ITERATIVE DECODING OF TURBO CODES
AND OTHER CONCATENATED CODES

A Dissertation

Submitted to the School of Electronic Engineering
Faculty of Engineering
University of South Australia
for the Degree of

Doctor of Philosophy

by

Sorin Adrian Barbulescu

February 1996
"For more information/links on turbo codes visit http://people.myoffice.net.au/~abarbulescu"
# TABLE OF CONTENTS

Table of Contents ............................................................... i
List of Figures ................................................................. iv
List of Tables ................................................................. viii
Glossary ....................................................................... ix
List of abbreviations ........................................................... x
Summary ....................................................................... xi
Statement of originality .......................................................... xii
Acknowledgements ............................................................... xiii

1 INTRODUCTION ............................................................... 1
  1.1 Fundamentals ............................................................. 2
  1.3 Methodology ............................................................ 8
  1.4 Thesis outline ........................................................... 8
  1.5 Summary of the original contributions ................................. 9

2 A MAP DECODING ALGORITHM ............................................ 10
  2.1 Introduction ............................................................. 10
  2.2 Recursive systematic encoder ........................................... 11
  2.3 Description of the MAP algorithm ..................................... 12
  2.4 Definitions of $\alpha$ and $\beta$ ....................................... 13
  2.5 Derivation of $\alpha$ .................................................... 14
  2.6 Derivation of $\beta$ .................................................... 15
  2.7 Comparison with other MAP algorithms ............................... 16
  2.8 Derivation of $\gamma$ and graphical representations ..................... 16
  2.9 Steps of the algorithm ................................................ 20
  2.10 Comparison between the MAP and the Soft Output Viterbi Algorithms ........ 20
  2.11 MAP decoding with overlapping blocks ................................ 23
  2.12 Conclusions ........................................................... 24

3 ITERATIVE DECODING OF TURBO CODES ............................... 26
  3.1 Introduction ............................................................. 26
  3.2 Literature review ........................................................ 26
  3.3 Description of the turbo encoder ..................................... 30
3.4 Principles of iterative decoding ........................................ 31
3.5 Application of the MAP algorithm in iterative decoding .......... 33
3.6 Interleaver design ......................................................... 36
  3.6.1 Proposed “odd–even” interleaver ................................. 36
  3.6.2 Proposed “simile” interleaver ..................................... 38
  3.6.3 Interleaving the tail or not? ....................................... 41
  3.6.4 Space–time distribution for helical interleavers ................. 42
3.7 Performance of turbo codes on an AWGN channel .................. 44
  3.7.1 Four state rate half turbo code ................................. 44
  3.7.2 Four state rate third turbo code ............................... 49
  3.7.3 Frame error ratio for the four state turbo code ............... 52
  3.7.4 Sixteen state rate half turbo code ......................... 54
  3.7.5 Sixteen state rate third turbo code .......................... 57
  3.7.6 Frame error ratio for the sixteen state turbo code ......... 59
3.8 Performance of turbo codes on a Rayleigh channel ............... 61
  3.8.1 Four state rate half turbo code ................................. 63
  3.8.2 Four state rate third turbo code ............................... 66
  3.8.3 Frame error ratio for the four state turbo code ............... 68
  3.8.4 Sixteen state rate half turbo code ............................ 70
  3.8.5 Sixteen state rate third turbo code ......................... 72
  3.8.6 Frame error ratio for the sixteen state turbo code ......... 74
3.9 Guidelines for the interleaver design ................................ 76
3.10 Interleaver design for encryption purposes ......................... 77
  3.10.1 Pseudo random interleaver with noise ....................... 77
  3.10.2 Pseudo random interleaver for the encrypted data with noise 78
3.11 Conclusions ............................................................... 79

4 A DIVERSITY SCHEME USING TURBO CODES ................. 80
4.1 Introduction ............................................................... 80
4.2 Diversity combining methods .......................................... 80
4.3 Turbo Diversity Scheme ............................................... 82
4.4 AWGN channel ........................................................... 83
4.5 Fading channel ......................................................... 86
  4.5.1 Traditional maximum ratio combining ......................... 86
  4.5.2 Maximum ratio combining with TDS ........................... 87
  4.5.3 Traditional equal gain combining .............................. 88
  4.5.4 Equal gain combining with TDS ............................... 89
  4.5.5 Traditional selection combining ............................... 90
  4.5.6 Selection combining with TDS ................................. 91
4.6 Conclusions ............................................................... 92
5 RATE–COMPATIBLE TURBO CODES ................................. 93
  5.1 Introduction .......................................................... 93
  5.2 Rate one half turbo codes ....................................... 94
  5.3 Rate one third turbo codes ...................................... 94
  5.4 Methods of combining the two different rates ............... 95
     5.4.1 Different frame sizes with different coding rates .... 95
     5.4.2 Same frame size with different coding rates and one interleaver 98
     5.4.3 Same frame size with different coding rates and two interleavers 102
  5.5 Interleaver design for three dimensional turbo codes .......... 104
     5.5.1 Three dimensional four state turbo code ............ 107
     5.5.2 Three dimensional sixteen state turbo code .......... 111
  5.6 Conclusions ......................................................... 114

6 ITERATIVE DECODING OF PRODUCT CODES .................... 116
  6.1 Introduction ....................................................... 116
  6.2 Review of different product coding schemes ................ 116
  6.3 Turbo coding the (15, 11) binary cyclic code ............... 118
     6.3.1 Construction I .............................................. 119
     6.3.2 Construction II ............................................ 123
     6.3.3 Limiting factors for product codes .................... 125
  6.4 Conclusions ......................................................... 126

7 ITERATIVE DECODING OF CONCATENATED CODES .............. 127
  7.1 Introduction ....................................................... 127
  7.2 Review of decoding techniques .................................. 127
  7.3 Regenerate Transmitted Symbol Algorithm (RTS) ............ 129
  7.4 Computer simulation results ..................................... 131
  7.5 Conclusions ......................................................... 134

8 CONCLUSIONS .......................................................... 135

9 REFERENCES ............................................................ 141
LIST OF FIGURES

Figure 1.1 – A block diagram of a communication system ........................... 1
Figure 1.2 – Channel capacity for Phase Shift Keying modulation ............... 4
Figure 1.3 – Bandwidth efficiency for Phase Shift Keying modulation ........... 5
Figure 1.4 – Concatenated coding system ................................................ 6
Figure 1.5 – C_{soft} and C_{hard} as a function of SNR ................................. 7
Figure 2.1 – Rate half systematic convolutional encoder with feedback ......... 11
Figure 2.2 – Graphical representation of (2.28) ......................................... 17
Figure 2.3 – Graphical representation of (2.30) ......................................... 18
Figure 2.4 – Soft outputs for MAP and SOVA decoders .............................. 22
Figure 2.5 – MAP decoding using overlapping blocks ............................... 23
Figure 3.1 – Convolutional encoder including m−fold time division interleaving . 28
Figure 3.2 – Two tier coding with rate 2/3 component codes ....................... 28
Figure 3.3 – Optimal interleaving for the ith group .................................... 29
Figure 3.4 – Turbo encoder ................................................................. 30
Figure 3.5 – QPSK modulator ............................................................... 30
Figure 3.6 – AWGN channel (0, \sigma^2) ..................................................... 31
Figure 3.7 – Soft−in/Soft−out Decoder ................................................... 32
Figure 3.8 – Turbo encoder with time synchronisation ............................... 34
Figure 3.9 – Iterative decoder ............................................................... 35
Figure 3.10 – A four state systematic convolutional encoder ....................... 38
Figure 3.11 – Difference between TIE and TNIE encoders ......................... 41
Figure 3.12 – Space−time distribution for a block interleaver ...................... 43
Figure 3.13 – Space−time distribution for a helical interleaver .................... 43
Figure 3.14 – BER for a block size B = 102 bits (4 states, rate half, AWGN) ... 44
Figure 3.15 – BER for a block size B = 204 bits (4 states, rate half, AWGN) ... 44
Figure 3.16 – BER for a block size B = 306 bits (4 states, rate half, AWGN) ... 45
Figure 3.17 – BER for a block size B = 408 bits (4 states, rate half, AWGN) ... 45
Figure 3.18 – BER for a block size B = 930 bits (4 states, rate half, AWGN) ... 46
Figure 3.19 – BER for a block size B = 3660 bits (4 states rate half, AWGN) ... 46
Figure 3.20 – BER for a block size B = 768 bits (4 states, rate half, AWGN)
using a pseudo random interleaver of 256 bits ................................. 47
Figure 3.21 – BER for a block size B = 3072 bits (4 states, rate half, AWGN)
using a pseudo random interleaver of 1024 bits ................................. 48
Figure 3.22 – BER for a block size B = 49152 bits (4 states, rate half, AWGN)
using a pseudo random interleaver of 16384 bits ................................. 48
Figure 3.23 – BER for a block size B = 102 bits (4 states, rate third, AWGN) .... 48
Figure 3.24 – BER for a block size B = 204 bits (4 states, rate third, AWGN) .... 49
Figure 3.25 – BER for a block size B = 306 bits (4 states, rate third, AWGN) .... 50
Figure 3.26 – BER for a block size B = 408 bits (4 states, rate third, AWGN) .... 50
Figure 3.27 – BER for a block size B = 3660 bits (4 states, rate third, AWGN) ..... 51
Figure 3.28 – Frame error ratio (4 states, rate half, AWGN) .......................... 53
Figure 3.29 – Frame error ratio (4 states, rate third, AWGN) .......................... 53
Figure 3.30 – BER for a block size B = 110 bits (16 states, rate half, AWGN) ..... 54
Figure 3.31 – BER for a block size B = 220 bits (16 states, rate half, AWGN) ..... 54
Figure 3.32 – BER for a block size B = 420 bits (16 states, rate half, AWGN) ..... 55
Figure 3.33 – BER for a block size B = 930 bits (16 states, rate half, AWGN) ..... 55
Figure 3.34 – BER for a block size B = 3660 bits (16 states, rate half, AWGN) ..... 56
Figure 3.35 – BER for a block size B = 110 bits (16 states, rate third, AWGN) ..... 57
Figure 3.36 – BER for a block size B = 220 bits (16 states, rate third, AWGN) ..... 57
Figure 3.37 – BER for a block size B = 420 bits (16 states, rate third, AWGN) ..... 58
Figure 3.38 – BER for a block size B = 930 bits (16 states, rate third, AWGN) ..... 57
Figure 3.39 – BER for a block size B = 3660 bits (16 states, rate third, AWGN) ..... 59
Figure 3.40 – Frame error ratio (16 states, rate half, AWGN) .......................... 60
Figure 3.41 – Frame error ratio (16 states, rate third, AWGN) .......................... 60
Figure 3.42 – Rayleigh channel ..................................................................... 62
Figure 3.43 – BER for a block size B = 102 bits (4 states, rate half, fading) .... 63
Figure 3.44 – BER for a block size B = 204 bits (4 states, rate half, fading) .... 63
Figure 3.45 – BER for a block size B = 306 bits (4 states, rate half, fading) .... 64
Figure 3.46 – BER for a block size B = 408 bits (4 states, rate half, fading) .... 64
Figure 3.47 – BER for a block size B = 930 bits (4 states, rate half, fading) .... 65
Figure 3.48 – BER for a block size B = 102 bits (4 states, rate third, fading) .... 66
Figure 3.49 – BER for a block size B = 204 bits (4 states, rate third, fading) .... 66
Figure 3.50 – BER for a block size B = 306 bits (4 states, rate third, fading) .... 67
Figure 3.51 – BER for a block size B = 408 bits (4 states, rate third, fading) .... 67
Figure 3.52 – BER for a block size B = 930 bits (4 states, rate third, fading) .... 68
Figure 3.53 – Frame error ratio (4 states, rate half, fading) .......................... 69
Figure 3.54 – Frame error ratio (4 states, rate third, fading) .......................... 69
Figure 3.55 – BER for a block size B = 110 bits (16 states, rate half, fading) .... 70
Figure 3.56 – BER for a block size B = 220 bits (16 states, rate half, fading) .... 70
Figure 3.57 – BER for a block size B = 420 bits (16 states, rate half, fading) .... 71
Figure 3.58 – BER for a block size B = 930 bits (16 states, rate half, fading) .... 71
Figure 3.59 – BER for a block size B = 110 bits (16 states, rate third, fading) .... 72
Figure 3.60 – BER for a block size B = 220 bits (16 states, rate third, fading) .... 73
Figure 3.61 – BER for a block size B = 420 bits (16 states, rate third, fading) .... 73
Figure 3.62 – BER for a block size B = 930 bits (16 states, rate third, fading) .... 74
Figure 3.63 – Frame error ratio (16 states, rate half, fading) .......................... 75
Figure 3.64 – Frame error ratio (16 states, rate third, fading) .......................... 75
Figure 3.65 – Extrinsic information for turbo codes ........................................ 76
Figure 3.66 – Three dimensional turbo code ................................................. 77
Figure 3.67 – Pseudo random interleaver for the encrypted data with noise .... 78
Figure 4.1 – Diversity scheme using two satellites ........................................... 80
Figure 4.2 – Turbo encoder .............................................................................. 82
Figure 4.3 – One satellite channel received ....................................................... 83
Figure 4.4 – Two satellite channels received ...................................................... 83
Figure 4.5 – BER for diversity with a four state turbo code (AWGN channel) ....... 84
Figure 4.6 – FER for diversity with a four state turbo code (AWGN channel) ....... 84
Figure 4.7 – BER for diversity with a sixteen state turbo code (AWGN channel) ... 85
Figure 4.8 – FER for diversity with a sixteen state turbo code (AWGN channel) ... 85
Figure 4.9 – MRC for a fading channel ............................................................ 86
Figure 4.10 – MRC–TDS for a fading channel .................................................. 87
Figure 4.11 – EGC for a fading channel ............................................................ 88
Figure 4.12 – EGC–TDS for a fading channel ................................................... 89
Figure 4.13 – SLC for a fading channel ............................................................. 90
Figure 4.14 – SLC–TDS for a fading channel ................................................... 91
Figure 4.15 – TDS for a fading channel for a sixteen state turbo code .............. 92
Figure 5.1 – Rate one half turbo encoder ......................................................... 94
Figure 5.2 – Rate one half turbo decoder ......................................................... 94
Figure 5.3 – Rate one third turbo encoder ....................................................... 94
Figure 5.4 – Rate one third turbo decoder ....................................................... 95
Figure 5.5 – BER for two blocks of size B = 104 bits each
at SNR = 0.0 dB, 1.0 dB and 2.0 dB ............................................................... 96
Figure 5.6 – Different interleaver sizes at SNR = 0.0dB ................................. 97
Figure 5.7 – BER for a 1/4 ratio at SNR = 0.0, 1.0 and 2.0 dB (B = 310) ......... 99
Figure 5.8 – BER for a 1/2 ratio at SNR = 0.0, 1.0 and 2.0 dB (B = 310) ......... 99
Figure 5.9 – BER for a 3/4 ratio at SNR = 0.0 dB (B = 310) .......................... 100
Figure 5.10 – BER for a 1/2 ratio at SNR 1.0dB (B = 104) .............................. 101
Figure 5.11 – BER for a 1/2 ratio at SNR 1.0dB (B = 206) .............................. 101
Figure 5.12 – Block structure for two level UEP ........................................... 102
Figure 5.13 – BER for a two level UEP with two level interleaving ................. 103
Figure 5.14 – Three dimensional turbo code (3D–TC) .................................. 104
Figure 5.15 – “Simile” interleaver for 3D–TC ................................................. 105
Figure 5.16 – Parallel structure for a 3D–TC decoder .................................. 105
Figure 5.17 – Serial structure for a 3D–TC decoder ...................................... 106
Figure 5.18 – BER for a four state rate half, third and quarter codes
and interleaver size of 102 bits (AWGN channel) ........................................ 107
Figure 5.19 – BER for a four state rate half, third and quarter codes
and interleaver size of 204 bits (AWGN channel) ........................................ 107
Figure 5.20 – BER for rate half, third and quarter codes and
interleaver size of 306 bits (AWGN channel) ........................................... 108
Figure 5.21 – BER for rate half, third and quarter code and
interleaver size of 3660 bits (AWGN channel) ........................................ 108
Figure 5.22 – BER for four state rate half code at Eb/N0 = 0 dB, 1 dB and 2 dB
(interleaver size of 3660 bits, AWGN channel) ...................... 109
Figure 5.23 – BER for four state rate third code at Eb/N0 = 0 dB, 1 dB and 2 dB
(interleaver size of 3660 bits, AWGN channel) ...................... 109
Figure 5.24 – BER for a four state rate quarter code at Eb/N0 = 0 dB
(interleaver size of 3660 bits, AWGN channel) ...................... 110
Figure 5.25 – BER for a four state rate quarter code at Eb/N0 = 0 dB
(interleaver size of 16500 bits, AWGN channel) ...................... 110
Figure 5.26 – BER for a sixteen state rate half, third and quarter codes
and interleaver size of 110 bits (AWGN channel) ...................... 111
Figure 5.27 – BER for a sixteen state rate half, third and quarter codes
and interleaver size of 220 bits (AWGN channel) ...................... 111
Figure 5.28 – BER for a sixteen state rate half, third and quarter codes
and interleaver size of 420 bits (AWGN channel) ...................... 112
Figure 5.29 – BER for a sixteen state rate half, third and quarter codes
and interleaver size of 930 bits (AWGN channel) ...................... 112
Figure 5.30 – A n-dimensional turbo encoder ............................ 113
Figure 5.31 – A “mixed-state” turbo encoder ............................ 114
Figure 6.1 – Parity check product code ................................. 117
Figure 6.2 – Encoder for binary (15,11) code with g(x) = x^4 + x + 1 ........................ 118
Figure 6.3 – Path in the trellis for codeword 101001011101101 ........ 118
Figure 6.4 – Turbo encoder used for block codes to produce a product code ... 119
Figure 6.5 – Turbo encoder output ....................................... 119
Figure 6.6 – Construction I of a product code ......................... 120
Figure 6.7 – BER for a product code based on the (15, 11) cyclic code and 11 \times 12 bit
helical interleaver using construction I (rate = 0.58) .................. 121
Figure 6.8 – Comparison between a product code and a turbo code
(16 states, 110 bit helical interleaver) .................................. 121
Figure 6.9 – Comparison between a product code and a turbo code
(16 states, 220 bit helical interleaver) .................................. 122
Figure 6.10 – BER for a product code based on the (15, 11) cyclic code
11 \times 12 bit helical interleaver with punctured parity bits (rate = 0.73) . 122
Figure 6.11 – Construction II of a product code ....................... 123
Figure 6.12 – BER comparison for a product code based on the (15, 11) cyclic code
using construction I (rate = 0.58) and construction II (rate = 0.49) ... 124
Figure 6.13 – Extrinsic information for product codes .................. 125
Figure 7.1 – Standard concatenated coding system .................... 127
Figure 7.2 – Iterative decoding using non-uniform RS codes .......... 128
Figure 7.3 – Regenerate Transmitted Symbol Decoder ................ 130
Figure 7.4 – Simulated model ............................................. 131
Figure 7.5 – Encoder for rate 8/9 6-D 8PSK Trellis Code .............. 132
Figure 7.6 – BER for the rate 8/9 6-D 8PSK Trellis Code before and after one
iteration ................................................................. 133
LIST OF TABLES

Table 2.1 – BER for a four state code using the MAP decoder and SOVA decoder . . 23
Table 3.1 – Odd positioned coded bits ......................................................... 36
Table 3.2 – Even positioned coded bits ......................................................... 36
Table 3.3 – The output for a pseudo–random interleaver ............................... 36
Table 3.4 – 3x5 block interleaver ................................................................. 37
Table 3.5 – Even positioned coded bits ......................................................... 37
Table 3.6 – Information bits and multiplexed coded bits for an
  “odd–even” interleaver .............................................................................. 37
Table 3.7 – State sequence for a four state turbo encoder .............................. 39
Table 3.8 – Final encoder state for n = 2 for N information bits ................. 39
Table 3.9 – An example of a “simile” interleaver with depth n + 1 for n = 2 .... 40
Table 3.10 – The output of a “simile” interleaver ....................................... 40
Table 3.11 – “Simile odd–even” block helical interleaver ........................... 40
Table 3.12 – The output of a “simile odd–even” block helical interleaver ....... 40
Table 3.13 – A block interleaver with data stored row–wise ......................... 42
Table 7.1 – Block deinterleaver .................................................................... 129
GLOSSARY

B_d .................... Doppler spread of the channel
C ......................... channel capacity
C_hard ................... capacity for a discrete–input hard–output memoryless channel
C_soft .................... capacity for a discrete–input soft–output memoryless channel
(Δt)_c ..................... coherence time of the channel
(Δf)_c ................. coherence bandwidth of the channel
E{ } ...................... expectation operator
E_b ....................... energy per information bit
E_s ....................... energy per two dimensional symbol
η .......................... bandwidth efficiency
GF(q) ..................... Galois field of q elements
L(d_k) .................... log likelihood ratio associated with the decoded bit d_k
N_0/2 ...................... double sided noise density
p_k ....................... normal variable
q_k ....................... normal variable
S_k ....................... encoder state at time k
σ^2 ....................... noise variance
T_m ....................... multipath spread of the channel
W ......................... bandwidth
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP</td>
<td>A Posteriori Probability</td>
</tr>
<tr>
<td>AWGN</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Ratio</td>
</tr>
<tr>
<td>BPSK</td>
<td>Binary Phase Shift Keying</td>
</tr>
<tr>
<td>BSC</td>
<td>Binary Symmetric Channel</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>DEC</td>
<td>DECoder block</td>
</tr>
<tr>
<td>EGC</td>
<td>Equal Gain Combining</td>
</tr>
<tr>
<td>ENC</td>
<td>ENCoder block</td>
</tr>
<tr>
<td>FER</td>
<td>Frame Error Ratio</td>
</tr>
<tr>
<td>IDMC</td>
<td>Input Discrete Memoryless Channel</td>
</tr>
<tr>
<td>INT</td>
<td>INTerleaver block</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum A Posteriori</td>
</tr>
<tr>
<td>MD–TC</td>
<td>M–Dimensional Turbo Code</td>
</tr>
<tr>
<td>MLDA</td>
<td>Maximum Likelihood Decoding Algorithm</td>
</tr>
<tr>
<td>MRC</td>
<td>Maximum Ratio Combining</td>
</tr>
<tr>
<td>PCBC</td>
<td>Parallel Concatenated Block Codes</td>
</tr>
<tr>
<td>PRM</td>
<td>Products of Random Matrices</td>
</tr>
<tr>
<td>PSK</td>
<td>Phase Shift Keying</td>
</tr>
<tr>
<td>QAM</td>
<td>Quadrature Amplitude Modulation</td>
</tr>
<tr>
<td>QPSK</td>
<td>Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>RCPC</td>
<td>Rate Compatible Punctured Codes</td>
</tr>
<tr>
<td>RCTC</td>
<td>Rate Compatible Turbo Codes</td>
</tr>
<tr>
<td>RS</td>
<td>Reed Solomon</td>
</tr>
<tr>
<td>RSC</td>
<td>Recursive Systematic Code</td>
</tr>
<tr>
<td>RTS</td>
<td>Regenerate Transmitted Symbol</td>
</tr>
<tr>
<td>SLC</td>
<td>Selection Combining</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SOVA</td>
<td>Soft Output Viterbi Algorithm</td>
</tr>
<tr>
<td>TCM</td>
<td>Trellis Coded Modulation</td>
</tr>
<tr>
<td>TDMA</td>
<td>Time Division Multiple Access</td>
</tr>
<tr>
<td>TDS</td>
<td>Turbo Diversity Scheme</td>
</tr>
<tr>
<td>TIE</td>
<td>Tail Interleaved Encoder</td>
</tr>
<tr>
<td>TNIE</td>
<td>Tail Not Interleaved Encoder</td>
</tr>
<tr>
<td>UEP</td>
<td>Unequal Error Protection</td>
</tr>
</tbody>
</table>
SUMMARY

This thesis investigates iterative decoding techniques applied to concatenated coding schemes. These techniques are used to avoid the complexity of maximum likelihood decoding algorithms with as little penalty as possible.

