1

Micred
MICRO-RECORD

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**CONVERGENCE BEHAVIOUR OF
A JOINTLY-ADAPTIVE TRANSVERSAL AND
MEMORY-BASED ECHO CANCELLER**

by
KEN YAMAZAKI, B.ENG.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
MASTER OF ENGINEERING

Ottawa-Carleton Institute For Electrical Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario

September 20, 1989

© 1989 Ken Yamazaki

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-54362-0

Canada

Thesis Acceptance Form
M.ENG CANDIDATE

The undersigned recommend to the
Ottawa-Carleton Institute for Electrical Engineering
acceptance of the thesis

"CONVERGENCE BEHAVIOUR OF
A JOINTLY-ADAPTIVE TRANSVERSAL AND
MEMORY BASED DIGITAL ECHO CANCELLER"

submitted by **Ken Yamazaki, B.ENG.**
in partial fulfillment of the requirements for
the degree of **Master of Engineering**

Thesis Supervisor

Co-Supervisor

Chair, Department of

Carleton University

89/10/10

ABSTRACT

A theoretical convergence analysis and simulation of a jointly-adaptive transversal and memory-based or "table-lookup" echo canceller is presented. The coefficients of the transversal echo canceller (TEC) and the memory echo canceller (MEC) are updated simultaneously using the well-known LMS algorithm with separate adaptation constants and a common cancellation error. The transient and steady state values of the coefficients and the average mean square error (AMSE) over time are examined and compared to those from an identical structure employing a "sequential" adaptation method where the TEC coefficients are adapted first and permitted to converge before the MEC coefficients are adapted. The intent of this investigation is to identify the advantages and/or disadvantages of joint coefficient adaptation of the TEC/MEC combination when considering factors such as convergence time, steady state AMSE and stability.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to my supervisors Dr. Dave Falconer and Dr. Sami Aly for their invaluable guidance and support and the members of Dept. 1T23 at Bell Northern Research (BNR) for providing me with the state-of-the-art facilities used in my research. I would individually like to thank Dr. Andrew Deczky, Majid Taheri, Babak Sayar, Paul Hung, Osama Baghat, Luis Viriato, and Eugene Kirkham. I would also like to thank Dr. Sasan Ardalan, Dr. Jim Faber and Ramin Nobahkt of the CAPSIM development group at North Carolina State University for their personalized support of the CAPSIM simulator installation at BNR. A special thanks to another "M.Eng.-to-be" and experienced CAPSIM user, Mirna Hage, for her invaluable comments about the "idiosyncrasies" of the simulation environment during its infancy.

TABLE OF CONTENTS

Thesis Acceptance Form.....	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS	v
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF APPENDICES.....	xiv
CHAPTER 1: INTRODUCTION	1 6
1.1 INTRODUCTION	1 6
1.2 THE COMBINED TEC/MEC CONFIGURATION	2 4
1.3 SUMMARY	2 9
CHAPTER 2: ANALYSIS OF THE JOINTLY-ADAPTIVE TEC/MEC	3 1
2.1 THE LMS ALGORITHM FOR JOINT ADAPTATION	3 1
2.2 AVERAGE MEAN SQUARE ERROR	3 6
2.2.1 MINIMUM MEAN SQUARE ERROR	3 7
2.2.1.1 THE ERROR SURFACE.....	3 7
2.2.1.2 CONDITIONS FOR MINIMUM MEAN SQUARE ERROR	4 3
2.2.1.3 COEFFICIENT TRAJECTORY ANALYSIS.....	4 6
2.2.1.4 AN EXPRESSION FOR MINIMUM MEAN SQUARE ERROR	6 0
2.2.2 MEAN SQUARE ERROR.....	6 3
2.2.3 AVERAGE EXCESS MEAN SQUARE ERROR.....	6 5
2.2.3.1 COEFFICIENT ERROR POWERS.....	6 6
2.2.3.2 AN EXPRESSION FOR AVERAGE EXCESS MEAN SQUARE ERROR	7 6

2.2.4	AN EXPRESSION FOR AVERAGE MEAN SQUARE ERROR	77
2.3	CONVERGENCE BEHAVIOUR	78
2.3.1	MEAN COEFFICIENT CONVERGENCE	78
2.3.2	AVERAGE MEAN SQUARE ERROR CONVERGENCE	86
2.3.2.1	TRANSIENT AMSE CHARACTERISTICS	87
2.3.2.2	STEADY STATE AMSE CHARACTERISTICS	102
2.3.2.3	ACCURACY OF THE THEORETICAL AMSE EXPRESSION	104
2.4	SUMMARY	111
CHAPTER 3: ANALYSIS OF THE SEQUENTIALLY-ADAPTIVE TEC/MEC		
3.1	THE LMS ALGORITHM FOR SEQUENTIAL ADAPTATION	115
3.2	AVERAGE MEAN SQUARE ERROR	117
3.2.1	MINIMUM MEAN SQUARE ERROR	117
3.2.1.1	THE ERROR SURFACE.....	117
3.2.1.2	CONDITIONS FOR MINIMUM MEAN SQUARE ERROR	119
3.2.1.3	COEFFICIENT TRAJECTORY ANALYSIS	119
3.2.1.4	EXPRESSIONS FOR MINIMUM MEAN SQUARE ERROR	123
3.2.2	MEAN SQUARE ERROR.....	124
3.2.3	AVERAGE EXCESS MEAN SQUARE ERROR.....	125
3.2.3.1	COEFFICIENT ERROR POWERS.....	125
3.2.3.2	EXPRESSIONS FOR AVERAGE EXCESS MEAN SQUARE ERROR	129
3.2.3	EXPRESSIONS FOR AVERAGE MEAN SQUARE ERROR	129
3.3	CONVERGENCE BEHAVIOUR	131
3.3.1	MEAN COEFFICIENT CONVERGENCE.....	133
3.3.2	AVERAGE MEAN SQUARE ERROR CONVERGENCE	134

3.3.2.1	TRANSIENT AMSE CHARACTERISTICS	134
3.3.2.2	STEADY STATE AMSE CHARACTERISTICS	150
3.4	SUMMARY	153
CHAPTER 4:	A COMPARISON OF THE JOINT AND SEQUENTIAL COEFFICIENT ADAPTATION APPROACHES	157
4.1	COEFFICIENT CONVERGENCE	157
4.2	AVERAGE MEAN SQUARE ERROR CONVERGENCE	158
4.3	STEADY STATE AVERAGE MEAN SQUARE ERROR	161
4.4	SUMMARY	162
CHAPTER 5:	CONCLUSIONS	163
APPENDIX A:	AVERAGE MEAN SQUARE ERROR DETAILS	167
A.1	THE JOINT INPUT AUTOCORRELATION MATRIX	167
A.2	THE ECHO PATH LINEAR/NONLINEAR INPUT CROSSCORRELATION MATRIX	169
A.3	COEFFICIENT ERROR POWERS	172
A.4	DESIGN OF THE LINEAR/NONLINEAR ECHO PATH MODEL	182
A.5	SOFTWARE FOR THEORETICAL CALCULATIONS	184
A.5.1	AMSE CALCULATION	184
A.5.2	REGION OF CONVERGENCE CALCULATION	215
A.5.3	OPTIMUM ADAPTATION STEP-SIZE DETERMINATION FOR FASTEST AMSE CONVERGENCE	219
A.5.4	EVALUATION OF FACTORS AFFECTING STEADY STATE AMSE LEVELS	222
APPENDIX B:	CAPSIM SIMULATIONS	227
B.1	A BRIEF INTRODUCTION TO CAPSIM	227
B.1.1	A BRIEF HISTORY OF CAPSIM	228
B.1.2	BUILDING BLOCKS	229
B.1.2.1	BLOCK CONNECTIONS	232
B.1.2.2	THE STAR	234
B.1.2.2.1	THE STAR DEFINITION	236
B.1.2.2.2	THE STAR LIBRARY	243

B.1.2.3	THE GALAXY	244
B.1.2.4	THE UNIVERSE	250
B.1.3	SIMULATION EXECUTION	250
B.2	SIMULATION TOPOLOGIES	253
B.2.1	THE JOINTLY-ADAPTIVE TEC/MEC UNIVERSE	253
B.2.1.1	THE JOINTLY-ADAPTIVE TEC/MEC GALAXY	258
B.2.2	THE SEQUENTIALLY-ADAPTIVE TEC/MEC UNIVERSE	261
B.2.2.1	THE SEQUENTIALLY-ADAPTIVE TEC/MEC GALAXY.....	264
B.3	STAR SOURCE FILES.....	267
APPENDIX C:	GLOSSARY	305
C.1	TEXT CONVENTIONS	305
C.2	ABBREVIATIONS	306
C.3	PRINCIPAL SYMBOLS	307
REFERENCES		316

LIST OF TABLES

TABLE 1.1	A mapping between the m -baud symbol vector $\mathbf{x}'(k)$ and MEC address vector $\mathbf{a}(k)$	27
TABLE 2.1	Results of the testing of the outer ROC boundary defined by $n\alpha + \beta = 2$ for $n = 30$ and $m = 5$	94
TABLE 2.2	Computed adaptation step-sizes for optimum convergence time and maximum λ_e for $n = 16, 20, 25, 30$ and $m = 3, 4, 5, 6$	98
TABLE 2.3	Theoretical and simulated AMSE_{ss} values for $c = 0.5, 1, 1.5$	104
TABLE 3.1	Results of testing of the outer phase II ROC boundary defined by $(n-m)\alpha_1 + \beta_1 = 2$ for $n = 30$ and $m = 5$	139
TABLE 3.2	Computed adaptation step-sizes for optimum Phase II convergence time and maximum λ_{se} for $n = 16, 20, 25, 30$ and $m = 3, 4, 5, 6$	144
TABLE 3.3	Theoretical and simulated phase II AMSE_{ss} values for $c = 0.5, 1, 1.5$	153
TABLE C.1	Abbreviations.....	306
TABLE C.2	Principal Symbols.....	308

LIST OF FIGURES

Fig. 1.1	Typical transceiver and echo canceller arrangement.....	18
Fig. 1.2	The Combined TEC/MEC Configuration.....	25
Fig. 2.1	A comparison between theoretical asymptotic mean TEC coefficient values and an actual snap shot of the TEC coefficients after convergence.....	80
Fig. 2.2	A comparison between theoretical asymptotic mean MEC coefficient values and an actual snap shot of the MEC coefficients after convergence.....	81
Fig. 2.3	TEC coefficients after convergence illustrating the sharing of the cancellation load over the first $m = 5$ bauds of the linear echo.....	84
Fig. 2.4	MEC coefficients after convergence illustrating the sharing of the cancellation load over the first $m = 5$ bauds of the linear echo.....	85
Fig. 2.5	A contour plot of the smallest eigenvalue of T	90
Fig. 2.6	A contour plot of the largest eigenvalue of T	91
Fig. 2.7	AMSE convergence sensitivity to the choice of adaptation step-sizes near the ROC boundary.....	95
Fig. 2.8	Verifying the optimum step-size for fastest convergence for $n = 30$ and $m = 5$	101
Fig. 2.9	A comparison between the theoretical and simulated $AMSE(k)$ for $n = 30$, $m = 5$, and no additive noise.....	106
Fig. 2.10	A comparison between the theoretical and simulated $AMSE(k)$ for $n = 30$, $m = 5$, and -50 dB additive independent white noise introduced into the echo signal.....	107

Fig. 2.11	A comparison between the theoretical and simulated AMSE(k) for $n = 16$, $m = 4$, and no additive noise.....	109
Fig. 2.12	A comparison between the theoretical and simulated AMSE(k) for $n = 16$, $m = 4$, and -50 dB additive independent white noise introduced into the echo signal.....	110
Fig. 3.1	A contour plot of the smallest eigenvalue of T_s	136
Fig. 3.2	A contour plot of the largest eigenvalue of T_s	137
Fig. 3.3	Phase II AMSE convergence sensitivity to the choice of adaptation step-sizes near the ROC boundary.....	141
Fig. 3.4	Phase I AMSE divergence with phase II AMSE convergence.....	142
Fig. 3.5	Phase I and phase II AMSE divergence.....	143
Fig. 3.6	Verifying the optimum step-size for fastest phase II AMSE convergence for $n = 30$ and $m = 5$	147
Fig. 3.7	The effect of the use of single vs dual TEC adaptation step-sizes on the overall convergence time of the Sequentially-Adaptive TEC/MEC.....	149
Fig. 4.1	A comparison of the fastest converging Jointly-Adaptive and Sequentially-Adaptive TEC/MEC.....	160
Fig. A.1	The GETAMSEJ script file.....	192
Fig. A.2	The BGEN function file.....	193
Fig. A.3	The BSGEN function file.....	194
Fig. A.4	The EDK2GEN function file.....	196
Fig. A.5	The FMGEN function file.....	197
Fig. A.6	The FNGEN function file.....	198

Fig. A.7	The FPHIGEN function file.....	199
Fig. A.8	The MMSEGEN function file.....	200
Fig. A.9	The MOPTGEN function file.....	201
Fig. A.10	The PDPGEN function file.....	203
Fig. A.11	The PPGEN function file.....	205
Fig. A.12	The QGEN function file.....	207
Fig. A.13	The SPOPTGEN function file.....	208
Fig. A.14	The TGEN function file.....	210
Fig. A.15	The TSGEN function file.....	211
Fig. A.16	The YGEN function file.....	213
Fig. A.17	The ZOGEN function file.....	215
Fig. A.18	The EIGTGEN function file.....	217
Fig. A.19	The EIGTSGEN function file.....	219
Fig. A.20	The OPTIMUM function file.....	220
Fig. A.21	OPTIMUMS function file.....	221
Fig. A.22	The JNTXSGEN function file.....	223
Fig. A.23	The SEQXSGEN function file.....	224
Fig. A.24	The TIBGEN function file.....	225
Fig. A.25	The TISBSGEN function file.....	226
Fig. B.1	An example of a CAPSIM universe.....	231
Fig. B.2	An exploded view of the FILTER galaxy.....	232

Fig. B.3	An interblock connection.....	233
Fig. B.4	A sample star - gain.s.....	242
Fig. B.5	A system identification example illustrating the use of feedback.....	246
Fig. B.6	A more efficient version of Fig. B.5 using a galaxy.....	249
Fig. B.7	The Jointly-Adaptive TEC/MEC universe, JNTEC.....	254
Fig. B.8	The Jointly-Adaptive TEC/MEC universe topology file, JNTEC.T.....	258
Fig. B.9	The Jointly-Adaptive TEC/MEC galaxy, ECBIN.....	259
Fig. B.10	The Jointly-Adaptive TEC/MEC galaxy topology file, ECBIN.T.....	261
Fig. B.11	The Sequentially-Adaptive TEC/MEC Universe topology file, SEQEC.T.....	264
Fig. B.12	The Sequentially-Adaptive TEC/MEC Galaxy, ECBIN1.....	265
Fig. B.13	The Sequentially-Adaptive TEC/MEC Galaxy topology file, ECBIN1.T.....	267
Fig. B.14	Source code for the CONTROL star.....	270
Fig. B.15	Source code for the ECC star.....	277
Fig. B.16	Source code for the EPROM star.....	281
Fig. B.17	Source code for the FIRL star.....	285
Fig. B.18	Source code for the MECBIN star.....	293
Fig. B.19	Source code for the SHIFTRREG star.....	297
Fig. B.20	Source code for the TEC star.....	304

LIST OF APPENDICES

APPENDIX A: AVERAGE MEAN SQUARE ERROR DETAILS	167
A.1 THE JOINT INPUT AUTOCORRELATION MATRIX.....	167
A.2 THE ECHO PATH LINEAR/NONLINEAR INPUT CROSSCORRELATION MATRIX	169
A.3 COEFFICIENT ERROR POWERS	172
A.4 DESIGN OF THE LINEAR/NONLINEAR ECHO PATH MODEL	182
A.5 SOFTWARE FOR THEORETICAL CALCULATIONS.....	184
A.5.1 AMSE CALCULATION	184
A.5.2 REGION OF CONVERGENCE CALCULATION.....	215
A.5.3 OPTIMUM ADAPTATION STEP-SIZE DETERMINATION FOR FASTEST AMSE CONVERGENCE	219
A.5.4 EVALUATION OF FACTORS AFFECTING STEADY STATE AMSE LEVELS	222
APPENDIX B: CAPSIM SIMULATIONS.....	227
B.1 A BRIEF INTRODUCTION TO CAPSIM	227
B.1.1 A BRIEF HISTORY OF CAPSIM.....	228
B.1.2 BUILDING BLOCKS	229
B.1.2.1 BLOCK CONNECTIONS	232
B.1.2.2 THE STAR	234
B.1.2.2.1 THE STAR DEFINITION	236
B.1.2.2.2 THE STAR LIBRARY	243
B.1.2.3 THE GALAXY	244
B.1.2.4 THE UNIVERSE	250
B.1.3 SIMULATION EXECUTION	250
B.2 SIMULATION TOPOLOGIES.....	253
B.2.1 THE JOINTLY-ADAPTIVE TEC/MEC UNIVERSE	253
B.2.1.1 THE JOINTLY-ADAPTIVE TEC/MEC GALAXY	258
B.2.2 THE SEQUENTIALLY-ADAPTIVE TEC/MEC UNIVERSE	261
B.2.2.1 THE SEQUENTIALLY-ADAPTIVE TEC/MEC GALAXY.....	264

B.3	STAR SOURCE FILES.....	267
APPENDIX C:	GLOSSARY	305
C.1	TEXT CONVENTIONS	305
C.2	ABBREVIATIONS	306
C.3	PRINCIPAL SYMBOLS	307

**Carleton
University
Library**

Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6

Text complete; leaves 1 - 15 omitted in numbering.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Full-duplex transmission over an ISDN subscriber loop is implemented over existing 2-wire telephone lines through the use of 4-wire to 2-wire converters or "hybrids". The hybrid's function within a transceiver is to isolate the locally transmitted signal from the received signal sent from the far-end transmitter. Isolation is possible through the use of a "balancing" impedance within the hybrid. If the balancing impedance exactly matches that of the 2-wire line, then total isolation is possible. However, line impedance is a complex function of frequency, time, and temperature, and is therefore difficult to match exactly [Ada'88]. In practice, a hybrid containing a single compromise impedance may provide only 6-10 dB of attenuation in the local transmitter to local receiver path. This feedthrough of the transmitted signal combined with another component that is "reflected" back along the receive path constitute an "echo" signal which is superimposed on the received signal. Unfortunately, the attenuation of the received signal after travelling

through the line from the far-end transmitter may be as high as 40-50 dB on long loops. This presents an undesirable signal-to-echo ratio of -30 to -40 dB to the receiver's input. For reliable detection of the received data from the received signal, the SNR at the receiver's input must be typically at least 20 dB. This implies an echo attenuation specification of at least 50-60 dB in excess of that which is provided by the transceiver hybrid [Aga'82], [Mess'84].

The purpose of the echo canceller is to replicate the characteristics of the echo path and form a replica of the echo signal based on the transmitted symbol sequence. The replica is then "subtracted" from the received signal. The resulting error signal is then used to adapt the echo canceller's internal coefficients in order to produce a subsequent output that is a closer replica of the actual echo signal. The echo canceller must adapt its coefficients to handle the extra 50-60 dB echo attenuation specification economically and in a reasonable amount of time. Fig. 1.1 illustrates the typical arrangement of a transceiver with an echo canceller.

Fig. 1.1 Typical transceiver and echo canceller arrangement

The simplest form of echo canceller is the adaptive-FIR or transversal echo canceller (TEC) [Son'67]. The TEC produces an echo estimate by forming a linear combination of the transmitted symbol sequence and its weights. After adapting, the TEC weights will contain an approximation of the impulse response due to only the linear distortions occurring in the echo path.

An obvious complication exists for purely linear echo cancellers like the TEC. In practical systems, nonlinear distortions will occur in varying degrees on the echo signal due to various nonlinearities introduced primarily by circuit components in the transceiver itself. The 3 major sources of nonlinearities are transmitted pulse asymmetry, saturation in the transformers, and nonlinearity of the

A/D and D/A converters in the system [Aga'82]. The degree of the nonlinearities can be reduced by more careful design of pulse generation circuits, using bulkier transformers and more stringent tolerancing of circuit components all of which results in increased complexity and/or cost. Further reductions require the addition of a nonlinear echo canceller to handle that portion of the echo signal subjected to nonlinear distortions which cannot be replicated and cancelled by the linear TEC. Typically, the nonlinear echo component is much smaller than the linear component and lasts for the first 6-7 bauds of the echo path impulse response [Mog'87].

The simplest form of nonlinear echo canceller is the "table-lookup" or single stage memory echo canceller (MEC) [Hol'81]. A MEC that handles a m -baud nonlinear component in a binary system must have 2^m coefficients. Each coefficient is responsible for cancelling the echo due to one of the 2^m possible transmitted m -baud symbol sequences [Hol'81]. By contrast, a TEC needs only m coefficients to handle a m -baud linear echo component. By its very nature, the MEC can handle the linear component as well as the nonlinear component of the echo. Drawbacks of such a canceller are the fact that, as inferred from above, the requirements of memory size increase exponentially with the duration of the echo to be cancelled. Furthermore, it can be shown that a MEC of length m requires approximately 2^m times more iterations of the LMS algorithm to converge than a TEC of the same size [Hol'81]. This practically restricts the MEC to only handling the shorter nonlinear distortion.

In the interests of reducing memory requirements and convergence time, a multi-memory approach may be used [Lev'84], [Liu'74], [Pet'87]. The basis of this approach involves the partitioning of the memory into several smaller memories each of which handle a smaller portion of the nonlinear component. These structures exploit the fact that a nonlinear echo of length m typically is not a completely nonlinear function of all m symbols. Rather, it is a linear combination of several shorter nonlinear functions which, in total, have a span of m . As an illustration of the typical savings achieved using this approach, suppose a single-memory MEC of length m is partitioned into p equally-sized memories each addressed by a portion of size x of the original m MEC input symbols. The total memory size required would drop to $p2^x$. Correspondingly, the convergence time would drop by the same factor.

The multi-memory approach to cancelling both the linear and nonlinear echo components may be undesirable because of the fact that it still requires a longer time to cancel the linear echo component when compared against a TEC. The linear echo component typically constitutes most of the total echo signal and therefore it is more desirable to cancel the linear echo component using the fastest converging structure. The solution adopted by some involves the combination of a TEC and a single-memory [Mog'87] or multi-memory [Ada'88] echo canceller. The motivation behind this combination is the fact that the TEC can quickly adapt to the larger linear echo component while leaving the slower converging memory

echo canceller with the job of handling the smaller nonlinear component. The approach taken by [Ada'88] requires the TEC to initially adapt mutually exclusive of the memory echo canceller. After the TEC has converged, its coefficients are frozen and the memory echo canceller is adapted. In the designs described by both [Mog'87] and [Ada'88], no analytical work was presented for the case where the TEC and the memory echo canceller are permitted to adapt simultaneously.

The main conclusion here is that due to the presence of both linear and nonlinear distortions in the echo path, some combination of linear and nonlinear echo canceller must be used.

Many issues arise when a combination of echo cancellers is used to cancel a common echo. A primary issue is the partitioning of work for each echo canceller. Each echo canceller may be assigned a specific section of the echo path impulse response. These sections may or may not overlap. The choice of coefficient adaptation method becomes crucial in this situation. Each echo canceller may be updated using a local cancellation error generated by the subtraction of its output from the echo signal or a common error may be used for all echo cancellers. Not only can the echo canceller sections be overlapped, but the adaptation of each echo canceller can be overlapped in time. That is, a sequence of operation is established where each stage may be assigned a specific time window and error threshold level for adaptation. Outside of this time window, the stage is not permitted to adapt. When these time windows are

overlapped, the corresponding stages adapt simultaneously. When a common error is used to adapt the stages, each time-overlapped stage affects the other in some way.

What is of interest in this thesis is whether joint adaptation of separate echo canceller stages positively or negatively affect the overall performance of the echo canceller.

In the investigation of the interactions that occur during joint adaptation, the previously mentioned linear and nonlinear echo canceller combination, consisting of the simple TEC and single-memory MEC, was chosen for examination. This configuration was chosen merely because it is structurally simpler than one using a multi-memory echo canceller even though the latter is likely to offer better performance.

In our scenario, both the TEC and MEC will share the duty of cancelling the initial linear portion of the echo path impulse response that is equal in length to the nonlinear portion of the response. The remaining linear portion of the response is solely handled by the TEC. Therefore, the MEC is said to be overlapped completely by the TEC. The TEC and MEC coefficients will be updated simultaneously using the LMS algorithm and a common cancellation error. However, separate adaptation step-sizes for the TEC and MEC will be used. There are several arguments to support this adaptation method.

The first argument is that simultaneous adaptation doesn't waste any information in the echo signal, that is, when compared to the method where only one of the echo cancellers adapts at a time.

Thus, the overall convergence time of the coefficients is minimized. This conclusion is valid if one assumes that an insignificant amount of "bad" adaptation interaction occurs between the TEC and the MEC.

The second argument is that the inherently faster adaptation speed of the TEC allows it to cancel a greater share of that linear component which is common to the TEC and MEC. The slower MEC is left with little linear echo to cancel and therefore most of its job would involve the cancelling of the significantly smaller nonlinear component.

The third argument is that the greater number of adaptation states in a sequential adaptation method imposes greater complexity of the "master FSM (Finite State Machine)" that presumably controls the operation of the transceiver. This in turn implies greater circuit complexity.

This thesis will investigate the validity of the above arguments by analysis and comparison of the mean coefficient trajectories and average mean square error (AMSE) behaviour of this "Jointly-Adaptive TEC/MEC" with those of an identical combined structure that instead uses a sequential approach to adapting its coefficients.

The remaining part of this chapter describes the combined TEC/MEC structure. In Chapter 2, an analysis of the Jointly-Adaptive TEC/MEC will identify the factors affecting the convergence of the coefficients and the AMSE. The investigation will identify whether any bad adaptation interactions occur during joint adaptation and if they are significant enough to warrant a retraction of the first

argument. In Chapter 3, a similar analysis of a "Sequentially-Adaptive" TEC/MEC structure will be performed. In Chapter 4, the joint and sequential adaptation methods will be compared in terms of the coefficient and AMSE behaviour. In Chapter 5, the conclusions of these investigations will be stated.

1.2 THE COMBINED TEC/MEC CONFIGURATION

The basic structure of the combined TEC/MEC is illustrated in Fig. 1.2. It consists of a transversal echo canceller and a single-stage memory echo canceller. Both cancellers operate at the symbol rate and share a common input symbol shift register.

The transmission specifications for an ISDN subscriber loop U-interface [Ans'88] require that 2B1Q signals be utilized, however, to facilitate the theoretical analysis that will follow, we will assume that the input symbols are binary " ± 1 " symbols and have a zero mean. It is expected that the results obtained will be similar to those obtained by an analysis using 2B1Q symbols.

The TEC operates on the current and past $n-1$ input symbols and the MEC derives the address of the current MEC output coefficient from the current and past $m-1$ input symbols. For practical purposes, we will assume that the nonlinear echo component, which must be cancelled by the MEC, has a duration less than or equal to that of the linear component. Thus, we can assume that the same relationship exists between the TEC and MEC spans, ie. $n \geq m$.

Fig. 1.2 The Combined TEC/MEC Configuration

The TEC output at time k is formed by the sum of the products of the current and past $n-1$ input symbols with their respective TEC coefficients. The MEC output at time k is the contents of a single MEC

memory location that is selected by an address generated from the current and past $m-1$ input symbols. Note that the first m input symbols are common to the TEC and the MEC. Accordingly, we can state that the combined TEC/MEC structure has an "overlap" of m . The TEC and MEC outputs are both subtracted from the current sample of the echo signal, $d(k)$, resulting in a cancellation error of $e(k)$. $e(k)$ is used to update both the TEC and the MEC. Each TEC coefficient is updated by $\alpha e(k)$ where α is the TEC adaptation step-size. The currently addressed MEC coefficient is updated by $\beta e(k)$ where β is the MEC adaptation step-size.

The introduction of noise into the echo signal is modelled by an independent, additive, and white noise source $v(k)$. A far-end signal, received during adaptation, may be treated as a signal that is independent from the echo signal $d(k)$. Therefore, we may assume that the received far-end signal constitutes part of $v(k)$.

There are n TEC coefficients which can be represented by the $(n,1)$ column vector $s(k)$. These are associated with the n current and past input symbols which we can denote by the $(n,1)$ column vector $x(k)$. A partition can be defined between the first m and remaining $n-m$ coefficients and their associated input symbols. The first m , which we shall denote by the $(m,1)$ column vector $s'(k)$, are associated with the first m input symbols which we shall denote accordingly by the $(m,1)$ column vector $x'(k)$. These quantities are associated with the overlapped portion of the combined TEC/MEC. We shall denote the remaining coefficients and input symbols,

associated with the non-overlapped portion, by the $(n-m,1)$ column vectors $s''(k)$ and $x''(k)$ respectively.

The MEC generates a unique address based on the current and past $m-1$ binary input symbols, ie. $x'(k)$. Each unique address corresponds to a MEC coefficient stored in the MEC's memory. The memory therefore must contain 2^m coefficients. We shall denote the address by the $(2^m,1)$ column vector $a(k)$ and the contents of the memory by the $(2^m,1)$ column vector $m(k)$. The address vector is defined in a similar way to the "augmented input vector" described in [Ada'88]. The address vector consists entirely of zero entries with the exception of a single "1" entry whose location within the vector depends uniquely on $x'(k)$. We may define the mapping between $x'(k)$ and its associated address vector as shown in TABLE 1.1.

TABLE 1.1 A mapping between the m -baud symbol vector $x'(k)$ and MEC address vector $a(k)$.

$x'(k)$	$a(k)$
$[-1 \ -1 \ -1 \ \dots \ -1]^T$	$[1 \ 0 \ 0 \ 0 \ \dots \ 0]^T$
$[+1 \ -1 \ -1 \ \dots \ -1]^T$	$[0 \ 1 \ 0 \ 0 \ \dots \ 0]^T$
$[-1 \ +1 \ -1 \ \dots \ -1]^T$	$[0 \ 0 \ 1 \ 0 \ \dots \ 0]^T$
$[+1 \ +1 \ -1 \ \dots \ -1]^T$	$[0 \ 0 \ 0 \ 1 \ \dots \ 0]^T$
$\dots$	$\dots$
$[+1 \ +1 \ +1 \ \dots \ +1]^T$	$[0 \ 0 \ 0 \ 0 \ \dots \ 1]^T$

The mapping defined by TABLE 1.1 permits us to use $\mathbf{a}(k)$ instead of $\mathbf{x}'(k)$ as the input to the MEC. It will become evident in CHAPTER 2 that this representation of the MEC input is more convenient because it allows us to express the MEC output and coefficient update in the form of the familiar LMS algorithm. The mapping from $\mathbf{a}(k)$ to $\mathbf{x}'(k)$ can be represented more compactly by multiplication with a $2^m \rightarrow m$ linear transformation matrix, $\mathbf{A}$ with dimensions $(m, 2^m)$

$$\mathbf{x}'(k) = \mathbf{A}\mathbf{a}(k) \quad (1.1)$$

where,

$$\mathbf{A} = \begin{bmatrix} \mathbf{x}'_0 & \mathbf{x}'_1 & \mathbf{x}'_2 & \mathbf{x}'_3 & \dots & \mathbf{x}'_{\phi-1} \end{bmatrix} \quad (1.2)$$

$$\mathbf{x}'_0 = [-1 \ -1 \ -1 \ \dots \ -1]^T$$

$$\mathbf{x}'_1 = [+1 \ -1 \ -1 \ \dots \ -1]^T$$

$$\mathbf{x}'_2 = [-1 \ +1 \ -1 \ \dots \ -1]^T$$

$$\mathbf{x}'_3 = [+1 \ +1 \ -1 \ \dots \ -1]^T$$

...

$$\mathbf{x}'_{\phi-1} = [+1 \ +1 \ +1 \ \dots \ +1]^T$$

$$\phi = 2^m \quad (1.3)$$

The $\mathbf{x}'_i$'s represent all possible $\mathbf{x}'(k)$ and are precisely the rows of TABLE 1.1.

1.3 SUMMARY

The idea of utilizing dual echo canceller configurations to handle both the linear and nonlinear echo components is not new. Although both components can be handled by a single-memory-based or "table-lookup" structure [Hol'81], the convergence time is considerably longer than that of a TEC and the memory requirements increase exponentially with the number of bauds over which the canceller operates. Therefore, alternate memory-based approaches with lower memory requirements and convergence times more comparable to that of a TEC are typically used. Some adopt a multi-memory structure [Lev'84], [Liu'74], and [Pet'87]. Although these multi-memory approaches require less memory and hence have shorter convergence times, they still fall short of a TEC in the area of linear echo cancellation. This problem is tackled by other configurations that employ a combination of a purely linear echo canceller such as the TEC and a single-memory-based [Mog'87] or a multi-memory-based echo canceller [Ada'88]. The latter structure involves a sequencing of cancellers where the TEC is allowed to initially adapt mutually exclusive of the memory echo canceller. After the TEC converges, its taps are frozen and the memory echo canceller is adapted to cancel the residual echo consisting largely of the nonlinear echo component. An analytical description of the linear/nonlinear echo canceller behaviour was not provided for the case where the adaptation scheme involved the simultaneous updating of the TEC and MEC coefficients.

This thesis will center around the analysis and simulation of a structure consisting of the combination of a TEC with a single-memory-based echo canceller (MEC) as described in [Mog'87]. The main contribution will be the analytical description of the coefficient and the cancellation error behaviour during the joint adaptation of the two cancellers. The intent is to discover whether this joint adaptation offers any advantages over a sequential adaptation approach in which the TEC is initially allowed to converge in the absence of any MEC intervention before the MEC itself is permitted to adapt.

In this chapter we have also described the structure of the combined TEC/MEC configuration. In this configuration, the TEC operates over the first n bauds of the echo path impulse response and the MEC operates over first m bauds where $m \leq n$. The "overlap" of the TEC/MEC refers to the size of the portion of the input symbol shift register used by both the TEC and the MEC. As the MEC span is assumed to be always less than or equal to the TEC span, the overlap in this case is m . The TEC and MEC are adapted from a common error $e(k)$ which results from the subtraction of the TEC and MEC outputs from the echo. The TEC and MEC coefficients are adapted at a rate determined by their respective adaptation step-sizes α and β . We have adopted a vector notation for the coefficients and input symbols used by the TEC and MEC to facilitate the analysis in the following chapters.

CHAPTER 2

ANALYSIS OF THE JOINTLY-ADAPTIVE TEC/MEC

2.1 THE LMS ALGORITHM FOR JOINT ADAPTATION

The LMS algorithm [Hof60] is used to simultaneously update both the TEC and MEC coefficients, that is, both echo cancellers are adapted at the same time from a common error, $e(k)$. The output of the TEC/MEC can be expressed as follows.

$$\text{TEC/MEC Output}(k) = \sum_{i=0}^{n-1} s_i x(k-i) + m_j(k) \quad (2.1)$$

where,

$s_i(k)$	i th TEC coefficient at time k
$m_j(k)$	j th MEC coefficient at time k where j is identified by an address formed from the first m input symbols

The updating of the TEC and MEC coefficients via the LMS algorithm can be described by the following recursive equations.

$$s_i(k+1) = s_i(k) + \alpha e(k)x(k-i) \quad i = 0, 1, 2, \dots, n-1 \quad (2.2)$$

$$m_j(k+1) = m_j(k) + \beta e(k) \quad j = \text{address} \quad (2.3)$$

where,

$$e(k) = d(k) - \text{TEC/MEC Output}(k) \quad (2.4)$$

We can reexpress (2.1)-(2.4) by adopting the vector notation introduced in section 1.2.

$$\text{TEC/MEC Output}(k) = \mathbf{x}^T(k)\mathbf{s}(k) + \mathbf{a}^T(k)\mathbf{m}(k) \quad (2.5)$$

$$\mathbf{s}(k+1) = \mathbf{s}(k) + \alpha e(k)\mathbf{x}(k) \quad (2.6)$$

$$\mathbf{m}(k+1) = \mathbf{m}(k) + \beta e(k)\mathbf{a}(k) \quad (2.7)$$

where,

$$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{s}(k) - \mathbf{a}^T(k)\mathbf{m}(k) \quad (2.8)$$

An equivalent representation of (2.5), (2.6) and (2.8) using the partitioned vector quantities introduced in section 1.2 is

$$\text{TEC/MEC Output}(k) = \mathbf{x}'^T(k)\mathbf{s}'(k) + \mathbf{x}''^T(k)\mathbf{s}''(k) + \mathbf{a}^T(k)\mathbf{m}(k) \quad (2.9)$$

$$\mathbf{s}'(k+1) = \mathbf{s}'(k) + \alpha e(k)\mathbf{x}'(k) \quad (2.10)$$

$$\mathbf{s}''(k+1) = \mathbf{s}''(k) + \alpha e(k)\mathbf{x}''(k) \quad (2.11)$$

where,

$$e(k) = d(k) - \mathbf{x}'^T(k)\mathbf{s}'(k) - \mathbf{x}''^T(k)\mathbf{s}''(k) - \mathbf{a}^T(k)\mathbf{m}(k) \quad (2.12)$$

Note that by using the address vector $\mathbf{a}(k)$ we have expressed the output and updating of the MEC coefficients in a form similar to

that of the TEC. This is convenient because, in the following chapters, it allows us to analyze the MEC's behaviour in much the same way as the TEC.

Another convenient representation of the previous equations involves a repartitioning of the input and coefficient vectors to form the "joint overlapped TEC/MEC input vector" $\mathbf{x}_a(k)$ and "joint overlapped coefficient vector" $\mathbf{c}(k)$ as follows.

"joint overlapped TEC/MEC input vector"

$$\mathbf{x}_a(k) \equiv \begin{bmatrix} \mathbf{x}'(k) \\ \mathbf{a}(k) \end{bmatrix} \quad (2.13)$$

"joint overlapped coefficient vector"

$$\mathbf{c}(k) \equiv \begin{bmatrix} \mathbf{s}'(k) \\ \mathbf{m}(k) \end{bmatrix} \quad (2.14)$$

Those equations containing $\mathbf{x}'(k)$, $\mathbf{a}(k)$, $\mathbf{s}'(k)$ and $\mathbf{m}(k)$ are expressed using these repartitioned vectors as follows:

$$\text{TEC/MEC Output}(k) = \mathbf{x}_a^T(k)\mathbf{c}(k) + \mathbf{x}''^T(k)\mathbf{s}''(k) \quad (2.15)$$

$$\mathbf{c}(k+1) = \mathbf{c}(k) + \mathbf{J}e(k)\mathbf{x}_a(k) \quad (2.16)$$

where,

$$e(k) = d(k) - \mathbf{x}_a^T(k)\mathbf{c}(k) - \mathbf{x}''^T(k)\mathbf{s}''(k) \quad (2.17)$$

$\mathbf{J}$ "overlapped coefficient adaptation step-size matrix"

where $\mathbf{J}$ has the form

$$\mathbf{J} = \begin{bmatrix} \alpha \mathbf{I}_m & 0 \\ 0 & \beta \mathbf{I}_\phi \end{bmatrix} \quad (2.18)$$

$\mathbf{I}_m$ (m,m) identity matrix

$\mathbf{I}_\phi$ $(2^m, 2^m)$ identity matrix

This repartitioning allows one to describe the TEC/MEC operation as only 2 separate processes. The joint vector equations describe the output and adaptation processes associated with the overlapped portion and the double-primed vector equations describe the processes associated with the non-overlapped portion.

At this point we make several "fundamental" assumptions about the input vectors using the first 3 assumptions made by [Hay'86].

Assumption 1: Each input vector is statistically independent of all previous input vectors. Thus,

$$E\{\mathbf{x}(i)\mathbf{x}^T(j)\} = 0 \quad j = 0, 1, 2, \dots, i-1$$

$$E\{\mathbf{a}(i)\mathbf{a}^T(j)\} = 0 \quad j = 0, 1, 2, \dots, i-1$$

Assumption 2: Each input vector is statistically independent of all previous samples of the echo response. Thus,

$$E\{\mathbf{x}(i)d(j)\} = 0 \quad j = 0, 1, 2, \dots, i-1$$

$$E\{\mathbf{a}(i)d(j)\} = 0 \quad j = 0, 1, 2, \dots, i-1$$

Assumption 3: Each sample of the echo response is statistically dependent on the corresponding input vectors. Thus,

$$E\{\mathbf{x}(k)d(k)\} \neq 0$$

$$E\{\mathbf{a}(k)d(k)\} \neq 0$$

It is worthy to note that Assumption 1 is not actually true due to the fact that an input vector is merely a time-shifted version of the previous input vector with the exception of the current input symbol. Nevertheless, it has been shown by [Maz'79] that the difference between results obtained by adopting these assumptions and those from analysis recognizing the statistical dependencies is often small provided that the adaptation step-sizes are small. In the interest of simplicity, we will apply these assumptions to the following analyses hoping that our omission of the statistical dependencies will not degrade our predictions significantly. In any case, confidence in the accuracy of the analytical results will be derived from comparison of these results with those obtained from simulations.

$\mathbf{x}''(k)$ is shifted in time with respect to $\mathbf{x}'(k)$ and hence $\mathbf{a}(k)$. Therefore, from Assumption 1, we can conclude that

$$\mathbf{E}\{\mathbf{x}'(k)\mathbf{x}''^T(k)\} = 0 \quad (2.19)$$

$$\mathbf{E}\{\mathbf{a}(k)\mathbf{x}''^T(k)\} = 0 \quad (2.20)$$

The coefficient update equations described earlier reveal that a coefficient vector at time $k+1$ is dependent on only 3 quantities: previous input vectors, previous samples of the noisy echo response, and the initial coefficient vector. Specifically, we have that

$\mathbf{s}'(k+1)$ depends on $\mathbf{x}'(k)$, $\mathbf{x}'(k-1)$, ..., $\mathbf{x}'(0)$, $d(k)$, $d(k-1)$, ..., $d(0)$,
and $\mathbf{s}'(0)$.

$s''(k+1)$ depends on $x''(k), x''(k-1), \dots, x''(0), d(k), d(k-1), \dots, d(0)$,
and $s''(0)$.

$m(k+1)$ depends on $a(k), a(k-1), \dots, a(0), d(k), d(k-1), \dots, d(0)$,
and $m(0)$.

