17.87</td>
</tr>
<tr>
<td></td>
<td>-24.72</td>
</tr>
<tr>
<td></td>
<td>-15.28</td>
</tr>
<tr>
<td></td>
<td>-5.59</td>
</tr>
<tr>
<td></td>
<td>-4.31^1 [1.02]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-26.76</td>
</tr>
<tr>
<td></td>
<td>-18.10</td>
</tr>
<tr>
<td></td>
<td>-25.77</td>
</tr>
<tr>
<td></td>
<td>-15.41</td>
</tr>
<tr>
<td></td>
<td>-5.61</td>
</tr>
<tr>
<td></td>
<td>-4.32</td>
</tr>
</tbody>
</table>

Table A5.3.6 Accuracy of Cepstrum, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 2 (HFS)

ROOT FINDING (CLARK-HAU)

### CHANNEL 2 (HFS) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-30.55</td>
</tr>
<tr>
<td></td>
<td>-26.51^6 [1.02]</td>
</tr>
<tr>
<td></td>
<td>-30.00</td>
</tr>
<tr>
<td></td>
<td>-28.45</td>
</tr>
<tr>
<td></td>
<td>-24.67 [1.02]</td>
</tr>
<tr>
<td></td>
<td>-8.49^8 [1.2]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-30.55</td>
</tr>
<tr>
<td></td>
<td>-28.52</td>
</tr>
<tr>
<td></td>
<td>-30.00</td>
</tr>
<tr>
<td></td>
<td>-29.67</td>
</tr>
<tr>
<td></td>
<td>-29.12</td>
</tr>
<tr>
<td></td>
<td>-4.20</td>
</tr>
</tbody>
</table>

Table A5.3.7 Accuracy of Root Finding algorithms (any), given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 2 (HFS)

### CHANNEL 2 (HFS) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-28.45</td>
</tr>
<tr>
<td></td>
<td>-22.13^6 [1.03]</td>
</tr>
<tr>
<td></td>
<td>-24.67^4 [1.02]</td>
</tr>
<tr>
<td></td>
<td>-19.86^7 [1.05]</td>
</tr>
<tr>
<td></td>
<td>-8.49^8 [1.3]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-29.52</td>
</tr>
<tr>
<td></td>
<td>-27.58</td>
</tr>
<tr>
<td></td>
<td>-29.10</td>
</tr>
<tr>
<td></td>
<td>-25.64</td>
</tr>
<tr>
<td></td>
<td>-4.20</td>
</tr>
<tr>
<td></td>
<td>-4.52</td>
</tr>
</tbody>
</table>

Table A5.3.8 Accuracy of Root Finding algorithms (any), given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 2 (HFS)

N I T

### CHANNEL 2 (HFS) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-30.59</td>
</tr>
<tr>
<td></td>
<td>-22.90</td>
</tr>
<tr>
<td></td>
<td>-30.34</td>
</tr>
<tr>
<td></td>
<td>-28.81</td>
</tr>
<tr>
<td></td>
<td>-23.61</td>
</tr>
<tr>
<td></td>
<td>-6.18</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-30.59</td>
</tr>
<tr>
<td></td>
<td>-22.90</td>
</tr>
<tr>
<td></td>
<td>-30.34</td>
</tr>
<tr>
<td></td>
<td>-28.81</td>
</tr>
<tr>
<td></td>
<td>-23.61</td>
</tr>
<tr>
<td></td>
<td>-6.18</td>
</tr>
</tbody>
</table>

Table A5.3.9 Accuracy of N I T, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 2 (HFS)
### CHANNEL 2 (HFS) [2nd Transmission instant]

<table>
<thead>
<tr>
<th></th>
<th>60</th>
<th>40</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Psi_2$</td>
<td>32</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>-28.80</td>
<td>-19.61</td>
<td>-23.61</td>
</tr>
<tr>
<td>$\Psi_3$</td>
<td>32</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>-28.80</td>
<td>-19.61</td>
<td>-23.61</td>
</tr>
</tbody>
</table>

Table A5.3.10 Accuracy of N IT, given by the values of $\Psi_2$ and $\Psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 2 (HFS)

### CEPSTRUM

### CHANNEL 2 (HFS) [1st Transmission instant]

<table>
<thead>
<tr>
<th></th>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>$\Psi_2$</td>
<td>-29.49</td>
<td>-21.97</td>
</tr>
<tr>
<td>$\Psi_3$</td>
<td>-33.59</td>
<td>-21.51</td>
</tr>
</tbody>
</table>

Table A5.3.11 Accuracy of Cepstrum, given by the values of $\Psi_2$ and $\Psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 2 (HFS)

### ROOT FINDING (CLARK-HAU)

### CHANNEL 2 (HFS) [1st Transmission instant]

<table>
<thead>
<tr>
<th></th>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>$\Psi_2$</td>
<td>-15.24 $^{15}$ [1.04]</td>
<td>-15.22 $^{15}$</td>
</tr>
<tr>
<td>$\Psi_3$</td>
<td>-15.24</td>
<td>-15.22</td>
</tr>
</tbody>
</table>

Table A5.3.12 Accuracy of Root Finding algorithms (any), given by the values of $\Psi_2$ and $\Psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 2 (HFS)

### N IT

### CHANNEL 2 (HFS) [1st Transmission instant]

<table>
<thead>
<tr>
<th></th>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>$\Psi_3$</td>
<td>-32.35</td>
<td>-26.28</td>
</tr>
</tbody>
</table>

Table A5.3.14 Accuracy of N IT given by the values of $\Psi_2$ and $\Psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 2 (HFS)
### Table A5.3.15 Accuracy of NIT given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 2 (HF5)

<table>
<thead>
<tr>
<th>Signal/Noise (dB)</th>
<th>60</th>
<th>40</th>
<th>20</th>
<th>32</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
<td>-31.08</td>
<td>-25.69</td>
<td>-28.81</td>
<td>-22.43</td>
<td>-9.16</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-30.35</td>
<td>-25.11</td>
<td>-27.29</td>
<td>-22.43</td>
<td>-9.16</td>
</tr>
</tbody>
</table>

### CEPSTRUM

### Table A5.3.16 Accuracy of Cepstrum given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

<table>
<thead>
<tr>
<th>Signal/Noise (dB)</th>
<th>128</th>
<th>80</th>
<th>60</th>
<th>40</th>
<th>20</th>
<th>32</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
<td>-27.30</td>
<td>-23.72</td>
<td>-27.13</td>
<td>-26.65</td>
<td>-25.44</td>
<td>-18.70</td>
<td></td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-25.98</td>
<td>-22.76</td>
<td>-25.82</td>
<td>-25.37</td>
<td>-24.26</td>
<td>-18.76</td>
<td></td>
</tr>
</tbody>
</table>

### Table A5.3.17 Accuracy of Cepstrum given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 3 (HF4)

<table>
<thead>
<tr>
<th>Signal/Noise (dB)</th>
<th>60</th>
<th>40</th>
<th>20</th>
<th>32</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
<td>-26.65</td>
<td>-23.25</td>
<td>-25.44</td>
<td>-22.47</td>
<td>-18.71</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-25.37</td>
<td>-22.38</td>
<td>-24.26</td>
<td>-21.75</td>
<td>-18.76</td>
</tr>
</tbody>
</table>

### NIT

### Table A5.3.18 Accuracy of NIT given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

<table>
<thead>
<tr>
<th>Signal/Noise (dB)</th>
<th>60</th>
<th>40</th>
<th>20</th>
<th>32</th>
<th>16</th>
</tr>
</thead>
</table>

### Table A5.3.19 Accuracy of NIT given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 3 (HF4)

<table>
<thead>
<tr>
<th>Signal/Noise (dB)</th>
<th>60</th>
<th>40</th>
<th>20</th>
<th>32</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
<td>-28.52</td>
<td>-19.00</td>
<td>-28.05</td>
<td>-18.66</td>
<td>-21.42</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-28.52</td>
<td>-19.00</td>
<td>-28.05</td>
<td>-18.66</td>
<td>-21.42</td>
</tr>
</tbody>
</table>
### CEPSTRUM

#### CHANNEL 3 (HF4) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 16 80 60 40 20</td>
<td></td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-23.77 -21.90 [1.01] -23.61 -22.64 -21.47 [1.03] -11.66 [1.04]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-26.27 -22.80 -26.17 -24.17 -21.52 -10.44</td>
</tr>
</tbody>
</table>

Table A5.3.20 Accuracy of Cepstrum given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

#### CHANNEL 3 (HF4) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>60 40 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
</tr>
<tr>
<td>$\psi_3$</td>
</tr>
</tbody>
</table>

Table A5.3.21 Accuracy of Cepstrum, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

### ROOT FINDING (CLARK-HAU)

#### CHANNEL 3 (HF4) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 16 80 60 40 20</td>
<td></td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-12.80 -12.50 -12.70 -12.62 -12.30 -3.79 [1.09]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-12.80 -12.55 -12.79 -12.72 -12.49 -7.18</td>
</tr>
</tbody>
</table>

Table A5.3.22 Accuracy of Root Finding Algorithms, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 3 (HF4)

### N I T

#### CHANNEL 3 (HF4) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 16 80 60 40 20</td>
<td></td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-27.23 -22.58 -27.23 -26.70 -24.56 -12.33 [1.01]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-27.33 -22.58 -27.23 -26.70 -24.56 -12.33</td>
</tr>
</tbody>
</table>

Table A5.3.23 Accuracy of N I T, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

#### CHANNEL 3 (HF4) [2nd Transmission instant]

<table>
<thead>
<tr>
<th>60 40 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_2$</td>
</tr>
<tr>
<td>$\psi_3$</td>
</tr>
</tbody>
</table>

Table A5.3.24 Accuracy of N I T, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 3 (HF4)
## CEPSTRUM

### CHANNEL 3 (HF4) [3rd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-29.16</td>
</tr>
<tr>
<td></td>
<td>-29.24^1 [1.02]</td>
</tr>
<tr>
<td></td>
<td>-29.14</td>
</tr>
<tr>
<td></td>
<td>-29.00</td>
</tr>
<tr>
<td></td>
<td>-28.99</td>
</tr>
<tr>
<td></td>
<td>-8.88^2 [1.02]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-21.59</td>
</tr>
<tr>
<td></td>
<td>-17.82</td>
</tr>
<tr>
<td></td>
<td>-21.50</td>
</tr>
<tr>
<td></td>
<td>-21.18</td>
</tr>
<tr>
<td></td>
<td>-20.10</td>
</tr>
<tr>
<td></td>
<td>-9.29</td>
</tr>
</tbody>
</table>

Table A5.3.25 Accuracy of Cepstrum, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

### CHANNEL 3 (HF4) [3rd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-29.77</td>
</tr>
<tr>
<td></td>
<td>-29.33^1 [1.01]</td>
</tr>
<tr>
<td></td>
<td>-28.99</td>
</tr>
<tr>
<td></td>
<td>-28.28</td>
</tr>
<tr>
<td></td>
<td>-8.88^2 [1.02]</td>
</tr>
<tr>
<td></td>
<td>-8.60^3 [1.02]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-21.18</td>
</tr>
<tr>
<td></td>
<td>-17.50</td>
</tr>
<tr>
<td></td>
<td>-20.00</td>
</tr>
<tr>
<td></td>
<td>-16.57</td>
</tr>
<tr>
<td></td>
<td>-9.26</td>
</tr>
<tr>
<td></td>
<td>-7.71</td>
</tr>
</tbody>
</table>

Table A5.3.26 Accuracy of Cepstrum, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

## ROOT FINDING (CLARK-HAU)

### CHANNEL 3 (HF4) [3rd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-9.21</td>
</tr>
<tr>
<td></td>
<td>-9.16</td>
</tr>
<tr>
<td></td>
<td>-9.20^15 [1.04]</td>
</tr>
<tr>
<td></td>
<td>-9.20^15 [1.04]</td>
</tr>
<tr>
<td></td>
<td>-9.17^15 [1.04]</td>
</tr>
<tr>
<td></td>
<td>-7.85^13 [1.3]</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-9.21</td>
</tr>
<tr>
<td></td>
<td>-9.16</td>
</tr>
<tr>
<td></td>
<td>-9.20</td>
</tr>
<tr>
<td></td>
<td>-9.20</td>
</tr>
<tr>
<td></td>
<td>-9.17</td>
</tr>
<tr>
<td></td>
<td>-7.85</td>
</tr>
</tbody>
</table>

Table A5.3.27 Accuracy of Root Finding algorithms (any), given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)

## NIT

### CHANNEL 3 (HF4) [3rd Transmission instant]

<table>
<thead>
<tr>
<th>Number of Bits</th>
<th>Signal/Noise (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>-30.16</td>
</tr>
<tr>
<td></td>
<td>-6.82</td>
</tr>
<tr>
<td></td>
<td>-30.35</td>
</tr>
<tr>
<td></td>
<td>-30.31</td>
</tr>
<tr>
<td></td>
<td>-27.18</td>
</tr>
<tr>
<td></td>
<td>-9.35</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>-30.16</td>
</tr>
<tr>
<td></td>
<td>-6.82</td>
</tr>
<tr>
<td></td>
<td>-30.35</td>
</tr>
<tr>
<td></td>
<td>-30.31</td>
</tr>
<tr>
<td></td>
<td>-27.18</td>
</tr>
<tr>
<td></td>
<td>-9.35</td>
</tr>
</tbody>
</table>

Table A5.3.28 Accuracy of NIT, given by the values of $\psi_2$ and $\psi_3$, for different Signal-to-Noise Ratios and Limited Precision Arithmetic over Channel 3 (HF4)
### A5.4 N I T implementation on a TMS320C25-30 \[3,6,1\]

The N I T algorithm consists basically of two parts that implement convolution and deconvolution. These two basic building blocks can be viewed as Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters respectively. The FIR filter is defined by

\[
y(n) = \sum_{i=1}^{N} h(i)x(n-i)
\]

where \( y(n) \) is the output sample at time \( n \), \( h(i) \) is the \( i \)th coefficient of weighting factor, and \( x(n-i) \) is the \((n-i)\)th input sample. From equation 5.3.1.1 it can be seen that an FIR filter has, as the name implies a finite-length response to a unit sample.

Denoting the \( z \) transforms of \( x(n) \), \( y(n) \), and \( h(n) \) as \( X(z) \), \( Y(z) \), and \( H(z) \) respectively, then

\[
H(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{N} h(i)z^{-N}
\]

Equations 5.3.1.1-2 lead to direct form realisation of an FIR filter and can be implemented in a straightforward and efficient manner. In particular, equation 5.3.1.1 is the general form of an FIR filter [4] as well as the convolution of two sequences of numbers \( a(i) \) and \( x(i) \) [5].

The evaluation of equation 5.3.1.1 is computation-intensive task but the TMS320C25 reduces the execution time of all filters (FIR and IIR) by virtue of its 100-ns instruction cycle time and optimised instructions for filter operations. As implied earlier by

---

**Table A5.3.29** Accuracy of N I T, given by the values of \( \psi_2 \) and \( \psi_3 \), for different Signal-to-Noise Ratios combined with Limited Precision Arithmetic over Channel 3 (HF4)

<table>
<thead>
<tr>
<th></th>
<th>60</th>
<th>16</th>
<th>40</th>
<th>16</th>
<th>20</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>-30.30</td>
<td>-20.96</td>
<td>-27.18</td>
<td>-18.96</td>
<td>-9.36</td>
<td>-7.66</td>
<td>-7.66</td>
</tr>
</tbody>
</table>
equation 5.3.1.1 the FIR filter is simply the sum of products in a sampled data system.
A simple implementation of such a filter uses the MACD each filter tap and the
RPT/RPTK instruction to repeat the MACD for each tap. Thus, a 64 tap filter can be
implemented as

\[
\text{RPTK 63} \\
\text{MACD *}, \text{COEFFP}
\]

The coefficients can be stored anywhere in program memory (in the reconfigurable on-chip RAM, in the on-chip ROM, or in external memories). When the coefficients are stored in on-chip ROM or externally, the entire on-chip data RAM can be used to store the sample sequence. This allows filters of up to 512 taps to be implemented.

Execution of the filter will be at full speed, or 100 ns per tap, as the memory (either on-chip RAM or high-speed external RAM) supports full-speed execution. The deconvolution part of the algorithm can be implemented as an IIR filter. Now, equation 5.3.1.1 can be extended to implement an IIR filter. In particular, the following difference equation

\[
y(n) = \sum_{i=1}^{N} a_i y(n-i) + \sum_{i=1}^{M} b_i x(n-i) \quad 5.4.1.3
\]

shows that the output of the filter is a weighted sum of past values of the input to the filter and of the output of the filter. Using techniques similar to those for an FIR filter, the realisation implied by equation 5.3.1.3 can lead to very efficient implementations of the filter on a TMS320C25. It has been shown [4] that the z transform of the unit sample response of an IIR filter corresponding to 5.3.1.3 is

\[
H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 - \sum_{i=1}^{N} a_i z^{-i}} \quad 5.4.1.4
\]

where \( H(z) \), \( Y(z) \), and \( X(z) \) are the z transforms of \( h(n) \), \( y(n) \), and \( x(n) \), respectively. By far the most commonly used IIR structure is the cascade-form realisation since it results in less sensitivity to quantisation noise. Now, equation 5.3.1.4 can be written in the equivalent form
\[ H(z) = \prod_{i=1}^{N/2} \frac{\beta_{0i} + \beta_{1i}z^{-1} + \beta_{2i}z^{-2}}{1 - a_{1i}z^{-1} - a_{2i}z^{-2}} \]  

5.4.1.5

where the filter is realised as a series of biquads. It can be seen that the transfer function of the IIR filters contains both poles and zeros, and its output depends on both the input and the past output.

However, in order to avoid the division operation (which is performed in software, thus requiring an increased number of cycles), the deconvolution part of the algorithm (i.e. the division of two polynomials) can be realised in essence as convolution assuming that the inverse of the first term of the denominator polynomial is available.

In the case of the TMS320C30 digital filtering (with either fixed or adaptive coefficients), can be implemented very efficiently. In the case of the FIR filters the two features that facilitate their implementation on a 320C30 are: parallel/add operations and circular addressing. The first feature permits a multiplication and an addition to execute in one machine cycle, while the second makes a finite buffer of length \( N \) sufficient for the data \( x[n] \). Note that the filter takes one cycle of execution per tap.

The C code that implements the IIR filter consisting of \( N \) biquads (with \( a1[i] \), \( a2[i] \) numerator coefficients of the \( i \)th biquad and \( b0[i] \), \( b1[i] \), \( b2[i] \) the denominator coefficients of the same biquad) is

\[
\begin{align*}
y[0,n] &= x[n]; \\
\text{for } (i=0; i<N; i++) \\n\text{d}[i,n] &= a2[i]*d[i,n-2]+a1[i]*d[i,n-1]+y[i-1,n]; \\
y[i,n] &= b2[i]*d[i,N-2]+b1[i]*d[i,N-1]+b0[i]*d[i,n]; \\
\end{align*}
\]

\[ y[n]=y[N-1,n]; \]

The code for the TMS320C30 implementation is given in the Section labelled PROGRAMS of this Appendix.

The time that the N I T algorithm takes when running with full precision on a TMS320C25/30 can be easily calculated by utilising Table A5.4.1 which show the number of floating point operations each time the algorithm operates over a particular channel.
Table A5.4.2 shows the 'calculated' timing of the algorithm over Channels 1-3, when running with limited precision (16 bits). Tables A5.4.3 show the 'actual' timing of the algorithm when implemented on a TMS320C25. The actual timing of the N I T algorithm is less than the simulated perhaps due to the very efficient filtering implementations available for both the TMS320C25 and TMS320C30.