Iterative decoding techniques are applied in conjunction with special encoding strategies which combine powerful codes in such a way that allows each code to be decoded independently. The way the information is passed from one decoder to the next one, and the type of soft output decoding algorithm which is used, are keys to the improvement in performance from one iteration to the next.

An optimised maximum a posteriori decoding algorithm is described and compared with a soft output Viterbi algorithm. This optimised algorithm is used in all simulations throughout this thesis.

The new class of concatenated codes, turbo codes, is investigated for a new interleaver design which is more bandwidth efficient and improves the performance of the optimised maximum a posteriori decoding algorithm. Small interleaver sizes needed in mobile satellite channels are investigated. Both the additive white Gaussian noise channel model and Rayleigh fading channel model are used in computer simulations.

A new diversity combining technique for turbo codes on fading channels is presented. Comparisons with other classical diversity combining methods for the same codes are made.

Rate compatible turbo codes and multi-dimensional turbo codes are introduced as a way to change the coding gain for time varying channels using the same encoder/decoder structure.

Product codes and turbo codes are compared for similar parameters using the same optimised maximum a posteriori decoding algorithm. Although based on similar encoder structures, the two types of codes prove to perform differently.

Iterative decoding is applied in real time to a standard concatenated scheme in order to improve the bit error rate of the overall coding scheme. The new algorithm can be applied to any feed forward concatenated scheme.

Among the most important applications of this revolutionary technique are deep-space communications where the decoding delay is not an issue. The new diversity combining technique presented in this thesis can also improve the quality of communications over mobile satellite channels. Multi-dimensional turbo codes can achieve high coding gains at acceptable delays for speech channels. Rate compatible turbo codes can easily cope with a time varying channel and offer different levels of protection.

Iterative decoding is a very successful method of approaching channel capacity and its applications have just begun to be discovered.
STATEMENT OF ORIGINALITY

I declare that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge it does not contain any material previously published or written by another person except where due reference is made in the text.

Sorin Adrian Barbulescu
ACKNOWLEDGMENTS

I would like to express my gratitude to Dr. Steven S. Pietrobon, for guiding and inspiring my research and also for reviewing the draft versions of this thesis.

I am grateful to Professor Michael J. Miller who encouraged me to start this work and made it possible through ensuring financial support from the Institute for Telecommunications Research.

I would like to give special thanks to Dr. William Cowley who introduced me to the new area of turbo codes, and to Dr. Mark Rice and Dr. Weimin Zhang for their support in the area of diversity combining techniques for fading channels.

Many thanks go to all my colleagues at the Satellite Communications Research Centre for their help and encouragement. Also of great assistance to me, in the area of concatenated codes, was the group’s senior research leader, Mr. Ian Morrison.

I also want to thank my wife and my daughter for their patience and continuous support to overcome the difficulties of these years.
1 INTRODUCTION

The ever increasing demand for information exchange is a common characteristic of most areas of modern civilisation. The transfer of information from the source to its destination has to be done in such a way that the quality of the received information should be as close as possible to the quality of the transmitted information. A typical communication system may be represented by the block diagram shown in Figure 1.1.

The information to be transmitted can be machine generated (e.g., images, computer data) or human generated (e.g., speech). Regardless of its source, the information must be translated into a set of signals optimised for the channel over which we want to send it. The first step is to eliminate the redundant part in order to maximise the information transmission rate. This is achieved by the source encoder block in Figure 1.1. In order to ensure the secrecy of the information we want to transmit, an encryption scheme might be used. The data must also be protected against perturbations which could lead to misinterpretation of the transmitted message at the receiving end. This protection can be achieved through error control strategies: forward error correction (FEC), that is, using error correcting codes that are able to correct errors at the receiving end, or automatic repeat request (ARQ) systems. The modulator block generates a signal suitable for the transmission channel.

In this thesis we discuss the error control aspects of a communication system, although some concepts specific to the encryption and modulation blocks are given as well. From coding theory, it is known that by increasing the codeword’s length or the encoder memory, greater protection, or coding gain, can be achieved. At the same time the complexity of maximum likelihood decoding algorithms (MLDA) increases exponentially and the algorithms become difficult to implement. The increased error correction capability of long block or convolutional codes requires a very high computational effort. This led to research for new coding schemes which could replace the MLDA with simpler decoding strategies. These strategies combine more powerful codes in such a way that allows each code to be decoded separately with less complex decoders. Using this iterative technique, we can approach the performance of the MLDA.

![Figure 1.1 – A block diagram of a communication system](image-url)
The purpose of this thesis is to study iterative decoding techniques and new coding strategies which can approach, as much as possible, the channel capacity at low signal-to-noise ratios, with acceptable delay for speech transmissions.

This introductory chapter gives a brief discussion of the basic concepts involved in this technique followed by a methodology, a thesis outline, and a summary of the original contributions.

1.1 Fundamentals

The encoding techniques used in iterative decoding are those coding techniques which combine different codes in such a way that each of them can be decoded independently. We include in this category product block codes [1], concatenated codes [2], multilevel codes [3], and, most importantly, turbo codes [4]. The common feature of all these techniques is that decoding can be done sequentially, using one decoder at a time (i.e., after one code is decoded another decoder is used for the next code, and so on). This is a necessary condition for simpler decoding algorithms to replace the MLDA.

Historically, product codes represented the first attempt at achieving a higher error correction capability without the decoding complexity required by long codes. Unfortunately, the alternative decoding of the component codes did not produce the improvements in performance as expected, even though the error correction power of the product code increased as a whole.

Better results were obtain with concatenated codes. The component codes are called the inner and the outer codes. First, the decoding is done for each inner code vector. The decoded bytes are then decoded again according to the outer code used.

Multilevel coding uses several error-correcting codes with different error correcting capabilities. The transmitted symbols are constructed by combining symbols of codewords of these codes. In [5], a staged decoder for multilevel partition code passes reliability information only in one direction, from the first decoding stage to the last one.

All these codes have the disadvantage that no information is exchanged between the two decoders. This gap was filled by the new class of turbo codes, which encode the same information twice but in a different order. This made possible the information exchange between the two decoders. The more “scrambled” the information sequence is for the second encoder, the more “uncorrelated” the information exchange is. This is one of the keys that allows a continuous improvement in correction capability when the decoding process is iterated. In this thesis we study the performance of turbo codes using some new bandwidth efficient interleavers. New concepts like rate compatible turbo codes and multi-dimensional turbo codes are studied using the same interleaving techniques. We focus mainly on small interleaver sizes which allows turbo codes to be used for speech channels.

We also introduce some new applications using turbo codes in the areas of diversity combining techniques and cryptography.
In the traditional approach, the demodulator block from Figure 1.1 takes a hard decision of the received symbol and passes it to the error control decoder block. This is equivalent to deciding which of two logical values – say 0 and 1 – was transmitted. No information was passed about how reliable the hard decision was. Better results were obtained when the quantised analog received signal was passed directly to the demodulator.

This led to the development of soft input decoding algorithms which was the best solution if only one code was used. For the same reason, in the case of combining more codes as explained above, new soft output algorithms were developed in order to pass more information from one decoder to another.

Soft output decision algorithms provide as an output a real number which is a measure of the probability of error in decoding a particular bit. This can also be interpreted as a measure of the reliability of the decoder’s hard decision. This extra information is very important for the next stage in an iterative decoding process, as will be shown later. There are two important categories of soft output decision algorithms. The first category includes the maximum likelihood decoding algorithms which minimise the probability of bit error, such as the maximum a posteriori (MAP) algorithm [6]. The second category includes the maximum likelihood decoding algorithms which minimise the probability of word or sequence error, such as the Viterbi algorithm [7]. In this thesis we investigate in detail the MAP algorithm although a brief comparison between the two algorithms is presented. Our purpose is to present a simplified version of this algorithm which can be implemented in hardware. An original method to reduce delay and large memory requirements of a MAP decoder are also introduced.

The advantage of using soft outputs for the inner decoder in the context of a concatenated coding scheme with reliability information is clearly explained in [8]. The inner code’s primary function is not only to change the distribution of errors, but to effectively increase the signal to noise ratio of the received signal. The inner decoder can be thought of as a noise filter. It is shown that the channel capacity of a discrete–input real–output memoryless channel (Csoft) is greater than that for a discrete–input discrete–output (hard outputs) memoryless channel (Chard). We present here a review of those results in order to underline the importance of using soft outputs over hard outputs. We start first with the definition of the channel capacity.

In terms of digital communications systems, the maximum average mutual information which can be exchanged through a channel is called the capacity of the channel and is denoted by C. In 1948, Shannon demonstrated [9] that given a suitable channel encoder and decoder we can transmit digital information through the channel at a rate up to the channel capacity with arbitrarily small probability of error. Error control coding is the technique used to achieve this goal. Error control schemes add redundancy to the information sequence in such a way that the transmitted signals become more tolerant to the perturbations affecting the channel. The receiver uses this extra redundancy to correct the
errors introduced by the channel. Shannon’s famous capacity bound for a band–limited additive white Gaussian channel (AWGN) is

\[ C = \log_2\left(1 + \frac{E_s}{N_0}\right) \text{ bits per two–dimensional symbol (bit/sym)}, \quad (2.1) \]

where \( E_s \) is the energy per two dimensional symbol and \( N_0/2 \) is the double sided noise density. \( E_s/N_0 \) is called the signal to noise ratio (SNR) for two dimensional modulation schemes. The capacity \( C \) is plotted in Figure 1.2 from [10] for Phase Shift Keying (PSK) modulation against signal to noise ratio (SNR).

![Figure 1.2 – Channel capacity for Phase Shift Keying modulation](image)

If we normalise the capacity \( C \) by the bandwidth expansion factor \( W/T \) (\( W \) is the signal bandwidth and \( T \) is the symbol period), we obtain the bandwidth efficiency \( \eta \). This represents the number of information bits transmitted in each signalling interval assuming perfect Nyquist signaling (i.e., \( W/T = 1 \)). For reliable communication, (2.1) can be written in terms of bandwidth efficiency as

\[ \eta \leq \log_2\left(1 + \frac{E_b}{N_0}\right), \quad (2.2) \]

where \( E_b \) is the energy per information bit. This is plotted in Figure 1.3 [10].

From Figure 1.3 we see that the minimum \( E_b/N_0 \) that can be achieved is \( \ln(2) \) or \(-1.59 \) dB when \( \eta \) approaches zero. Figure 1.3 also gives us the theoretical minimum value of
Eb/N0 for a desired bandwidth efficiency. For a given bandwidth efficiency to approach a minimum value of Eb/N0 we have to use very powerful coding techniques.

![Figure 1.3 – Bandwidth efficiency for Phase Shift Keying modulation](image)

In Figure 1.4, we show a concatenated coding system where the darker area represents the inner channel which is an input–discrete memoryless channel (IDMC) [8]. We assume sufficient interleaving such that the outer channel (lighter area) is also an IDMC.

Now let us consider the input alphabet, defined as $X = \{x_0, x_1, \ldots, x_{q-1}\}$, and the output alphabet, $Y = \{-\infty, \infty\}$. The capacity of the outer channel in bits per channel use is [8]

$$C = \max_{P(x_j)} I(X; Y) = \max_{P(x_j)} \sum_{j=0}^{q-1} \int_{-\infty}^{\infty} \frac{p(y|x_j)}{p(y)} \log \frac{p(y|x_j)}{p(y)} \, dy.$$  

(1.1)

where $x_j$ are the inputs of the inner decoder with probabilities $P(x_j)$, $y_j$ are the outputs of the inner decoder with the probability density function $p(y)$. The overall outer channel is specified by the probability density function $p(y|x)$ [8].

For binary inputs, $X = \{-1, +1\}$ with equal probabilities, $P(-1) = P(+1) = 0.5$, the above equation becomes

$$C_{\text{soft}} = \frac{1}{2} \int_{-\infty}^{\infty} p(y) + 1 \log_2 \frac{p(y) + 1}{p(y)} \, dy + \frac{1}{2} \int_{-\infty}^{\infty} p(y) - 1 \log_2 \frac{p(y) - 1}{p(y)} \, dy.$$  

(1.2)
where \( p(y) = 0.5p(y|+1) + 0.5p(y|-1) \). In the case of symmetry \( p(-y|+1) = p(y|-1) \) and we obtain

\[
C_{\text{soft}} = \int_{-\infty}^{\infty} p(y| + 1) \log_2 \frac{p(y| + 1)}{p(y)} dy.
\] (1.3)

Figure 1.4 – Concatenated coding system

In Figure 1.5 we plot \( C_{\text{soft}} \) for an AWGN channel as a function of the SNR. We note that \( C_{\text{soft}} \) increases monotonically from 0 to 1 bit per symbol as the SNR increases.

Let us now consider an IDMC \([11]\) where the input alphabet is defined as \( X = \{x_0, x_1, \ldots, x_{q-1}\} \) and the output alphabet as \( Y = \{y_0, y_1, \ldots, y_{Q-1}\} \). The set of transition probabilities \( P(y_i|x_i) \) is defined as

\[
P(y_i|x_j) = P(Y = y_i|X = x_j),
\]

where \( i = 0, 1, \ldots, Q - 1 \) and \( j = 0, 1, \ldots, q - 1 \). The maximum average mutual information defines the capacity of this DMC as

\[
C = \max_{P(x_j)} I(X; Y) = \max_{P(x_j)} \sum_{j=0}^{q-1} \sum_{i=0}^{Q-1} P(x_j)P(y_i|x_j) \log \frac{P(y_i|x_j)}{P(y_i)},
\] (1.4)

where \( P(x_j) \geq 0 \) and \( \sum_{j=0}^{q-1} P(x_j) = 1 \). For a binary symmetric channel (BSC) with input probabilities \( P(-1) = P(+1) = 0.5 \), and hard outputs, the transition probabilities can be expressed as

\[
P(-1| + 1) = \int_{-\infty}^{0} p(y| + 1) dy = \hat{P} \quad \text{and}
\]

\[
P(+1|-1) = \int_{0}^{\infty} p(y| - 1) dy = \hat{P}.
\]
With this notation, the capacity is:

\[ C_{\text{hard}} = 1 + \hat{P} \log_2(1 + \hat{P}) + (1 - \hat{P}) \log_2(1 - \hat{P}). \]  

(1.5)

For an AWGN channel with zero mean and variance \( \sigma^2 \):

\[ p(y|x = m) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - m)^2}{2\sigma^2}\right) \]

where \( m = +/-1 \) and

\[ \hat{P} = Q\left(\sqrt{\frac{2E_s}{N_0}}\right), \]

\[ Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) dt. \]

In Figure 1.5 we plot (1.5) as a function of SNR. We note that \( C_{\text{hard}} \) increases monotonically from 0 to 1 bit per symbol as the SNR increases.

![Figure 1.5 – \( C_{\text{soft}} \) and \( C_{\text{hard}} \) as a function of SNR](image)

The most important conclusion from Figure 1.5 is that \( C_{\text{soft}} \) is greater than \( C_{\text{hard}} \) by approximately 2 dB at low signal to noise ratios. This is the main reason for using soft output algorithms.

The error control coding technique described in this thesis achieves a given bandwidth efficiency at lower \( E_b/N_0 \) than traditional coding methods. The iterative decoding method is applied to the new class of turbo codes, to product codes, and also to Reed–Solomon codes concatenated with convolutional codes. Iterative decoding can improve the performance of these coding schemes when extra information is passed from one iteration to the next. This is the reason why a new importance has been given to soft
output decision algorithms. Efforts were also made to simplify these algorithms to allow efficient hardware implementations.

Iterative decoding relies on a link between soft output decision algorithms, special encoding, and information transfer techniques. In the last couple of years, these concepts have been put together from a new perspective to create a more powerful tool. The link between the three created a new powerful decoding technique and led to the appearance of turbo codes, which made possible communications very close to channel capacity.

1.2 Methodology

The research methodologies applied throughout this thesis were mathematical analysis and computer simulations. Experimental methods involving real data were also used [12] but, due to confidentiality, the results could not be published in this thesis. The symbolic manipulation package Mathematica [13] was used to speed up and carry out some complicated calculations. Some simulations were performed using the Comdisco Signal Processing Worksystem (SPW) [14]. Most of the simulations were done with custom coded C language programs running on a dual processor SPARC 10 work-station. The only public domain software used was the normal distribution generator published in [15]. The results of the computer simulations were displayed using graphs in which the bit or frame error rate was plotted versus $E_b/N_0$ or SNR. Tables were also used to describe various interleaving techniques.

1.3 Thesis outline

This thesis investigates iterative decoding techniques applied to concatenated coding schemes. These techniques are used to approach channel capacity with lower complexity than maximum likelihood decoding algorithms.

Chapter 1 introduces the important concepts used in iterative decoding, followed by a brief description of research methodologies, a thesis outline, and a summary of original contributions.

In Chapter 2 we present a detailed description and optimisation of a maximum a posteriori (MAP) algorithm. The improvements are compared with other implementations in order to emphasise the reduction in complexity which can be achieved. A comparison with a soft output Viterbi algorithm shows the advantages of the MAP algorithm.

Chapter 3 begins with a literature review of turbo codes, followed by a description of the principles of iterative decoding. A real time turbo decoder and a modified MAP algorithm to suit iterative decoding are presented. An original bandwidth efficient interleaver design which improves the performance of the MAP decoder is explained. The performance of turbo codes using this new interleaving design is studied for AWGN and Rayleigh channels. The analysis is performed for low signal to noise ratios and for small interleaver sizes which are suitable for use in mobile satellite communications.
An original diversity combining technique is presented in Chapter 4. It is compared with some other classical diversity combining methods to emphasise the improved performance. Both AWGN and Rayleigh channels are studied.

Chapter 5 introduces the new concept of rate compatible turbo codes. They offer a variable coding gain using the same turbo encoder/decoder structure. Different methods to combine two different rates are presented. Multi-dimensional turbo codes are introduced as an alternative to increase the performance of turbo codes when small interleaver sizes are used.

In Chapter 6 a comparison between the performance of turbo codes and product codes is made. Two construction methods are used for product codes. Original conclusions are drawn about how the interleaver size affects the bit error ratio for product codes.

A brief description of an iterative decoding technique for concatenated codes is presented in Chapter 7. This was specially designed to reuse existing hardware for decoding concatenated codes in real time.

Chapter 8 gives the conclusions and some interesting directions for future research.

1.4 Summary of the original contributions

We summarise below the original contributions of this dissertation:

- original conclusions drawn from a comparison with the soft outputs of a Viterbi algorithm; overlapping blocks to reduce the delay of the maximum a posteriori algorithm (Chapter 2).
- a new type of interleaver for turbo codes which improves the bandwidth efficiency and maximises the performance of the MAP algorithm (Chapter 3).
- an original application of turbo codes in cryptography (Chapter 3).
- an original diversity combining technique applied to turbo codes which gives more than 0.5 dB improvement in coding gain compared to standard diversity combining techniques (Chapter 4).
- new rate compatible turbo codes which can provide a variable gain using the same encoder/decoder hardware (Chapter 5).
- new three-dimensional turbo codes and their performance for small interleaver sizes (Chapter 5).
- original conclusions drawn from a comparison between turbo codes and product codes to show why product codes do not improve their performance by increasing the interleaver size (Chapter 6).
- a simulation of an iterative decoding technique which can be applied to standard concatenated codes in real time (Chapter 7).

This thesis has made important contributions to a better understanding of the iterative decoding techniques. The application of the new ideas described in this thesis should make possible the transmission of information at even lower signal to noise ratios than before.
2 A MAP DECODING ALGORITHM

2.1 Introduction

Soft decision algorithms output a real number, the *a posteriori* probability (APP), which is a measure of the probability of a correct decision. Algorithms which minimise the symbol (or bit) error probability, also called maximum *a posteriori* (MAP) algorithms, were proposed in [16–27]. The original MAP algorithm requires substantial memory and is computationally intensive in both forward and backward recursion [6]. In this chapter, a slightly simpler MAP algorithm derived by Dr. Steven Pietrobon is presented [28]. The originality of [28] consists of the simplified relations which describe the algorithm which make possible its hardware implementation. It is estimated that its complexity is about four times the complexity of a standard Viterbi decoder.