By our independence assumptions, $x(k+1)$ and $d(k+1)$ are independent of all previous $x(k)$'s and $d(k)$'s respectively hence we can conclude that a coefficient vector at time k is statistically independent of the corresponding input vector and the echo response at that time. Specifically,

$$E\{s'^T(k)x'(k)\} = E\{s'^T(k)\}E\{x'(k)\} \quad (2.21)$$

$$E\{s''^T(k)x''(k)\} = E\{s''^T(k)\}E\{x''(k)\} \quad (2.22)$$

$$E\{m^T(k)a(k)\} = E\{m^T(k)\}E\{a(k)\} \quad (2.23)$$

These results will be applied repeatedly throughout the analyses in the following chapters.

2.2 AVERAGE MEAN SQUARE ERROR

In this section an expression describing the evolution of the Average Mean Square Error (AMSE) with time k will be derived. The $AMSE(k)$ is defined mathematically as $E\{MSE(k)\}$ where " $E\{\}$ " refers to mathematical expectation or more precisely ensemble average. In order to derive the expression, we must investigate the Minimum Mean Square Error (MMSE), coefficient trajectories, coefficient error

powers, and the Mean Square Error (MSE). Factors affecting the AMSE convergence will also be identified.

2.2.1 MINIMUM MEAN SQUARE ERROR

2.2.1.1 THE ERROR SURFACE

In this section, we will derive an expression for MSE as a function of the TEC/MEC coefficients. We begin with an expression of the error omitting the k subscript on the coefficient vectors because we are not interested in adapting them at this point.

$$e(k) = d(k) - \mathbf{x}'^T(k)\mathbf{s}' - \mathbf{x}''^T(k)\mathbf{s}'' - \mathbf{a}^T(k)\mathbf{m} \quad (2.24)$$

Squaring and taking the expectation of both sides results in

$$\begin{aligned} \text{MSE}(\mathbf{s}', \mathbf{s}'', \mathbf{m}) &\equiv E\{e^2(k)\} \\ &= E\{d^2(k)\} - 2\mathbf{s}'^T E\{d(k)\mathbf{x}'(k)\} - 2\mathbf{m}^T E\{d(k)\mathbf{a}(k)\} \\ &\quad - 2\mathbf{s}''^T E\{d(k)\mathbf{x}''(k)\} + \mathbf{s}'^T E\{\mathbf{x}'(k)\mathbf{x}'^T(k)\}\mathbf{s}' \\ &\quad + 2\mathbf{s}'^T E\{\mathbf{x}'(k)\mathbf{a}^T(k)\}\mathbf{m} \\ &\quad + 2\mathbf{s}'^T E\{\mathbf{x}'(k)\mathbf{x}''^T(k)\}\mathbf{s}'' \\ &\quad + 2\mathbf{m}^T E\{\mathbf{a}(k)\mathbf{x}''^T(k)\}\mathbf{s}'' \\ &\quad + \mathbf{m}^T E\{\mathbf{a}(k)\mathbf{a}^T(k)\}\mathbf{m} \\ &\quad + \mathbf{s}''^T E\{\mathbf{x}''(k)\mathbf{x}''^T(k)\}\mathbf{s}'' \end{aligned} \quad (2.25)$$

We now define the following "input/echo crosscorrelation vectors".

"overlapped TEC input/echo crosscorrelation vector"

$$\mathbf{p}' \equiv E\{d(k)\mathbf{x}'(k)\} \quad (2.26)$$

"non-overlapped TEC input/echo crosscorrelation vector"

$$\mathbf{p}'' \equiv E\{d(k)\mathbf{x}''(k)\} \quad (2.27)$$

"MEC input/echo crosscorrelation vector"

$$\mathbf{q} \equiv E\{d(k)\mathbf{a}(k)\} \quad (2.28)$$

The overlapped TEC input/echo crosscorrelation vector, $\mathbf{p}'$, can be rewritten in terms of the MEC input/echo crosscorrelation vector, $\mathbf{q}$, by combining the mapping between $\mathbf{x}'(k)$ and $\mathbf{a}(k)$, given by (1.1), and (2.26) as follows.

$$\mathbf{p}' \equiv E\{d(k)\mathbf{x}'(k)\} = E\{d(k)\mathbf{A}\mathbf{a}(k)\} = \mathbf{A}E\{d(k)\mathbf{a}(k)\} = \mathbf{A}\mathbf{q} \quad (2.29)$$

The input autocorrelation matrices are evaluated using the assumptions we made in 2.1. $E\{\mathbf{x}'(k)\mathbf{x}'^T(k)\}$ is the (m,m) "overlapped TEC input autocorrelation matrix" and contains 2 types of terms. The diagonal terms will have the form

$$\begin{aligned} E\{x'^2(k-i)\} &= \text{Pr}(-1)(-1)^2 + \text{Pr}(+1)(+1)^2 \quad i = 0, 1, 2, \dots, m-1 \\ &= 0.5(1) + 0.5(1) \\ &= 1 \end{aligned}$$

The off-diagonal terms will have the form

$$\begin{aligned} E\{x'(k-i)x'(k-j)\} &= E\{x'(k-i)\}E\{x'(k-j)\} & i, j = 0, 1, 2, \dots, m-1 \\ &= 0 & i \neq j \end{aligned}$$

Thus we may write

"overlapped TEC input autocorrelation matrix"

$$E\{x'(k)x'^T(k)\} = I_m \quad (2.30)$$

Similarly, for $E\{x''(k)x''^T(k)\}$ we may write

"non-overlapped TEC input autocorrelation matrix"

$$E\{x''(k)x''^T(k)\} = I_{n-m} \quad (2.31)$$

where,

I_{n-m} $(n-m, n-m)$ identity matrix

$E\{a(k)a^T(k)\}$ is the $(2m, 2m)$ "MEC input autocorrelation matrix" which can be evaluated by considering a general term.

$$a(k)a^T(k) = \begin{bmatrix} 0_\phi & 0_\phi & \dots & a(k) & 0_\phi & \dots & 0_\phi \end{bmatrix}$$

where,

0_ϕ $(2m, 1)$ column matrix of zeroes

The position of the $a(k)$ depends on the position of the "1" in $a^T(k)$. There are 2^m equiprobable $a(k)$'s so the expectation of this matrix can be written as follows.

$$\begin{aligned}
E\{\mathbf{a}(k)\mathbf{a}^T(k)\} = & \left(\begin{bmatrix} \mathbf{a}_0 & 0_\phi & 0_\phi & 0_\phi & \dots & 0_\phi \end{bmatrix} \right. \\
& + \begin{bmatrix} 0_\phi & \mathbf{a}_1 & 0_\phi & 0_\phi & \dots & 0_\phi \end{bmatrix} \\
& + \begin{bmatrix} 0_\phi & 0_\phi & \mathbf{a}_2 & 0_\phi & \dots & 0_\phi \end{bmatrix} + \dots \\
& \left. + \begin{bmatrix} 0_\phi & 0_\phi & 0_\phi & 0_\phi & \dots & \mathbf{a}_\phi \end{bmatrix} \right) / 2^m
\end{aligned}$$

where,

$\mathbf{a}_i$'s have a length of 2^m

$$\mathbf{a}_0 = [1 \ 0 \ 0 \ \dots \ 0]^T$$

$$\mathbf{a}_1 = [0 \ 1 \ 0 \ \dots \ 0]^T$$

$$\mathbf{a}_2 = [0 \ 0 \ 1 \ \dots \ 0]^T$$

...

$$\mathbf{a}_{\phi-1} = [0 \ 0 \ 0 \ \dots \ 1]^T$$

The resulting summation gives

"MEC input autocorrelation matrix"

$$E\{\mathbf{a}(k)\mathbf{a}^T(k)\} = 2^{-m}\mathbf{I}_\phi \quad (2.32)$$

We define $\mathbf{Y}$ as the $(m, 2^m)$ "overlapped TEC input/MEC input crosscorrelation matrix", $E\{\mathbf{x}'(k)\mathbf{a}^T(k)\}$. Since $\mathbf{a}(k)$ is entirely dependent on $\mathbf{x}'(k)$, we cannot ignore this matrix on the basis of any independence assumption. $\mathbf{Y}$ is evaluated by using (2.32) and the linear transformation between $\mathbf{x}'(k)$ and $\mathbf{a}(k)$ given by (1.1).

"overlapped TEC/MEC input crosscorrelation matrix"

$$\begin{aligned}
 \mathbf{Y} &= \mathbf{E}\{\mathbf{x}'(k)\mathbf{a}^T(k)\} \\
 &= \mathbf{E}\{(\mathbf{A}\mathbf{a}(k))\mathbf{a}^T(k)\} \\
 &= \mathbf{A}\mathbf{E}\{\mathbf{a}(k)\mathbf{a}^T(k)\} \\
 \mathbf{Y} &= 2^{-m}\mathbf{A}
 \end{aligned} \tag{2.33}$$

The product, $\mathbf{Y}\mathbf{Y}^T$, will appear from time to time in the analysis. Fortunately, $\mathbf{A}$ has the convenient property of having mutually orthogonal rows, that is,

$$\mathbf{A}\mathbf{A}^T = 2^m\mathbf{I}_m \tag{2.34}$$

Thus by (2.33) and (2.34) the product $\mathbf{Y}\mathbf{Y}^T$ is simply

$$\mathbf{Y}\mathbf{Y}^T = 2^{-m}\mathbf{I}_m \tag{2.35}$$

Combining (2.26)-(2.33) with (2.25) and eliminating the input crosscorrelation terms containing $\mathbf{x}''(k)$ as a result of (2.19) and (2.20), we obtain

$$\begin{aligned}
 \text{MSE}(\mathbf{s}', \mathbf{s}'', m) &= \mathbf{E}\{d^2(k)\} - 2\mathbf{s}'^T\mathbf{p}' - 2\mathbf{m}^T\mathbf{q} - 2\mathbf{s}''^T\mathbf{p}'' + 2\mathbf{s}'^T\mathbf{Y}\mathbf{m} \\
 &\quad + \mathbf{s}'^T\mathbf{s}' + 2^{-m}\mathbf{m}^T\mathbf{m} + \mathbf{s}''^T\mathbf{s}''
 \end{aligned} \tag{2.36}$$

The expression in (2.36) gives the MSE as a function of the coefficient vectors. An imaginary plot of (2.36) versus the coefficient vectors would produce a multi-dimensional error "surface".

An equivalent form of the error surface can be derived by expressing the MSE as a function of the joint overlapped coefficient vector, $\mathbf{c}$, defined in (2.14). The corresponding expression of the error is essentially that of (2.17) with coefficients that do not change with time.

$$e(k) = d(k) - \mathbf{x}_a^T(k)\mathbf{c} - \mathbf{x}^T(k)\mathbf{s} \quad (2.37)$$

The square and expectation of both sides is

$$\begin{aligned} \text{MSE}(\mathbf{c}, \mathbf{s}) = & E\{d^2(k)\} - 2\mathbf{c}^T E\{d(k)\mathbf{x}_a(k)\} - 2\mathbf{s}^T E\{d(k)\mathbf{x}(k)\} \\ & + 2\mathbf{c}^T E\{\mathbf{x}_a(k)\mathbf{x}^T(k)\}\mathbf{s} + \mathbf{c}^T E\{\mathbf{x}_a(k)\mathbf{x}_a^T(k)\}\mathbf{c} \\ & + \mathbf{s}^T E\{\mathbf{x}(k)\mathbf{x}^T(k)\}\mathbf{s} \end{aligned} \quad (2.38)$$

The various joint correlations in (2.38), when expressed in terms of their constituent vectors, would consist of the correlation terms we have already seen. Applying our previous evaluations of the correlation terms to (2.38) produces

$$\text{MSE}(\mathbf{c}, \mathbf{s}) = E\{d^2(k)\} - 2\mathbf{c}^T \mathbf{p}_q - 2\mathbf{s}^T \mathbf{p} + \mathbf{c}^T \mathbf{R}_m \mathbf{c} + \mathbf{s}^T \mathbf{T} \mathbf{s} \quad (2.39)$$

where,

$\mathbf{p}_q$ "joint overlapped TEC/MEC input/echo crosscorrelation vector" where $\mathbf{p}_q$ has the form

$$\mathbf{p}_q \equiv E\{d(k)\mathbf{x}_a(k)\} = \begin{bmatrix} \mathbf{p}' \\ \mathbf{q} \end{bmatrix} \quad (2.40)$$

$\mathbf{R}_m$ "joint overlapped input autocorrelation matrix" where $\mathbf{R}_m$ has the form

$$\mathbf{R}_m \equiv E\{\mathbf{x}_a(k)\mathbf{x}_a^T(k)\} = \begin{bmatrix} \mathbf{I}_m & \mathbf{Y} \\ \mathbf{Y}^T & 2^{-m}\mathbf{I}_\phi \end{bmatrix} \quad (2.41)$$

The last 3 terms of (2.36) and the last 2 terms of (2.39) are the products of the coefficient vectors and their transposes indicating that the error surface is a quadratic function of the individual coefficients. This reinforces the argument for the use of the LMS algorithm to adapt the coefficients because it is an approximation of the "steepest descent" gradient search method [Ste'85] which depends on the slope of a quadratic error surface to find the minimum. In the next section, the set of coefficients which define the minimum of this error surface will be derived.

2.2.1.2 CONDITIONS FOR MINIMUM MEAN SQUARE ERROR

In this section a set of coefficient vectors that minimizes the MSE will be derived. The minimum can be found by exploiting the quadratic nature of the error surface. We begin by finding the gradient of the error surface of (2.36) with respect to the coefficient vectors.

$$\begin{aligned}
\frac{\partial E(e^2(k))}{\partial \begin{bmatrix} s' \\ s'' \\ m \end{bmatrix}} &= -2 \begin{bmatrix} p' \\ 0 \\ 0 \end{bmatrix} - 2 \begin{bmatrix} 0 \\ p'' \\ 0 \end{bmatrix} - 2 \begin{bmatrix} 0 \\ 0 \\ q \end{bmatrix} + 2 \begin{bmatrix} Ym \\ 0 \\ Y^T s' \end{bmatrix} + 2 \begin{bmatrix} s' \\ 0 \\ 0 \end{bmatrix} \\
&\quad + 2 \begin{bmatrix} 0 \\ s'' \\ 0 \end{bmatrix} + 2 \cdot 2^{-m} \begin{bmatrix} 0 \\ 0 \\ m \end{bmatrix} \\
\frac{\partial E(e^2(k))}{\partial \begin{bmatrix} s' \\ s'' \\ m \end{bmatrix}} &= \begin{bmatrix} -2p' + 2Ym + 2s' \\ -2p'' + 2s'' \\ -2q + 2Y^T s' + 2 \cdot 2^{-m} m \end{bmatrix} \quad (2.42)
\end{aligned}$$

The minimum of the quadratic error surface lies where the gradient of the surface is 0. Setting (2.42) to 0 we obtain the following conditions for MMSE.

$$s'^* - p' + Ym^* = 0_m \quad (2.43)$$

$$s''^* - p'' = 0_{n-m} \quad (2.44)$$

$$2^{-m} m^* - q + Y^T s'^* = 0_\phi \quad (2.45)$$

where,

s'^* , s''^* , and m^* set of optimum coefficient vectors

The conditions for MMSE can be alternatively expressed in terms of s'' and the joint overlapped coefficient vector, c , by

following a similar procedure but using (2.39) and taking the derivative with respect to s'' and c . The results are

$$\begin{aligned} \mathbf{R}_m \mathbf{c}^* &= \mathbf{p}_q \\ \mathbf{s}''^* - \mathbf{p}'' &= \mathbf{0}_{n-m} \end{aligned} \quad (2.46)$$

where,

the second equation is identical to (2.44)

(2.44) suggests that the partition of TEC coefficients, s'' , that is not associated with overlapped portion of the TEC/MEC structure has a single solution, namely the corresponding nonoverlapped TEC input/echo crosscorrelation vector $\mathbf{p}''$. For the joint overlapped coefficient vector, $\mathbf{c}$, associated with the overlapped portion, the situation is different. The difference lies in the nature of $\mathbf{R}_m$. (2.46) indicates that there is a single optimum vector $\mathbf{c}^*$ if and only if the joint overlapped input autocorrelation matrix, $\mathbf{R}_m$ is nonsingular. The definition of $\mathbf{R}_m$ is such that this is not the case (see Appendix A.1). The singularity of $\mathbf{R}_m$ implies that there is a multitude of solutions for the MMSE and, in an adaptive situation, the overlapped coefficients may never reach a single stable solution. Every time the coefficients approach a solution on the quadratic error surface, there may be other degrees of freedom which might eventually cause the coefficients to "wander" out.

In the next section we will investigate the mean behaviour of the coefficient vectors as they are adapted using the LMS algorithm

to discover if the adaptation process itself constrains the mean value of the coefficients to a path leading to a single stable solution instead of multiple unstable solutions.

2.2.1.3 COEFFICIENT TRAJECTORY ANALYSIS

In this section we will investigate how the mean value of the coefficient vectors change during adaptation. Establishing whether or not the coefficients converge in the mean sense is the first step in establishing conditions for the overall convergence of the error. We will begin by analyzing the evolution of the mean overlapped coefficients over time. Substituting (2.12) into (2.10) results in

$$\begin{aligned} s'(k+1) = & s'(k) + \alpha d(k)x'(k) - \alpha x'(k)x'^T(k)s'(k) - \alpha x'(k)x''^T(k)s''(k) \\ & - \alpha x'(k)a^T(k)m(k) \end{aligned} \quad (2.47)$$

Taking the ensemble average of (2.47) and using the results of our previous independence assumptions we have

$$\begin{aligned} E\{s'(k+1)\} = & E\{s'(k)\} + \alpha E\{d(k)x'(k)\} - \alpha E\{x'(k)x'^T(k)\}E\{s'(k)\} - \\ & \alpha E\{x'(k)x''^T(k)\}E\{s''(k)\} - \alpha E\{x'(k)a^T(k)\}E\{m(k)\} \\ E\{s'(k+1)\} = & (1-\alpha)E\{s'(k)\} + \alpha p' - \alpha Y E\{m(k)\} \end{aligned} \quad (2.48)$$

Using a similar procedure for $E\{m(k+1)\}$ we arrive at

$$E\{m(k+1)\} = (1-2^{-m}\beta)E\{m(k)\} + \beta q - \beta Y^T E\{s'(k)\} \quad (2.49)$$

(2.48) and (2.49) suggest that the mean overlapped coefficients will exhibit interdependent behaviour. The form of the equations are similar so we suspect that there may be some simple relationship between the mean overlapped coefficient vectors. The search for this relationship begins with the original LMS update equations for the overlapped coefficient vectors given in (2.7) and (2.10). We multiply both sides of (2.7) by αY because the resulting terms are related to $\alpha Y E\{m(k)\}$ which appears in (2.48).

$$\alpha Y m(k+1) = \alpha Y m(k) + \alpha \beta e(k) Y a(k) \quad (2.50)$$

Using the relationship between $x'(k)$ and $a(k)$ given by (1.1) and the evaluation of Y given by (2.33), (2.50) can be rewritten as

$$\alpha Y m(k+1) = \alpha Y m(k) + \alpha 2^{-m} \beta e(k) x'(k) \quad (2.51)$$

We can duplicate the last term of (2.51) by multiplying both sides of (2.10) by $2^{-m} \beta$.

$$2^{-m} \beta s'(k+1) = 2^{-m} \beta s'(k) + \alpha 2^{-m} \beta e(k) x'(k) \quad (2.52)$$

Before continuing, we shall make the assumption that all TEC/MEC coefficients are initially zero. That is,

$$s'(0) = 0_m \quad (2.53)$$

$$s''(0) = 0_{n-m} \quad (2.54)$$

$$m(0) = 0_m \quad (2.55)$$

(2.51) and (2.52) may be compared by evaluating them both by simple induction from the initial zero state. The result gives a simple relationship between $s'(k)$ and $m(k)$.

$$\alpha Y m(k) = 2^{-m} \beta s'(k) \quad (2.56)$$

or alternatively,

$$\alpha Y E\{m(k)\} = 2^{-m} \beta E\{s'(k)\} \quad (2.57)$$

Note that this relationship does not depend on the echo signal, $d(k)$, nor does it depend on the cancellation error, $e(k)$! This simple yet important relationship between the overlapped coefficient vectors will now be used to simplify the mean overlapped coefficient update equations in (2.48) and (2.49).

Substituting (2.57) directly into (2.48) gives

$$E\{s'(k+1)\} = (1 - (\alpha + 2^{-m}\beta))E\{s'(k)\} + \alpha p' \quad (2.58)$$

The above recursion is solved by induction from the zero initial state with the following result.

$$E\{s'(k)\} = \alpha p' \sum_{i=0}^{k-1} (1 - (\alpha + 2^{-m}\beta))^i \quad (2.59)$$

The condition for convergence of the above series, hence the mean overlapped TEC coefficient vector, is

$$0 < \alpha + 2^{-m}\beta < 2 \quad (2.60)$$

Assuming this to be true, (2.59) can be simplified to

$$E\{s'(k)\} = \alpha p' \frac{1 - (1 - (\alpha + 2^{-m}\beta))^k}{\alpha + 2^{-m}\beta} \quad (2.61)$$

or expressed in terms of its transient and steady state components.

$$E\{s'(k)\} = \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) p' - \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) (1 - (\alpha + 2^{-m}\beta))^k p' \quad (2.62)$$

In this form, it can be seen that the mean overlapped TEC coefficient vector increases in an exponential fashion to some asymptotic value.

The asymptotic value of the mean overlapped TEC coefficient vector is just the steady state component. In other words,

$$\lim_{k \rightarrow \infty} E\{s'(k)\} = E\{s'(\infty)\} = \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) p' \quad (2.63)$$

The mean overlapped TEC coefficient vector appears in the update equation for the mean MEC coefficient vector so we may use the result obtained here to find its nonrecursive representation. Substituting (2.62) into (2.49) and using the relationships between p' and q and Y and A given by (2.29) and (2.33) respectively we obtain

$$\begin{aligned} E\{m(k+1)\} &= (1 - 2^{-m}\beta)E\{m(k)\} \\ &\quad + \left(\frac{\alpha + 2^{-m}\beta}{\alpha + 2^{-m}\beta}\right)(1 - (\alpha + 2^{-m}\beta))^k A^T A q \\ &\quad + \left(\beta I - \left(\frac{\alpha + 2^{-m}\beta}{\alpha + 2^{-m}\beta}\right) A^T A\right) q \end{aligned} \quad (2.64)$$

A nonrecursive representation of (2.64) is obtained by induction from the initial zero state of the MEC coefficients. The result is

$$\begin{aligned} E\{m(k)\} &= \left(\frac{\alpha + 2^{-m}\beta}{\alpha + 2^{-m}\beta}\right) \sum_{i=0}^{k-1} (1 - 2^{-m}\beta)^{k-1-i} (1 - (\alpha + 2^{-m}\beta))^i A^T A q \\ &\quad + \left(\beta I - \left(\frac{\alpha + 2^{-m}\beta}{\alpha + 2^{-m}\beta}\right) A^T A\right) q \sum_{i=0}^{k-1} (1 - 2^{-m}\beta)^i \end{aligned} \quad (2.65)$$

(2.65) can be further rearranged to

$$\begin{aligned}
E\{m(k)\} = & \left(\frac{\alpha + 2^{-m}\beta}{\alpha + 2^{-m}\beta} \right) (1 - 2^{-m}\beta)^{k-1} \sum_{i=0}^{k-1} \left(\frac{1 - (\alpha + 2^{-m}\beta)}{1 - 2^{-m}\beta} \right)^i A^T A q \\
& + \left(\beta I - \left(\frac{\alpha + 2^{-m}\beta}{\alpha + 2^{-m}\beta} \right) A^T A \right) q \sum_{i=0}^{k-1} (1 - 2^{-m}\beta)^i \quad (2.66)
\end{aligned}$$

The convergence of the fractional term is guaranteed if the adaptation constants are chosen to satisfy the convergence requirements of the overlapped TEC coefficients in (2.60). The second term's convergence follows from the convergence of the fractional term because the range of β is limited to a smaller range than that required for convergence of the second term alone. Therefore, the convergence requirements for the mean overlapped TEC and MEC coefficients are identical.

The result of evaluating the summation terms in (2.66) and factoring is

$$\begin{aligned}
E\{m(k)\} = & \left(2^{-m} I - \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) A^T A \right) q \\
& - \left(\frac{2^{-m}\beta}{\alpha + 2^{-m}\beta} \right) (1 - (\alpha + 2^{-m}\beta))^k A^T A q \\
& - (1 - 2^{-m}\beta)^k (2^{-m} I - A^T A) q \quad (2.67)
\end{aligned}$$

(2.67) expresses the mean MEC coefficient vector in terms of 2 transient components and a steady state component. Like the mean overlapped TEC coefficient vector, the transient terms are decaying exponential functions. Note that the scalar part of the first transient

component is the "complement" of the scalar part of the mean overlapped TEC coefficient vector's transient component. The asymptotic value of the mean MEC coefficient vector is the steady state component. In other words,

$$\lim_{k \rightarrow \infty} E\{m(k)\} = E\{m(\infty)\} = \left(2^{-m} I - \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) A^T A \right) q \quad (2.68)$$

At this point, we have found expressions for the mean overlapped TEC and MEC coefficient vectors. Each are expressed as some function of their respective input/echo crosscorrelation vectors and both adaptation step-sizes. We have also discovered that each of the mean overlapped coefficient vectors converge to a single solution which is a simple function of their respective input/echo crosscorrelation vectors. This behaviour of the mean overlapped coefficient vectors is contrary to the previously expected "wandering" behaviour of the coefficients and is a direct result of the discovery of a deterministic relationship, given by (2.56), between the overlapped coefficient vectors.

We may now proceed to verify whether the asymptotic values of the overlapped coefficient vectors satisfy the conditions for MMSE in the mean sense.

Expanding (2.68) and using the relationship between p and q given by (2.29) results in

$$E\{m(\infty)\} = 2^m q - A^T \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) p' \quad (2.69)$$

Combining (2.69) and (2.63) we obtain the relationship between the mean overlapped coefficient vectors in the limit.

$$E\{m(\infty)\} = 2^m q - A^T E\{s'(\infty)\} \quad (2.70)$$

Comparing (2.70) with the condition for MMSE given by (2.45), we see that they are equivalent. Furthermore, if we multiply both sides of (2.70) by Y and simplify the result using the characteristics of Y given by (2.33) and (2.35), we obtain the other condition for MMSE given by (2.43). Therefore we may conclude that

$$s'^* = E\{s'(\infty)\} = \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) p' \quad (2.71)$$

$$m^* = E\{m(\infty)\} = \left(2^m I - \left(\frac{\alpha}{\alpha + 2^{-m}\beta} \right) A^T A \right) q \quad (2.72)$$

We have concluded that the overlapped TEC and MEC coefficients will converge to the MMSE solution at least in the mean sense.

We will now evaluate the overlapped TEC input/echo and MEC input/echo crosscorrelation vectors p' and q respectively. In order to evaluate the crosscorrelations, we must model the actual echo generation process.

$$d(k) = \mathbf{x}_h^T(k)\mathbf{h}_l + \mathbf{a}_h^T(k)\mathbf{h}_n + v(k) \quad (2.73)$$

where,

- $\mathbf{h}_l$ "Linear echo path response vector". A $(u,1)$ vector representing the parameters of the actual linear echo path response which is assumed to have a duration of u symbol periods.
- $\mathbf{h}_n$ "Nonlinear echo path response vector". A $(2^v,1)$ vector representing the parameters of the actual nonlinear echo path response which is assumed to have a duration of v symbols periods.
- $\mathbf{x}_h$ "Linear echo path input vector". A $(u,1)$ input symbol vector over which the echo path linearity exists.
- $\mathbf{a}_h$ "Nonlinear echo path input vector". A $(2^v,1)$ input "address" vector for addressing a single nonlinear echo path response parameter.
- $v(k)$ "Noise model". An independent, additive, and white noise source.

For our analysis, the following assumptions about the relationships between the TEC/MEC spans and the duration of the linear and nonlinear echo path response are made.

$$u \geq v$$

$$v \leq n \leq u$$

$$m \leq v$$

The above assumptions were made to reflect the conditions in practical situations.

In order to evaluate p' and q (2.73) is substituted into their definitions given by (2.26) and (2.28). The noise terms are zero because, by definition, the noise is independent and has zero mean.

$$p' = E\{x'(k)x_h^T(k)\}h_l + E\{x'(k)a_h^T(k)\}h_n \quad (2.74)$$

$$q = E\{a(k)x_h^T(k)\}h_l + E\{a(k)a_h^T(k)\}h_n \quad (2.75)$$

The expectations in (2.74) and (2.75) are correlations between the overlapped TEC and MEC input symbol vectors and the possibly longer symbol sequences which generate the actual echo. The correlations can be evaluated by reexpressing the input symbol vectors in terms of the echo input symbol vectors through the use of special transformation matrices defined below.

$$x'(k) = F_m x_h(k) \quad (2.76)$$

where,

$$F_m \equiv \begin{bmatrix} I_m & 0_{(m,u-m)} \end{bmatrix} \quad (2.77)$$

$0_{(m,u-m)}$ $(m,u-m)$ zero matrix

$$a(k) = F_\phi a_h(k) \quad (2.78)$$

where,

$$F_{\phi} \equiv [I_{\phi} \ I_{\phi} \ \dots \ I_{\phi}] \text{ where there are } 2^{v-m} \text{ } I_{\phi}'\text{'s} \quad (2.79)$$

$x'(k)$ is just a truncated version of $x_h(k)$ and the structure of F_m performs this truncation. $a(k)$ on the other hand is not a truncated version $a_h(k)$. The function of F_{ϕ} accounts for the MEC "addressing error" which occurs when the actual echo nonlinearity duration, v , is longer than the MEC length, m . An "addressing error" occurs because the MEC uses the address vector, $a(k)$, which is based on the first m bauds of the input symbol shift register. The nonlinear echo generation is based on the address, $a_h(k)$, which is dependent on the first v bauds of the input shift register. Given a particular m baud symbol pattern, the MEC will address and output a single corresponding memory coefficient. There are 2^{v-m} possible echo input symbol vectors of length v beginning with the same m baud symbol sequence. Consequently, there are not 1 but 2^{v-m} possible nonlinear echo values that could be output. The MEC cannot effectively adapt its single corresponding coefficient to the multitude of possible nonlinear echo output values and thus creates a permanent error component. F_{ϕ} translates the 2^{v-m} possible address vectors corresponding to a v -baud input symbol sequence, beginning with a particular m -baud input symbol sequence, into an address vector that corresponds only to the particular m -baud sequence.

Substituting (2.76) into (2.74) and (2.78) into (2.75) gives

$$\mathbf{p}' = \mathbf{F}_m \mathbf{E}\{\mathbf{x}_h(k)\mathbf{x}_h^T(k)\}\mathbf{h}_l + \mathbf{F}_m \mathbf{E}\{\mathbf{x}_h(k)\mathbf{a}_h^T(k)\}\mathbf{h}_n \quad (2.80)$$

$$\mathbf{q} = \mathbf{F}_\phi \mathbf{E}\{\mathbf{a}_h(k)\mathbf{x}_h^T(k)\}\mathbf{h}_l + \mathbf{F}_\phi \mathbf{E}\{\mathbf{a}_h(k)\mathbf{a}_h^T(k)\}\mathbf{h}_n \quad (2.81)$$

The echo path input autocorrelations of (2.80) and (2.81) are evaluated in the same way as the TEC and MEC input autocorrelations. The results are

$$\begin{aligned} \mathbf{E}\{\mathbf{x}_h(k)\mathbf{x}_h^T(k)\} &= \mathbf{I}_u && (u,u) \text{ identity matrix} \\ \mathbf{E}\{\mathbf{a}_h(k)\mathbf{a}_h^T(k)\} &= 2^{-v}\mathbf{I}_v && (2^v,2^v) \text{ identity matrix} \end{aligned}$$

We also define the "linear/nonlinear echo path input crosscorrelation" as follows.

"linear/nonlinear echo path input crosscorrelation matrix"

$$\mathbf{Y}_h \equiv \mathbf{E}\{\mathbf{x}_h(k)\mathbf{a}_h^T(k)\} \quad (2.82)$$

Using the evaluations of the autocorrelations and the definition above, (2.80) and (2.81) are reduced to

$$\mathbf{p}' = \mathbf{F}_m \mathbf{h}_l + \mathbf{F}_m \mathbf{Y}_h \mathbf{h}_n \quad (2.83)$$

$$\mathbf{q} = \mathbf{F}_\phi \mathbf{Y}_h^T \mathbf{h}_l + 2^{-v} \mathbf{F}_\phi \mathbf{h}_n \quad (2.84)$$

Therefore, by evaluating (2.83) and (2.84) and combining the results with the expressions given by (2.62), (2.63), (2.67), and

(2.68), we may completely characterize the behaviour of the mean overlapped coefficient vectors.

Turning our attention now towards the remaining nonoverlapped TEC coefficients, (2.12) is substituted into (2.11) and the expectation of both sides is taken. After applying (2.19), (2.20) and (2.27), we obtain

$$E\{s''(k+1)\} = (1 - \alpha)E\{s''(k)\} + \alpha p'' \quad (2.85)$$

This familiar recursion is that of a single TEC working alone. A nonrecursive form follows from induction from the initial zero coefficient state provided that α is limited as indicated below.

$$E\{s''(k)\} = p'' - (1 - \alpha)^k p'' \quad (2.86)$$

where,

$$0 < \alpha < 2 \quad (2.87)$$

Comparing this condition for convergence with that of the overlapped coefficients we see that the condition for the overlapped coefficients imposes a tighter bound on α . Therefore, since we are interested in the convergence of all mean coefficient vectors, α must be restricted to the bounds given by (2.60).

The asymptotic value is

$$\lim_{k \rightarrow \infty} E\{s''(k)\} = E\{s''(\infty)\} = p'' \quad (2.88)$$

It is immediately evident that $E\{s''(\infty)\}$ satisfies the MMSE condition given by (2.44) therefore

$$s''^* = E\{s''(\infty)\} = p'' \quad (2.89)$$

Lastly, we can express p'' as a function of the echo path response parameters as we did for p' and q by following a similar procedure. A transformation matrix, F_n , is defined so that $x''(k)$ can be expressed in terms of the echo path input symbol vector $x_h(k)$.

$$x''(k) = F_n x_h(k) \quad (2.90)$$

where,

$$F_n \equiv \begin{bmatrix} 0_{(n-m, m)} & I_{(n-m)} & 0_{(n-m, u-n)} \end{bmatrix} \quad (2.91)$$

F_n accounts for the fact that $x''(k)$ is that part of the echo input symbol vector which is associated with the non-overlapped section of the TEC/MEC. Substituting (2.90) into (2.27) and cancelling the noise term as before gives

$$p'' = F_n E\{x_h(k)x_h^T(k)\}h_1 + F_n E\{x_h(k)a_h^T(k)\}h_n \quad (2.92)$$

The echo path input correlations have already been encountered in the derivation of the p' and q . Applying the definition in (2.82) gives

$$p'' = F_n h_l + F_n Y_h h_n \quad (2.93)$$

Therefore, by evaluating (2.93) and combining the result with the expressions given by (2.86) and (2.88), we may completely characterize the behaviour of the mean non-overlapped TEC coefficient vector.

In this section we have discovered that, although there exists multiple solutions for the MMSE, the TEC/MEC coefficients will in fact adapt to a single stable MMSE solution in the mean sense provided that the adaptation constants are chosen within the bounds given by (2.60). A complete characterization of the mean coefficient vectors and their asymptotic values was performed.

2.2.1.4 AN EXPRESSION FOR MINIMUM MEAN SQUARE ERROR

The minimum mean square error (MMSE) is the value of the MSE when the coefficient vector values are such that the corresponding location on the MSE error surface is a minimum. We have already derived the conditions for MMSE in section 2.2.1.2 and have shown in section 2.2.1.3 that the mean coefficient vectors will satisfy those conditions in the limit. Substituting the coefficient vectors in (2.36) for the optimum ones and factoring, we have

$$\begin{aligned}
\text{MMSE} = \text{MSE}(\mathbf{s}'^*, \mathbf{s}''^*, \mathbf{m}^*) &= E\{d^2(k)\} + 2\mathbf{s}'^{*T}(\mathbf{s}'^* - \mathbf{p}' - \mathbf{Y}\mathbf{m}^*) \\
&+ 2\mathbf{m}^{*T}(2^{-m}\mathbf{m}^* - \mathbf{q} + \mathbf{Y}^T\mathbf{s}'^*) + 2\mathbf{s}''^{*T}(\mathbf{s}''^* - \mathbf{p}'') \\
&- \mathbf{s}'^{*T}\mathbf{p}' - \mathbf{m}^{*T}\mathbf{q} - \mathbf{s}''^{*T}\mathbf{p}'' \quad (2.94)
\end{aligned}$$

The terms in the parentheses are zero as a result of (2.43)-(2.45) and thus we have

$$\text{MMSE} = E\{d^2(k)\} - \mathbf{s}'^{*T}\mathbf{p}' - \mathbf{m}^{*T}\mathbf{q} - \mathbf{s}''^{*T}\mathbf{p}''$$

The above result may be simplified further by using the relationship between $\mathbf{p}'$ and $\mathbf{q}$ and $\mathbf{Y}$ and $\mathbf{A}$ given by (2.29) and (2.33) respectively.

$$\begin{aligned}
\text{MMSE} &= E\{d^2(k)\} - \mathbf{s}'^{*T}\mathbf{A}\mathbf{q} - \mathbf{m}^{*T}\mathbf{q} - \mathbf{s}''^{*T}\mathbf{p}'' \\
&= E\{d^2(k)\} - \mathbf{q}^T(\mathbf{A}^T\mathbf{s}'^* + \mathbf{m}^*) - \mathbf{s}''^{*T}\mathbf{p}'' \\
&= E\{d^2(k)\} - 2\mathbf{m}\mathbf{q}^T(\mathbf{Y}^T\mathbf{s}'^* + 2^{-m}\mathbf{m}^*) - \mathbf{s}''^{*T}\mathbf{p}''
\end{aligned}$$

The quantity in the parenthesis is just $\mathbf{q}$ by (2.45) and therefore (2.94) finally simplifies to

$$\text{MMSE} = E\{d^2(k)\} - 2\mathbf{m}\mathbf{q}^T\mathbf{q} - \mathbf{s}''^{*T}\mathbf{p}'' \quad (2.95)$$

We can express the MMSE in terms of the echo path parameters but first we must evaluate $E\{d^2(k)\}$, the average power of

the echo signal. Squaring (2.73), taking the expectation of the result, applying the previous evaluations of the echo path input correlations, and eliminating the first order noise terms as before we have

$$E\{d^2(k)\} = \mathbf{h}_l^T \mathbf{h}_l + 2\mathbf{h}_l^T \mathbf{Y}_h \mathbf{h}_n + 2\sigma^2 \mathbf{h}_n^T \mathbf{h}_n + E\{v^2(k)\} \quad (2.96)$$

The expression for MMSE can be constructed by combining (2.89), (2.95), and (2.96). The result is

$$\begin{aligned} \text{MMSE} = & \mathbf{h}_l^T \mathbf{h}_l + 2\mathbf{h}_l^T \mathbf{Y}_h \mathbf{h}_n + 2\sigma^2 \mathbf{h}_n^T \mathbf{h}_n - 2\mathbf{q}^T \mathbf{q} - \mathbf{p}^T \mathbf{p} \\ & + E\{v^2(k)\} \end{aligned} \quad (2.97)$$

where,

$$\mathbf{q} = \mathbf{F}_\phi \mathbf{Y}_h^T \mathbf{h}_l + 2\sigma^2 \mathbf{F}_\phi \mathbf{h}_n \quad \text{from (2.84)}$$

$$\mathbf{p} = \mathbf{F}_n \mathbf{h}_l + \mathbf{F}_n \mathbf{Y}_h \mathbf{h}_n \quad \text{from (2.93)}$$

(2.97) expresses the MMSE in terms of the echo path response vectors, $\mathbf{h}_l$ and $\mathbf{h}_n$, and the linear/nonlinear echo path input crosscorrelation matrix, $\mathbf{Y}_h$. The noise component of the MMSE is just the average power of the noise. This was expected as it was assumed that the noise was independent and hence uncorrelated with any of the other quantities.

It is important to note that the MMSE is a theoretical lower bound on the MSE. We have found that the mean value of the coefficient vectors satisfy the MMSE criteria, however, due to the fact that the LMS algorithm uses an approximation of the gradient of the

error surface to adapt the coefficient vectors, the coefficients will actually vary about their mean values in a noisy fashion. The expected consequence of this is that the actual steady state MSE is somewhat greater than the theoretical MMSE. Clearly, the convergence of the coefficients in the mean is not enough to guarantee an overall error convergence. The establishment of the conditions regarding the convergence of the coefficient variance about their means is equivalent to establishing conditions for the convergence of the Average Mean Square Error (AMSE). The AMSE will be investigated in the following sections.