<table>
<thead>
<tr>
<th>Operation</th>
<th>HF CHANNELS (Channels 1-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition &amp; Subtraction</td>
<td>3800</td>
</tr>
<tr>
<td>Multiplication</td>
<td>7600</td>
</tr>
<tr>
<td>Division</td>
<td>20</td>
</tr>
<tr>
<td>Total Operational Count</td>
<td>53320</td>
</tr>
</tbody>
</table>

Table A5.4.1 Number of Arithmetic Operations when applying the N I T algorithm over HF Channels (Channels 1-3)

<table>
<thead>
<tr>
<th>Simulated</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF CHANNELS (Channels 1-3)</td>
</tr>
<tr>
<td>TMS320C25</td>
</tr>
<tr>
<td>5.33</td>
</tr>
</tbody>
</table>

Table A5.4.2 Calculated speed (ms) of the N I T algorithm for TMS320C25 and TMS320C30 implementations

<table>
<thead>
<tr>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF CHANNELS (Channels 1-3)</td>
</tr>
<tr>
<td>TMS320C25</td>
</tr>
<tr>
<td>2.64</td>
</tr>
</tbody>
</table>

Table A5.4.3 Actual speed (ms) of the N I T algorithm for TMS320C25 and TMS320C30 implementations
REFERENCES


%THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF TELEPHONE CHANNEL 1;

%Attenuation characteristic of telephone circuit 1 every 100 Hz;
ATC1=[27 11 7 5 3 2 1.3 1 0.5 0.3 0.2 0 0 0 0.6 1.3 1.5 2.2 5 2.6 3 3.9
4.6 5 5.3 6 6 7 7.3 8 8.6 9.3 10 10.3 10.6 11];

%Attenuation characteristic of telephone circuit 1, sampled now every 50 Hz;
ATC1=interp(ATC1,2);

%Attenuation characteristic of telephone circuit 1, sampled now every 25 Hz, to avoid aliasing;
ATC1=interp(ATC1,2);
ATC1=[30 ATC1 11.5];

%Group delay characteristic of telephone circuit 1, sampled now every 100 Hz;
GDC1=[9.2 2.6 1.2 0.7 0.4 0.3 0.1 zeros(8:13) 0.1
0.1 0.15 0.2 0.2 0.3 0.33 0.4 0.4 0.5 0.6
0.8 1 1.3 1.6 1.8 2.2 2.5 3 3.3 3.8 4.3
4.8];

%Group delay characteristic of telephone circuit 1, sampled now every 50 Hz;
GDC1=interp(GDC1,2);

%Group delay characteristic of telephone circuit 1, sampled now every 25 Hz, to avoid aliasing;
GDC1=interp(GDC1,2);
GDC1=[15 GDC1 5];

%Attenuation characteristic of filter set 1;
ATF1=[98 66.7 44.4 30 18.9 12.2 6.7 3.3 1.1 1.1 zeros(11:23)
1.1 2 2.2 4.4 6.6 11.1 14.4 20 26.7 36.7 47.8 61.1 76.7 91.1];

ATF1=interp(ATF1,2);
ATF1=interp(ATF1,2);
ATF1=[100 ATF1 91.5];
PROGRAMS

%group delay characteristic of filter set 1:

GDF1=[1.7 2.3 2.7 3.04 3.26 3.4 3.48 3.42 3.37 3.26 3.21 3.1 2.99 2.93 2.88 2.88 2.85 2.88 2.91 2.91 2.96 2.99 3.1 3.2 3.2 3.37 3.42 3.48 3.5 3.48 3.42 3.21 2.88 2.34 1.63 0.98];

GDF1=interp(GDF1,2);
GDF1=interp(GDF1,2);
GDF1=[1.6 GDF1 0.99];
AT1=(ATC1+ATF1)/2;
GD1 =(GDC1 +GDF1)/2;

for l=1:max(size(AT1))
    A(l)=10.^(-abs(AT1(l))/20);
end;

DF=angle(GD1);
for l=2:max(size(GD1))
    PHI(l)=PHI(l-1)-pi/2*ABS((GD1(l)+GD1(l-1))*10.^(-3));
end;

R=A.*cos(PHI);
I=A.*sin(PHI);
Y=R+i*I;
YY=[Y zeros(1:874)];
y=ifft(YY);

TELCHN2.M

%THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF TELEPHONE CHANNEL 2;

%attenuation characteristic of telephone circuit 2 every 100 hz;

ATC2=[40 11 7.8 4.4 2.2 2.2 1.1 1 zeros(9:18) 1 1.1 1.1 1.1 1.7 1.8 2.2 2.2 2.8 3.3 4.4 4.4 5.5 8.8 11 14.4 18.9 24.4 32.2];

%attenuation characteristic of telephone circuit 2, sampled now every 50 hz;

ATC2=interp(ATC2,2);

%attenuation characteristic of telephone circuit 2, sampled now every 25 hz, to avoid aliasing;
ATC2=interp(ATC2,2);
ATC2=[85 ATC2 37];

% group delay characteristic of telephone circuit 2, sampled every 100 hz;

GDC2=[9.8 7.5 5.7 4.4 3.3 2.4 1.8 1.3 1 0.7 0.65
  0.5 0.5 0.32 0.2 0.16 zeros(17:25) 0.16 0.16
  0.16 0.2 0.3 0.4 0.49 0.82 1.14 1.6 2.4 3.1];

% group delay characteristic of telephone circuit 2, sampled now every 50 hz;
GDC2=interp(GDC2,2);

% group delay characteristic of telephone circuit 2, sampled now every 25 hz, to avoid aliasing;
GDC2=interp(GDC2,2);
GDC2=[11 GDC2 4];

% attenuation characteristic of filter set 1;
ATF1=[98 66.7 44.4 30 18.9 12.2 6.7 3.3 1.1 1.1
  zeros(11:23) 1.1 2.2 4.4 6.6 11.1 14.4 20
  26.7 36.7 47.8 61.1 76.7 91.1];

ATF1=interp(ATF1,2);
ATF1=interp(ATF1,2);
ATF1=[100 ATF1 91.5];

% group delay characteristic of filter set 1;
GDF1=[1.7 2.3 2.7 3.04 3.26 3.4 3.48 3.42
  3.37 3.26 3.21 3.1 2.99 2.93 2.88 2.88 2.85
  2.88 2.91 2.91 2.91 2.96 2.99 3.1 3.2 3.26
  3.37 3.42 3.48 3.5 3.48 3.42 3.21 2.88 2.34
  1.63 0.98];

GDF1=interp(GDF1,2);
GDF1=interp(GDF1,2);
GDF1=[1.6 GDF1 0.99];
AT2=(ATC2+ATF1)/2;
GD2=(GDC2+GDF1)/2;

for l=1:max(size(AT2))
  A(l)=10.^(-abs(AT2(l))/20);
end;

DF=25;
PHI=angle(GD2);

for l=2:max(size(GD2))
  PHI(l)=PHI(l-1)-pi*DF*((GD2(l)+GD2(l-1))*10^(-3));
end;

R=A.*cos(PHI);
l=A.*sin(PHI);
Y=R+i*l;
YY=[Y zeros(1:874)];
y=ifft(YY);

% TELCHN3.M

% THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF TELEPHONE CHANNEL 3;

% attenuation characteristic of telephone circuit 3 every 100 hz;
ATC3=[40 9.9 5.5 2.7 1.1 zeros(6:10) 1 1.2 2.2
     2.7 3.8 4.9 6.6 7.7 8.8 9.9 11 12 14.3 16.5
     17.6 19.8 20.9 22.5 23.6 25.3 26.4 28.6 29.7
     30.8 31.3 32.4 33];

% attenuation characteristic of telephone circuit 3, sampled now every 50 hz;
ATC3=interp(ATC3,2);

% attenuation characteristic of telephone circuit 3, sampled now every 25 hz, to avoid aliasing;
ATC3=interp(ATC3,2);
ATC3=[85 ATC3 35];

% group delay characteristic of telephone circuit 3, sampled every 100 hz;
GDC3=[19.2 12.4 8 5.5 3.8 2.9 2.1 1.5 1.1 0.9
     0.8 0.7 0.4 0.3 0.27 0.2 0.1 zeros(18:22) 0.3
     0.4 0.55 0.58 0.82 1.1 1.4 1.8 2.3 3.2 4.6 6 8.2
     10.2 12.4];

% group delay characteristic of telephone circuit 3, sampled now every 50 hz
GDC3=interp(GDC3,2);

% group delay characteristic of telephone circuit 3, sampled now every 25 hz, to avoid aliasing;
GDC3=interp(GDC3,2);
GDC3=[24 GDC3 13];

% attenuation characteristic of filter set 1;
ATF1=[98.667 44.4 30 18.9 12.2 6.7 3.3 1.1
1.1 zeros(11:23) 1.1 2.2 4.4 6.6 11.1 14.4
20 26.7 36.7 47.8 61.1 76.7 91.1];

ATF1=interp(ATF1,2);
ATF1=interp(ATF1,2);
ATF1=[100 ATF1 91.5];

% group delay characteristic of filter set 1;

GDF1=[1.7 2.3 2.7 3.04 3.26 3.4 3.48 3.42 3.37
2.68 3.21 3.1 2.99 2.93 2.88 2.88 2.85
2.88 2.91 2.91 2.91 2.96 2.99 3.1 3.2
3.26 3.37 3.42 3.48 3.5 3.48 3.42 3.21
2.88 2.34 1.63 0.98];

GDF1=interp(GDF1,2);
GDF1=interp(GDF1,2);

GDF1=[1.6 GDF1 0.99];
AT3=(ATC3+ATF1)/2;
GD3=(GDC3+GDF1)/2;

for l=1:max(size(AT3))
    A(l)=10.^(-abs(AT3(l))/20);
end;

DF=25;
PHI=angle(GD3);

for l=2:max(size(GD3))
    PHI(l)=PHI(l-1)-pi*DF*((GD3(l))+
    +GD3(l-1))*10^(-3));
end;

R=A.*cos(PHI);
l=A.*sin(PHI);
Y=R+i*l;
YY=[Y zeros(1:874)];
y=ifft(YY);
end


TELCN4.M
% THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF
TELEPHONE CHANNEL 4;

% attenuation characteristic of telephone circuit 4 every 100 hz;
ATC4=[22 11 6.6 4.4 1.5 1 0 0 0.5 1 2.7 3.8 5.5 7.7
9.9 12 15 17.6 18.7 22 25 28.6 31.9 35.7
38.5 42.9 47.3 51.6 55 58.2 62.6 67 71.4
73.6 80.2 83.5];

% attenuation characteristic of telephone circuit 4, sampled now every 50 hz;
ATC4=interp(ATC4,2);

% attenuation characteristic of telephone circuit 4, sampled now every 25 hz, to avoid aliasing;
ATC4=interp(ATC4,2);
ATC4=[30,ATC4 82];

% group delay characteristic of telephone circuit 4, sampled every 100 hz;
GDC4=[4.1 3.6 2 1.3 1 0.66 0.41 0.2 0.1 zeros(10:15)
0.16 0.32 0.49 0.81 1.1 1.3 1.8 2.3 3 3.8 4.6
5.4 6.3 7.1 8 8.9 9.6 10.5 11.4 12.2 13 13.7];

% group delay characteristic of telephone circuit 2, sampled now every 50 hz;
GDC4=interp(GDC4,2);

% group delay characteristic of telephone circuit 4, sampled now every 25 hz, to avoid aliasing;
GDC4=interp(GDC4,2);
GDC4=[5 GDC4 14];

% attenuation characteristic of filter set 1;
ATF1=[98 66.7 44.4 30 18.9 12.2 6.7 3.3 1.1 1.1
zeros(11:23) 1.1 2 2.2 4.4 6.6 11.1 14.4
20 26.7 36.7 47.8 61.1 76.7 91.1];

ATF1=interp(ATF1,2);
ATF1=interp(ATF1,2);
ATF1=[100 ATF1 91.5];

% group delay characteristic of filter set 1;
GDF1=[1.7 2.3 2.7 3.04 3.26 3.4 3.48 3.42 3.37
3.26 3.21 3.1 2.99 2.93 2.88 2.88 2.85
2.88 2.91 2.91 2.91 2.96 2.99 3.1 3.2
3.26 3.37 3.42 3.48 3.5 3.48 3.42 3.21
2.88 2.34 1.63 0.98];

GDF1=interp(GDF1,2);
GDF1 = interp(GDF1, 2);
GDF1 = [1.6 GDF1 0.99];
AT4 = (ATC4 + ATF1) / 2;
GD4 = (GDC4 + GDF1) / 2;

for l = 1:max(size(AT4))
    A(l) = 10.^(abs(AT4(l)) / 20);
end;

DF = 25;
PHI = angle(GD4);

for l = 2:max(size(GD4))
    PHI(l) = PHI(l-1) - pi*DF*((GD4(l) + +GD4(l-1))^10^(-3));
end;

R = A.*cos(PHI);
I = A.*sin(PHI);
Y = R + i*I;
YY = [Y zeros(1:874)];
y = ifft(YY);

% TELCHN4.M
%
% THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF TELEPHONE CHANNEL 5;
%
% attenuation characteristic of telephone circuit 5 every 100 hz;

ATC5 = [27 11 7 5 3 2 1.3 1 0.5 0.3 0.2 0 0 0 0.6 1.3 1.5 2.2 2.5 2.6 3 3.9 4.6 5.3 6 6.7
         7.3 8 8.6 9.3 10 10.3 10.6 11];
%
% attenuation characteristic of telephone circuit 5, sampled now every 50 hz;

ATC5 = interp(ATC5, 2);

% attenuation characteristic of telephone circuit 5, sampled now every 25 hz, to avoid aliasing;

ATC5 = interp(ATC5, 2);
ATC5 = [30 ATC5 11.5];
%
% group delay characteristic of telephone circuit 5, sampled every 100 hz;
GDC5=[9.2 2.6 1.2 0.7 0.4 0.3 0.1 zeros(8:13)
0.1 0.1 0.15 0.2 0.2 0.3 0.33 0.4 0.4
0.5 0.6 0.8 1 1 1.3 1.6 1.8 2 2.2 2.5 3 3.3
3.8 4.3 4.8];

%group delay characteristic of telephone circuit 5, sampled now every 50 hz;
GDC5=interp(GDC5,2);

%group delay characteristic of telephone circuit 5, sampled now every 25 hz, to avoid aliasing;
GDC5=interp(GDC5,2);
GDC5=[15 GDC5 5];

%attenuation characteristic of filter set 2;
ATF2=[66.7 32 20 12 6.7 2.7 1.3 zeros(8:28) 1 4 8 17.3 32
46.7 73.3 100 106.7];
ATF2=interp(ATF2,2);
ATF2=interp(ATF2,2);
ATF2=[95 ATF2 110];

%group delay characteristic of filter set 2;
GDF2=[1.3 2.1 1.5 1.4 1.4 1.25 1.1 0.9 0.76
0.65 0.54 0.43 0.38 0.38 0.39 0.38 0.38 0.35
0.34 0.38 0.43 0.46 0.49 0.65 0.76 0.92 1.14
1.36 1.63 1.9 2.34 2.17 1.9 1.58 0.98 0.6 0.38];
GDF2=interp(GDF2,2);
GDF2=interp(GDF2,2);
GDF2=[1 GDF2 0.5];
ATF5=(ATC5+ATF2)/2;
GD5=(GDC5+GDF2)/2;

for l=1:max(size(AT5))
A(l)=10^(-abs(AT5(l))/20);
end;

DF=25;
PHI=angle(GD5);

for l=2:max(size(GD5))
PHI(l)=PHI(l-1)+pi*DF*((GD5(l)+
+GD5(l-1))*10^(-3));
end;
R=A.*cos(PHI);
l=A.*sin(PHI);
Y=R+i*l;
YY=[Y zeros(1:874)];
y=ifft(YY);

TELCHN6.M

%THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF TELEPHONE CHANNEL 6;

%attenuation characteristic of telephone circuit 6 every 100 hz;

ATC6=[40 11 7.8 4.4 2.2 2.2 1.1 1 zeros(9:18) 1 1.1 1.1 1.1 1.7 1.8 2.2 2.2 2.8 3.3 4.4 4.4 5.5 8.8 11 14.4 18.9 24.4 32.2];

%attenuation characteristic of telephone circuit 6, sampled now
% every 50 hz;

ATC6=interp(ATC6,2);

%attenuation characteristic of telephone circuit 6, sampled now every 25 hz, to avoid aliasing;

ATC6=interp(ATC6,2);
ATC6=[85 ATC6 37];

%group delay characteristic of telephone circuit 6, sampled every 100 hz;

GDC6=[9.8 7.5 5.7 4.4 3.3 2.4 1.8 1.3 1 0.7 0.65 0.5 0.5 .32 0.2 0.16 zeros(17:25) 0.16 0.16 0.16 0.2 0.3 0.4 0.49 0.82 1.14 1.6 2.4 3.1];

%group delay characteristic of telephone circuit 6, sampled now every 50 hz;

GDC6=interp(GDC6,2);

%group delay characteristic of telephone circuit 6, sampled now every 25 hz, to avoid aliasing;

GDC6=interp(GDC6,2);
GDC6=[11 GDC6 4];

%attenuation characteristic of filter set 2;
PROGRAMS

ATF2=[66.7 32 20 12 6.7 2.7 1.3 zeros(8:28) 1 4 8 17.3 32 46.7 73.3 100 106.7];

ATF2=interp(ATF2,2);
ATF2=interp(ATF2,2);
ATF2=[95 ATF2 110];

% group delay characteristic of filter set 2;

GDF2=[1.3 2.1 1.5 1.4 1.4 1.25 1.1 0.9 0.76
     0.65 0.54 0.43 0.39 0.38 0.38 0.35
     0.34 0.38 0.43 0.46 0.49 0.65 0.76 0.92 1.14
     1.36 1.63 1.9 2.34 2.17 1.9 1.58 0.98 0.6 0.38];

GDF2=interp(GDF2,2);
GDF2=interp(GDF2,2);
GDF2=[1 GDF2 0.5];
AT6=(ATC6+ATF2)/2;
GD6=(GDC6+GDF2)/2;

for i=1:max(size(AT6))
    A(i)=10.^(1-abs(AT6(i))/20);
end;

DF=25;
PHI=angle(GD6);
for i=2:max(size(GD6))
    PHI(i)=PHI(i-1)-pi*DF*((GD6(i)+GD6(i-1))*10^(-3));
end;
R=A.*cos(PHI);
l=A.*sin(PHI);
Y=R+i*l;
YY=[Y zeros(1:874)];
y=ifft(YY);

TELCHN7.M

% THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF TELEPHONE CHANNEL 7;

% attenuation characteristic of telephone circuit 7 every 100 hz;

ATC7=[40 9.9 5.5 2.7 1.1 zeros(6:10) 1 1.2 2.2 2.7 3.8 4.9 6.6
     7.7 8.8 9.9 11 12 14.3 16.5 17.6 19.8 20.9 22.5 23.6
     25.3 26.4 28.6 29.7 30.8 31.3 32.4 33];
% attenuation characteristic of telephone circuit 7, sampled now every 50 hz;

ATC7=interp(ATC7,2);

% attenuation characteristic of telephone circuit 7, sampled now every 25 hz, to avoid aliasing;

ATC7=interp(ATC7,2);
ATC7=[85 ATC7 35];

% group delay characteristic of telephone circuit 7, sampled every 100 hz;

GDC7=[19.2 12.4 8 5.5 3.8 2.9 2.1 1.5 1.1 0.9 0.8 0.7 0.4 0.3 0.27 0.2 0.1 zeros(18:22) 0.3 0.4 0.55 0.58 0.82 1.1 1.4 1.8 2.3 3.2 4.6 6 8.2 10.2 12.4];

% group delay characteristic of telephone circuit 7, sampled now every 50 hz;

GDC7=interp(GDC7,2);

% group delay characteristic of telephone circuit 7, sampled now every 25 hz, to avoid aliasing;

GDC7=interp(GDC7,2);
GDC7=[24 GDC7 13];

% attenuation characteristic of filter set 2;

ATF2=[66.7 32 20 12 6.7 2.7 1.3 zeros(2:28) 1 4 8 17.3 32 46.7 73.3 100 106.7];
ATF2=interp(ATF2,2);
ATF2=interp(ATF2,2);
ATF2=[95 ATF2 110];

% group delay characteristic of filter set 2;

GDF2=[1.3 2.1 1.5 1.4 1.4 1.25 1.1 0.9 0.76 0.65 0.54 0.43 0.46 0.49 0.65 0.76 0.92 1.14 1.36 1.63 1.9 2.34 2.17 1.9 1.58 0.98 0.6 0.38];

GDF2=interp(GDF2,2);
GDF2=interp(GDF2,2);
GDF2=[1 GDF2 0.5];
AT7=(ATC7+ATF2)/2;
GD7=(GDC7+GDF2)/2;

for I=1:max(size(AT7))
    A(I)=10.^(-abs(AT7(I))/20);
end;
DF=25;
PHI=angle(GD7);

for I=2:max(size(GD7))
    PHI(I)=PHI(I-1)-pi*DF*((GD7(I)+
    +GD7(I-1))*10^(-3));
end;

R=A.*cos(PHI);
l=A.*sin(PHI);
Y=R+i*l;
YY=[Y zeros(1:874)];
y=ifft(YY);

TELCHN8.M

%THIS PROGRAM CALCULATES THE IMPULSE RESPONSE OF TELEPHONE CHANNEL 8;

%attenuation characteristic of telephone circuit 8 every 100 hz;
ATC8=[22 11 6.6 4.4 1.5 1 0 0 0 .5 1 2.7
      3.8 5.5 7.7 9.9 12 15 17.6 18.7 22
      25 28.6 31.9 35.7 38.5 42.9 47.3
      51.6 55 58.2 62.6 67 71.4 73.6
      80.2 83.5];

%attenuation characteristic of telephone circuit 8, sampled now every 50 hz;
ATC8=interp(ATC8,2);

%attenuation characteristic of telephone circuit 8, sampled now every 25 hz, to avoid aliasing;
ATC8=interp(ATC8,2);
ATC8=[30,ATC8 82];

%group delay characteristic of telephone circuit 8, sampled every 100 hz;
GDC8=[4.1 3.6 2 1.3 1 0.66 0.41 0.2
      0.1 zeros(10:15) 0.16 0.32
      0.49 0.81 1 1.3 1.8 2.3 3 3.8
      4.6 5.4 6.3 7.1 8 8.9 9.6
      10.5 11.4 12.2 13 13.7];
% group delay characteristic of telephone circuit 8, sampled now every 50 hz;

GDC8=interp(GDC8,2);

% group delay characteristic of telephone circuit 8, sampled now every 25 hz, to avoid aliasing;

GDC8=interp(GDC8,2);
GDC8=[5 GDC8 14];

% attenuation characteristic of filter set 2;

ATF2=[66.7 32 20 12 6.7 2.7 1.3 zeros(8:28)
     1 4 8 17.3 32 46.7 73.3 100 106.7];
ATF2=interp(ATF2,2);
ATF2=interp(ATF2,2);
ATF2=[95 ATF2 110];

% group delay characteristic of filter set 2;

GDF2=[1.3 2.1 1.5 1.4 1.4 1.25 1.1 0.9
     0.76 0.6 5 0.54 0.43 0.38
     0.38 0.39 0.38 0.38 0.35 0.34
     0.38 0.43 0.46 0.49 0.65 0.76
     0.92 1.14 1.36 1.63 1.9 2.34 2.17
     1.9 1.58 0.98 0.6 0.38];
GDF2=interp(GDF2,2);
GDF2=interp(GDF2,2);
GDF2=[1 GDF2 0.5];
AT8=(ATC8+ATF2)/2;
GD8=(GDC8+GDF2)/2;

for l=1:max(size(AT8))
    A(l)=10.^(-abs(AT8(l))/20);
end;

DF=25;
phi=angle(GD8);
for l=2:max(size(GD8))
    phi(l)=phi(l-1)-pi*DF*((GD8(l)+GD8(l-1))*10^(-3));
end;
R=A.*cos(phi);
l=A.*sin(phi);
Y=R+i*l;
YY=[Y zeros(1:874)];
y=ifft(YY);

% CEPSTRUM - KOLMOGOROFF ALGORITHM
% THIS PROGRAM PERFORMS MINIMUM PHASING OF AN INPUT SEQUENCE (REAL OR IMAGINARY) BY MEANS OF THE KOLMOGOROFF METHOD.