Original conclusions are drawn from a comparison with the soft outputs of a Viterbi algorithm. A new method to reduce the decoding delay of the MAP decoder is introduced.

The Viterbi algorithm can also be modified to produce the APP [29–32] but it does not guarantee a minimal bit error ratio. However its performance is very close to that of the MAP algorithm at high SNR. A brief description of a soft output Viterbi algorithm will be presented in Section 2.10. For the particular case of iterative decoding which we are interested in, the small advantage in performance at each iteration which the MAP algorithm has over the Viterbi algorithm leads to a considerable gain for the whole iterative decoder. As noted in [26], there is a large deviation between the APP sequences produced by the two algorithms which explains the better performance of the MAP algorithm when used in an iterative decoder.

The MAP algorithm was modified to be used with turbo codes in [4], but the derivation presented was unnecessarily complex. To avoid an increase of the bit error ratio at low signal to noise ratios, the extrinsic information had to be divided by a heuristic formula. Also there was no detailed description of the interleaver design.

In [24] the derivation of the MAP algorithm is similar to ours but the definitions of the state metrics $\alpha$ and $\beta$ are different for a non-recursive code than for a recursive code. Also the final formula to find the APP for recursive codes is more complicated.

The algorithm as described here is also close to the one given in [25]. The definitions of $\alpha$ and $\beta$ in [25] are different and lead to a complicated denominator which can be eliminated. With this simplification, the algorithms become essentially the same.

The decoding approach in [26] was based on the theory of products of random matrices (PRM) which focuses on the products of independent and identically distributed random matrices [33]. This theory was applied to a generalised soft decision decoder to estimate the errors introduced due to finite decoding depth. Also, a scaling scheme to avoid range overflow of the soft outputs was presented. In [30], a comparison of the MAP algorithm with the Soft Output Viterbi Algorithm produced similar results to ours.
2.2 Recursive systematic encoder

It is known from the literature that a given convolutional code can be encoded by many different encoders and the choice of the encoding matrix can be of great importance [34]. We introduce a rate half convolutional code encoded by a systematic encoding matrix defined by (2.3) and described in Figure 2.1. This is also called a rate half recursive systematic encoder.

\[
G(D) = \left( \begin{array}{c} 1 \\ G^2(D) \\ G^1(D) \end{array} \right) = \left( \begin{array}{c} 1 \\ 1 + Dg_1^2 + D^2g_2^2 + D^3g_3^2 + \ldots, D^v g_v^2 \\ 1 + Dg_1^1 + D^2g_2^1 + D^3g_3^1 + \ldots, D^v g_v^1 \end{array} \right), \quad (2.3)
\]

where \( g_i^j \in \{0, 1\} \) for \( i = 1, 2, \ldots, v \) and \( j = 1, 2 \). The information bits \( d_k \) are the input to the encoder made of \( v \) memory cells. The output of each memory cell is multiplied by the corresponding \( g_i^j \) and added modulo two accordingly to each generator polynomial.

![Figure 2.1 – Rate half systematic convolutional encoder with feedback](image)

The implementation described in Figure 2.1 is the controller canonical form of the encoding matrix (2.3) of constraint length \( v \). The \( s_i \) with \( i = 0, 1, \ldots, v-1 \) define the state of the encoder. We investigate this type of rational encoding matrices because they are more powerful than those without feedback when used together with maximum likelihood decoding algorithms at low SNR [4].

In the next section we describe the MAP algorithm which can be applied to the more general problem of estimating the \textit{a posteriori} probabilities of the states and transitions of a Markov source at the output of a discrete memoryless channel. It can be applied to block codes whose irregular trellis structure is equivalent to a time–varying Markov source, or to convolutional codes whose regular trellis structure is equivalent to a stationary Markov source [6].
2.3 Description of the MAP algorithm

In the following derivation we consider a rate half recursive systematic code (RSC). The final results can be applied for both recursive and non-recursive systematic encoders. Let us assume that at time \( k \), the data bit \( d_k \) is present at the input of a rate half recursive systematic encoder whose state is \( S_k \). The outputs of the encoder are the uncoded data bit, \( d_k \), and the coded bit, \( Y_k \). These outputs are modulated with a QPSK modulator and sent through an AWGN channel. At the receiver end, we define the log likelihood ratio, \( L(d_k) \), as

\[
L(d_k) = \log \frac{P_r(d_k = 1 \mid \text{observation})}{P_r(d_k = 0 \mid \text{observation})},
\]

where \( P_r(d_k = i \mid \text{observation}) \), \( i = 0, 1 \) is the APP of the data bit \( d_k \). For a recursive systematic code with encoder memory order \( v \) (\( v \) memory cells), the encoder state \( S_k \) at time \( k \), is represented by the integer

\[
S_k = \sum_{i=0}^{v-1} 2^i s_i.
\]

Also suppose that the information bit sequence \( \{d_k\} \) is made up of \( N \) independent bits \( d_k \), taking values 0 and 1 with equal probability and that the encoder initial state \( S_0 \) and final state \( S_N \) are both equal to zero. In practice this can be achieved by making the last \( v \) bits drive the encoder to state zero. These \( v \) bits are called a “tail”. This will decrease the rate by a factor of \( (N-v)/N \) which can be made very close to one for long sequences. The encoder outputs at time \( k \) are translated into +/- 1 values by the following mapper

\[
a_k = 2d_k - 1, \\
b_k = 2Y_k - 1.
\]

The pair \((a_k, b_k)\) define the transmitted symbol \( C_k \) at time \( k \). The sequence of transmitted symbols is given by

\[
C_1^N = (C_1,...,C_k,...,C_N),
\]

which is the input to a discrete Gaussian memoryless channel whose output sequence is defined as

\[
R_1^N = (R_1,...,R_k,...,R_N),
\]

with \( R_k = (x_k, y_k) \) being the received symbol at time \( k \); \( x_k \) and \( y_k \) are defined as

\[
x_k = (2d_k - 1) + p_k, \\
y_k = (2Y_k - 1) + q_k,
\]

where \( p_k \) and \( q_k \) are two independent normal variables with variance \( \sigma^2 \). The APP of a decoded data bit \( d_k \) can be derived from the joint probability defined by

\[
\lambda_k^i(m) = P_r(d_k = i, S_k = m \mid R_1^N),
\]

with \( i = 0,1 \) and \( m = 0, 1, ..., 2^v-1 \). The APP of a decoded data bit \( d_k \) is thus equal to

\[
P_r(d_k = i \mid R_1^N) = \sum_m \lambda_k^i(m),
\]
where the summation is over all \( m = 0, 1, \ldots, 2^y - 1 \). From (2.4) and (2.11), the \( L(d_k) \) associated with a decoded bit \( d_k \) can be written as

\[
L(d_k) = \log \frac{\sum_{m} \lambda_k^1(m)}{\sum_{m} \lambda_k^0(m)}.
\]  

(2.12)

This represents the soft output of the MAP decoder. This can be used as an input to another decoder in a concatenated scheme or in the next iteration in an iterative decoder. Finally the decoder can make a hard decision by comparing \( L(d_k) \) to a threshold equal to zero:

- if \( L(d_k) \geq 0 \), the decoded bit is 1
- if \( L(d_k) < 0 \), the decoded bit is 0.

### 2.4 Definitions of \( \alpha \) and \( \beta \)

In order to compute the joint probability defined in (2.10), it is useful to compute the following probability functions

\[
\alpha_k^i(m) = P_t(d_k = i, S_k = m, R_k^1),
\]

(2.13)

\[
\beta_k^i(m) = P_t(R_{k+1}^N | d_k = i, S_k = m),
\]

(2.14)

where \( i \) is the value of the information bit at time \( k \). These definitions are different from those used in [6, 24, 25] in the way that both \( \alpha \) and \( \beta \) depend on the value of the information bit \( d_k \). This is very helpful later on when we can simplify the final relations to compute the Log Likelihood Ratio, \( L(d_k) \).

Using Bayes’ rule, the joint probability from (2.10) can be rewritten as follows

\[
\lambda_k^i(m) = \frac{P_t(d_k = i, S_k = m, R_k^1, R_{k+1}^N)}{P_t(R_1^N)},
\]

(2.15)

which can be further expanded to

\[
\lambda_k^i(m) = \frac{P_t(d_k = i, S_k = m, R_k^1)P_t(R_{k+1}^N | d_k = i, S_k = m, R_k^1)}{P_t(R_1^N)}.
\]

(2.16)

Taking into account the fact that events after time \( k \) are not influenced by that part of the observation up to the time \( k \), since \( d_k \) and \( S_k \) are known, (2.16) can be modified to

\[
\lambda_k^i(m) = \frac{P_t(d_k = i, S_k = m, R_k^1)P_t(R_{k+1}^N | d_k = i, S_k = m)}{P_t(R_1^N)}.
\]

(2.17)

Substituting (2.13) and (2.14) in (2.17) we obtain

\[
\lambda_k^i(m) = \frac{\alpha_k^i(m)\beta_k^i(m)}{P_t(R_1^N)}.
\]

(2.18)
The advantage of describing the joint probability as in (2.18) is that the denominator does not depend on the index \( i \) (the value of the information bit at time \( k \)). We go back to (2.12) in which we use (2.18) and obtain:

\[
L(d_k) = \log \frac{\sum_m \alpha_k^1(m) \beta_k^1(m)}{\sum_m \alpha_k^0(m) \beta_k^0(m)},
\]

(2.19)

where the denominator of (2.18) was cancelled out.

The next step is to find the relations to compute \( \alpha_k^i \) and \( \beta_k^i \). In the next sections it is shown how \( \alpha_k^i \) can be expressed as a forward recursive function (using past values of \( \alpha_k^i \) with reference to time \( k \)) and \( \beta_k^i \) as a backward recursive function (using future values of \( \beta_k^i \) with reference to time \( k \)).

### 2.5 Derivation of \( \alpha \)

We can rewrite (2.13) as

\[
\alpha_k^i(m) = P_r(d_k = i, S_k = m, R_{k-1}^{i}, R_k),
\]

where we split the received sequence \( R_k \) into the received symbol at time \( k \), \( R_k \), and the previous received symbols, \( R_{k-1}^{i} \). The above probability of events at time \( k \) can be expressed as the summation of all possible transitions from time \( k-1 \).

\[
\alpha_k^i(m) = \sum_{m'} \sum_{j=0}^{1} P_r(d_k = i, d_{k-1} = j, S_k = m, S_{k-1} = m', R_{k-1}^{i}, R_k). \quad (2.20)
\]

Using Bayes’ rule, the above expression can be modified as

\[
\sum_{m'} \sum_{j=0}^{1} P_r(d_{k-1} = j, S_{k-1} = m', R_{k-1}^{i}) \times P_r(d_k = i, S_k = m, R_k | d_{k-1} = j, S_{k-1} = m', R_{k-1}^{i})
\]

\[
= \sum_{m'} \sum_{j=0}^{1} P_r(d_{k-1} = j, S_{k-1} = m', R_{k-1}^{i}) \times P_r(d_k = i, S_k = m, R_k | d_{k-1} = j, S_{k-1} = m')
\]

since \( d_{k-1} \) and \( S_{k-1} \) completely define the path at time \( k-1 \), the received sequence \( R_{k-1}^{i} \) is irrelevant. We define

\[
\gamma_{ij}(R_k, m, m') = P_r(d_k = i, S_k = m, R_k | d_{k-1} = j, S_{k-1} = m') \quad (2.21)
\]

which will be further simplified in Section 2.8. All we need to say at this moment is that \( \gamma_{ij}(R_k, m, m') \) can be determined from the transition probabilities of the channel and the transition probabilities of the encoder trellis. We also notice that:

\[
P_r(d_{k-1} = j, S_{k-1} = m', R_{k-1}^{i}) = \alpha_{k-1}^j(m'). \quad (2.22)
\]
By substituting (2.21) and (2.22) in the double summation above, we obtain the iterative equation

$$\alpha_k^i(m) = \sum_{m'} \sum_{j=0}^{1} \alpha_{k-1}^i(m') \gamma_{i,j}(R_k, m, m'). \quad (2.23)$$

Assuming the encoder starts from state zero, we have to initialise $\alpha_0^i(m) = 0$ for all states $m$ which are different than state zero and $i = 0, 1$. Only $\alpha_0^i(0)$ will be initialized to non-zero values as described in Section 2.9.

2.6 Derivation of $\beta$

In a similar way, after the whole block of data is received, we can recursively calculate the probability $\beta_k^i(m)$ from the probability $\beta_{k+1}^i(m)$. Relation (2.14) becomes

$$\beta_k^i(m) = P_t(R_{k+1}, R_{k+2}^N|d_k = i, S_k = m)$$

$$= \sum_{m'} \sum_{j=0}^{1} P_t(d_{k+1} = j, S_{k+1} = m', R_{k+1}, R_{k+2}^N|d_k = i, S_k = m)$$

$$= \sum_{m'} \sum_{j=0}^{1} P_t(R_{k+2}^N|d_{k+1} = j, S_{k+1} = m', R_{k+1}, d_k = i, S_k = m)$$

$$\times P_t(d_{k+1} = j, S_{k+1} = m', R_{k+1}|d_k = i, S_k = m).$$

We notice that, in the first term of the product of the double summation above, we do not need the data and the encoder state at time $k$ if we have them for time $k+1$. $R_{k+1}$ also is irrelevant since the path at time $k+1$ is completely defined. The above expression can thus be written as

$$= \sum_{m'} \sum_{j=0}^{1} P_t(R_{k+2}^N|d_{k+1} = j, S_{k+1} = m')$$

$$\times P_t(d_{k+1} = j, S_{k+1} = m', R_{k+1}|d_k = i, S_k = m).$$

Similar to the definition (2.21), changing the index of time from $k$ to $k+1$ and swapping $i$ with $j$ and $m$ with $m'$, we obtain

$$\gamma_{j,i}(R_{k+1}, m', m) = P_t(d_{k+1} = j, S_{k+1} = m', R_{k+1}|d_k = i, S_k = m). \quad (2.24)$$

We also notice that

$$P_t(R_{k+2}^N|d_{k+1} = j, S_{k+1} = m') = \beta_{k+1}^j(m'). \quad (2.25)$$

Substituting (2.24) and (2.25) in the double summation above, we obtain the iterative equation

$$\beta_k^i(m) = \sum_{m'} \sum_{j=0}^{1} \beta_{k+1}^j(m') \gamma_{j,i}(R_{k+1}, m', m). \quad (2.26)$$

The initialisation of $\beta_N^i(m)$ will be described in Section 2.9.
2.7 Comparison with other MAP algorithms

In order to emphasise the simplified formulae we use in our algorithm, we present the relations used to compute \( \alpha_i \) and \( \beta_k \) in [4]

\[
\alpha^i_k(m) = \frac{\sum_{m'} \sum_{j=0}^{1} \alpha^i_{k-1}(m') \gamma_i(R_k, m', m)}{\sum_{m} \sum_{m'} \sum_{j=0}^{1} \sum_{i=0}^{1} \alpha^i_{k-1}(m') \gamma_i(R_k, m', m)}.
\]

\[
\beta_k(m) = \frac{\sum_{m'} \sum_{j=0}^{1} \beta_{k+1}(m') \gamma_i(R_{k+1}, m, m')}{\sum_{m} \sum_{m'} \sum_{j=0}^{1} \sum_{i=0}^{1} \alpha^i_{k-1}(m') \gamma_i(R_k, m', m)}.
\]

The definitions of \( \alpha \) and \( \beta \) in [25] are also different and the final relations to compute \( \alpha \) and \( \beta \) are

\[
\alpha_k(m) = \frac{\sum_{m'} \sum_{i=0}^{1} \alpha_{k-1}(m') \gamma_i(R_k, m', m)}{\sum_{m} \sum_{m'} \sum_{j=0}^{1} \alpha^j_{k-1}(m') \gamma_j(R_k, m', m)}.
\]

\[
\beta_k(m) = \frac{\sum_{m'} \sum_{j=0}^{1} \beta_{k+1}(m') \gamma_i(R_{k+1}, m, m')}{\sum_{m} \sum_{m'} \sum_{j=0}^{1} \alpha^j_{k-1}(m') \gamma_j(R_k, m', m)}.
\]

The computational advantage in using (2.23) and (2.26) instead of the above formulae is obvious, thus allowing for a less complex hardware implementation of the MAP algorithm.

2.8 Derivation of \( \gamma \) and graphical representations

The probability \( \gamma_{ij}(R_k, m, m') \) can be determined from transition probabilities of the discrete Gaussian memoryless channel and transition probabilities of the encoder trellis. Using Bayes’ rule we have

\[
\gamma_{ij}(R_k, m, m') = P_t(d_k = i, S_k = m, R_k | d_{k-1} = j, S_{k-1} = m')
\]

\[
= P_t(R_k | d_k = i, S_k = m, d_{k-1} = j, S_{k-1} = m') \times P_t(d_k = i | S_k = m, d_{k-1} = j, S_{k-1} = m') \times P_t(S_k = m | d_{k-1} = j, S_{k-1} = m').
\]
Knowing that given the information bit and the encoder state at time $k$, the path in the trellis is completely defined and that each possible information bit are equally likely, we obtain

$$\gamma_{ij}(R_k, m, m') = P_t(R_k|d_k = i, S_k = m)P_t(d_k = i)P_t(S_k = m|d_{k-1} = j, S_{k-1} = m')$$

$$= P_t(R_k|d_k = i, S_k = m)(1/2)P_t(S_k = m|d_{k-1} = j, S_{k-1} = m').$$

If from $S_{k-1} = m'$, given $d_{k-1} = j$, we go to $S_k = m$, the above relation becomes

$$\gamma_{ij}(R_k, m, m') = (1/2) \times P_t(R_k|d_k = i, S_k = m) \quad (2.27)$$

otherwise it is zero. This is why the summation over all $m'$ will disappear in (2.23) and the only surviving $m'$ will be symbolically represented as the state in which you arrive if you go backward in time from state $m$ on that branch where $d_{k-1} = j$. This is written as $S_b^j(m)$ and (2.23) can be modified as

$$\alpha_k^i(m) = \sum_{j=0}^1 \alpha_{k-1}^j(S_b^j(m))(1/2)P_t(R_k|d_k = i, S_k = m)$$

$$\alpha_k^i(m) = \frac{1}{2} P_t(R_k|d_k = i, S_k = m) \sum_{j=0}^1 \alpha_{k-1}^j(S_b^j(m)). \quad (2.28)$$

A more intuitively representation of (2.28) is given in Figure 2.2.

In Figure 2.2, $\alpha_k^i(m)$ and $\frac{1}{2} P_t(R_k|d_k = i, S_k = m)$ can be thought of as the state and branch metrics respectively of the algorithm. Looking now at $\gamma_{j,i}(R_{k+1}, m', m)$, we can modify it in a similar way

$$\gamma_{j,i}(R_{k+1}, m', m) = P_t(d_{k+1} = j, S_{k+1} = m', R_{k+1}|d_k = i, S_k = m)$$

$$= P_t(R_{k+1}|d_{k+1} = j, S_{k+1} = m', d_k = i, S_k = m) \times P_t(d_{k+1} = j|S_{k+1} = m', d_k = i, S_k = m)$$
\[ \times P_r(S_{k+1} = m'|d_k = i, S_k = m). \]

As shown before, this can be further simplified to

\[ = P_r(R_{k+1}|d_{k+1} = j, S_{k+1} = m')P_r(d_{k+1} = j)P_r(S_{k+1} = m'|d_k = i, S_k = m) \]

\[ = P_r(R_{k+1}|d_{k+1} = j, S_{k+1} = m')(1/2)P_r(S_{k+1} = m'|d_k = i, S_k = m). \]

If from \( S_k = m \), given \( d_k = i \), we go to \( S_{k+1} = m' \), the above relation becomes

\[ \gamma_{j,i}(R_{k+1}, m', m) = (1/2)P_r(R_{k+1}|d_{k+1} = j, S_{k+1} = m') \quad (2.29) \]

otherwise it is zero. The summation over all \( m' \) will disappear in (2.26) and the only surviving \( m' \) will be symbolically represented as the state in which you arrive if you go forward in time from state \( m \) on that branch where \( d_k = i \). This is written as \( S_i^j(m) \) and (2.26) can be modified as follows

\[ \beta^j_k(m) = \frac{1}{2} \sum_{j=0}^{1} \beta^j_{k+1}(S_i^j(m))P_r(R_{k+1}|d_{k+1} = j, S_{k+1} = S_i^j(m)). \quad (2.30) \]

A graphical representation of (2.30) is given in the Figure 2.3.