2.2.2 MEAN SQUARE ERROR

In this section we wish to derive an expression for the Mean Square Error (MSE) as a function of time k . We begin with the expression for $e(k)$ in (2.12). Squaring both sides, taking an ensemble average and using the coefficient and input independence characteristics given by (2.19)-(2.23), an MSE expression is obtained that is similar to (2.36) but has the distinction of being expressed in terms of coefficient vectors that change with time k .

$$\begin{aligned} \text{MSE}(k) = E\{e^2(k)\} = E\{d^2(k)\} - 2s'^T(k)p' - 2m^T(k)q - 2s''^T(k)p'' \\ + 2s'^T(k)Ym(k) + s'^T(k)s'(k) + 2^{-m}m^T(k)m(k) \\ + s''^T(k)s''(k) \end{aligned} \quad (2.98)$$

We now define a set of "coefficient error" vectors as follows:

$$\mathbf{s}'_e(k) = \mathbf{s}'(k) - \mathbf{s}'^* \quad (2.99)$$

$$\mathbf{s}''_e(k) = \mathbf{s}''(k) - \mathbf{s}''^* \quad (2.100)$$

$$\mathbf{m}_e(k) = \mathbf{m}(k) - \mathbf{m}^* \quad (2.101)$$

The advantages of using these coefficient error vectors, instead of the coefficient vectors themselves, will become evident in the remaining analysis. (2.99)-(2.101) are substituted into (2.98). The result after factoring is

$$\begin{aligned} \text{MSE}(k) = & E\{d^2(k)\} - \mathbf{s}'^{*T} \mathbf{p}' - \mathbf{m}^{*T} \mathbf{q} - \mathbf{s}''^{*T} \mathbf{p}'' \\ & + 2\mathbf{s}'_e{}^T(k)(\mathbf{s}'^* - \mathbf{p}' + \mathbf{Y}\mathbf{m}^*) \\ & + 2\mathbf{m}_e{}^T(k)(2\mathbf{m}^* - \mathbf{q} + \mathbf{Y}^T \mathbf{s}'^*) \\ & + 2\mathbf{s}_e''{}^T(k)(\mathbf{s}''^* - \mathbf{p}'') \\ & + \mathbf{s}'^{*T}(\mathbf{s}'^* - \mathbf{p}' + \mathbf{Y}\mathbf{m}^*) \\ & + \mathbf{m}^{*T}(2\mathbf{m}^* - \mathbf{q} + \mathbf{Y}^T \mathbf{s}'^*) \\ & + \mathbf{s}''^{*T}(\mathbf{s}''^* - \mathbf{p}'') \\ & + \mathbf{s}'_e{}^T(k)\mathbf{s}'_e(k) \\ & + \mathbf{s}''_e{}^T(k)\mathbf{s}''_e(k) \\ & + 2\mathbf{m}_e{}^T(k)\mathbf{m}_e(k) \\ & + 2\mathbf{s}'_e{}^T(k)\mathbf{Y}\mathbf{m}_e(k) \end{aligned} \quad (2.102)$$

The 1st 4 terms constitute the MMSE expression given by the intermediate form of (2.95). All terms in the parentheses vanish as a result of the MMSE conditions in (2.43)-(2.45). The result is

$$\begin{aligned} \text{MSE}(k) = & \text{MMSE} + \mathbf{s}'_e{}^T(k)\mathbf{s}'_e(k) + \mathbf{s}''_e{}^T(k)\mathbf{s}''_e(k) \\ & + 2\mathbf{m}_e{}^T(k)\mathbf{m}_e(k) + 2\mathbf{s}'_e{}^T(k)\mathbf{Y}\mathbf{m}_e(k) \end{aligned} \quad (2.103)$$

(2.103) suggests that if all coefficient vectors adapted to the optimum coefficient vectors, the coefficient error vectors would be zero vectors. This eliminates the last 4 terms leaving the MMSE as the sole component. Due to the noisy error surface gradient estimate used by the LMS algorithm, the coefficient vectors will be noisy and, as we showed earlier, vary in an unbiased fashion about the optimum values after convergence has occurred. This means that the coefficient error vectors are noisy and have zero mean. The $\text{MSE}(k)$ depends on the inner products of the coefficient error vectors with themselves hence it is a quadratic function of the individual coefficient errors. Therefore, we expect that the steady state value of the average $\text{MSE}(k)$ will be some positive quantity in excess of the MMSE. In the next section, this excess quantity termed the average excess mean square error, will be examined.

2.2.3 AVERAGE EXCESS MEAN SQUARE ERROR

The expression for MSE in (2.103) is that of a noisy signal by virtue of the noisy coefficient adaptation process. A more useful expression is the ensemble average of that expression or the "average mean square error" (AMSE). The expectation of (2.103) is

$$\begin{aligned}
 \text{AMSE}(k) = & \text{MMSE} + E\{s_e'^T(k)s_e'(k)\} + E\{s_e''^T(k)s_e''(k)\} \\
 & + 2 \cdot m E\{m_e^T(k)m_e(k)\} \\
 & + 2E\{s_e'^T(k)Ym_e(k)\}
 \end{aligned} \tag{2.104}$$

The last 4 terms of (2.104) constitute the average mean squared error component in excess of the MMSE or the "average excess mean square error" (AMSE_{ex}). Thus (2.104) can be rewritten as

$$\text{AMSE}(k) = \text{MMSE} + \text{AMSE}_{\text{ex}}(k) \tag{2.105}$$

where,

$$\begin{aligned}
 \text{AMSE}_{\text{ex}}(k) = & E\{s_e'^T(k)s_e'(k)\} + E\{s_e''^T(k)s_e''(k)\} \\
 & + 2 \cdot m E\{m_e^T(k)m_e(k)\} \\
 & + 2E\{s_e'^T(k)Ym_e(k)\}
 \end{aligned} \tag{2.106}$$

One can interpret the terms in (2.106) as the average power of the coefficient errors and cross terms. These coefficient error power terms will be investigated in the next section.

2.2.3.1 COEFFICIENT ERROR POWERS

There are 4 coefficient error power terms to consider.

$$v'(k) = E\{s_e'^T(k)s_e'(k)\} \tag{2.107}$$

$$v''(k) = E\{s_e''^T(k)s_e''(k)\} \tag{2.108}$$

$$w(k) = 2 \cdot m E\{m_e^T(k)m_e(k)\} \tag{2.109}$$

$$z(k) = 2E\{s_e'^T(k)Ym_e(k)\} \quad (2.110)$$

In order to track the changes of the $AMSE_{ex}$ with time k , we need to know the coefficient error powers at every k . This can be done by initially deriving a recursive form of (2.107)-(2.110) by expressing the LMS coefficient update equations in (2.7), (2.10), and (2.11) in terms of coefficient error vectors and taking the expectation of the inner products of both sides. The initial coefficient states are known, so a nonrecursive form can be obtained from the results. The coefficient error power recursions are evaluated in APPENDIX A.3. The following are the results.

$$\begin{aligned} v'(k+1) = & (1 - 2\alpha + m\alpha^2)v'(k) + m\alpha^2v''(k) \\ & + m\alpha^2w(k) \\ & - \alpha(1 - m\alpha)z(k) \\ & + m\alpha^2MMSE \end{aligned} \quad (2.111)$$

$$\begin{aligned} v''(k+1) = & (n-m)\alpha^2v'(k) + (1 - 2\alpha + (n-m)\alpha^2)v''(k) \\ & + (n-m)\alpha^2w(k) \\ & + (n-m)\alpha^2z(k) \\ & + (n-m)\alpha^2MMSE \end{aligned} \quad (2.112)$$

$$\begin{aligned} w(k+1) = & 2^{-m}\beta^2v'(k) + 2^{-m}\beta^2v''(k) \\ & + (1 - 2 \cdot 2^{-m}\beta + 2^{-m}\beta^2)w(k) \\ & - 2^{-m}\beta(1 - \beta)z(k) \\ & + 2^{-m}\beta^2MMSE \end{aligned} \quad (2.113)$$

$$\begin{aligned}
z(k+1) = & -2 \cdot 2^{-m} \beta (1 - m\alpha) v'(k) + 2m\alpha \cdot 2^{-m} \beta v''(k) \\
& + 2m\alpha \cdot 2^{-m} \beta w(k) \\
& + (1 - \alpha - 2^{-m} \beta + 2m\alpha \cdot 2^{-m} \beta) z(k) \\
& - 2\alpha w_y(k) + 2m\alpha \cdot 2^{-m} \beta \text{MMSE}
\end{aligned} \tag{2.114}$$

$$\begin{aligned}
w_y(k+1) = & (2^{-m} \beta)^2 m v'(k) + (2^{-m} \beta)^2 m v''(k) \\
& + (2^{-m} \beta)^2 m w(k) - 2^{-m} \beta (1 - 2^{-m} \beta) z(k) \\
& + (1 - 2 \cdot 2^{-m} \beta) w_y(k) + (2^{-m} \beta)^2 m \text{MMSE}
\end{aligned} \tag{2.115}$$

Examining (2.111)-(2.115) we notice that for each coefficient error power, the next value is dependent on its current value and also the current values for the other coefficient error powers. In addition, the dependence of the next error power on its current value is determined by a factor of the form $(1-\epsilon)$ where ϵ is some small quantity. This prompts us to express (2.111)-(2.115) in vector and matrix notation as follows.

$$z(k+1) = (\mathbf{I} - \mathbf{T})z(k) + \text{MMSE} \cdot \mathbf{b} \tag{2.116}$$

where,

$$z(k) \equiv \begin{bmatrix} v'(k) \\ v''(k) \\ w(k) \\ z(k) \\ w_y(k) \end{bmatrix} \tag{2.117}$$

$$\mathbf{T} \equiv \begin{bmatrix} \alpha(2-m\alpha) & -m\alpha^2 & -m\alpha^2 & \alpha(1-m\alpha) & 0 \\ -(n-m)\alpha^2 & \alpha(2-(n-m)\alpha) & -(n-m)\alpha^2 & -(n-m)\alpha^2 & 0 \\ -2^{-m}\beta^2 & -2^{-m}\beta^2 & 2^{-m}\beta(2-\beta) & 2^{-m}\beta(1-\beta) & 0 \\ 2 \cdot 2^{-m}\beta(1-m\alpha) & -2m\alpha 2^{-m}\beta & -2m\alpha 2^{-m}\beta & \alpha + 2^{-m}\beta - 2m\alpha 2^{-m}\beta & 2\alpha \\ -(2^{-m}\beta)^2 m & -(2^{-m}\beta)^2 m & -(2^{-m}\beta)^2 m & 2^{-m}\beta(1-2^{-m}\beta) & 2 \cdot 2^{-m}\beta \end{bmatrix} \quad (2.118)$$

$$\mathbf{b} \equiv \begin{bmatrix} m\alpha^2 \\ (n-m)\alpha^2 \\ 2^{-m}\beta^2 \\ 2m\alpha 2^{-m}\beta \\ (2^{-m}\beta)^2 m \end{bmatrix} \quad (2.119)$$

The vector $\mathbf{b}$ is a constant vector which, as we shall see, partially determines the level of the AMSE_{e_x} at steady state conditions.

During an investigation into its properties, it was discovered that the matrix $\mathbf{T}$ was singular (ie. $\det(\mathbf{T}) = 0$, $\mathbf{T}^{-1}$ doesn't exist). Although it may not be evident at this point, this property hindered further analysis of $\mathbf{z}(k)$. The rank of $\mathbf{T}$ was found to be 4. This fact coupled with the fact that most of the entries in the last column were zero, warranted the possibility that the last row, which corresponds to $w_y(k)$, could be some linear function of the other rows. The last column suggested the possibility that this function involved $(2^{-m}\beta/\alpha) \cdot 4\text{th row}$. Beginning with this guess, and continuing with a "systematic" trial and error procedure, it was discovered that the last

row was infact a function of the first and fourth rows corresponding to $v'(k)$ and $z(k)$ respectively.

$$\text{last row of } \mathbf{T} = -\left(\frac{2^{-m}\beta}{\alpha}\right)^2 * \text{1st row} + \left(\frac{2^{-m}\beta}{\alpha}\right) * \text{4th row} \quad (2.120)$$

The relation in (2.120) allows us to express the recursion for $w_y(k)$ as follows.

$$\begin{aligned} w_y(k+1) = w_y(k) &+ \left(\frac{2^{-m}\beta}{\alpha}\right)^2 v'(k) - \left(\frac{2^{-m}\beta}{\alpha}\right) z(k) \\ &- \left(\frac{2^{-m}\beta}{\alpha}\right)^2 v'(k+1) + \left(\frac{2^{-m}\beta}{\alpha}\right) z(k+1) \end{aligned} \quad (2.121)$$

Evaluating (2.121) for the first few k and using induction, a nonrecursive form of $w_y(k)$ is obtained.

$$\begin{aligned} w_y(k) = w(0) &+ \left(\frac{2^{-m}\beta}{\alpha}\right)^2 v'(0) - \left(\frac{2^{-m}\beta}{\alpha}\right) z(0) \\ &- \left(\frac{2^{-m}\beta}{\alpha}\right)^2 v'(k) + \left(\frac{2^{-m}\beta}{\alpha}\right) z(k) \end{aligned} \quad (2.122)$$

Substituting (2.122) into (2.114) gives

$$\begin{aligned} z(k+1) = &-2*2^{-m}\beta(1 - m\alpha - 2^{-m}\beta/\alpha)v'(k) + 2m\alpha*2^{-m}\beta v''(k) \\ &+ 2m\alpha*2^{-m}\beta w(k) \\ &+ (1 - \alpha - 3*2^{-m}\beta + 2m\alpha*2^{-m}\beta)z(k) \\ &+ 2m\alpha*2^{-m}\beta \text{MMSE} \end{aligned}$$

$$- 2\alpha(w_y(0) + (2^{-m}\beta/\alpha)^2 v'(0) - (2^{-m}\beta/\alpha)z(0)) \quad (2.123)$$

The last term of (2.123) apparently breaks the regularity of the form of the coefficient error power equations. Fortunately, the following evaluation shows that this term is in fact 0. Initially, the coefficient error powers are just the powers of the optimum coefficients as shown below.

$$\begin{aligned} v'(0) &= E\{s_c^T(0)s_c(0)\} = s_c^T(0)s_c(0) = (\mathbf{0} - \mathbf{s}^{**})^T(\mathbf{0} - \mathbf{s}^{**}) \\ v'(0) &= \mathbf{s}^{**T}\mathbf{s}^{**} \end{aligned} \quad (2.124)$$

Similarly,

$$v''(0) = \mathbf{s}^{**T}\mathbf{T}\mathbf{s}^{**} \quad (2.125)$$

$$w(0) = 2^{-m}\mathbf{m}^*\mathbf{T}\mathbf{m}^* \quad (2.126)$$

$$z(0) = 2\mathbf{s}^{**T}\mathbf{Y}\mathbf{m}^* \quad (2.127)$$

$$w_y(0) = \mathbf{m}^*\mathbf{T}\mathbf{Y}\mathbf{T}\mathbf{Y}\mathbf{m}^* \quad (2.128)$$

Substituting (2.124), (2.127), and (2.128) into the last term of (2.123) gives

$$\begin{aligned} w_y(0) + (2^{-m}\beta/\alpha)^2 v'(0) - (2^{-m}\beta/\alpha)z(0) \\ = \mathbf{m}^*\mathbf{T}\mathbf{Y}\mathbf{T}\mathbf{Y}\mathbf{m}^* + (2^{-m}\beta/\alpha)^2 \mathbf{s}^{**T}\mathbf{s}^{**} - 2(2^{-m}\beta/\alpha)\mathbf{s}^{**T}\mathbf{Y}\mathbf{m}^* \end{aligned} \quad (2.129)$$

We recall the fact that the optimum coefficient vectors are just the asymptotic values of their mean trajectories. This allows us to use the relationship between the means of the overlapped coefficient vectors, given by (2.57), to simplify (2.129) as follows.

$$\begin{aligned}
 w_y(0) + (2^{-m}\beta/\alpha)^2 v'(0) - (2^{-m}\beta/\alpha)z(0) \\
 = (2^{-m}\beta/\alpha)^2 s'^* T s'^* + (2^{-m}\beta/\alpha)^2 s'^* T s'^* - 2(2^{-m}\beta/\alpha)^2 s'^* T s'^* \\
 = 0
 \end{aligned} \tag{2.130}$$

Therefore (2.123) becomes

$$\begin{aligned}
 z(k+1) = & -2 \cdot 2^{-m}\beta(1 - m\alpha - 2^{-m}\beta/\alpha)v'(k) + 2m\alpha \cdot 2^{-m}\beta v''(k) \\
 & + 2m\alpha \cdot 2^{-m}\beta w(k) \\
 & + (1 - \alpha - 3 \cdot 2^{-m}\beta + 2m\alpha \cdot 2^{-m}\beta)z(k) \\
 & + 2m\alpha \cdot 2^{-m}\beta \text{MMSE}
 \end{aligned} \tag{2.131}$$

The recursion in (2.131) has fully absorbed $w_y(k)$ hence the associated row and column of the matrix T can be eliminated to obtain an equivalent (4,4) representation. Furthermore, $z(k)$ now need not contain $w_y(k)$. We will refer to the new truncated $z(k)$ as the "coefficient error power vector". The recursion for the coefficient error power vector is still given by (2.116), however, we now have that

$$z(k) \equiv \begin{bmatrix} v'(k) \\ v''(k) \\ w(k) \\ z(k) \end{bmatrix} \text{ "coefficient error power vector"} \quad (2.132)$$

Likewise, the new matrix T is referred to as the "coefficient error power update matrix".

$$T \equiv \begin{bmatrix} \alpha(2-m\alpha) & -m\alpha^2 & -m\alpha^2 & \alpha(1-m\alpha) \\ -(n-m)\alpha^2 & \alpha(2-(n-m)\alpha) & -(n-m)\alpha^2 & -(n-m)\alpha^2 \\ -2^{-m}\beta^2 & -2^{-m}\beta^2 & 2^{-m}\beta(2-\beta) & 2^{-m}\beta(1-\beta) \\ 2 \cdot 2^{-m}\beta(1-m\alpha-(2^{-m}\beta/\alpha)) & -2m\alpha 2^{-m}\beta & -2m\alpha 2^{-m}\beta & \alpha+3 \cdot 2^{-m}\beta-2m\alpha 2^{-m}\beta \end{bmatrix}$$

"Coefficient Error Power Update Matrix" (2.133)

The vector b also becomes

$$b \equiv \begin{bmatrix} m\alpha^2 \\ (n-m)\alpha^2 \\ 2^{-m}\beta^2 \\ 2m\alpha 2^{-m}\beta \end{bmatrix} \quad (2.134)$$

We expect the new coefficient error power update matrix, T , to be nonsingular now that the number of rows and columns matches the rank of the original T matrix (ie. 4) and indeed this was found to be the case. We may now continue our analysis of $z(k)$ without further undue complications.

The expression for $AMSE_{ex}(k)$ given by (2.106) can be rewritten in a simpler form using the coefficient error power vector which contains all of the necessary terms.

$$AMSE_{ex}(k) = \Sigma\{z(k)\} \quad (2.135)$$

where,

$\Sigma\{\}$ denotes the summation of all elements within a vector

(2.135) can be expressed nonrecursively provided that the initial coefficient error powers are known. These have already been evaluated and are given by (2.124)-(2.127) hence,

$$z(k) = (I - T)^k z(0) + MMSE \cdot \left[\sum_{i=0}^{k-1} (I - T)^i \right] b \quad (2.136)$$

If we were dealing with scalar quantities, the convergence of (2.136) would depend on the restriction of T to a certain range of values. An equation of the form in (2.136), involving integer powers of square matrices, may be alternatively expressed in terms of the their eigenvalues and eigenvectors. This yields a particularly useful expression because the conditions for convergence are established in an analogous way to the scalar case by the restriction of the eigenvalues of the matrices to a certain range of values. In our case, this involves the "diagonalization" of T as shown below.

$$\mathbf{z}(k) = \mathbf{E}(\mathbf{I} - \Lambda_T)^k \mathbf{E}^{-1} \mathbf{z}(0) + \text{MMSE} * \mathbf{E} \left[\sum_{i=0}^{k-1} (\mathbf{I} - \Lambda_T)^i \right] \mathbf{E}^{-1} \mathbf{b} \quad (2.137)$$

where,

Λ_T a diagonal matrix containing the eigenvalues of T

E a matrix whose columns consist of the eigenvectors of T corresponding to the eigenvalues in Λ_T where

$$\mathbf{E}\mathbf{E}^{-1} = \mathbf{I} \quad (2.138)$$

and

$$\mathbf{E}^{-1} \mathbf{T} \mathbf{E} = \Lambda_T \text{ or } \mathbf{E} \Lambda_T \mathbf{E}^{-1} = \mathbf{T} \quad (2.139)$$

The "convergence" of the coefficient error power vector is necessary for convergence of the AMSE_{ex} . From (2.137), it is evident that the following condition must exist for convergence to occur.

$$0 < \lambda_{Tj} < 2 \quad j = 0, \dots, 3 \quad (2.140)$$

where,

λ_{Tj} the j th eigenvalue of T

The topic of convergence is discussed in the last section of this chapter. Meanwhile, we shall assume that the conditions given by (2.140) exist so that convergence of $\mathbf{z}(k)$ is possible. This allows us to evaluate the summation in (2.137) and factor the result to arrive at

$$\mathbf{z}(k) = (\mathbf{I} - \mathbf{T})^k (\mathbf{z}(0) - \text{MMSE} * \mathbf{T}^{-1} \mathbf{b}) + \text{MMSE} * \mathbf{T}^{-1} \mathbf{b} \quad (2.141)$$

where,

T^{-1} always exists. The existence of the inverse is equivalent to the statement that $\det(T) \neq 0$. This fact is ensured by the fact that the eigenvalues of T cannot be 0 by (2.140).

The form of (2.141) is that of an exponentially decaying vector which decays to a non-zero steady state vector. The first term is the transient term and the remaining term is the steady state term. The existence of the steady state term is a preview to the fact that the steady state AMSE value is not the MMSE but rather a greater value. Expressions for the $AMSE_{ex}$ and AMSE are given in the following sections.

2.2.3.2 AN EXPRESSION FOR AVERAGE EXCESS MEAN SQUARE ERROR

In the previous section, a nonrecursive expression of the coefficient error power vector was found. The $AMSE_{ex}$ is simply the sum of the elements in this vector. Substituting (2.141) into (2.135) results in

$$AMSE_{ex}(k) = \Sigma \{ (I - T)^k (z(0) - MMSE \cdot T^{-1}b) \} + MMSE \cdot \Sigma \{ T^{-1}b \} \quad (2.142)$$

In the next section we will derive the final expression of the AMSE using the result obtained here.

2.2.4 AN EXPRESSION FOR AVERAGE MEAN SQUARE ERROR

The AMSE is merely the sum of the MMSE and the $AMSE_{ex}$. Combining (2.142) and (2.105) gives

$$AMSE(k) = \Sigma \{ (I - T)^k (z(0) - MMSE \cdot T^{-1}b) \} + MMSE(1 + \Sigma \{ T^{-1}b \}) \quad (2.143)$$

where,

$$z(0) \equiv \begin{bmatrix} v'(0) \\ v''(0) \\ w(0) \\ z(0) \end{bmatrix}$$

where,

$v'(0)$ is given by (2.124)

$v''(0)$ is given by (2.125)

$w(0)$ is given by (2.126)

$z(0)$ is given by (2.127)

The first term is the transient term. The remaining term represents the steady state term. The steady state term clearly indicates that the AMSE will not reach the MMSE but rather a value that is fractionally greater than it. The matrix T and b determine the degree to which the steady state value is greater than the MMSE. We may conclude that due to the noisy coefficient adaptation process, the AMSE will not reach the theoretical MMSE.

2.3 CONVERGENCE BEHAVIOUR

In the previous sections we have encountered 2 types of convergence, namely, the coefficient convergence in the mean sense and the cancellation error in the mean square sense. In the analysis of the mean coefficient trajectories in section 2.2.1.3 and the coefficient error powers in section 2.2.3.1, conditions for the convergence of those quantities were established but were not elaborated upon. This section will provide a more detailed examination of the expressions derived for the coefficient trajectories and the AMSE and the corresponding conditions for their convergence. Conclusions will be validated by results obtained from simulations of the Jointly-Adaptive TEC/MEC.

2.3.1 MEAN COEFFICIENT CONVERGENCE

A theoretical description of the trajectory of the mean TEC/MEC coefficients is given by (2.62), (2.67), and (2.86). Their asymptotic values are given by (2.63), (2.68), and (2.88) respectively. Most important was the fact that these asymptotic values satisfied the conditions for MMSE. The conditions for convergence of the mean overlapped TEC/MEC coefficients and the non-overlapped TEC coefficients were established independently of one another. The requirement for mean convergence of both sets of coefficients was chosen as that which imposed the tightest bounds on the choice of adaptation step-sizes, namely, that of the overlapped coefficient set which is given by (2.60). However, convergence of the coefficients in

the mean sense is not enough to guarantee convergence of the type required by the LMS algorithm.

The aim of the LMS algorithm is the minimization and convergence of the cancellation error in the mean square sense. It is evident from (2.135) that convergence in the mean square sense requires the convergence of the coefficient error power vector. This second-order type of coefficient convergence is guaranteed by a different condition to that of (2.60) and is given by (2.140). It will be shown in the next section that this condition also guarantees the convergence of the mean coefficients. As a result of this fact, we will only study the asymptotic values of the mean TEC/MEC coefficients in more detail.

The asymptotic values of the mean TEC/MEC coefficients were calculated for several different TEC/MEC sizes and adaptation step-sizes using (2.63), (2.68), and (2.88). The adaptation step-sizes were chosen to achieve AMSE convergence within the bounds presented in the next section. A simulation of each case was run where a snapshot of the coefficients was taken after convergence. These were compared against their theoretically derived asymptotic values. The echo path was simulated as described in Appendix A.4. The case of no additive noise in the echo and -50 dB of noise were simulated. Fig. 2.1 and Fig. 2.2 present a set of typical results giving comparisons for the TEC and MEC coefficients respectively.

Fig. 2.1 A comparison between theoretical asymptotic mean TEC coefficient values and an actual snap shot of the TEC coefficients after convergence. $\alpha = 1.67 \text{ e-}2, \beta = 0.5$.

Fig. 2.2 A comparison between theoretical asymptotic mean MEC coefficient values and an actual snap shot of the MEC coefficients after convergence. $\alpha = 1.67 \text{ e-}2, \beta = 0.5$.

Note that even though an ensemble average of several coefficient sets was not performed, the agreement with the

theoretical asymptotic mean values was quite good. The maximum deviation was roughly 10% of the largest asymptotic value however most of other deviations were much lower than this. As expected, adding noise increased the deviations slightly. We have established the validity of our theoretical asymptotic mean coefficient expressions.

An important issue regarding the asymptotic coefficient values here is the degree to which sharing of the linear echo cancellation load occurs between the overlapped sections of the TEC and MEC. The theoretical expressions provide some insight into this. A common factor, $(\alpha/(\alpha+2^{-m}\beta))$, is visible in the expressions for the overlapped coefficients given by (2.63) and (2.68). The following gives the results of choosing one of the adaptation step-sizes to be equivalent or much larger than the other.

$$E\{s'(\infty)\} = \frac{1}{2} p' \quad (2.144)$$

$$E\{m(\infty)\} = \left(2^{-m} I - \frac{1}{2} A^T A \right) q \quad (2.145)$$

where,

$$\alpha = 2^{-m}\beta$$

$$E\{s'(\infty)\} = p' \quad (2.146)$$

$$E\{m(\infty)\} = (2^m I - A^T A) q \quad (2.147)$$

where,

$$\alpha \gg 2^{-m}\beta$$

$$E\{s'(\infty)\} \approx 0 \quad (2.148)$$

$$E\{m(\infty)\} \approx 2^m q \quad (2.149)$$

where,

$$2^{-m}\beta \gg \alpha$$

Fig. 2.3 and Fig. 2.4 give examples which illustrate the sharing of the linear echo cancellation load by the TEC and MEC respectively. They show the actual linear and nonlinear echo path coefficient values that were used to simulate the echo path compared with a snapshot of the TEC and MEC coefficients after convergence. The reason for these comparisons is that if the TEC was allowed to converge in the absence of the MEC, then the n -coefficient TEC would converge to the first n linear echo path coefficients. Similarly, in the absence of the TEC, the 2^m memory MEC would converge to the first 2^m nonlinear echo path coefficients. Therefore, during joint adaptation, any sharing of the linear echo cancellation load by the TEC and MEC coefficients would be evident by the deviations of the coefficients from the linear and nonlinear echo path coefficients respectively. These deviations are given for the case of no additive noise and -50 dB of additive noise in the echo signal.

Fig. 2.4 MEC coefficients after convergence illustrating the sharing of the cancellation load over the first $m = 5$ bauds of the linear echo. $n = 30, \alpha = 1.67e-2, \beta = 0.5$.

Fig 2.3 shows the deviation of the first $m=5$ TEC coefficients from the actual linear echo path coefficients can be seen. The first 5

TEC coefficients are clearly some fraction of the first 5 linear echo path coefficients. Fig. 2.4 shows the approximately 40-fold increase in dynamic range required by the MEC coefficients as a result of its handling part of the substantially larger linear echo path component. The effect of the noise on both the TEC and MEC coefficients is a slight increase in the deviation from their respective echo path coefficients.

In the simulation of the echo path, the power of the nonlinear component was designed to be a typical value of 40 dB below the linear component. Therefore it is not surprising that cancelling even a small fraction of the linear component can result in a relatively large increase in the magnitude of the MEC coefficients. This fact must be considered when choosing the MEC coefficient word size for a jointly-adaptive situation. A method of controlling the magnitude of the MEC coefficients such as coefficient saturation or wrap-around must be implemented for the case when they exceed the representable range of the selected word size.

2.3.2 AVERAGE MEAN SQUARE ERROR CONVERGENCE

The theoretical expression describing the behaviour of the AMSE over time is given by (2.143). It was arrived at by assuming that the condition for its convergence, given by (2.140), was true. In the following sections the transient and steady state characteristics of the AMSE will be investigated in detail. The investigation into the transient characteristics will address the problem of finding the

bounds of the region in which adaptation step-size values will guarantee AMSE convergence. It will also determine an optimum set of adaptation step-size values in the sense of fastest convergence time. The steady state analysis will describe the relationship between the adaptation step-sizes and the steady state AMSE.

2.3.2.1 TRANSIENT AMSE CHARACTERISTICS

(2.140) imposes a bound on the eigenvalues of the coefficient error power update matrix, T . An examination of the elements of this matrix, revealed by (2.133), exposes the fact that after choosing the size of the TEC and MEC, the matrix T is determined entirely by the TEC and MEC adaptation step-sizes, α and β respectively. Therefore, as expected, the bounds that are imposed on the eigenvalues of T are indirectly imposing bounds on α and β . In order to derive an equivalent condition of convergence in terms of the more tangible α and β quantities, it appears that some mapping between them and the eigenvalues of T must be found. This is complicated by the fact that obtaining a closed form expression of the eigenvalues of this 4×4 matrix is difficult when one considers that the task involves finding an analytical solution to its associated 4th order characteristic equation. This is the most straightforward approach and yields the greatest information because it provides an exact description of all 4 eigenvalues of T in terms of α and β .

The approach taken here is different and rather more economical in nature. We exploit the fact that the conditions only

require that all eigenvalues be between 0 and 2. This equivalent to stating that the smallest and largest eigenvalues must be between 0 and 2. Hence we see that the most straightforward approach gives more information than necessary. Instead of finding a closed form expression of the smallest and largest eigenvalues, a computer using numerical techniques was used to calculate the eigenvalues for a large range of positive α and β . A contour plot was generated with axes α and β and the contour lines representing loci of points of equal eigenvalues. The "region of convergence" (ROC) was determined simply by plotting only those contours with a "height" between 0 and 2. Fig. 2.5 and Fig. 2.6 give typical contour plots for the smallest and largest eigenvalue respectively. Note that the eigenvalues are plotted against a "normalized" TEC adaptation step, $n\alpha$. The reason for this arises from the conditions given by [Hay'86] for the AMSE convergence of a general transversal LMS-adaptive filter.

$$0 < \mu < \frac{2}{\text{input power}} \quad (2.150)$$

where,

μ	adaptation step-size
input power	"the sum of the mean-square values of the individual tap inputs"

In our case, the TEC input consists of n equiprobable binary ± 1 symbols with a total power of n . Therefore we have

$$0 < n\alpha < 2 \quad (2.151)$$

The MEC input is a 2^m -symbol addressing vector consisting of a single non-zero value of 1. The input power in this case is just 1, hence

$$0 < \beta < 2 \quad (2.152)$$

Using the normalized form of α , $n\alpha$, allows us to make a direct comparison with the square-shaped ROC defined by (2.151) and (2.152).

Fig. 2.5 A contour plot of the smallest eigenvalue of T . The outermost contour is of height 0. The innermost contour is of height 0.017. The spacing of the contours is 0.001. $n = 30, m = 5$.

Fig. 2.6 A contour plot of the largest eigenvalue of T . The bottom left hand contour is of height 0.005. The upper right hand corner contour is of height 0.2. The spacing of the contours is 0.005. $n = 30$, $m = 5$.

There are several significant conclusions that can be made from these contour plots.

- 1) The smallest eigenvalue of T , λ_{ϵ} , imposes a tighter bound on $n\alpha$ and β than the largest eigenvalue of T and therefore we need not worry about the largest eigenvalue of T .

- 2) The ROC convergence has a triangular shape that is exactly half of the ROC for the case of mutually exclusive adaptation. The ROC boundary is defined by the following relationship.

$$0 < n\alpha + \beta < 2 \quad (2.153)$$

The bounds imposed on the adaptation step-sizes for coefficient convergence in the mean are given by (2.60). Comparing those bounds to those above, it becomes clear that the bounds governing the convergence of the AMSE are tighter than those for mean coefficient convergence. Therefore, (2.153) describes a ROC for convergence in the mean square and mean sense. Choosing the adaptation step-sizes within this region guarantees that, in the limit, the TEC/MEC coefficients will vary in a bounded fashion about their optimum values given by (2.71), (2.72), and (2.89). For $n\alpha + \beta$ outside this range of (2.153), the corresponding eigenvalues of T are 0 or negative. This is below the lower bound of 0 imposed on the eigenvalues by (2.140).

- 3) The eigenvalue function of the adaptation step-sizes is unimodal with a single maximum. (2.137) indicates that the overall convergence time is determined by the smallest eigenvalue of T , λ_{ϵ} . Therefore, the adaptation step-sizes which produce the maximum λ_{ϵ} are optimum in the sense of convergence time.

- 4) The exact location of the maximum λ_e warrants further attention.

(2.153) is equivalent to (2.140) but is expressed in terms of the more tangible TEC/MEC adaptation step-sizes, α and β . We have arrived at this result avoiding the more direct approach requiring the analytical determination of the closed form expressions of the eigenvalues of T . Simulations of the Jointly-Adaptive TEC/MEC were performed to test the ROC boundary given by (2.153). Adaptation step-sizes were chosen just inside and just outside of the ROC boundary. At each point, it was noted whether or not the jointly-adaptive TEC/MEC converged. This was done for several combinations of n and m . TABLE 2.1 summarizes the result of the ROC testing for the case of $n = 30$ and $m = 5$.

TABLE 2.1 Results of the testing of the outer ROC boundary defined by $n\alpha + \beta = 2$ for $n = 30$ and $m = 5$. C = Converged, NC = Did not converge.

$n\alpha$	β	$n\alpha + \beta$	C/NC
0.25	1.73	1.98	C
0.25	1.77	2.02	NC
0.75	1.23	1.98	C
0.75	1.27	2.02	NC
1.25	0.73	1.98	C
1.25	0.77	2.02	NC
1.75	0.23	1.98	C
1.75	0.27	2.02	NC

As Table 2.1 indicates, the outer ROC boundary defined by (2.153) is quite precise. Fig. 2.7 gives simulation results that illustrate the sensitivity of AMSE error convergence to the outer ROC boundary.

2

Micred
LITHOGRAPH

Fig. 2.7 AMSE convergence sensitivity to the choice of adaptation step-sizes near the ROC boundary. The divergent curve is the result of choosing $n\alpha = 0.75$ and $\beta = 1.27$ hence $n\alpha + \beta = 2.02$. The converging curve is the result of choosing $n\alpha = 0.75$ and $\beta = 1.23$ hence $n\alpha + \beta = 1.98$.

We will now investigate the location of the maximum λ_e which is responsible for giving the fastest convergence. It can be shown [Fal'87] that the AMSE convergence of a n -coefficient TEC adapting with equiprobable binary ± 1 symbols in the absence of a MEC is determined by the following factor.

$$(1 - 2\alpha + n\alpha^2)^k$$

Similarly, the factor governing the convergence of a 2^m -coefficient MEC adapting in the absence of the TEC is given by

$$(1 - 2 \cdot 2^{-m}\beta + 2^{-m}\beta^2)^k$$

The optimum adaptation step-sizes for mutually exclusive coefficient adaptation are derived by finding the α and β that minimize the above factors. Equivalently, the quadratic functions that are subtracted from 1 are maximized and the results are

$$(2\alpha_{\text{opt}} - n\alpha_{\text{opt}}^2) = \frac{1}{n}$$

where,

$$n\alpha_{\text{opt}} = 1 \tag{2.154}$$

$$(2 \cdot 2^{-m}\beta_{\text{opt}} - 2^{-m}\beta_{\text{opt}}^2) = \frac{1}{2^m}$$

where,

$$\beta_{\text{opt}} = 1 \quad (2.155)$$

We now wish to investigate if the optimum α and β for the jointly-adaptive case are related in any way to those of the mutually exclusive adaptation case shown above.

A computer "hill-climbing" technique was used to search the unimodal ROC "surface" to find the maximum λ_e for different n and m and hence iteratively determining the adaptation step-sizes for optimum convergence time. The results are shown in Table 2.2.

TABLE 2.2 Computed adaptation step-sizes for optimum convergence time and maximum λ_e for $n = 16, 20, 25, 30$ and $m = 3, 4, 5, 6$. Search step-size for $n\alpha$ and $\beta = 0.01$.

n	m	$n\alpha_{\text{opt}}$	β_{opt}	$\lambda_e (* 10^{-2})$
16	3	0.70	0.33	4.57
16	4	0.51	0.51	3.42
16	5	0.33	0.69	2.25
16	6	0.19	0.83	1.32
20	3	0.75	0.28	3.86
20	4	0.57	0.45	3.01
20	5	0.39	0.64	2.06
20	6	0.23	0.78	1.25
25	3	0.79	0.23	3.23
25	4	0.63	0.39	2.61
25	5	0.44	0.58	1.87
25	6	0.28	0.74	1.18
30	3	0.82	0.20	2.78
30	4	0.67	0.35	2.31
30	5	0.49	0.53	1.71
30	6	0.32	0.70	1.11

There are several significant conclusions that can be made from the results given in TABLE 2.2.

- 1) The expected result that TEC/MEC structures covering larger spans exhibit longer convergence times is reaffirmed by the fact that λ_e decreases with increasing n and m . A smaller λ_e implies a longer overall convergence time.
- 2) The normalized optimum TEC adaptation step-size and the MEC adaptation step-size have the following relationship.

$$n\alpha_{\text{opt}} + \beta_{\text{opt}} \equiv 1 \quad (2.156)$$

This is similar to (2.154) and (2.155), however, now it is the sum of the adaptation step-sizes that equals unity. Therefore the optimum step-sizes will be some complementary fractions of unity.

- 3) The fraction of unity that the normalized optimum TEC adaptation step-size and the MEC adaptation step-size receive appears to be determined by the following trial-and-error-determined relationship.

$$n\alpha_{\text{opt}} = \frac{n}{n+2m} \quad (2.157)$$

or

$$\alpha_{\text{opt}} = \frac{1}{n+2m} \quad (2.158)$$

and

$$\beta_{\text{opt}} = \frac{2m}{n+2m} \quad (2.159)$$

(2.158) and (2.159) give expressions for the adaptation step-sizes for optimum convergence time. It is easily seen that

$$\beta_{\text{opt}} = 2^m \alpha_{\text{opt}} \quad (2.160)$$

This is rather an intuitive result. It can be attributed to the nature in which the MEC adapts its coefficients. The MEC updates only 1 of its 2^m coefficients for each iteration of the LMS algorithm whereas the TEC updates all of its coefficients. This means that it takes the MEC 2^m times longer than the TEC to update all of its coefficients. (2.160) can be viewed as a "mathematical handicap rule" that allows the MEC to adapt at a more equitable pace with respect to the TEC.

The expressions for the adaptation step-sizes for optimum convergence time were tested by simulation for different sets of n and m . Besides the optimum adaptation step-sizes, 4 other sets of adaptation step-sizes were chosen. These specified points that were equally spaced around the optimum point within the ROC. Fig. 2.8 gives a typical set of AMSE curves for the case of $n = 30$ and $m = 5$.

Fig. 2.8 Verifying the optimum step-size for fastest convergence for $n = 30$ and $m = 5$. The fastest converging curve corresponds to $\alpha = \alpha_{opt} = 1.61 \text{ e-}2$ and $\beta = \beta_{opt} = 5.16 \text{ e-}1$. The others correspond to a) $\alpha = \alpha_{opt} + 0.4/n$, $\beta = \beta_{opt}$, b) $\alpha = \alpha_{opt}$, $\beta = \beta_{opt} + 0.4$, c) $\alpha = \alpha_{opt} - 0.4/n$, $\beta = \beta_{opt}$, d) $\alpha = \alpha_{opt}$, $\beta = \beta_{opt} - 0.4$.

2.3.2.2 STEADY STATE AMSE CHARACTERISTICS

In the previous section we defined a ROC in which AMSE convergence is guaranteed. We found the optimum set of adaptation step-sizes with regards to convergence time. However, convergence time is not the only factor to consider. We will now investigate how the choice of adaptation step-size affects the steady state AMSE level.

The steady state AMSE is given by the 2nd term of the AMSE expression in (2.143).

$$\text{AMSE}_{ss} = \text{MMSE}(1 + \Sigma\{\mathbf{T}^{-1}\mathbf{b}\}) \quad (2.161)$$

The summation term $\Sigma\{\mathbf{T}^{-1}\mathbf{b}\}$ is the factor that determines the extent to which the AMSE_{ss} exceeds the MMSE. The constituents of the coefficient error power update matrix, $\mathbf{T}$, and $\mathbf{b}$ are given by (2.133) and (2.134). The expressions indicate that $\mathbf{T}$ and $\mathbf{b}$, and hence $\Sigma\{\mathbf{T}^{-1}\mathbf{b}\}$, are entirely determined by n, m, α and β . For a fixed TEC and MEC span, this term was computed over the adaptation step-size values that lay in the ROC. A contour plot with the same axes as the ROC was generated for $\Sigma\{\mathbf{T}^{-1}\mathbf{b}\}$. An example plot need not be shown here because of its simple nature. The plot consisted of parallel contours that had the form

$$n\alpha + \beta = c \quad (2.162)$$

where,

$$c > 0$$

The result implies that different sets of adaptation step-sizes chosen along the same contour will result in the same $AMSE_{ss}$. By (2.153), it can be seen that these contours are also parallel to the outermost contour of the ROC. Furthermore, the value of the contours depended on c in the following way.

$$\begin{aligned} \Sigma(T^{-1}b) &= 1, & c &= 1 \\ &< 1, & c &< 1 \\ &> 1, & c &> 1 \end{aligned} \tag{2.163}$$

The case where $c = 1$ is familiar because, by (2.156), it is the relationship of the adaptation step-sizes for optimum convergence time. Therefore we may conclude that when the adaptation step-sizes are chosen for optimum convergence time, then the $AMSE_{ss}$ is precisely double the value or 3 dB above the MMSE. The other cases merely indicate that as adaptation step-sizes are chosen along contours parallel and closer to the ROC boundary, the resulting AMSE is higher. It was also noticed that the spacing between the contours decreased as c increased. Therefore, not only does the $AMSE_{ss}$ increase towards the ROC boundary, but the increase is exponential in nature.