% g: IS THE LENGTH OF THE INPUT SEQUENCE;
    g=max(size(y));

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT % TRANSFORMS.
    n=input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE
    FF=fft(y,n); 
    F=abs(FF);
%F=F./sqrt(sum(abs(F).^2)); 
% TAKE LOG OF MAGNITUDE OF THE SPECTRUM
    LF=log(F);

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)
    ILF=(ifft(LF,n));

% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN
    odd=fix(rem(n,2));
    wn=[1; 2*ones((n+odd)/2-1,1); 1; zeros((n-odd)/2-1,1)];
    FWN=(fft((ILF.*wn'),n));

% EXPONENTIATE CEPSTRUM
    EFWN=(exp(FWN));

% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE FOURIER TRANSFORM
% ymt: TRUNCATED MINIMUM PHASE OUTPUT
    YM=(ifft(EFWN,n));
    k=input('k=');
    YMT=YM(1:k);
    YMT=YMT./sqrt(sum(abs(YMT).^2));

end;
end.
% D: MINIMUM PHASE COMBINED IMPULSE RESPONSE OF
CHANNEL & FILTER AFTER DROPPING THE FIRST G
COEFFICIENTS.
T=input('number of taps=')
[D0 R]=deconv(y,YMT);

for l=2:T
    [D(l) R]=deconv([R(2:max(size(R))) 0],YMT);
end

D0=[D0 D(2:T)];
D=conj(fliplr(DD));

C=conv(y,D);

CT=C(T:max(size(C)));
CT=CT./sqrt(sum(abs(CT.^2)));

end;
end.

NEWTON.M [MATLAB]

% THIS PROGRAM IS AN IMPLEMENTATION OF
NEWTON RAPHSON METHOD

function [p0,y0,err,P] = newton(f,df,p0,delta,epsilon,max1)
% [p0,y0,err] = newton(f,df,p0,delta,epsilon,max1)
% [p0,y0,err,P] = newton(f,df,p0,delta,epsilon,max1)
% Newton's method is used to locate a root.
% f is the function, input.
% df is the derivative of f, input.
% p0 is the starting point, input.
% delta is the tolerance for p0, input.
% epsilon is the tolerance for y0, input.
% max1 is the maximum number of iterations, input.
% p0 is the root, output.
% y0 is the function value, output.
% err is the error estimate for p0, output.
% P is the is the vector of iterations, output.
P(1) = p0;
y0 = polyval(f,p0);
for k=1:max1,
df0 = polyval(df,p0);

    p1 = p0 - dp;
y1 = polyval(f,p1);
err = abs(dp);
rele = err/(abs(p1)+eps);

end;
\[ p_0 = p_1; \]
\[ y_0 = y_1; \]
\[ P = [P; p_1]; \]
\[ \text{if (err<delta)|(relerr<delta)|(abs(y_1)<epsilon), break, end} \]
\[ \text{end} \]

**POLYVAL.M [JNL]**

```matlab
{{function y = polyval(c,x)
%POLYVAL Polynomial evaluation.
% If V is a vector whose elements are the coefficients of a
% polynomial, then POLYVAL(V,s) is the value of the
% polynomial evaluated at s. If S is a matrix or vector,
% the polynomial is evaluated at all points in S.
% See POLYVALM for evaluation in a matrix sense.
% Polynomial evaluation c(x) using Horner's method

[m,n] = size(x);
nc = max(size(c));
if (m+n) == 2
    % Make it scream for scalar X. Polynomial evaluation can be
    % implemented as a recursive digital filter.
    y = filter([1 -1],c);
    y = y(nc);
    return
end
% Do general case where X is an array
y = zeros(m,n);
for i=1:nc
    y = x.*y + c(i) * ones(m,n);
end
}}}
```

**LAGUER.M [CRD]**

```matlab
function [x,iter]=laguer(aa,xx,myeps,polish);
% [X,ITERATIONS]=laguer(AA,XX,MYEPS,POLISH).
% % Laguer root finder for vector polynomial AA, given
% % starting scalar value XX, typically 0.0. The
% % first root encountered is returned as X.
% % If the POLISH flag is 0, then the fractional
% % resolution will be MYEPS; otherwise, it will
% % be MatLab accuracy "eps". If MYEPS is 0.0,
% % a default value of 1e-8 is assumed.
% % ITER is the number of Laguer iterates
```
% used to achieve the root.
% The MATLAB procedure "ROOTS1" uses this routine.

if aa == []
    x=[];
    return;
end

a=aa;

% Initialization.

[m,n]=size(a);
m=m-1; % Actual order of polynomial.
xx=x;
dxold=abs(xx);
eps1=myeps;
if myeps == 0.0
    eps1=1.0e-8;
end
epss=eps;
maxit=100;

mm1=m-1;
mp1=m+1;

bb=a; % Polynomial.

ord=(m:-1:0).';
dd=ord.*bb;
dd=dd(1:m); % First derivative.

ff = 0;
mf=1;
if m > 1
    mf=mm1;
    ord=(mf:-1:0).';
    ff=ord.*dd;
    ff = ff(1:mf); % Second derivative.
end

absbb=abs(bb); % For error estimation.

for iter=1:1:maxit
    xpad=[1 ;-x];

    y=filter(1,xpad,ff);
f=y(mf);
    y=filter(1,xpad,dd);
d=y(m);
\[ y = \text{filter}(1, x^{\text{pad}}, bb); \quad \% \text{Need } y \text{ below.} \]
\[ b = y(mp1); \]
\[ \text{absbb} = \text{abs}(y); \]
\[ y = \text{filter}(1, [1 - \text{abs}(x)], \text{absbb}); \]
\[ \text{err} = \text{epss} \ast y(mp1); \]

\[
\text{if abs}(b) \leq \text{err} \]
\[ \text{dx} = 0.0; \]
\[ \text{return}; \]
\[ \text{else} \]
\[ g = d/b; \]
\[ g^2 = g \ast g; \]
\[ h = g^2 - f/b; \quad \% \text{Note no factor of } 2 \text{ on } f. \]
\[ \text{sq} = \sqrt{\text{mm}1 \ast (m \ast h - g^2)}; \]
\[ \text{gp} = g + \text{sq}; \]
\[ \text{gm} = g - \text{sq}; \]
\[ \text{dx} = m / \text{max}([\text{gp}; \text{gm}]); \]
\[ \text{end} \]
\[ x_1 = x - \text{dx}; \]
\[ \text{if } x = x_1 \]
\[ \text{return}; \]
\[ \text{end} \]
\[ x = x_1; \]
\[ \text{cdx} = \text{abs}(\text{dx}); \]
\[ \text{if } (\text{iter} > 6) \& (\text{cdx} \geq \text{dxold}) \]
\[ \text{return}; \]
\[ \text{end} \]
\[ \text{dxold} = \text{cdx}; \]
\[ \text{if polish} = 0 \]
\[ \text{if abs}(\text{dx}) \leq \text{eps}1 \ast \text{abs}(x); \]
\[ \text{return}; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{end} \]

\textbf{CLARK.M}

\% \text{THIS PROGRAM IMPLEMENTS THE CLARK-HAU METHOD}
\text{CCC}=[];
\text{g}=\max(\text{size}(y));
\text{lamda} = \text{input}('\text{lamda}=');
\text{g}=1:g;
\text{c}=1;
\text{d}=10^\text{(-8)};
\% \text{c} = \text{input}('\text{c}=1');
\% \text{d} = \text{input}('\text{d}=10^\text{(-8)}');
\text{for } k=1:40
\text{e0}=0; \text{ei}=0;
e0=polyval(y,lambda);
ei=polyder(y); ei=polyval(ei,lambda);
lambda=lambda-c*(e0/ei)
CC=(lambda);
CCC=[CCC 1/CC]

if abs(e0/ei).^2<dtabs(lambda)>1|k>40,break, end
end

POLYDER.M [MATLAB]
{function d=polyder(c);
k=length(c);
for j=1:(k-1)
c(j)=(k-j)*c(j);
end
c=c(1:k-1);d=c; }}

GRAMSMIT.M

%THIS PROGRAM IMPLEMENTS THE GRAM-SMIT METHOD

n=max(size(y));
g=input('number of taps=');
k=n+g-1;
T0=[y zeros(1:(k-n))];
Y=[ ];
Y1=[ ];
for l=1:g-1
T=[zeros(1:l) y zeros(1:(k-n-l))];
Y1=[Y1 T];
end
Y=[T0' Y1];
b=(k-g);
D=[ ];
Z=[ ];
end

MGS.M [CH]

{function Q = mgs(A)
% Q = mgs(A)
% Applies the modified Gram-Schmidt process to the matrix A
% (assuming that A has full rank), and returns an orthogonal
% matrix Q whose columns span the range of A.
Q = A; [m,n] = size(A);
for k=1:n
Q(:,k) = Q(:,k)/norm(Q(:,k));
Q(:,k+1:n) = Q(:,k+1:n) - Q(:,k)*(Q(:,k)'*Q(:,k+1:n));
end
}}

Q=mgs(Y);
Z=conj(Q).^Y;
Z=Z(:,1)
end

BAUER.M

%THIS PROGRAM IMPLEMENTS THE BAUER METHOD

%Form the Autocorrelation Function
R=conv(y,conj(fliplr(y)));
R=R(length(y):length(R));

% EXPLOIT THE TOEPLTZ STRUCTURE
T=toeplitz(R);

%PERFORM THE LU FACTORIZATION
[L,U]=LU(T);

%THEN THE MINIMUM PHASE SPECTRAL FACTOR IS
Ymin=U(:,1) %SLIGHTLY BETTER APPROXIMATION

% OR
Ymin=L(:,1);
end

TOEPLTZ.M [MATLAB]

function t = toeplitz(c,r)
%TOEPLTZ(C,R) is a non-symmetric Toeplitz matrix having C as
%its first column and R as its first row. TOEPLTZ(C) is symmetric
%(or Hermitian) Toeplitz matrix.
t = [];
if nargin == 1
    [m,n] = size(c);
    if n == 1
        r = c';
    else
        r = c;
        c = c';
        if n
            c(1) = r(1);
    end
end

if nargin == 2
    r = r';
end
end

else
    if all(size(r)&size(c))
        if r(1) == c(1)
            disp('')
            disp('Column wins diagonal conflict.')
        end
    end
end

n = max(size(r));
m = max(size(c));
c = c(:);  % Make sure C is a column vector
r = r(:)';  % Make sure R is a row vector
t = ones(m,n);  % Allocate T
k = min(m,n);

for i=1:k  % Fill in the matrix
    t(i;i:n) = r(1:n-i+1);  % ith row, upper triangle
    t(i:m,i) = c(1:m-i+1);  % ith col, lower triangle
end

% THIS PROGRAM IMPLEMENTS THE TOEPLITZ METHOD

R=conv(y, conj(fliplr(y)));  
R=R(length(y):length(R));  
% N=length of spectral factor
N=length(R)-1;
% USE LEVINSON RECURSION
L=levinson(R,N)

LEVINSON.M [MATLAB]

function a=levinson(R,N);  
if nargin<2, N=length(R)-1; end  
if length(R)<(N+1), error('correlation vector too short'),end

a=-R(2)/R(1);  
V=R(1)-((R(2).^2)/R(1));

for n=1:N-1,
    alfa=[1 conj(a).]*R(n+2:-1:2);
    rho=-alfa./V;
    V=V+rho*conj(alfa);
    a=[a+rho*flipud(conj(a)); rho];
end

a=[1; a].';
end
else
b=-R(2:N+1);
a=[1; toeplitz(R(1:N))\(b(;)];
end

\%a=a./sqrt((a(1)*a(length(a)))/R(length(R));

end

N IT. M

\%THIS PROGRAM IMPLEMENTS THE NIT ALGORITHM

CCT=[];
var=input('var=');
NO=input('NO=');
F=flipr(sort(abs(y)));

YMT=F; compare;
for l=1:NO
YMT=CT; A=CT; compare;
CT=CT./sqrt(sum(abs(CT.^2)));
\%CCT=[CCT CT.];
end

COMPARE. M
{compare
{(R=[];
DDDD=[];
T=length(y);

[D0 R]=deconv(y,YMT);

T=var*T; T=round(T);
for l=2:T
[DC(l) R]=deconv([R(2:max(size(R))) 0]),YMT);
end

DDC=[D0 DC(2:T)];
DF=conj(flipr(DDC));
C=conv((y),(DF));
C=C(1:(length(y)+length(DF)-1));
CT=C((T):max(size(C))); 
CT=CT./sqrt(sum(abs(CT.^2))).';}

end

HF2.M

% COMPUTER SIMULATION PROGRAM OF AN HF CHANNEL; 
% WITH TWO SKYWAVES; 
% RELATIVE DELAYS 0.3ms; FREQUENCY SPREAD 2Hz; 

% SEEDS FOR GENERATING RANDOM INPUT SEQUENCES; 
no=input('nO='); 
n1=input('n1='); 
n3=input('n3='); 
n4=input('n4='); 
n6=input('n6='); 
n7=input('n7='); 
rand('normal'); 
n=input('n=') 
rand('seed',n0); w0r=rand(n,1); w0r=0.43*w0r; 
rand('seed',n1); w0r1=rand(n,1); w0r1=0.38*w0r1; 
rand('seed',n3); w0i=rand(n,1); w0i=0.38*w0i; 
rand('seed',n4); w0i1=rand(n,1); w0i1=0.42*w0i1; 
rand('uniform'); 
rand('seed',n6); a0r=rand(n,1); 
rand('seed',n7); a0i=rand(n,1); 

% GENERATION OF SELF ORTHOGONAL INPUT SEQUENCE; 
for k=1:n  
if a0r(k) >= 0.5, a0r(k) = 1.15; else a0r(k) = -1.15; end 
if a0i(k) >= 0.5, a0i(k) = 1.15; else a0i(k) = -1.15; end 
end 

% DIFFERENTIAL CODING OF INPUT SEQUENCE; 
a0 = [a0r,a0i]; 
b0 = diffenc(a0(1,:),[0 0]); bt0 = b0; 
for k = 1:n  
b0 = diffenc(a0(k,:),bt0); b0 = [b0;bt0]; 
end 

%SAMPLED IMPULSE RESPONSE OF TX FILTER(DELAY T=0); 
SIRTxR=[-0.1796 -3.077 -9.9409 -11.7869 -3.4618 4.4438 3.0642]; 
SIRTxR=[SIRTxR -1.3596 -1.4973 0.2925 0.5180 -0.1842 -0.3167]; 
SIRTxR=[SIRTxR 0.00218 0.04438 0.05155]; 
SIRTxR=[SIRTxR 3.1582 1.7365 4.7046 -11.7870 -5.5819]; 
SIRTxR=[SIRTxR -2.107 0.0392 0.0098]; 
%SAMPLED IMPULSE RESPONSE OF TX FILTER(RELATIVE 
DELAY T=3); 
SIRTxR=[-1.3136 -7.1104 -12.3469 -7.5848 2.2354 4.5938 0.0931];
%SAMPLED IMPULSE RESPONSE OF RX FILTER;
SIRRxR=[-1.9418 -15.9798 -35.1418 -34.4789 -11.2302 7.8155 7.5124];
SIRRxR=[SIRRxR -0.5058 -3.3707 -0.6760 1.0483 0.3622 -0.3216];
SIRRxR=[SIRRxR 0.0438 0.0739 -0.0217];
SIRRxl=[1.3626 11.5941 27.3343 28.0870 7.2715 -9.2602 -5.0954];
SIRRxl=[SIRRxl 3.2327 1.8975 -1.2817 -0.4830 0.7615 0.1979 -0.1533];
SIRRxl=[SIRRxl 0.0940 -0.0212];
TAPS=[-1.6218 0.6650 -1.6954 0.7536 -0.8010];
SIRTx=SIRTxR+i*SIRTxl;
SIRTx=SIRTxR+i*SIRTxl;
SIRRx=SIRRxl+i*SIRRxl;
Q1R=conv(TAPS,w0r);
Q1l=conv(TAPS,w0i);
Q2R=conv(TAPS,w0r1);
Q2l=conv(TAPS,w0i1);
%MINIMUM PHASE VERSIONS OF THE Tx FILTERS;
SIRTx=polystab(SIRTx);
SIR1Tx=polystab(SIR1Tx);
Z1=conv(SIRTx,(Q1R-i*Q1l));
Z2=conv(SIR1Tx,(Q2R-i*Q2l));
Z=Z1+Z2;
Y=1/4800*conv(SIRRx,Z);
end

HF5.M

% COMPUTER SIMULATION PROGRAM OF AN HF CHANNEL;
%WITH THREE SKYWAVES;
%RELATIVE DELAYS 0,1,3ms; FREQUENCY SPREAD 2Hz;