![Graphical representation](image)

Figure 2.3 – Graphical representation of (2.30)

In Figure 2.3 we defined

\[ \delta^j_{k+1}(S_i^j(m)) = \frac{1}{2} P_r(R_{k+1}|d_{k+1} = j, S_{k+1} = S_i^j(m)). \]

Using (2.28) and (2.30) we can compute (2.19). Before doing so, we rearrange the following probability

\[ P_r(R_k|d_k = i, S_k = m) = P_r(x_k|d_k = i, S_k = m)P_r(y_k|d_k = i, S_k = m) \]

We consider an AWGN channel with zero mean and variance \( \sigma^2 \). The above probabilities can be computed from the probability density functions when \( dx \to 0 \) and \( dy \to 0 \) as

\[ = \frac{dx}{\sqrt{2\pi}\sigma} \exp \left( -\frac{1}{2\sigma^2}(x_k - (2d_k - 1))^2 \right) \frac{dy}{\sqrt{2\pi}\sigma} \exp \left( -\frac{1}{2\sigma^2}(y_k - (2Y_k - 1))^2 \right) \]
\[
\begin{align*}
&= \frac{1}{2\pi\sigma^2} \exp \left( -\frac{x_k^2 + y_k^2}{2\sigma^2} \right) \exp \left( \frac{2(x_kd_k + y_kY_k) - x_k - y_k)}{\sigma^2} \right) dx dy \\
&= \frac{1}{2\pi\sigma^2} \exp \left( -\frac{x_k^2 + y_k^2}{2\sigma^2} \right) \exp \left( -\frac{x_k - y_k}{\sigma^2} \right) \exp \left( \frac{2(x_kd_k + y_kY_k) - x_k - y_k)}{\sigma^2} \right) dx dy \\
&= K_k \exp \left( \frac{x_kd_k + y_kY_k}{\sigma^2/2} \right) \\
&= K_k \exp \left( \frac{x_ki + y_kY_k(i, m)}{\sigma^2/2} \right) 
\end{align*}
\]

(2.31)

where $K_k$ is a constant. We wrote $Y_k$ as $Y_k(i, m)$ to underline that $Y_k$ is a function of the input bit and the encoder state. We can do a similar simplification for the probability

\[
\begin{align*}
&= K_k \exp \left( \frac{x_ki + y_kY_k(i, m)}{\sigma^2/2} \right) \\
&= K_k \exp \left( \frac{x_kd_k + y_kY_k}{\sigma^2/2} \right) \\
&= K_k \exp \left( \frac{x_kd_k + y_kY_k}{\sigma^2/2} \right) \\
&= K_k \exp \left( \frac{x_ki + y_kY_k(i, m)}{\sigma^2/2} \right) \\
&= K_k \exp \left( \frac{x_kd_k + y_kY_k}{\sigma^2/2} \right) \\
&= K_k \exp \left( \frac{x_ki + y_kY_k(i, m)}{\sigma^2/2} \right)
\end{align*}
\]

(2.32)

Using relations (2.31) and (2.32) in (2.28) and (2.30), we obtain

\[
\begin{align*}
\alpha_k^i(m) &= C_k^{\alpha} \exp \left( \frac{x_ki + y_kY_k(i, m)}{\sigma^2/2} \right) \sum_{j=0}^{1} \alpha_{k-1}^j(S_b^j(m)) \\
\beta_k^i(m) &= C_k^{\beta} \sum_{j=0}^{1} \beta_{k+1}^j(S_b^j(m)) \exp \left( \frac{x_k + j + y_k + Y_k(j, S_b^j(m))}{\sigma^2/2} \right)
\end{align*}
\]

(2.33)

(2.34)

where $C_k^{\alpha}$ and $C_k^{\beta}$ are constants. Since these constants will cancel out when used in (2.19) we can redefine $\alpha_k^i(m)$ and $\beta_k^i(m)$ as

\[
\begin{align*}
\hat{\alpha}_k^i(m) &= \exp \left( \frac{x_ki + y_kY_k(i, m)}{\sigma^2/2} \right) \sum_{j=0}^{1} \hat{\alpha}_{k-1}^j(S_b^j(m)) \\
\hat{\beta}_k^i(m) &= \sum_{j=0}^{1} \hat{\beta}_{k+1}^j(S_b^j(m)) \exp \left( \frac{x_k + j + y_k + Y_k(j, S_b^j(m))}{\sigma^2/2} \right)
\end{align*}
\]

(2.35)

(2.36)

With these equations we can finally compute the Log Likelihood Ratio as

\[
L(d_k) = \log \frac{\sum_m \hat{\alpha}_k^1(m)\hat{\beta}_k^1(m)}{\sum_m \hat{\alpha}_k^0(m)\hat{\beta}_k^0(m)}.
\]

(2.37)
2.9 Steps of the algorithm

The steps of the decoding algorithm are

- **Step 1:** Initialise the following probabilities for \( i = 0, 1 \):
  \[
  \alpha^i_0(S^i_b(0)) = 1 \quad \text{and} \quad \alpha^i_0(S^i_b(m)) = 0, \quad \text{for} \quad m \neq 0
  \]
  \[
  \beta^i_N(S^i_b(0)) = 1 \quad \text{and} \quad \beta^i_N(S^i_b(m)) = 0, \quad \text{for} \quad m \neq 0
  \]

- **Step 2:** After the whole sequence of \( N \) symbols is received, for each observation \( R_k \), for \( i = 0, 1 \) and all states \( m \), compute the following probability
  \[
  \beta^i_k(m) = \sum_{j=0}^{1} \beta^j_{k+1}(S^j_b(m)) \exp \left( \frac{x_k i j + y_{k+1} Y_{k+1}(j, S^j_b(m))}{\sigma^2/2} \right).
  \]

- **Step 3:** For each observation \( R_k \), for \( i = 0, 1 \) and all states \( m \), compute the \( \hat{\alpha}^i_k(m) \) probabilities and \( L(d_k) \) using
  \[
  \hat{\alpha}^i_k(m) = \exp \left( \frac{x_k i + y_k Y_k(i, m)}{\sigma^2/2} \right) \sum_{j=0}^{1} \hat{\alpha}^j_{k-1}(S^j_b(m))
  \]
  \[
  L(d_k) = \log \frac{\sum_{m} \hat{\alpha}^1_k(m) \hat{\beta}^1_k(m)}{\sum_{m} \hat{\alpha}^0_k(m) \hat{\beta}^0_k(m)}.
  \]

2.10 Comparison between the MAP and the Soft Output Viterbi Algorithms

It is well known that the Viterbi algorithm minimises the sequence error probability. There are a few variations such as the weighted output Viterbi algorithms [29, 35] and SOVA [8, 31]. All of these algorithms have in common the derivation of a soft output for each symbol from the metric difference of the competing paths. For each path decision there is a backward recursion to update the reliability values along that path for all symbols. In [26], a comparison was made between the MAP algorithm and the SOVA proposed in [30] (SOVA–30). The conclusion was that the SOVA–30 output \( \hat{P}_e \) underestimates the probability of decision error \( P_e (\hat{P}_e < P_e) \) for a majority of bits. The optimistic estimate of \( \hat{P}_e \) is increasingly inaccurate for decreasing SNR. This is the price to be paid for a less complicated hardware implementation of SOVA against the MAP algorithm. Another conclusion was that even at low SNR’s, where \( \hat{P}_e \) has large deviations from \( P_e \), the time domain sequences are similar.

We came to similar conclusions comparing our MAP algorithm with the soft output Viterbi algorithm proposed in [31] (SOVA–31). We have to acknowledge that, unlike the
MAP algorithm, SOVA–31 employs only forward recursion and requires no storage for the observation sequence.

In this section we briefly describe the simplified SOVA–31 algorithm and we compare the output sequence with the MAP output sequence in the time domain. We assume that two paths diverge at time \( k-d \) and remerge at time \( k+1 \). The survivors for the two states A and B at time \( k \), from which the two paths will remerge are

\[
V_{Ak} = (u_{1,(k-1)}, u_{1,(k-2)}, \ldots, u_{1,(k-d)}, u_{(k-d-1)} , \ldots)
\]
\[
V_{Bk} = (u_{2,(k-1)}, u_{2,(k-2)}, \ldots, u_{2,(k-d)}, u_{(k-d-1)} , \ldots)
\]

where \( u_{(k-d-1)} \) is the common soft output before diverging. At time \( k+1 \), the path metric for the path coming from state A is

\[
m_1 = M_{A,k} + |z_k - y_{1,k}|^2
\]

where \( M_{A,k} \) is the path metric at time \( k \) for the path coming from state A, \( z_k \) is the observation at time \( k \) and \( y_{1,k} \) is the desired output for path 1. Similar definition applies for the path coming from state B

\[
m_2 = M_{B,k} + |z_k - y_{2,k}|^2
\]

We assume \( m_1 < m_2 \). The hard output Viterbi decoder would generate

\[
V_1,(k+1) = (u_{1,k}, u_{1,(k-1)}, u_{1,(k-2)}, \ldots, u_{1,(k-d)}, u_{(k-d-1)} , \ldots).
\]

For the SOVA–31 output we first define

\[
\Delta = \frac{P_r(\text{path1})}{P_r(\text{path1}) + P_r(\text{path2})}
\]

which for an AWGN channel can be approximated to

\[
\Delta = \frac{\exp(-\frac{m_1}{2\sigma^2})}{\exp(-\frac{m_1}{2\sigma^2}) + \exp(-\frac{m_1}{2\sigma^2})}
\]

With these notations the SOVA–31 output becomes

\[
V_1,(k+1) = ((1 - \Delta)u_{1,k} + \Delta u_{2,k}, \ldots, (1 - \Delta)u_{1,(k-d)} + \Delta u_{2,(k-d)} , u_{(k-d-1)} , \ldots)
\]

From the above relation the following conclusions can be drawn

- if the hard decisions for the decoded bits at time \( i \) for the two paths are the same, the soft output will not be different from the hard output.
- if the hard decisions for the decoded bits at time \( i \) for the two paths are different, each decision along the decoding depth adds some uncertainty to the decoded bit.

In Figure 2.4 we plot the two soft outputs obtained with the MAP decoder and the SOVA–31 decoder. It is interesting to notice the higher dynamic range for the MAP outputs compared with the SOVA–31 outputs. Also when the SOVA–31 output is almost constant for a sequence of bits, say around +1, the MAP output for the same bits varies considerably. If a hard limiter is used, all these become unimportant and there is not much difference between the two hard outputs.
One could take advantage of the more sensitive outputs of the MAP decoder in an iterative decoder as shown in the next chapters. This application of the MAP decoder can provide better bit error ratios than a SOVA decoder, which justifies the use of a more computational intensive algorithm.

When implemented in hardware, further modifications of the MAP algorithm can be used [28]. Instead of working with probabilities, the logarithm of those probabilities was used to avoid overflow or underflow. This approach together with further simplifications is under current investigation [36].

The performance of both algorithms for the standard rate half four state systematic convolutional code \((g^1 = 7_8, g^2 = 5_8)\) is given in Table 2.1. The BER values are almost the same for an \(E_b/N_0\) in the range 0 dB to 5 dB. The simulations were performed for at least 1500 errors in each case. In (2.38) and (2.39), the variance of the noise needs to be determined. For the energy per two dimensional symbol normalised to one, we use the formula

\[
\sigma^2 = \left( \frac{2\eta E_b}{N_0} \right)^{-1}
\]

where \(E_b\) is the energy per information bit, \(\eta\) is the number of information bits per two dimensional symbol and \(N_0\) is the single sided noise power spectral density. For a QPSK
constellation where $I = +/- 1$ and $Q = +/- 1$, the energy per symbol is 2, so the variance of the noise will be

$$\sigma^2 = 2 \left( \frac{2\eta E_b}{N_0} \right)^{-1} = \left( \frac{\eta E_b}{N_0} \right)^{-1}$$

Table 2.1 – BER for a four state code using the MAP decoder and SOVA decoder

<table>
<thead>
<tr>
<th>$E_b/N_0$(dB)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>7.81e−2</td>
<td>4.03e−2</td>
<td>1.53e−2</td>
<td>4.27e−3</td>
<td>9.36e−4</td>
<td>1.28e−4</td>
</tr>
<tr>
<td>SOVA</td>
<td>7.93e−2</td>
<td>4.09e−2</td>
<td>1.56e−2</td>
<td>4.26e−3</td>
<td>9.48e−4</td>
<td>1.28e−4</td>
</tr>
</tbody>
</table>

### 2.11 MAP decoding with overlapping blocks

One important observation is that for MAP decoding the final state of the encoder must be known. This means that after one block of data a certain pattern of bits must be appended such that the encoder state becomes zero (or any other predefined value). This decreases the effective transmission rate and increases the decoding delay because the decoder has to wait until the whole block of data is received. One way to avoid this problem is to use overlapping blocks at the receiver end as described in Figure 2.5. Here we show a block of data split into three parts which reduces the decoding delay to a third. The price we pay is the increased number of computations per each block of data.

![Figure 2.5 – MAP decoding using overlapping blocks](image-url)
The backward recursion starts from an unknown state before the whole block of data is received. We assume equal state metrics for all states. We define the error $\varepsilon$ as the absolute value

$$\varepsilon = \frac{|L_{ks} - L_{us}|}{L_{ks}}$$

where $L_{ks}$ is the soft output produced when the backward recursion starts from a known state, and $L_{us}$ is the soft output produced when the backward recursion starts from an unknown state. The first decoded values, $L_{us}$, are erroneous, but, gradually, they become closer to $L_{ks}$. Similar, the value of $\varepsilon$ for the first decoded outputs is significant and decreases with each new decoding step. In our computer simulations we considered $\lambda$ steps, where $\lambda$ was chosen equal to ten times the encoder memory. For this value of $\lambda$, the error $\varepsilon$ becomes less than two percent after $\lambda$ steps. In Figure 2.5 we start the decoding after receiving only a third of the data block. The first $\lambda$ soft outputs are ignored as being unreliable due to the backward recursion. Those $\lambda$ outputs will be recomputed when the second third of the data block is received. The last third of the data block ends in the right state, so all the soft outputs are reliable. The block definition is not useful any more and can be ignored. The only requirement is that at the end of the transmission, some extra bits should be appended to drive the encoder in state zero. The decoding process becomes closer to the Viterbi decoding where there is a comparable delay due to the trace back operations. The only drawback of this method is that we compute twice a block of size $\lambda$ symbols (the dashed blocks in Figure 2.5) for each pair of overlapping blocks.

### 2.12 Conclusions

This chapter presented a simplified MAP decoding algorithm. This algorithm minimises the symbol (or bit) error probability and outputs a real number (soft decision) which is a measure of the probability of decoding a particular bit correctly. The decoding process is performed block by block employing a forward and a backward recursion. The branch and state metrics of the algorithm can be calculated using simplified expressions. The complexity of this simplified MAP algorithm is estimated in [28] to be four times the complexity of a Viterbi decoder.

Although the output sequences for the MAP and the Viterbi decoder have surprisingly similar patterns in the time domain, the soft outputs from the MAP decoder have larger deviations when compared with the soft outputs from a Viterbi decoder. The similar time domain patterns explain why the performance of the two decoding algorithms is so close, while the difference in their deviations makes the MAP algorithm more suitable in iterative decoding.

The advantages of the MAP algorithm can be summarised as follows:

- minimises the symbol (bit) error probability
- provides soft outputs
has a higher dynamic range than a Viterbi algorithm which is very important in an iterative decoding scheme.

The disadvantages of the MAP algorithm are:

- the time delay due to the necessity of receiving the whole block of data before the decoding process can start (although this can be overcome with increased complexity as presented in the previous section)
- it requires an accurate estimate of the noise variance
- increased complexity

Although the inherent delay of the MAP algorithm restricts its area of application, it is the best algorithm which can be used in an iterative decoding scheme. The need for an accurate estimate of the noise variance can be used to advantage in an encryption scheme to be presented in Section 3.10.

In the following chapters (except Chapter 7) we will only use the MAP algorithm. When this algorithm is used in conjunction with turbo codes, it will allow us to operate at $E_b/N_0$ levels lower than with any other forward error correction scheme used to date.
3 ITERATIVE DECODING OF TURBO CODES

3.1 Introduction

After the description of the MAP algorithm in the previous chapter, Chapter 3 introduces the concept of turbo codes and how to use the MAP algorithm in a turbo decoder. This is followed by a detailed analysis of the performance of turbo codes using new interleaver design criteria for small data frames.

Section 3.2 is a review of the available published work related to turbo codes.

Sections 3.3 and 3.4 present a review of the definition of turbo codes and the principles of iterative decoding. Section 3.5 describes the application of the MAP decoding algorithm in a turbo decoder.

Two new criteria for designing more bandwidth efficient and better interleavers for small frame sizes are presented in Section 3.6. Their performance is analysed for AWGN channels in Section 3.7 and for fading channels in Section 3.8.

Section 3.9 gives guidelines for interleaver design and in Section 3.10 a possible application of turbo codes to cryptography is highlighted.

3.2 Literature review

The definition of turbo codes was given for the first time in [4]. They represent a particular class of parallel concatenation of two recursive systematic convolutional codes (see Section 3.3). Here we review a few published studies on the performance of turbo codes for AWGN and fading channels, for continuous or short frame transmission systems using the MAP or the Soft Output Viterbi decoding algorithms.

In [4], the first rate half turbo code with memory order 4, generators (37, 21)₈, pseudo–random interleaver matrix of 256 by 256, was investigated for an AWGN channel. Using a modified Bahl \textit{et al.} algorithm [6], a bit error rate of $1.0 \times 10^{-5}$ at $E_b/N_0 = 0.7$ dB was achieved after 18 iterations. This result is based on counting only 80 errors, which is a very small number from a statistical point of view. A very similar performance is claimed to be achieved using a soft output Viterbi algorithm [32], the complexity of which is approximately twice that of the Viterbi algorithm.

In [37], the bit error rate (BER) and the frame error rate (FER) for short frame transmission (192 bits) over an AWGN channel was investigated. The best pseudo–random interleaver from about 800 tested was selected, and a MAP estimator as in [4] with proper termination of the second code was used. A BER of $1.2 \times 10^{-3}$ and a FER of $2.5 \times 10^{-2}$, was achieved at an $E_b/N_0 = 2.0$ dB after 10 iterations. These results are reported to be 1.2 to 1.7 dB better than the performance achieved with a non-systematic convolutional code with the same memory order.

In [38] it was concluded that for BER $> 10^{-3}$ and large FER of $10^{-2}$, the effect of the chosen interleaver on the performance of the corresponding turbo code is almost negligible.
in the case of short frame transmission (192 bits). However for a slightly larger block size (399 bits) the interleaver type can be optimised to double the performance of the turbo code [39].

The application of turbo codes to a TDMA/CDMA mobile radio system was investigated in [40]. Gains of 0.4 to 1.2 dB over non–systematic convolutional codes can be achieved by using turbo codes for the considered mobile radio system using joint detection with coherent receiver antenna diversity. Bad urban and typical urban models specified by COST 207 [42] were used. The complexity of the decoder is increased because it is necessary to determine the variance of the disturbance, arising due to noise and channel estimation errors, at the input of the decoder.

High spectral efficiency modulation schemes using turbo codes for AWGN and Rayleigh channels were presented in [43]. An AWGN channel was assumed with the encoder using a non-uniform interleaver (64 by 64) and 16QAM modulation. Three iterations were used in the decoder. A BER of $10^{-6}$ was achieved at an $E_b/N_0 = 4.35$ dB for 2 bit/s/Hz spectral efficiency and at an $E_b/N_0 = 6.2$ dB for 3 bit/s/Hz spectral efficiency. For a Rayleigh channel with four decoder iterations and a second interleaver, a BER of $10^{-5}$ was achieved at an $E_b/N_0 = 6.5$ dB for 2bit/s/Hz spectral efficiency and an $E_b/N_0 = 9.6$ dB for 3bit/s/Hz spectral efficiency. One important conclusion is that a turbo decoder optimised for an AWGN channel, is also optimal for a Rayleigh channel if the logarithms of the likelihood ratio associated with each decoded bit is modified to include the Rayleigh random variable. In [44], two Ungerboeck codes in combination with TCM were used as component codes in a turbo encoder. This proved to give an extra 0.5 dB improvement.

In [45] and [25], a study of the performance of turbo codes as a function of the interleaver size for an AWGN channel and independent Rayleigh fading channel was performed. A memory four rate half turbo code with a soft output Viterbi decoder were used. At an $E_b/N_0 = 2$ dB, the BER changes from $1 \times 10^{-2}$ to $2 \times 10^{-5}$ when the frame size increases from 100 to 10,000. The memory two turbo code for a BER $< 10^{-4}$ is even better than the memory four turbo code for the Rayleigh fading channel.

Benedetto and Montorsi have proposed an analytical upper bound to the average performance of turbo codes [49–51]. Their investigation is based on the input–redundancy weight enumerating function and using an “uniform” interleaver. This uniform interleaver is defined as a probabilistic device which maps a given input word of weight w into all distinct permutations $\binom{k}{w}$ with equal probability $\binom{k}{w}^{-1}$. An important observation used in the performance analysis is that the two sequences which enter the two encoders share the same information weight.

Another interesting concept closely related to turbo codes is the search for a method of concatenating the convolutional codes in such a way that the combination of coding and
interleaving results in codewords from all the component codes being present in the final output [53–55]. A time division interleaver is introduced as in Figure 3.1.

![Figure 3.1](image)

Figure 3.1 – Convolutional encoder including m–fold time division interleaving

A new type of interconnection is presented in Figure 3.2. CE1(m1) is a rate 2/3 encoder with m1–fold interleaving for code one. CE2(m2) is also a rate 2/3 encoder with m2–fold interleaving for code number two. c1q(m1) is the q–th valid codeword for code one with m1–fold interleaving and c2q(m2) is the q–th valid codeword for code two with m2–fold interleaving.

![Figure 3.2](image)

Figure 3.2 – Two tier coding with rate 2/3 component codes

The above concatenation scheme gives very good results, close to that of the turbo coding scheme. In [54], as in [55], the decoding strategy is to overlay a two segment processing window onto the received bits. The first segment of the window identifies the portion of the bits that will be decoded and the second segment acts as a future view for the processing to be performed. The forward and backward recursion of the MAP processing are performed over the entire window. In [58], multistage decoding is used for punctured convolutional codes and parity check codes in an AWGN channel. The incoming information bit stream is divided into three sub–streams that are separately encoded. The use of the reliability information together with interleaving between the coded bit streams at every coding level allows for a substantial improvement of the BER after several iterations. It is shown that the performance of trellis coded modulation (TCM) schemes can be reached with a simple multilevel scheme which has uncoded bits in its third level. A
coding gain of 1.4 dB compared with TCM at almost the same complexity is achieved when the third level is encoded.

Weight distributions and bounds are discussed in [57]. An optimum criteria for any real constructible interleaver for turbo codes is given. The union upper bounds on the bit error rate for turbo codes based on block codes and convolutional codes are also presented. We describe briefly this optimum criteria used in conjunction with the “fully optimal interleaving” for block codes.