All of the above conclusions were verified by simulation for different n and m . The $AMSE_{ss}$ was computed using (2.161) and also determined by simulation for a selection of $n\alpha$ and β along the

contours defined by (2.162). The MMSE was computed using (2.97). TABLE 2.3 displays the results for the case of $n = 30$ and $m = 5$.

TABLE 2.3 Theoretical and simulated $AMSE_{ss}$ values for $c = 0.5, 1, 1.5$. Computed MMSE = -70 dB.

c	$n\alpha$	β	$\Sigma\{T^{-1}b\}$	Computed $AMSE_{ss}$ (dB)	Actual $AMSE_{ss}$ (dB)
0.50	0.10	0.40	0.33	-68.75	-68.5
"	0.25	0.25	"	"	-68.4
"	0.40	0.10	"	"	-68.7
1.00	0.10	0.90	1.00	-66.99	-67.1
"	0.50	0.50	"	"	-67.3
"	0.90	0.10	"	"	-67.3
1.50	0.10	1.40	3.00	-63.98	-64.0
"	0.75	0.75	"	"	-64.3
"	1.40	0.10	"	"	-64.3

Note that the simulation results given by TABLE 2.3 strongly support the notions given by (2.161)-(2.163).

2.3.2.3 ACCURACY OF THE THEORETICAL AMSE EXPRESSION

In this section, the accuracy of the theoretical expression of the AMSE, given by (2.143) will be illustrated. (2.143) describes the behaviour of the AMSE over time when the adaptation step-sizes are

chosen within the ROC. Fig. 2.9 and Fig. 2.10 give comparisons between the theoretical and simulated AMSE for the case of $n = 30$ and $m = 5$ with different step-sizes within the ROC. The case where no additive noise exists in the echo is given in Fig. 2.9. Fig. 2.10 illustrates the effect on the AMSE of a -50 dB independent white additive noise component introduced into the echo signal.

Fig. 2.9 A comparison between the theoretical and simulated AMSE(k) for $n = 30$, $m = 5$, and no additive noise. The smooth curves represent the theoretical AMSE. From the slowest to the fastest converging curve, the adaptation step-sizes are

a) $\alpha = 3.00 \text{ e-}2$, $\beta = 9.00 \text{ e-}1$, b) $\alpha = 3.33 \text{ e-}3$, $\beta = 1.50 \text{ e-}1$
 c) $\alpha = 1.67 \text{ e-}2$, $\beta = 5.00 \text{ e-}1$

Fig. 2.10 A comparison between the theoretical and simulated AMSE(k) for $n = 30$, $m = 5$, and -50 dB additive independent white noise introduced into the echo signal. The smooth curves represent the theoretical AMSE. From the slowest to the fastest converging curve, the adaptation step-sizes are

a) $\alpha = 3.00 \text{ e-}2$, $\beta = 9.00 \text{ e-}1$, b) $\alpha = 3.33 \text{ e-}3$, $\beta = 1.50 \text{ e-}1$
 c) $\alpha = 1.67 \text{ e-}2$, $\beta = 5.00 \text{ e-}1$

The comparisons given by Fig. 2.9 and Fig. 2.10 indicate that the theoretical expression for AMSE given by (2.143) is a reasonable predictor of the convergence time but a better predictor of the steady state AMSE level. Similar comparisons were made for smaller values of n and m and the predictions were noticeably better. Fig. 2.11 and Fig. 2.12 illustrate the better agreement between theory and the simulated results when n and m are made smaller. In particular, the case for $n = 16$ and $m = 4$ for the noise and no noise cases are given.

Fig. 2.11 A comparison between the theoretical and simulated AMSE(k) for $n = 16$, $m = 4$, and no additive noise. The smooth curves represent the theoretical AMSE. From the slowest to the fastest converging curve, the adaptation step-sizes are

a) $\alpha = 5.938 \text{ e-}2$, $\beta = 9.50 \text{ e-}1$, b) $\alpha = 6.250 \text{ e-}3$, $\beta = 1.50 \text{ e-}1$
 c) $\alpha = 3.125 \text{ e-}2$, $\beta = 5.00 \text{ e-}1$

Fig. 2.12 A comparison between the theoretical and simulated $AMSE(k)$ for $n = 16$, $m = 4$, and -50 dB additive independent white noise introduced into the echo signal. The smooth curves represent the theoretical $AMSE$. From the slowest to the fastest converging curve, the adaptation step-sizes are

a) $\alpha = 5.938 \text{ e-}2$, $\beta = 9.50 \text{ e-}1$, b) $\alpha = 6.250 \text{ e-}3$, $\beta = 1.50 \text{ e-}1$
 c) $\alpha = 3.125 \text{ e-}2$, $\beta = 5.00 \text{ e-}1$

The differences in agreement between theory and simulation were attributed to the many approximations concerning independence of the input symbols and coefficients that were made to simplify the derivation of the theoretical AMSE expression. It is apparent that these approximations become less reliable for larger n and m .

2.4 SUMMARY

In this chapter, the behaviour of the jointly-adaptive TEC/MEC was investigated from both theoretical and simulation-oriented viewpoints.

The theoretical study provided expressions describing the minimum mean square error (MMSE) and the associated TEC/MEC coefficient relationships, the mean behaviour of the TEC/MEC coefficients as they were adapted using the LMS algorithm, and the resulting average mean square error (AMSE).

There were 2 classifications made of the TEC/MEC coefficients. The MEC was responsible for cancelling the nonlinear component over the first m bauds of echo path response and the TEC was responsible for cancelling the linear component over the first n bauds where it was assumed for practical purposes that $n \geq m$. The "overlapped" coefficients consisted of all MEC coefficients and the first m TEC coefficients between which was shared a common input symbol span. The "nonoverlapped" coefficients consisted of the remaining $n-m$ TEC coefficients that did not share any input span

with the MEC. The study of the MSE as a function of the overlapped and nonoverlapped coefficients, (2.36), (2.39), revealed that there were many solutions, (2.46), to the minimum of this "error surface". Three expressions, (2.43)-(2.45), were found that described the relationships between the overlapped and nonoverlapped coefficients at the minimum. Two of them (2.43), (2.45), were dependent and described the relationships between the overlapped TEC and MEC coefficients. The other independent expression, (2.44), described the conditions for the nonoverlapped TEC coefficients.

The mean coefficient trajectory analysis revealed that the mean of the coefficients, (2.62), (2.67), (2.86), converges to a MMSE solution, (2.71), (2.72), (2.89), provided that the adaptation step-sizes are chosen to be within certain bounds (2.60). This was attributed to the discovery that, at any time, the TEC coefficients were a function of the MEC coefficients (2.56). The MEC coefficients would take on the added duty of cancelling a portion of the linear echo due to the span over which the MEC operated. The degree to which this cancellation load sharing occurred was attributed to a ratio function of the adaptation step-sizes.

The AMSE analysis produced an expression of the AMSE behaviour over time based on a set of coefficient error power quantities and an associated coefficient error power update matrix, T , whose elements were simple functions of n , m , and the adaptation step-sizes α and β (2.143). The coefficient error quantities consisted of the differences between the overlapped and nonoverlapped

coefficients and their associated optimum values. The transient behaviour of the AMSE was found to be governed by the eigenvalues of T (2.137). The conditions for AMSE convergence were also established in terms of the eigenvalues of T (2.140). The excess AMSE, or the component that causes the steady state AMSE to exceed of the MMSE, was found to be dependent on the MMSE itself and some factor containing T^{-1} (2.142).

The simulation studies revealed the acceptable accuracy of the theoretical expressions describing the asymptotic value of the mean coefficients and the evolution of the AMSE over time. A more detailed investigation into the convergence of the AMSE yielded several significant characteristics. The smallest eigenvalue of T was found to be the key factor that governed whether or not the echo canceller converged and also the rate of convergence. The determination of the adaptation step-sizes that allowed the smallest eigenvalue of T to remain within the bounds imposed for AMSE convergence yielded a contour plot that showed a "region of convergence", (2.153), that was precisely half of that of a combined TEC/MEC structure whose TEC coefficients were adapted in the absence of any MEC adaptation and vice versa (2.151), (2.152). Furthermore, the surface function determining the values of the eigenvalues within the region of convergence was found to be positive and unimodal and hence the set of TEC and MEC adaptation step-sizes corresponding to the maximum point on this surface was optimum in the sense of convergence time. Expressions for the

optimum adaptation step-sizes were found after examining their values for many n and m (2.158), (2.159). The theoretically derived optimum adaptation step-sizes were verified by simulation.

A contour plot of the factor determining the excess AMSE component was examined. It revealed that the steady state AMSE level increased as adaptation step-sizes were chosen closer to the outer region of convergence boundary. It was also shown that adaptation step-sizes chosen along contours parallel to the outer region of convergence boundary gave the same steady state AMSE level (2.162), (2.163).

CHAPTER 3

ANALYSIS OF THE SEQUENTIALLY-ADAPTIVE TEC/MEC

3.1 THE LMS ALGORITHM FOR SEQUENTIAL ADAPTATION

The TEC and MEC coefficients of the Sequentially-Adaptive TEC/MEC are adapted in a 2-phase sequence using the LMS algorithm. The approach is the same as that described in [Ada'88] with the exception that a multi-memory type of MEC was used instead of the simpler single-memory MEC used here.

In the initial adaptation phase, all n TEC coefficients are updated while the MEC is idle. This phase allows the TEC to adapt to the linear portion of the echo path response without any interaction with the MEC. Eventually, the coefficients will converge to some solution and the AMSE will reach a steady state value. At this time, the 2nd phase is initiated. The updating of the first m TEC coefficients is halted and the 2^m MEC coefficients are allowed to adapt. The TEC output procedure and the updating of the remaining $n-m$ coefficients remains unchanged from the initial phase. Eventually, the MEC coefficients will converge to some solution and

the AMSE will reach a second steady state value. The MEC coefficients and the remaining $n-m$ TEC coefficients continue to adapt after this initial convergence. In this case, further tracking of the first m bauds of the linear echo path response component is handled by the MEC.

The following equations describe the 2 coefficient adaptation phases.

Phase I

$$\text{TEC/MEC Output}(k) = \mathbf{s}^T(k)\mathbf{x}(k) \quad (3.1)$$

$$\mathbf{s}(k+1) = \mathbf{s}(k) + \alpha_1 e_1(k)\mathbf{x}(k) \quad (3.2)$$

$$\mathbf{m}(k) = \mathbf{m}(0) = \mathbf{0} \quad (3.3)$$

where,

$$e_1(k) = d(k) - \mathbf{s}^T(k)\mathbf{x}(k) \quad (3.4)$$

Phase II

$$\text{TEC/MEC Output}(k) = \mathbf{s}'^T \mathbf{x}'(k) + \mathbf{s}''^T \mathbf{x}''(k) + \mathbf{m}^T(k)\mathbf{a}(k) \quad (3.5)$$

$$\mathbf{s}''(k+1) = \mathbf{s}''(k) + \alpha_1 e_2(k)\mathbf{x}''(k) \quad (3.6)$$

$$\mathbf{m}(k+1) = \mathbf{m}(k) + \beta_1 e_2(k)\mathbf{a}(k) \quad (3.7)$$

where,

$$\begin{aligned} e_2(k) = & d(k) - \mathbf{s}'^T \mathbf{x}'(k) - \mathbf{s}''^T \mathbf{x}''(k) \\ & - \mathbf{m}^T(k)\mathbf{a}(k) \end{aligned} \quad (3.8)$$

In (3.5)-(3.8) we have assumed that during Phase I, the entire TEC coefficient vector will converge and the first m coefficients will

be frozen at s'_f . The remaining TEC coefficients, s'' , will continue to be adapted. The TEC and MEC adaptation step-sizes for the Sequentially-Adaptive TEC/MEC are denoted as α_1 and β_1 respectively to avoid confusion in later comparisons of performance with the Jointly-Adaptive TEC/MEC configuration.

3.2 AVERAGE MEAN SQUARE ERROR

In this section, an expression describing the evolution of the AMSE over time k is derived separately for Phase I and Phase II. The analysis proceeds in the same manner as that of the jointly-adapted scheme. The goal is to identify the transient and steady state components of the AMSE for the sequential adaptation method so that they can be compared to those of the jointly-adapted case.

3.2.1 MINIMUM MEAN SQUARE ERROR

3.2.1.1 THE ERROR SURFACE

The error surface for the Sequentially-Adaptive TEC/MEC during Phase II is precisely the same as that for the Jointly-Adaptive TEC/MEC configuration. In the derivation of the error surface, the only factors affecting the surface topology are the characteristics of the input and the coefficient vectors. Both the input and the coefficient vectors used in Phase II are the same as their combined configuration counterparts. The error surface is thus given by (2.36). However, during Phase I, the MEC is idle and hence not contributing to reducing the error. Accordingly, the error surface that the

TEC/MEC sees is somewhat simpler. The error surface seen by a sole TEC is a well known result of the "classical" analysis of the LMS algorithm. The result obtained by following a description given in [Ste'85] is

$$\text{MSE}(\mathbf{s}) = E\{e_1^2(k)\} = E\{d^2(k)\} - 2\mathbf{s}^T \mathbf{p} + \mathbf{s}^T \mathbf{s} \quad (3.9)$$

where,

$$\mathbf{p} = E\{d(k)\mathbf{x}(k)\}$$

but,

$$\mathbf{x}(k) = \begin{bmatrix} \mathbf{x}'(k) \\ \mathbf{x}''(k) \end{bmatrix}$$

therefore by (2.26) and (2.27),

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}' \\ \mathbf{p}'' \end{bmatrix} \quad (3.10)$$

where,

$\mathbf{p}'$ is given by (2.83)

$\mathbf{p}''$ is given by (2.93)

The input autocorrelation matrix, $\mathbf{R}$, that generally appears in this expression is an identity matrix in our case. This result is just an extension of the results given by (2.30) and (2.31). It will subsequently not appear in any further expressions.

3.2.1.2 CONDITIONS FOR MINIMUM MEAN SQUARE ERROR

During phase I, the combined TEC/MEC is just a TEC. The nonlinear echo component is not cancelled and therefore we expect the corresponding MMSE to be greater than that of phase II. From [Ste'85] we have that a single solution for the MMSE exists.

$$\mathbf{s}^* - \mathbf{p} = \mathbf{0} \quad (3.11)$$

During phase II, all TEC/MEC coefficients are contributing to reducing the echo. Therefore, regardless of the adaptation method used, the coefficient vectors must have the relationships given by (2.43)-(2.45) for the MMSE to be achieved.

3.2.1.3 COEFFICIENT TRAJECTORY ANALYSIS

It has been already shown [Ste'85] that the mean of the TEC coefficient vector, $\mathbf{s}(k)$, changes in an exponential fashion. The behaviour in this case is

$$\mathbf{E}\{\mathbf{s}(k)\} = (1 - (1 - \alpha_1)^k)\mathbf{p} \quad (3.12)$$

The asymptotic value is

$$\lim_{k \rightarrow \infty} \mathbf{E}\{\mathbf{s}(k)\} = \mathbf{E}\{\mathbf{s}(\infty)\} = \mathbf{p} = \mathbf{s}^* \quad (3.13)$$

provided that,

$$0 < \alpha_1 < 2 \quad (3.14)$$

In phase II, the MEC is allowed to adapt while the first m coefficients of the TEC, $s'(k)$, are "frozen" and the remaining $n-m$ continue to adapt. We will assume that the TEC has converged significantly enough so that we may state that

$$s'_f \approx E\{s'(\infty)\} = p' \quad (3.15)$$

Furthermore, we will assume that the echo path parameters are varying very slowly with time so that we can restrict ourselves to analyzing the initial convergence behaviour only.

We have discovered from section 2.2.1.3 that the last $n-m$ TEC coefficients adapt independently from the other coefficients. Therefore, freezing the first m TEC coefficients and allowing the MEC coefficients to adapt will have no affect on them. This implies that after the last $n-m$ TEC coefficients have converged and phase II begins, they will not change significantly. Therefore, although they continue their adaptation, the value of the last $n-m$ coefficients is approximately

$$s''(k) \approx E\{s''(\infty)\} = p'' \quad (3.16)$$

Although we are approximating the value of this coefficient vector as a constant, we will not ignore the error component in

excess of the MMSE that is caused by their adaptation when deriving the expressions for the AMSE(k) in the following sections.

For the MEC, we substitute (3.8) into (3.7) and use the approximations in (3.15) and (3.16) to obtain

$$\begin{aligned} \mathbf{m}(k+1) = & (1 - \beta_1 \mathbf{a}(k) \mathbf{a}^T(k)) \mathbf{m}(k) \\ & + \beta_1 (d(k) \mathbf{a}(k) - \mathbf{a}(k) \mathbf{x}'^T(k) \mathbf{p}' - \mathbf{a}(k) \mathbf{x}''^T(k) \mathbf{p}'') \end{aligned} \quad (3.17)$$

After taking expectations and eliminating the crosscorrelations involving $\mathbf{x}''(k)$ as in Chapter 2, the result is

$$E\{\mathbf{m}(k+1)\} = (1 - 2^{-m} \beta_1) \mathbf{m}(k) + \beta_1 (\mathbf{q} - \mathbf{Y}^T \mathbf{p}') \quad (3.18)$$

The recursion is evaluated as usual. The result, after reexpressing $\mathbf{Y}^T \mathbf{p}'$ in terms of $\mathbf{A}$ and $\mathbf{q}$ using (2.29) and (2.33) and factoring, is

$$E\{\mathbf{m}(k)\} = (2^m \mathbf{I} - \mathbf{A}^T \mathbf{A}) \mathbf{q} (1 - (1 - 2^{-m} \beta_1)^k) \quad (3.19)$$

In the limit (3.19) becomes

$$\lim_{k \rightarrow \infty} E\{\mathbf{m}(k)\} = E\{\mathbf{m}(\infty)\} = (2^m \mathbf{I} - \mathbf{A}^T \mathbf{A}) \mathbf{q} \quad (3.20)$$

provided that,

$$0 < 2^{-m}\beta_1 < 2 \quad (3.21)$$

Note that (3.20) is the same as the expression for the asymptotic value of the mean MEC coefficients in the jointly-adaptive case given by (2.68) with the exception that a fraction of $A^T A$ is subtracted from $2^m I$ in the jointly-adaptive case. This supports the expected result that the asymptotic mean MEC coefficients in the jointly-adaptive case will be larger than those adapted using the sequential adaptation scheme.

Now it is just a matter of verifying whether (3.15), (3.16) and (3.20) satisfy the MMSE conditions for phase II. Clearly (3.16) satisfies the MMSE condition given by (2.44) thus

$$s^{**} = E\{s''(\infty)\} = p'' \quad (3.22)$$

If we use (2.29) and substitute (3.15) into (3.20) and manipulate the result slightly, we obtain

$$2^{-m}E\{m(\infty)\} - q + Y^T s'_f = 0 \quad (3.23)$$

But (3.23) has the form of the MMSE condition given by (2.45). If we also multiply both sides of (3.23) by Y and use (2.35), the result will have the form given by (2.43). Therefore, we may conclude that

$$\mathbf{s}^* = \mathbf{s}'_f = \mathbf{p}' \quad (3.24)$$

$$\mathbf{m}^* = E\{\mathbf{m}(\infty)\} = (2^m \mathbf{I} - \mathbf{A}^T \mathbf{A}) \mathbf{q} \quad (3.25)$$

3.2.1.4 EXPRESSIONS FOR MINIMUM MEAN SQUARE ERROR

During phase I, the error surface seen by the TEC/MEC is given by (3.9). Substituting the (3.11) into (3.9) we have

$$\text{MMSE}_1 = \text{MSE}(\mathbf{s}^*) = E\{d^2(k)\} - \mathbf{p}^T \mathbf{p} \quad (3.26)$$

where,

$$E\{d^2(k)\} \quad \text{is given by (2.96)}$$

$$\mathbf{p} \quad \text{is given by (3.10), (2.83) and (2.93)}$$

During phase II, the error surface seen by the Sequentially-Adaptive TEC/MEC is precisely the same as that seen by the Jointly-Adaptive TEC/MEC and is given by (2.36). The minimum of this surface is given by (2.95). Substituting (3.22) into (2.95) gives

$$\text{MMSE}_2 = E\{d^2(k)\} - 2^m \mathbf{q}^T \mathbf{q} - \mathbf{p}''^T \mathbf{p}'' \quad (3.27)$$

where,

$$E\{d^2(k)\} \quad \text{is given by (2.96)}$$

$$\mathbf{q} \quad \text{is given by (2.84)}$$

$$\mathbf{p}'' \quad \text{is given by (2.93)}$$

The above result is not surprisingly the same as the expression for the MMSE of the Jointly-Adaptive TEC/MEC.

3.2.2 MEAN SQUARE ERROR

The MSE during phase I is different from that of phase II because the TEC/MEC sees a different error surface. An expression for the MSE in this TEC-only case has already been derived in [Ste'85]. Using (3.4) and following a procedure similar to that of section 2.2.2 yields

$$\text{MSE}_1(k) = \text{MMSE}_1 + \mathbf{s}_e^T(k)\mathbf{s}_e(k) \quad (3.28)$$

where,

$$\mathbf{s}_e(k) = \mathbf{s}(k) - \mathbf{s}^* \quad (3.29)$$

In section 2.2.2, we derived an expression of the MSE based on all of the TEC/MEC coefficient vectors. During phase II, all of the coefficients are utilized, so that expression is applicable in this situation. The MSE expression in (2.103) can be simplified in our case. We have assumed that at the beginning of phase II, the TEC coefficients are more or less at their optimum values. The first m TEC coefficients were assumed to be fixed at their optimum values while the remaining coefficients were at their optimum values yet were allowed to continue to adapt. This implies that the coefficient error vector, $\mathbf{s}'_e(k)$ is negligible and thus the terms containing this vector can be ignored. Although, the coefficient error vector for the remaining TEC coefficients will be small, the fact that they are still being adapted will not be ignored. The adaptation of these

coefficients, like the MEC coefficients, will cause an error component in excess of the MMSE. The modified expression is

$$\text{MSE}_2(k) = \text{MMSE}_2 + \mathbf{s}_e^T(k) \mathbf{s}_e(k) + 2^{-m} \mathbf{m}_e^T(k) \mathbf{m}_e(k) \quad (3.30)$$

3.2.3 AVERAGE EXCESS MEAN SQUARE ERROR

The AMSE during phase I and II are the expectations of the respective MSE expressions. Taking the expectation of (3.28) and (3.30) yields

$$\text{AMSE}_1(k) = \text{MMSE}_1 + E\{\mathbf{s}_e^T(k) \mathbf{s}_e(k)\} \quad (3.31)$$

$$\begin{aligned} \text{AMSE}_2(k) = \text{MMSE}_2 + E\{\mathbf{s}_e^T(k) \mathbf{s}_e(k)\} \\ + 2^{-m} E\{\mathbf{m}_e^T(k) \mathbf{m}_e(k)\} \end{aligned} \quad (3.32)$$

The corresponding AMSE_{ex} quantities are those components of the AMSE other than the MMSE, namely,

$$\text{AMSE}_{\text{ex}1}(k) = E\{\mathbf{s}_e^T(k) \mathbf{s}_e(k)\} \quad (3.33)$$

$$\text{AMSE}_{\text{ex}2}(k) = E\{\mathbf{s}_e^T(k) \mathbf{s}_e(k)\} + 2^{-m} E\{\mathbf{m}_e^T(k) \mathbf{m}_e(k)\} \quad (3.34)$$

As in Chapter 2, the expectations represent the average power of the coefficient errors. These will be evaluated in the next section.

3.2.3.1 COEFFICIENT ERROR POWERS

For phase I, there is only 1 coefficient error power to consider.

$$v_n(k) = E\{s_e^T(k)s_e(k)\} \quad (3.35)$$

It is evaluated in a similar manner to those of the previous chapter. The result in nonrecursive form is

$$v_n(k) = (1 - 2\alpha_1 + n\alpha_1^2)^k v_n(0) + \text{MMSE}_1 n\alpha_1^2 \left[\sum_{i=0}^{k-1} (1 - 2\alpha_1 + n\alpha_1^2)^i \right] \quad (3.36)$$

This familiar form converges provided that

$$0 < n\alpha_1 < 2 \quad (3.37)$$

Assuming this to be the case, (3.37) can be further reduced to the final expression of the coefficient error power during Phase I.

$$v_n(k) = (1 - 2\alpha_1 + n\alpha_1^2)^k \left(v_n(0) - \text{MMSE}_1 \frac{n\alpha_1^2}{2\alpha_1 - n\alpha_1^2} \right) + \text{MMSE}_1 \left(\frac{n\alpha_1^2}{2\alpha_1 - n\alpha_1^2} \right) \quad (3.38)$$

The coefficient error powers for phase II have already been evaluated in section 2.2.3.1. Due to the exclusive adaptation of the MEC during phase II, there is no overlap of TEC and MEC coefficients

adaptation-wise. We need concern ourselves with only a subset of the coefficient error powers, specifically,

$$\begin{aligned}v''(k) &= E\{s''_e{}^T(k)s''_e(k)\} \\w(k) &= 2^{-m}E\{m_e{}^T(k)m_e(k)\}\end{aligned}$$

Taking the recursive coefficient error power expressions for the MEC and nonoverlapped TEC coefficients in (2.112) and (2.113) and following a similar procedure to that in Chapter 2, a nonrecursive vector form representing the coefficient error powers during phase II is found to be

$$\mathbf{z}_s(k) = (\mathbf{I} - \mathbf{T}_s)^k \mathbf{z}_s(0) + \text{MMSE}_2 \star \left[\sum_{i=0}^{k-1} (\mathbf{I} - \mathbf{T}_s)^i \right] \mathbf{b}_s \quad (3.39)$$

where,

$\mathbf{z}_s(k)$ "coefficient error power vector for phase II" of the sequential adaptation scheme

$$\mathbf{z}_s(k) \equiv \begin{bmatrix} v''(k) \\ w(k) \end{bmatrix} \quad (3.40)$$

$\mathbf{T}_s$ "coefficient error power update matrix for phase II"

$$\mathbf{T}_s = \begin{bmatrix} \alpha_1(2-(n-m)\alpha_1) & -(n-m)\alpha_1^2 \\ -2^{-m}\beta_1^2 & 2^{-m}\beta_1(2-\beta_1) \end{bmatrix} \quad (3.41)$$

$\mathbf{b}_s$ constant vector which partially determines extent of AMSE_{ex} for phase II

$$\mathbf{b}_s = \begin{bmatrix} (n-m)\alpha_1^2 \\ 2^{-m}\beta_1^2 \end{bmatrix} \quad (3.42)$$

The expression for $\text{AMSE}_{\text{ex}2}$ can be alternatively expressed in this vector notation as

$$\text{AMSE}_{\text{ex}2}(k) = \Sigma\{\mathbf{z}_s(k)\} \quad (3.43)$$

(3.39) is identical in form to (2.136) and thus its convergence is possible if the following condition exists

$$0 < \lambda_{Tsj} < 2 \quad j = 0, 1 \quad (3.44)$$

where,

λ_{Tsj} the j th eigenvalue of $\mathbf{T}_s$

The topic of convergence will be discussed in the last section of this chapter. Meanwhile, we may assume that the conditions for convergence exist. Consequently, as in Chapter 2, the coefficient error power vector can be expressed in terms of a transient and steady state component.

$$\mathbf{z}_s(k) = (\mathbf{I} - \mathbf{T}_s)^k (\mathbf{z}_s(0) - \text{MMSE}_2 \cdot \mathbf{T}_s^{-1} \mathbf{b}_s) + \text{MMSE}_2 \cdot \mathbf{T}_s^{-1} \mathbf{b}_s \quad (3.45)$$

As before, the first term is the transient component and the 2nd is the steady state component.

3.2.3.2 EXPRESSIONS FOR AVERAGE EXCESS MEAN SQUARE ERROR

The average excess MSE for phase I, by (3.33), is just the error power quantity given by (3.38) and hence we may immediately write

$$\begin{aligned} \text{AMSE}_{\text{ex1}}(k) = (1 - 2\alpha_1 + n\alpha_1^2)^k & \left(v_n(0) - \text{MMSE}_1 \frac{n\alpha_1^2}{2\alpha_1 - n\alpha_1^2} \right) \\ & + \text{MMSE}_1 \left(\frac{n\alpha_1^2}{2\alpha_1 - n\alpha_1^2} \right) \end{aligned} \quad (3.46)$$

(3.43) expresses the AMSE_{ex} during phase II as the sum of the associated coefficient error power vector. Therefore, summing the elements of (3.45) gives

$$\begin{aligned} \text{AMSE}_{\text{ex2}}(k) = \Sigma \{ (\mathbf{I} - \mathbf{T}_s)^k (\mathbf{z}_s(0) - \text{MMSE}_2 \cdot \mathbf{T}_s^{-1} \mathbf{b}_s) \} \\ + \text{MMSE}_2 \cdot \Sigma \{ \mathbf{T}_s^{-1} \mathbf{b}_s \} \end{aligned} \quad (3.47)$$

3.2.3 EXPRESSIONS FOR AVERAGE MEAN SQUARE ERROR

The AMSE of phase I and phase II are found by combining (3.31) and (3.32) with (3.46) and (3.47) respectively.

$$\begin{aligned} \text{AMSE}_1(k) = (1 - 2\alpha_1 + n\alpha_1^2)k & \left(v_n(0) - \text{MMSE}_1 \frac{n\alpha_1^2}{2\alpha_1 - n\alpha_1^2} \right) \\ & + \text{MMSE}_1 \left(1 + \frac{n\alpha_1^2}{2\alpha_1 - n\alpha_1^2} \right) \end{aligned} \quad (3.48)$$

where,

$$v_n(0) = \mathbf{s}^* \mathbf{T} \mathbf{s}^* = \mathbf{p}^T \mathbf{p} \quad (3.49)$$

$$\begin{aligned} \text{AMSE}_2(k) = \Sigma \{ (\mathbf{I} - \mathbf{T}_s)^k (\mathbf{z}_s(0) - \text{MMSE}_2^* \mathbf{T}_s^{-1} \mathbf{b}_s) \} \\ + \text{MMSE}_2 (1 + \Sigma \{ \mathbf{T}_s^{-1} \mathbf{b}_s \}) \end{aligned} \quad (3.50)$$

where,

$$\mathbf{z}_s(0) = \begin{bmatrix} v''(0) \\ w(0) \end{bmatrix} \quad (3.51)$$

where,

$v''(0)$ the initial value of the nonoverlapped n - m TEC coefficient error power at the beginning of phase II

$$\begin{aligned} v''(0) &= E\{\mathbf{s}''^T(0) \mathbf{s}''(0)\} \\ &\approx (\mathbf{s}''^* - \mathbf{s}''^*)^T (\mathbf{s}''^* - \mathbf{s}''^*) \\ &\approx 0 \end{aligned}$$

$w(0)$ the initial value of the nonoverlapped MEC coefficient error power at the beginning of phase II

$$w(0) = 2^{-m} \mathbf{m}^* \mathbf{T} \mathbf{m}^*$$

The optimum TEC coefficients for phase I are evaluated using (3.10). The optimum MEC coefficients are evaluated using (3.25),

(2.83), and (2.84). $v''(0)$ can be approximated as 0 as shown if we assume that the TEC coefficients approximately converged during phase I. In reality it will not be 0 but rather some small positive value due to the fluctuations in the nonoverlapped coefficient vector about the optimum point due to the noisy adaptation process. Nevertheless, since it is small and is part of the transient component we will let the approximation stand.

The AMSE during phase I and phase II both consist of transient terms which originate essentially from the adaptation of a single canceller. The phase I transient is caused by the adaptation of all n TEC coefficients. The phase II transient is essentially determined by the adaptation of the MEC coefficients because the transient component due to the previously converged nonoverlapped TEC coefficients is negligible. The steady state term for phase II consists of contributions from the adaptation of the MEC coefficients and the last $n-m$ coefficients of the TEC which are permitted to adapt for the remainder of phase II. There is no steady state component contribution from the first m TEC coefficients because they were fixed at the end of phase I. This is evident from the contents of the steady state matrix product $T_s^{-1}b_s$ which consists of terms multiplied by $(n-m)$ instead of m .

3.3 CONVERGENCE BEHAVIOUR

The Sequentially-Adaptive TEC/MEC is similar to a TEC/MEC configuration that adopts the mutually exclusive idea of adaptation

where typically the TEC is allowed to adapt first and freeze its coefficients after which the MEC is enabled for the remainder of the canceller's operation. The convergence characteristics of such a configuration are known because, at any time, the echo canceller is only 1 type of canceller. The convergence behaviour of the individual cancellers is known [Fal'87]. The only difference with regards to the Sequentially-Adaptive TEC/MEC is the fact that the TEC's nonoverlapped coefficients are permitted to adapt alongside the MEC's for the remainder of the canceller's operation after the initial convergence of the TEC. We have assumed that any fluctuations in the echo path over time occur very slowly and are very small. Moreover, we have discovered that the nonoverlapped TEC coefficients of the Jointly-Adaptive TEC/MEC adapt independently from the overlapped ones. The conclusion is that further study of the mean coefficient convergence here is not necessary because we can safely assume that they converge in much the same way as the case of mutually exclusive adaptation. There are, however, some comments that can be made with regards to their asymptotic values.

The evolution of the AMSE over time is a different issue. The first phase is already known because the echo canceller is a TEC acting alone. The expression given by (3.48) describing the AMSE behaviour for the TEC therefore needs no more attention. However, the AMSE behaviour during phase II is of interest here because its

expression, given by (3.50), has the same form as that of the Jointly-Adaptive case.

3.3.1 MEAN COEFFICIENT CONVERGENCE

The mean coefficient behaviour is approximately that of a TEC adapting separately followed by the MEC. The asymptotic values of the n TEC coefficients will essentially be the first n linear echo path coefficients. The MEC coefficients will approximately assume the values of the first 2^m nonlinear echo path coefficients with negligible acquisition of the linear component. This is beneficial to the MEC because the linear component is 40 dB above the component cancelled by the MEC. As we have seen in the case of the Jointly-Adaptive TEC/MEC, any slight acquisition of the linear component by the MEC coefficients will drastically increase their dynamic range requirements.

The implied dynamic range advantage must be clarified. The lower dynamic range requirement of the MEC coefficients in the sequentially-adaptive case only applies when we assume that the portion of the linear echo, corresponding to the span of the MEC, does not change significantly over time. In a practical situation, however, the linear echo component may vary as much as 40 dB due to circuit heating and other time-dependent effects in the echo path. As the corresponding linear TEC coefficients are frozen after the initial convergence, the MEC coefficients must adapt to the changes in the echo path. Therefore, in the case where the echo path is not constant

over time, the MEC coefficients of the sequentially-adaptive scheme may not have a significant dynamic range advantage.

3.3.2 AVERAGE MEAN SQUARE ERROR CONVERGENCE

In this section we will deal with issues concerning the AMSE convergence during phase II operation.

The theoretical expression describing the behaviour of the AMSE over time is given by (3.50). As in Chapter 2, it was arrived at by assuming that the conditions for its convergence were met. (3.44) imposes a restriction, similar to that given in Chapter 2, on the eigenvalues of the phase II coefficient error power adaptation matrix, T_s , for AMSE convergence. In the following sections the transient and steady state characteristics of the phase II AMSE will be investigated in detail. As in Chapter 2, the investigation into the transient characteristics will yield a "region of convergence" or ROC in which corresponding adaptation step-size values guarantee AMSE convergence. The adaptation step-size values that give the fastest convergence time will also be found. The steady state analysis will describe the relationship between the adaptation step-sizes and the steady state AMSE.

3.3.2.1 TRANSIENT AMSE CHARACTERISTICS

In this section we will follow a procedure similar to that in section 2.3.2.1 to investigate the transient AMSE characteristics of the phase II operation of the Sequentially-Adaptive TEC/MEC. The reasons for each step given there also apply here.

The expression for the coefficient error power vector for phase II, $\mathbf{z}_s(k)$, given by (3.39), indicates that convergence of this vector, and hence the AMSE, is contingent on the bounding of the eigenvalues of the corresponding coefficient error power update matrix, $\mathbf{T}_s$. These bounds are given by (3.44).

Fig. 3.1 and Fig. 3.2 give the contour plots of the smallest and largest eigenvalues of $\mathbf{T}_s$. All of the contours represent values within the allowable bounds. The contours are generated for the case of $n = 30$ and $m = 5$. Note that, as in Chapter 2, the contours were plotted against a "normalized" TEC adaptation step-size. As before, α_1 was normalized by the power of the input into the TEC. In this case, the TEC spans $n-m$ bauds and therefore the normalizing factor was $n-m$.

Fig. 3.1 A contour plot of the smallest eigenvalue of T_s . The outermost contour is of height 0. The innermost contour is of height 0.017. The spacing of the contours is 0.001. $n = 30, m = 5$.

Fig. 3.2 A contour plot of the largest eigenvalue of T_s . The bottom left hand contour is of height 0.005. The upper right hand corner contour is of height 0.1. The spacing of the contours is 0.005. $n = 30$, $m = 5$.

The conclusions that were made upon examination of the contour plots are similar to those obtained in Chapter 2.

- 1) The smallest eigenvalue of T_s , $\lambda_{s\epsilon}$, imposes a tighter bound on $(n-m)\alpha_1$ and β_1 than the largest eigenvalue of T_s and therefore we need not worry about the largest eigenvalue of T_s .

- 2) The ROC convergence has a triangular shape that is exactly half of the ROC for the case of mutually exclusive adaptation employing a TEC spanning $n-m$ bauds. The ROC boundary is defined by the following relationship.

$$0 < (n-m)\alpha_1 + \beta_1 < 2 \quad (3.52)$$

The bounds imposed on the adaptation step-sizes for coefficient convergence in the mean are given by (3.14) and (3.21). Comparing those bounds to the ones above, it becomes clear that the bounds governing the convergence of the AMSE are tighter than those for mean coefficient convergence. Therefore, (3.52) describes a ROC for convergence in the mean square and mean sense. For $(n-m)\alpha_1 + \beta_1$ values outside this region, the corresponding eigenvalues of T_s are 0 or negative. This is below the lower bound of 0 imposed on the eigenvalues by (3.44).

- 3) The eigenvalue function of the adaptation step-sizes is unimodal with a single maximum. The overall convergence time is determined by the smallest eigenvalue of T_s , $\lambda_{s\epsilon}$. Therefore, the adaptation step-sizes which produce the maximum $\lambda_{s\epsilon}$ are optimum in the sense of convergence time.
- 4) The exact location of the maximum $\lambda_{s\epsilon}$ warrants further attention.

Simulations of the Sequentially-Adaptive TEC/MEC were performed to test the ROC boundary given by (3.52). Adaptation step-sizes were chosen just inside and just outside of the ROC boundary. At each point, it was noted whether or not the jointly-adaptive TEC/MEC converged. This was done for several combinations of n and m . TABLE 3.1 summarizes the result of the ROC testing for the case of $n = 30$ and $m = 5$.

TABLE 3.1 Results of testing of the outer phase II ROC boundary defined by $(n-m)\alpha_1 + \beta_1 = 2$ for $n = 30$ and $m = 5$.
C $\equiv$ Converged, NC $\equiv$ Did not converge.

$(n-m)\alpha_1$	β_1	$(n-m)\alpha_1 + \beta_1$	C/NC
0.25	1.73	1.98	C
0.25	1.77	2.02	NC
0.75	1.23	1.98	C
0.75	1.27	2.02	NC
1.25	0.73	1.98	C
1.25	0.77	2.02	NC
1.75	0.10	1.85	C
1.75	0.27	2.02	NC

Fig. 3.3 - Fig. 3.5 gives simulation results that illustrate the sensitivity of AMSE error convergence to the outer ROC boundary during both phase I and II. Fig. 3.3 illustrates the case where the phase I AMSE converges but the phase II AMSE does not. Fig. 3.4

illustrates the interesting case where the phase I AMSE does not converge but the phase II AMSE does. In this case, convergence takes longer than usual because phase II has to make up for the initial divergence during phase I. Fig. 3.5 illustrates the case where the AMSE during both phases doesn't converge.

Recall that phase I AMSE convergence is governed by (3.37).

Fig. 3.3 Phase II AMSE convergence sensitivity to the choice of adaptation step-sizes near the ROC boundary. $n = 30$, $m = 5$. The divergent curve is the result of choosing $(n-m)\alpha_1 = 1.25$ and $\beta_1 = 0.77$ hence $(n-m)\alpha_1 + \beta_1 = 2.02$. The converging curve is the result of choosing $(n-m)\alpha_1 = 1.25$ and $\beta = 0.73$ hence $(n-m)\alpha_1 + \beta_1 = 1.98$. Phase I converges in this case because $n\alpha_1 = 1.5$.

Fig. 3.4 Phase I AMSE divergence with phase II AMSE convergence. $(n-m)\alpha_1 = 1.75$, $\beta_1 = 0.1$ therefore $(n-m)\alpha_1 + \beta_1 = 1.85$. $n = 30, m = 5, n\alpha_1 = 2.10$.

Fig. 3.5 Phase I and phase II AMSE divergence. $(n-m)\alpha_1 = 1.75$, $\beta_1 = 0.27$ therefore $(n-m)\alpha_1 + \beta_1 = 2.02$. $n = 30, m = 5$, $n\alpha = 2.10$.

We now investigate the location of the maximum λ_{SE} which is responsible for giving the fastest convergence during phase II. A

computer "hill-climbing" technique was used to search the unimodal ROC "surface" to find the maximum λ_{se} for different n and m and hence iteratively determining the adaptation step-sizes for optimum convergence time. The results are shown in Table 3.2.

TABLE 3.2 Computed adaptation step-sizes for optimum Phase II convergence time and maximum λ_{se} for $n = 16, 20, 25, 30$ and $m = 3, 4, 5, 6$. Search step-size for $(n-m)\alpha_1$ and $\beta_1 = 0.01$.

n	m	$(n-m)\alpha_{1opt}$	β_{1opt}	$\lambda_{se} (* 10^{-2})$
16	3	0.62	0.38	4.76
16	4	0.43	0.57	3.57
16	5	0.26	0.74	2.33
16	6	0.14	0.86	1.35
20	3	0.68	0.32	4.00
20	4	0.50	0.50	3.13
20	5	0.32	0.68	2.13
20	6	0.18	0.82	1.28
25	3	0.73	0.27	3.33
25	4	0.57	0.43	2.70
25	5	0.38	0.62	1.92
25	6	0.23	0.77	1.20
30	3	0.77	0.23	2.86
30	4	0.62	0.38	2.38
30	5	0.44	0.56	1.75
30	6	0.27	0.73	1.14

There are several significant conclusions that can be made from the results given in TABLE 3.2.