%SEEDS FOR GENERATING RANDOM INPUT SEQUENCES;
no=input('nO=');
n1 =input('n1 =');
n2=input('n2=');
n3=input('n3=');
n4=input('n4=');
n5=input('n5=');
n6=input('n6=');
n7=input('n7=');
rand('normal');
n=input('n=')
rand('seed',no);w0r=rand(n,1); w0r=0.43*w0r;
rand('seed',n1);w0r1=rand(n,1); w0r1=0.38*w0r1;
PROGRAMS

\[
\text{rand('seed',n2);w0r2=\text{rand}(n, 1); w0r2=0.41\times w0r2;}
\]
\[
\text{rand('seed',n3);w0i=\text{rand}(n, 1); w0i=0.38\times w0i;}
\]
\[
\text{rand('seed',n4);wOi=\text{rand}(n, 1); wOi=0.42\times wOi;}
\]
\[
\text{rand('seed',n5);wOi1=\text{rand}(n, 1); wOi1=0.37\times wOi1;}
\]
\[
\text{rand('uniform');}
\]
\[
\text{\% GENERATION OF SELF ORTHOGONAL INPUT SEQUENCE;}
\]
\[
\text{for } k=1:n
\quad \text{if } a0r(k) \geq 0.5, a0r(k) = 1.15; \text{ else } a0r(k) = -1.15; \quad \text{end}
\]
\[
\text{if } a0i(k) \geq 0.5, a0i(k) = 1.15; \text{ else } a0i(k) = -1.15; \quad \text{end}
\]
\[
\text{end}
\]
\[
\text{\%DIFFERENTIAL ENCODING OF INPUT SEQUENCE;}
\]
\[
\text{a0 = [a0r,a0i];}
\]
\[
\text{b0 = \text{diffenc}(a0(1,:), [0 0]); bt0 = b0;}
\]
\[
\text{for } k = 1:n
\quad b0 = \text{diffenc}(a0(k,:),bt0); b0 = [b0;bt0];
\quad \text{end}
\]
\[
\%	ext{SAMPLED IMPULSE RESPONSE OF TX FILTER(DELAY T=0)};
\]
\[
\text{SIRTxR} = [-0.1796 -3.077 -9.9409 -11.7869 -3.4618 4.4438 3.0642];
\]
\[
\text{SIRTxR} = [\text{SIRTxR} -1.3596 -1.4973 0.2925 0.5180 -0.1842 -0.3167];
\]
\[
\text{SIRTxR} = [\text{SIRTxR} 0.00218 0.04438 0.05155];
\]
\[
\text{SIRTxR} = [2.3539 20.7590 45.5585 41.4959 8.7046 -11.7870 -5.5819];
\]
\[
\text{SIRTxR} = [\text{SIRTxR} 3.1582 1.7365 -0.7777 -0.1293 0.2880 -0.2325 ];
\]
\[
\text{SIRTxR} = [\text{SIRTxR} -0.2107 0.0392 0.0098];
\]
\[
\%	ext{SAMPLED IMPULSE RESPONSE OR TX FILTER(RELATIVE DELAY T=1)};
\]
\[
\text{SIRT1TxR} = [-0.3990 -4.2813 -11.0911 -10.7873 -1.3721 4.9023];
\]
\[
\text{SIRT1TxR} = [\text{SIRT1TxR} 2.0668 -1.7787 -1.1223 0.4833 0.3966 -0.2946];
\]
\[
\text{SIRT1TxR} = [\text{SIRT1TxR} -0.2415 -0.0048 -0.0236 -0.0085];
\]
\[
\%	ext{SAMPLED IMPULSE RESPONSE OF TX FILTER(RELATIVE DELAY T=3)};
\]
\[
\text{SIRT2TxR} = [0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
\]
\[
\text{SIRT2TxR} = [-1.3136 -7.1104 -12.3469 -7.5848 2.2354 4.5938 0.0931];
\]
\[
\text{SIRT2TxR} = [\text{SIRT2TxR} -1.9704 -0.3233 0.6313 0.1036 -0.3866 -0.0734];
\]
\[
\text{SIRT2TxR} = [\text{SIRT2TxR} -0.0386 0.0608 -0.0713];
\]
\[
\text{SIRT2TxR} = [11.0689 37.2136 47.9575 22.8262 -7.2498 -10.0026 0.6695];
\]
\[
\text{SIRT2TxR} = [\text{SIRT2TxR} 3.1072 -0.2261 -0.5552 0.2882 -0.0156 -0.3215];
\]
\[
\text{SIRT2TxR} = [\text{SIRT2TxR} -0.01077 0.0140 0.0135];
\]
\[
\%	ext{SAMPLED IMPULSE RESPONSE OF RX FILTER};
\]
\[
\text{SIRRxR} = [-1.9418 -15.9786 -35.1418 -34.4789 -11.2302 7.8155 7.5124];
\]
\[
\text{SIRRxR} = [\text{SIRRxR} -0.5058 -3.3707 -0.6760 1.0483 0.3622 -0.3216 ];
\]
\[
\text{SIRRxR} = [\text{SIRRxR} 0.0438 0.0739 -0.0217];
\]
\[
\text{SIRRxl} = [1.3626 11.5941 27.3343 28.0870 7.2715 -9.2602 -5.0954];
\]
PROGRAMS

SIRRxl=[SIRRxl 3.2327 1.8975 -1.2817 -0.4830 0.7615 0.1979 -0.1533];
SIRRxl=[SIRRxl 0.0940 -0.0212];
TAPS=[-1.6218 0.6650 -1.6954 0.7536 -0.8010];
SIRTx=SIRTxR+i*SIRTxl;
SIR1Tx=SIR1TxR+i*SIR1Txl;
SIR2Tx=SIR2TxR+i*SIR2Txl;
SIRRx=SIRRxR+i*SIRRxl;
Q1R=conv(TAPS,w0);  
Q1l=conv(TAPS,w0i);
Q2R=conv(TAPS,w0r1);
Q2l=conv(TAPS,w0i1);
Q3R=conv(TAPS,w0r2);
Q3l=conv(TAPS,w0i2);

%MINIMUM PHASE VERSIONS OF THE Tx FILTERS;
SIRTx=polystab(SIRTx);
SIR1Tx=polystab(SIR1Tx);
SIR2Tx=polystab(SIR2Tx);
Z1=conv(SIRTx,(Q1R-i*Q1l));
Z2=conv(SIR1Tx,(Q2R-i*Q2l));
Z3=conv(SIR2Tx,(Q3R-i*Q3l));
Z=Z1+Z2+Z3;
Y=1/4800*conv(SIRRx,Z);

end

HF4.M

% COMPUTER SIMULATION PROGRAM OF AN HF CHANNEL; 
% WITH THREE SKYWAVES; 
% RELATIVE DELAYS 0,1,3ms; FREQUENCY SPREAD 2Hz;

%SEEDS FOR GENERATING RANDOM INPUT SEQUENCES;
n0=input('n0=');
n1=input('n1=');
n2=input('n2=');
n3=input('n3=');
n4=input('n4=');
n5=input('n5=');
n6=input('n6=');
n7=input('n7=');
rand('normal');
n=input('n=')
rand('seed',n0);w0=rand(n,1); w0r=0.43*w0;
rand('seed',n1);w0r1=rand(n,1); w0r1=0.38*w0r1;
rand('seed',n2);w0r2=rand(n,1); w0r2=0.41*w0r2;
rand('seed',n3);w0l=rand(n,1); w0l=0.38*w0l;
rand('seed',n4);w0i1=rand(n,1); w0i1=0.42*w0i1;
rand('seed',n5);w0i2=rand(n,1); w0i2=0.37*w0i2;
rand('uniform');
rand('seed',n6); a0r=rand(n,1);
rand('seed',n7); a0l=rand(n,1);
% GENERATION OF SELF ORTHOGONAL INPUT SEQUENCE;
for k=1:n
    if aOr(k) >= 0.5, aOr(k) = 1.15; else aOr(k) = -1.15; end
    if aOi(k) >= 0.5, aOi(k) = 1.15; else aOi(k) = -1.15; end
end

% DIFFERENTIAL ENCODING OF INPUT SEQUENCE;
    aO = [aOr,aOi];
bO = diffenc(aO(1,:), [0 0]); bO0 = bO;
    for k = 1:n
        bO = diffenc(aO(k,:),bO0); bO = [bO;bO0];
    end

% SAMPLED IMPULSE RESPONSE OF TX FILTER (DELAY T=0);
SIRTxR=[-0.1796 -3.077 -9.9409 -11.7869 -3.4618 4.4438 3.0642];
SIRTxR=[SIRTxR -1.3596 -1.4973 0.2925 0.5180 -0.1842 -0.3167];
SIRTxR=[SIRTxR 0.00218 0.04438 0.05155];
SIRTxR=[SIRTxR 20.7590 45.5585 41.4959 8.7046 -11.7870 -5.5819];
SIRTxR=[SIRTxR 3.1582 1.7365 -0.7777 -0.1293 0.2880 -0.2325];
SIRTxR=[SIRTxR 0.2107 0.0392 0.0098];
% SAMPLED IMPULSE RESPONSE OR TX FILTER (RELATIVE DELAY T=1);
SIR1TxR=[-0.3990 -4.2813 -11.0911 -10.7873 -1.3721 4.9023];
SIR1TxR=[SIR1TxR 2.0668 -1.7787 -1.1223 0.4833 0.3966 -0.2946];
SIR1TxR=[SIR1TxR -0.2415 -0.0048 -0.0236 2 -0.0085];
% SAMPLED IMPULSE RESPONSE OF TX FILTER (RELATIVE DELAY T=3);
SIRT2TxR=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.1237 -7.1214 -12.3470];
SIRT2TxR=[SIRT2TxR -1.3136 -7.1104 -12.3469 -7.5848 2.2354 4.5938 0.0931];
SIRT2TxR=[SIRT2TxR -1.9704 0.3233 0.6313 0.1036 -0.3866 -0.0734];
SIRT2TxR=[SIRT2TxR 0.0386 0.0608 -0.0713];
SIRT2TxR=[SIRT2TxR 0.0386 0.0608 -0.0713];
SIRT2TxR=[SIRT2TxR 0.0386 0.0608 -0.0713];
SIR2TxR=[11.0689 37.2136 47.9575 22.8262 -7.2498 -10.0026 0.8695];
SIR2TxR=[SIRT2TxR 3.1072 -0.2261 -0.5552 0.2882 -0.0156 -0.3215];
SIR2TxR=[SIRT2TxR -0.0107 0.0140 0.0135];

% SAMPLED IMPULSE RESPONSE OF TX FILTER (RELATIVE DELAY T=3);
SIRT5RxR=[-1.9418 -15.9798 -35.1418 -34.4789 -11.2302 7.8155 7.5124];
SIRT5RxR=[SIRT5RxR -0.5058 -3.3707 -0.6760 1.0483 0.3622 -0.3216];
SIRT5RxR=[SIRT5RxR 0.0438 0.0739 -0.0217];
SIRT5RxR=[SIRT5RxR 1.3062 11.5941 27.3343 28.0870 7.2715 -9.2602 -5.0954];
SIRT5RxR=[SIRT5RxR 3.2327 1.8975 -1.2817 -0.4830 0.7615 0.1979 -0.1533];
SIRT5RxR=[SIRT5RxR 0.0940 -0.0212];
TAPS=[-1.6218 0.6650 -1.6954 0.7536 -0.8010];
SIRTx=SIRTxR+i*SIRTxl;
SIR1Tx=SIR1TxR+i*SIR1Txl;
SIR2Tx=SIR2TxR+i*SIR2Txl;
SIRRx=SIRRxR+i*SIRRxl;
Q1R=conv(TAPS,wOr);
Q1=conv(TAPS,w0i);
Q2R=conv(TAPS,wOr1);
Q2=conv(TAPS,w0i1);
Q3R=conv(TAPS,wOr2);
Q3=conv(TAPS,w0i2);

%MINIMUM PHASE VERSIONS OF THE Tx FILTERS;
SIRTx=polystab(SIRTx);
SIR1Tx=polystab(SIR1Tx);
SIR2Tx=polystab(SIR2Tx);
Z1=conv(SIRTx,(Q1R-i*Q11));
Z2=conv(SIR1Tx,(Q2R-i*Q21));
Z3=conv(SIR2Tx,(Q3R-i*Q31));
Z=Z1+Z2+Z3;
Y=1/4800*conv(SIRRx,Z);

CHAPTER 5

CLARKL.M

% THIS PROGRAM IMPLEMENTS THE CLARK-HAU
% ALGORITHM WITH LIMITED PRECISION ARITHMETIC

CCC=[];
CHP=input('CHP=')
g=max(size(y));
lamda=input('lamda=');
g=1:g;
c=1;
d=10^(-4);d=chop(d,CHP);

for k=1:40
  e0=0; ei=0;
e0=polyvac((y),(lamda));e0=chop(e0,CHP);
ei=polydec((y));ei=chop(ei,CHP);
ei=polyvac(ei,(lamda));ei=chop(ei,CHP);
vv=e0/aei=chop(vv,CHP);
C=c*(vv);C=chop(C,CHP);
lamda=(lamda-C);lamda=chop(lamda,CHP);
CC=(lamda);
CCC=[CCC 1/CC]
if abs(e0/aei)<sqrt(d)abs(lamda)>1|k>40,break
end
end

POLYDEC.M
POLYDEC.M

function d=polydec(c);
CHP=input('CHP=');
k=length(c);
for j=1:(k-1)
c(j)=(k-j)*c(j);
cr(j)=chop(real(c(j)),CHP);
ci(j)=chop(imag(c(j)),CHP);
c(j)=cr(j)+i*ci(j);
end
c=c(1:k-1);d=c;
end

POLYVAC.M

function y = polyvac(c,x)
%POLYVAC Polynomial evaluation.

[m,n] = size(x);
nc = max(size(c));

if (m+n) == 2
% Make it scream for scalar X. Polynomial evaluation can be
% implemented as a recursive digital filter.
y = filter(1,[1 -x],c);
y = y(nc);
return
end

% Do general case where X is an array
y = zeros(m,n);
for i=1:nc
y = CHOP((CHOP((x .* y),10) + CHOP((c(i) * ones(m,n)),10)),10);
end

LAGUEC.M

function [x,iter]=laguec(aa,xx,myeps,polish,CHP,S);
% [X,ITERATIONS]=laguer(AA,XX,MYEPS,POLISH,CHP).
%
% Laguer root finder for vector polynomial AA, given
% starting scalar value XX, typically 0.0. The
% first root encountered is returned as X.
% If the POLISH flag is 0, then the fractional
% resolution will be MYEPS; otherwise, it will
% be MatLab accuracy "eps". If MYEPS is 0.0,
% a default value of 1e-8 is assumed.
% ITER is the number of Laguer iterates
% used to achieve the root.
%
if aa == [ ]
    x=[ ];
    return;
end
a=aa;

% Initialization.
[m,n]=size(a);
m=m-1;  % Actual order of polynomial.
x=xx;

dxold=abs(x);
eps1=myeps;
if myeps == 0.0
    eps1=1.0e-8;
end
epss=eps;
epss=10^(8)
maxit=100;

mm1=m-1;
mp1=m+1;

bb=a;  % Polynomial.
ord=(m:-1:0).';

dd=ord.*bb;

dd=chop(dd,CHP);

dd=dd(1:m);  % First derivative.

ff = 0;
mf=1;
if m > 1
    mf=mm1;
    ord=(mf:-1:0).';
    ff=ord.*dd;
    ff=chop(ff,CHP);
    ff = ff(1:mf);  % Second derivative.
absbb = abs(bb);  % For error estimation.
absbb = chop(absbb, CHP)
for iter = 1:maxit
    xpad = [1 ; -x];
    y = filter(1, xpad, ff); kkk = max(size(y)); y = y + kkk*S(1:kkk).';
    y = chop(y, CHP);
    f = y(mf);
    f = chop(f, CHP);
    y = filter(1, xpad, dd); kkk = max(size(y)); y = y + kkk*S(1:kkk).';
    y = chop(y, CHP);
    d = y(m);
    d = chop(d, CHP);
    y = filter(1, xpad, bb); kkk = max(size(y)); y = y + kkk*S(1:kkk).';  % Need y below.
    y = chop(y, CHP);
    b = y(mp1);
    b = chop(b, CHP);
    absbb = abs(y);
    absbb = chop(absbb, CHP);
    y = filter(1, [1 -abs(x)], absbb); kkk = max(size(y)); y = y + kkk*S(1:kkk).';
    y = chop(y, CHP);
    err = epss*y(mp1);
    err = chop(err, CHP)
    if chop(((abs(b))), CHP) <= err
        dx = 0.0;
        return;
    else
        g = d/b;
        g = chop(g, CHP);
        g2 = g*g;
        g2 = chop(g2, CHP);
        h = g2 - f/b;  % Note no factor of 2 on f.
        h = chop(h, CHP);
    end
    sq = sqrt(chop((mm1*chop(((chop((m*h), CHP) - g2)), CHP)), CHP));
    sq = chop(sq, CHP);
    gp = g + sq;
    gp = chop(gp, CHP);
    gm = g - sq;
    gm = chop(gm, CHP);
    dx = m/max([gp ; gm])
    dx = chop(dx, CHP);
end
x1 = x-dx;
x1 = chop(x1, CHP);
if x == x1
    return;
end
x=x1;
cdx=abs(dx);
cdx=chop(cdx,CHP)
if ((iter > 6) & (cdx >= dxold))
    return;
end
dxold=cdx;
if polish == 0
    if abs(dx) <= chop((eps1*abs(x)),CHP);
        return;
    end
end
end

N I TCHOP.M

% THIS PROGRAM IMPLEMENTS THE N I T METHOD
% WITH LIMITED PRECISION ARITHMETIC

CCT=[];
CHP=input('CHP=');
var=input('var=');
NO=input('NO=');
F=fliplr(sort(abs(y);
YMT=F; comparec;
for l=1:NO
    YMT=(CT);A=CT; comparec;
    CT=CT./sqrt(sum(abs(CT.^2)))
    CCT=[CCT CT.'];
end

COMPAREC.M
% THIS PROGRAM IS CALLED BY N I TCHO.M

R=[];
DDDD=[];
T=length(y);
[D0 R]=deconv((y),(YMT));
D0=chop(D0,CHP);
R=chop(R,CHP);
\[ T = \text{var} \times T; T = \text{round}(T); \]
\[ \text{for } i = 2:T \]
\[ [\text{DC}(i) \ R] = \text{deconv}([[R(2:\text{max(size(R))) 0] \}, \text{YMT})]; \]
\[ \text{DC}(i) = \text{chop}(\text{DC}(i), \text{CHP}); \]
\[ R = \text{chop}(R, \text{CHP}); \]
\[ \text{end} \]

\[ \text{DDC} = [D0 \ D0(2:T)]; \]
\[ \text{DF} = \text{conj}(\text{flip}(\text{DDC})); \]
\[ \text{KKK} = \text{max}(\text{size}(\text{DF})); \]
\[ \text{DF} = \text{DF} + S(1:K KK); \]
\[ \%	ext{flops} \]
\[ \%	ext{flops}(0) \]
\[ \%	ext{DDDD} = [\text{DDDD} \ DF]; \]
\[ C = \text{coc}(y, \text{DF}); \]
\[ C = \text{chop}(C, \text{CHP}); \]
\[ \%	ext{C} = \text{ifft}(\text{fft}(y, \text{kkk}) \cdot \text{fft}((\text{DF}), \text{kkk})); \]
\[ C = C(1:(\text{length}(y) + \text{length}(\text{DF}) - 1)); \]
\[ \%	ext{C} = \text{conv}(y, \text{DF}); \]
\[ \text{CT} = C((T):\text{max(size(C))}); \]
\[ \text{CT} = \text{CT} / \text{sqrt}(\text{sum(abs(CT.^2))}); \]
\[ \text{end} \]

\[ \text{COMPAREC.M} \]

\{\{R=[ ]; \}
\[ \text{DDDD}=[ ]; \]
\[ T = \text{length}(y); \]
\[ [D0 \ R] = \text{deconv}(y, \text{YMT}); \]
\[ \text{D0} = \text{chop}(D0, \text{CHP}); \]
\[ R = \text{chop}(R, \text{CHP}); \]
\[ T = \text{var} \times T; T = \text{round}(T); \]
\[ \text{for } i = 2:T \]
\[ [\text{DC}(i) \ R] = \text{deconv}([[R(2:\text{max(size(R))) 0] \}, \text{YMT})]; \]
\[ \text{DC}(i) = \text{chop}(\text{DC}(i), \text{CHP}); \]
\[ R = \text{chop}(R, \text{CHP}); \]
\[ \text{end} \]

\[ \text{DDC} = [D0 \ D0(2:T)]; \]
\[ \text{DF} = \text{conj}(\text{flip}(\text{DDC})); \]
\[ \text{KKK} = \text{max}(\text{size}(\text{DF})); \]
\[ \text{DF} = \text{DF} + S(1:K K K); \]
\[ C = \text{coc}(y, \text{DF}); \]
\[ C = \text{chop}(C, \text{CHP}); \]
\[ \%	ext{C} = \text{ifft}(\text{fft}(y, \text{kkk}) \cdot \text{fft}((\text{DF}), \text{kkk})); \]
\[ C = C(1:(\text{length}(y) + \text{length}(\text{DF}) - 1)); \]
\[ \%	ext{C} = \text{conv}(y, \text{DF}); \]
\[ \text{CT} = C((T):\text{max(size(C))}); \]
\[ \text{CT} = \text{CT} / \text{sqrt}(\text{sum(abs(CT.^2))}); \]
\[ \text{end}\}
CEPS1C.M

% THIS PROGRAM IMPLEMENTS THE CEPSSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 1

%** NOTE THAT THE SCALING IS PERFORMED IN THE INPUT**
y=y/4;

% g IS THE LENGTH OF THE INPUT SEQUENCE;
g=max(size(y));

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
% TRANSFORMS.

n=input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE
FF=fft(y,n);

F=abs(FF);
%F=F./sqrt(sum(abs(F.^2)));
% TAKE LOG OF MAGNITUDE OF THE SPECTRUM
LF=log(F*4);

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)

ILF=(ifft(LF,n));
ILF1=ILF/6;

% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN

odd=fix(rem(n,2));
wn=[1; 2*ones((n+odd)/2-1,1); 1; zeros((n-odd)/2-1,1)];
FWn=fft((ILF1.*wn'),n);
% EXPONENTIATE CEPSTRUM

EFWn=(exp(FWn*6))/4;
% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE FOURIER TRANSFORM
% ymt: TRUNCATED MINIMUM PHASE OUTPUT

YM=ifft(EFWh*4,n);

k=input('k=');
YMT=YM(1:k);
YMT=YMT./sqrt(sum(abs(YMT).^2));
YMT=real(YMT)+k*S(1:k)+i*imag(YMT)+k*S(1:k);

end;
end.

flops
flops(0)

% D: MINIMUM PHASE COMBINED IMPULSE RESPONSE OF CHANNEL & FILTER
% AFTER DROPPING THE FIRST G COEFFICIENTS.
T=input('number of taps=');
[D0 R]=deconv([y zeros(1:(T-max(size(y))))],YMT);
for l=2:T
[D(l) R]=deconv([R(2:max(size(R))) 0],YMT);
end
DD=[D0 D(2:T)];
D=conj(fliplr(DD));
D=real(D)+4*S(1:max(size(D)))+i*imag(D)+4*S(1:max(size(D)));
C=conv(y,D);

CT=C(T:max(size(C)));
CT=CT./sqrt(sum(abs(CT).^2));

end;
end.