We consider a turbo code based on two identical block codes \((n,k)\) for which we divide all \(2^k - 1\) non–zero codewords into \(k\) groups so that the \(i^{th}\) group consists of \(\binom{k}{i}\) codewords of weight \(i\) in the information part, where \(i = 1, 2, \ldots, k\). A turbo codeword is defined as a concatenation of the \(k\) information bits, the \(n-k\) first redundancy bits, and the \(n-k\) second redundancy bits obtained after interleaving within each group. The interleaving has to produce turbo codewords with overall weights \((W)\) as large as possible. One way to achieve this is to order the codewords with non–decreasing weights of the first redundancy part and associate them with the second redundancy part according to the rule

\[
W(i, l) = i + j(i, l) + j(i, \binom{k}{i} - l + 1)
\]

where \(j(i, l), l = 1, 2, \ldots, \binom{k}{i}\), is the weight of the redundancy part of the \(i^{th}\) codeword in the \(i^{th}\) ordered group. This association rule is shown in Figure 3.3 which is reproduced from [57].

![Figure 3.3 – Optimal interleaving for the \(i^{th}\) group](image)

An interleaver which achieves the same weight distribution as the above rule is called a fully optimal interleaver.

The \(W(i, l)\) is considered as a random variable of \(l\). The optimal criteria is the minimisation of the variance of this random variable. It is proved that for a fully optimal interleaver, the variance reaches its minimal possible value and the weight \(W(i, l)\) in the \(i^{th}\) group is maximal.
3.3 Description of the turbo encoder

In Figure 3.4 we present a generic turbo encoder in two dimensions. The input sequence of the information bits is organised in blocks of length N. The first block of data will be encoded by the ENC block which is a rate half recursive systematic encoder. The same block of information bits is interleaved by the interleaver INT, and encoded by ENC| which is also a rate half systematic recursive encoder. The coded output of each encoder is the output of each encoder block.

\[ \text{dk} \rightarrow \text{INT} \rightarrow \text{d}_i \rightarrow \text{ENC}^{-} \rightarrow Y^{-}_k \rightarrow \text{Y|} \rightarrow \text{ENC|} \rightarrow Y|_k \rightarrow X_k \]

Figure 3.4 – Turbo encoder

Due to the similarities with product codes [52], we can also call the ENC block as the encoder in the “horizontal” dimension and the ENC| block as the encoder in the “vertical” dimension. The interleaver block, INT, changes the input order of the information bits for the second encoder following some criteria which will be explained in Section 3.6. Ignoring for the moment the delay for each block, we assume both encoders output data simultaneously. The switch alternatively selects one of the coded bits produced by the two systematic encoders. The output of the turbo encoder is the pair \((X_k, Y_k)\) which can be BPSK or QPSK modulated and sent through the channel as in Figure 3.5 and Figure 3.6.

\[ X_k = 2X_k - 1 \]
\[ Y_k = 2Y_k - 1 \]

Figure 3.5 – QPSK modulator
We notice that the two encoders do not have to be identical. The only requirements are to be systematic and of rate half. The puncturing implemented by the switch could also have a different pattern than the alternate selection of the coded outputs of the two encoders.

3.4 Principles of iterative decoding

The decoding techniques used for product codes can also be used for turbo codes. Let us consider the two dimensional case in which \( k_1 \times k_2 \) information bits are ordered in a rectangular matrix. Let \( C_1 \) be a linear systematic code \((n_1, k_1)\) in one dimension and \( C_2 \) a linear systematic code \((n_2, k_2)\) in the second dimension which are used to encode the same data. This two-dimensional code is a direct product of \( C_1 \) and \( C_2 \). If the code \( C_1 \) has minimum weight \( d_1 \) and the code \( C_2 \) has minimum weight \( d_2 \), the minimum weight of the product code is \( d_1 \times d_2 \) [52]. From information theory we know that the optimum decoding method is that which decodes the whole product code. However for practical considerations, i.e., complexity, we will investigate the iterative decoding method: first decode the code in one dimension and then decode the code in the second dimension. For an error pattern to be correctable it is necessary that after the correction in one dimension, the errors which remain should be correctable in the second dimension. An improvement is obtained by iterating the decoding process. However there are some error patterns of weights \( w_e \) in the range

\[
\lfloor \text{max}(d_1, d_2) - 1 \rfloor / 2 < w_e < \lfloor (d_1 \times d_2 - 1) / 2 \rfloor
\]

which can not be corrected by each individual code but could be corrected by the product code.

The main purpose of the interleaver is that for the above type of error patterns, after the correction in one dimension the remaining errors should be spread such as to become correctable error patterns in the second dimension, so the error correction capability of the turbo code will approach that of the product code (\( \lfloor (d_1 \times d_2 - 1) / 2 \rfloor \)).

The principles of iterative decoding were clearly presented for the first time in [25]. We will briefly present the conclusions of that paper in order to understand how to pass the information from one iteration to another using the MAP algorithm. Let us assume we have
a soft–in/soft–out decoder as in Figure 3.7 which can provide at the output the Log Likelihood Ratio for each decoded bit. The three inputs are defined as follows:

- \( L_a(d_k) \) is the *a priori* value for bit \( d_k \)
- \( x_k, y_k \) are the received uncoded and coded bits affected by noise

The Log Likelihood Ratio of the information bit \( d_k \) conditioned only by the received symbol \( x_k \) is

\[
L(d_k|x_k) = \log \frac{P_r(d_k = +1|x_k)}{P_r(d_k = -1|x_k)}
\]  
(3.1)

which can be expressed as

\[
L(d_k|x_k) = \log \frac{p(x_k|d_k = +1)}{p(x_k|d_k = -1)} + \log \frac{P_r(d_k = +1)}{P_r(d_k = -1)}.
\]  
(3.2)

For an AWGN channel, the probability density function is

\[
p(x) = \frac{1}{\sqrt{2\pi \sigma}} \exp\left(-\frac{1}{2\sigma^2}(x - m)^2\right)
\]  
(3.3)

where \( m \) and \( \sigma \) are the mean and the variance of the noise respectively. The conditional pdf can be written

\[
p(x_k|d_k = +1) = \frac{1}{\sqrt{2\pi \sigma}} \exp\left(-\frac{1}{2\sigma^2}(x - 1)^2\right)
\]  
(3.4)

\[
p(x_k|d_k = -1) = \frac{1}{\sqrt{2\pi \sigma}} \exp\left(-\frac{1}{2\sigma^2}(x + 1)^2\right).
\]  
(3.5)

We can now compute

\[
\log \frac{p(x_k|d_k = +1)}{p(x_k|d_k = -1)} = \frac{2}{\sigma^2} x_k = 4 \frac{E_s}{N_0} x_k.
\]  
(3.6)

Using the above result in (3.2) we obtain

\[
L(d_k|x_k) = 4 \frac{E_s}{N_0} x_k + \log \frac{P_r(d_k = +1)}{P_r(d_k = -1)}
\]  
(3.7)

We make the following notations:

\[
L_c = 4 \frac{E_s}{N_0}
\]  
(3.8)

which is called the reliability value of the channel, and
which is called the a priori value for bit $d_k$. Using the above notations we obtain

$$L(d_k|x_k) = L_{c}x_k + L_{a}(d_k).$$

If we consider the Log Likelihood Ratio of the information bit $d_k$ conditioned by the whole observation sequence, as defined in (2.4)

$$L(d_k) = \log \frac{P_r(d_k = 1|\text{observation})}{P_r(d_k = 0|\text{observation})}$$

we can now rewrite it in a form similar to (3.10)

$$L(d_k) = L_{c}x_k + L_{a}(d_k) + L_{e}(d_k)$$

where $L_{e}(d_k)$ is the extrinsic information derived from all $y_k$ and all $x_{k'}$, $L_{a}(d_k)$, $k'$ different from $k$. For the first decoder in say the horizontal dimension, (first half of an iteration), there is no a priori information ($L_{a}(d_k) = 0$), so we have

$$L(d_k) = L_{c}x_k + L_{e}(d_k).$$

For the decoder in the vertical dimension, (second half of an iteration), the a priori value is the extrinsic information from the decoder in the horizontal dimension and its output can be expressed as

$$L(d_k) = L_{c}x_k + L_{e}(d_k) + L_{e}^{-1}(d_k).$$

Now, the $L_{e}(d_k)$ becomes the a priori value $L_{a}(d_k)$ for the next iteration and so on. At the beginning the extrinsic values are statistically independent and the gain from one iteration to another is high. However, since the same information is used, the improvement after a few iterations becomes very small. After the last iteration, the soft output decision will be the sum of the last two extrinsic values and $L_{c}x_k$. The decoder will make a hard decision by comparing the soft output to a threshold equal to zero

- if $L(d_k) \geq 0$, the decoded bit is 1
- if $L(d_k) < 0$, the decoded bit is 0.

### 3.5 Application of the MAP algorithm in iterative decoding

For a better understanding on how an iterative decoder works, we present in Figure 3.8 a turbo encoder including the time delay necessary for synchronisation.

The DELAY1 block is a delay line equal to the delay introduced by one MAP decoder. The output of the turbo encoder from Figure 3.8 is QPSK modulated and sent through an AWGN channel as described in (2.8) and (2.9), and shown in Figure 3.6. Given the principles of iterative decoding presented in the previous section, the detailed implementation of an iterative decoder is described in Figure 3.9. This represents only one iteration of the decoding process. The whole block can be repeated for the number of desired iterations. We assume that the reliability value of the channel ($L_{c} = \frac{4E_s}{N_0}$) can be
estimated. The received $x_k$ and $y_k$ symbols are then multiplied by $L_c$ and input to the iterative decoder.

Each elementary decoder is implemented as a two input–one output block. One input is the received coded symbol multiplied by $L_c$. The second input is the sum between the extrinsic information output from the previous decoder and the received information symbol multiplied by $L_c$. For the first iteration only, the a priori input to $\text{DEC}^-$ is always zero, no a priori information being available. The output is the Log Likelihood Ratio as defined in (2.12).

Each elementary decoder, $\text{DEC}^-$ and $\text{DEC}^\dagger$ uses the simplified MAP algorithm. In order to use the a priori information in an iterative decoder, we need to make the following notations

\begin{align*}
\Delta^1_k &= \exp(L_a(d_k)) \\
\Delta^0_k &= 1
\end{align*}

where $L_a(d_k)$ is the a priori value for information bit $d_k$.

Using the notations from Figure 3.9, the equations of the algorithm as defined in Section 2.9 will change to

\begin{align*}
\tilde{\alpha}^i_k(m) &= \Delta^1_k \exp \left( \frac{2(x_k^i + y_k Y_k(i, m))}{\sigma^2} \right) \sum_{j=0}^{1} \alpha^i_{k-1}(S^i_b(m)) \\
&= \exp[(L_a(d_k) + L_c x_k)i + L_c y_k Y_k(i, m)] \sum_{j=0}^{1} \alpha^i_{k-1}(S^i_b(m)) \\
\tilde{\beta}^i_k(m) &= \sum_{j=0}^{1} \beta^i_{k+1}(S^i_f(m)) \Delta^1_{k+1} \exp \left( \frac{2(x_k+1j + y_k+1 Y_k+1(j, S^i_f(m)))}{\sigma^2} \right) \\
&= \sum_{j=0}^{1} \beta^i_{k+1}(S^i_f(m)) \exp[(L_a(d_k+1) + L_c x_{k+1})j + L_c y_{k+1} Y_{k+1}(j, S^i_f(m))] 
\end{align*}
With this modifications we redefine the soft outputs as

\[ L(d_k) = \log \frac{\sum_m \tilde{\alpha}_k(m) \tilde{\beta}_k(m)}{\sum_m \tilde{\alpha}_k^0(m) \tilde{\beta}_k^0(m)}. \]  

(3.18)

\begin{align*}
\Sigma & \quad \text{DEC}^- \quad \text{DEC}^\dagger \quad \text{DEC}^- \quad \text{DEC}^\dagger \\
+ L_c x_k + L_a(d_k) & \quad \text{delay} \quad \Sigma & \quad L_c x_j + L_c(d_j) & \quad \Sigma & \quad L_c x_i + L_a(d_i) \\
\text{DEC}^- & \quad \text{INT} & \quad \text{DEC}^\dagger & \quad \text{DEINT} \\
\text{delay} & \quad \Sigma & \quad \text{delay} & \quad \Sigma & \quad \text{DEINT} \\
L_c y_k & \quad L_c y_k & \quad L(d_j) & \quad L(d_j) & \quad L(d_{k-\Delta}) \\
L_c x_k & \quad \text{delay} \Delta & \quad L_c x_k & \quad \text{delay} \Delta & \quad L_c y_k & \quad \text{delay} \Delta \\
\end{align*}

DEC$^-$ is the decoder in the “horizontal” dimension
DEC$^\dagger$ is the decoder in the “vertical” dimension
INT is the interleaver block
delay is the delay through DEC$^-$ or DEC$^\dagger$
delay $\Delta$ is the delay for one iteration of the decoding process
$La(d_k)^-$ is the a priori information at the input of the k–th iteration
$Lc(d_j)^-$ = $La(d_j)^\dagger$ the extrinsic information output by DEC$^-$ and is equal to the a priori information input to DEC$^\dagger$
$x_k$ is the received uncoded information sample
$y_k$ is the received coded sample
$L(d_j)$ is the Log Likelihood Ratio associated with information bit $d_j$
$Lc(d_{k-\Delta})$ is the extrinsic information at the output of the k–th iteration
$x_{k-\Delta}$ is the delayed received uncoded information sample
$y_{k-\Delta}$ is the delayed received coded sample
$d_{k-\Delta}$ is the hard decision at the output of the k–th iteration

Figure 3.9 – Iterative decoder
3.6 Interleaver design

The performance of turbo codes using iterative decoding algorithms depends on the type and depth of the interleaver used. This is because the interleaver structure affects the distance property of the resulting turbo code. Block interleavers of size 400 and 1024 were used in [25], and 256 x 256 pseudo–random interleavers in [4]. A study of more than 800 pseudo–random interleavers for short frame transmission systems (192 bits) was presented in [37]. For a particular encoder, the best pseudo–random interleaver is that which leads to the fewest output sequences with low weights [37].

In this section we present two criteria for designing interleavers of small sizes (less than 1000 bits) which is the case for applications such as mobile communications. In this case the interleaver size is limited by the maximum acceptable speech delay. Pseudo–random interleavers were found to give better results only for higher interleaver sizes (more than 1000 bits). These larger interleavers can be used in deep–space communications for which the decoding delay is not so important.

3.6.1 Proposed “odd–even” interleaver

For a rate half encoder, a particular type of interleaver which we call an “odd–even” interleaver, was found to give significant improvements when used in a turbo encoder design [39]. Let us assume that we have a random sequence of binary data input to a rate one half systematic encoder and we store only the odd coded bits, as shown in Table 3.1.

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
<th>x₅</th>
<th>x₆</th>
<th>x₇</th>
<th>x₈</th>
<th>x₉</th>
<th>x₁₀</th>
<th>x₁¹</th>
<th>x₁²</th>
<th>x₁³</th>
<th>x₁⁴</th>
<th>x₁⁵</th>
</tr>
</thead>
<tbody>
<tr>
<td>y₁</td>
<td>_</td>
<td>y₃</td>
<td>_</td>
<td>y₅</td>
<td>_</td>
<td>y₇</td>
<td>_</td>
<td>y₉</td>
<td>_</td>
<td>y₁¹</td>
<td>_</td>
<td>y₁³</td>
<td>_</td>
<td>y₁⁵</td>
</tr>
</tbody>
</table>

If we were now to interleave the same sequence of binary data in a pseudo–random order, encode it and store the even positioned coded bits, the result would be as in Table 3.2.

<table>
<thead>
<tr>
<th>xₐ</th>
<th>x₉</th>
<th>x₃</th>
<th>x₉</th>
<th>x₅</th>
<th>x₇</th>
<th>x₉</th>
<th>x₃</th>
<th>x₉</th>
<th>x₃</th>
<th>x₉</th>
<th>x₃</th>
<th>x₉</th>
<th>x₉</th>
<th>x₉</th>
<th>x₉</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>z₉</td>
<td>_</td>
<td>z₉</td>
<td>_</td>
<td>z₉</td>
<td>_</td>
<td>z₉</td>
<td>_</td>
<td>z₉</td>
<td>_</td>
<td>z₉</td>
<td>_</td>
<td>z₉</td>
<td>_</td>
<td>z₉</td>
</tr>
</tbody>
</table>

The data which is actually sent through the channel is as shown in Table 3.3; the original sequence of information bits xi, i = 1,..., 15 as in Table 3.1 and a multiplexed sequence of the odd and even positioned coded bits from both Table 3.1 and Table 3.2.

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
<th>x₅</th>
<th>x₆</th>
<th>x₇</th>
<th>x₈</th>
<th>x₉</th>
<th>x₁₀</th>
<th>x₁¹</th>
<th>x₁²</th>
<th>x₁³</th>
<th>x₁⁴</th>
<th>x₁⁵</th>
</tr>
</thead>
<tbody>
<tr>
<td>y₁</td>
<td>z₉</td>
<td>y₃</td>
<td>z₉</td>
<td>y₅</td>
<td>z₉</td>
<td>y₇</td>
<td>z₉</td>
<td>y₉</td>
<td>z₉</td>
<td>y₁¹</td>
<td>z₉</td>
<td>y₁³</td>
<td>z₉</td>
<td>y₁⁵</td>
</tr>
</tbody>
</table>
In Table 3.1 all the odd information bits have their own coded bit present. Due to the pseudo–random way of interleaving, some of the coded bits stored in Table 3.2 can be for even information bits and some for odd information bits. This means that some of the information bits will have two coded bits associated with them and others will have no coded bit associated with them, so the coding power is not uniformly distributed to all the bits. So for errors which affect information bits which do not have any coded bit associated with them the decoder will perform worse in both dimensions.

An example of an “odd–even” type of interleaver is a block interleaver with an odd number of rows and an odd number of columns as in Table 3.4, in which we store row–wise the sequence of random data.

Table 3.4 – 3x5 block interleaver

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
</tr>
</thead>
<tbody>
<tr>
<td>X6</td>
<td>X7</td>
<td>X8</td>
<td>X9</td>
<td>X10</td>
</tr>
<tr>
<td>X11</td>
<td>X12</td>
<td>X13</td>
<td>X14</td>
<td>X15</td>
</tr>
</tbody>
</table>

We produce the coded bits and store only the odd coded bits as in Table 3.1. Now we read column–wise, encode and store the even positions of the coded bits as in Table 3.5.

Table 3.5 – Even positioned coded bits

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
<th>x10</th>
<th>x11</th>
<th>x12</th>
<th>x13</th>
<th>x14</th>
<th>x15</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>z6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>z2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>z12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>z8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>z4</td>
<td></td>
<td></td>
<td>z4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>z14</td>
<td></td>
<td></td>
<td>z14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>z10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Table 3.5 all the even information bits have their own coded bit present and in Table 3.1 all the odd information bits have their own coded bit present as well. When we multiplex the coded bits from both Table 3.1 and Table 3.5 we produce the coded sequence as in Table 3.6. This means that each of the information bits will have its own coded bit associated with it, so the coding power is uniformly distributed.

Table 3.6 – Information bits and multiplexed coded bits for an “odd–even” interleaver

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
<th>x10</th>
<th>x11</th>
<th>x12</th>
<th>x13</th>
<th>x14</th>
<th>x15</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>z6</td>
<td>y3</td>
<td>z2</td>
<td>y5</td>
<td>z12</td>
<td>y7</td>
<td>z8</td>
<td>y9</td>
<td>z4</td>
<td>y11</td>
<td>z14</td>
<td>y13</td>
<td>z10</td>
<td>y15</td>
</tr>
</tbody>
</table>

From simulations performed for the sixteen state rate half systematic convolutional code given in [4] at an $E_b/N_0$ of 2.0 dB, after eight iterations and for a block interleaver of $20 \times 20$, a BER of $1.55 \times 10^{-4}$ was achieved. Using an “odd–even” interleaver of $21 \times 19$, a BER of $7.5 \times 10^{-5}$ was achieved. A second simulation was performed under the same conditions for a similar code but with memory size two. For a block interleaver of $20 \times 20$ the BER was $7.91 \times 10^{-4}$, and for an “odd–even” interleaver of $21 \times 19$, the BER was $3.97 \times 10^{-4}$. These results show the improved performance of “odd–even” interleaver design.
3.6.2 Proposed “simile” interleaver

In [45, 46] it was stated that there is no simple solution to terminate both encoders at the same time in the same state using only \( v \) bits where \( v \) is the encoder memory size of the convolutional code. To achieve this, a complicated method based on inserting certain bits in pre-determined positions was presented in [47]. We explain in this section a simple interleaver design for rate half turbo codes which allows both encoders of a turbo encoder to end in the same state [48]. This method is more bandwidth efficient and will improve the performance of turbo codes when the MAP decoding algorithm is used.

Until now, the main purpose of the interleaver was to increase the minimum distance of the turbo code such that after the correction in one dimension the remaining errors should become correctable error patterns in the second dimension. In doing so, the error correcting capability of the turbo code will approach that of the product code. In the previous section we introduced an “odd–even” type of interleaver where each information bit has associated with it one and only one coded bit. In this way the correction capability of the code is uniformly distributed over all information bits. We now impose another restriction on the interleaver design: after encoding both sequences of information bits, (the straight and the interleaved one), the state of both encoders of the turbo code are to be the same. This will allow only one tail to be appended to the information bits, which will drive both encoders to the zero state. We call this a “simile” type of interleaver.

In order to keep things simple we consider a four state turbo code which has each elementary rate half systematic encoder as in Figure 3.10.

![Figure 3.10 – A four state systematic convolutional encoder](image)

Assume the initial state of the encoder to be zero. We start to encode the sequence of information bits from time \( k = 1 \) to \( k = N \). The state bits, \( S^0 \) and \( S^1 \), for the first 9 information bits will change as shown in Table 3.7.