1) Like λ_{ϵ} in Chapter 2, λ_{se} decreases with increasing n and m .

A smaller λ_{se} implies a longer overall convergence time.

2) The normalized optimum TEC adaptation step-size and the MEC adaptation step-size have the following relationship.

$$(n-m)\alpha_{1opt} + \beta_{1opt} \equiv 1 \quad (3.53)$$

This is similar to the optimum relationship given for the Jointly-Adaptive TEC/MEC given by (2.156).

3) The fraction of unity that the normalized optimum TEC adaptation step-size and the MEC adaptation step-size receive appears to be determined by the following relationship.

$$(n-m)\alpha_{1opt} = \frac{n-m}{n-m+2^m} \quad (3.54)$$

or

$$\alpha_{1opt} = \frac{1}{n-m+2^m} \quad (3.55)$$

and

$$\beta_{1opt} = \frac{2^m}{n-m+2^m} \quad (3.56)$$

(3.55) and (3.56) give expressions for the adaptation step-sizes for optimum convergence time that are similar to those found in Chapter 2. The following relationship between α_{1opt} and β_{1opt} is identical to that between the adaptation step-sizes for the Jointly-Adaptive TEC/MEC.

$$\beta_{1opt} = 2^m \alpha_{1opt} \quad (3.57)$$

The expressions for the adaptation step-sizes for optimum convergence time were tested by simulation for different sets of n and m . Besides the optimum adaptation step-sizes, 4 other sets of adaptation step-sizes were chosen. The points that were specified were equally spaced around the optimum point within the ROC. Fig. 3.6 gives a typical set of AMSE curves for the case of $n = 30$ and $m = 5$.

Fig. 3.6 Verifying the optimum step-size for fastest phase II AMSE convergence for $n = 30$ and $m = 5$. The fastest converging curve corresponds to $\alpha_1 = \alpha_{1opt} = 1.75 \text{ e-}2$, $\beta_1 = \beta_{1opt} = 5.61 \text{ e-}2$. The others correspond to
a) $\alpha_1 = \alpha_{1opt} + 0.4/n$, $\beta_1 = \beta_{1opt}$, b) $\alpha_1 = \alpha_{1opt}$,
 $\beta_1 = \beta_{1opt} + 0.4$, c) $\alpha_1 = \alpha_{1opt} - 0.4/n$, $\beta_1 = \beta_{1opt}$,
d) $\alpha_1 = \alpha_{1opt}$, $\beta_1 = \beta_{1opt} - 0.4$.

Finally, a comparison between the optimum TEC adaptation step-size for phase II, given by (3.55), and that of the TEC alone given by (2.154) revealed that the TEC should converge faster if there were different TEC adaptation step-sizes used during each phase. (3.55) indicates that α_{1opt} varies inversely with $n-m+2^m$ whereas (2.154) indicates that α_{opt} only varies inversely with n . Therefore, during phase I when the echo canceller is in TEC-only mode, it makes sense to use the larger optimum value given by (2.154). During phase II, the value given by (3.55) must be used. Fig. 3.7 illustrates the overall decrease in convergence time as a result of using the different optimum TEC adaptation step-sizes for each phase.

Fig. 3.7 The effect of the use of single vs dual TEC adaptation step-sizes on the overall convergence time of the Sequentially-Adaptive TEC/MEC. $n = 30$, $m = 5$. The faster converging curve corresponds to $\alpha_1 = \alpha_{1opt1} = 1/n$ for phase I and $\alpha_1 = \alpha_{1opt2} = 1/(n-m+2^m)$ for phase II. The slower converging curve corresponds to the use of the same TEC adaptation step-size during both phases. $\alpha_1 = \alpha_{1opt2}$.

Therefore, it can be seen that using TEC adaptation step-sizes that are optimum specifically for each phase gives better convergence speed. Also, the expression for the optimum TEC adaptation step-size for phase II suggests that the difference in convergence speed becomes more noticeable as the MEC span is increased.

3.3.2.2 STEADY STATE AMSE CHARACTERISTICS

In this section we will investigate the relationship between the adaptation step-sizes and the steady state AMSE level as we did in Chapter 2.

The steady state AMSE is given by the 2nd term of the phase II AMSE expression given by (3.50). We concluded, from (3.27), that the MMSE for the Jointly-Adaptive TEC/MEC was the same as $MMSE_2$. Therefore we have

$$AMSE_{ss} = MMSE(1 + \Sigma\{T_s^{-1}b_s\}) \quad (3.58)$$

As in the jointly-adaptive case, the summation term, $\Sigma\{T_s^{-1}b_s\}$, is the factor that determines the extent to which the $AMSE_{ss}$ exceeds the MMSE. This term was computed over the adaptation step-size values that lay in the ROC. A contour plot with the same axes as the ROC for phase II was generated for $\Sigma\{T_s^{-1}b_s\}$. The plot consisted of parallel contours that had the form

$$(n-m)\alpha_1 + \beta_1 = c \quad (3.59)$$

where,

$$c > 0$$

As in Chapter 2, the result implies that different sets of adaptation step-sizes chosen along the same contour will result in the same $AMSE_{ss}$. Furthermore, the contours exhibit the same characteristics as before. Specifically, the contours are parallel to the outermost contour of the ROC and the value of the contours depend on c in the following way.

$$\begin{aligned} \Sigma\{\mathbf{T}_s^{-1}\mathbf{b}_s\} &= 1, & c &= 1 \\ &< 1, & c &< 1 \\ &> 1, & c &> 1 \end{aligned} \quad (3.60)$$

Note, again, from (3.58) and (3.60), that the case where $c=1$ corresponds to an AMSE level that is 3 dB above the MMSE. Actually, $\Sigma\{\mathbf{T}_s^{-1}\mathbf{b}_s\}$ and $\Sigma\{\mathbf{T}^{-1}\mathbf{b}\}$ were found to be equivalent in the following way.

$$\Sigma\{\mathbf{T}_s^{-1}\mathbf{b}_s\} = \Sigma\{\mathbf{T}^{-1}\mathbf{b}\} \quad (3.61)$$

where,

$$(n-m)\alpha_1 + \beta_1 = n\alpha + \beta$$

(3.61) implies that, given a Jointly-Adaptive TEC/MEC and Sequentially-Adaptive TEC/MEC with the same dimensions and adaptation step-sizes, the factor, $\Sigma\{\mathbf{T}_s^{-1}\mathbf{b}_s\}$, and consequently the AMSE_{ss} for the sequentially-adaptive case will be lower. However, it is more typical to use a different set of adaptation step-sizes in the jointly-adaptive case. Consequently, it is possible to achieve a lower AMSE_{ss} by adjusting α and β to give a lower value of c . In fact, if we reexamine the relationships between the normalized TEC adaptation step-size and the MEC adaptation step-size for fastest AMSE convergence given by (2.156) and (3.53), we find that these correspond to the case where $c=1$ and hence where the AMSE_{ss} is 3 dB above the MMSE. Therefore, we can conclude that the jointly-adaptive and sequentially-adaptive schemes will give the same AMSE_{ss} when their adaptation step-sizes are chosen for optimal AMSE convergence speed.

The above relationships were verified by simulation for different n and m . The AMSE_{ss} was computed using (3.58) and also determined by simulation for a selection of $(n-m)\alpha_1$ and β_1 along the contours defined by (3.59). The MMSE was computed using (2.97). TABLE 3.3 displays the results for the case of $n = 30$ and $m = 5$.

TABLE 3.3 Theoretical and simulated phase II AMSE_{ss} values for $c = 0.5, 1, 1.5$. Computed MMSE = -70 dB.

c	$(n-m)\alpha_1$	β	$\Sigma\{\mathbf{T}_s^{-1}\mathbf{b}\}$	Computed AMSE_{ss}	Actual AMSE_{ss}
				(dB)	(dB)
0.50	0.10	0.40	0.33	-68.75	-68.5
"	0.25	0.25	"	"	-68.5
"	0.40	0.10	"	"	-68.9
1.00	0.10	0.90	1.00	-66.99	-67.1
"	0.50	0.50	"	"	-67.5
"	0.90	0.10	"	"	-67.3
1.50	0.10	1.40	3.00	-63.98	-64.3
"	0.75	0.75	"	"	-64.6
"	1.40	0.10	"	"	-64.5

Note that the simulation results given by TABLE 3.3 strongly support the notions given by (3.58)-(3.60). The validity of (3.61) is demonstrated by the fact that the theoretical quantities are precisely the same as those of the jointly-adaptive case in TABLE 2.3.

3.4 SUMMARY

In this chapter, the behaviour of the sequentially-adaptive TEC/MEC was investigated from both theoretical and simulation-oriented viewpoints.

The theoretical study provided expressions, similar to those derived in Chapter 2, describing the minimum mean square error (MMSE) and the associated TEC/MEC coefficient relationships, the mean behaviour of the TEC/MEC coefficients as they were adapted using the LMS algorithm, and the resulting average mean square error (AMSE).

There were 2 distinct phases of operation identified. In phase I, the echo canceller operates in a TEC-only mode. In this mode, all n TEC coefficients are adapted in the absence of any MEC intervention. In phase II, the first m TEC coefficients are frozen and the m -baud MEC is activated and allowed to adapt with the remaining $n-m$ "nonoverlapped" coefficients. The "error surface" and MMSE for phase II were found to be identical to those of the jointly-adaptive case.

The mean coefficient trajectory analysis revealed that the mean of the coefficients, (3.12), (3.19), converges to a MMSE solution, (3.22), (3.24), (3.25), provided that the adaptation step-sizes are chosen to be within certain bounds (3.14), (3.21). Assuming the echo path parameters to be constant over time, the behaviour of the coefficients was found to be approximately that of the mutually exclusive adaptation case where all n TEC coefficients are frozen after phase I allowing the MEC to adapt for the remainder of the echo canceller operation during phase II. However, in a practical situation, the previous assumption was not realistic because it is typical for the echo signal to vary as much as 40 dB due to circuit

heating and other time-dependent effects. This implied that the MEC would be forced to adjust to any variations in the nonlinear and, in particular, the larger linear echo component over which the MEC spanned.

The AMSE analysis produced expressions of the phase I and phase II AMSE behaviour (3.48), (3.50). The phase I AMSE behaviour was that of a TEC alone. The phase II AMSE behaviour was based on a set of coefficient error power quantities and an associated coefficient error power update matrix, T_s , as in Chapter 2. T_s was in fact derived by a truncation of the update matrix, T , for the jointly-adaptive case. The convergence behaviour of the phase II AMSE was found to be governed by the eigenvalues of T_s (3.44). The excess AMSE was found to be dependent on the MMSE itself and some factor containing T_s^{-1} (3.47).

The simulation studies revealed several significant characteristics of the phase II AMSE similar to those of the AMSE of joint-adaptation. The smallest eigenvalue of T_s determined whether or not the echo canceller converged and also the rate of convergence. A contour plot of the smallest eigenvalue showed a "region of convergence", (3.52), that was precisely half of that of a mutually exclusive scheme in which the TEC spans $n-m$ bauds. The surface function determining the values of the eigenvalues within the region of convergence was found to be positive and unimodal like that of the jointly-adaptive case. Consequently, the set of TEC and MEC adaptation step-sizes corresponding to the maximum point on this

surface gave the fastest phase II AMSE convergence. Expressions for the optimum adaptation step-sizes were found after examining their values for many n and m (3.55), (3.56). They were similar to those obtained for the Jointly-Adaptive TEC/MEC. The theoretically derived optimum adaptation step-sizes were verified by simulation.

A contour plot of the factor determining the excess AMSE component was examined. It was revealed that this factor, and consequently the steady state AMSE, was equivalent to that of the jointly-adaptive case given the same normalized adaptation step-sizes (3.61). Furthermore, in the case where the adaptation step-sizes were chosen for optimum AMSE convergence time, the resulting steady state AMSE of the sequentially-adaptive case was shown to be identical to that of the jointly-adaptive case. In both cases, the steady state AMSE was 3 dB higher than the MMSE as a result of using the optimum adaptation step-sizes.

CHAPTER 4

A COMPARISON OF THE JOINT AND SEQUENTIAL COEFFICIENT ADAPTATION APPROACHES

4.1 COEFFICIENT CONVERGENCE

The mean coefficient trajectories for the Jointly-Adaptive TEC/MEC are given by (2.62), (2.63), (2.67), (2.68), (2.86), (2.88). Those for the Sequentially-Adaptive TEC/MEC are given by (3.12), (3.13), (3.19), and (3.20). The simulations of the Jointly-Adaptive TEC/MEC confirmed that the MEC was cancelling a portion of the linear echo component due to the first m bauds over which the MEC operated. The dynamic range requirements of the MEC coefficients are significantly greater compared to that of a mutually exclusive adaptation scheme because the linear echo component is typically 40 dB above nonlinear component. The possibility of the MEC taking on a share of the linear echo cancellation was also discussed for the sequentially-adaptive scheme in light of time-dependent variations in the echo path parameters that can typically occur over time.

4.2 AVERAGE MEAN SQUARE ERROR CONVERGENCE

The evolution of the AMSE over time of the Jointly-Adaptive TEC/MEC is described by (2.143). The phase I AMSE of the Sequentially-Adaptive TEC/MEC is that of a TEC only and is given by (3.48). The phase II AMSE expression is similar to that of the Jointly-Adaptive TEC/MEC and is given by (3.50).

The region of convergence for the jointly-adaptive case is given by (2.153). The convergence of the phase I AMSE for the sequentially-adaptive case is that of a TEC only and is given by (3.37). The convergence of the phase II AMSE is governed by a region of convergence similar to that of the jointly-adaptive case and is given by (3.52). Therefore, all things equal, the ROC for the Sequentially-Adaptive TEC/MEC is larger.

The TEC/MEC coefficients that give the fastest AMSE convergence for the jointly-adaptive case are given by (2.158) and (2.159). During phase I of the sequentially-adaptive case, the optimum TEC coefficients are given by (2.154). The phase II optimum coefficients are given by (3.55) and (3.56).

Intuitively, we expect that the AMSE behaviour for the jointly-adaptive case is primarily determined by the convergence of the TEC during the initial adaptation period because the linear echo path component is much larger than the nonlinear component. However, the TEC adaptation step-size is not the optimum one for a TEC acting alone as in phase I. Therefore, we expect the phase I AMSE to be faster during this initial period. This statement is supported by

theory as follows. TABLE 2.2 gives the values of the smallest eigenvalues, λ_e , of the coefficient error power update matrix, T . These eigenvalues determine the overall rate of convergence of the AMSE for the jointly-adaptive case. The larger eigenvalues give faster convergence. Similarly the rate of convergence during phase I is determined by the factor given by (2.154). This factor is consistently larger than the eigenvalues given in TABLE 2.2 supporting the intuitive conclusion that the phase I AMSE converges at a faster rate.

During phase II, we again intuitively suspect that the AMSE will converge at a faster rate than that of the jointly-adaptive case. (2.154) indicates that, by itself, the rate of convergence of the TEC is inversely proportional to its span. In other words, a longer TEC span requires a longer time to converge. Therefore, since the span of the TEC during phase II is only $n-m$, we expect the AMSE to converge at a faster rate than that of the jointly-adaptive case where the TEC spans the full n bauds. This is again supported by theory. An examination of the smallest eigenvalues of the T and T_s matrix reveals that those of the T_s matrix are consistently larger given the same n and m . The larger eigenvalues imply a faster rate of convergence for the phase II AMSE.

Fig. 4.1 gives a comparison between the fastest converging curves for the jointly-adaptive and sequentially-adaptive cases.

Fig. 4.1 A comparison of the fastest converging Jointly-Adaptive and Sequentially-Adaptive TEC/MEC. $n = 30, m = 5$. For the slower converging Jointly-Adaptive AMSE, $\alpha = \alpha_{opt} = 1/(n+2^m)$ and $\beta = \beta_{opt} = 2^m/(n+2^m)$. The faster converging Sequentially-Adaptive AMSE used phase I $\alpha_1 = \alpha_{opt1} = 1/n$, phase II $\alpha_1 = \alpha_{opt2} = 1/(n-m+2^m)$, and $\beta_1 = \beta_{opt} = 2^m/(n-m+2^m)$. Phase II begins after 220 bauds.

As expected, Fig. 4.1 verifies that the Sequentially-Adaptive TEC/MEC offers faster overall AMSE convergence to -65 dB. Note, however, that the difference between the convergence times diminishes as the steady state AMSE is approached.

4.3 STEADY STATE AVERAGE MEAN SQUARE ERROR

The expression for the $AMSE_{ss}$ for the jointly-adaptive case is given by (2.161). An almost identical expression for the sequentially-adaptive case is given by (3.58). In Chapter 3, it was discovered that values of $\Sigma\{T_s^{-1}b_s\}$ and $\Sigma\{T^{-1}b\}$ were identical given that

$$(n-m)\alpha_1 + \beta_1 = n\alpha + \beta$$

It was also known that $(n-m)\alpha_1 + \beta_1 = c_1$ represented a contour of equal values of $\Sigma\{T_s^{-1}b_s\}$ and $n\alpha + \beta = c$ represented a contour of equal values of $\Sigma\{T^{-1}b\}$. Larger values of c and c_1 yielded larger values of the summations and hence larger $AMSE_{ss}$ levels. Given the above observations, we can conclude that a Sequentially-Adaptive TEC/MEC will have a lower $AMSE_{ss}$ level than a Jointly-Adaptive TEC/MEC with the same spans and adaptation step-sizes. An examination of Table 2.3 and 3.3 supports the above relationships. However, the adaptation step-sizes for the jointly-adaptive case are clearly not restricted to being the same as those of the sequentially-adaptive case and therefore can be adjusted, according to the above expression, to achieve the same $AMSE_{ss}$. In fact, it was shown in

section 3.3.2.2 that the jointly-adaptive and sequentially-adaptive $AMSE_{ss}$ would be identical when their adaptation step-sizes were chosen for fastest AMSE convergence.

4.4 SUMMARY

In this chapter we have compared the Jointly-Adaptive TEC/MEC and the Sequentially-Adaptive TEC/MEC on the basis of the final coefficient values, the rate of convergence of the AMSE and the steady state AMSE. The Sequentially-Adaptive TEC/MEC was shown to be superior with respect to the first criteria.

In both cases, the possibility of the MEC taking a share of the linear echo cancellation load was found. As the linear echo component is typically much greater than the nonlinear component, the size requirement of the MEC coefficients in both cases is directly affected by the degree of load sharing taking place.

The AMSE during both phases of the Sequentially-Adaptive TEC/MEC operation converged at a faster optimum rate than a Jointly-Adaptive TEC/MEC with the same dimensions.

It was shown that the steady state AMSE level for the Jointly-Adaptive TEC/MEC could be made equivalent to that of a Sequentially-Adaptive TEC/MEC of the same dimensions by proper choice of the adaptation step-sizes. This equivalence could be achieved when the adaptation step-sizes for fastest AMSE convergence were used.

CHAPTER 5

CONCLUSIONS

In this thesis, we have investigated and compared the convergence characteristics of the Jointly-Adaptive TEC/MEC and Sequentially-Adaptive TEC/MEC. Both were evaluated on the basis of their coefficient behaviour, average mean square error (AMSE), and factors affecting their convergence.

The dual nature of the Jointly-Adaptive TEC/MEC implied the possibility of unstable coefficient convergence behaviour due to the infinite number of "solutions" for the minimum mean square error (MMSE). A mean coefficient trajectory analysis revealed that the coefficients could in fact reach a single and stable solution at least in the mean sense (2.71), (2.72), (2.89). Restrictions on the size of the adaptation step-sizes were established which permitted this first order type of convergence (2.60). A study of the asymptotic values of the coefficients revealed that the MEC coefficients take on a share of the linear echo cancellation load. The MEC cancels that portion of the linear echo component which is due to the input equal in span to that of the MEC. The requirements of dynamic range on the MEC

coefficients increase significantly because the linear echo component is on the order of 40 dB greater than the nonlinear component which the MEC was originally designed to handle.

A "region of convergence" (ROC), (2.153), for the AMSE of the Jointly-Adaptive TEC/MEC was found. This effectively limited the choice of adaptation step-sizes by precisely 1/2 when compared to a mutually exclusive adaptation scheme where the TEC and MEC are independent of each other (2.151), (2.152). The factor affecting the rate of convergence was identified and found to be a unimodal and positive function with the confines of the ROC. This factor was related to the TEC and MEC spans and their adaptation step-sizes. Consequently, the adaptation step-sizes that yielded the fastest converging AMSE were found (2.158), (2.159). A study of the steady-state AMSE revealed the expected connection to the adaptation step-sizes. That is, larger adaptation step-sizes give larger steady-state AMSE levels (2.162), (2.163). Finally, the accuracy of a theoretical expression of the evolution of the AMSE over time, (2.143), was verified by simulation.

The conclusion of the study of the mean coefficients of the Sequentially-Adaptive TEC/MEC was that their asymptotic values were essentially those that would be obtained by a mutual exclusive adaptation scheme. The TEC and MEC coefficients would approximately assume the values of their respective echo path coefficients. In this case, it appeared that the MEC's coefficients could avoid the dynamic range problem encountered in the jointly-

adaptive scheme because they did not share the load of cancelling the significantly greater linear echo component. However, this was found to be not necessarily true after considering the fact that the MEC's coefficients may inevitably be forced to adapt to time-varying changes in the linear echo path component over which they spanned.

There were 2 distinct phases of operation for the sequentially-adaptive TEC/MEC. Phase I was a TEC-only mode in which all TEC coefficients were permitted to adapt without any MEC intervention until they converged. Phase II was similar to the jointly-adaptive case with the exception that TEC coefficients corresponding to the input span common to the MEC were frozen after converging during phase I. The phase I AMSE was that of a TEC acting alone which was already well understood (3.48). The phase II AMSE was found to be similar in behaviour to that of the jointly-adaptive case (3.50). A similar ROC was defined for the adaptation step-sizes for phase II (3.52). The factor affecting the rate of convergence was also identified and found to be unimodal and positive within the ROC as in jointly-adaptive case. The adaptation step-sizes for optimum convergence time were found (3.55), (3.56). A study of the steady state AMSE revealed that the factor affecting its level was related to that of the jointly-adaptive case (3.59)-(3.61).

Finally, a comparison of the Jointly-Adaptive TEC/MEC and the Sequentially-Adaptive TEC/MEC showed that the latter had faster AMSE convergence but essentially equivalent steady state AMSE when using adaptation step-sizes for fastest convergence. In both

configurations, the MEC coefficient size requirements could be relatively large due to the sharing of the relatively large linear echo component.

APPENDIX A

AVERAGE MEAN SQUARE ERROR DETAILS

A.1 THE JOINT INPUT AUTOCORRELATION MATRIX

The joint input autocorrelation matrix, $\mathbf{R}_m$, is defined as $E\{\mathbf{x}_a(k)\mathbf{x}_a^T(k)\}$ where $\mathbf{x}_a(k)$ is the joint input vector formed from the concatenation of the overlapped input vectors $\mathbf{x}'(k)$ and $\mathbf{a}(k)$. The structure of $\mathbf{R}_m$ from (2.41) is

$$\mathbf{R}_m \equiv E\{\mathbf{x}_a(k)\mathbf{x}_a^T(k)\} = \begin{bmatrix} \mathbf{I}_m & \mathbf{Y} \\ \mathbf{Y}^T & 2^{-m}\mathbf{I}_\phi \end{bmatrix} \quad (\text{A1.1})$$

The singular or nonsingular nature of $\mathbf{R}_m$ is of interest as it will dictate the nature of the solution for the MMSE in (2.46). If $\det(\mathbf{R}_m)$ is 0, then $\mathbf{R}_m$ is singular implying that there are many $\mathbf{x}_a$'s that would satisfy the MMSE requirements. The determinant of $\mathbf{R}_m$ is evaluated by considering the eigenvalues of $\mathbf{R}_m$. The eigenvalues for many different $\mathbf{R}_m$ were calculated by computer. All $2^m + m$ eigenvalues were identified:

$$\lambda_0 = 1 + 2^{-m} \quad \text{with a multiplicity of } m \quad (\text{A1.2})$$

$$\lambda_1 = 2^{-m} \quad \text{with a multiplicity of } 2^m - m \quad (\text{A1.3})$$

$$\lambda_2 = 0 \quad \text{with a multiplicity of } m \quad (\text{A1.4})$$

If we were to solve for the eigenvalues analytically, we would have to solve the following characteristic equation.

$$\det(\lambda \mathbf{I} - \mathbf{R}_m) = 0 \quad (\text{A1.5})$$

where,

λ is an eigenvalue of $\mathbf{R}_m$

However (A1.4) indicates that m of the eigenvalues are 0. This implies that $\lambda = 0$ must be a solution to (A1.5) and therefore we have

$$\det(\mathbf{R}_m) = 0 \quad (\text{A1.6})$$

Thus we can conclude that $\mathbf{R}_m$ is singular.

Now we need only confirm that our experimentally derived eigenvalues are always true. Expanding (A1.5) we obtain

$$\det \left(\begin{bmatrix} (\lambda - 1)\mathbf{I}_m & -\mathbf{Y} \\ -\mathbf{Y}^T & (\lambda - 2^{-m})\mathbf{I}_\phi \end{bmatrix} \right) = 0 \quad (\text{A1.7})$$

The matrix of (A1.7) can be expressed as a product of an upper and lower triangular matrix so that the determinant can be readily

evaluated as the product of the diagonal elements. Using (2.35), we make the following decomposition.

$$\begin{aligned} & \begin{bmatrix} (\lambda-1)\mathbf{I}_m & -\mathbf{Y} \\ -\mathbf{Y}^T & (\lambda-2^{-m})\mathbf{I}_\phi \end{bmatrix} \\ &= \begin{bmatrix} \left(\frac{\lambda - (1 + 2^{-m})}{\lambda - 2^{-m}}\right)\mathbf{I}_m & \left(\frac{-1}{\lambda - 2^{-m}}\right)\mathbf{Y} \\ \mathbf{0}_{(\phi,m)} & \mathbf{I}_\phi \end{bmatrix} \begin{bmatrix} \lambda\mathbf{I}_m & \mathbf{0}_{(m,\phi)} \\ -\mathbf{Y}^T & (\lambda - 2^{-m})\mathbf{I}_\phi \end{bmatrix} \quad (\text{A1.8}) \end{aligned}$$

The determinant of (A1.8) is the product of the diagonal elements. The resulting characteristic equation is

$$\left(\frac{\lambda - (1 + 2^{-m})}{\lambda - 2^{-m}}\right)^m (\lambda - 2^{-m}) 2^m \lambda^m = 0 \quad (\text{A1.9})$$

Rearranging (A1.9) we obtain

$$(\lambda - (1 + 2^{-m}))^m (\lambda - 2^{-m}) (2^m - m) \lambda^m = 0 \quad (\text{A1.10})$$

From (A1.10) it is evident that the eigenvalues of $\mathbf{R}_m$ are indeed those listed in (A1.2)-(A1.4).

A.2 THE ECHO PATH LINEAR/NONLINEAR INPUT CROSSCORRELATION MATRIX

The echo path linear/nonlinear input crosscorrelation matrix represents the correlation between the input symbols used in the generation of the linear echo component and the address vector used

in the generation of the nonlinear echo component. The derivation of Y_h is similar to that of the overlapped TEC/MEC input crosscorrelation matrix, Y . The difference here is that the lengths of $x_h(k)$ and $a_h(k)$ are u and 2^v respectively where u does not necessarily equal 2^v . The consequence is that Y_h will contain some zero rows. The evaluation proceeds as follows.

$$\begin{aligned}
 Y_h &\equiv E\{x_h(k)a_h^T(k)\} \\
 &= \left(\sum_{i=0}^{2^{u-v}-1} [x_{0i} \ 0_u \ 0_u \ 0_u \ \dots \ 0_u] + \right. \\
 &\quad \sum_{i=0}^{2^{u-v}-1} [0_u \ x_{1i} \ 0_u \ 0_u \ \dots \ 0_u] + \\
 &\quad \sum_{i=0}^{2^{u-v}-1} [0_u \ 0_u \ x_{2i} \ 0_u \ \dots \ 0_u] + \\
 &\quad \left. \dots + \sum_{i=0}^{2^{u-v}-1} [0_u \ 0_u \ 0_u \ 0_u \ \dots \ x_{\psi-1i}] \right) / 2^u \quad (A2.1)
 \end{aligned}$$

where,

Each matrix has 2^v columns

x_{ji} 's have length u

x_{0i} = the i th symbol vector of length u where the first v bauds are $[-1 \ -1 \ -1 \ \dots \ -1]^T$

x_{1i} = the i th symbol vector of length u where the first v bauds are $[+1 \ -1 \ -1 \ \dots \ -1]^T$

x_{2i} = the i th symbol vector of length u where the first v bauds are $[-1 \ +1 \ -1 \ \dots \ -1]^T$

$\mathbf{x}_{3i}$ = the i th symbol vector of length u where the first v bauds are $[+1 +1 -1 \dots -1]^T$

...

$\mathbf{x}_{\psi-1i}$ = the i th symbol vector of length u where the first v bauds are $[+1 +1 +1 \dots +1]^T$

$\mathbf{0}_u$ $(u,1)$ column vector of zeroes

The summations are necessary because, in this case, $\mathbf{x}_h(k)$ consists of the first u symbols and $\mathbf{a}_h(k)$ is dependent on only the first v where we have assumed previously that $u \geq v$. The summations account for the fact that for each $\mathbf{a}_h(k)$, hence for each column position, there are 2^{u-v} possible $\mathbf{x}_h(k)$'s that are associated with that $\mathbf{a}_h(k)$. Since we are dealing with ± 1 binary symbols, for each j a summation over i of $\mathbf{x}_{ji}$ will simply produce a truncated version $\mathbf{x}_{ji}$ whose last $u-v$ symbols are 0. (A2.1) simplifies to

$$\begin{aligned} \mathbf{Y}_h = & \left(\begin{aligned} & [2^{u-v} \mathbf{x}_0' \ 0 \ 0 \ 0 \ \dots \ 0] \\ & + [0 \ 2^{u-v} \mathbf{x}_1' \ 0 \ 0 \ \dots \ 0] \\ & + [0 \ 0 \ 2^{u-v} \mathbf{x}_2' \ 0 \ \dots \ 0] + \dots \\ & + [0 \ 0 \ 0 \ 0 \ \dots \ 2^{u-v} \mathbf{x}_{\psi-1}'] \end{aligned} \right) / 2^u \\ \mathbf{Y}_h = & 2^{-v} [\mathbf{x}_0' \ \mathbf{x}_1' \ \mathbf{x}_2' \ \dots \ \mathbf{x}_{\psi-1}'] \end{aligned} \quad (\text{A2.2})$$

where,

Each matrix has 2^v columns

$\mathbf{x}_j$'s have length u and the last $u-v$ elements are 0

$$\begin{aligned}
\mathbf{x}_0' &= [-1 \ -1 \ -1 \ \dots \ -1 \ 0 \ 0 \ 0 \ \dots \ 0]^T \\
\mathbf{x}_1' &= [+1 \ -1 \ -1 \ \dots \ -1 \ 0 \ 0 \ 0 \ \dots \ 0]^T \\
\mathbf{x}_2' &= [-1 \ +1 \ -1 \ \dots \ -1 \ 0 \ 0 \ 0 \ \dots \ 0]^T \\
\mathbf{x}_3' &= [+1 \ +1 \ -1 \ \dots \ -1 \ 0 \ 0 \ 0 \ \dots \ 0]^T \\
&\dots \\
\mathbf{x}_{\psi-1}' &= [+1 \ +1 \ +1 \ \dots \ +1 \ 0 \ 0 \ 0 \ \dots \ 0]^T
\end{aligned}$$

Therefore we see that the form of $\mathbf{Y}_h$ is similar to $\mathbf{Y}$ with the exception of the multiplying term and the $u-v$ zero rows.

A.3 COEFFICIENT ERROR POWERS

The coefficient error powers are the average powers of the errors of the individual coefficient partitions and a cross term involving the overlapped coefficient vectors. They have been defined as follows.

$$v'(k) = E\{\mathbf{s}'_e{}^T(k)\mathbf{s}'_e(k)\} \quad (\text{A3.1})$$

$$v''(k) = E\{\mathbf{s}''_e{}^T(k)\mathbf{s}''_e(k)\} \quad (\text{A3.2})$$

$$w(k) = 2 \cdot m E\{\mathbf{m}_e{}^T(k)\mathbf{m}_e(k)\} \quad (\text{A3.3})$$

$$z(k) = 2E\{\mathbf{s}'_e{}^T(k)\mathbf{Y}\mathbf{m}_e(k)\} \quad (\text{A3.4})$$

In order to describe the evolution of these quantities over time k we use the fact that their evolution from time k to $k+1$ is governed by the LMS adaptation process. We thus begin by expressing the

LMS adaptation equations in terms of the coefficient error vectors. Subtracting s'^* from both sides of (2.10) and using (2.99) we obtain

$$s'_e(k+1) = s'_e(k) + \alpha e(k)x'(k) \quad (\text{A3.5})$$

Similarly, for the other coefficient vectors we obtain

$$s''_e(k+1) = s''_e(k) + \alpha e(k)x''(k) \quad (\text{A3.6})$$

$$m_e(k+1) = m_e(k) + \beta e(k)a(k) \quad (\text{A3.7})$$

We can also express the error, $e(k)$, in terms of the coefficient error vectors using (2.12) and (2.99)-(2.101) as follows.

$$\begin{aligned} e(k) &= d(k) - x'^T(k)s'(k) - x''^T(k)s''(k) - a^T(k)m(k) \\ &= d(k) - x'^T(k)(s'_e(k) + s'^*) - x''^T(k)(s''_e(k) + s''^*) \\ &\quad - a^T(k)(m_e(k) + m^*) \\ e(k) &= d(k) - x'^T(k)s'_e(k) - x'^T(k)s'^* - x''^T(k)s''_e(k) - x''^T(k)s''^* \\ &\quad - a^T(k)m_e(k) - a^T(k)m^* \end{aligned} \quad (\text{A3.8})$$

Substituting (A3.8) into (A3.5) we obtain

$$\begin{aligned} s'_e(k+1) &= s'_e(k) + \alpha x'(k)(d(k) - x'^T(k)s'_e(k) - x'^T(k)s'^* \\ &\quad - x''^T(k)s''_e(k) - x''^T(k)s''^* - a^T(k)m_e(k) - a^T(k)m^*) \\ s'_e(k+1) &= (I - \alpha x'(k)x'^T(k))s'_e(k) \\ &\quad + \alpha(d(k)x'(k) - x'(k)x'^T(k)s'^* - x'(k)x''^T(k)s''_e(k) \end{aligned}$$

$$\begin{aligned}
& - \mathbf{x}'(k)\mathbf{x}''^T(k)\mathbf{s}''^* - \mathbf{x}'(k)\mathbf{a}^T(k)\mathbf{m}_e(k) \\
& - \mathbf{x}'(k)\mathbf{a}^T(k)\mathbf{m}^*
\end{aligned} \tag{A3.9}$$

Similarly for the other coefficient error vectors we have

$$\begin{aligned}
\mathbf{s}_e''(k+1) = & (\mathbf{I} - \alpha\mathbf{x}''(k)\mathbf{x}''^T(k))\mathbf{s}_e''(k) \\
& + \alpha(d(k)\mathbf{x}''(k) - \mathbf{x}''(k)\mathbf{x}'^T(k)\mathbf{s}'_e(k) - \mathbf{x}''(k)\mathbf{x}'^T(k)\mathbf{s}'^* \\
& - \mathbf{x}''(k)\mathbf{x}''^T(k)\mathbf{s}''^* - \mathbf{x}''(k)\mathbf{a}^T(k)\mathbf{m}_e(k) \\
& - \mathbf{x}''(k)\mathbf{a}^T(k)\mathbf{m}^*)
\end{aligned} \tag{A3.10}$$

$$\begin{aligned}
\mathbf{m}_e(k+1) = & (\mathbf{I} - \beta\mathbf{a}(k)\mathbf{a}^T(k))\mathbf{m}_e(k) \\
& + \beta(d(k)\mathbf{a}(k) - \mathbf{a}(k)\mathbf{x}'^T(k)\mathbf{s}'_e(k) - \mathbf{a}(k)\mathbf{x}'^T(k)\mathbf{s}'^* \\
& - \mathbf{a}(k)\mathbf{x}''^T(k)\mathbf{s}''_e(k) - \mathbf{a}(k)\mathbf{x}''^T(k)\mathbf{s}''^* \\
& - \mathbf{a}(k)\mathbf{a}^T(k)\mathbf{m}^*)
\end{aligned} \tag{A3.11}$$

Combining (A3.9) and (A3.1) we obtain

$$\begin{aligned}
v'(k) = & E\{\mathbf{s}'_e{}^T(k)(\mathbf{I} - \alpha\mathbf{x}'(k)\mathbf{x}'^T(k))^2\mathbf{s}'_e(k)\} \\
& + 2\alpha E\{\mathbf{s}'_e{}^T(k)(\mathbf{I} - \alpha\mathbf{x}'(k)\mathbf{x}'^T(k))(d(k)\mathbf{x}'(k) \\
& - \mathbf{x}'(k)\mathbf{x}'^T(k)\mathbf{s}'^* - \mathbf{x}'(k)\mathbf{x}''^T(k)\mathbf{s}''_e(k) - \mathbf{x}'(k)\mathbf{x}''^T(k)\mathbf{s}''^* \\
& - \mathbf{x}'(k)\mathbf{a}^T(k)\mathbf{m}_e(k) - \mathbf{x}'(k)\mathbf{a}^T(k)\mathbf{m}^*)\} \\
& + \alpha^2 E\{(d(k)\mathbf{x}'^T(k) - \mathbf{s}'^*{}^T\mathbf{x}'(k)\mathbf{x}'^T(k) - \mathbf{s}'_e{}^T(k)\mathbf{x}''(k)\mathbf{x}'^T(k) \\
& - \mathbf{s}'^*{}^T\mathbf{x}''(k)\mathbf{x}'^T(k) - \mathbf{m}_e{}^T(k)\mathbf{a}(k)\mathbf{x}'^T(k) \\
& - \mathbf{m}^*{}^T\mathbf{a}(k)\mathbf{x}'^T(k))(d(k)\mathbf{x}'(k) - \mathbf{x}'(k)\mathbf{x}'^T(k)\mathbf{s}'^*
\end{aligned}$$

$$\begin{aligned}
& - \mathbf{x}'(k) \mathbf{x}''^T(k) \mathbf{s}'_e(k) - \mathbf{x}'(k) \mathbf{x}''^T(k) \mathbf{s}'^* - \mathbf{x}'(k) \mathbf{a}^T(k) \mathbf{m}_e(k) \\
& - \mathbf{x}'(k) \mathbf{a}^T(k) \mathbf{m}^* \} \quad (\text{A3.12})
\end{aligned}$$

Evaluating the first expectation we have

$$\begin{aligned}
& E\{\text{term 1}\} \\
& = E\{\mathbf{s}'_e{}^T(k)(\mathbf{I} - 2\alpha \mathbf{x}'(k) \mathbf{x}'^T(k) \\
& \quad + \alpha^2 \mathbf{x}'(k) \mathbf{x}'^T(k) \mathbf{x}'(k) \mathbf{x}'^T(k)) \mathbf{s}'_e(k)\}
\end{aligned}$$

Using the independence of the coefficient vectors and the input vectors described by (2.21)-(2.23) the expectation becomes

$$\begin{aligned}
& E\{\text{term 1}\} \\
& = E\{\mathbf{s}'_e{}^T(k)(\mathbf{I} - 2\alpha E\{\mathbf{x}'(k) \mathbf{x}'^T(k)\} \\
& \quad + \alpha^2 E\{\mathbf{x}'(k) \mathbf{x}'^T(k) \mathbf{x}'(k) \mathbf{x}'^T(k)\}) \mathbf{s}'_e(k)\}
\end{aligned}$$

The last inner expectation is a 4th order expectation and can be simplified by using the property for binary ± 1 symbols

$$\mathbf{x}'^T(k) \mathbf{x}'(k) = \sum_{i=0}^{m-1} (1)^2 = m \quad (\text{A3.13})$$

Therefore,

$$E\{\text{term 1}\}$$

$$\begin{aligned}
&= E\{s_e^T(k)(I - 2\alpha I + m\alpha^2 I)s_e(k)\} \\
&= (1 - 2\alpha + m\alpha^2)E\{s_e^T(k)s_e(k)\} \\
&= (1 - 2\alpha + m\alpha^2)v'(k) \tag{A3.14}
\end{aligned}$$

The second expectation of (A3.12) is

$$\begin{aligned}
&E\{\text{term 2}\} \\
&= E\{s_e^T(k)(I - \alpha x'(k)x'^T(k))(d(k)x'(k) - x'(k)x'^T(k)s'^* \\
&\quad - x'(k)x''^T(k)s_e(k) - x'(k)x''^T(k)s'^* \\
&\quad - x'(k)a^T(k)m_e(k) - x'(k)a^T(k)m^*)\} \\
&= E\{s_e^T(k)[d(k)x'(k) - x'(k)x'^T(k)s'^* - x'(k)x''^T(k)s_e(k) \\
&\quad - x'(k)x''^T(k)s'^* - x'(k)a^T(k)m_e(k) - x'(k)a^T(k)m^* \\
&\quad - \alpha(d(k)x'(k)x'^T(k)x'(k) - x'(k)x'^T(k)x'(k)x'^T(k)s'^* \\
&\quad - x'(k)x'^T(k)x'(k)x''^T(k)s_e(k) \\
&\quad - x'(k)x'^T(k)x'(k)x''^T(k)s'^* \\
&\quad - x'(k)x'^T(k)x'(k)a^T(k)m_e(k) \\
&\quad - x'(k)x'^T(k)x'(k)a^T(k)m^*)]\}
\end{aligned}$$