CEPS1CD.M

% THIS PROGRAM IMPLEMENTS THE CEPSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 1
% ** NOTE THAT THE SCALING IS NOW DISTRIBUTED
% THROUGHOUT THE VARIOUS STAGES**

% g: IS THE LENGTH OF THE INPUT SEQUENCE;
g=max(size(y));
% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT % TRANSFORMS.

n=input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE
FF=fft(y,n);

F=abs(FF);

%F=F./sqrt(sum(abs(F.^2)));

% TAKE LOG OF MAGNITUDE OF THE SPECTRUM
LF=log(F);

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)

ILF=ifft(LF*1.2,n)*7.5;

% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN
ILF1=[ILF(1:20) ILF(21:39)*3 ILF(40:64)];

odd=fix(rem(n,2));

wn=[1; 2*ones((n+odd)/2-1,1); 1; zeros((n-odd)/2-1,1)];

FWn=fft((ILF1.*((wn*1/2.5)))/1.2,n)/7.5;

% EXPONENTIATE CEPSTRUM

EFWn=(exp(FWn/(1/2.5)));

% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE FOURIER TRANSFORM
% ymt: TRUNCATED MINIMUM PHASE OUTPUT

YM=ifft(EFWn*4,n);

k=input('k=');

YMT=YM(1:k);

YMT=YMT./sqrt(sum(abs(YMT).^2));

YMT=real(YMT)+4*S(1:k)+i*imag(YMT)+4*S(1:k);

end;

end.
flops
flops(0)

% D : MINIMUM PHASE COMBINED IMPULSE RESPONSE OF CHANNEL & FILTER
% AFTER DROPPING THE FIRST G COEFFICIENTS.

T=input('number of taps=')

[DO R]=deconv([y zeros(1:(T-max(size(y))))],YMT);
for l=2:T
    [D(l) R]=deconv([R(2:max(size(R))) 0],YMT);
CEPS2C.M

% THIS PROGRAM IMPLEMENTS THE CEPSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 2

% ** NOTE THAT THE SCALING IS NOW DISTRIBUTED
% THROUGHOUT THE VARIOUS STAGES**

% g: IS THE LENGTH OF THE INPUT SEQUENCE;

\[
g = \text{max}(\text{size}(y));
\]

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
% TRANSFORMS.

\[
n = \text{input('n=');}
\]

% CALCULATE SPECTRUM OF INPUT SEQUENCE
\[
\text{FF} = \text{fft}(y, n);
\]

\[
\text{F} = \text{abs}((\text{FF}/4);
\]

% TAKE LOG OF MAGNITUDE OF THE SPECTRUM
\[
\text{LF} = \log(\text{F} * 4);
\]

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)
\[
\text{ILF} = \text{ifft}((\text{LF}));
\]
\[
\text{ILF1} = \text{ILF}^* 8;
\]

% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN
\[
\text{odd} = \text{fix}((\text{rem}(n, 2));
\]
\[
\text{wn} = [1; 2 * \text{ones}((n+\text{odd})/2-1, 1); 1; \text{zeros}((n-\text{odd})/2-1, 1)];
\]
\[
\text{FWn} = \text{fft}((\text{ILF1} * \text{wn}^*)/4), n);
\]
% EXPONENTIATE CEPSTRUM

EFWn=((exp((FWn*4)/8))/4);

% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE FOURIER TRANSFORM
% ymt: TRUNCATED MINIMUM PHASE OUTPUT

YM=ifft(EFWn*7,n);

    k=input('k=');
    YMT=YM(1:k);
    YMT=YMT/sqrt(sum(abs(YMT).^2));
    YMT=real(YMT)+4*S(1:k)+i*imag(YMT)+4*S(1:k);

end;
end.
flops
flops(0)

% D: MINIMUM PHASE COMBINED IMPULSE RESPONSE OF CHANNEL & FILTER
% AFTER DROPPING THE FIRST G COEFFICIENTS.
T=input('number of taps=')
[D0 R]=deconv([y zeros(1:(T-max(size(y))))],YM);
for l=2:T
    [D(l) R]=deconv([R(2:max(size(R))) 0],YMT);
end
    DD=[D0 D(2:T)];
    D=conj(fliplr(DD));
    D=real(D)+S(1:max(size(D)))+i*imag(D)+S(1:max(size(D)));

C=conv(y,D);
CT=C(T:max(size(C)));
CT=CT./sqrt(sum(abs(CT.^2)));

end;
end.

CEPS3C.M

% THIS PROGRAM IMPLEMENTS THE CEPSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 3

% ** NOTE THAT THE SCALING IS DISTRIBUTED
% THROUGHOUT THE VARIOUS STAGES**
\[ y = y \times 0.99; \]

\% g: IS THE LENGTH OF THE INPUT SEQUENCE:

\[ g = \text{max}(\text{size}(y)); \]

\% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
\% TRANSFORMS.

\[ n = \text{input}('n='); \]

\% CALCULATE SPECTRUM OF INPUT SEQUENCE
\[ \text{FF} = \text{fft}(y, n); \]

\[ \text{F} = |\text{FF}|; \]

\%F=F./sqrt(sum(abs(F).^2));

\% TAKE LOG OF MAGNITUDE OF THE SPECTRUM

\[ \text{LF} = \log(\text{F}); \]

\% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)

\[ \text{ILF} = \text{ifft}(\text{LF} \times 1.2, n) \times 7.5; \]

\% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN
\[ \text{ILF} = \{ \text{ILF}(1) \text{ILF}(2) \text{ILF}(3:24) \text{ILF}(25:40) \text{ILF}(41:63) \text{ILF}(64) \}; \]

\[ \text{ILF} = \text{ILF} / 10; \]

\% odd=fix(rem(n,2));

\[ \text{wn} = \{ 1; 2 \times \text{ones}(n + \text{odd})/2 - 1, 1; \text{zeros}(n - \text{odd})/2 - 1, 1 \}; \]

\[ \text{FWn} = \text{fft}((10 \times \text{ILF} \times (\text{wn} \times 1/2.5)) / 1.2, n) / 7.5; \]

\% EXPONENTIATE CEPSTRUM

\[ \text{EFWn} = \exp(\text{FWn} / 1.25); \]

\% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE FOURIER TRANSFORM

\% ymt: TRUNCATED MINIMUM PHASE OUTPUT

\[ \text{YM} = \text{ifft}(\text{EFWn} \times 4, n); \]

\[ k = \text{input}('k='); \]

\[ \text{YMT} = \text{YM}(1:k); \]

\[ \text{YMT} = \text{YMT} / \sqrt{\text{sum}((\text{YMT})^2)}; \]

\[ \text{YMT} = \text{real}(\text{YMT}) + 4 \times \text{S}(1:k) + i \times \text{imag}(\text{YMT}) + 4 \times \text{S}(1:k); \]

\end

\% D : MINIMUM PHASE COMBINED IMPULSE RESPONSE OF
\% CHANNEL & FILTER
\% AFTER DROPPING THE FIRST G COEFFICIENTS.
T=input('number of taps=')

[D0 R]=deconv([y zeros(1:(T-max(size(y))))],YMT);
for l=2:T
    [D(l) R]=deconv([R(2:max(size(R))) 0],YMT);
end
DD=[D0 D(2:T)];
D=conj(fliplr(DD));
D=real(D)+S(1:max(size(D)))+i*imag(D)+S(1:max(size(D)));
flops(0)
C=conv(y,D);

CT=C(T:max(size(C)));
CT=CT./sqrt(sum(abs(CT.^2)));
end;
end .

CEPS4C.M

% THIS PROGRAM IMPLEMENTS THE CEPSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 4
%
% ** NOTE THAT THE SCALING IS DISTRIBUTED
%    THROUGHOUT THE VARIOUS STAGES**

y=y*0.99;

% g: IS THE LENGTH OF THE INPUT SEQUENCE;

g=max(size(y));

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
% TRANSFORMS.

n=input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE
FF=fft(y,n);

F=abs(FF);

% F=F./sqrt(sum(abs(F.^2)));

% TAKE LOG OF MAGNITUDE OF THE SPECTRUM
LF=log(F);

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)
ILF=(ifft(LF*1.2,n)*7.5);
% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN
ILF1=[ILF(1) ILF(2) ILF(3:24) ILF(25:40) ILF(41:63) ILF(64)];
ILF1=ILF1/10;
odd=fix(rem(n,2));
wn=[1; 2*ones((n+odd)/2-1,1); 1; zeros((n-odd)/2-1,1)];
FWn=fft(10*ILF1.*(wn*1/2.5))/1.2,n)/7.5;
% EXPONENTIATE CEPSTRUM
EFWn=(exp(FWn/(1/2.5)));
% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE
% FOURIER TRANSFORM
YM=ifft(EFWn*4,n);
k=input('k=');
YMT=YM(1:k);
YMT=YMT./sqrt(sum(abs(YMT).^2));
YMT=real(YMT)+4*S(1:k)+i*imag(YMT)+4*S(1:k);
end;
end.
flops
flops(0)
% D : MINIMUM PHASE COMBINED IMPULSE RESPONSE OF
% CHANNEL & FILTER
% AFTER DROPPING THE FIRST G COEFFICIENTS.
T=input('number of taps=');
[D0 R]=deconv([y zeros(1:(T-max(size(y))))],YMT);
for l=2:T
    [D(l) R]=deconv([R(2:max(size(R))) 0],YMT);
end
DD=[D0 D(2:T)];
D=conj(fliplr(DD));
D=real(D)+S(1:max(size(D)))+i*imag(D)+S(1:max(size(D)));
C=conv(y,D);
CT=C(T:max(size(C)));
CT=CT./sqrt(sum(abs(CT.^2)));
end

CEPS5C.M
% THIS PROGRAM IMPLEMENTS THE CEPSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 5
% ** NOTE THAT THE SCALING IS DISTRIBUTED
% THROUGHOUT THE VARIOUS STAGES**

y=y/0.99;

% g: IS THE LENGTH OF THE INPUT SEQUENCE;
    
g=max(size(y));

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
% TRANSFORMS.

n=input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE

FF=fft(y,n);

F=abs(FF/5);

% TAKE LOG OF MAGNITUDE OF THE SPECTRUM

LF=log(F*S);

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)

ILF=(ifft(LF,n));

ILF1=ILF;

% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN

odd=fix(rem(n,2));

wn=[1; 2*ones((n+odd)/2-1,1); 1; zeros((n-odd)/2-1,1)];

FWn=fft(((ILF1.*wn')/4),n);

% EXPONENTIATE CEPSTRUM

EFWn=((exp((FWn*4))/4);

% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE
% FOURIER TRANSFORM

% ymt: TRUNCATED MINIMUM PHASE OUTPUT

YM=ifft(EFWn,n);

k=input('k=');

YMT=YM(1:k);

YMT=YMT/sqrt(sum(abs(YMT).^2));

YMT=real(YMT)+4*S(1:k)+i*imag(YMT)+4*S(1:k);
CEPS6C.M

% THIS PROGRAM IMPLEMENTS THE CEPSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 6

% ** NOTE THAT THE SCALING IS DISTRIBUTED
% THROUGHOUT THE VARIOUS STAGES**

y=y*0.99;

% g: IS THE LENGTH OF THE INPUT SEQUENCE;

g=max(size(y));

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
% TRANSFORMS.

n=input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE
FF=fft(y,n);
F=abs(FF);
%F=F./sqrt(sum(abs(F.^2)));  
% TAKE LOG OF MAGNITUDE OF THE SPECTRUM

LF=log(F);
% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)

ILF=(ifft(LF*1.2,n)*7.5);
% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN
ILF1=[ILF(1) ILF(2) ILF(3:24) ILF(25:40) ILF(41:63) ILF(64)];
odd=fix(rem(n,2));
wn=[1; 2*ones((n+odd)/2-1,1); 1; zeros((n-odd)/2-1,1)];
FWn=fft(((10*ILF1.*(wn*1/2.5))/1.2,n)/7.5;
% EXPONENTIATE CEPSTRUM

EFWn=(exp(FWn/(1/2.5)));
% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE FOURIER TRANSFORM
% ymt: TRUNCATED MINIMUM PHASE OUTPUT

YM=ifft(EFWn*4,n);
k=input(k=');
YMT=YM(1:k);
YMT=real(YMT)+4*S(1:k)+i*imag(YMT)+4*S(1:k);

end;
end.

% D : MINIMUM PHASE COMBINED IMPULSE RESPONSE OF CHANNEL & FILTER
% AFTER DROPPING THE FIRST G COEFFICIENTS.
T=input(number of taps=')
[D0 R]=deconv([y zeros(1:(T-max(size(y)))],YMT);
for l=2:T
[D(l) R]=deconv([R(2:max(size(R))) 0],YMT);
end
DD=[D0 D(2:T)];
D=conj(flipr(DD));
D=real(D)+S(1:max(size(D)))+i*imag(D)+S(1:max(size(D)));
C=conv(y,D);

CT=C(T:max(size(C)));
CT=CT./sqrt(sum(abs(CT.^2)));  

end;
end.
CEPS7.M

% THIS PROGRAM IMPLEMENTS THE CEPSTRUM ALGORITHM
% WITH LIMITED PRECISION ARITHMETIC OVER CHANNEL 7

% ** NOTE THAT THE SCALING IS DISTRIBUTED
% THROUGHOUT THE VARIOUS STAGES**

y=y/0.99;

% g: IS THE LENGTH OF THE INPUT SEQUENCE;

g=max(size(y));

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
% TRANSFORMS.

n=input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE

FF=fft(y,n);

F=abs(FF);

% TAKE LOG OF MAGNITUDE OF THE SPECTRUM

LF=log(F);

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)

ILF=(ifft(LF,n));

ILF1=ILF;

% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN

odd=fix(rem(n,2));

wn=[1; 2*ones((n+odd)/2-1,1); 1; zeros((n-odd)/2-1,1)];

FWn=fft(((ILF1.*wn')/4),n);

% EXPONENTIATE CEPSTRUM

EFWn=((exp((FWn*4))));

% ym: MINIMUM PHASE OUTPUT BY PERFOMING INVERSE
% FOURIER TRANSFORM

% ymt: TRUNCATED MINIMUM PHASE OUTPUT
YM=ifft(EFWn*6,n);

k=input('k=');
YMT=YM(1:k);
YMT=YMT./sqrt(sum(abs(YMn).^2));
YMT=real(YMn+4*S(1:k))+i*imag(YMn+4*S(1:k));

end;
end.
flops
flops(0)

% D : MINIMUM PHASE COMBINED IMPULSE RESPONSE OF CHANNEL & FILTER

% AFTER DROPPING THE FIRST G COEFFICIENTS.
T=input('number of taps=')
[D0 R]=deconv([y zeros(1:(T-max(size(y))))]),YMT);
for l=2:T
[D(l) R]=deconv([R(l):max(size(R)) 0],YMT);
end
DD=[D0 D(2:T)];
D=conj(fliplr(DD));
D=real(D)+S(1:max(size(O)))+i*imag(D)+S(1:max(size(D)));

C=conv(y,D);

CT=C(T:max(size(C)));
CT=CT./sqrt(sum(abs(CT.^2)));
% g: IS THE LENGTH OF THE INPUT SEQUENCE;
    g = max(size(y));

% n: SPECIFIES THE LENGTH OF THE PERFORMED FFT & IFFT
% TRANSFORMS.
    n = input('n=');

% CALCULATE SPECTRUM OF INPUT SEQUENCE
    FF = fft(y, n);
    F = abs(FF);

% TAKE LOG OF MAGNITUDE OF THE SPECTRUM
    LF = log(F);

% CALCULATE CEPSTRUM (ie IFFT of LOG-MAG of SPECTRUM)
    ILF = ifft(LF, n);
    ILF1 = ILF;

% PERFORM WINDOWING IN THE CEPSTRAL DOMAIN
    odd = fix(rem(n, 2));
    wn = [1; 2*ones((n+odd)/2-1, 1); 1; zeros((n-odd)/2-1, 1)];
    FWn = fft(((ILF1.*wn')/4), n);

% EXPONENTIATE CEPSTRUM
    EFWn = ((exp((FWn*4))));

% ym: MINIMUM PHASE OUTPUT BY PERFORMING INVERSE
% FOURIER TRANSFORM
% ymt: TRUNCATED MINIMUM PHASE OUTPUT

    YM = ifft(EFWn*6, n);
    k = input('k=');
    YMT = YM(1:k);
    YMT = YMT/sqrt(sum(abs(YMT).^2));
    YMT = real(YMT)+4*S(1:k)+i*imag(YMT)+4*S(1:k);

    end;
end.

% D: MINIMUM PHASE COMBINED IMPULSE RESPONSE OF
% CHANNEL & FILTER
% AFTER DROPPING THE FIRST G COEFFICIENTS.
    T = input('number of taps=');
\[
[D_0 \, R] = \text{deconv}([y \, \text{zeros}(1:(T-\max(size(y)))]), YMT);
\]
for \( i = 2:T \)
\[
[D(i) \, R] = \text{deconv}([R(2:\max(size(R))) 0], YMT);
\]
end
\[
DD = [D_0 \, D(2:T)];
\]
\[
D = \text{conj}(\text{fliplr}(DD));
\]
\[
C = \text{conv}(y, D);
\]
\[
CT = C(T: \max(size(C)));
\]
\[
CT = \frac{CT}{\sqrt{\text{sum}(\text{abs}(CT^2))}};
\]
end;
end.