The results derived for the particular case of \( v = 2 \) can be extrapolated to any encoder memory size as long as the feedback polynomial is similar, i.e., all the delayed bits are added modulo two at the encoder input.
Table 3.7 – State sequence for a four state turbo encoder

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>$S^0_{1k}$</th>
<th>$S^1_{1k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$d_2$</td>
<td>$d_1$</td>
<td>0</td>
</tr>
<tr>
<td>$d_3$</td>
<td>$d_1 + d_2$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>$d_4$</td>
<td>$d_2 + d_3$</td>
<td>$d_1 + d_2$</td>
</tr>
<tr>
<td>$d_5$</td>
<td>$d_1 + d_3 + d_4$</td>
<td>$d_2 + d_3$</td>
</tr>
<tr>
<td>$d_6$</td>
<td>$d_1 + d_2 + d_4 + d_5$</td>
<td>$d_1 + d_3 + d_4$</td>
</tr>
<tr>
<td>$d_7$</td>
<td>$d_2 + d_3 + d_5 + d_6$</td>
<td>$d_1 + d_2 + d_4 + d_5$</td>
</tr>
<tr>
<td>$d_8$</td>
<td>$d_1 + d_3 + d_4 + d_6 + d_7$</td>
<td>$d_2 + d_3 + d_5 + d_6$</td>
</tr>
<tr>
<td>$d_9$</td>
<td>$d_1 + d_2 + d_4 + d_5 + d_7 + d_8$</td>
<td>$d_1 + d_3 + d_4 + d_6 + d_7$</td>
</tr>
</tbody>
</table>

We can reorganize the whole block of $N$ information bits in the following $v + 1$ sequences:

- Sequence 0 = \{ $d_k$ | $k \text{ mod } (v + 1) = 0$ \}
- Sequence 1 = \{ $d_k$ | $k \text{ mod } (v + 1) = 1$ \}
- Sequence 2 = \{ $d_k$ | $k \text{ mod } (v + 1) = 2$ \}.

It is easy to observe that for a given $N$ the final state of the encoder will be as in Table 3.8.

Table 3.8 – Final encoder state for $v = 2$ for $N$ information bits

<table>
<thead>
<tr>
<th>$N \text{ mod } (v + 1)$</th>
<th>$S^0_{1N}$</th>
<th>$S^1_{1N}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sequence1 + Sequence2</td>
<td>Sequence0 + Sequence1</td>
</tr>
<tr>
<td>1</td>
<td>Sequence2 + Sequence0</td>
<td>Sequence1 + Sequence2</td>
</tr>
<tr>
<td>2</td>
<td>Sequence0 + Sequence1</td>
<td>Sequence2 + Sequence0</td>
</tr>
</tbody>
</table>

The important conclusion is that from the point of view of the final encoder state, the order of the individual bits in each sequence does not matter, as long as they belong to the same sequence. The “simile” interleaver has to perform the interleaving of the bits within each particular sequence in order to drive the encoder to the same state as that which occurs without interleaving. Since both encoders end in the same state, we need only one tail to drive both encoders to state zero at the same time.

One particular implementation is a $(7, 3)$ block helical interleaver of Type 1 [59], where the depth is chosen to be $v + 1 = 3$. The information bits are stored row–wise as shown in Table 3.9. For this particular case, the three sequences are:

- Sequence 0 = \{ $d_3, d_6, d_9, d_{12}, d_{15}, d_{18}, d_{21}$ \}
- Sequence 1 = \{ $d_1, d_4, d_7, d_{10}, d_{13}, d_{16}, d_{19}$ \}
- Sequence 2 = \{ $d_2, d_5, d_8, d_{11}, d_{14}, d_{17}, d_{20}$ \}.  

39
Table 3.9 – An example of a “simile” interleaver with depth $v + 1$ for $v = 2$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_4$</td>
<td>$x_5$</td>
<td>$x_6$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
<td>$x_9$</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>$x_{14}$</td>
<td>$x_{15}$</td>
</tr>
<tr>
<td>$x_{16}$</td>
<td>$x_{17}$</td>
<td>$x_{18}$</td>
</tr>
<tr>
<td>$x_{19}$</td>
<td>$x_{20}$</td>
<td>$x_{21}$</td>
</tr>
</tbody>
</table>

The interleaved sequence produced is shown in Table 3.10.

Table 3.10 – The output of a “simile” interleaver

<table>
<thead>
<tr>
<th>$x_{19}$</th>
<th>$x_{17}$</th>
<th>$x_{15}$</th>
<th>$x_{10}$</th>
<th>$x_{8}$</th>
<th>$x_{6}$</th>
<th>$x_{1}$</th>
<th>$x_{20}$</th>
<th>$x_{18}$</th>
<th>$x_{13}$</th>
<th>$x_{11}$</th>
<th>$x_{9}$</th>
<th>$x_{4}$</th>
<th>$x_{2}$</th>
<th>$x_{21}$</th>
<th>$x_{16}$</th>
<th>$x_{14}$</th>
<th>$x_{12}$</th>
<th>$x_{7}$</th>
<th>$x_{5}$</th>
<th>$x_{3}$</th>
</tr>
</thead>
</table>

It can be easily observed that the index for the interleaved output follows the same pattern as for the straight sequence. For example $x_{19}$ belongs to sequence 1, $x_{17}$ belongs to sequence 2, $x_{15}$ belongs to sequence 0, $x_{10}$ belongs to sequence 1, and so on. At this point we notice that this particular “simile” interleaver is not an “odd–even” interleaver.

A necessary condition to generate a “simile odd–even” block helical interleaver is to choose the interleaver depth as an even number which is a multiple of $(v + 1)$. After the column counter reaches the interleaver depth, it will be reset and the index of the new output bit is odd. This is why the previous output bit must have an even index, thus the restriction of the interleaver depth to be an even number. Also the two dimensions of the interleaver block have to be relatively prime [59]. An example of such an interleaver is given in Table 3.11.

Table 3.11 – “Simile odd–even” block helical interleaver

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
<td>$x_9$</td>
<td>$x_{10}$</td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>$x_{14}$</td>
<td>$x_{15}$</td>
<td>$x_{16}$</td>
<td>$x_{17}$</td>
<td>$x_{18}$</td>
</tr>
<tr>
<td>$x_{19}$</td>
<td>$x_{20}$</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
<td>$x_{24}$</td>
</tr>
<tr>
<td>$x_{25}$</td>
<td>$x_{26}$</td>
<td>$x_{27}$</td>
<td>$x_{28}$</td>
<td>$x_{29}$</td>
<td>$x_{30}$</td>
</tr>
</tbody>
</table>

Part of the interleaved sequence is shown in Table 3.12.

Table 3.12 – The output of a “simile odd–even” block helical interleaver

| $x_{25}$ | $x_{20}$ | $x_{15}$ | $x_{10}$ | $x_{5}$ | $x_{30}$ | $x_{19}$ | $x_{14}$ | $x_{9}$ | $x_{4}$ | $x_{29}$ | $x_{24}$ | $x_{13}$ | $x_{8}$ | $x_{3}$ | $x_{28}$ | $x_{23}$ | $x_{18}$ | $x_{7}$ | $x_{2}$ | ...
|--------|--------|--------|--------|-------|--------|--------|--------|-------|-------|--------|--------|--------|-------|-------|--------|--------|--------|-------|-------|--------|

40
Simulation results are given in Section 3.7 for AWGN channels and in Section 3.8 for fading channels.

The use of a “simile odd–even” interleaver gives very good performance compared to other interleaver types. We have shown that both encoders of a turbo encoder can be terminated in the same state using this new type of interleaver. This means that only one “tail” appended after the information bits will be sufficient to drive both encoders to state zero. This will allow the maximum performance of a MAP decoder to be obtained.

3.6.3 Interleaving the tail or not?

After encoding the straight sequence by ENC, and the interleaved sequence by ENC, both encoders end in the same state. There are two options for handling the tail bits:

- add the tail bits and encode them only with ENC. All resulting coded bits are sent through the channel (no puncturing). Both decoders can start the backward recursion using the same tail bits followed by the straight/interleaved punctured sequence. We call this encoder the “tail not interleaved encoder” (TNIE).
- add the tail bits to the information bits and interleave them altogether. The end state for both encoders will be also zero. We call this encoder “tail interleaved encoder” (TIE).

The performance of both encoder types can be seen in Figure 3.11. We simulated the

![Figure 3.11 – Difference between TIE and TNIE encoders](image)
TIE for a block of data with 102 information bits plus two “tail” bits for a four state turbo code. The interleaver size is 104 bits which can be implemented with a simile odd–even interleaver with 17 rows and 6 columns.

For the TNIE we used a block of data with 104 information bits plus the two “tail” bits. The same simile odd–even interleaver of 104 bits with 17 rows and 6 columns is used to only interleave the data.

Since the performance of the TNIE is better than that of TIE, we will only consider the TNIE option in describing the performance of turbo codes.

### 3.6.4 Space–time distribution for helical interleavers

It is interesting to notice the difference in the distributions of output bits for a block interleaver and a helical interleaver. We consider a small interleaver size of 30 bits organised in 5 rows and 6 columns. Data is stored row–wise and is indexed by time as in Table 3.13.

<table>
<thead>
<tr>
<th>d₁</th>
<th>d₂</th>
<th>d₃</th>
<th>d₄</th>
<th>d₅</th>
<th>d₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₇</td>
<td>d₈</td>
<td>d₉</td>
<td>d₁₀</td>
<td>d₁₁</td>
<td>d₁₂</td>
</tr>
<tr>
<td>d₁₃</td>
<td>d₁₄</td>
<td>d₁₅</td>
<td>d₁₆</td>
<td>d₁₇</td>
<td>d₁₈</td>
</tr>
<tr>
<td>d₁₉</td>
<td>d₂₀</td>
<td>d₂₁</td>
<td>d₂₂</td>
<td>d₂₃</td>
<td>d₂₄</td>
</tr>
<tr>
<td>d₂₅</td>
<td>d₂₆</td>
<td>d₂₇</td>
<td>d₂₈</td>
<td>d₂₉</td>
<td>d₃₀</td>
</tr>
</tbody>
</table>

For a block interleaver we obtain the output shown in Figure 3.12. As expected, a uniform distribution is achieved and each bit is at the same distance from all the other bits in the sequence.

For a helical interleaver we obtain the output shown in Figure 3.13. It can be noticed that the bits which are positioned at multiples of the number of columns of the interleaver (dark squares) are preceded and followed by bits which are further away than the interleaving depth of 5 (which is the number of rows). This means that for these bits there is a higher degree of “uncorrelation” and they will have a lower probability of error compared to the rest of the bits in the same block. This will be obvious in Chapter 5 where the bit error ratio is plotted as a function of the bit position in the block.

One could use this property to obtain better error protection for those bits placed at multiples of the number of columns of the interleaver. This unequal error protection property could be exploited in mobile satellite communications where some bits in the speech frame are more important than others.

In the following sections we use this simile odd–even block helical interleaver to study the performance of turbo codes for AWGN and fading channels. Most of the computer simulations are performed for small interleaver sizes, less than 1000 bits.
Figure 3.12 – Space–time distribution for a block interleaver

Figure 3.13 – Space–time distribution for a helical interleaver
3.7 Performance of turbo codes on an AWGN channel

3.7.1 Four state rate half turbo code

The bit error ratios for a four state rate half turbo code as shown in Figure A.1, using block helical interleavers of 102, 204, 306, 408, 930 and 3660 bits, are presented in Figures 3.14 to 3.19.

Figure 3.14 – BER for a block size $B = 102$ bits (4 states, rate half, AWGN)

Figure 3.15 – BER for a block size $B = 204$ bits (4 states, rate half, AWGN)
Figure 3.16 – BER for a block size $B = 306$ bits (4 states, rate half, AWGN)

Figure 3.17 – BER for a block size $B = 408$ bits (4 states, rate half, AWGN)
Figure 3.18 – BER for a block size $B = 930$ bits (4 states, rate half, AWGN)

Figure 3.19 – BER for a block size $B = 3660$ bits (4 states rate half, AWGN)
From the computer simulations we notice that increasing the interleaver size to more than 930 bits, does not improve the performance of the four state rate half turbo. An interleaver depth of ten times the constraint length of the code (30 for the four state turbo code) seems to be the optimum. The main limiting factor was the correlation of the interleaved information bits at the input of the vertical encoder. To overcome this problem, each information bit in the interleaved sequence should be preceded and followed by at least ten times the constraint length of the code with different bits as compared with the straight sequence. Once this is achieved a second limiting factor becomes important: the interleaved/deinterleaved bursts of errors at the decoder output should be “randomised” as much as possible. The possible error bursts produced by one decoder are not “randomised enough” by the block helical interleaver. To obtain a better performance we will use a better interleaver which still has the “simile” property.

For the four state turbo code we can reorganize the whole block of N information bits to the following three sequences

Sequence 0 = \{d_k \mid k \mod 3 = 0\}
Sequence 1 = \{d_k \mid k \mod 3 = 1\}
Sequence 2 = \{d_k \mid k \mod 3 = 2\} where k = 1, 2, ..., N.

Each of these sequences is interleaved independently, using a pseudo–random interleaver. We simulated three interleaver sizes: 256, 1024 and 16384 bits. This means that blocks of 768, 3072 and 49152 information bits were used.

For the block size of 768 bits, the first limiting criteria still applies even if we use a pseudo–random interleaver. The results are slightly worse than those for the block helical interleaver of similar size and are plotted in Figure 3.20.
The results presented in Figures 3.21 and 3.22 show an improvement in the bit error rate for interleaver sizes greater than one thousand bits using pseudo-random interleavers.

Figure 3.21 – BER for a block size $B = 3072$ bits (4 states, rate half, AWGN) using a pseudo-random interleaver of 1024 bits

Figure 3.22 – BER for a block size $B = 49152$ bits (4 states, rate half, AWGN) using a pseudo-random interleaver of 16384 bits
3.7.2 Four state rate third turbo code

We simulate for the same interleaver sizes, the rate third turbo code as shown in Figure A.2 (the same code as the rate half, but without puncturing the parity bits). The results are presented in Figures 3.23 to 3.27 for block helical odd even simile interleavers.

Figure 3.23 – BER for a block size $B = 102$ bits (4 states, rate third, AWGN)

Figure 3.24 – BER for a block size $B = 204$ bits (4 states, rate third, AWGN)
Figure 3.25 – BER for a block size $B = 306$ bits (4 states, rate third, AWGN)

Figure 3.26 – BER for a block size $B = 408$ bits (4 states, rate third, AWGN)
From the simulation results, a few conclusions may be drawn. Firstly there is an improvement in the BER at low $E_b/N_0$ values when we increase the interleaver size. This is valid for both punctured (rate half) and not punctured (rate third) codes. However it is interesting to observe that for both codes at $E_b/N_0 = 3$ dB we get similar results for interleavers with interleaver depth ten times the constraint length of the code and higher. At this $E_b/N_0$ there is almost no difference between the performance of punctured and non punctured codes.

Secondly we note a flattening of the BER curve when increasing the $E_b/N_0$. This was also noticed in [51] when the curve was plotted for $E_b/N_0$ from 0 dB to 10 dB. After 2 dB the decrease of the BER – plotted in a logarithmic scale – is linear as we increase $E_b/N_0$. This is because the turbo code performance approaches that of the individual code. This seems to be a characteristic of turbo codes also noticed in [51].

Thirdly, at low values of $E_b/N_0$ we need more iterations than at higher values of $E_b/N_0$. At 3 dB we achieve most of the gain in only three iterations compared to eight or more at low values of $E_b/N_0$. Later on we shall see that for three dimensional turbo codes we need at least twenty iterations to reach a floor at low $E_b/N_0$.

Also, in the selection of the pseudo–random interleavers there are large differences in the BER performance as a function of the generator polynomial used. As in [51] there are "bad" and "good" interleavers which must be selected by computer simulations. This can
be explained by the fact that there are some error patterns which cannot be spread out by the “bad” interleavers. In this case neither of the decoders can correct the error pattern no matter how many iterations are performed.

3.7.3 Frame error ratio for the four state turbo code

The frame error ratio is another interesting parameter which is plotted in Figure 3.28 for a rate half turbo code and in Figure 3.29 for a rate third turbo code. The results are given after eight iterations.

For rate half turbo codes and QPSK modulation the number of symbols transmitted in a frame is equal to the interleaver size plus the encoder memory size. For example, for an interleaver size of 102 information bits, 104 QPSK symbols are transmitted per frame. In Figure 3.28 the curves are named after the corresponding interleaver size. We consider a frame to be in error if at least one information bit in the frame is in error.

It is interesting to note that, from the frame error rate point of view, the best results are for a frame size of 306 information bits. For smaller or larger frames the results are worse. This is useful to know when an automatic repeat request scheme is available. Some frames have residual errors which cannot be corrected by the MAP algorithm no matter how many iterations are used. Retransmitting those frames would improve the frame error ratio and also the bit error ratios for small interleaver sizes.

For the mobile satellite channel where speech is involved, a frame error rate of less than 0.04 is acceptable. From Figure 3.28 we could use frames as small as 204 information bits (206 QPSK sym/frame) at an Eb/N0 level of 1.8 dB or frames of 306 information bits (308 QPSK sym/frame) at an Eb/N0 level of 1.4 dB in order to achieve the above frame error ratio.

For rate third turbo codes and QPSK modulation the number of symbols transmitted in a frame is equal to 1.5 times the interleaver size plus the encoder memory size. For example, for an interleaver size of 102 information bits, 156 QPSK symbols are transmitted per frame. This could be reduced to 155 QPSK symbols because the tail is not interleaved and is encoded only once.

From Figure 3.29, very close results are obtained for frames corresponding to interleaver sizes of 204, 306 and 408 bits. For a smaller interleaver size (102 bits) or larger interleaver size (3660 bits) the frame error rate is worse.

For the rate 1/3 code, a frame error rate of 0.04 can be achieved for an interleaver size of 204 information bits (308 QPSK sym/frame) at an Eb/N0 of 1.25 dB. This result is very significant because low interleaver sizes will give small delays over the speech channel.
Figure 3.28 – Frame error ratio (4 states, rate half, AWGN)

Figure 3.29 – Frame error ratio (4 states, rate third, AWGN)
3.7.4 Sixteen state rate half turbo code

We now consider the sixteen state rate half turbo code as shown in Figure A.3 and simulate odd–even interleavers of sizes 110, 220, 420, 930 and 3660 bits. The bit error ratios for eight iterations and $E_b/N_0$ from 0 dB to 3 dB are presented in Figures 3.30 to 3.34.

Figure 3.30 – BER for a block size $B = 110$ bits (16 states, rate half, AWGN)

Figure 3.31 – BER for a block size $B = 220$ bits (16 states, rate half, AWGN)
Figure 3.32 – BER for a block size $B = 420$ bits (16 states, rate half, AWGN)

Figure 3.33 – BER for a block size $B = 930$ bits (16 states, rate half, AWGN)
Comparing the simulation results of the four state rate half turbo code with the sixteen state rate half turbo code we can say that the two turbo codes perform almost identically at low Eb/N0 levels (less than 1 dB) and for very small interleaver sizes (less than 400 bits). In this case the free distance of the code does not play an important role.

If one wants to obtain an even lower BER, there is no advantage in increasing the constraint length of the component convolutional code of the turbo code. As shown in Section 5.5, a multi-dimensional turbo code must be used instead. There it is explained that the probability of error for an M-dimensional turbo code is proportional to $N^{-(M-1)}$.

However, for an Eb/N0 of 2 dB and interleaver size of 420 bits, the sixteen state rate half turbo code gives a lower BER than the four state rate half turbo code. Increasing the interleaver size to 930, the improvement becomes significant at a lower Eb/N0 of 1 dB. The gain is even higher for an interleaver size of 3660.

Another difference is the increased number of iterations which is needed until the flooring effect appears for larger interleaver sizes. In Figure 3.34, at an Eb/N0 of 1 dB, 15 iterations were required to achieve the flooring effect.

The main conclusion is that the interleaver depth is the most important parameter to consider when deciding which component convolutional code to be used in a turbo code for a particular Eb/N0. Also the dimension of the turbo code is an important consideration when using larger constraint lengths codes.
3.7.5 Sixteen state rate third turbo code

We now consider the sixteen state rate third turbo code shown in Figure A.4 and simile odd–even interleavers of sizes 110, 220, 420, 930 and 3660 bits. The bit error ratios for eight iterations and $E_b/N_o$ from 0 dB to 3 dB are presented in Figures 3.35 to 3.39.

![Figure 3.35 – BER for a block size $B = 110$ bits (16 states, rate third, AWGN)](image1)

![Figure 3.36 – BER for a block size $B = 220$ bits (16 states, rate third, AWGN)](image2)
Figure 3.37 – BER for a block size $B = 420$ bits (16 states, rate third, AWGN)

Figure 3.38 – BER for a block size $B = 930$ bits (16 states, rate third, AWGN)
We can draw similar conclusions as for the four state turbo code. For small interleaver sizes (less than 220 bits) and low $E_b/N_0$ (less than 1dB) both codes, punctured or not, give very close results. For higher interleaver sizes (more than 420 bits) the sixteen state rate third code gives 0.7 dB improvement for a BER between $10^{-4}$ and $10^{-5}$ compared with the sixteen state rate half code.

### 3.7.6 Frame error ratio for the sixteen state turbo code

In Figure 3.40 we plot the frame error rate for the sixteen state rate half turbo code. For rate half turbo codes and QPSK modulation the number of symbols transmitted in a frame is equal to the interleaver size plus the encoder memory size. For example, for an interleaver size of 110 information bits, 114 QPSK symbols are transmitted per frame. In Figure 3.40 the curves are named after the corresponding interleaver size. We consider a frame to be in error if at least one information bit in the frame is in error.

The best results are obtained for the interleaver size of 420 information bits (424 QPSK sym/frame). The same frame error ratio of 0.04 can be achieved in this case at an $E_b/N_0$ of 1.6 dB. As we noticed for the four state turbo code the frame error ratio is worse for smaller or larger interleaver sizes.

Comparing these results with those for the four state rate half turbo code, we can say that an improvement of 0.2 dB can be achieved in the frame error rate.
In Figure 3.41 we have the frame error rate for the sixteen state rate third turbo code. The frame error ratio of 0.04 can be achieved at 0.75 dB $E_b/N_0$ for a 420 information bits interleaver size. As for the rate half code, for smaller or larger interleaver sizes, the frame error rate increases.

Figure 3.40 – Frame error ratio (16 states, rate half, AWGN)

Figure 3.41 – Frame error ratio (16 states, rate third, AWGN)
3.8 Performance of turbo codes on a Rayleigh channel

In this section we consider channels with randomly time–varying impulse responses. An example is the ionospheric/tropospheric radio channel with continuous changes of the physical characteristics of the media which results in multiple propagation paths. Each path is characterised by a propagation delay, $\tau_n(t)$, and an attenuation factor, $\alpha_n(t)$, both of them time–varying.