Using the independence of $x''(k)$ with the other input vectors (2.19)-(2.20) and the previous simplifications we obtain

$$\begin{aligned}
&E\{\text{term 2}\} \\
&= E\{s_e^T(k)\}p' - E\{s_e^T(k)\}s'^* - E\{s_e^T(k)Ym_e(k)\} \\
&\quad - E\{s_e^T(k)\}Ym^* - \alpha(mE\{s_e^T(k)\}p' - mE\{s_e^T(k)\}s'^* \\
&\quad - mE\{s_e^T(k)Ym_e(k)\} - mE\{s_e^T(k)\}Ym^*)
\end{aligned}$$

$$\begin{aligned}
&= -E\{s'_e{}^T(k)\}(s'^* - p' + Ym^*) + \alpha m E\{s'_e{}^T(k)\}(s'^* - p' + Ym^*) \\
&\quad - ((1 - m\alpha)/2)z(k)
\end{aligned} \tag{A3.15}$$

The MMSE relationships in (2.43)-(2.45) allow us to cancel out the 1st 2 terms in (A3.15) to obtain

$$E\{\text{term 2}\} = -((1 - m\alpha)/2)z(k) \tag{A3.16}$$

The third expectation of (A3.12) is

$E\{\text{term 3}\}$

$$\begin{aligned}
&= E\{d^2(k)x'^T(k)x'(k) - 2d(k)x'^T(k)x'(k)x'^T(k)s'^* \\
&\quad - 2d(k)x'^T(k)x'(k)x''^T(k)s'_e(k) - 2d(k)x'^T(k)x'(k)x''^T(k)s'^* \\
&\quad - 2d(k)x'^T(k)x'(k)a^T(k)m_e(k) - 2d(k)x'^T(k)x'(k)a^T(k)m^* \\
&\quad + s'^*T x'(k)x'^T(k)x'(k)x'^T(k)s'^* + \\
&\quad 2s'^*T x'(k)x'^T(k)x'(k)x''^T(k)s'_e(k) \\
&\quad + 2s'^*T x'(k)x'^T(k)x'(k)x''^T(k)s'^* \\
&\quad + 2s'^*T x'(k)x'^T(k)x'(k)a^T(k)m_e(k) \\
&\quad + 2s'^*T x'(k)x'^T(k)x'(k)a^T(k)m^* \\
&\quad + s'_e{}^T(k)x''(k)x'^T(k)x'(k)x''^T(k)s'_e(k) \\
&\quad + 2s'_e{}^T(k)x''(k)x'^T(k)x'(k)x''^T(k)s'^* \\
&\quad + 2s'_e{}^T(k)x''(k)x'^T(k)x'(k)a^T(k)m_e(k) \\
&\quad + 2s'_e{}^T(k)x''(k)x'^T(k)a^T(k)m^* \\
&\quad + s'^*T x''(k)x'^T(k)x'(k)x''^T(k)s'^*
\end{aligned}$$

$$\begin{aligned}
& + 2s'^*T x''(k)x'^T(k)x'(k)a^T(k)m_e(k) \\
& + 2s'^*T x''(k)x'^T(k)x'(k)a^T(k)m^* \\
& + m_e^T(k)a(k)x'^T(k)x'(k)a^T(k)m_e(k) \\
& + 2m_e^T(k)a(k)x'^T(k)x'(k)a^T(k)m^* \\
& + m^*Ta(k)x'^T(k)x'(k)a^T(k)m^* \} \\
= & mE\{d^2(k)\} - 2mp'^Ts'^* - 2mp''^TE\{s'_e(k)\} - 2mp''^Ts'^* \\
& - 2mq^TE\{m_e(k)\} - 2mq^Tm^* + ms'^*Ts'^* + 2ms'^*TYE\{m_e(k)\} \\
& + 2ms'^*TYm^* + mE\{s'_e{}^T(k)s'_e(k)\} + 2mE\{s'_e{}^T(k)\}s'^* \\
& + ms'^*Ts'^* + 2^{-m}mE\{m_e^T(k)m_e(k)\} + 2^{-m}2mE\{m_e^T(k)\}m^* \\
& + 2^{-m}mm^*Tm^* \\
= & m(E\{d^2(k)\} - p'^Ts'^* - p''^Ts'^* - q^Tm^*) + m(s'^*T - p'^T + m^*TYT)s'^* \\
& + m(s'^*T - p''^T)s'^* + m(2^{-m}m^*T - q^T + s'^*TY)m^* \\
& + 2m(s'^*T - p''^T)E\{s'_e(k)\} \\
& + 2m(2^{-m}m^*T - q^T + s'^*TY)E\{m_e(k)\} + mv''(k) + mw(k)
\end{aligned}$$

Using the intermediate formula for the MMSE in (2.95) and the other MMSE relationships as before we obtain

$$E\{\text{term 3}\} = m\text{MMSE} + mv''(k) + mw(k) \quad (\text{A3.17})$$

Combining (A3.14), (A3.16), and (A3.17) with (A3.12) we obtain our final result of

$$\begin{aligned}
v'(k+1) = & (1 - 2\alpha + m\alpha^2)v'(k) + m\alpha^2v''(k) + m\alpha^2v'(k) \\
& - \alpha(1 - m\alpha)z(k) + m\alpha^2\text{MMSE}
\end{aligned} \quad (\text{A3.18})$$

The evaluation of $v''(k+1)$, $w(k+1)$, and $z(k+1)$ is similar to that of $v'(k+1)$. The only differences to note are the following. When evaluating $v''(k+1)$, the 4th order expectations contain the term $x''^T(k)x''(k)$ which can be simplified to

$$x''^T(k)x''(k) = \sum_{i=0}^{n-m-1} (1)^2 = n-m \quad (\text{A3.19})$$

When evaluating $w(k+1)$, the 4th order expectations contain the term $a^T(k)a(k)$ which can be simplified using the fact that at any time k only one of the 2^m elements of $a(k)$ is non-zero and is a "1". Therefore the result is

$$\begin{aligned} a^T(k)a(k) &= 0^2 + 0^2 + \dots + 1^2 + 0^2 + \dots 0^2 \\ a^T(k)a(k) &= 1 \end{aligned} \quad (\text{A3.20})$$

When evaluating $z(k+1)$, the 4th order expectations contain the term $x'^T(k)Ya(k)$ or its transpose. This can be simplified if we use the relationship between Y and the mapping matrix A in (2.33) followed by the mapping definition in (1.1). The evaluation is as follows.

$$x'^T(k)Ya(k) = 2^{-m}x'^T(k)Aa(k) = 2^{-m}x'^T(k)x'(k) \quad (\text{A3.21})$$

The inner product is evaluated using (A3.13). The result is

$$\mathbf{x}'^T(k) \mathbf{Y} \mathbf{a}(k) = 2^{-m} m \quad (\text{A3.22})$$

Therefore terms such as $E\{\mathbf{x}'(k) \mathbf{x}'^T(k) \mathbf{Y} \mathbf{a}(k) \mathbf{a}^T(k)\}$, and $E\{\mathbf{x}'(k) \mathbf{x}'^T(k) \mathbf{Y} \mathbf{a}(k) d(k)\}$ would simplify to

$$E\{\mathbf{x}'(k) \mathbf{x}'^T(k) \mathbf{Y} \mathbf{a}(k) \mathbf{a}^T(k)\} = 2^{-m} m E\{\mathbf{x}'(k) \mathbf{a}^T(k)\} = 2^{-m} m \mathbf{Y} \quad (\text{A3.23})$$

$$E\{\mathbf{x}'(k) \mathbf{x}'^T(k) \mathbf{Y} \mathbf{a}(k) d(k)\} = 2^{-m} m E\{\mathbf{x}'(k) d(k)\} = 2^{-m} m \mathbf{p}'$$

In addition to the 4th order terms, 2 other special terms have to be considered when evaluating $z(k+1)$ namely $E\{\mathbf{s}'_e{}^T(k) \mathbf{Y} \mathbf{Y}^T \mathbf{s}'_e(k)\}$ and $E\{\mathbf{m}_e{}^T(k) \mathbf{Y}^T \mathbf{Y} \mathbf{m}_e(k)\}$. The former can be evaluated using the fact that $\mathbf{Y} \mathbf{Y}^T = 2^{-m} \mathbf{I}$ from (2.34) so that

$$\begin{aligned} E\{\mathbf{s}'_e{}^T(k) \mathbf{Y} \mathbf{Y}^T \mathbf{s}'_e(k)\} &= 2^{-m} E\{\mathbf{s}'_e{}^T(k) \mathbf{s}'_e(k)\} \\ &= 2^{-m} v'(k) \end{aligned} \quad (\text{A3.24})$$

The latter term can not be easily expressed in terms of $w(k)$ therefore it is evaluated recursively like the other coefficient error power quantities. We make the following extra definition.

$$w_y(k) \equiv E\{\mathbf{m}_e{}^T(k) \mathbf{Y}^T \mathbf{Y} \mathbf{m}_e(k)\} \quad (\text{A3.25})$$

Using the same technique to evaluate $v'(k+1)$ and (A3.19), (A3.20), (A3.23), and (A3.24), $v''(k+1)$, $w(k+1)$, $z(k+1)$, and $w_y(k)$ are found to be

$$\begin{aligned} v''(k+1) = & (n-m)\alpha^2 v'(k) + (1 - 2\alpha + (n-m)\alpha^2)v''(k) \\ & + (n-m)\alpha^2 w(k) + (n-m)\alpha^2 z(k) \\ & + (n-m)\alpha^2 \text{MMSE} \end{aligned} \quad (\text{A3.26})$$

$$\begin{aligned} w(k+1) = & 2^{-m}\beta^2 v'(k) + 2^{-m}\beta^2 v''(k) \\ & + (1 - 2*2^{-m}\beta + 2^{-m}\beta^2)w(k) - 2^{-m}\beta(1 - \beta)z(k) \\ & + 2^{-m}\beta^2 \text{MMSE} \end{aligned} \quad (\text{A3.27})$$

$$\begin{aligned} z(k+1) = & -2*2^{-m}\beta(1 - m\alpha)v'(k) + 2m\alpha*2^{-m}\beta v''(k) \\ & + 2m\alpha*2^{-m}\beta w(k) + (1 - \alpha - 2^{-m}\beta + 2m\alpha*2^{-m}\beta)z(k) \\ & - 2\alpha w_y(k) + 2m\alpha*2^{-m}\beta \text{MMSE} \end{aligned} \quad (\text{A3.28})$$

$$\begin{aligned} w_y(k+1) = & (2^{-m}\beta)^2 m v'(k) + (2^{-m}\beta)^2 m v''(k) \\ & + (2^{-m}\beta)^2 m w(k) - 2^{-m}\beta(1 - 2^{-m}\beta)z(k) \\ & + (1 - 2*2^{-m}\beta)w_y(k) + (2^{-m}\beta)^2 m \text{MMSE} \end{aligned} \quad (\text{A3.29})$$

(A3.18) and (A3.26)-(A3.29) recursively describe the progression of the coefficient error powers over time k . We have also found that an extra recursive relation, (A3.29), was required solely for a term in the $z(k)$ recursion.

The coefficient error power expressions derived here are used in the derivation of the $\text{AMSE}_{\text{ex}}(k)$ (see 2.2.3).

A.4 DESIGN OF THE LINEAR/NONLINEAR ECHO PATH MODEL

The criteria used in the design of the echo path model was mainly simplicity of generation, control of length, and control of residual components. The intent here was not to exactly simulate the impulse response of a cable and the nonlinear distortions introduced by various transceiver components. The approach taken in this thesis was the investigation of the echo canceller behaviour as a general adaptive filtering problem with some general constraints.

The linear "echo" was simulated with a FIR filter with u coefficients. The simple exponential function shown below was used in the computation of the first $u-1$ coefficients.

$$c(k) = a \exp(-k/\tau) \quad k = 0, 1, 2, \dots, u-2 \quad (\text{A4.1})$$

The average power of the linear echo could be determined by the inner product of the FIR coefficients with themselves because the output was the result of convolution with binary ± 1 symbols. u was chosen to be $n+1$ so that the residual or uncancellable component could be determined by the last FIR coefficient, $c(u-1)$. The residual was designed to be -70 dB with respect to the binary symbol power of 0 dB as follows.

$$10 \log(c^2(u-1)) = -70 \text{ dB} \quad (\text{A4.2})$$

The parameter, τ , was chosen so that the exponential would span approximately 5 time constants over the first $u-1$ coefficients. For example in the case where $n = 16$, u is 17 and thus we generate exponential coefficients for $k = 0$ to 15 or in other words 0 to 5τ . Therefore, in this case τ is set to 3.

The parameter, a , was chosen so that the average power in the cancellable linear component was -10 dB. The average power of the cancellable and residual linear component would therefore be approximately -10 dB. In other words,

$$\sum_{i=0}^{u-2} a^2 \exp(2k/\tau) = -10 \text{ dB} \quad (\text{A4.3})$$

The nonlinear component was simulated with a memory-based filter like that in the MEC with fixed coefficients. The coefficients were designed simply to meet the relative average power requirement of -40 dB with respect to the linear component. The nonlinear span, v , was chosen to be $m+1$. Therefore, for binary symbols, there would be 2^m residual memory coefficients. 2^m coefficients were randomly generated with an average power of -50 dB. The 2^v -coefficient memory was filled with 2 duplicates of the 2^m generated coefficients. Another set of 2^v coefficients were generated with an average power of $2 \cdot 10^{-11}$. The last 2^m coefficients were zeroed and the result was added to first set of 2^v coefficients. The end result was a nonlinear memory which, when given a 2^v binary

symbol sequence, would output a coefficient from the first 2^m or the last 2^m addresses in the memory depending on the value of the oldest input symbol. The last 2^m nonlinear coefficients are the same as the first 2^m with a small residual signal superimposed on it. The MEC's 2^m memory effectively can only cancel the first 2^m coefficients in the nonlinear memory. Therefore when the MEC converges, the cancellation error will contain the small residual signal. Actually, this residual will exist 50% of the time because the nonlinear memory will output a coefficient from the first cancellable 2^m coefficients 50% of the time. The average power of this residual is 1×10^{-11} or -110 dB.

A.5 SOFTWARE FOR THEORETICAL CALCULATIONS

All theoretical calculations were performed using the MATLAB mathematical calculator environment from the MATHWORKS Ltd. MATLAB is both an interactive and programmable environment which is optimized for matrix and vector intensive operations. The version of MATLAB used was installed on an IBM PC AT compatible computer with an Intel 80287 coprocessor. The following is a listing of the MATLAB script and function files written to calculate the AMSE, the region of convergence for the adaptation step-sizes, and the optimum step-sizes.

A.5.1 AMSE CALCULATION

```

%-----
%
%   GETAMSEJ.M
%
%           This SCRIPT file facilitates the generation of a graph of
%           Average Mean Square Error (AMSE) vs "time" for
%           theJointly-Adaptive
%           TEC/MEC.
%
%   Programmer:   Ken Yamazaki
%   Date:         Jan. 11, 1989
%-----

%
% display opening message
%

fprintf('*****\n')
fprintf('AMSE PLOT GENERATOR FOR JOINTLY-ADAPTIVE TEC/MEC\n')
fprintf('*****\n\n')

%
% get TEC span
%
n = input('Enter TEC span ->');

%
% get TEC adaptation step-size
%
alpha = input('Enter TEC adaptation step-size ->');

%
% get MEC span
%
ok = 0;
while ~ok
    m = input('Enter MEC span ->');
    if m > n

```

```

        fprintf(' * MEC span must <= TEC span *\n')
    else
        ok = 1;
    end
end

%
% get MEC adaptation step-size and calculate beta0
%
beta = input('Enter MEC adaptation step-size ->');
beta0 = beta*2^(-m);

ok = 0;
while ~ok
    %
    % get linear echo span
    %
    u = input('Enter LINEAR span ->');

    %
    % get LINEAR CHANNEL vector from file
    %
    linchan = input('Name of file containing LINEAR coefficients -
>','s');
    command = ['load ',linchan];
    eval(command);
    hl = eval(linchan);
    if length(hl) < u
        fprintf('* Not enough LINEAR coefficients in file *\n')
    else
        if length(hl) > u
            fprintf('* Accepting first %.0f LINEAR coefficients...
*\n',u)
            hl = hl(1:u);
        end
        ok = 1;
    end
end

ok = 0;

```

```

while ~ok
    %
    % get nonlinear echo span
    %
    v = input('Enter NONLINEAR span ->');
    if v > u | v < m
        fprintf('* MEC span <= NONLINEAR span <= LINEAR span
*\n')
    else
        %
        % get NONLINEAR CHANNEL vector from file
        %
        hnsiz = 2^v;
        nonlinchan = input('Name of file containing NONLINEAR
coefficients ->', 's');
        command = ['load ', nonlinchan];
        eval(command);
        hn = eval(nonlinchan);
        if length(hn) < hnsiz
            fprintf('* Not enough NONLINEAR coefficients in file
*\n')
        else
            if length(hn) > hnsiz
                fprintf('* Accepting first %.0f NONLINEAR
coefficients... *\n', hnsiz)
                hn = hn(1:hnsiz);
            end
            ok = 1;
        end
    end
end

%
% get NOISE power in dB
%
noise = input('Enter ADDITIVE NOISE (dB) (-999 specifies NO noise) -
>');

%
% ask user if a plot of MSE(k) is desired

```



```

%
pflag = input('Generate Plot? (Y - plot, other - no plot) ->', 's');

%
% calculate AVERAGE ECHO POWER
%
echopow = edk2gen(hl, hn, noise);

%
% calculate MMSE
%

mmse = mmsegen(n, m, hl, hn, noise);

%
% calculate initial coefficient error power vector and get optimum
% coefficients
%
[z0, spopt, sdopt, mopt] = z0gen(n, alpha, m, beta, hl, hn);

%
% calculate b vector
%
b = bgen(n, alpha, m, beta);

%
% calculate COEFFICIENT ERROR POWER UPDATE MATRIX
%
t = tgen(n, alpha, m, beta);

%
% calculate EXCESS STEADY STATE MATRIX COMPONENT
%

xsscomp = mmse*inv(t)*b;

%
% calculate STEADY STATE AMSE
%
ssamse = sum(xsscomp) + mmse;

```

```

%
% calculate EXCESS TRANSIENT MATRIX COMPONENT
%
xtranscomp0 = z0 - xsscomp;

%
% calculate AMSE(0) for normalization purposes
%
amse0 = sum(z0) + mmse;

if pflag == 'y' | pflag == 'Y'
    %
    % get start time
    %
    start = input('Enter START time ->');

    %
    % get finish time
    %
    ok = 0;
    while ~ok
        finish = input('Enter FINISH time ->');
        if finish < start
            fprintf('* Finish time must be >= Start time *\n')
        else
            ok = 1;
        end
    end

    %
    % get time step
    %
    ok = 0;
    while ~ok
        step = input('Enter time step ->');
        if step > finish-start
            fprintf('* Time step must be <= Finish time - Start
time *\n')
        else

```

```

        ok = 1;
    end
end

%
% calculate AMSE(k)
%
temp = (eye(4) - t)^start;
delta = (eye(4) - t)^step;

for k = start:step:finish
    temp2 = temp*xstranscomp0;
    amse((k-start)/step + 1) = sum(temp2) + ssamse;
    temp = temp * delta;
end

%
% ask user if a normalized or unnormalized plot is desired
%
nflag = input('Normalize plot wrt AMSE(0) ? (Y - yes, other - no)
->', 's');

if nflag == 'y' | nflag == 'Y'
    %
    % normalize AMSE wrt AMSE(0)
    %
    amse = 10*log10(amse/amse0);
else
    amse = 10*log10(amse);
end

%
% set up plot axis
%
amseaxis = start:step:finish;

%
% get compare file
%
comparefile = input('Enter compare file name or NOFILE ->', 's');

```

```

        if comparefile(1:6) ~= 'nofile' & comparefile(1:6) ~= 'NOFILE'
            command = ['load ',comparefile];
            eval(command);
            compdata = eval(comparefile);
            if nflag == 'y' | nflag == 'Y'
                compdata =
10*log10(compdata(start+1:finish)/amse0);
            else
                compdata = 10*log10(compdata(start+1:finish));
            end
            compaxis = start:length(compdata)-1;
            plot(amseaxis,amse,compaxis,compdata,':');
        else
            plot(amseaxis,amse);
        end

%
% label plot
%
grid;

xlabel('k');
if nflag == 'y' | nflag == 'Y'
    ylabel('AMSE(k) - AMSE(0) (dB)');
else
    ylabel('AMSE(k) (dB)');
end

%
% generate plot title
%
% Note: this must be done in parts because the printf and
sprintf functions
%           can only handle 3 arguments at a time
%
ttl0 = sprintf('N=%0.f a=%0.f M=%0.f ',n,alpha,m);
ttl1 = sprintf('b=%0.f u=%0.f v=%0.f',beta,u,v);
ttl = [ttl0, ttl1];

title(ttl);

```

3

Micro-D

```

        pause;
end

%
% display important system values
%
fprintf('\n=====\\n');
fprintf('          A Summary of System Values\\n');
fprintf('=====\\n\\n');
fprintf('n=%0.0f alpha=%0f m=%0.0f ',n,alpha,m);
fprintf('beta=%0f u=%0.0f v=%0.0f\\n\\n',beta,u,v);
fprintf('Linear Echo Path Coefficients:\\n');
hl'
fprintf('\\nNonlinear Echo Path Coefficients:\\n');
hn'
fprintf('\\nAdditive Noise: %0f dB\\n',noise);
fprintf('Average Total Echo Power with Noise: %0f
dB\\n',10*log10(echopow));
fprintf('-----\\n\\n');
fprintf('Initial Average Mean Square Error (AMSE(0)): %0f
dB\\n',10*log10(amse0));
fprintf('Minimum Mean Square Error (MMSE): %0f
dB\\n',10*log10(mmse));
fprintf('Steady State Mean Square Error (SSAMSE): %0f
dB\\n',10*log10(ssamse));
fprintf('Excess Mean Square Error (SSAMSE-MMSE): %0f
dB\\n\\n',10*log10(ssamse/mmse));

fprintf('Press Any Key to Continue ->');
pause;
fprintf('Optimum Overlapped TEC Coefficients:\\n');
spopt'
fprintf('\\nOptimum Non-overlapped TEC Coefficients:\\n');
sdpopt'
fprintf('\\nOptimum MEC Coefficients:\\n');
mopt'

```

Fig. A.1 The GETAMSEJ script file.

```

%-----
%
%      b = BGEN(n,alpha,m,beta)
%
%          b generator for the Jointly-Adaptive TEC/MEC.
%
%      Parameters:
%
%          n          TEC span
%          alpha      TEC adaptation step-size
%          m          MEC span (where m <= n)
%          beta       MEC adaptation step-size
%
%      Programmer:   Ken Yamazaki
%      Date:         Aug.2, 1989
%-----

function b = bgen(n,alpha,m,beta)

b(1,1) = m*alpha^2;
b(2,1) = (n-m)*alpha^2;
b(3,1) = 2^(-m)*beta^2;
b(4,1) = 2*m*alpha*2^(-m)*beta;

```

Fig. A.2 The BGEN function file.

```

%-----
%
%      bs = BSGEN(n,alpha,m,beta)
%
%              bs generator for Phase II of the Sequentially-Adaptive
%              TEC/MEC.
%
%      Parameters:
%
%              n          TEC span
%              alpha      TEC adaptation step-size
%              m          MEC span (where m <= n)
%              beta       MEC adaptation step-size
%
%      Programmer:      Ken Yamazaki
%      Date:            Aug. 5, 1989.
%-----

function bs = bsgen(n,alpha,m,beta)

bs(1,1) = (n-m)*alpha^2;
bs(2,1) = 2^(-m)*beta^2;

```

Fig. A.3 The BSGEN function file.


```

%-----
%
%      Edk2 = EDK2GEN(hl,hn,noise)
%
%      Mean Squared Echo,  $E\{d(k)^2\}$ , generator.
%
%      Parameters:
%
%      hl          linear echo path coefficients
%      hn          nonlinear echo path coefficients (length must
%                be a power of 2)
%      noise       additive noise (dB) (-999 specifies no noise)
%
%      Programmer:  Ken Yamazaki
%      Date:        March 5, 1989
%-----

```

```

function edk2 = edk2gen(hl,hn,noise)

%
% calculate lengths
%
u = length(hl);
v = round(log10(length(hn))/log10(2));

%
% Calculate Yh.
%
yh = ygen(u,v);

%
% convert noise from dB's to squared units
%
if noise == -999

```

```
        nu = 0;
else
    nu = 10^(noise/10);
end

edk2 = hl'*hl + 2*hl'*yh*hn + hn'*hn*2^(-v) + nu;
```

Fig. A.4 The EDK2GEN function file.

```

%-----
%
%   Fm = FMGEN(m,u)
%
%           Linear Echo Path Input -> Overlapped TEC Input
%           Transformation Matrix, Fm, generator.
%
%   Parameters:
%
%           m           overlapped span
%           u           linear echo input span
%
%   Programmer:   Ken Yamazaki
%   Date:         April 27, 1989
%-----

```

```

function fm = fmgen(m,u)

fm = [eye(m) zeros(m,u-m)];

```

Fig. A.5 The FMGEN function file.

```

%-----
%
%   Fn = FNGEN(n, m, u)
%
%           Linear Echo Path Input -> Non-overlapped TEC input
%           Transformation Matrix, Fn, generator.
%
%   Parameters:
%           n           TEC span
%           m           MEC span (where m <= n)
%           u           Linear Echo span (where u > n)
%
%   Programmer:   Ken Yamazaki
%   Date:         May 5, 1989
%-----
%
function fn = fngen(n,m,u)

%
% generate fn
%
fn = [zeros(n-m,m) eye(n-m) zeros(n-m,u-n)];

```

Fig. A.6 The FNGEN function file.

```

%-----
%
%   Fphi = FPHIGEN(m, v)
%
%           Nonlinear Echo Path Input -> MEC input Transformation
%           Matrix, Fphi, generator.
%
%   Parameters:
%       m           MEC input span
%       v           Nonlinear input span
%
%   Programmer:   Ken Yamazaki
%   Date:         March 13, 1989
%-----

function fphi = fphigen(m,v)

%
% calculate sizes
%

msz = 2^m;
vsz = 2^v;
dsz = 2^(v-m);

fphi = zeros(msz, vsz);

for i = 0:dsz-1
    for j = 1:msz
        fphi(j, i*msz+j) = 1;
    end
end
end

```

Fig. A.7 The FPHIGEN function file.

```

%-----
%
%      MMSE = MMSEGEN(n,m,h1,hn,noise)
%
%      Minimum Mean Square Error, MMSE, generator.
%
%      Parameters:
%
%      n          TEC span
%      m          MEC span (where m <= n)
%      h1         linear echo path coefficients
%      hn         nonlinear echo path coefficients (length must
%                be a power of 2)
%      noise      additive noise (dB) (-999 specifies no noise)
%
%      Programmer: Ken Yamazaki
%      Date:       May 5, 1989
%-----
function mmse = mmsegen(n,m,h1,hn,noise)

%
% generate  $E\{d(k)^2\}$ 
%
edk2 = edk2gen(h1,hn,noise);

%
% generate  $p''$  and  $q$ 
%
pdp = pdpgen(n,m,h1,hn);
q = qgen(m,h1,hn);

%
% calculate mmse
%

mmse = edk2 - q'*q*2^m - pdp'*pdp;

```

Fig. A.8 The MMSEGEN function file.

```

%-----
%
%      mopt = MOPTGEN(alpha,m,beta,h1,hn)
%
%      Optimum MEC coefficient vector, mopt, generator.
%
%      Parameters:
%
%      alpha      TEC adaptation step-size
%      m          nonlinear input span
%      beta       MEC adaptation step-size
%      h1         linear echo path coefficients
%      hn         nonlinear echo path coefficients (length must
%                be a power of 2)
%
%      Programmer:  Ken Yamazaki
%      Date:       Aug. 2, 1989
%-----

function mopt = moptgen(alpha,m,beta,h1,hn)

%
% calculate A
%
a = ygen(m,m)*2^m;

%
% calculate q
%
q = qgen(m,h1,hn);

%
% calculate mopt
%
mopt = (eye(2^m)*2^m - a'*a*alpha/(alpha+beta*2^(-m)))*q;

```

Fig. A.9 The MOPTGEN function file.

```

%-----
%
%      pdp = PDPGEN(n,m,h1,hn)
%
%      Non-overlapped TEC input/echo crosscorrelation vector,
%      pdp, generator.
%
%      Parameters:
%
%      n          TEC span
%      m          MEC span
%      h1         linear echo path coefficients
%      hn         nonlinear echo path coefficients (length must
%               be a power of 2)
%
%      Programmer: Ken Yamazaki
%      Date:       Aug. 2, 1989
%-----

```

```

function pdp = pdpgen(n,m,h1,hn)

%
% calculate lengths
%
u = length(h1);
v = round(log10(length(hn))/log10(2));

%
% calculate fn
%
fn = fngen(n,m,u);

%
% calculate Yh
%

```



```
yh = ygen(u,v);  
  
%  
% calculate pdoubleprime  
%  
pdp = fn*(hl + yh*hn);
```

Fig. A.10 The PDPGEN function file.

```

%-----
%
%   pp = PPGEN(m,h1,hn)
%
%           Overlapped TEC input/echo crosscorrelation vector, pp,
%           generator.
%
%   Parameters:
%
%           m           MEC span
%           h1          linear echo path coefficients
%           hn          nonlinear echo path coefficients (length must
%                      be a power of 2)
%
%   Programmer:   Ken Yamazaki
%   Date:         Aug. 2, 1989
%-----

```

```
function pp = ppgen(m,h1,hn)
```

```

%
% calculate lengths
%
u = length(h1);
v = round(log10(length(hn))/log10(2));

%
% calculate Fm
%
fm = fmgen(m,u);

%
% calculate Yh
%

```

```
yh = ygen(u,v);  
  
%  
% calculate pprime  
%  
pp = fm*(hl + yh*hn);
```

Fig. A.11 The PPGEN function file.

```

%-----
%
%   q = QGEN(m,h1,hn)
%
%           MEC input/echo cross correlation vector, q, generator.
%
%   Parameters:
%
%           m           MEC span
%           h1          linear echo path coefficients
%           hn          nonlinear echo path coefficients (length must
%                   be a power of 2)
%
%   Programmer:   Ken Yamazaki
%   Date:        Aug. 2, 1989
%-----

```

```

function q = qgen(m,h1,hn)

```

```

%
% calculate lengths
%
u = length(h1);
v = round(log10(length(hn))/log10(2));

%
% calculate Fphi
%
fphi = fphigen(m,v);

%
% calculate Yh
%

```

```
yh = ygen(u,v);  
  
%  
% calculate q  
%  
q = fphi*(yh'*hl + hn*2^(-v));
```

Fig. A.12 The QGEN function file.

```

%-----
%
%      spopt = SPOPTGEN(alpha,m,beta,h1,hn)
%
%      Optimum overlapped TEC coefficients, spopt, generator.
%
%      Parameters:
%
%      alpha      TEC adaptation step-size
%      m          nonlinear input span
%      beta       MEC adaptation step-size
%      h1         linear echo path coefficients
%      hn         nonlinear echo path coefficients (length must
%                be a power of 2)
%
%      Programmer:  Ken Yamazaki
%      Date:        Aug. 2, 1989
%-----

function spopt = spoptgen(alpha,m,beta,h1,hn)

%
% calculate pprime
%
pp = ppgen(m,h1,hn);

%
% calculate spopt
%
spopt = pp*alpha/(alpha + beta*2^(-m));

```

Fig. A.13 The SPOPTGEN function file.

```

%-----
%
%      T = TGEN(n,alpha,m,beta)
%
%              Coefficient Error Power Update Matrix, T, generator for
%              the Jointly-Adaptive TEC/MEC.
%
%      Parameters:
%          n          TEC span
%          alpha      TEC adaptation step-size
%          m          MEC span
%          beta       MEC adaptation step-size
%
%      Programmer:   Ken Yamazaki
%      Date:         June 23, 1989
%-----

```

```
function t = tgen(n,alpha,m,beta)
```

```

a2 = alpha^2;
b2 = beta^2;
m2 = 2^(-m);
b0 = beta*m2;

```

```

t(1,1) = alpha*(2 - m*alpha);
t(1,2) = -m*a2;
t(1,3) = -m*a2;
t(1,4) = alpha*(1 - m*alpha);;
t(2,1) = -(n-m)*a2;
t(2,2) = alpha*(2 - (n-m)*alpha);
t(2,3) = -(n-m)*a2;
t(2,4) = -(n-m)*a2;
t(3,1) = -m2*b2;
t(3,2) = -m2*b2;
t(3,3) = b0*(2 - beta);

```

```
t(3,4) = b0*(1 - beta);  
t(4,1) = 2*b0*(1-m*alpha-b0/alpha);  
t(4,2) = -2*m*alpha*b0;  
t(4,3) = -2*m*alpha*b0;  
t(4,4) = alpha + 3*b0 - 2*m*alpha*b0;
```

Fig. A.14 The TGEN function file.


```

%-----
%
%      Ts = TSGEN(n,alpha,m,beta)
%
%      Coefficient Error Power Update Matrix, Ts, generator for
%      the Sequentially-Adaptive TEC/MEC's Phase II of
%      operation
%
%      Parameters:
%      n          TEC span
%      alpha      TEC adaptation step-size
%      m          MEC span
%      beta       MEC adaptation step-size
%
%      Programmer:  Ken Yamazaki
%      Date:       July 3, 1989
%-----

function ts = tsgen(n,alpha,m,beta)

a2 = alpha^2;
b2 = beta^2;
m2 = 2^(-m);
b0 = m2*beta;

ts(1,1) = alpha*(2 - (n-m)*alpha);
ts(1,2) = -(n-m)*a2;
ts(2,1) = -m2*b2;
ts(2,2) = b0*(2-beta);

```

Fig. A.15 The TSGEN function file.

```

%-----
%
%   Y = YGEN(n, m)
%
%           Linear/Nonlinear Input Crosscorrelation Matrix Generator
%           Used for calculation of Y and Yh.
%
%   Parameters:
%
%           n           Linear input span
%           m           Nonlinear input span (where m <= n)
%
%   Programmer:   Ken Yamazaki
%   Date:         Jan. 11, 1989
%-----

```

```
function y = ygen(n,m)
```

```

%
% calculate sizes
%
msz = 2^m;
rmsz = 1/msz;

%
% initialize cross correlation matrix
%
y = zeros(n,msz);

%
% calculate cross correlation elements
%
sign = -1;
for u = 0:m-1
    for v = 0:2^u-1
        for w=1:2^u:msz

```

```
        y(u+1,w+v)=sign*rmsz;  
        sign=-sign;  
    end  
end  
end
```

Fig. A.16 The YGEN function file.

```

%-----
%
%      [z0,spopt,sdpopt,mopt] = Z0GEN(n,alpha,m,beta,h1,hn)
%
%      Initial Coefficient Error Power Vector, z0, generator.
%      Also returns optimum coefficient vectors.
%
%      Parameters:
%
%      n          TEC span
%      alpha      TEC adaptation step-size
%      m          MEC span
%      beta       MEC adaptation step-size
%      h1         linear echo path coefficients
%      hn         nonlinear echo path coefficients (length must
%                be a power of 2)
%
%      Programmer:  Ken Yamazaki
%      Date:       Aug. 2, 1989
%-----

```

```

function [z0,spopt,sdpopt,mopt] = z0gen(n,alpha,m,beta,h1,hn)

```

```

%
% calculate optimum TEC/MEC coefficients
%
spopt = spoptgen(alpha,m,beta,h1,hn);
sdopt = pdpgen(n,m,h1,hn);
mopt = moptgen(alpha,m,beta,h1,hn);
%
% generate y
%
y = ygen(m,m);
%
% calculate initial coefficient error power vector

```

```
%  
z0(1,1) = spopt'*spopt;  
z0(2,1) = sdpop't*sdpop't;  
z0(3,1) = mop't*mop't*2^(-m);  
z0(4,1) = 2*spopt'*y*mop't;
```

Fig. A.17 The Z0GEN function file.

A.5.2 REGION OF CONVERGENCE CALCULATION

```

%-----
%
%      [el,es] = EIGTGEN(n,m,na1,nastep,na2,b1,bstep,b2)
%
%      This function generates a largest-eigenvalue matrix and a
%      smallest-eigenvalue matrix containing the largest and
%      smallest eigenvalues of the T matrix (Coefficient Error
%      Power Update Matrix for the Jointly-Adaptive TEC/MEC)
%      respectively for a given TEC and MEC size and a range of
%      alpha and beta's.
%
%      Parameters:
%
%      n          TEC span
%      m          MEC span
%      na1        n*starting TEC adaptation step-size
%      nastep     n*TEC adaptation step-size step amount
%      na2        n*ending TEC adaptation step-size
%      b1         starting MEC adaptation step-size
%      bstep      MEC adaptation step-size step amount
%      b2         ending MEC adaptation step-size
%
%      Date:  July 3, 1989
%      Programmer:  Ken Yamazaki
%-----

```

```
function [el,es] = eigtgen(n,m,na1,nastep,na2,b1,bstep,b2)
```

```
[na,b]=meshdom(na1:nastep:na2,b1:bstep:b2);
```

```
esize = size(na);
```

```
for i = 1:esize(1,2)
    for j = 1:esize(1,1)
```

```
        eig = eig(tgen(n,na(j,i)/n,m,b(j,i)));  
        el(j,i) = max(eigt);  
        es(j,i) = min(eigt);  
    end  
end
```

Fig. A.18 The EIGTGEN function file.

```

%-----
%
%      [el,es] = EIGTSGEN(n,m,nma1,nmastep,nma2,b1,bstep,b2)
%
%      This function generates a largest-eigenvalue matrix and a
%      smallest-eigenvalue matrix containing the largest and
%      smallest eigenvalues of the Ts matrix (Coefficient Error
%      Power Update Matrix for Phase II of the Sequentially
%      -Adaptive TEC/MEC) respectively for a given TEC and
%      MEC size and a range of alpha and beta's.
%
%      Parameters:
%
%      n          TEC span
%      m          MEC span
%      nma1       (n-m)*starting TEC adaptation step-size
%      nmastep    (n-m)*TEC adaptation step-size step amount
%      nma2       (n-m)*ending TEC adaptation step-size
%      b1         starting MEC adaptation step-size
%      bstep      MEC adaptation step-size step amount
%      b2         ending MEC adaptation step-size
%
%      Date:  July 3, 1989
%      Programmer:  Ken Yamazaki
%-----

```

```
function [el,es] = eigtsge(n,m,nma1,nmastep,nma2,b1,bstep,b2)
```

```
[nma,b]=meshdom(nma1:nmastep:nma2,b1:bstep:b2);
```

```
esize = size(nma);
```

```
for i = 1:esize(1,2)
```

```
    for j = 1:esize(1,1)
```



```
        eigts = eig(tsgen(n,nma(j,i)/(n-m),m,b(j,i)));  
        el(j,i) = max(eigts);  
        es(j,i) = min(eigts);  
    end  
end
```

Fig. A.19 The EIGTSGEN function file.

A.5.3 OPTIMUM ADAPTATION STEP-SIZE DETERMINATION FOR FASTEST AMSE CONVERGENCE

```

%-----
%
%      [aopt,bopt,opt] = OPTIMUM(n,nastep,m,bstep)
%
%      This function finds the optimum step-sizes for the Jointly
%      -Adaptive TEC/MEC and also returns the maximum
%      smallest eigenvalue of the associated coefficient error
%      power update matrix, T.
%
%      Parameters:
%
%          n          TEC span
%          nastep     optimum n*alpha search step-size
%          m          MEC span
%          bstep      optimum beta search step-size
%
%      Programmer:   Ken Yamazaki
%      Date:         July 7, 1989
%-----

function [aopt,bopt,opt] = optimum(n,nastep,m,bstep)

opt = -99;

for na = nastep:nastep:1
    for b = bstep:bstep:1
        x = min(eig(tgen(n,na/n,m,b)));
        if x > opt
            opt = x;
            naopt = na;
            bopt = b;
        end
    end
end

aopt = naopt/n;

```

Fig. A.20 The OPTIMUM function file.

```

%-----
%
%      [aopt,bopt,opt] = OPTIMUMS(n,nmstep,m,bstep)
%
%      This function finds the optimum step-sizes for Phase II of
%      the Sequentially-Adaptive TEC/MEC and also returns the
%      maximum smallest eigenvalue of the associated
%      coefficient error power update matrix, Ts.
%
%      Parameters:
%          n          TEC span
%          nmstep     optimum (n-m)*alpha search step-size
%          m          MEC span
%          bstep      optimum beta search step-size
%
%      Programmer:   Ken Yamazaki
%      Date:         July 17, 1989
%-----

function [aopt,bopt,opt] = optimums(n,nmstep,m,bstep)

opt = -99;

for nma = nmstep:nmstep:1
    for b = bstep:bstep:1
        x = min(eig(tsgen(n,nma/(n-m),m,b)));
        if x > opt
            opt = x;
            nmaopt = nma;
            bopt = b;
        end
    end
end

aopt = nmaopt/(n-m);

```

Fig. A.21 OPTIMUMS function file.