\section*{CHOP.M [MATLAB]}

\begin{verbatim}
{{function X=chop(Xin,n,unit)
  \% CHOP. CHOP(X,n) rounds the elements of X to n significant figures.
  \% CHOP(X,n,unit) rounds the elements of X to n significant figures whose digits (mantissa) are exactly divisible by unit.
  \%
  \% Set last sig. fig. rounding to 1 if only two input arguments.
  \% if nargin<3, unit=1; end
  \% Cater for -ve numbers and numbers = 0.
  X=abs(Xin) +(Xin==0);
  [nx,mx] = size(X);
  exponent=unit.*((10*ones(nx,mx)).*(floor(log10(X))-n+1));
  X=round(X./exponent).*exponent;
  \% Put back sign and zeros
  X=sign(Xin).*X.*(Xin~=0);}}
\end{verbatim}
1. FFT FOR THE TMS320C25 IMPLEMENTATION [P P- J M C] *
* 64 - POINT COMPLEX, RADIX-2 FFT IN THE TMS320C25. (MEMORY-TO-MEMORY) *
* THE PROGRAM IS BASED ON THE BOOK 'DIGITAL SIGNAL PROCESSING APPLICATIONS' *
* FROM TEXAS INSTRUMENTS P. 69. IT IS OPTIMIZED FOR THE TMS320C25 INCLUDING *
* BIT REVERSAL ADDRESSING MODE. *
* EXECUTE TIME SIZE INSTRUCTION FREQ SIZE
  (MS) (WORDS) CYCLES (MHz) (BITS)
0.24704 (mS) 3218 3088 50 16

*********************************************************************
*****
***** THIS PROGRAM INCLUDES FOLLOWING FILES: 
*****
***** THE FILE 'TWID64.Q12' CONSISTS OF TWIDDLE FACTORS IN Q12 FORMAT
***** THE FILE 'ARRAY' CONSISTS OF VARIABLE NAMES
*****
*********************************************************************
**
** ALL INPUT REAL AND IMAGINARY DATA POINTS ARE ASSUMED TO BE IN CONSECUTIVE LOCATIONS (A TOTAL OF 64) IN EXTERNAL DATA MEMORY LOCATIONS.
**
**
N .set 64 ; NUMBER OF POINTS FOR FFT
.include TWID64.Q12
.include ARRAY64
*
* DEF: ARP=2 FOR INPUT AND OUTPUT
* DEF: AR2 -> QR,QI,QR+1,...
* DEF: AR3 -> PR,P1,PR+1,...
ZEROI $MACRO

*  
*  CALCULATE Re[P+Q] AND Re[P-Q]
*  QR'=(PR-QR)/2
*  PR'=(PR+QR)/2
*  PI'=(PI+QI)/2
*  PI'=(PI-QI)/2
*  AR1 AR2 ARP
LAC * ,15,AR1 ; ACC := (1/2)(QR) PR QR 1
ADD * ,15 ; ACC := (1/2)(PR+QR) PR QR 1
SACH *+,0,AR2 ; PR := (1/2)(PR+QR) PI QR 2
SUBH * ; ACC := (1/2)(PR+QR)-(QR) PI QR 2
SACH *+ ; QR := (1/2)(PR-QR) PI QI 2
*
*CALCULATE Im[P+Q] AND Im[P-Q]*
*LAC * ,15,AR1 ; ACC := (1/2)(QI) PI QI 1
ADD * ,15 ; ACC := (1/2)(PI+QI) PI QI 1
SACH *+,0,AR2 ; PI := (1/2)(PI+QI) PR+1 QI 2
SUBH * ; ACC := (1/2)(PI+QI)-(QI) PR+1 QI 2
SACH *+ ; QI := (1/2)(PI-QI) PR-QI 2
$ENDM

PBY2I $MACRO

*  
*  CALCULATE Re[P+JQ] AND Re[P-JQ]
*  PT=(PI-QR)/2
*  (QI) QR'=(PI+QR)/2
*  PR'=(PR+QI)/2
*  QI'=QR'
*  QR'=(PR-QI)/2
*MAR *+ ; PR QI 2
LAC * ,15,AR1 ; ACC := (1/2)(QI) PR QI 1
ADD * ,15 ; ACC := (1/2)(PR+QI) PR QI 1
SACH *+,0,AR2 ; PR := (1/2)(PR+QI) PI QI 2
SUBH * ; ACC := (1/2)(PR+QI)-(QI) PI QR 2
DMOV * ; QR -> QI PI QR 2
SACH *+,0,AR1 ; QR := (1/2)(PR-QI) PI QI 1
*
*CALCULATE Im[P+JQ] AND Im[P-JQ]*
*<QI:=QR>
LAC * ,15,AR2 ; ACC := (1/2)(PI) PI QI 2
SUB * ,15,AR1 ; ACC := (1/2)(PI-QR) PI QI 1
SACH *+,0,AR2 ; PI := (1/2)(PI-QR) PR+1 QI 2
ADDH * ; ACC := (1/2)(PI+QR) PR+1 QI 2
SACH *+ ; QI := (1/2)(PI+QR) PR+1 QI+1 2
$ENDM

* PBY4J $MACRO
* T = SIN(45) = COS(45)
* QI' = (QI-QR)/2  TMP = (QI+QR)/2
* PR' = (PR+TMP*W45)/2  QR' = (PR-PR+QI'*W45)/2
* PI' = (PI+QI'*W45)/2  QI* = (PI-QI'*W45)/2

LAC *+,15 ; ACC := (1/2)(QR) PR QI 2
ADD *-,15 ; ACC := (1/2)(QI+QR) PR QR 2
SACH TMP ; TMP := (1/2)(QI+QR) PR QR 2
SUBH *+ ; ACC := (1/2)(QI-QR) PR QI 2
SACH *-,0,AR1 ; QI := (1/2)(QI-QR) PR QR 1

MPY TMP ; P := (1/4)(QI+QR)*W PR QR 1
LAC *+,14 ; ACC := (1/4)(PR) PR QR 1
APAC ; ACC := (1/4)[PR+(QI+QR)*W] PR QR 1
SACH *+,1,AR2 ; PR := (1/2)[PR+(QI+QR)*W] PI QR 2
SPAC ; ACC := (1/4)(PI) PI QR 2
SPAC ; ACC := (1/4)[PR-(QI+QR)*W] PI QR 2
SACH *+,1,AR1 ; QR := (1/2)[PR-(QI+QR)*W] PI QI 1
LAC *+,14,AR2 ; ACC := (1/4)(PI) PI QI 2
MPY *,AR1 ; P := (1/4)(QI-QR)*W PI QI 1
APAC ; ACC := (1/4)[PI+(QI-QR)*W] PI QI 1
SACH *+,1,AR2 ; PI := (1/2)[PI+(QI-QR)*W] PR+1 QI 2
SPAC ; ACC := (1/4)(PI) PR+1 QI 2
SPAC ; ACC := (1/4)[PI-(QI-QR)*W] PR+1 QI 2
SACH *+,1 ; QI := (1/2)[PI-(QI-QR)*W] PR+1 QI+1 2
$ENDM

* PBY4I $MACRO
LT W45 ; SIN(45) = COS(45)
PBY4J
$ENDM

* P3BY4J $MACRO
* T = W45
* QI' = (QI-QR)/2  TMP = (QI+QR)/2
* PR' = (PR+QI'*W45)/2  QR' = (PR-QI'*W45)/2
* \( P_I' = (P_I - \text{TMP} \times W45)/2 \) \( Q_I' = (P_I + \text{TMP} \times W45)/2 \)

\$\text{LOOP} 8 \\
\text{ZEROI} \\
\text{PBY4J} \\
\text{PBY2I} \\
\text{LAC \,*+,15 \,; \,ACC} := (1/2)(QR) \quad \text{PR} \quad Q_I \quad 2 \\
\text{ADD \,*-,15 \,; \,ACC} := (1/2)(Q_I+QR) \quad \text{PR} \quad QR \quad 2 \\
\text{SACH \,TMP \,; \,TMP} := (1/2)(Q_I+QR) \quad \text{PR} \quad QR \quad 2 \\
\text{SUBH \,*+ \,; \,ACC} := (1/2)(Q_I-PR) \quad \text{PR} \quad Q_I \quad 2 \\
\text{SACH \,*0,AR1 \,; \,Q_I} := (1/2)(Q_I+QR) \quad \text{PR} \quad Q_I \quad 1 \\
\text{LAC \,*+,14,AR2 \,; \,ACC} := (1/4)(PR) \quad \text{PR} \quad Q_I \quad 2 \\
\text{MPY \,*-,AR1 \,; \,PREG} := (1/4)(Q_I-QR)*W \quad \text{PR} \quad QR \quad 1 \\
\text{APAC \,; \,ACC} := (1/4)[PR+(Q_I-QR)*W] \quad \text{PR} \quad QR \quad 1 \\
\text{SACH \,*+,1,AR2 \,; \,PR} := (1/2)[PR+(Q_I-QR)*W] \quad \text{PI} \quad QR \quad 2 \\
\text{SPAC \,; \,ACC} := (1/4)(PR) \quad \text{PI} \quad QR \quad 2 \\
\text{MPYS \,TMP \,; \,ACC} := (1/4)[PR-(Q_I-QR)*W] \quad \text{PI} \quad QR \quad 2 \\
\text{SACH \,*+,1,AR1 \,; \,QR} := (1/2)[PR-(Q_I-QR)*W] \quad \text{PI} \quad QI \quad 1 \\
\text{LAC \,*+,14 \,; \,ACC} := (1/4)(PR) \quad \text{PI} \quad QI \quad 1 \\
\text{SPAC \,; \,ACC} := (1/4)[PI-(Q_I+QR)*W] \quad \text{PI} \quad QI \quad 1 \\
\text{SACH \,*0+,1,AR2 \,; \,PI} := (1/2)[PI-(Q_I+QR)*W] \quad PR+5 \quad QI \quad 2 \\
\text{APAC \,; \,ACC} := (1/4)(PI) \quad PR+5 \quad QR \quad 2 \\
\text{APAC \,; \,ACC} := (1/4)[PI+(Q_I+QR)*W] \quad PR+5 \quad QI \quad 2 \\
\text{SACH \,*0+,1 \,; \,QI} := (1/2)[PI+(Q_I+QR)*W] \quad PR+5 \quad QR+5 \quad 2 \\
\text{ENDLOOP} \\
\text{ENDM} \\
\text{P3BY4I} \ \$\text{MACRO} \\
\text{QI}' = (Q_I-QR)/2 \quad \text{TMP}' = (Q_I+QR)/2 \\
\text{PR}' = (PR+Q_I*W45)/2 \quad \text{QR}' = (PR-QI*W45)/2 \\
\text{PI}' = (PI-TMP*W45)/2 \quad \text{QI}'' = (PI+TMP*W45)/2 \\
\text{LT \,W45 \; \,\text{SIN}(45) = \text{COS}(45)} \\
\text{LAC \,*+,15 \,; \,ACC} := (1/2)(QR) \quad \text{PR} \quad QI \quad 2 \\
\text{ADD \,*-,15 \,; \,ACC} := (1/2)(Q_I+QR) \quad \text{PR} \quad QR \quad 2 \\
\text{SACH \,TMP \,; \,TMP} := (1/2)(Q_I+QR) \quad \text{PR} \quad QR \quad 2 \\
\text{SUBH \,*+ \,; \,ACC} := (1/2)(Q_I-QR) \quad \text{PR} \quad QI \quad 2 \\
\text{SACH \,*0,AR1 \,; \,Q_I} := (1/2)(Q_I-QR) \quad \text{PR} \quad QI \quad 1 \\
\text{LAC \,*+,14,AR2 \,; \,ACC} := (1/4)(PR) \quad \text{PR} \quad QI \quad 2 \\
\text{MPY \,*-,AR1 \,; \,PREG} := (1/4)(Q_I-QR)*W \quad \text{PR} \quad QR \quad 1 \\
\text{APAC \,; \,ACC} := (1/4)[PR+(Q_I-QR)*W] \quad \text{PR} \quad QR \quad 1 \\
\text{SACH \,*+,1,AR2 \,; \,PR} := (1/2)[PR+(Q_I-QR)*W] \quad \text{PI} \quad QR \quad 2
PROGRAMS

SPAC ; ACC := (1/4)(PR) PI QR 2
MPYS TMP ; ACC := (1/4)[PR-(QI-QR)*W] PI QR 2
* P := (1/4)(QI+QR)*W
SACH *+,1,AR1 ; QR := (1/2)[PR-(QI-QR)*W] PI QI 1
* LAC *,14 ; ACC := (1/4)(PI) PI QI 1
SPAC ; ACC := (1/4)[PI-(QI+QR)*W] PI QI 1
SACH *+,1,AR2 ; PI := (1/2)[PI-(QI+QR)*W] PR+1 QI 2
APAC ; ACC := (1/4)(PI) PR+1 QI 2
APAC ; ACC := (1/4)[PI+(QI+QR)*W] PR+1 QI 2
SACH *+,1 ; QI := (1/2)[PI+(QI+QR)*W] PR+1 QR+1 2
$ENDM

* COMBOI $MACRO RI,R2,R3,R4
* 
* CALCULATE PARTIAL TERMS FOR R3,R4,I3 AND I4
* 
LAC :R3:,14 ; ACC := (1/4)(R3)
ADD :R4:,14 ; ACC := (1/4)(R3+R4)
SACH :R3:,1 ; R3 := (1/2)(R3+R4)
SUB :R4:,15 ; ACC := (1/4)(R3+R4)-(1/2)(R4)
SACH :R4:,1 ; R4 := (1/2)(R3-R4)
LAC :R3:+1,14 ; ACC := (1/4)(I3)
ADD :R4:+1,14 ; ACC := (1/4)(I3+I4)
SACH :R3:+1,1 ; I3 := (1/2)(I3+I4)
SUB :R4:+1,15 ; ACC := (1/4)(I3+I4)-(1/2)(I4)
SACH :R4:+1,1 ; I4 := (1/2)(I3-I4)

* CALCULATE PARTIAL TERMS FOR R2,R4,I2 AND I4
* 
LAC :R1:,14 ; ACC := (1/4)(R1)
ADD :R2:,14 ; ACC := (1/4)(R1+R2)
SACH :R1:,1 ; R1 := (1/2)(R1+R2)
SUB :R2:,15 ; ACC := (1/4)(R1+R2)-(1/2)(R2)
ADD :R4:+1,15 ; ACC := (1/4)[(R1-R2)+(I3-I4)]
SACH :R2: ; R2 := (1/4)[(R1-R2)+(I3-I4)]
SUBH :R4:+1 ; ACC := (1/4)[(R1-R2)-(I3-I4)]
DMOV :R4: ; I4 := R4 = (1/2)(R3-R4)
SACH :R4: ; R4 := (1/4)[(R1-R2)-(I3-I4)]
LAC :R1:+1,14 ; ACC := (1/4)(I1)
ADD :R2:+1,14 ; ACC := (1/4)(I1+I2)
SACH :R1:+1,1 ; I1 := (1/2)(I1+I2)
SUB :R2:+1,15 ; ACC := (1/4)(I1+I2)-(1/2)(I2)
SUB :R4:+1,15 ; ACC := (1/4)[(I1-I2)-R3-R4]
SACH :R2:+1 ; I2 := (1/4)[(I1-I2)-R3-R4]
ADDH :R4:+1 ; ACC := (1/4)[(I1-I2)+(R3-R4)]
SACH :R4:+1 ; I4 := (1/4)[(I1-I2)+(R3-R4)]

* CALCULATE PARTIAL TERMS FOR R1,R3,I1 AND I3
* LAC :R1:+15 ; ACC := (1/4)(R1+R2).
ADD :R3:+15 ; ACC := (1/4)[(R1+R2)+(R3+R4)]
SACH :R1 ; R1 := (1/4)[(R1+R2)+(R3+R4)]
SUBH :R3 ; ACC := (1/4)[(R1+R2)-(R3+R4)]
SACH :R3 ; R3 := (1/4)[(R1+R2)-(R3+R4)]
LAC :R1:+1,15 ; ACC := (1/4)(I1+I2)
ADD :R3:+1,15 ; ACC := (1/4)[(I1+I2)+(I3+I4)]
SACH :R1:+1 ; I1 := (1/4)[(I1+I2)+(I3+I4)]
SUBH :R3:+1 ; ACC := (1/4)[(I1+I2)-(I3+I4)]
SACH :R3:+1 ; I3 := (1/4)[(I1+I2)-(I3+I4)]
$ENDM
BTRFLI $MACRO WR, WI
*  QR' = (QR*WR+QI*WI)/2  QI' = (QI*WR-QR*WI)/2
*  PR' = (PR+QR')/2      QR* = (PR-QR')/2
*  PI' = (PI+QI')/2      QI" = (PI-QI')/2
*  AR1 AR2 ARP
LT *+ ; TREG:= QR
MPYK :WR ; PREG:= (1/16)(QR*WR) PR QI 2
LTP *- ; ACC := (1/16)(QR*WR); T=QI PR QR 2
MPYK :WI ; PREG:= (1/16)(QI*WI) PR QR 2
LTA * ; ACC := (1/16)(QR*WR+QI*WI) PR QR 2
*  TREG:= QR
SACH *+,3 ; QR := (1/2)(QR*WR+QI*WI) PR QI 2
MPYK :-WI ; PREG:= (1/16)(-QR*WI) PR QI 2
LTP * ; ACC := (1/16)(-QR*WI); T=QI PR QI 2
MPYK :WR ; PREG:= (1/16)(QI*WR) PR QI 2
APAC ; ACC := (1/16)(QI*WR-QR*WI) PR QI 2
SACH *+,3,AR1 ; QI := (1/2)(QI*WR-QR*WI) PR QI 2
*  AR1 AR2 ARP
LAC *,14,AR2 ; ACC := (1/4)PR PR QR 2
ADD *,15,AR1 ; ACC := (1/4)[PR+(QR*WR+QI*WI)] PR QR 1
SACH *+,1,AR2 ; PR := (1/2)[PR+(QR*WR+QI*WI)] PI QR 2
SUBH * ; ACC := (1/4)[PR-(QR*WR+QI*WI)] PI QR 2
SACH *+,1,AR1 ; QR := (1/2)[PR-(QR*WR+QI*WI)] PI QI 1
*  AR1 AR2 ARP
LAC *,14,AR2 ; ACC := (1/4)PI PI QI 2
ADD *,15,AR1 ; ACC := (1/4)[PI+(QI*WR-QR*WI)] PI QI 1
SACH *+,1,AR2 ; PI := (1/2)[PI+(QI*WR-QR*WI)] PR+1 QI 2
SUBH * ; ACC := (1/4)[PI-(QI*WR-QR*WI)] PR+1 QI 2
SACH *+,1 ; QI := (1/2)[PI-(QI*WR-QR*WI)] PR+1 QI+1 2
$ENDM
* .def W45,TMP,DATA,BUFFER
  .def INIT,FFT64,INPUT
.def    STAGE1,STAGE3,STAGE4,STAGE5,STAGE6
.def    OUTPUT,END
*
.bss   W45,1
.bss   TMP,1
DATA   .usect  "DAT",N*2
BUFFER .usect  "DAT",N*2
*
.text

INIT:  SP 0
CNFD
ROVM
SSXM
LDPK  0
LALK  05A82H ; SIN(45)
SACL   W45 ; W=SIN(45)
LARP   AR2
*
* INPUT DATA TO PORT 0
* INPUT:
LRLK   AR2,BUFFER ; READ 128 WORDS FROM PORT 0
RPTK   2*N-1
IN     *+,PA0
*
* FFT CODE WITH BIT-REVERSED INPUT SAMPLES
* FFT64:
LRLK   AR2,DATA ; TRANSFER 128 WORDS FROM B1->B0
LARK   AR0,N
RPTK   N-1
BLKD   BUFFER,*BR0+
LRLK   AR2,DATA+1 ; AR3=0201H
RPTK   N-1
BLKD   BUFFER+040h,*BR0+
*
* FFT CODE FOR STAGES 1 AND 2
* STAGE1: LDPK   4
    COMBOI X000,X001,X002,X003
    COMBOI X004,X005,X006,X007
    COMBOI X008,X009,X010,X011
    COMBOI X012,X013,X014,X015
    COMBOI X016,X017,X018,X019
    COMBOI X020,X021,X022,X023
    COMBOI X024,X025,X026,X027
    COMBOI X028,X029,X030,X031
    COMBOI X032,X033,X034,X035
    COMBOI X036,X037,X038,X039
COMBO! X040,X041,X042,X043
COMBO! X044,X045,X046,X047
COMBO! X048,X049,X050,X051
COMBO! X052,X053,X054,X055
COMBO! X056,X057,X058,X059
COMBO! X060,X061,X062,X063

* FFT CODE FOR STAGE 3 *

STAGE3: LARK AR0,N/8+1
LRLK AR1,DATA
LRLK AR2,DATA+N/8
LDPK 0
LT W45 ; SIN(45)=COS(45)->PBY4I,P3BY4I
P3BY4J ; REPEAT 4 MACROS 8 TIMES

* FFT CODE FOR STAGE 4 *

STAGE4: LARK AR0,N/4
LRLK AR1,DATA
LRLK AR2,DATA+N/4
STGE4M $MACRO
ZEROI
BTRFLI C004,S004
PBY4I
BTRFLI C012,S012
PBY2I
BTRFLI C020,S020
P3BY4I
BTRFLI C028,S028
$ENDM
STGE4M MAR *0+,AR1
MAR *0+,AR2
STGE4M MAR *0+,AR1
MAR *0+,AR2
STGE4M MAR *0+,AR1
MAR *0+,AR2
STGE4M

* FFT CODE FOR STAGE 5 *

STAGE5: LARK AR0,N/2
LRLK AR1,DATA
LRLK AR2,DATA+N/2
ZEROI
BTRFLI C002,S002
BTRFLI C004,S004
BTRFLI C006,S006
PBY4I
BTRFLI C010,S010
BTRFLI C012,S012
BTRFLI C014,S014
PBY2I
BTRFLI C018,S018
BTRFLI C020,S020
BTRFLI C022,S022
P3BY4I
BTRFLI C026,S026
BTRFLI C028,S028
BTRFLI C030,S030

* MAR *0+,AR1
* MAR *0+,AR2
ZEROI
BTRFLI C002,S002
BTRFLI C004,S004
BTRFLI C006,S006
PBY4I
BTRFLI C010,S010
BTRFLI C012,S012
BTRFLI C014,S014
PBY2I
BTRFLI C018,S018
BTRFLI C020,S020
BTRFLI C022,S022
P3BY4I
BTRFLI C026,S026
BTRFLI C028,S028
BTRFLI C030,S030

* FFT CODE FOR STAGE 6
*
STAGE6: LRLK AR1,DATA
LRLK AR2,DATA+N
ZEROI
BTRFLI C001,S001
BTRFLI C002,S002
BTRFLI C003,S003
BTRFLI C004,S004
BTRFLI C005,S005
BTRFLI C006,S006
BTRFLI C007,S007
PBY4I
BTRFLI C009,S009
BTRFLI C010,S010
BTRFLI C011,S011
BTRFLI C012,S012
BTRFLI C013,S013
BTRFLI C014,S014
BTRFLI C015,S015
PBY21
BTRFLI C017,S017
BTRFLI C018,S018
BTRFLI C019,S019
BTRFLI C020,S020
BTRFLI C021,S021
BTRFLI C022,S022
BTRFLI C023,S023
P3BY41
BTRFLI C025,S025
BTRFLI C026,S026
BTRFLI C027,S027
BTRFLI C028,S028
BTRFLI C029,S029
BTRFLI C030,S030
BTRFLI C031,S031