Usually, the attenuation factor does not change significantly. The propagation delay for different signal paths is the most important factor and it changes randomly. At the receiver end, the signal vectors from different paths can add with opposite phases, resulting in a very small signal or they can add constructively, increasing the signal power. For large numbers of paths we can apply the central limit theorem so that the received signal can be modelled as a complex–valued Gaussian random process [11].

If we consider a zero mean complex–valued Gaussian process, the envelope of the received signal at any instant in time is Rayleigh distributed and the channel is said to be a Rayleigh fading channel. If the mean is not zero due to some fixed signal reflectors, the channel is said to be a Ricean fading channel. Here we consider only the Rayleigh channel.

One important characteristic is the Doppler power spectrum of the channel [11]. This characterises the time variations of the channel or, in other words, the spectral broadening which can be observed in the transmission of a pure frequency tone. The range of frequencies over which this spectrum is nonzero is called the Doppler spread $B_d$ of the channel. This defines the coherence time:

$$(\Delta t)_c \approx 1/B_d.$$

A slowly changing channel has a large coherence time due to the small Doppler spread.

The second important parameter of a fading channel is its delay power spectrum. The duration in time for which the average power output of the channel is nonzero when a very narrow pulse was transmitted is called the multipath spread of the channel and is denoted by $T_m$. This is used to measure the coherence bandwidth of the channel:

$$(\Delta f)_c \approx 1/T_m.$$

The channel is said to be frequency selective if the coherence bandwidth is small compared to the signal bandwidth.

Our channel model used in computer simulations is a frequency–nonselective, slowly fading Rayleigh channel. This means that the signal bandwidth $W$ and the signaling interval $T$ satisfy the following

$$W \ll (\Delta f)_c$$

$$T \ll (\Delta t)_c.$$

The last condition translates to fixed channel attenuation and phase shift for at least one signaling interval. When $W \approx 1/T$ the frequency–nonselective, slowly fading Rayleigh channel must satisfy $T_m B_d < 1$ [11].
The channel model we use is described in Figure 3.42. $X_k$ and $Y_k$ are the outputs of a QPSK modulator as in Figure 3.5. The Rayleigh variable $a_k$ is generated as

$$a_k = \sqrt{b_k^2 + c_k^2},$$

where $b_k$ and $c_k$ are zero mean statistically independent Gaussian random variables each having a variance $\sigma^2$. We consider the power normalized to one as

$$E\{a_k^2\} = 2\sigma^2 = 1,$$

which gives a variance of 0.5 for the Gaussian variables.

\[ x_k = a_k X_k + p_k \]
\[ y_k = a_k Y_k + q_k \]

\[ p_k(0, \sigma_N^2) \]
\[ q_k(0, \sigma_N^2) \]

Figure 3.42 – Rayleigh channel

$p_k$ and $q_k$ are two uncorrelated Gaussian variables, with zero mean, variance $\sigma_N^2$, and independent of the variable $a_k$. In this model we use independent Rayleigh distribution. The results will be plotted as a function of $E_b/N_0$ where $E_b = 2\sigma^2E_b = E_b$ (3.20).

The MAP decoding algorithm has to be modified slightly by changing equation (3.8) (which defines the reliability value of the channel) to

$$L_c = 4 \frac{E_s}{N_0} a_k$$

(3.21)

With this modification we can use the same decoder structure which was described in Figure 3.9. All the simulation results in this section are compared with an uncoded transmission through the same Rayleigh channel.

In all our simulations we assume an accurate fade estimate at the receiving end and an independent Rayleigh distribution of the fades in order to compare our results with previous work in this area. For a real fading channel, an exact fade detector is difficult to implement and the fades are more or less correlated. This will decrease the performance when compared with the BER results for the fading channel model used here.

The performance of four state and sixteen state turbo codes using similar interleaver sizes as for the AWGN channel are plotted for values of $E_b/N_0$ in the range of 2 dB to 5 dB and for eight iterations.
3.8.1 Four state rate half turbo code

We now consider an independent Rayleigh channel and interleaver sizes of 102, 204, 306, 408 and 930 bits. The four state rate half turbo code is shown in Figure A.1. The bit error ratios after eight iterations are presented in Figures 3.43 to 3.47.

Figure 3.43 – BER for a block size $B = 102$ bits (4 states, rate half, fading)

Figure 3.44 – BER for a block size $B = 204$ bits (4 states, rate half, fading)
Figure 3.45 – BER for a block size $B = 306$ bits (4 states, rate half, fading)

Figure 3.46 – BER for a block size $B = 408$ bits (4 states, rate half, fading)
Clearly, the performance of turbo codes in an independent Rayleigh channel is worse than that for the AWGN channel by approximately 2.5 dB in coding gain. Increasing the interleaver size increases the code’s performance. The number of iterations needed is about the same as for the AWGN channel.

Comparing our results with those in [25] for the four state turbo code and interleaver size of 408 bits we obtain almost 0.3 dB improvement at BER of $10^{-2}$, $10^{-3}$ and $10^{-4}$. For a higher interleaver size of 930 bits we still get 0.2 dB improvement at a BER of $10^{-2}$ and $10^{-3}$. For BER less than $10^{-4}$ we notice a flattening of our curve, the results being slightly worse than those in [25]. The results in [25] were given for a similar independent Rayleigh channel but using a soft output Viterbi decoding algorithm.

It is worth mentioning that the interleaver embedded in turbo codes helps for short fades. For fades longer than the interleaver size turbo codes fail as any other coding scheme.

A more significant improvement will be seen in the next section where we do not puncture the coded bits, so that we have rate third turbo codes. The gain obtained for fading channels is larger than that for AWGN channels when we compare the rate half turbo codes with the rate third turbo codes.
3.8.2 Four state rate third turbo code

We now consider the four state rate third turbo code as shown in Figure A.2 for similar interleaver sizes. The BER for an independent Rayleigh channel are plotted in Figures 3.48 to 3.52.

Figure 3.48 – BER for a block size $B = 102$ bits (4 states, rate third, fading)

Figure 3.49 – BER for a block size $B = 204$ bits (4 states, rate third, fading)
Figure 3.50 – BER for a block size $B = 306$ bits (4 states, rate third, fading)

Figure 3.51 – BER for a block size $B = 408$ bits (4 states, rate third, fading)
As expected, the four state rate third turbo code performs better than the four state rate half turbo code. Comparing the coding gain for AWGN and fading channels, we notice that the rate third code has a greater gain.

For an AWGN channel, the gain from rate half to rate third at a BER $= 10^{-3}$ and interleaver size of 102 bits is 0.4 dB; for a BER $= 10^{-4}$ and interleaver size of 204 bits, the coding gain is 0.6 dB; for a BER $= 10^{-5}$ and interleaver size of 306 bits, the coding gain is 0.3 dB. For the independent Rayleigh fading channel, the gain from rate half to rate third at a BER $= 2\times 10^{-3}$ and interleaver size of 102 bits is 1.4 dB; for a BER $= 10^{-4}$ and interleaver size of 306 bits, the coding gain is 1.1 dB.

Comparing different interleaver sizes, we notice that the 306 bits interleaver gives a BER as good as the 930 bits interleaver size except for very low $E_b/N_0$. At $E_b/N_0 = 2$ dB, the 930 bits interleaver size gives a better BER if we increase the number of iterations to thirty.

### 3.8.3 Frame error ratio for the four state turbo code

The frame error ratio for a Rayleigh channel is plotted in Figure 3.53 for the rate half turbo code and in Figure 3.54 for the rate third turbo code with eight iterations. As for the AWGN channel the best FER is achieved for an interleaver size of 306 bits for both the rate half and rate third four state turbo codes.
Figure 3.53 – Frame error ratio (4 states, rate half, fading)

Figure 3.54 – Frame error ratio (4 states, rate third, fading)
3.8.4  Sixteen state rate half turbo code

The performance of the sixteen state rate half turbo code shown in Figure A.3 is plotted in Figures 3.55 to 3.58 for an independent Rayleigh channel.

Figure 3.55 – BER for a block size $B = 110$ bits (16 states, rate half, fading)

Figure 3.56 – BER for a block size $B = 220$ bits (16 states, rate half, fading)
Figure 3.57 – BER for a block size $B = 420$ bits (16 states, rate half, fading)

Figure 3.58 – BER for a block size $B = 930$ bits (16 states, rate half, fading)
The first observation is that the sixteen state rate half turbo code performs better than the four state turbo code using the block helical odd even simile interleaver. This contradicts the results given in [25] where the four state turbo code produced a better performance. The only differences are the block interleaver and the soft output Viterbi decoding algorithm used in [25].

Comparing the results given in [25], for a 408 bits interleaver size we obtain 1 dB improvement for BERs from $10^{-2}$ to $10^{-5}$. For 930 bits interleaver size we have 1 dB improvement for BERs from $10^{-1}$ to $10^{-3}$ and 0.6 dB improvement for BERs from $10^{-4}$ to $10^{-5}$. Again there is a flattening of our curve and the performance at an Eb/N0 of 5 dB is slightly worse than in [25]. The results in [25] were given for a similar independent Rayleigh channel but using a soft output Viterbi decoding algorithm.

Another observation is that sometimes we need more than eight iterations in order to achieve the best BER. For example in Figure 3.58 at Eb/N0 = 3 dB, after eight iterations the BER is $6.385 \times 10^{-3}$ and after thirty iterations the BER is $4.386 \times 10^{-3}$.

### 3.8.5 Sixteen state rate third turbo code

We now consider the sixteen state rate third turbo code shown in Figure A.4. The BER for an independent Rayleigh channel is plotted in Figures 3.59 to 3.62.

![Figure 3.59 – BER for a block size B = 110 bits (16 states, rate third, fading)](image-url)
Figure 3.60 – BER for a block size $B = 220$ bits (16 states, rate third, fading)

Figure 3.61 – BER for a block size $B = 420$ bits (16 states, rate third, fading)
From the above figures, we can conclude that the sixteen state rate third turbo code performs much better than the four state rate third turbo code for an independent Rayleigh channel.

For low $E_b/N_0$ we notice that a larger number of iterations are needed to obtain the best possible BER. If at an $E_b/N_0 = 4$ dB we need only three iterations to obtain a BER close to the best one, at an $E_b/N_0 = 2$ dB we need ten or more iterations.

We need to remark at this point that the flattening effect noticed in the BER curves at higher $E_b/N_0$ is caused by the correlation between the interleaver output and the interleaver input. To avoid this, pseudo-random or random interleavers should be used for interleaver sizes greater than 400 bits.

### 3.8.6 Frame error ratio for the sixteen state turbo code

The frame error ratio for a Rayleigh channel is plotted in Figure 3.63 for the rate half turbo code and in Figure 3.64 for the rate third turbo code with eight iterations.

It is important to notice that the best frame error rate for the rate half code is achieved for an interleaver size of 930 bits up to an $E_b/N_0$ of 4.3 dB, while for the rate third code the best interleaver size is 420 bits up to an $E_b/N_0$ of 4.5 dB. Increasing further the value of $E_b/N_0$, the two interleaver sizes are swapped around to achieve the best frame error rate.
Figure 3.63 – Frame error ratio (16 states, rate half, fading)

Figure 3.64 – Frame error ratio (16 states, rate third, fading)
### 3.9 Guidelines for the interleaver design

We provide here an intuitive way of understanding why the performance of turbo codes depends so much on the interleaver design. Using computer simulations we came to the conclusion that there are three main limiting factors.

The first and the most important factor is the depth of the interleaver. This can be understood from Figure 3.65 where we describe how the soft outputs of the turbo codes are produced.

![Extrinsic information for turbo codes](image)

- **Transmitted sequence**
  - $T_i$
- **Received sequence**
  - $R_i$
- **Extrinsic information output**
  - $E_i$

**Figure 3.65 – Extrinsic information for turbo codes**

The extrinsic information $E_i$ associated with the information bit $d_i$ is produced by the MAP algorithm using a forward and backward recursion applied to the whole received sequence. From computer simulations we can say that 95% of the value of the extrinsic information $E_i$ is determined by the previous $N$ and the following $N$ received symbols, where $N$ is ten times the constraint length of the code. The larger the interleaver size is, the more “uncorrelated” the $2N$ window of the straight sequence of data is compared with the $2N$ window of the interleaved data. This gives two “independent” criteria to estimate the soft value of the same bit and can be passed as extra information from one decoder to another.

Once the first condition is fulfilled, to obtain a further improvement in the performance of turbo codes, we have to consider the second limiting factor: with what degree of “randomness” can the interleaver/deinterleaver spread the bursts of errors from one decoder output to the next decoder input. From this point of view the ideal interleaver is a random interleaver. This is the best choice only when the first limiting factor is eliminated.

The third limiting factor is the free distance of the code. This can be seen at higher values of $E_b/N_0$ (3 dB) when if the first two conditions are fulfilled, the iterative decoding algorithm converges very quickly (usually within three iterations). Increasing the number of iterations does not give better performance.
3.10 Interleaver design for encryption purposes

It is well known that coding techniques which add reliability to the information sequence are incompatible with encryption techniques which try to remove as much correlation from the transmitted sequence as possible.

In this section we only want to highlight one possible application of turbo codes in the area of cryptography. The idea is similar to that of spread spectrum techniques, to bury the signal in noise and recover it at the receiver end. This is possible with turbo codes due to the low $E_b/N_0$ at which they operate. The assumption we make here is that we use a very low $E_b/N_0$ for which each individual sequence of information bits or coded bits is very hard to detect. Only with the high gain of the turbo decoder could one be able to successfully recover the transmitted data from noise.

3.10.1 Pseudo–random interleaver with noise

In Section 3.6.2 we introduced the idea of a “simile” interleaver for the purpose of terminating the trellis of the turbo code to the same state. We explained that the information sequence can be divided into $v + 1$ strings which can be interleaved independently, where $v$ is the number of delay elements of the encoder. In Figure 3.66 we describe a rate third sixteen state turbo code with each interleaver constructed from five pseudo–random interleavers. Each pseudo–random interleaver uses a different generator polynomial which can start from a sequence of states determined by a long pseudo–random generator. The uncoded information bits are interleaved as well. The outputs of the turbo encoder are buried in noise whose variance can be changed in each frame or even in each interleaved sequence. The long pseudo–random generator which generates the starting states of the encoder together with the variance of the noise are the keys to the proposed encryption system. We assume these keys to be secret and known at the receiver end. From the attacker’s point of view, assuming the coding scheme is known, it would be very difficult to find all the missing variables needed by a turbo decoder.

![Figure 3.66 – Three dimensional turbo code](image)
3.10.2 Pseudo–random interleaver for the encrypted data with noise

The previous scheme assumed that the encryption block was used before the turbo encoder. To make life more difficult we can imbed the encryption block in the turbo encoder as in Figure 3.67.

![Figure 3.67](image)

The turbo decoder can be easily modified to work with an imbedded decryption block. Modelling the encryption process as a modulo two summation between the information data and a pseudo–random sequence, the turbo decoder will pass the soft outputs inverted or not accordingly to the pseudo–random sequence.

Turbo codes can definitely be used with more success than the traditional coding schemes to increase the protection of data from the secrecy point of view. The analysis of this encryption technique is beyond the scope of this thesis. We just wanted to point out another useful application of turbo codes which could be investigated in the future.
3.11 Conclusions

This chapter described how to apply the MAP decoding algorithm to turbo codes for iterative decoding. The information transfer from one decoder to another is essential for improving the performance from one step to the next in the iterative decoding process.

New interleaver design criteria were presented in order to increase the performance of turbo codes and increase the bandwidth efficiency by appending only one tail to the information bits. This is significant for the small frame sizes used in mobile communications.

Guidelines to the limiting factors caused by the interleaver design were provided. The correlation at the input of the interleaver as well as the error randomising capability of the interleaver play different roles as a function of the interleaver size.

The choice of the interleaver depends on the allowable delay in the system; this gives us the maximum interleaver size/depth. If the interleaver depth is less than ten times the constraint length of the turbo code, a simple block helical interleaver is as good as any, plus it has the advantage of using only one tail to terminate both trellises to state zero. If the interleaver depth is more than ten times the constraint length of the turbo code, we can obtain better results with a pseudo–random or random interleaver. This is possible because long bursts of errors become uncorrelated using random interleavers.

Computer simulation results were presented for AWGN and fading channels, different interleaver sizes and number of iterations, and for the four and sixteen states turbo codes. Rate 1/2 and 1/3 turbo codes were also investigated.

One very important observation is that all the simulation results were based on an accurate estimate of the channel state. If the channel fade estimates are not accurate, further degradation in performance is expected.

Another factor to be considered is that real fading channels have a degree of correlation between the fades. We considered only the independent Rayleigh fading channel in order to compare our results with previous work done in this area.

There are also many directions left to be investigated. A few of them are:

- how closely does the iterative decoding technique approach the maximum likelihood decoding of the turbo code?
- how to choose the constituent codes for better performance?
- are there any other concatenation techniques more suitable for iterative decoding?

Turbo codes were not a sudden invention. As always the case, they rely on older concepts like soft output algorithms and iterative decoding of multi–dimensional codes. Which is the best combination of these ideas is an interesting area which remains to be investigated further.
4 A DIVERSITY SCHEME USING TURBO CODES

4.1 Introduction

Turbo codes have a built-in structure that is suitable for diversity techniques used to improve the quality of communications over a mobile satellite channel. The novelty of the proposed scheme is that the transmitted rate half turbo code is transformed at the receiver into a more powerful rate third turbo code. This concept came up from team discussions at the Institute for Telecommunications Research about diversity techniques for convolutional codes. We extended this idea to turbo codes and studied their performance for both AWGN and Rayleigh channels. We shall first present different diversity techniques in order to understand the advantage of using turbo codes over standard convolutional or block codes for fading channels.

4.2 Diversity combining methods

The basic idea of diversity techniques is that if the receiver can have available the same information sequence transmitted over independently fading channels, the probability that all the received symbols will be affected by big fades simultaneously is very small. This can be achieved by time diversity, frequency diversity or space diversity when multiple antennas are used.

Time diversity means transmitting the same information in different time slots, where the separation between the successive time slots is larger than the coherence time of the channel [11].

Frequency diversity is achieved when the same information sequence is sent on different carriers where the separation between successive carriers is greater than the coherence bandwidth of the channel [11].

Both time and frequency diversity techniques are applied in mobile communications.

Figure 4.1 – Diversity scheme using two satellites
From the receiver point of view, there are three combining methods to be considered [12]: maximum ratio combining (MRC), equal gain combining (EGC) and selection combining (SLC).

Let us assume that the same modulated symbol $x$ is sent through two satellite channels. The two received signals are

$$
y_1 = a_1 x + n_1$$

$$
y_2 = a_2 x + n_2$$

where $a_1$ and $a_2$ are two fading amplitudes and $n_1$ and $n_2$ are two independent AWGN variables. The MRC technique combines these two signals as

$$y_{\text{MRC}} = a_1 y_1 + a_2 y_2.$$  

The EGC method is simpler and can be expressed as

$$y_{\text{EGC}} = y_1 + y_2.$$  

The SLC is defined as

$$y_{\text{SLC}} = \begin{cases} y_1 ; & a_1 \geq a_2 \\ y_2 ; & a_1 < a_2 \end{cases}.$$  

The performance of MRC and SLC depends on the quality of the channel estimator. The better the estimator is the better the performance which can be achieved. The EGC is independent of the channel estimator and can be easily implemented in any receiver.

In [11] it is shown that both time and frequency diversity techniques are a simple form of repetition coding with block interleaving in order to obtain independent errors. A better performance can be obtained using different coding techniques. Convolutional codes, Reed–Solomon codes, Golay code and turbo codes have been investigated in [12]. Among these, turbo codes were found to achieve the best performance.

A new diversity combining scheme which we call turbo diversity scheme (TDS), using turbo codes and MRC, EGC, or SLC is described in the next section. For mobile communications, the information bits are organised in frames or packets in the range of 100 to 200 symbols/packet. Due to speech constraints, the allowable decoding delay is in the order of two or three packets, which translated in information bits is less than 500 bits for a standard QPSK modulation scheme with rate half coding. In the previous chapter we have shown that the performance of turbo codes improves by increasing the interleaver size. Using TDS, even for small interleaver sizes we can achieve very low BER. It is interesting to observe that the best interleavers found for the AWGN channel are also the best for the fading channel.

In the following sections we consider only the rate half turbo codes and QPSK modulation in which case $E_b/N_0$ is equal to $E_s/N_0$. We will plot the BER against the equivalent $E_b/N_0$. Since this scheme requires two transmitters, we send half the power through each satellite channel. This means that for an $E_b/N_0$ of 3 dB read from the plots, we transmit data on each satellite channel at an $E_s/N_0$ of 0 dB.
4.3 Turbo Diversity Scheme

We consider a systematic rate half turbo encoder, which produces two coded bits $y_i$ and $z_i$ for each information bit $x_i$ as in Figure 4.2.

Using QPSK modulation, we send the same information sequence via two satellites in the following way:

- for satellite $S_1$: I signal is modulated by: $x_1, x_2, x_3, x_4, x_5, x_6, ...$
  Q signal is modulated by: $y_1, y_2, y_3, y_4, y_5, y_6, ....$

- for satellite $S_2$: I signal is modulated by: $x_1, x_2, x_3, x_4, x_5, x_6, ...$
  Q signal is modulated by: $z_1, z_2, z_3, z_4, z_5, z_6, ....$

For a clear presentation of the multiplexing which is performed at the output of the encoder, we ignore the delays and the random order of the coded bits in the vertical dimension after interleaving. Each information bit has one coded bit, so this is a rate half code for each satellite channel. The decoder must follow the same demultiplexing pattern. The two dimensional turbo decoder is built from two uni–dimensional decoders. The “horizontal” decoder receives the information bits and the coded bits in the horizontal dimension. The “vertical” decoder receives the interleaved information bits and the coded bits in the vertical dimension. This is why we can symbolically represent the two dimensional turbo decoder as a black box with three inputs and one output.