A.5.4 EVALUATION OF FACTORS AFFECTING STEADY STATE AMSE LEVELS

```

%-----
%
%   jxs = JNTXSGEN(n,m,step)
%
%       Excess AMSE factor, sum(Tinv*b), contour generator for
%       the Jointly-Adaptive TEC/MEC. The factor is calculated
%       over a range of n*alpha and beta defined by the
%       triangular region in which the eigenvalues of T are
%       between 0 and 2. A common step for n*alpha and beta is
%       accepted. The results are returned in a square matrix in
%       which the lower triangle contains non-zero
%       elements.
%
%   Parameters:
%
%       n          TEC span
%       m          MEC span (where m <= n)
%       step       common step of n*alpha and beta
%
%   Programmer:   Ken Yamazaki
%   Date:         Aug. 5, 1989.
%-----

```

```

function jxs = jntxsgen(n,m,step)

[na,b]=meshdom(step:step:2-step,step:step:2-step);

jxssize = size(na);

for i = 1:jxssize(1,1)
    for j = i:jxssize(1,1)
        jxs(j,i) = tibgen(n,na(j,i)/n,m,b(j,i));
    end
end
end

```

Fig. A.22 The JNTXSGEN function file.

```

%-----
%
%   sxs = SEQXSGEN(n,m,step)
%
%       Excess AMSE factor, sum(Tsinv*bs), contour generator for
%       the Sequentially-Adaptive TEC/MEC. The factor is
%       calculated over a range of (n-m)*alpha and beta defined
%       by the triangular region in which the eigenvalues of Ts
%       are between 0 and 2. A common step for (n-m)*alpha
%       and beta is accepted. The results are returned in
%       a square matrix in which the lower triangle contains
%       non-zero elements.
%
%   Parameters:
%
%       n          TEC span
%       m          MEC span (where m <= n)
%       step       common step value for (n-m)*alpha and beta
%
%   Programmer:   Ken Yamazaki
%   Date:         Aug. 5, 1989.
%-----

function sxs = seqxsgen(n,m,step)

[nma,b]=meshdom(step:step:2-step,step:step:2-step);

sxssize = size(nma);

for i = 1:sxssize(1,1)
    for j = i:sxssize(1,1)
        sxs(j,i) = tsibsgen(n,nma(j,i)/(n-m),m,b(j,i));
    end
end
end

```

Fig. A.23 The SEQXSGEN function file.

```

%-----
%
%      Tib = TIBGEN(n,alpha,m,beta)
%
%              Excess AMSE factor, sum(Tinv*b), generator for the
%              Jointly-Adaptive TEC/MEC.
%
%      Parameters:
%
%              n          TEC span
%              alpha      TEC adaptation step-size
%              m          MEC span (where m <= n)
%              beta       MEC adaptation step-size
%
%      Programmer:      Ken Yamazaki
%      Date:            Aug. 5, 1989.
%-----

function tib = tibgen(n,alpha,m,beta)

%
% Note that we don't need to actually find the inverse of T because
% the product can be found by simple gaussian elimination by
% specifying the "\" operator.
%
tib = sum(tgen(n,alpha,m,beta)\bgen(n,alpha,m,beta));

```

Fig. A.24 The TIBGEN function file.

```

%-----
%
%      Tsibs = TSIBSGEN(n,alpha,m,beta)
%
%              Excess AMSE factor, sum(Tsinv*bs), generator for Phase II
%              of the Sequentially-Adaptive TEC/MEC.
%
%      Parameters:
%
%              n          TEC span
%              alpha      TEC adaptation step-size
%              m          MEC span (where m <= n)
%              beta       MEC adaptation step-size
%
%      Programmer:   Ken Yamazaki
%      Date:         Aug. 5, 1989.
%-----

```

function tsibs = tsibsgen(n,alpha,m,beta)

```

%
% Note that we don't need to actually find the inverse of Ts because
% the product can be found by simple gaussian elimination by
% specifying the "\" operator.
%
tsibs = sum(tsgen(n,alpha,m,beta)\bsgen(n,alpha,m,beta));

```

Fig. A.25 The TISBSGEN function file.

APPENDIX B

CAPSIM SIMULATIONS

B.1 A BRIEF INTRODUCTION TO CAPSIM

CAPSIM, or CAPture and SIMulate, is a graphic workstation-based discrete-time systems simulation environment. It is especially suited for simulation of digital signal processing and sampled-data communication systems.

CAPSIM's attractiveness lies in the ease in which a simulation can be configured and altered. Simulations are configured in the graphic domain by creating a block diagram of the system. The block diagram can be hierarchical. At any particular level in the hierarchy, a block may internally consist of an interconnection of more blocks. Furthermore, this idea of hierarchy extends to the parameters of a block. A block's parameter set can consist of the parameters from any block in the hierarchy of blocks that it contains.

Simulation input is derived from files or data generation blocks. A simulation is accomplished in 2 stages. As the block diagram is specified, CAPSIM reads or "captures" the block and interconnection information and converts it into a more compact

non-graphical form. When the simulation is run, CAPSIM proceeds using this non-graphical form. Simulation output may be in the form of files or graphic plots of signals which can be displayed and printed.

The user-interface is window-based with pop-up menus for mouse-selectable functions and keyboard entry.

Besides CAPSIM, there are similar commercially available graphically-oriented block diagram communications and signal processing simulation environments, namely, SPW (Signal Processing Worksystem) and BOSS (Block Oriented Systems Simulator) from Comdisco Ltd., and MATRIX_x from Integrated Systems Inc.

B.1.1 A BRIEF HISTORY OF CAPSIM

CAPSIM had its roots in an older non-graphical block oriented systems simulation environment called BLOSIM [Mes'84] which was developed at the University of California at Berkeley. The hierarchical block diagram concept was supported through text files containing the block and interconnection information. The fact that users could not easily visualize the simulation upon examination of the text and the bug-ridden nature of the software prompted developers at North Carolina State University, with funding from members of the Centre for Communications and Signal Processing (CCSP), to improve the reliability and ease of use of BLOSIM by completely restructuring the software and devising a graphical-

interface. This new improved graphically based ascendant of BLOSIM became CAPSIM [Nob'88].

Currently, CAPSIM is in its 3rd release. Future development of CAPSIM will involve its porting to UNIX based workstations supporting the X-Windows standard. It is currently available on DEC VAX workstations operating under the VMS operating system. Currently it is only available in a non-graphical form on UNIX workstations.

As CAPSIM v3.0 was unstable at the time of the writing of this thesis, all descriptions of CAPSIM in this appendix refer to the next most recent release, that is, CAPSIM v2.0. The major improvements in CAPSIM v3.0 pertain mainly to the user-interface and therefore the descriptions here are still generally applicable.

B.1.2 BUILDING BLOCKS

There are only 3 types of simulation entities designated by CAPSIM terminology. Due to the regular hierarchy supported by CAPSIM, each can be referred to generically as a "block". All blocks, with the exception of the highest level block type, have a number of inputs and or outputs. A block with no inputs is a "source" block. A block with no output connections most likely outputs data directly to a file or the screen.

A "star" represents the lowest or most primitive level in the hierarchy. It is defined using a pseudo-C type language and is written by the user or developer to perform a relatively simple task

such as adding or simple filtering. Multiple "instances" of a star with the same or different parameters are supported within a simulation.

A "galaxy" can be one or many levels above that of the star. It is a specification of a group of blocks and their interconnections. The term block is used here because they may be stars or other galaxies. A galaxy has galaxy inputs and outputs. Typical galaxies might be an adaptive echo canceller or adaptive predictive coder. Like stars, multiple instances of a galaxy are supported.

Connections between stars and galaxies are implemented via variable length FIFO (First In First Out) buffers. A connection consists of an output buffer from the source block and an input buffer of the destination block. The buffering operations are handled automatically by CAPSIM. Implementing connections in this way allows for lower simulation execution overhead and provides a convenient representation of the data for many DSP operations.

The uppermost block is referred to as the "universe". The universe represents the simulation as a whole. A universe contains no input nor output connections. All input and output functions are handled by source blocks and sink blocks which may read and write data files or plot data. Typical universes might be an ISDN U-interface transceiver or data compression and expansion demonstration.

Fig. B.1 illustrates a simple example of a universe with stars and a galaxy as they would appear in the CAPSIM environment. The

numbers on the connections denote input and output buffer numbers. Fig. B.2 gives an exploded view of the galaxy in Fig. B.1.

Fig. B.1 An example of a CAPSIM universe.

Fig. B.2 An exploded view of the FILTER galaxy.

The following sections provide some more details about the blocks and connections between them. A more comprehensive description is found in [Ard'88].

B.1.2.1 BLOCK CONNECTIONS

Stars and galaxies are connected with FIFO (First In First Out) buffers. A connection is made when the output buffer of a block is

connected to the input buffer of another block. Fig. B.3 illustrates how a connection is made between 2 blocks.

Fig. B.3 An interblock connection.

When a block on the source end of a connection is "visited" during execution, its output is stored in the output buffer. It may process and output any number of data samples into this buffer (subject to memory limitations) which automatically grows to meet the storage requirements. If this block is a star, it may access the present and past output within this buffer by making a special function call. This provides a convenient way of simulating systems requiring output feedback such as IIR filters.

When a block on the destination end of a connection is visited during execution, it derives its input from the input buffer. When a request for an input sample is made, a sample from the output FIFO buffer of the source block is shifted into the input buffer. Multiple requests for input causes the input buffer to grow automatically. If this block is a star, it may access the present and past input within this buffer by making a special function call. This provides a tapped-delay line feature which is useful for many applications such as FIR filtering.

As mentioned earlier, the implementation of star connections using FIFO's has the benefit of providing access to the present and past input and output samples which is typically required for many processing operations. It is convenient for the star programmer because no extra data structures need be defined. However, the primary advantages gained are execution efficiency and the ability of the star programmer to control data flow between stars. These advantages are made evident in the following sections.

B.1.2.2 THE STAR

The star is the most primitive block type. It is defined by a series of statements in a pseudo-C type language in a file with the extension ".s". The degree of processing performed by the star is at the discretion of the star author. The actual functionality of the star is defined by pure C statements and hence is limited only by those constraints imposed by the C language itself. Therefore, if desired,

one might create a star that implements a galaxy-type function such as an adaptive echo canceller. Such monolithic implementation of complex processing has the advantage of speed since there is less I/O "overhead" which is introduced with each block-to-block interconnection.

During the execution of a simulation, a star is typically "visited" several times. If there is input available, it may read in and process any number of samples up to the number available at the time of the "visitation". The number of samples read in and processed is controlled by the star programmer. In an ordinary situation, when a star is visited, the star reads in and processes all of the available samples and outputs the results to its output buffer. Each visitation incurs a certain amount of system overhead so the buffering idea is efficient in the sense that it requires the minimum number of visitations to the star to process all available samples flowing through the simulation. This is in contrast to the no-buffering situation where only 1 sample is read in, processed, and output per visitation. Clearly this amounts to the maximum system overhead being incurred and thus is the least efficient. However, in some cases it may be necessary for a star to process a limited amount of data per visitation. These cases arise when considering stars which have a feedback input or stars which must process data in chunks such as those which implement FFT functions. In section B.1.3, an example of one of these cases is discussed. It is sufficient to say at this moment

that the star programmer must exercise extreme caution in managing the throughput of the star in these cases.

B.1.2.2.1 THE STAR DEFINITION

The series of pseudo-C statements that define a star is divided into 9 sections. They are delimited with special star section delimiters of the form "section_name ... end". They are described below. "[" delimit optional statements. "<>" delimit required statements. "|" denotes a choice of arguments. Bold-faced words are reserved words in CAPSIM.

1) Comment Section

This section is delimited by standard C comment delimiters /* */. The star programmer typically provides the usual program header information and a description of each star parameter and its type. The comment section is of course optional.

2) Parameter Definition Section

The parameters of a star are defined in this section. They are typically the parameters of the processing algorithm implemented by the star. The value of a parameter can be changed by the user at run-time.

In the following sections, parameters are used like constants and are referenced by the parameter name.

This section can be omitted in a star with no parameters.

parameters

<parameter_type> <parameter_name>;

end

where,

**<parameter_type> float | int | file | function
 | array**

3) Input Buffer Definition Section

This section defines the input buffers used by the star. More than one input buffer can be defined. The optional **delay_min** and **delay_max** statements define the minimum and maximum number of input samples into the past that are accessible by the star. If unspecified, the default minimum delay is 0 which specifies the current input. Specifying a minimum delay value greater than 0 is an easy way of introducing a delayed input. The default for the maximum delay is the value of the minimum delay.

In the following sections, current and past input buffer elements are referenced in the form like that of a function call where the buffer name is the function name and a single integer argument, say *i*, references the *i*th data value input. *i* must be in the range from **delay_min** to **delay_max**. An argument value of 0 references the current input value.

This section is omitted in stars such as signal generators which do not require any input. It may also be omitted in cases where the number of input buffers is determined at run-time by the number of input connections made. This "auto-fan-in" capability is supported through the use of special function calls which allow access to a

particular element in a particular buffer. Another case arises when the type of the buffer elements is something other than the standard `float` or `int` types. Functions are supplied which allow the programmer to specify the size in bytes of a buffer element. In this case, it is the responsibility of the programmer to define the appropriate data structure "overlays" to access the data. This feature is particularly useful when simulating stars which perform vector or matrix operations where data must be passed from one star to the next in chunks.

```

input_buffers
    [delay_min = min_delay_value_1;]
    [delay_max = max_delay_value_1;]
    <input_buffer_1_type> <input_buffer_name>;
    ...
end
where,
    <input_buffer_1_type>      float | int

```

4) Output Buffer Definition Section

This section defines the output buffers used by the star. More than one output buffer can be defined. An optional `delay_max` statement is the maximum number of output samples in the past that can be accessed. The default value is 0 which indicates that only the current output is accessible. In the following sections, current

and past output buffer elements are accessed in the same way as input buffer elements. (see Input Buffer Definition Section).

This section can be omitted in stars which provide no output or determine the number of output buffers connected at run-time. This "auto-fan-out" capability is supported in an analogous way to the "auto-fan-in" feature. Like input buffers, output buffer types other than the standard int and float can be accommodated.

output_buffers

```
[delay_max = max_delay_value_1;]
```

```
<output_buffer_1_type>      <output_buffer_name>;
```

```
...
```

end

where,

```
<output_buffer_1_type>      float | int
```

5) Variable Declaration Section

This section contains declarations of local variables that do not have to retain their values from one visitation to the next during execution. They are typically used for storing intermediate results and loop variables. The statements can be any valid C variable declarations.

declarations

```
<valid_C_variable_type>      <valid_C_variable_name>
```

```
...
```

end

6) State Variable Declaration Section

This section contains declarations of valid C variables that retain their values from one visitation to the next during execution. They are analogous to C "static" variables. They are typically used in stars with memory elements such as the coefficients of adaptive filters, statistical counters which determine the total number of iterations performed during simulation, the state variables of finite state machines, and file pointer variables. An optional initial value may be assigned to these variables.

states

<valid_C_variable_type> <valid_C_variable_name>
[= <initial_value>]

...

end

7) Initialization Code Section

This section contains any valid C code that the star programmer wishes to be executed only once when the the star is visited for the first time. This is useful for parameter range checking, checking the number of input or output buffers connected for a auto-fan-in or auto-fan-out star, allocation of and initialization of data structures, and opening of files.

initialization_code

<valid C code>

end

8) Main Code Section

This section contains any valid C code that is executed during every visit to the star. The input, processing, and output functions of the star are implemented here. The processing functions are implemented with any valid C code. Special CAPSIM-supplied functions are used for accessing and controlling the input and output buffers. In this way, simulation code written in C for another simulation environment requires little modification for integration into the star code.

main_code

<valid C code>

end

9) Wrapup Code Section

This section contains any valid C code that the star programmer wishes to be executed once at the end of the simulation. This is typically used for deallocation of memory in data structures, output of final results and star statistics, and closing files.

Example

A simple example of a GAIN star's definition is given below to illustrate the usage of the some of the sections described above.

```

/* gain.s */
/*-----*/
GAIN - Single gain star
        This star applies gain to its single input and outputs the
        result.

Parameters:
        multiplier FLOAT value determining amount of gain
Inputs:
        x        single FLOAT input
        y        single FLOAT output

Date:      July 30, 1989
Programmer: Ken Yamazaki
Modifications: none
/*-----*/
parameters
        float      multiplier;
end

input_buffers
        float      x;
end

output_buffers
        float      y;
end

main_code
        while(it_in(0)) {
                it_out(0);
                y(0) = x(0) * multiplier;
        }
        return(0);
end

```

Fig. B.4 A sample star - gain.s.

The functions `it_in()` and `it_out()` are CAPSIM-supplied functions. In the example, `it_in(0)` checks whether there is any input data available for input buffer 0 or equivalently input buffer "x". If data is available, the input buffer is shifted by 1 and a data value is read into the buffer. `it_out(0)` shifts the output buffer 0, or equivalently the output buffer "y", and outputs the value assigned to it into the buffer. As can be seen, we are referencing only the current input (ie. `x(0)`) and the current output (ie. `y(0)`) in the processing. The use of the `it_in()` function in this example forces the star, upon visitation, to read in and processes every data value available at its input before terminating the visitation with the `"return(0)"` statement.

More elaborate star definitions, that use more or all of the sections mentioned, can be found in the star listings in Appendix B.3.

B.1.2.2.2 THE STAR LIBRARY

After its source file is "conceived", a "star is born" when it is formally integrated into the CAPSIM environment. The process is almost the same as that required to add a C subroutine to a library.

The processing of adding or removing a star from the CAPSIM environment is handled by a single CAPSIM-supplied star preprocessor called PRECAPSIM. PRECAPSIM adds a star to the environment by converting the pseudo-C code in the star's ".s" file into standard C code which is stored in an associated ".c" file. This file is then compiled, using a standard C compiler, into object code

and inserted into a special CAPSIM star library. The entire CAPSIM executable file is then created by linking this library to other system files. In some cases, stars from other CAPSIM installations may only be available in the standard ".c" or object format. In these cases, the appropriate steps are omitted in the integration process.

PRECAPSIM converts the pseudo-C code in a star's ".s" file into standard C code by invoking an updated version of the STARGAZE star preprocessor described in [Mes'86].

Commonly used functions such as adding, multiplying, and branch or data distribution points are available in a core star library that is supplied by the CCSP developers.

B.1.2.3 THE GALAXY

The galaxy is a group of interconnected blocks that, like an individual star, may have inputs, outputs, and parameters. They typically represent logical or physical groupings of processing functions within a simulation. Blocks can be both stars and other galaxies so many levels of hierarchy may be supported within a single simulation. Like stars, galaxies can be stored and retrieved after they are defined.

The inputs, outputs, and parameters are unique in that they relate to the group of blocks as a whole. When viewed at this level, the galaxy will look no different from a star. The inputs, outputs and parameters ultimately originate from stars that may be at any level within the galactic hierarchy. After the interconnection of blocks is

established, the galaxy inputs are specified by associating them with the input buffers of blocks contained within the galaxy. Similarly, galaxy outputs are specified by their association with the output buffers of internal blocks. Galactic parameters must also be associated with the parameters of its internal blocks.

There are many advantages of using galaxies. The first and most obvious is that the resulting hierarchy supports the natural way of understanding the system under simulation. The basic understanding of the simulation is attained at the highest level. Lower levels show increasingly more details of the implementation of the simulation until finally, at the star level, the user is confronted with the actual programming definitions.

Another hierarchy-related advantage is the inherent imposition of structured programming and debugging practices during the construction of the simulation. The construction of a moderately complex galaxy requires that the underlying stars be written and thoroughly tested before any interconnections are made. Alternatively, top-down design can be supported by the creation of a "null" star that does no processing but simply routes its input to its output. This is analogous to the "stub" in software terminology.

The final advantage to be mentioned here is not obvious but nonetheless very important for simulations involving feedback loops. As mentioned previously, the buffered I/O concept allows for the most efficient simulation as far as incurred system overhead is concerned. This statement is actually provisional on the ability of

the blocks to read in and process every piece of data available on all of its input buffers. In some cases, particularly in the simulation of adaptive systems, there will be some stars which cannot meet this criteria. Consider a system identification scenario illustrated below in which an adaptive filter must adapt its coefficients using an "error" term that is generated by the subtraction of the filter's output from an unknown system's output.

Fig. B.5 A system identification example illustrating the use of feedback.

The adaptive filter, "afilter", outputs an estimate of the system output based on the current values of its internal coefficients and the data derived from input buffer 0. The subtraction of the filter output from the "system" output is performed by the "wsum" or weighted sum star. The result is a feedback quantity that is required by the adaptive filter for adapting the coefficients for the next estimate output. We have a situation where there must be data available on both of the adaptive filter's inputs in order for it to perform the output and update functions.

The problems becomes evident when one considers the fact that the "wsum" star must be visited after the filter has made a single output in order to generate the feedback quantity necessary for the filter to update its coefficients for the next iteration. This strict order of execution limits the adaptive filter star to processing only 1 new input per visitation. Therefore the number of visitations required to process all of the samples from the data generator, and hence the system overhead, is maximized. Furthermore, this has consequences for both the data generator itself and the stars in the signal path following the filter.

In CAPSIM, all properly written source-type stars are limited to outputting a fixed number of samples per visitation, say n . After the data generator has output n samples, the adaptive filter will input and process only 1 of these before the "wsum" star is visited leaving $n-1$ unprocessed samples on its input buffer. After the "wsum" star has performed its single subtraction, it will find no other

samples available on its input buffer and therefore the plot star will be visited. The plot star will see only the result of the single subtraction on its input buffer and therefore terminate its visitation after processing that result. Therefore, like the adaptive filter, both stars are therefore limited to processing only one sample per visitation. However, the story is not yet complete. The visitation sequence repeats at this point and the data generator is visited. The data generator will output another n samples adding to the $n-1$ unprocessed samples left in the filter's input buffer from the previous visitation. Therefore we see that the adaptive filter's input buffer will grow due to the imbalance in the rate of data being supplied and consumed. If a large number of data is output by the data generator, the available memory will be eventually exhausted. The only remedy for this situation is to force the data generator to output only one sample per visitation. The data generator is effectively forced into the same inefficient mode of execution as the rest of the stars in the simulation.

A portion of these problems can be overcome by enclosing all stars within the feedback loop in a galaxy as shown below.

Fig. B.6 A more efficient version of Fig. B.5 using a galaxy.

Unlike stars, the number of times a galaxy reads in, processes, and outputs data per visitation is not under the control of the programmer. The rule for galaxies is rather familiar, that is, during a visitation to a galaxy, the galaxy will continue to execute until all of the data available on the galaxy input buffer has been exhausted. This is precisely the optimal case for star execution. In the new version of the system identification example, the inefficient execution of the stars within the "adapt" galaxy still remains, however, the galaxy as a whole will process all n data elements available on its input buffer and output the results to its output

buffer before the next block is visited. Therefore blocks in the signal path following the galaxy, namely the "plot" star, will see all of the n data elements it is entitled to see when it is visited. The data generator is no longer limited to producing a single output per visitation because, when it is visited again, all of the data it had output to the galaxy would have been consumed. Therefore the advantage of defining galaxies that enclose feedback loops is the localization of the execution inefficiency to the feedback loop itself leaving the rest of the simulation unsmeared.

B.1.2.4 THE UNIVERSE

The universe is the specification of all blocks, connections, and parameters that define the entire simulation. It cannot be termed the "grand-daddy" galaxy because unlike a galaxy, it has no open-ended input and output connections. All inputs into a universe must be supplied by source-type stars and all outputs must be connected to sink-type stars. Source-type stars are typically, wave generators, random number generators, and file readers. Sink-type stars are typically plotters and file writers. Like stars and galaxies, a universe can be stored and retrieved once it has been defined. Once the universe has been defined or loaded, the simulation can be "run".

B.1.3 SIMULATION EXECUTION

Simulations are executed when a universe is defined or loaded and a "run" command is issued to CAPSIM. The execution proceeds

in a straightforward manner, however, before this is explained, there are several preliminary steps involved that are worth noting.

As a universe or galaxy is created graphically, every edit made is translated into a more compact text form that can be stored. The names and parameters of the stars and galaxies brought into the graphical workspace are stored. The actual star code need not be loaded as this was made part of the actual CAPSIM executable file when the PRECAPSIM command was issued. The graphical connections between blocks are translated into a text "netlist" that contains the names of the interconnected blocks and their associated input and output buffer numbers. This compact text form is complete in that nothing more is required in order to run the simulation. Information regarding the actual graphic elements is stored separately and is only required by the graphical user-interface. When a universe or galaxy is saved, the text information is stored in a "topology" file and the graphical information is stored in a separate "graphics" file.

When a universe is loaded, both text and graphics files are loaded. The graphics file is interpreted and the graphical representation of the universe is displayed. All of the information necessary for the execution of the simulation is stored in the text files describing the universe and the galaxies. These are loaded and are interpreted "on the fly" by CAPSIM to internally configure the simulation.

The running of a simulation actually consists of 2 separate phases [Mes'84]. During the first phase, a scheduler determines the order in which the blocks within a simulation are to be executed. Blocks which supply data to the simulation are scheduled to execute first. Among these blocks, the actual scheduling order is chosen at random. Those blocks with all of their input buffers connected to those just previously scheduled are then scheduled to executed next and so on. The scheduler makes as many passes as necessary to schedule all of the blocks. In the case where there are still unscheduled blocks left that don't meet the scheduling criteria, the execution of an arbitrary block is forcibly scheduled and the scheduling is continued as before. When a scheduled block is a galaxy, the blocks contained within are subject to the same scheduling algorithm. When all blocks have been scheduled, the execution phase begins.

During the execution phase, the blocks are simply executed in the predetermined scheduling order. The condition that determines the actual termination of the simulation is termed "deadlock". In the optimum case, where all blocks are designed to process all samples on their input buffers during a visitation, the term "deadlock" would refer to the event when all blocks have either exhausted their input buffers or filled their output buffers to a predetermined limit. The latter case can occur when a block is connected to another block which cannot execute because it requires data on another input buffer which is exhausted. The term "deadlock" is actually more

general in scope because a star may be programmed to abstain from input buffer access and output buffer access regardless of the actual status of the input buffer and output buffer. The definition as given in [Mes'84] is "all fail to successfully access an input or output fifo".

B.2 SIMULATION TOPOLOGIES

A "topology" in CAPSIM terminology is a generic term that refers to a description of blocks and their interconnections. Therefore both galaxies and the universe are referred to as topologies. The text topology files associated with both galaxies and universes have the extension ".t".

The following are block diagrams of the important universes and galaxies created for the simulation of the Jointly-Adaptive and Sequentially-Adaptive TEC/MEC structures that were investigated in this thesis. Each is followed by a corresponding universe or galaxy topology text file generated from the block diagram by CAPSIM.

Note that the TEC/MEC galaxies contain extra blocks, namely, the IREC and DCTP. These implement a dual-pole adaptive gain IIR echo canceller and DC canceller respectively and are not used here. In the universe topologies, the CONTRL block is used to activate and deactivate the appropriate echo cancellers within the echo canceller galaxies.

B.2.1 THE JOINTLY-ADAPTIVE TEC/MEC UNIVERSE

The following is a block diagram of the Jointly-Adaptive TEC/MEC universe.

Fig. B.7 The Jointly-Adaptive TEC/MEC universe, INTEC.

The universe shown in Fig. B.7 is fairly straightforward. BITGEN generates a random stream of binary 1's and 0's. LNCODE converts the symbols into a stream of binary ± 1 symbols. CONTRL commands the echo canceller galaxy, ECBIN, to operate in a coefficient update and output mode. The output of the WSUM block is the cancellation error which is fed back to ECBIN for coefficient adaptation. The error is also used in the calculation of an approximation of the AMSE. The error is squared by EXPON and filtered by FIR which is configured as a simple moving average function spanning 10 samples. The smoothed result is then expressed in terms of dB's by LOGFN and plotted by TPLOT and written to a file by PRFILE. The results were compared against another topology that more properly produced an average from an ensemble of raw MSE curves. The moving average method was found to be an adequate alternative to the more time consuming ensemble average method.

The following is the text topology file, JNTEC.T, that is associated with the Jointly-Adaptive TEC/MEC universe shown above.

```
# jntec.t
param array 3 -1 1 1
star 10664 sum

param array 1 12
param array 1 3001
param int 1
star 11967 control

star 13012 fork

param int 0
param int 1
param int 0
star 13486 linecoder

star 13541 drain

param array 3 0 0 0
param int 30
param int 0
param float 1.670000e-02
param float 1.670000e-02
param file nofile
param array 1 5
param array 1 0
param float 0.5
param file nofile
param float 0
param float 0
param float 0
param file nofile
param float 0
param float 0
param file nofile
galaxy 13620 ecbin.t
```

param float 2
star 14292 expon

param int 3001
param float 12
param int 1
star 14597 bit_genf

param int 6
param file eprom4.cof
star 14688 eprom

param int 31
param file fir14.cof
star 14798 fir1

param array 10 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
star 14833 fir

param float 10
star 14994 logfn

param int 3000
param float 1
param int 0
param file plot
param file x
param file y
param int 0
star 15052 tplot

star 15112 fork

param file jnt3053.mse
star 15317 printfile

connect 10664 0 13620 1
connect 10664 1 14292 0
connect 11967 0 13620 2

```
connect 13012 0 14798 0
connect 13012 1 14688 0
connect 13012 2 13620 0
connect 13486 0 13541 0
connect 13486 1 13012 0
connect 13620 0 10664 0
connect 14292 0 14833 0
connect 14597 0 13486 0
connect 14688 0 10664 1
connect 14798 0 10664 2
connect 14833 0 14994 0
connect 14994 0 15112 0
connect 15112 0 15052 0
connect 15112 1 15317 0
```

Fig. B.8 The Jointly-Adaptive TEC/MEC universe topology file, JNTEC.T.

B.2.1.1 THE JOINTLY-ADAPTIVE TEC/MEC GALAXY

The following is a block diagram of the Jointly-Adaptive TEC/MEC galaxy, ECBIN.

Fig. B.9 The Jointly-Adaptive TEC/MEC galaxy, ECBIN.

The following is the text topology file, ECBIN.T, that is associated with the Jointly-Adaptive TEC/MEC galaxy shown above.

```
# ecbin.t
star 10447 ecc

star 10612 fork

param arg 1
param arg 2
param arg 3
param arg 4
param arg 5
star 10779 tec

param arg 10
param arg 11
param arg 12
param arg 13
star 10977 irec

param arg 14
param arg 15
param arg 16
star 11065 dctp

star 11110 add

param arg 0
star 11391 shiftreg

param arg 6
param arg 7
param arg 8
param arg 9
star 11494 mecbin

connect input 2 10447 0
connect input 1 10612 0
```

```

connect input 0 11391 0
connect 10447 0 10779 2
connect 10447 1 11494 2
connect 10447 2 10977 2
connect 10447 3 11065 1
connect 10447 4 11391 1
connect 10612 0 10779 1
connect 10612 1 11494 1
connect 10612 2 10977 1
connect 10612 3 11065 0
connect 10779 0 11110 0
connect 10977 0 11110 2
connect 11065 0 11110 3
connect 11391 0 10779 0
connect 11391 1 11494 0
connect 11391 2 10977 0
connect 11494 0 11110 1
connect 11110 0 output 0

```

Fig. B.10 The Jointly-Adaptive TEC/MEC galaxy topology file, ECBIN.T.

B.2.2 THE SEQUENTIALLY-ADAPTIVE TEC/MEC UNIVERSE

The Sequentially-Adaptive TEC/MEC universe block diagram is identical to that of the Jointly-Adaptive TEC/MEC universe (see Fig. B.7). The parameters of the universe are slightly different in that 2 distinct echo cancellation phases are implemented by the CONTRL block.

The following is the text topology file, SEQEC.T, that is associated with the Sequentially-Adaptive TEC/MEC universe.

```
# seqec.t
param array 3 -1 1 1
star 10664 sum

param array 2 24 25
param array 2 1000 2001
param int 1
star 11967 control

star 13012 fork

param int 0
param int 1
param int 0
star 13486 linecoder

star 13541 drain

param int 3001
param float 12
param int 1
star 14597 bit_genf

param int 6
param file eprom4.cof
star 14688 eprom

param int 31
param file firl4.cof
star 14798 firl

param array 3 0 0 5
param int 5
param int 0
param float 7.000000e-02
param float 7.000000e-02
```

param file nofile
param array 1 5
param array 1 0
param float 0.27
param file nofile
param int 25
param int 0
param float 7.000000e-02
param float 7.000000e-02
param file nofile
param float 0
param float 0
param file nofile
galaxy 14820 ecbin1.t

param float 2
star 15092 expon

param array 10 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
star 15233 fir

param float 10
star 15394 logfn

param int 3000
param float 1
param int 0
param file plot
param file x
param file y
param int 0
star 15452 tplot

star 15512 fork

param file sqncv2ex.mse
star 15817 printfile

connect 10664 0 14820 1
connect 10664 1 15092 0

```

connect 11967 0 14820 2
connect 13012 0 14798 0
connect 13012 1 14688 0
connect 13012 2 14820 0
connect 13486 0 13541 0
connect 13486 1 13012 0
connect 14597 0 13486 0
connect 14688 0 10664 1
connect 14798 0 10664 2
connect 14820 0 10664 0
connect 15092 0 15233 0
connect 15233 0 15394 0
connect 15394 0 15512 0
connect 15512 0 15452 0
connect 15512 1 15817 0

```

Fig. B.11 The Sequentially-Adaptive TEC/MEC Universe topology file, SEQECT.

B.2.2.1 THE SEQUENTIALLY-ADAPTIVE TEC/MEC GALAXY

The following is a block diagram of the Sequentially-Adaptive TEC/MEC galaxy, ECBIN1.

Fig. B.12 The Sequentially-Adaptive TEC/MEC Galaxy, ECBIN1.

The following is the text topology file, ECBIN1.T, associated with the Sequentially-Adaptive TEC/MEC galaxy shown above.

```
# ecbin1.t
star 10447 ecc

star 10612 fork

param arg 1
param arg 2
param arg 3
param arg 4
param arg 5
star 10779 tec

param arg 15
param arg 16
param arg 17
star 11065 dctp

star 11110 add

param arg 0
star 11391 shiftreg

param arg 6
param arg 7
param arg 8
param arg 9
star 11494 mecbn

param arg 10
param arg 11
param arg 12
param arg 13
param arg 14
star 11571 tec

connect input 2 10447 0
```



```

connect input 1 10612 0
connect input 0 11391 0
connect 10447 0 10779 2
connect 10447 1 11494 2
connect 10447 2 11571 2
connect 10447 3 11065 1
connect 10447 4 11391 1
connect 10612 0 10779 1
connect 10612 1 11494 1
connect 10612 2 11571 1
connect 10612 3 11065 0
connect 10779 0 11110 0
connect 11065 0 11110 3
connect 11391 0 10779 0
connect 11391 1 11494 0
connect 11391 2 11571 0
connect 11494 0 11110 1
connect 11571 0 11110 2
connect 11110 0 output 0

```

Fig. B.13 The Sequentially-Adaptive TEC/MEC Galaxy topology file, ECBIN1.T.

B.3 STAR SOURCE FILES

The following is an alphabetical listing of important CAPSIM "stars" or source code created for the simulation of the Jointly-Adaptive and Sequentially-Adaptive TEC/MEC structures.

```
/* control.s */
```

```
/******
```

Star: CONTROL - Multi-State Control Generator

Description:

This star outputs a succession of user-defined control values. Up to 10 different control values and the number of repetitions of each can be specified.

Parameters:

CTLVAL = Array parameter containing up to 10 control values

REPETITIONS = Array parameter containing number of times each control value is output. The 1st entry corresponds to the 1st control value, the 2nd entry corresponds to the 2nd control value and so on.

Note: The size of the CONTROL and REPETITIONS array must be identical.

NUMPERSCHED = Number of values output each time this star is scheduled.

Input: None

Output: Y = Control output

Programmer: Ken Yamazaki

Date : Nov. 27, 1988

Version : 1.0

Modified :

```
*****/
```

parameters

array ctlval;

array repetitions;

int numpersched = 1;

end

```

states
    int count = 0;
    int ctl_count = 0;
end

output_buffers
    int y;
end

declarations
    int i;
end

initialization_code
    /* check that the number of both array parameters match */
    if (n_ctlval != n_repetitions) {
        fprintf(stderr,
            "control: control and repetitions array sizes must match\n");
        return(1);
    }

    /* check that there is at least 1 control value */
    if (n_ctlval < 1) {
        fprintf(stderr,
            "control: at least 1 control value must be specified\n");
        return(2);
    }

    /* check that all repetition values are at least 1 */
    for (i = 0; i < n_repetitions; i++)
        if (repetitions[i] < 1.0) {
            fprintf(stderr,
                "control: number of repetitions must be at least 1\n");
            return(3);
        }

    /* check that the number of values output per schedule is valid */
    if (numpersched < 1) {
        fprintf(stderr,
            "control: at least 1 value must be output per schedule\n");
    }

```

```

        return(4);
    }
end

main_code
    for (i = 0; i < numpersched; i++)
        if (ctl_count < n_ctlval)
            if (count++ < (int)repetitions[ctl_count]) {
                it_out(0);
                y(0) = (int)ctlval[ctl_count];
            }
            else {
                count = 0;
                ctl_count++;
            }
        return(0);
    end

```

Fig. B.14 Source code for the CONTROL star.

/* ecc.s */

/*****

Star: ECC - Echo Cancellor Controller

Description:

This star is used exclusively in the galaxy EC. Its function is to receive a EC control word from the FSM and send the appropriate mode control signals to the other EC blocks.

Parameters: None

Input 0: EC_CTL = EC control word from the FSM

- 0: Enter the OFF state
 Stop update of all EC stages
 EC_OUT = 0
- 1: Enter the FAST state
 Normal operation of all EC stages
 TEC_BETA = TEC_BETA1
 DC_BETA = DC_BETA1
- 2: Enter the SLOW state
 Normal operation of all EC stages
 TEC_BETA = TEC_BETA2
 DC_BETA = DC_BETA2
- 3: Enter the RESET state
 Reset memory of all EC stages to 0
- 4: Enter the FROZEN state
 Calculate EC_OUT without any updating
- 5: Enter the TEC only FAST state
- 6: Enter the TEC only SLOW state
- 7: Enter the TEC only FROZEN state

- 8: Enter the MEC only state
- 9: Enter the MEC only FROZEN state
- 10: Enter the IREC only state
- 11: Enter the IREC only FROZEN state
- 12: Enter the TEC FAST and MEC only state
- 13: Enter the TEC SLOW and MEC only state
- 14: Enter the TEC FROZEN and MEC only state
- 15: Enter the TEC FAST and MEC FROZEN only state
- 16: Enter the TEC SLOW and MEC FROZEN only state
- 17: Enter the TEC FROZEN and MEC FROZEN only state
- 18: Enter the DCTP only FAST state
- 19: Enter the DCTP only SLOW state
- 20: Enter the DCTP only FROZEN state
- 21: Enter the DCTP FAST and TEC FAST only state
- 22: Enter the DCTP SLOW AND TEC SLOW only state
- 23: Enter the DCTP FROZEN and TEC FROZEN state

*** NOTE ***

The following modes assume that another TEC is used in place of the IREC in the EC galaxy.

- 24: Enter the TEC 1 FAST and TEC 2 FAST ONLY state
- 25: Enter the TEC 1 FROZEN, TEC 2 SLOW and MEC ON ONLY state

Output 0: TEC_CTL = Control word output to the TEC

Bit 0: TEC RESET	= 0	No reset
	= 1	$C_i(k) = 0$ for all i $ECT(k) = 0$
Bit 1: TEC ON/OFF	= 0	$ECT(k) = 0$ No update of $C_i(k)$
	= 1	$ECT(k) = \text{echo estimate}$
Bit 2: TEC UPDATE	= 0	No update of $C_i(k)$
	= 1	Update $C_i(k)$
Bit 3: TEC SPEED	= 0	Use $TEC_BETA = TEC_BETA1$
	= 1	Use $TEC_BETA = TEC_BETA2$

Output 1: MEC_CTL = Control word output to the MEC

Bit 0: MEC RESET	= 0	No reset
	= 1	All memory in all stages are reset $ECM_SUM(k) = 0$
Bit 1: MEC ON/OFF	= 0	$ECM_SUM(k) = 0$ No update of $ECM_i(k)$'s
	= 1	$ECM_SUM(k) = \text{echo estimate}$
Bit 2: MEC UPDATE	= 0	No update of $ECM_i(k)$'s
	= 1	Update $ECM_i(k)$'s

Output 2: IREC_CTL = Control word output to the IREC

Bit 0: IREC RESET	= 0	No reset
	= 1	$ECI1_H(k) = 0$
		$ECI2_H(k) = 0$
		$ECI(k) = 0$
Bit 1: IREC ON/OFF	= 0	$ECI(k) = 0$ No update of $ECI_j_H(k)$
	= 1	$ECI(k) = \text{echo estimate}$

Bit 2: IREC UPDATE = 0 No update of $ECIj_H(k)$
 = 1 Update $ECIj_H(k)$

Output 3: DCTP_CTL = Control word output to the DCTP

Bit 0: DCTP RESET = 0 No reset
 = 1 $DC(k) = 0$

Bit 1: DCTP ON/OFF = 0 $DC(k) = 0$
 No update of $DC(k)$
 = 1 $DC(k) =$ latest update of $DC(k)$

Bit 2: DCTP UPDATE = 0 No update of $DC(k)$
 = 1 Update $DC(k)$

Bit 3: DCTP SPEED = 0 Use $DC_BETA = DC_BETA1$
 = 1 Use $DC_BETA = DC_BETA2$

Output 4: ECSR_CTL = Control word output to the ECSR

Bit 0: ECSR RESET = 0 No reset
 = 1 Reset all ECSR memory to 0
 All ECSR outputs are 0

Functionality:

The block accesses a control table which essentially maps an EC control word from the FSM to several control words that are output to the blocks of the EC.