* OUTPUT DATA TO PORT 0 *

OUTPUT: LRLK AR2,DATA ; WRITE 128 WORDS TO PORT 0
LARP AR2
RPTK 2*N-1
OUT *+,PA0

END: B END
.end

S000: .equ 0
S001: .equ 401
S002: .equ 799
S003: .equ 1189
S004: .equ 1567
S005: .equ 1931
S006: .equ 2276
S007: .equ 2598
S008: .equ 2896
S009: .equ 3166
S010: .equ 3406
S011: .equ 3612
S012: .equ 3784
S013: .equ 3920
S014: .equ 4017
S015: .equ 4076
S016: .equ 4095
S017: .equ 4076
S018: .equ 4017
S019: .equ 3920
S020: .equ 3784
S021: .equ 3612
S022: .equ 3406
S023: .equ 3166
S024: .equ 2896
S025: .equ 2598
S026: .equ 2276
S027: .equ 1931
S028: .equ 1567
S029: .equ 1189
S030: .equ 799
S031: .equ 401
C000: .equ 4095
C001: .equ 4076
C002: .equ 4017
C003: .equ 3920
C004: .equ 3784
C005: .equ 3612
C006: .equ 3406
C007: .equ 3166
C008: .equ 2896
C009: .equ 2598
C010: .equ 2276
C011: .equ 1931
C012: .equ 1567
C013: .equ 1189
C014: .equ 799
C015: .equ 401
C016: .equ 0
C017: .equ -401
C018: .equ -799
C019: .equ -1189
C020: .equ -1567
C021: .equ -1931
C022: .equ -2276
C023: .equ -2598
C024: .equ -2896
C025: .equ -3166
C026: .equ -3406
C027: .equ -3612
C028: .equ -3784
C029: .equ -3920
C030: .equ -4017
C031: .equ -4076
<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X000</td>
<td>.set 0</td>
</tr>
<tr>
<td>X001</td>
<td>.set 2</td>
</tr>
<tr>
<td>X002</td>
<td>.set 4</td>
</tr>
<tr>
<td>X003</td>
<td>.set 6</td>
</tr>
<tr>
<td>X004</td>
<td>.set 8</td>
</tr>
<tr>
<td>X005</td>
<td>.set 0Ah</td>
</tr>
<tr>
<td>X006</td>
<td>.set 0Ch</td>
</tr>
<tr>
<td>X007</td>
<td>.set 0Eh</td>
</tr>
<tr>
<td>X008</td>
<td>.set 10h</td>
</tr>
<tr>
<td>X009</td>
<td>.set 12h</td>
</tr>
<tr>
<td>X010</td>
<td>.set 14h</td>
</tr>
<tr>
<td>X011</td>
<td>.set 16h</td>
</tr>
<tr>
<td>X012</td>
<td>.set 18h</td>
</tr>
<tr>
<td>X013</td>
<td>.set 1Ah</td>
</tr>
<tr>
<td>X014</td>
<td>.set 1Ch</td>
</tr>
<tr>
<td>X015</td>
<td>.set 1Eh</td>
</tr>
<tr>
<td>X016</td>
<td>.set 20h</td>
</tr>
<tr>
<td>X017</td>
<td>.set 22h</td>
</tr>
<tr>
<td>X018</td>
<td>.set 24h</td>
</tr>
<tr>
<td>X019</td>
<td>.set 26h</td>
</tr>
<tr>
<td>X020</td>
<td>.set 28h</td>
</tr>
<tr>
<td>X021</td>
<td>.set 2Ah</td>
</tr>
<tr>
<td>X022</td>
<td>.set 2Ch</td>
</tr>
<tr>
<td>X023</td>
<td>.set 2Eh</td>
</tr>
<tr>
<td>X024</td>
<td>.set 30h</td>
</tr>
<tr>
<td>X025</td>
<td>.set 32h</td>
</tr>
<tr>
<td>X026</td>
<td>.set 34h</td>
</tr>
<tr>
<td>X027</td>
<td>.set 36h</td>
</tr>
<tr>
<td>X028</td>
<td>.set 38h</td>
</tr>
<tr>
<td>X029</td>
<td>.set 3Ah</td>
</tr>
<tr>
<td>X030</td>
<td>.set 3Ch</td>
</tr>
<tr>
<td>X031</td>
<td>.set 3Eh</td>
</tr>
<tr>
<td>X032</td>
<td>.set 40h</td>
</tr>
<tr>
<td>X033</td>
<td>.set 42h</td>
</tr>
<tr>
<td>X034</td>
<td>.set 44h</td>
</tr>
<tr>
<td>X035</td>
<td>.set 46h</td>
</tr>
<tr>
<td>X036</td>
<td>.set 48h</td>
</tr>
<tr>
<td>X037</td>
<td>.set 4Ah</td>
</tr>
<tr>
<td>X038</td>
<td>.set 4Ch</td>
</tr>
<tr>
<td>X039</td>
<td>.set 4Eh</td>
</tr>
<tr>
<td>X040</td>
<td>.set 50h</td>
</tr>
<tr>
<td>X041</td>
<td>.set 52h</td>
</tr>
<tr>
<td>X042</td>
<td>.set 54h</td>
</tr>
<tr>
<td>X043</td>
<td>.set 56h</td>
</tr>
<tr>
<td>X044</td>
<td>.set 58h</td>
</tr>
<tr>
<td>X045</td>
<td>.set 5Ah</td>
</tr>
<tr>
<td>X046</td>
<td>.set 5Ch</td>
</tr>
</tbody>
</table>
2. GENERIC RADIX-4 COMPLEX FFT FOR THE TMS320C30 IMPLEMENTATION [P P]

*********************************************************************
********
* Name:
*    fft_4 --- radix-4 complex FFT to be called as a C function.
* *
* Synopsis:
*    int fft_4(N, M, data)
*    int N    FFT size: N=4**M
*    int M    Number of stages = log4(N)
*    float *data Array with input and output data
* *
Description:
*    Generic function to do a radix-4 FFT computation on the 320C30.
*    The data array is 2*N-long, with real and imaginary values alternating.
*    In order to have the final result in bit-reversed order, the two middle
*    branches of the radix-4 butterfly are interchanged during storage. Note
*    this difference when comparing with the program in p. 117.
*    The computation is done in place, and the original data is destroyed.
*    Bit reversal is implemented at the end of the function. If this is not
*    necessary, this part can be commented out.
*    The sine/cosine table for the twiddle factors is expected to be supplied
*    during link time, and it should have the following format:
* *
* .global _sine
* .data
* _sine .float value1 = sin(0*2*pi/N)
* _sine .float value2 = sin(1*2*pi/N)
* ..
* .float value(5N/4) = sin((5*N/4-1)*2*pi/N)
* *
* The values value1, value2, etc., are the sine wave values. For an
* N-point FFT, there are N+N/4 values for a full and a quarter period of
* the sine wave. In this way, a full sine and cosine period are available
* (superimposed).
* *
* Stack structure upon the call:
* +------------------+
* -FP(4) | data |
* -FP(3) | M |
* -FP(2) | N |
* -FP(1) | return addr |
* -FP(0) | old FP |
* +------------------+
* *
* Registers used: R0, R1, R2, R3, R4, R5, R6, R7, AR0, AR1, AR2, AR3, AR4,
* AR5, AR6, AR7, IR0, IR1, RS, RE, RC
* *
* FP .set AR3
*
*.GLOBL _fft_4 ; ENTRY POINT FOR EXECUTION
*.GLOBL _sine ; ADDRESS OF SINE TABLE
*
.BSS FFTSIZ,1
.BSS LOGFFT,1
.BSS INPUT,1
*
.TEX
.SINTAB .word _sine
*
; INITIALIZE C FUNCTION

_fft_4: PUSH FP ; SAVE DEDICATED REGISTERS
     LDI SP,FP
     PUSH R4
     PUSH R5
     PUSHF R6
     PUSHF R7
     PUSH AR4
     PUSH AR5
LPA # ZERO: INITIALIZE FFT ROUTINE

.LPA 1,1  ; FFT STAGE 
.LPA RPTCNT,1 ; REPEAT COUNTER  
.LPA IEINDX,1 ; IE INDEX FOR SINE/COSINE  
.LPA LPCNT,1 ; SECOND-LOOP COUNT  
.LPA JT,1 ; JT COUNTER IN PROGRAM, P. 117  
.LPA IA1,1 ; IA1 INDEX IN PROGRAM, P. 117

LDI @FFTSIZ,R0  
LDI @FFTSIZ,IR0  
LDI @FFTSIZ,IR1  
LDI 0,AR7  
STI AR7,@STAGE ; @STAGE HOLDS THE CURRENT STAGE NUMBER

LSH 1,IR0 ; IR0=2*N1 (BECAUSE OF REAL/IMAG)  
LSH -2,IR1 ; IR1=N/4, POINTER FOR SINE/COS TABLE  
LDI 1,AR7  
STI AR7,@RPTCNT ; INITIALIZE REPEAT COUNTER OF FIRST LOOP

LSH -2,R0  
STI AR7,@IEINDX ; INITIALIZE IE INDEX  
ADDI 2,R0  
STI R0,@JT ; JT=R0/2+2  
SUBI 2,R0  
LSH 1,R0 ; R0=N2

; OUTER LOOP

LOOP:  
LDI @INPUT,AR0 ; AR0 POINTS TO X(I)  
ADDI R0,AR0,AR1 ; AR1 POINTS TO X(I1)  
ADDI R0,AR1,AR2 ; AR2 POINTS TO X(I2)  
ADDI R0,AR2,AR3 ; AR3 POINTS TO X(I3)  
LDI @RPTCNT,RC  
SUBI 1,RC ; RC SHOULD BE ONE LESS THAN DESIRED #
RPTB BLK1
ADDF *+AR0,*+AR2,R1 ; R1=Y(I)+Y(I2)
ADDF *+AR3,*+AR1,R3 ; R3=Y(I1)+Y(I3)
ADDF R3,R1,R6 ; R6=R1+R3
SUBF *+AR2,*+AR0,R4 ; R4=Y(I)-Y(I2)
STF R6,*+AR0 ; Y(I)=R1+R3
SUBF R3,R1 ; R1=R1-R3
LDF *AR2,R5 ; R5=X(I2)
||
LDF *+AR1,R7 ; R7=Y(I1)
ADDF *AR3,*AR1,R3 ; R3=X(I1)+X(I3)
ADDF R5,*AR0,R1 ; R1=X(I)+X(I2)
||
STF R1,*+AR1 ; Y(I)=R1-R3
ADDF R3,R1,R6 ; R6=R1+R3
SUBF R5,*AR0,R2 ; R2=X(I)-X(I2)
||
STF R6,*AR0++(IR0) ; X(I)=R1+R3
SUBF R3,R1 ; R1=R1-R3
SUBF *AR3,*AR1,R6 ; R6=X(I1)-X(I3)
SUBF R7,*+AR3,R3 ; -R3=Y(I1)-Y(I3) !!!
||
STF R1,*AR1++(IR0) ; X(I1)=R1-R3
SUBF R6,R4,R5 ; R5=R4-R6
ADDF R6,R4 ; R4=R4+R6
STF R5,*+AR2 ; Y(I2)=R4-R6
||
STF R4,*+AR3 ; Y(I3)=R4+R6
SUBF R3,R2,R5 ; R5=R2-R3 !!!
ADDF R3,R2 ; R2=R2+R3 !!!
BLK1 STF R5,*AR2++(IR0) ; X(I2)=R2-R3 !!!
||
STF R2,*AR3++(IR0) ; X(I3)=R2+R3 !!!

; IF THIS IS THE LAST STAGE, YOU ARE DONE

LDI @STAGE,AR7
ADDI 1,AR7
CMPI @LOGFFT,AR7
BZD END
STI AR7,@STAGE ; CURRENT FFT STAGE

; MAIN INNER LOOP

LDI 1,AR7
STI AR7,@IA1 ; INIT IA1 INDEX
LDI 2,AR7
STI AR7,@LPCNT ; INIT LOOP COUNTER FOR INNER LOOP

INLOP:
LDI 2,AR6 ; INCREMENT INNER LOOP COUNTER
ADDI @LPCNT,AR6
LDI @LPCNT,AR0
LDI @IA1,AR7
PROGRAMS

ADDI @IEINDEX,AR7 ; IAI=IAI+IE
ADDI @INPUT,AR0 ; (X(I),Y(I)) POINTER
STI AR7,@IA1
ADDI R0,AR0,AR1 ; (X(I1),Y(I1)) POINTER
STI AR6,@LPCNT
ADDI R0,AR1,AR2 ; (X(I2),Y(I2)) POINTER
ADDI R0,AR2,AR3 ; (X(I3),Y(I3)) POINTER
LDI @RPTCNT,RC
SUBI 1,RC ; RC SHOULD BE ONE LESS THAN DESIRED #
CMPI @JT,AR6 ; IF LPCNT=JT, GO TO
BZD SPCL ; SPECIAL BUTTERFLY
LDI @IA1,AR7
LDI @IA1,AR4
ADDI @SINTAB,AR4 ; CREATE COSINE INDEX AR4
ADDI AR4,AR7,AR5
SUBI 1,AR5 ; IA2=IA1+IA1-1
ADDI AR7,AR5,AR6
SUBI 1,AR6 ; IA3=IA2+IA1-1

; SECOND LOOP

RPTB BLK2
ADDF *+AR2,*+AR0,R3 ; R3=Y(I)+Y(I2)
ADDF *+AR3,*+AR1,R5 ; R5=Y(I1)+Y(I3)
ADDF R5,R3,R6 ; R6=R3+R5
SUBF *+AR2,*+AR0,R4 ; R4=Y(I)-Y(I2)
SUBF R5,R3 ; R3=R3-R5
ADDF *AR2,*AR0,R1 ; R1=X(I)+X(I2)
ADDF *AR3,*AR1,R5 ; R5=X(I1)+X(I3)
MPYF R3,*+AR5(IR1),R6 ; R6=R3*C02
|| STF R6,*+AR0 ; Y(I)=R3+R5
ADDF R5,R1,R7 ; R7=R1+R5
SUBF *AR2,*AR0,R2 ; R2=X(I)-X(I2)
SUBF R5,R1 ; R1=R1-R5
MPYF R1,*AR5,R7 ; R7=R1*S12
|| STF R7,*AR0++(IR0) ; X(I)=R1+R5
SUBF R7,R6 ; R6=R3*C02-R1*S12
SUBF *+AR3,*+AR1,R5 ; R5=Y(I1)-Y(I3)
MPYF R1,*+AR5(IR1),R7 ; R7=R1*C02
|| STF R6,*+AR1 ; Y(I1)=R3*C02-R1*S12
MPYF R3,*AR5,R6 ; R6=R3*S12
ADDF R7,R6 ; R6=R1*C02+R3*S12
ADDF R5,R2,R1 ; R1=R2+R5
SUBF R5,R2 ; R2=R2-R5
SUBF *AR3,*AR1,R5 ; R5=X(I1)-X(I3)
SUBF R5,R4,R3 ; R3=R4-R5
ADDF R5,R4 ; R4=R4+R5
MPYF R3,*+AR4(IR1),R6 ; R6=R3*C01
|| STF R6,*AR1++(IR0) ; X(I1)=R1*C02+R3*S12
PROGRAMS

11

BLK2

MPYF R1,*AR4,R7 ; R7=R1*SI1
SUBF R7,R6 ; R6=R3*CO1-R1*SI1
MPYF R1,*+AR4(IR1),R6 ; R6=R1*CO1

||
STF R6,*+AR2 ; Y(I2)=R3*CO1-R1*SI1
MPYF R3,*AR4,R7 ; R7=R3*SI1
ADDF R7,R6 ; R6=R1*CO1+R3*SI1
MPYF R4,*+AR6(IR1),R6 ; R6=R4*CO3

||
STF R6,*AR2++(IR0) ; X(I2)=R1*CO1+R3*SI1
MPYF R2,*AR6,R7 ; R7=R2*SI3
SUBF R7,R6 ; R6=R4*CO3-R2*SI3
MPYF R2,*+AR6(IR1),R6 ; R6=R2*CO3

||
STF R6,*+AR3 ; Y(I3)=R4*CO3-R2*SI3
MPYF R4,*AR6,R7 ; R7=R4*SI3
ADDF R7,R6 ; R6=R2*CO3+R4*SI3

CMPI @LPCNT,R0
BP INLOP ; LOOP BACK TO THE INNER LOOP.
BR CONT

; SPECIAL BUTTERFLY FOR W=J

SPCL LDI IR1,AR4
LSH -1,AR4 ; POINT TO SIN(45)
ADDF @SINTAB,AR4 ; CREATE COSINE INDEX
AR4=CO21

RPTB BLK3
ADDF *AR2,*AR0,R1 ; R1=X(I)+X(I2)
SUBF *AR2,*AR0,R2 ; R2=X(I)-X(I2)
ADDF *+AR2,*+AR0,R3 ; R3=Y(I)+Y(I2)
SUBF *+AR2,*+AR0,R4 ; R4=Y(I)-Y(I2)
ADDF *AR3,*AR1,R5 ; R5=X(I1)+X(I3)
SUBF R1,R5,R6 ; R6=R5-R1
ADDF R5,R1 ; R1=R1+R5
ADDF *+AR3,*+AR1,R5 ; R5=Y(I1)+Y(I3)
SUBF R5,R3,R7 ; R7=R3-R5
ADDF R5,R3 ; R3=R3+R5
STF R3,*+AR0 ; Y(I)=R3+R5

||
STF R1,*AR0++(IR0) ; X(I)=R1+R5
SUBF *AR3,*AR1,R1 ; R1=X(I1)-X(I3)
SUBF *+AR3,*+AR1,R3 ; R3=Y(I1)-Y(I3)
STF R6,*+AR1 ; Y(I1)=R5-R1

||
STF R7,*AR1++(IR0) ; X(I1)=R3-R5
ADDF R3,R2,R5 ; R5=R2+R3
SUBF R2,R3,R2 ; R2=-R2+R3 !!!
SUBF R1,R4,R3 ; R3=R4-R1
ADDF R1,R4 ; R4=R4+R1
SUBF R5,R3,R1 ; R1=R3-R5
PROGRAMS

MPYF *AR4,R1 ; R1=R1*CO21
ADDF R5,R3 ; R3=R3+R5
MPYF *AR4,R3 ; R3=R3*CO21
|| STF R1,+AR2 ; Y(I2)=(R3-R5)*CO21
SUBF R4,R2,R1 ; R1=R2-R4 !!!
MPYF *AR4,R1 ; R1=R1*CO21
|| STF R3,*AR2++(IR0) ; X(I2)=(R3+R5)*CO21
ADDF R4,R2 ; R2=R2+R4 !!!
MPYF *AR4,R2 ; R2=R2*CO21 !!!
BLK3 STF R1,+AR3 ; Y(I3)=-(R4-R2)*CO21 !!!
|| STF R2,*AR3++(IR0) ; X(I3)=(R4+R2)*CO21 !!!

CMP @LPCNT,R0
BPD INLOP ; LOOP BACK TO THE INNER LOOP.

CONT LDI @RPTCNT,AR7
LDI @IEINDX,AR6
LSH 2,AR7 ; INCREMENT REPEAT COUNTER FOR NEXT TIME

STI AR7,@RPTCNT
LSH 2,AR6 ; IE=4*IE
STI AR6,@IEINDX
LDI R0,IR0 ; N1=N2
LSH -3,R0
ADDI 2,R0
STI R0,@JT ; JT=N2/2+2
SUBI 2,R0
LSH 1,R0 ; N2=N2/4
BR LOOP ; NEXT FFT STAGE

; DO THE BIT-REVERSING OF THE OUTPUT

END: LDI @FFTSIZ,RC ; RC=N
SUBI 1,RC ; RC SHOULD BE ONE LESS THAN DESIRED #
LDI @FFTSIZ,IR0 ; IR0=SIZE OF FFT=N
LDI @INPUT,AR0
LDI @INPUT,AR1

RPTB BITRV
CMP AR0,AR1
BGE CONT
LDF *AR0,R0
|| LDF *AR1,R1
STF R0,*AR1
|| STF R1,*AR0
LDF *AR0(1),R0
|| LDF *AR1(1),R1
STF R0,*AR1(1)
FUNCTIONS FOR THE TMS320C30

EXPONENTIAL FUNCTION: R0 <= EXP(R0).

APPROXIMATE ACCURACY: 7 DECIMAL DIGITS.
INPUT RESTRICTIONS: |R0| <= 88.0.
REGISTERS FOR INPUT: R0.
REGISTERS USED AND RESTORED: DP AND SP.
REGISTERS ALTERED: R0-4.
REGISTERS FOR OUTPUT: R0.
ROUTINES NEEDED: FPINV.
EXECUTION CYCLES (MIN, MAX): 44 (R0 <= 0), 70.

EXTERNAL PROGRAM NAMES

.GLOBL EXP
.GLOBL FPINV

INTERNAL CONSTANTS

.DATA

; SCALING COEFF. FOR 2**-X
ENRM .FLOAT 1.442695041 ; 1/LN(2)

; POLYNOMIAL COEFFS. FOR 2**-X, 0 <= X < 1.

.FLOAT 1.00000000000 ; C0
.FLOAT -0.693147180 ; C1
.FLOAT 0.240226469 ; C2
.FLOAT -0.055503654 ; C3
.FLOAT 0.009615978 ; C4
.FLOAT -0.001328240 ; C5
.FLOAT 0.000147491 ; C6
C7 .FLOAT -0.000010863 ; C7

AC7 .WORD C7

.TEXT

; START OF EXP PROGRAM

EXP:

; SCALE VARIABLE X

PUSH DP ; SAVE DP
LDP @AC7 ; LOAD DATA PAGE POINTER
RND R0 ; ROUND X
NEG R0,R2 ; R2 <= -X
LDF R0,R1 ; R1 <= X
LDFN R2,R1 ; IF X < 0 THEN R1 <= |X|
MPYF @ENRM,R1 ; R1 <= X = |X|/LN(2)
FIX R1,R3 ; R3 <= |X| = INTEGER OF X
FLOAT R3,R0 ; R0 <= FLT. PT. I
SUBF R0,R1 ; R1 <= FRACTION OF |X|, 0 <= X < 1
NEGI R3 ; R3 <= -I
LSH 24,R3 ; MOVE -I TO EXP.
PUSH R3 ; SAVE AS INT.
POPF R3 ; R3 <= FLT. PT. 2**-I
LDI @AC7,AR0 ; AR0 -> COEFF. TABLE
POP DP ; UNSAVE DP

; EVALUATE TRUNCATED SERIES

RND R1 ; ROUND BEFORE *
MPYF *AR0-,R1,R0 ; R0 <= X*C7
ADDF *AR0-,R0 ; R0 <= C6 + R0

MPYF R1,R0 ; R0 <= X*(C6 + R0)
ADDF *AR0-,R0 ; R0 <= C5 + R0

MPYF R1,R0 ; R0 <= X*(C5 + R0)
ADDF *AR0-,R0 ; R0 <= C4 + R0

MPYF R1,R0 ; R0 <= X*(C4 + R0)
ADDF *AR0-,R0 ; R0 <= C3 + R0

RND R0 ; ROUND BEFORE *
MPYF R1,R0 ; R0 <= X*(C3 + R0)
ADDF *AR0-,R0 ; R0 <= C2 + R0

RND R0 ; ROUND BEFORE *
MPYF R1,R0 ; R0 <= X*(C2 + R0)
ADDF *AR0-,R0 ; R0 <= C1 + R0
PROGRAMS

RND R0 ; ROUND BEFORE *
MPYF R1,R0 ; R0 <= X*(C1 + R0)

; TEST FOR X < 0 AND RETURN
LDF R2,R2 ; TEST ORIGINAL -X
BND FPINV ; IF -X < 0 THEN R0 <= 1/X, (DELAYED)
ADDF *AR0,R0 ; R0 <= 2**-X = C0 + R0
RND R0 ; ROUND BEFORE *
MPYF R3,R0 ; R0 <= 2**(l + X)
RETS ; RETURN (IF NO FPINV BRANCH)

PROGRAM: LN

LOGARITHM FUNCTION BASE E: R0 <= LN(R0).

APPROXIMATE ACCURACY: 7 DECIMAL DIGITS.
INPUT RESTRICTIONS: R0 > 0.0.
REGISTERS FOR INPUT: R0.
REGISTERS USED AND RESTORED: DP AND SP.
REGISTERS ALTERED: AR0 AND R0-3.
REGISTERS FOR OUTPUT: R0.
ROUTINES NEEDED: NONE.
EXECUTION CYCLES (MIN, MAX): 43 , 43.

; EXTERNAL PROGRAM NAMES
.GLOBL LN

; INTERNAL CONSTANTS
.DATA

; SCALING COEFFS. FOR LN(1+X)
LNRM .FLOAT 0.6931471806 ; LN(2)
C0 .FLOAT 1.0000000000 ; C0 (1.0)

; POLYNOMIAL COEFFS. FOR LN(1+X), 0 <= X < 1.
.FLOAT 0.9999964239 ; TOP OF C1
.FLOAT -0.4998741238 ; TOP OF C2
.FLOAT 0.3317990258 ; TOP OF C3
.FLOAT -0.2407338084 ; TOP OF C4
.FLOAT 0.1676540711 ; TOP OF C5
.FLOAT -0.0953293897 ; TOP OF C6
.FLOAT 0.0360884937; TOP OF C7
C8 .FLOAT -0.0064535442; TOP OF C8

AC8 .WORD C8

.DOWN

; START OF LN PROGRAM

LN:

LDF R0,R0 ; TEST X
RETSLE ; RETURN NOW IF X <= 0

; SCALE VARIABLE X

PUSH DP ; SAVE DP
LDP @AC8 ; LOAD DATA PAGE POINTER
PUSHF R0 ; SAVE AS FLT. PT.
POP R3 ; R3 <= INTEGER FORMAT
ASH -24,R3 ; R3 <= E = SIGNED EXP.
FLOAT R3,R1 ; R1 <= FLT. PT. E VALUE
LDF @C0,R2 ; R2 <= 1.0
LDE R2,R0 ; EXP. R0 <= 0 (1 <= X < 2)
SUBRF R0,R2 ; R2 <= X - 1 (0 <= X < 1)
LDF @LNRM,R0 ; R0 <= LN(2)
MPYF R1,R0 ; R0 <= E*LN(2)
LDF R0,R3 ; R3 <= E*LN(2)
LDI @AC8,AR0 ; AR0 -> COEFF. TABLE
POP DP ; UNSAVE DP

; EVALUATE TRUNCATED SERIES

RND R2,R1 ; R1 <= RND X
MPYF *AR0--,R1,R0 ; R0 <= X*C8
ADDF *AR0--,R0 ; R0 <= C7 + R0

MPYF R1,R0 ; R0 <= X*(C7 + R0)
ADDF *AR0--,R0 ; R0 <= C6 + R0

MPYF R1,R0 ; R0 <= X*(C6 + R0)
ADDF *AR0--,R0 ; R0 <= C5 + R0

MPYF R1,R0 ; R0 <= X*(C5 + R0)
ADDF *AR0--,R0 ; R0 <= C4 + R0

MPYF R1,R0 ; R0 <= X*(C4 + R0)
ADDF *AR0--,R0 ; R0 <= C3 + R0

RND R0 ; ROUND BEFORE *
MPYF R1,R0 ; R0 <= X*(C3 + R0)
ADDF *AR0--,R0 ; R0 <= C2 + R0
PROGRAMS

RND  R0  ; ROUND BEFORE *
MPYF R1,R0  ; R0 <= X*(C2 + R0)
ADDF *AR0--,R0  ; R0 <= C1 + R0

; ADD IN SCALED EXPONENT AND RETURN

POP  R2  ; R2 <= RETURN ADDRESS
BUD  R2  ; RETURN (DELAYED)
RND  R0  ; ROUND BEFORE *
MPYF R1,R0  ; R0 <= X*(C1 + R0)
ADDF R3,R0  ; R0 <= LN(X) + E*LN(2)

PROGRAM: SQRT

SQUARE ROOT FUNCTION: R0 <= SQRT(R0).

APPROXIMATE ACCURACY: 8 DECIMAL DIGITS.
INPUT RESTRICTIONS: R0 >= 0.0.
REGISTERS FOR INPUT: R0.
REGISTERS USED AND RESTORED: DP AND SP.
REGISTERS ALTERED: R0-4.
REGISTERS FOR OUTPUT: R0
ROUTINES NEEDED: NONE.
EXECUTION CYCLES (MIN, MAX): 49 , 49.

; EXTERNAL PROGRAM NAMES
.GLOBL  SQRT

; INTERNAL CONSTANTS
.DATA
CNST1 .SET 0.5
CNST2 .SET 1.5
CNST3 .FLOAT 1.103553391 ; ADJUSTED 1.0
CNST4 .FLOAT 0.780330086 ; ADJUSTED SQRT(1/2)
SMSK .WORD 0FF7FFFFFH

.TEXT
; START OF SQRT PROGRAM.

SQRT:

LDF  R0,R3  ; TEST AND SAVE V
RETSLE  ; RETURN NOW IF V <= 0
GET APPROXIMATION TO $1/V$. FOR $V = (1+M)2^{*E}$
AND $0 <= M < 1$, FOR $E$ EVEN: $X[0] = (1-M/2)^*2^{*E}/2$
AND FOR $E$ ODD: $X[0] = \sqrt{1/2}(1-M/2)^*2^{*E}/2$

PUSH DP ; SAVE DP
LDP @SMSK ; LOAD DATA PAGE POINTER
PUSHF R0 ; SAVE V AS FLT. PT. $V = (1+M)2^{*E}$
POP R2 ; R2 <= V AS INTEGER
XOR @SMSK,R2 ; R2 <= COMPLEMENT ALL BUT SIGN
LDI R2,R1 ; R1 <= (1-M/2)^*2**-E
LDI R2,R4 ; R4 <= R1
LSH 8,R1 ; R1 <= R1 EXP. REMOVED
ASH -1,R2 ; R2 <= R2 WITH -E/2 EXP.
PUSH R2 ; SAVE R2 AS INTEGER
POPF R2 ; R2 <= FLT. PT.
LDE R2,R1 ; R1 <= (1-M/2)^*2**-E/2
LDF @CNST3,R2 ; R2 <= 1.1... FOR ODD E
LSH 7,R4 ; TEST LSB OF E (AS SIGN)
LDFNN @CNST4,R2 ; IF E EVEN R2 <= 0.78...
MPYF R2,R1 ; R1 <= CORRECTED ESTIMATE
POP DP ; UNSAVE DP

GENERATE $V/2$ (USES MPYF).

MPYF CNST1,R0 ; R0 <= $V/2$ TRUNC.
RND R0 ; R0 <= RND $V/2$

NEWTON ITERATION FOR $Y(X) = X - V**-2 = 0$

MPYF R1,R1,R2 ; R2 <= $X[0]**2$
MPYF R0,R2 ; R2 <= $(V/2)^*X[0]**2$
SUBRF CNST2,R2 ; R2 <= 1.5 - $(V/2)^*X[0]**2$
MPYF R2,R1 ; R1 <= $X[1] = X[0] * (1.5 - (V/2)^*X[0]**2)$

MPYF R1,R1,R2 ; R2 <= $X[1]**2$
MPYF R0,R2 ; R2 <= $(V/2)^*X[1]**2$
SUBRF CNST2,R2 ; R2 <= 1.5 - $(V/2)^*X[1]**2$
MPYF R2,R1 ; R1 <= $X[2] = X[1] * (1.5 - (V/2)^*X[1]**2)$

MPYF R1,R1,R2 ; R2 <= $X[2]**2$
MPYF R0,R2 ; R2 <= $(V/2)^*X[2]**2$
SUBRF CNST2,R2 ; R2 <= 1.5 - $(V/2)^*X[2]**2$
MPYF R2,R1 ; R1 <= $X[3] = X[2] * (1.5 - (V/2)^*X[2]**2)$

RND R1 ; ROUND BEFORE *
MPYF R1,R1,R2 ; R2 <= $X[3]**2$
RND R2 ; ROUND BEFORE *
MPYF R0,R2 ; R2 <= $(V/2)^*X[3]**2$
SUBRF CNST2,R2 ; R2 <= 1.5 - $(V/2)^*X[3]**2$
RND R2 ; ROUND BEFORE *
MPYF R2,R1 ; R1 <= $X[4] = X[3] * (1.5 - (V/2)^*X[3]**2)$
INVERT FINAL RESULT AND RETURN

POP  R2  ; R2 <= RETURN ADDRESS
BUD  R2  ; RETURN (DELAYED)
RND  R3  ; ROUND BEFORE *
RND  R1  ; ROUND BEFORE *
MPYF R1,R3,R0  ; R0 = SQRT(V) = V*SQRT(1/V)

PROGRAM: FPINV

FLOATING POINT INVERSE: R0 <= 1/R0

APPROXIMATE ACCURACY: 8 DECIMAL DIGITS.
INPUT RESTRICTIONS: R0 != 0.0.
REGISTERS FOR INPUT: R0.
REGISTERS USED AND RESTORED: DP AND SP.
REGISTERS ALTERED: R0-2 AND R4.
REGISTERS FOR OUTPUT: R0.
ROUNDTINES NEEDED: NONE.
EXECUTION CYCLES (MIN, MAX): 33 , 33.

; EXTERNAL PROGRAM NAMES
.GLOBL FPINV

; INTERNAL CONSTANTS
.DATA
ONE .SET 1.0
TWO .SET 2.0
MSK .WORD 0FF7FFFFFH

.TEXT

; START OF FPINV PROGRAM

FPINV:

LDF  R0,R0  ; TEST F
RETSZ  ; RETURN NOW IF F = 0

; GET APPROXIMATION TO 1/F. FOR F = (1+M) * 2**E
AND 0 <= M < 1, USE: \( X[0] = (1-M/2) \times 2^{-E} \)

```
PUSH DP ; SAVE DATA PAGE POINTER
LDP @MSK ; LOAD DATA PAGE POINTER
PUSHF R0 ; SAVE AS FLT. PT. F = (1+M) \times 2^E
POP R1 ; FETCH BACK AS INTEGER
XOR @MSK,R1 ; COMPLEMENT E & M BUT NOT SIGN BIT
PUSH R1 ; R1 <= X[0] = (1-M/2) \times 2^{-E}.
POP DP ; UNSAVE DP
```

**NEWTON ITERATION FOR: \( Y(X) = X - 1/F = 0 \) ...**

```
MPYF R1,R0,R4 ; R4 <= F \times X[0]
SUBRF TWO,R4 ; R4 <= 2 - F \times X[0]
MPYF R4,R1 ; R1 <= X[1] = X[0] \times (2 - F \times X[0])
```

```
MPYF R1,R0,R4 ; R4 <= F \times X[1]
SUBRF TWO,R4 ; R4 <= 2 - F \times X[1]
MPYF R4,R1 ; R1 <= X[2] = X[1] \times (2 - F \times X[1])
```

```
MPYF R1,R0,R4 ; R4 <= F \times X[2]
SUBRF TWO,R4 ; R4 <= 2 - F \times X[2]
MPYF R4,R1 ; R1 <= X[3] = X[2] \times (2 - F \times X[2])
```

```
```

```
RND R0,R4 ; ROUND F BEFORE LAST MULTIPLY
RND R1,R0 ; ROUND X[3] BEFORE MULTIPLIES
MPYF R0,R4 ; R4 <= F \times X[3] = 1 + EPS
```

**FINISH ITERATION AND RETURN**

```
POP R2 ; R2 <= RETURN ADDRESS
BUD R2 ; RETURN (DELAYED)
SUBRF ONE,R4 ; R4 <= 1 - F \times X[3] = EPS
MPYF R0,R4 ; R4 <= X[3] \times EPS
ADDF R4,R1,R0 ; R0 <= X[4] = (X[3] \times (1 - (F \times X[3]))) + X[3]
```

**PROGRAM: FDIV**

**FLOATING POINT DIVIDE FUNCTION:** \( R0 <= R0/R1 \).

**APPROXIMATE ACCURACY:** 8 DECIMAL DIGITS.

**INPUT RESTRICTIONS:** \( R1 \neq 0.0 \).

**REGISTERS FOR INPUT:** R0 (DIVIDEND) AND R1 (DIVISOR).

**REGISTERS USED AND RESTORED:** DP AND SP.
PROGRAMS

REGISTERS ALTERED: R0-4.
REGISTERS FOR OUTPUT: R0 (QUOTIENT).
ROUTINES NEEDED: FPINV.
EXECUTION CYCLES (MIN, MAX): 43 , 43.

; EXTERNAL PROGRAM NAMES

.GLOBL FDIV
.GLOBL FPINV

.TEXT

; START OF FDIV PROGRAM

FDIV:

RND R0,R3 ; R3 <= RND X
LDF R1,R0 ; R1 <= Y
CALL FPINV ; R0 <= 1/Y
RND R0 ; ROUND BEFORE *
MPYF R3,R0 ; R0 <= X

RETS ; RETURN

.END

FIR Filter

;===============================================
;===============================================
; SUBROUTINE FIR
; EQUATION: y(n) = h(0) * x(n) + h(1) * x(n-1) + ... + h(N-1) * x(n-(N-1))
; TYPICAL CALLING SEQUENCE:
; load AR0
; load AR1
; load RC
; load BK
; CALL FIR
; ; ARGUMENT ASSIGNMENTS:
; argument | function
PROGRAMS

---+----------~--
AR0 | address of h(N-1)
AR1 | address of x(N-1)
RC  | length of filter - 2 (N-2)
BK  | length of filter (N)

; REGISTERS USED AS INPUT: AR0, AR1, RC, BK
; REGISTERS MODIFIED: R0, R2, AR0, AR1, RC
; REGISTER CONTAINING RESULT: R0

; PROGRAM SIZE: 6 words

; EXECUTION CYCLES: 11 + (N-1)

;===============================================================================
; .global  FIR
; ; initialize R0:
FIR  MPYF3 *AR0++,*AR1++,%R0; h(N-1) * x(n-(N-1)) -> R0
    LDF  0.0,R2 ; initialize R2.

; filter (1 <= i < N)
    RPTS  RC ; setup the repeat single.
    MPYF3 *AR0++,*AR1++,%R0; h(N-1-i) * x(n-(N-1-i)) -> R0
    ADDF3 R0,R2,R2 ; multiply and add operation
    ADDF  R0,R2,R0 ; add last product

; return sequence
    RETS ; return

; end

; .end

IIR Filters (N biquads)

;===============================================================================
; SUBROUTINE IIR R 2
;===============================================================================
; IIR2 == IIR FILTER (N > 1 BIQUADS)
EQUATIONS: $y(0,n) = x(n)$

for $i = 0; i < N; i++$

\{
\begin{align*}
  d(i,n) &= a2(i) \times d(i,n-2) + a1(i) \times d(i,n-1) + y(i-1,n) \\
  y(i,n) &= b2(i) \times d(i,n-2) + b1(i) \times d(i,n-1) + b0(i) \times d(i,n)
\end{align*}
\}

$y(n) = y(N-1,n)$

TYPICAL CALLING SEQUENCE:

load R2
load AR0
load AR1
load IR0
load IR1
load BK
load RC
CALL IIR2

ARGUMENT ASSIGNMENTS:

argument | function
\begin{align*}
  R2 & | \text{input sample } x(n) \\
  AR0 & | \text{address of filter coefficients } (a2(0)) \\
  AR1 & | \text{address of delay node values } (d(0,n-2)) \\
  BK & | BK = 3 \\
  IR0 & | IR0 = 4 \\
  IR1 & | IR1 = 4*N-4 \\
  RC & | \text{Number of biquads } (N) - 2
\end{align*}

REGISTERS USED AS INPUT: R2, AR0, AR1, IR0, IR1, BK, RC

REGISTERS MODIFIED: R0, R1, R2, AR0, AR1, RC

REGISTER CONTAINING RESULT: R0

PROGRAM SIZE: 17 words

EXECUTION CYCLES: $23 + 6N$
MPYF3  *++AR0(1), R2, R2 ; b0(0) * d(0,n) -> R2
STF R2, *AR1--(1)% ; store d(0,n); point to d(0,n-2).

RPTB LOOP ; loop for 1 <= i < N

MPYF3  *++AR0(1), *++AR1(1R0), R0 ; a2(i) * d(i,n-2) -> R0
ADDF3 R0,R2,R2 ; first sum term of y(i-1,n)

MPYF3  *++AR0(1), *AR1--(1)% , R1 ; b2(i) * d(i,n-2) -> R1
ADDF3 R1,R2,R2 ; second sum term of y(i-1,n)

MPYF3  *++AR0(1), *AR1, R0 ; a1(i) * d(i,n-1) -> R0
ADDF3 R0, R2, R2 ; first sum term of d(i,n).

MPYF3  *++AR0(1), *AR1--(1)% , R0 ; b1(i) * d(i,n-1) -> R0
ADDF3 R0, R2, R2 ; second sum term of d(i,n).
STF R2, *AR1--(1)% ; store d(i,n); point to d(i,n-2).

LOOP MPYF3  *++AR0(1), R2, R2 ; b0(i) * d(i,n) -> R2

ADDF R0,R2 ; first sum term of y(N-1,n)
ADDF3 R1,R2,R0 ; second sum term of y(N-1,n)
NOP *AR1--(1R1) ; return to first biquad
NOP *AR1--(1)% ; point to d(0,n-1)

RET ; return

.end
# REFERENCES

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>Matlab function</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNL</td>
<td>J.N.Little, The MathWorks, Inc., 86, Revised 88 JNL</td>
</tr>
<tr>
<td>CRD</td>
<td>C.R.Denham, The MathWorks, Inc., 88</td>
</tr>
<tr>
<td>PCH</td>
<td>P.C.Hansen, UNI-C, 04/11/90.</td>
</tr>
<tr>
<td>JS-MC</td>
<td>John.So, Manfred Christ, Texas Instruments, 88</td>
</tr>
<tr>
<td>PP</td>
<td>Panos Papamichalis, Texas Instruments, 87/88</td>
</tr>
</tbody>
</table>