Control Table Contents:

Input	Output				
EC_CTL	TEC_CTL	MEC_CTL	IREC_CTL	DCTP_CTL	ECSR_CTL
0	xx00	x00	x00	xx00	0
1	0110	110	110	0110	0
2	1110	110	110	1110	0
3	xxx1	xx1	xx1	xxx1	1
4	x010	010	010	x010	0
5	0110	x00	x00	xx00	0
6	1110	x00	x00	xx00	0
7	x010	x00	x00	xx00	0
8	xx00	110	x00	xx00	0
9	xx00	010	x00	xx00	0
10	xx00	x00	110	xx00	0
11	xx00	x00	010	xx00	0
12	0110	110	x00	xx00	0
13	1110	110	x00	xx00	0
14	x010	110	x00	xx00	0
15	0110	010	x00	xx00	0
16	1110	010	x00	xx00	0
17	x010	010	x00	xx00	0
18	xx00	x00	x00	0110	0
19	xx00	x00	x00	1110	0
20	xx00	x00	x00	x010	0
21	0110	x00	x00	0110	0
22	1110	x00	x00	1110	0
23	x010	x00	x00	x010	0
24	0110	x00	0110	xx00	0
25	x010	110	1110	xx00	0

Note: User-defined modes are created by adding another row entry into the table

Programmer: Ken Yamazaki

Date: July 17, 1988

Version: 1.0

Modified: June 12, 1989 Ken Yamazaki: Added modes 24 and 25

*****/

```

/* definitions */
/* The following definitions contain the number of EC modes and the
number of devices controlled by the controller respectively. If a
column is added to the CONTROL TABLE then NUM_DEV must be
incremented. If a row is added, NUM_MODES must be incremented.
*/
#define num_modes 26
#define num_dev 5

input_buffers
    int ec_ctl;
end

output_buffers
    int tec_ctl;
    int mec_ctl;
    int irec_ctl;
    int dctp_ctl;
    int ecsr_ctl;
end

declarations
    int k;
    static int control_table[num_modes][num_dev] =
    {
        {0, 0, 0, 0, 0},
        {6, 6, 6, 6, 0},
        {14, 6, 6, 14, 0},
        {1, 1, 1, 1, 1},
        {2, 2, 2, 2, 0},
        {6, 0, 0, 0, 0},
        {14, 0, 0, 0, 0},
        {2, 0, 0, 0, 0},
        {0, 6, 0, 0, 0},
        {0, 2, 0, 0, 0},
        {0, 0, 6, 0, 0},
        {0, 0, 2, 0, 0},
        {6, 6, 0, 0, 0},
        {14, 6, 0, 0, 0},
        {2, 6, 0, 0, 0},
    }

```

```

        {6, 2, 0, 0, 0},
        {14, 2, 0, 6, 0},
        {2, 2, 0, 0, 0},
        {0, 0, 0, 6, 0},
        {0, 0, 0, 14, 0},
        {0, 0, 0, 2, 0},
        {6, 0, 0, 6, 0},
        {14, 0, 0, 14, 0},
        {2, 0, 0, 2, 0},
        {6, 0, 6, 0, 0},
        {2, 6, 14, 0, 0}
    };

end

main_code
    for (k = min_avail(); k > 0; k--) {
        it_in(0);

        it_out(0);
        it_out(1);
        it_out(2);
        it_out(3);
        it_out(4);

        if (ec_ctl(0) < 0 || ec_ctl(0) > num_modes - 1) {
            fprintf(stderr,
                "ecc: unknown ec mode encountered\n");
            return(1);
        }

        /* access lookup table to find control words to send out */

        tec_ctl(0) = control_table[ec_ctl(0)][0];
        mec_ctl(0) = control_table[ec_ctl(0)][1];
        irec_ctl(0) = control_table[ec_ctl(0)][2];
        dctp_ctl(0) = control_table[ec_ctl(0)][3];
        ecsr_ctl(0) = control_table[ec_ctl(0)][4];
    }
end

```

Fig. B.15 Source code for the ECC star.

/* eprom.s */

/******

Star: EPROM - Erasable Programmable Read Only Memory

Description:

This star implements a EPROM with binary address lines. Binary +1 and -1 symbols are assumed. The size of the memory is determined by the size of the address shift register specified by the user (size = 2^{SRSIZE}). The input acts as the address shift register. The contents of the input shift register addresses the associated location in the memory. Its contents is then output. The memory is "programmed" via a user-specified file which contains the values which are to be inserted into the memory. The number of values in the file must be at least equal to the size of the memory. This star is suitable for testing binary memory echo cancellers with relatively small memory configurations.

Parameters:

SRSIZE = Shift Register size
(memory size = 2^{SRSIZE})

MEM_FILE = Name of the file containing values to be
inserted into the memory.

Input 0: X = Input shift register

Output 0: Y = Output value from memory

Functionality:

Y = CONTENTS OF LOCATION ADDRESS BY
(X(0),X(1),...,X(SRSIZE-1))

Address is generated from X(0),X(1),...,X(SRSIZE-1) as shown in the following example:

eg. SRSIZE = 4

	X(0)	X(1)	X(2)	X(3)
SYMBOLS	- 1	+1	- 1	+1
ADDRESS	0	1	0	1

$$\begin{aligned}\text{Address} &= 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 \\ &= 10\end{aligned}$$

The first value in the file will be inserted into memory location 0, the 2nd value into memory location 1,...etc.

Programmer: Ken Yamazaki
 Date: Jan. 6, 1989
 Version: 1.0
 Modified:

```

*****/
parameters
    int srsz = 1;
    file mem_file = "mem.cof";
end

states
    float* memory;
    FILE* fp;
end

input_buffers
    float x;
end

output_buffers
    float y;
end

declarations
    int i, memsize, addr;
    FILE *fopen();
end

initialization_code

```

```

    /* check that the shift register size is valid */
    if (srsz < 1) {
        fprintf(stderr,
            "eprom: shift register size must be at least 1\n");
        return(1);
    }

    /* set the size of the input shift register */
    set_dmax_in (0, srsz - 1);

    /* allocate space for EPROM memory and initialize it */
    memsize = 1 << srsz;
    if ((memory = (float *)calloc(memsize,4)) == NULL) {
        fprintf(stderr,
            "eprom: couldn't allocate EPROM memory\n");
        return(2);
    }
    for (i = 0; i < memsize; i++)
        memory[i] = 0.0;

    /* open memory file */
    if ((fp = fopen(mem_file,"r")) == NULL) {
        fprintf(stderr,
            "eprom: couldn't open memory file\n");
        return(3);
    }

    /* read memory file into EPROM memory */
    for (i=0; i < memsize; i++)
        fscanf(fp,"%f",&memory[i]);
end

main_code
while (it_in(0)) {
    /* calculate address from contents of shift register */
    addr = 0;
    for (i = 0; i < srsz; i++)
        addr += (x(i) < 0.0 ? x(i) + 1.0 : x(i)) * (1 << i);

    /* output contents of addressed location */

```

```
        it_out(0);  
        y(0) = memory[addr];  
    }  
    return(0);  
end
```

Fig. B.16 Source code for the EPROM star.

/* firl.s */

Star: FIRL - Finite Impulse Response Filter
(long version for spans > 10)

Description: This star implements a FIR filter of user-defined length.

Parameters:

NUM_COEFF = No. of coefficients.

where,

NUM_COEFF must be $\leq$ actual number of coefficients in the file. If the number of coefficients in the file is $>$ **NUM_COEFF**, then only the first **NUM_COEFF** coefficients will be read and used.

COEFF_FILE = Name of file containing coefficients
(DEFAULT = STDIN)

Coefficients must be stored in the following form:

A_0
 A_1
 A_2
 $\dots$
 A_{n-1}

Input 0: X = symbols to be filtered

Output 0: **FIR_OUT** = filtered symbols

Functionality: $\text{FIR_OUT}(k) = A_0 \cdot X(k) + A_1 \cdot X(k-1) + \dots + A_{n-1} \cdot X(k-n+1)$

Programmer: Ken Yamazaki

Date: June 5, 1989

Version: 1.0

Modified:

*******/**


```

parameters
    int num_coeff;
    file coeff_file = "stdin";
end

states
    FILE* fp;
    float* coeff;
end

input_buffers
    float x;
end

output_buffers
    float fir_out;
end

declarations
    int i;
    float y;
    FILE *fopen();
end

initialization_code
    /* check for file */
    if (strcmp(coeff_file,"stdin") == 0)
        fp = stdin;
    else if ((fp = fopen(coeff_file,"r")) == NULL){
        fprintf(stderr,"firl: cannot open coefficient file\n");
        return(1);
    }

    /* check that number of coefficients is a valid quantity */
    if (num_coeff < 1) {
        fprintf(stderr,"firl: there must be at least 1 coefficient\n");
        return(4);
    }

```

```

    /* allocate coefficient space */
    if ((coeff = (float *)calloc(num_coeff,4)) == NULL) {
        fprintf(stderr,"firl: cannot allocate coefficient space\n");
        return(2);
    }

    /* read coefficients into coefficient space and count them */
    for (i = 0; (i < num_coeff) && (fscanf(fp, "%f", &coeff[i]) != EOF);
i++);
    if (i != num_coeff) {
        fprintf(stderr,"firl: incorrect number of coefficients found\n");
        /* free up coefficient space */
        free((char *)coeff);
        return(3);
    }

    /* set the maximum number of accessible past input sample */
    set_dmax_in (0, num_coeff - 1);
end

main_code
    while (it_in(0)) {
        y = 0;
        for (i = 0; i < num_coeff; i++)
            y = y + coeff[i] * x(i);
        it_out(0);
        fir_out(0) = y;
    }
    return(0);
end

wrapup_code
    /* close coefficient file if it was a physical file */
    if (fp != stdin)

```

```
    fclose(fp);

    /* free up coefficient memory */
    free((char *)coeff);
    return(0);
end
```

Fig. B.17 Source code for the FIRL star.

```
/* mecbin.s */
```

```
/******
```

Star: MECBIN - Memory Echo Cancellor (Binary Version)

Description:

This star implements a memory echo canceller for binary +1 and -1 symbols. The no. of taps can be specified. A staggered memory architecture can be created by specifying the size of each stage and the delay of each stage. Up to 10 stages can be accomodated.

THIS MEC CAN HANDLE ONLY +1 AND -1 SYMBOLS. USING ANY OTHER SYMBOLS MAY CAUSE MEMORY ACCESS VIOLATIONS.

* Note *

The taps are updated first followed by the calculation of the output. This allows both operations to be performed each time the star is scheduled thus increasing the simulation efficiency. Initially only an output operation is performed.

Parameters:

STAGE_SIZE[] = Array parameter containing the size of each stage

DELAY[] = Array parameter containing the delay of each stage

MEC_BETA = MEC adaptation gain

Input 0: DTX = Input data

Input 1: ERR = Error

Input 2: MEC_CTL = Control word for the MEC

Bit 0: MEC RESET	= 0	No reset
	= 1	All memory in all stages are reset
		ECM_SUM(k) = 0

Bit 1: MEC ON/OFF	= 0	ECM_SUM(k) = 0 No update of ECMi(k)'s
	= 1	ECM_SUM(k) = echo estimate

Bit 2: MEC UPDATE = 0 No update of ECMi(k)'s
 = 1 Update ECMi(k)'s

Output 0: ECM_SUM = echo estimate

Functionality:

Calculate

$$\text{ECM_SUM}(k) = \text{sum over all } i \{ \text{ECM}_i(k) \}$$

Update

$$ECM_j(k+1) = ECM_j(k) + MEC_BETA * ERR(k)$$

where,

i = 0,..no. of MEC stages - 1

$ECM_i(k)$ refers to the contents of a single memory location within MEC stage i . The location referenced depends on the address vector that is formed from the symbols input into each stage.

The address vector for each stage is formed from the symbols contained by that portion of the input "shift register" that is spanned by that stage. A convenient mapping exists between the symbols and the actual address vector elements:

-1 -> address vector element 0

+1 -> address vector element 1

This is convenient in that vector elements are identical to the symbols when the symbols are positive and are 1 greater when the symbols are negative.

The actual address within the stage is calculated by interpreting the address vector as a binary value.

dtx()	+1	- 1	+1	+1	- 1
address	1	0	1	1	0

The address is:

$$\text{ADDRESS} = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4$$

Programmer: Ken Yamazaki

Date: July 26, 1988

Version: 1.0

Modified: Sept. 12, 1988 Changed order of operations so that the taps are updated first followed by the output.

Jan. 6, 1988 Modified a copy of MEC.S to handle only BINARY +1 and -1 values creating MECBIN.S. Added ability to dump MEC coefficients to a file.

*****/

/* include files */

#include "nova.h"

#include <math.h>

parameters

array stage_size;

array delay;

float mec_beta;

file coeff_file = "nofile";

end

states

float** ecm;

int* addr;

int first = 1;

FILE* fp;

int dump_coeff = FALSE;

end

4

of/de

4

Metrol

```

input_buffers
    float dtx;
    float err;
    int mec_ctl;
end

output_buffers
    float ecm_sum;
end

declarations
    int i, j, m, addr_symbol;
    float sum;
    FILE *fopen();
end

initialization_code
    /* check that number of size and delay parameters match */
    if (n_stage_size != n_delay) {
        fprintf(stderr,
            "mecbin: number of size and delay parameters do not
match\n");
        return(1);
    }

    /* check size and delay array parameters and find the maximum
    delay value
    */
    m = 0;
    for (i = 0; i < n_stage_size; i++) {
        if ((int)stage_size[i] < 1) {
            fprintf(stderr,
                "mecbin: an invalid size was specified\n");
            return(2);
        }
        if ((int)delay[i] < 0) {
            fprintf(stderr,
                "mecbin: an invalid delay was specified\n");
            return(3);
        }
    }

```



```

        if ((int)delay[i] + (int)stage_size[i] - 1 > m)
            m = (int)delay[i] + (int)stage_size[i] - 1;
    }

    /* set the maximum delay relative to the current input sample to
       the maximum delay value
    */

    set_dmax_in(0, m);

    /* allocate coefficient space for MEC stages */
    if ( (ecm = (float **)calloc(n_stage_size,4)) == NULL) {
        fprintf(stderr,
            "mecbin: can't allocate coefficient space\n");
        return(4);
    }
    for (i = 0; i < n_stage_size; i++)
        if ( (ecm[i] = (float *)calloc((int)pow(2.0,stage_size[i]),4))
            == NULL) {
            fprintf(stderr,
                "mecbin: can't allocate coefficient space\n");
            return(5);
        }

    /* allocate space for address array */
    if ( (addr = (int *)calloc(n_stage_size,4)) == NULL) {
        fprintf(stderr,
            "mecbin: can't allocate address storage space\n");
        return(6);
    }

    /* initialize the coefficient space and the address array*/
    for (i = 0; i < n_stage_size; i++) {
        addr[i] = 0;
        m = pow(2.0,stage_size[i]);
        for (j = 0; j < m; j++)
            ecm[i][j] = 0.0;
    }

    /* get the coefficient file name is specified */

```

```

    if (strcmp(coeff_file,"stdout") == 0) {
        fp = stdout;
        dump_coeff = TRUE;
    }
    else
        if (strcmp(coeff_file,"nofile") != 0) {
            if ((fp = fopen(coeff_file,"w")) == NULL) {
                fprintf(stderr,
                    "mecbin: couldn't open final coefficients file\n");
                return(7);
            }
            dump_coeff = TRUE;
        }
    }
end

main_code
    if (first) {
        if ((avail(0) > 0) && (avail(2) > 0)) {
            it_in(0);
            it_in(2);

            it_out(0);
            ecm_sum(0) = 0.0;

            first = 0;
        }
        return(0);
    }

    if (min_avail() > 0) {
        /* UPDATE */
        it_in(1);
        if ((mec_ctl(0) & bit1) && (mec_ctl(0) & bit2))
            /* MEC UPDATE ON */
            for (i = 0; i < n_stage_size; i++)
                ecm[i][addr[i]] += mec_beta * err(0);

        /* CALCULATE */
        it_in(0);
        it_in(2);
    }

```

```

    it_out(0);
    sum = 0.0;

    /* check the bits of the control word */
    if (mec_ctl(0) & bit0)
        /* MEC RESET */
        for (i = 0; i < n_stage_size; i++) {
            m = pow(2.0, stage_size[i]);
            for (j = 0; j < m; j++)
                ecm[i][j] = 0.0;
        }
    else
        if (mec_ctl(0) & bit1) {
            /* MEC ON */
            /* calculate echo estimate */
            for (i = 0; i < n_stage_size; i++) {
                /* calculate the address in memory i
                 (ie. addr[i] in ecm[i])
                */
                addr[i] = 0;
                for (j = 0; j < (int)stage_size[i]; j++) {
                    addr_symbol = (dtx((int)delay[i]+j) < 0.0) ?
                        dtx((int)delay[i]+j) + 1.0
                        : dtx((int)delay[i]+j);
                    addr[i] += addr_symbol * pow(2.0, j);
                }
                sum += ecm[i][addr[i]];
            }
        }
    ecm_sum(0) = sum;
}

return(0);
end

wrapup_code
/* dump final coefficients if a file was specified */
if (dump_coeff) {
    for (i = 0; i < n_stage_size; i++) {
        m = pow(2.0, stage_size[i]);

```

```

        fprintf(fp,"MEC STAGE %d  SIZE = %d\n",i,m);
        for (j = 0; j < m; j++)
            fprintf(fp,"%f\n",ecm[i][j]);
        fprintf(fp,"\n");
    }
    if (fp != stdout)
        fclose(fp);
}

/* free up memory */
for (i = 0; i < n_stage_size; i++)
    free((char *)ecm[i]);
free((char *)ecm);
free((char *)addr);
end

```

Fig. B.18 Source code for the MECBIN star.

/* shiftreg.s */

Star: SHIFTREG - Shift Register

Description:

This star acts as a shift register. Data can be tapped off from any position in the shift register and routed to an output. Up to 10 taps can be accommodated and routed to their respective outputs. The output has an auto-fanout capability. If a 2nd input is connected, it is treated as a control input.

Parameters:

DELAY[] = Array parameter containing delays of the outputs

DELAY[0] corresponds to delay of output 0

DELAY[1] corresponds to delay of output 1

...

etc.

Input 0: X = Input data

Input 1: SR_CTL = SR control word (Optional)

Bit 0: SR RESET	= 0	No Reset
	= 1	Reset all SR memory to 0
		All SR outputs are 0

Outputs: Outputs of various delays

Programmer: Ken Yamazaki

Date: July 12, 1988

Version: 1.0

Modified: Dec. 22, 1988 Ken Yamazaki. Added a no-control ability. If no control input is connected, then no control is assumed.

*******/**

parameters

```

        array delay;
end

states
    int num_in_buffers;
    int max_delay;
end

declarations
    int i, j;
end

initialization_code
    /* check the number of inputs connected */
    if ((num_in_buffers = no_input_buffers()) < 1 ||
        no_input_buffers() > 2) {
        fprintf(stderr,
            "shiftreg: there must be 1 or 2 input buffers connected\n");
        return(1);
    }

    if (no_output_buffers() != n_delay) {
        fprintf(stderr,
            "shiftreg: no. of delay values and outputs do not match\n");
        return(2);
    }

    /* check for invalid delay values and find the maximum delay
       value */

    max_delay = 0;
    for (i = 0; i < n_delay; i++) {
        if (delay[i] < 0) {
            fprintf(stderr,
                "shiftreg: negative delay values not permitted\n");
            return(3);
        }
        if (delay[i] > max_delay)
            max_delay = delay[i];
    }

```

```

        /* set the maximum delay relative to the current input sample to
           the maximum delay value
        */

        set_dmax_in(0, max_delay);
    end

main_code
    if (num_in_buffers == 2)
        for (i = min_avail(); i > 0; i--) {
            it_in(0);
            it_in(1);

            /* check for SR RESET = 1 */

            if (*(int *)pin(1,0))
                for (j = 0; j <= max_delay; j++)
                    inf(0,j) = 0.0;

            /* output the delayed values */

            for (j = 0; j < n_delay; j++) {
                it_out(j);
                outf(j, 0) = inf(0,(int)delay[j]);
            }
        } /* big for */
    else
        while(it_in(0)) {
            /* output the delayed values */

            for (j = 0; j < n_delay; j++) {
                it_out(j);

```

```
        outf(j, 0) = inf(0,(int)delay[j]);  
    }  
} /* while */  
  
return(0);  
end
```

Fig. B.19 Source code for the SHIFTRREG star.


```
/* tec.s */
```

```
/******
```

Star: TEC - Transversal Echo Cancellor

Description:

This star implements a transversal echo canceller. The no. of taps can be specified. 2 adaptation gains can be specified for slow and fast convergence modes. Both linear and non-linear modes are supported. If the non-linear mode is specified, a set of coefficients is reserved for each distinct input symbol.

Note 1:

In the nonlinear mode, it is assumed that 2B1Q symbols are used. The zero symbol exists internally initially and causes no update to occur. The use of any other non-2B1Q symbol may cause memory errors. Currently, the input is not checked for valid 2B1Q symbols for speed considerations.

Note 2:

The taps are updated first followed by the calculation of the output. This allows both operations to be performed each time the star is scheduled thus increasing the simulation efficiency. Initially only an output operation is performed.

Parameters:

NUM_TAPS = Number of taps

NON_LIN = 0 Linear mode (DEFAULT)
 != 0 Non-linear mode (assuming 2B1Q symbols)

TEC_BETA1 = TEC adaptation gain 1

TEC_BETA2 = TEC adaptation gain 2

COEFF_FILE = Name of file containing FINAL tap coefficient values. In the nonlinear mode, coefficients are printed in the following order:

-3 coefficient set
 +1 coefficient set
 -1 coefficient set
 +3 coefficient set

= NOFILE No file is produced (DEFAULT)

Input 0: DTX = Input data

Input 1: ERR = Error

Input 2: TEC_CTL = Control word for the TEC

Bit 0: TEC RESET = 0 No reset
 = 1 $C_i(k) = 0$ for all i
 $ECT(k) = 0$

Bit 1: TEC ON/OFF = 0 $ECT(k) = 0$
 No update of $C_i(k)$
 = 1 $ECT(k) = \text{echo estimate}$

Bit 2: TEC UPDATE = 0 No update of $C_i(k)$
 = 1 Update $C_i(k)$

Bit 3: TEC SPEED = 0 Use $TEC_BETA = TEC_BETA1$
 = 1 Use $TEC_BETA = TEC_BETA2$

Output 0: ECT = echo estimate

Functionality:

Calculate

$$ECT(k) = \text{sum over all } i \{ C_{ri}(k) * DTX(k-i) \}$$

Update

$$C_{ri}(k+1) = C_{ri}(k) + TEC_BETA_j * ERR(k) * DTX(k-i)$$

where,

$$i = 0, \dots, \text{num_taps} - 1$$

$$j = 1, 2$$

$r = 0, \dots, 3$ Each different 2B1Q symbol causes a reference to its own coefficient set when NON_LIN is asserted. The following convenient mapping exists between the symbols and coefficient set:

3 => use C[3][i]	Note: This is convenient in that the first index is the actual symbol value when it is positive and 3 greater than the symbol value when the symbol value is negative.
1 => use C[1][i]	
-1 => use C[2][i]	
-3 => use C[0][i]	

where,
 $C[r][i] = C_{ri}$

Programmer: Ken Yamazaki

Date: July 26, 1988

Version: 1.0

Modified: Sept. 12, 1988 Changed order of operations so that the taps are updated first followed by the calculation of the output. (Ken Yamazaki)

Dec. 16, 1988 Fixed calculation of echo estimate in the normal mode. (Ken Yamazaki)

```

*****/
/* include file */

```

```

#include "nova.h"

```

```

parameters
    int num_taps;
    int non_lin = 0;
    float tec_beta1;
    float tec_beta2;
    file coeff_file = "nofile";
end

```

```

states
    float** c;
    int num_coeff_sets;
    int first = 1;
    FILE* fp;
    int dump_coeff = FALSE;
end

input_buffers
    float dtx;
    float err;
    int tec_ctl;
end

output_buffers
    float ect;
end

declarations
    int i, r;
    float sum;
    int coeff_set;
    float tec_beta;
    FILE *fopen();
end

initialization_code
    /* check that the number of taps is valid */
    if (num_taps <= 0) {
        fprintf(stderr,
            "tec: an invalid number of taps was specified\n");
        return(1);
    }

    /* configure the DTX input buffer to handle the number of taps
       specified
    */
    set_dmax_in (0, num_taps - 1);

    /* allocate coefficient space */

```

```

    num_coeff_sets = non_lin ? 4 : 1;
    if ( (c = (float **)calloc(num_coeff_sets,4)) == NULL) {
        fprintf(stderr,
            "tec: can't allocate coefficient space\n");
        return(2);
    }
    for (r = 0; r < num_coeff_sets; r++)
        if ( (c[r] = (float *)calloc(num_taps,4)) == NULL) {
            fprintf(stderr,
                "tec: can't allocate coefficient space\n");
            return(3);
        }

    /* get the coefficient file name if specified */
    if (strcmp(coeff_file,"stdout") == 0) {
        fp = stdout;
        dump_coeff = TRUE;
    }
    else
        if (strcmp(coeff_file,"nofile") != 0) {
            if ((fp = fopen(coeff_file,"w")) == NULL) {
                fprintf(stderr,
                    "tec: couldn't open final coefficients file\n");
                return(4);
            }
            dump_coeff = TRUE;
        }

    /* initialize the coefficient space */
    for (r = 0; r < num_coeff_sets; r++)
        for (i = 0; i < num_taps; i++)
            c[r][i] = 0.0;

end

main_code
    if (first) {
        if ((avail(0) > 0) && (avail(2) > 0)) {
            it_in(0);
            it_in(2);

```

```

        it_out(0);
        ect(0) = 0.0;

        first = 0;
    }
    return(0);
}

if (min_avail() > 0) {
    /* UPDATE */
    it_in(1);
    if ((tec_ctl(0) & bit1) && (tec_ctl(0) & bit2)) {
        /* TEC UPDATE ON */
        /* DETERMINE TEC SPEED */
        tec_beta = (tec_ctl(0) & bit3) ? tec_beta2 : tec_beta1;

        /* update coefficients */
        if (non_lin)
            for (i = 0; i < num_taps; i++){
                coeff_set = (dtx(i) < 0.0) ? dtx(i) + 3.0 : dtx(i);
                c[coeff_set][i] += tec_beta * err(0) * dtx(i);
            }
        else
            for (i = 0; i < num_taps; i++)
                c[0][i] += tec_beta * err(0) * dtx(i);
    }

    /* CALCULATE */
    it_in(0);
    it_in(2);
    it_out(0);
    sum = 0.0;

    /* check the bits of the control word */
    if (tec_ctl(0) & bit0)
        /* TEC RESET */
        for (r = 0; r < num_coeff_sets; r++)
            for (i = 0; i < num_taps; i++)
                c[r][i] = 0.0;

```

```

else
    if (tec_ctl(0) & bit1) {
        /* TEC ON */
        /* calculate and output echo estimate */
        sum = 0.0;
        if (non_lin)
            for (i = 0; i < num_taps; i++) {
                coeff_set = (dtx(i) < 0.0) ? dtx(i) + 3.0 : dtx(i)
                sum += c[coeff_set][i] * dtx(i);
            }
        else
            for (i = 0; i < num_taps; i++)
                sum += c[0][i] * dtx(i);
    }
    ect(0) = sum;
}
return(0);
end

wrapup_code
/* dump final coefficients if a file was specified */
if (dump_coeff) {
    for (r = 0; r < num_coeff_sets; r++)
        for (i = 0; i < num_taps; i++)
            fprintf(fp,"%f\n",c[r][i]);
    if (fp != stdout)
        fclose(fp);
}

/* free up coefficient space */
for (r = 0; r < num_coeff_sets; r++)
    free((char *)c[r]);
free((char *)c);
end

```

Fig. B.20 Source code for the TEC star.

APPENDIX C

GLOSSARY

C.1 TEXT CONVENTIONS

The following describes the text conventions used throughout this thesis to denote the various types of quantities.

- 1) *Italic* type denotes scalar quantities.
- 2) Lowercase **boldface** type denotes vector quantities.
- 3) Uppercase **BOLDFACE** type denotes matrix quantities.
- 4) The superscript symbol "T" denotes the matrix or vector transpose operation.
- 5) Symbols marked with the prime symbols, "'" and "'", are those related to or derived from the unprimed symbol.
- 6) The ordered pair (x,y) describes the dimensions of a matrix or vector in terms of the number of rows and columns respectively. This also appears in subscripted form adjacent to the relevant matrix or vector.
- 7) Variables of the form $V(k)$, $v(k)$, or $v(k)$ denote quantities at time or iteration number k .

C.2 ABBREVIATIONS

The following table contains a description of the abbreviations that were used in this thesis. They are listed below in alphabetical order.

TABLE C.1 Abbreviations

ABBREVIATION	MEANING
AMSE	Average Mean Square Error
AMSE _{ex}	Excess Average Mean Square Error
AMSE _{ss}	Steady State Average Mean Square Error
LMS	Least Mean Square
MEC	Memory Echo Cancellor
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
ϕ	2^m
ROC	Region of Convergence
TEC	Transveral Echo Cancellor
ψ	2^v

C.3 PRINCIPAL SYMBOLS

The following table contains a description of the principal symbols used in this thesis. They are listed in alphabetical order.

TABLE C.2 Principal Symbols

Symbol	Meaning
$\mathbf{0}_{(a,b)}$	A zero matrix with a rows and b columns.
α	The TEC coefficient adaptation step-size for the Jointly-Adaptive TEC/MEC.
α_1	The TEC coefficient adaptation step-size for the Sequentially-Adaptive TEC/MEC.
$\mathbf{a}(k)$	The $(2^m, 1)$ MEC address vector at time k . An equivalent representation of $\mathbf{x}'(k)$. ie. $\mathbf{x}'(k) = \mathbf{A}\mathbf{a}(k)$
$\mathbf{a}_h(k)$	The $(2^v, 1)$ address vector representation of transmitted symbols affected by the nonlinear echo path function.
$\mathbf{b}$	A $(4,1)$ vector partially governing the contribution of the adaptation process of each coefficient partition to the AMSE_{ex} of the Jointly-Adaptive TEC/MEC.
$\mathbf{b}_s$	A $(2,1)$ vector partially governing the contribution of the adaptation process of each coefficient partition to the AMSE_{ex} of phase II of the Sequentially-Adaptive TEC/MEC operation.
β	The MEC coefficient adaptation step-size for the Jointly-Adaptive TEC/MEC.

TABLE C.2 Principal Symbols (cont'd)

Symbol	Meaning
β_1	The MEC coefficient adaptation step-size for the Sequentially-Adaptive TEC/MEC.
$\mathbf{c}(k)$	The joint overlapped coefficient vector formed from the concatenation of $\mathbf{s}'(k)$ and $\mathbf{m}(k)$.
$\mathbf{c}^*$	A joint overlapped coefficient vector that satisfies the MMSE criteria.
$d(k)$	The echo path output sample at time k .
$e(k)$	The echo cancellation error at time k . ie. $e(k) = d(k) - \text{TEC/MEC Output}(k)$
$\mathbf{E}$	A matrix whose columns consist of the eigenvectors of $\mathbf{T}$. It diagonalizes $\mathbf{T}$. ie. $\Lambda_{\mathbf{T}} = \mathbf{E}^{-1}\mathbf{T}\mathbf{E}$ where $\Lambda_{\mathbf{T}}$ is the resulting diagonal matrix containing the corresponding eigenvalues of $\mathbf{T}$.
ϕ	An abbreviation of 2^m .
$\mathbf{F}\phi$	A $(2^m, 2^v)$ matrix that transforms the transmitted symbol information affected by the nonlinear echo path function into that actually "seen" by the overlapped part of the TEC of the Jointly-Adaptive TEC/MEC.

TABLE C.2 Principal Symbols (cont'd)

Symbol	Meaning
$\mathbf{F}_m$	A (m, u) matrix that transforms the transmitted symbol information affected by the linear echo path function into that actually "seen" by the MEC of the Jointly-Adaptive TEC/MEC.
$\mathbf{F}_n$	A $(n-m, u)$ matrix that transforms the transmitted symbol information affected by the linear echo path function into that which is actually "seen" by the nonoverlapped part of the TEC of the Jointly-Adaptive TEC/MEC.
$\mathbf{h}_l$	A $(u, 1)$ vector representing the linear part of the echo path function.
$\mathbf{h}_n$	A $(2^v, 1)$ vector representing the nonlinear part of the echo path function.
$\mathbf{I}_a$	An (a, a) identity matrix where a is typically n, m, u , or ψ .
$\mathbf{J}$	The $(m+2^m, m+2^m)$ adaptation step-size matrix for the joint overlapped TEC/MEC coefficient representation of the LMS algorithm for the Jointly-Adaptive TEC/MEC.
λ_ϵ	The smallest eigenvalue of $\mathbf{T}$.
λ_{se}	The smallest eigenvalue of $\mathbf{T}_s$.
λ_{Tj}	The j th eigenvalue of $\mathbf{T}$ where $j = 0, 1, 2, 3$.

TABLE C.2 Principal Symbols (cont'd)

Symbol	Meaning
λ_{Ts_j}	The j th eigenvalue of T_s where $j = 0, 1$.
Λ_T	A diagonal matrix containing the eigenvalues of T . ie. $\Lambda_T = E^{-1}TE$ where E is the matrix containing the eigenvectors of T .
m	The no. of current and past transmitted symbols used by the MEC.
$m_i(k)$	The i th MEC coefficient at time k .
$\mathbf{m}(k)$	A $(2^m, 1)$ vector containing the 2^m coefficients of the MEC at time k . ie. $\mathbf{m}(k) = [m_0(k) \ m_1(k) \ m_2(k) \ \dots \ m_{\phi-1}(k)]^T$
$\mathbf{m}^*$	A MEC coefficient vector that satisfies the MMSE condition.
$\mathbf{m}_e(k)$	The MEC coefficient error vector at time k . ie. $\mathbf{m}_e(k) = \mathbf{m}(k) - \mathbf{m}^*$
n	The no. of current and past transmitted symbols used by the TEC. For practical purposes $n \geq m$.
$\mathbf{p}$	The TEC input/echo crosscorrelation vector. ie. $\mathbf{p} = E\{d(k)\mathbf{x}(k)\}$
$\mathbf{p}'$	The overlapped TEC input/echo crosscorrelation vector. ie. $\mathbf{p}' = E\{d(k)\mathbf{x}'(k)\}$
$\mathbf{p}''$	The nonoverlapped TEC input/echo crosscorrelation vector. ie. $\mathbf{p}'' = E\{d(k)\mathbf{x}''(k)\}$

TABLE C.2 Principal Symbols (cont'd)

Symbol	Meaning
$\mathbf{p}_q$	The joint overlapped input/echo crosscorrelation vector formed by the concatenation of $\mathbf{p}'$ and $\mathbf{q}$.
$\mathbf{q}$	The MEC input/echo crosscorrelation vector. ie. $\mathbf{q} = E\{d(k)\mathbf{a}(k)\}$
$\mathbf{R}_m$	The joint overlapped TEC/MEC input autocorrelation matrix ie. $\mathbf{R}_m = E\{\mathbf{x}_a(k)\mathbf{x}_a^T(k)\}$
$s_i(k)$	The i th TEC coefficient at time k .
$\mathbf{s}(k)$	The $(n,1)$ vector containing the all n TEC coefficients at time k .
$\mathbf{s}'(k)$	The $(m,1)$ vector containing the first m TEC coefficients that are overlapped with those of the MEC at time k . ie. $\mathbf{s}'(k) = [s_0(k) s_1(k) s_2(k) \dots s_{m-1}(k)]^T$
$\mathbf{s}''(k)$	The $(n-m,1)$ vector containing the remaining $n-m$ TEC coefficients that are not overlapped with those of the MEC at time k . ie. $\mathbf{s}''(k) = [s_m(k) s_{m+1}(k) s_{m+2}(k) \dots s_{n-1}(k)]^T$
$\mathbf{s}^*$	A TEC coefficient vector that satisfies the MMSE criteria.
$\mathbf{s}'^*$	An overlapped TEC coefficient vector that satisfies the MMSE criteria.

TABLE C.2 Principal Symbols (cont'd)

Symbol	Meaning
$\mathbf{s}^{**}$	A nonoverlapped TEC coefficient vector that satisfies the MMSE criteria.
$\mathbf{s}_e(k)$	The TEC coefficient error vector at time k . ie. $\mathbf{s}_e(k) = \mathbf{s}(k) - \mathbf{s}^*$
$\mathbf{s}'_e(k)$	The overlapped TEC coefficient error vector at time k . ie. $\mathbf{s}'_e(k) = \mathbf{s}'(k) - \mathbf{s}^*$
$\mathbf{s}''_e(k)$	The nonoverlapped TEC coefficient error vector at time k . ie. $\mathbf{s}''_e(k) = \mathbf{s}''(k) - \mathbf{s}^{**}$
$\mathbf{T}$	The coefficient error power update matrix for the Jointly-Adaptive TEC/MEC. This (4,4) matrix governs how the components of the AMSE change with each iteration of the LMS algorithm.
$\mathbf{T}_s$	The coefficient error power update matrix for phase II of the Sequentially-Adaptive TEC/MEC operation. This (2,2) matrix governs how the components of the AMSE change with each iteration of the LMS algorithm.
u	The linear echo path span.
v	The nonlinear echo path span.
$v_n(k)$	The TEC coefficient error power at time k . ie. $v_n(k) = E\{\mathbf{s}_e^T(k)\mathbf{s}_e(k)\}$

TABLE C.2 Principal Symbols (cont'd)

Symbol	Meaning
$v'(k)$	The overlapped TEC coefficient error power at time k . ie. $v'(k) = E\{s'_e{}^T(k)s'_e(k)\}$
$v''(k)$	The nonoverlapped TEC coefficient error power at time k . ie. $v''(k) = E\{s''_e{}^T(k)s''_e(k)\}$
$w(k)$	The MEC coefficient error power at time k . ie. $w(k) = 2^{-m}E\{m_e{}^T(k)m_e(k)\}$
$x(k-i)$	The i th input symbol in the past.
$\mathbf{x}(k)$	The $(n,1)$ vector representing all n current and past TEC input symbols at time k . ie. $\mathbf{x}(k) = [x(k) \ x(k-1) \ x(k-2) \ \dots \ x(k-n+1)]^T$
$\mathbf{x}'(k)$	The $(m,1)$ vector representing the first m (overlapped) TEC input symbols at time k . ie. $\mathbf{x}'(k) = [x(k) \ x(k-1) \ x(k-2) \ \dots \ x(k-m+1)]^T$
$\mathbf{x}''(k)$	The $(n-m,1)$ vector representing the remaining $n-m$ TEC input symbols at time k . ie. $\mathbf{x}''(k) = [x(k-m) \ x(k-m-1) \ \dots \ x(k-n+1)]^T$
$\mathbf{x}_a(k)$	The $(m + 2m, 1)$ joint overlapped TEC/MEC input vector formed from the concatenation of $\mathbf{x}'(k)$ and $\mathbf{a}(k)$.
$\mathbf{x}_h(k)$	The $(u,1)$ echo path input vector that is affected by the linear echo path function at time k .

TABLE C.2 Principal Symbols (cont'd)

$\mathbf{Y}$	the linear/nonlinear TEC/MEC input crosscorrelation matrix. ie. $\mathbf{Y} = E\{\mathbf{x}'(k)\mathbf{a}^T(k)\}$
ψ	An abbreviation of 2^ν .
$\mathbf{Y}_h$	The linear/nonlinear echo path input cross-correlation matrix. ie. $\mathbf{Y}_h = E\{\mathbf{x}_h(k)\mathbf{a}_h^T(k)\}$
$z(k)$	The overlapped TEC coefficient error/MEC coefficient error joint power component at time k . ie. $z(k) = 2 \cdot E\{\mathbf{s}'_e^T(k)\mathbf{Y}\mathbf{m}_e(k)\}$
$\mathbf{z}(k)$	A (4,1) vector containing the 4 coefficient error powers related to the overlapped and nonoverlapped coefficient partitions. ie. $\mathbf{z}(k) = [v'(k) \ v''(k) \ w(k) \ z(k)]^T$
$\mathbf{z}_s(k)$	A (2,1) vector containing the MEC and nonoverlapped TEC coefficient error powers. ie. $\mathbf{z}_s(k) = [v''(k) \ w(k)]^T$

REFERENCES

- [Ada'88] P.F. Adams, C.F.N. Cowan, and M.J. Smith, "Nonlinear Echo Cancellers Based on Transpose Distributed Arithmetic", IEEE Transactions on Circuits and Systems, Vol. 35, No.1, pp. 6-18, January 1988.
- [Aga'82] O. Agazzi, D.A. Hodges, and D.G. Messerschmitt, "Nonlinear Echo Cancellation of Data Signals", IEEE Transactions on Communications, Vol. COM-30, No. 11, pp. 2421-2433, November 1982.
- [Ans'88] American National Standard for Telecommunications, "Integrated Services Digital Network - Basic Access Interface for Use on Metallic Loops for Applications on the Network Side of the NT - Layer 1 Specification (ANSI T1.601 - 1988), 1988.
- [Ard'88] S. Ardalan, J. Faber, and R. Nobahkt, "CAPSIM User's Manual", North Carolina State University, 1988.
- [Fal'87] D.D. Falconer, "Convergence of the Adaptive Echo Canceller", Bell Northern Research Ltd. internal document - Dept. 1T20, pp. 2-3, March 1987.
- [Hay'86] S. Haykin, "Adaptive Filter Theory", Prentice-Hall, Toronto, 1986.

- [Hof'60] M.E. Hoff Jr., and B. Widrow, "Adaptive Switching Circuits", IRE WESCON Conv. Rec., Part 4, pp.96-104, 1960.
- [Hol'81] N. Holte and S. Stueflotten, "A New Digital Echo Canceller for Two-Wire Subscriber Lines", IEEE Transactions on Communications, Vol. COM-29, No. 11, pp.1573-1581, November 1981.
- [Lev'84] M.Y. Levy, X.C. Maitre, and S.R. Surie, "The Multi-Memory Echo Canceller for Two-Wire Digital Transmission", Globecom 1984, Vol. 1, pp. 196-202, November 1984.
- [Liu'74] B. Liu and A. Peled, "A New Hardware Realization of Digital Filters", IEEE Transactions on ASSP, Vol. ASSP-22, No. 6, pp. 456-462, December 1974.
- [Maz'79] J.E. Mazo, "On the Independence Theory of Equalizer Convergence", Bell System Technical Journal, Vol. 58, pp. 963-993, June 1979.
- [Mes'84] D.G. Messerschmitt, "A Tool for Structured Functional Simulation" IEEE Journal on Selected Areas in Communications, pp. 137-147, January 1984.
- [Mess'84] D.G. Messerschmitt, "Echo Cancellation in Speech and Data Transmission", IEEE Journal on Selected Areas in Communications, Vol. SAC-2, No. 2, March 1984.
- [Mes'86] D.G. Messerschmitt, "STARGAZE - A Preprocessor for BLOSIM User STAR Functions", Dept. of Electrical Engineering and Computer Science, University of California at Berkeley, November 1986.

- [Mog'87] C. Mogavero, G. Nervo, and G. Paschetta, "Mixed Recursive Echo Canceller (MREC)", CSELT Technical Reports, Vol. XV, No. 2, pp. 103-107, March 1987.
- [Nob'88] R.A. Nobakht, P.W. Pate, S.H. Ardalan, "CAPSIM: A Graphical Simulation Tool for Communication Systems", IEEE Globecom 1988, 1988.
- [Pet'87] E.G. Peter, "Multistage RAM: An FIR Filter for Echo Cancellation in a Digital Two-Wire Subscriber Loop", IEEE Transactions on Circuits and Systems, Vol. CAS-34, No. 3, pp. 225-232, March 1987.
- [Son'67] M.M. Sondhi, "An Adaptive Echo Canceller", Bell System Technical Journal., vol. 46, pp. 497-511, March 1967.
- [Ste'85] S.D. Stearns and B. Widrow, "Adaptive Signal Processing", Prentice-Hall, Toronto, 1985.

END

2/6/09-90

FIN