The most significant difference from other conventional diversity techniques is that the TDS changes the rate half code into a more powerful rate third code as can be seen in the next section. The way the turbo codes are constructed fits naturally with this diversity scheme.

All the simulation results are given after eight iterations. By further increasing the number of iterations, a very small improvement can be achieved, approximately 0.1 dB after thirty iterations. Due to time limitations and the great number of simulations needed we traded off that extra improvement against time.

For the fading channel the TDS results are preceded by a similar combining scheme for the same turbo codes but without TDS in order to emphasise the BER improvement of this new technique.
4.4 AWGN channel

At the receiver end, if only one satellite channel is received, say S₁, the receiver will insert zeros in the place of the punctured bits as described in Figure 4.3,

\[
\begin{align*}
&x_1+p_1, \ x_2+p_2, \ x_3+p_3, \ x_4+p_4, \ x_5+p_5, \ldots \\
&y_1+q_1, \ 0, \ y_3+q_3, \ 0, \ y_5+q_5, \ldots \\
&0, \ z_2+q_2, \ 0, \ z_4+q_4, \ 0, \ldots
\end{align*}
\]

Figure 4.3 – One satellite channel received

where \( p_i \) is the AWGN for the I signal and \( q_i \) is the AWGN for the Q signal. This case was the rate half scheme already studied in the previous chapter. If both satellite channels are available, the input to the decoder will be a combination of the information bits and both coded bits associated with each information bit. This is described as in Figure 4.4 using equal gain combining for the information bits where \( p_i \) and \( q_i \) are AWGN for the I and Q signals for the second satellite. It is important to note that the turbo decoder is the same for both cases. No modification is needed when switching from one-to-two or two-to-one received channels.

\[
\begin{align*}
&x_1+(p_1+p_1)/2, \ x_2+(p_2+p_2)/2, \ x_3+(p_3+p_3)/2, \\
&y_1+q_1, \ y_2+q_2, \ y_3+q_3, \\
&z_1+q_1, \ z_2+q_2, \ z_3+q_3,
\end{align*}
\]

Figure 4.4 – Two satellite channels received

Using the four state rate half turbo code with block helical interleavers of 102, 204, 306 and 408 information bits we plotted the BER in Figure 4.5 and the FER in Figure 4.6 for \( E_b/N_0 \) from 1 dB to 3 dB. The \( E_b/N_0 \) used for each satellite channel equals \( E_b/N_0 \) less 3 dB.

The BER and FER for a sixteen state rate half turbo code with block helical interleavers of 110, 220 and 420 bits is plotted in Figure 4.7 and Figure 4.8.

From these simulations we can conclude that for the AWGN channel, the TDS has similar performance as in the case of a single satellite channel at the same \( E_b/N_0 \) and for similar interleaver sizes (see Sections 3.7.1 and 3.7.4).
Figure 4.5 – BER for diversity with a four state turbo code (AWGN channel)

Figure 4.6 – FER for diversity with a four state turbo code (AWGN channel)
Figure 4.7 – BER for diversity with a sixteen state turbo code (AWGN channel)

Figure 4.8 – FER for diversity with a sixteen state turbo code (AWGN channel)
4.5 Fading channel

4.5.1 Traditional maximum ratio combining

We consider first the classical MRC strategy. The Rayleigh fading channel used was described in Section 3.8. The same information and coded sequences are sent through both satellite channels. At the receiver, a classical maximum ratio combining strategy was used. The round brackets indicate the I and Q values for the received symbol. For the first received symbol the combining strategy is

\[ x_1 = (x_1 \times \text{fade}_1 + p_1) \times \text{fade}_1 + (x_1 \times \text{fade}_1 + p_1) \times \text{fade}_1 \]

\[ y_1 = (y_1 \times \text{fade}_1 + q_1) \times \text{fade}_1 + (y_1 \times \text{fade}_1 + q_1) \times \text{fade}_1 \]

For the second received symbol

\[ x_2 = (x_2 \times \text{fade}_2 + p_2) \times \text{fade}_2 + (x_2 \times \text{fade}_2 + p_2) \times \text{fade}_2 \]

\[ z_2 = (z_2 \times \text{fade}_2 + q_2) \times \text{fade}_2 + (z_2 \times \text{fade}_2 + q_2) \times \text{fade}_2 \]

and so on. The symbols \( \text{fade}_i \) and \( \text{fade}_i \) are the fade values at time \( i \) for the two channels. The same convention is used as for the AWGN channel. In this case we have a rate half turbo code at the receiver end, as at the transmitter end.

We consider only the four state rate half turbo code with a block helical interleaver of 204 information bits and the sixteen state rate half turbo code with a block helical interleaver of 220 information bits. The BER results are plotted in Figure 4.9. Compared to Figures 3.44 and 3.56, we can clearly see the advantage of using the diversity technique. The four and sixteen state codes gain 1.2 and 1.3 dB, respectively, over a single satellite scheme for a BER = \( 2 \times 10^{-3} \).
4.5.2 Maximum ratio combining with TDS

The MRC scheme can be slightly modified for TDS (MRC–TDS), as presented in this section. For the first transmitted information symbol through both channels, at the receiver end we input to the turbo decoder the following
\[
x_1 = (x_1 \cdot \text{fade}_1 + p_1) \cdot \text{fade}_1 + (x_1 \cdot \text{fade}_1 + p_1) \cdot \text{fade}_1
\]
\[
y_1 = (y_1 \cdot \text{fade}_1 + q_1) \cdot \text{fade}_1
\]
\[
z_1 = (z_1 \cdot \text{fade}_1 + q_1) \cdot \text{fade}_1
\]
For the second received information symbol, the three inputs are
\[
x_2 = (x_2 \cdot \text{fade}_2 + p_2) \cdot \text{fade}_2 + (x_2 \cdot \text{fade}_2 + p_2) \cdot \text{fade}_2
\]
\[
y_2 = (y_2 \cdot \text{fade}_2 + q_2) \cdot \text{fade}_2
\]
\[
z_2 = (z_2 \cdot \text{fade}_2 + q_2) \cdot \text{fade}_2
\]
and so on.

With this method, the transmitted rate half code is decoded by a rate third decoder at the receiver. We again consider only the four state rate half turbo code with a block helical interleaver of 204 information bits and the sixteen state rate half turbo code with a block helical interleaver of 220 information bits. The BER results are plotted in Figure 4.10.

![Figure 4.10 – MRC–TDS for a fading channel](image)

Comparing Figure 4.10 with Figure 4.9 we can conclude that the MRC–TDS gives an improvement of more than 0.5 dB over the classical MRC scheme. The improvement comes from the increased correction power of the rate third code formed at the receiver end.
4.5.3 Traditional equal gain combining

The classical EGC strategy is described by the following relations. We assume the same uncoded and coded bits are sent through both fading channels. For the first received symbol the combining strategy is

\[
\begin{align*}
x_1 &= (x_1 \times \text{fade}_1 + p_1) + (x_1 \times \text{fade}_1 + p_1) \\
y_1 &= (y_1 \times \text{fade}_1 + q_1) + (y_1 \times \text{fade}_1 + q_1)
\end{align*}
\]

For the second received symbol

\[
\begin{align*}
x_2 &= (x_2 \times \text{fade}_2 + p_2) + (x_2 \times \text{fade}_2 + p_2) \\
y_2 &= (z_2 \times \text{fade}_2 + q_2) + (z_2 \times \text{fade}_2 + q_2)
\end{align*}
\]

and so on. In this case we have a rate half turbo code at the receiver end, as at the transmitter end.

We consider only the four state rate half turbo code with a block helical interleaver of 204 information bits and the sixteen state rate half turbo code with a block helical interleaver of 220 information bits. The BER results are plotted in Figure 4.11.

![Figure 4.11 – EGC for a fading channel](image)

From Figure 4.11 and Figure 4.9 we can see that the performance of the EGC method is far worse than the MRC method. Also, the four state code and the sixteen code perform very closely.
4.5.4 Equal gain combining with TDS

The classical EGC can be modified for the TDS (EGC–TDS) as described by the following relations. For the first received information symbol the combining strategy is

\[ x_1 = x_1 \cdot \text{fade}_1 + p_1 + x_1 \cdot \text{fade}_1 + p_1 \]
\[ y_1 = y_1 \cdot \text{fade}_1 + q_1 \]
\[ z_1 = z_1 \cdot \text{fade}_1 + q_1 \]

For the second one, the combining strategy is

\[ x_2 = x_2 \cdot \text{fade}_2 + p_2 + x_2 \cdot \text{fade}_2 + p_2 \]
\[ y_2 = y_2 \cdot \text{fade}_2 + q_2 \]
\[ z_2 = z_2 \cdot \text{fade}_2 + q_2 \]

and so on. With this method, the transmitted rate half code becomes a rate third code at the receiver end.

We only consider the four state rate half turbo code with a block helical interleaver of 204 information bits and the sixteen state rate half turbo code with a block helical interleaver of 220 information bits. The BER results are plotted in Figure 4.11.

![Figure 4.12 – EGC–TDS for a fading channel](image)

As in the MRC case the EGC–TDS gives better results than EGC but worse results when compared with MRC–TDS.
4.5.5 Traditional selection combining

The SLC for turbo codes is defined by the rules below, where fade\textsubscript{i} and fade\textsubscript{i} are the fade values at time i for the two channels. For the first received symbol

If \( \text{fade}_1 > \text{fade}_1 \)
\[ x_1 = x_1 \cdot \text{fade}_1 + p_1 \]
\[ y_1 = y_1 \cdot \text{fade}_1 + q_1 \]
otherwise
\[ x_1 = x_1 \cdot \text{fade}_1 + p_1 \]
\[ y_1 = y_1 \cdot \text{fade}_1 + q_1 \]

For the second received symbol

If \( \text{fade}_2 > \text{fade}_2 \)
\[ x_2 = x_2 \cdot \text{fade}_2 + p_2 \]
\[ z_2 = z_2 \cdot \text{fade}_2 + q_2 \]
otherwise
\[ x_2 = x_2 \cdot \text{fade}_2 + p_2 \]
\[ z_2 = z_2 \cdot \text{fade}_2 + q_2 \]
and so on.

The BER results are plotted in Figure 4.13. We only consider the four state rate half turbo code with a block helical interleaver of 204 information bits and the sixteen state rate half turbo code with a block helical interleaver of 220 information bits.

![Figure 4.13 – SLC for a fading channel](image-url)
4.5.6 Selection combining with TDS

The SLC for TDS (SLC–TDS) is defined by the following combining rules. For the first received symbol the inputs to the turbo decoder are

If $\text{fade}_1 > \text{fade}_1$

\[
x_1 = x_1 \cdot \text{fade}_1 + p_1 \\
y_1 = y_1 \cdot \text{fade}_1 + q_1 \\
z_1 = z_1 \cdot \text{fade}_1 + q_1
\]

otherwise

\[
x_1 = x_1 \cdot \text{fade}_1 + p_1 \\
y_1 = y_1 \cdot \text{fade}_1 + q_1 \\
z_1 = z_1 \cdot \text{fade}_1 + q_1
\]

For the second received symbol

If $\text{fade}_2 > \text{fade}_2$

\[
x_2 = x_2 \cdot \text{fade}_2 + p_2 \\
y_2 = y_2 \cdot \text{fade}_2 + q_2 \\
z_2 = z_2 \cdot \text{fade}_2 + q_2
\]

otherwise

\[
x_2 = x_2 \cdot \text{fade}_2 + p_2 \\
y_2 = y_2 \cdot \text{fade}_2 + q_2 \\
z_2 = z_2 \cdot \text{fade}_2 + q_2
\]

and so on. We consider the same codes as before and the BER results are in Figure 4.14.

![Figure 4.14 – SLC–TDS for a fading channel](image-url)
4.6 Conclusions

In this chapter we presented a new diversity technique which can be used with turbo codes to give better results than the standard diversity combining methods. With this method the transmitted rate half code becomes a rate third code at the receiver end. For two AWGN channels the TDS performs as well as for one channel at the same $E_b/N_0$. For two fading channels the TDS gives more than 0.5 dB improvement over the same turbo codes using traditional diversity techniques.

Real data was also used in computer simulations and the TDS scheme proved to perform the best [12]. Among the different combining methods used with TDS, the MRC–TDS gave the lowest BER.

Computer simulated data was used in the BER plots presented in this chapter and a summary of the performance of different combining methods is plotted in Figure 4.15 for the sixteen state turbo code.

![Figure 4.15 – TDS for a fading channel for a sixteen state turbo code](image)

From the above figure we conclude that the best results are obtained for the MRC–TDS which has a 1 dB improvement over the EGC–TDS. This is true in the case of a perfect channel estimator as was considered in this chapter. The performance of the MAP algorithm decreases if there are errors in the channel estimates. This is expected to lead to a lower gain for a real receiver.
5 RATE–COMPATIBLE TURBO CODES

5.1 Introduction

There are some applications, such as speech or image compression, where some bits must have a higher level of protection than others. One solution for this would be to use different encoders/decoders for different groups of bits. This would increase the complexity of the communication system and, since some groups could be small, exclude the use of powerful long channel codes. The rate–compatible punctured convolutional codes (RCPC) introduced in [65] are a method to offer different levels of protection to different blocks of bits using the same encoder and decoder blocks.

Punctured convolutional codes [66] are convolutional codes which have some of the encoder outputs punctured, i.e., not transmitted over the channel. Puncturing is used to obtain convolutional codes with differing correction powers using the same Viterbi decoder. Punctured codes of rate $k/n$ can be decoded with trellises where two branches arrive at each node instead of $2^k$ branches, without any loss in performance.

Turbo codes can also offer this unequal error protection (UEP) in a more natural way than the RCPC. We have shown in the previous chapters that by increasing the interleaver size, a decrease of the BER is achieved. The BER dependence of the interleaver size was studied in Chapter 3. For the bits which need higher protection the interleaver size could be increased. This could be cumbersome from the point of view of hardware implementation. Another option is to vary the number of coded bits associated with each information bit.

This idea is investigated below for small blocks of data at low signal to noise ratios. We wish to find a scheme that could be advantageous to turbo codes in a TDMA communications system with mobile satellite channels. Due to speech constraints, the delay must be kept to a minimum. This delay affects the maximum interleaver size, which was found to be around 500 information bits.

A variable rate scheme which can offer different error correction levels using the same decoder is also investigated. This is useful when, due to channel variations, the coding rate has to be changed in order to keep the bit error rate constant.

The next section will describe two dimensional turbo codes which can vary their rate from one half to one third using the same encoder/decoder block.

This idea can be extended to multi–dimensional turbo codes where the rate can be varied from 1 to $1/n$, where $n$ is the dimension of the code. A three dimensional turbo code is presented in the following section using the same block helical simile odd–even interleaver introduced in Section 3.6.
5.2 Rate one half turbo codes

A two dimensional rate half turbo encoder is shown in Figure 5.1.

![Figure 5.1 – Rate one half turbo encoder](image)

For a rate half code, the inputs to the turbo decoder can be arranged as in Figure 5.2.

![Figure 5.2 – Rate one half turbo decoder](image)

5.3 Rate one third turbo codes

If a higher level of protection is needed, it is very easy to change the multiplexing rule at the output of the same encoder in order to allow both coded bits (in the horizontal and vertical dimension) associated with each information bit to be sent through the channel. This is easily seen from Figure 5.3. The order in which the bits are sent is of no importance.

![Figure 5.3 – Rate one third turbo encoder](image)

For a rate of one third, the decoder inputs can be viewed as in Figure 5.4.
5.4 Methods of combining the two different rates

In this chapter we define a block as a sequence of information bits of size equal to the interleaver size used in the turbo encoder. The block also includes the tail bits used to drive the encoder back to the zero state. As shown in the previous chapters, only one tail is needed, no matter how many dimensions the turbo encoder has. We also define a frame as the number of QPSK symbols needed to encode one block of information bits.

5.4.1 Different frame sizes with different coding rates

The easiest way to combine the two different rates is to encode one block of B bits with the rate half code and produce B QPSK symbols followed by another block of B bits encoded with the rate third code to produce 1.5B QPSK symbols. If it is acceptable to have frames of different lengths, then the same turbo decoder can be used in decoding both frames because the same interleaver size is used. In Figure 5.5 we present the bit error ratio for two consecutive frames as a function of the SNR for a block size B of 104 bits (102 information bits plus 2 tail bits).

The first frame is coded rate half and will have 104 QPSK symbols. In this case the SNR equals the $E_b/N_0$. The BER values correspond to those in Figure 3.14 for a rate half code and interleaver size of 102 bits.

The second frame is coded rate third and will have 156 QPSK symbols. This means that two information bits are transmitted in three QPSK symbols. The equivalent $E_b/N_0$ is $10\log(3/2) \approx 1.76$ dB more than the value of the SNR. The BER values correspond to those in Figure 3.23 for a rate third code and interleaver size of 102 bits.

It is interesting to note the shape of the BER as a function of the bit position in the block. The bits at the beginning and at the end of a block and those at a multiple of the number of columns have a slightly lower BER due to the structure of the helical interleaver as explained in Section 3.6.4.

The receiver must be able to identify each frame type and demultiplex the received symbols accordingly. The turbo decoder does not need to know which frame is which, the interleaver size being constant. From this point of view, the architecture is similar to that of RCPC codes.
If one needs a different block size for the information bits coded rate third than the block size for the information bits coded rate half, the interleaver size must be changed too. This is not difficult to achieve as will be seen in the following examples.

If having different frame sizes is too cumbersome from a synchronisation point of view, there are two options for using the same frame length. The first is to send frames of size equal to 1.25B QPSK symbols. At the receiver end, the first B QPSK symbols are decoded first, the rest of 0.25B being delayed until the next frame of 1.25B is received and a new iterative decoding can start. The second way to obtain equal size frames is to encode one block of B bits with the rate half code followed by another block of $\frac{2}{3}B$ bits encoded with the rate third code to produce a frame with the same number of QPSK symbols. The smaller interleaver size of the rate third code will slightly decrease the correction capability of the code. However, the turbo decoder must be able to switch from one interleaver size to another at each received frame.

Due to the fact that the rate is changed at block boundaries, the probability of error for each bit in a block is the same. There are no “spill over” effects when we cross block boundaries due to the fact that using a MAP algorithm, each block is independently decoded from the other block.

Using the second option, we plot in Figure 5.6 the bit error ratio for an $E_b/N_0$ of 0.0dB and for two consecutive frames, each of 234 QPSK symbols.

Figure 5.5 – BER for two blocks of size $B = 104$ bits each at SNR = 0.0, 1.0 and 2.0 dB
Figure 5.6 – Different interleaver sizes at SNR = 0.0dB
The first frame contains 232 information bits plus 2 bits for the tail (for a four state turbo code) coded by a rate half code with a simile odd–even helical interleaver with 13 rows and 18 columns. The second frame contains 154 information bits plus 2 bits for the tail (for a four state turbo code) coded by a rate one third code with a simile odd–even helical interleaver with 13 rows and 12 columns. Some overhead may be needed for rate compatible turbo codes (RCTC) to synchronise the multiplexer at the transmitter with the demultiplexer at the receiver end. The interleaver is usually implemented using up or down counters which count modulo the number of rows/columns. In the above example, to change from interleaver size of 234 to 154 the column counter will switch from counting modulo 18 to modulo 12. The drawback of this method is that, due to equal frame sizes, one cannot vary the ratio of information bits coded rate half and rate third inside a frame.

5.4.2 Same frame size with different coding rates and one interleaver

A solution to the previous problem is to combine the two rates in the same frame. In this case we can choose how many information bits we want with a higher protection level. We simulated three cases: the first 0.25B, 0.5B and 0.75B information bits are encoded with a rate third code and the rest are encoded with a rate half code for different SNR values. We consider a fixed block size B of 310 information bits.

The first case we considered is a block where the first 0.25B information bits are encoded with a rate third code and the rest are encoded with a rate half code. The probability of error for an AWGN channel as a function of bit position is plotted in Figure 5.7 for a block size of 312 bits (310 information bits plus two “tail” bits for a four state turbo code) and a SNR of 0.0 dB, 1.0 dB and 2.0 dB. Using QPSK modulation the frame size is $78 \times \frac{3}{2} + 234 = 351$ QPSK symbols.

From the graph we notice again the better BER for those bits at multiples of 24 (312 = 13 rows $\times$ 24 columns). Also, the transition between the two rates is not as sharp as in the previous case. Another observation is that the average BER for the rate half code is lower than expected for this interleaver size. Conversely, the average BER for the rate third code is higher than the expected value for the same interleaver size. It is as if some of the correcting power of the rate third code was transferred to the rate half code due to the mixing of the information bits coded rate half with those coded rate third.

The second case we considered is a block where the first 0.5B information bits are encoded with a rate third code and the rest are encoded with a rate half code. The probability of error for an AWGN channel as a function of bit position is plotted in Figure 5.8 for a block size of 312 bits (310 information bits plus two “tail” bits for a four state turbo code) and SNR of 0.0 dB, 1.0 dB and 2.0 dB. Using QPSK modulation, a frame size of $156 \times \frac{3}{2} + 156 = 390$ QPSK symbols is obtained.
Figure 5.7 – BER for a 1/4 ratio at SNR = 0.0, 1.0 and 2.0 dB (B = 310)

Figure 5.8 – BER for a 1/2 ratio at SNR = 0.0, 1.0 and 2.0 dB (B = 310)
The third case we considered is a block where the first 0.75B information bits are encoded with a rate third code and the rest are encoded with a rate half code. The probability of error for an AWGN channel as a function of bit position is plotted in Figure 5.9 for a block size of 312 bits (310 information bits plus two “tail” bits for a four state turbo code) and a SNR of 0.0 dB. For QPSK modulation, we get a frame size of \(234 \times \frac{3}{2} + 78 = 429\) QPSK symbols.

![Figure 5.9 – BER for a 3/4 ratio at SNR = 0.0 dB (B = 310)](image)

For a rate third turbo code, the SNR value of 0.0 dB gives an \(E_b/N_0\) of 1.76 dB. For a similar interleaver size of 306 bits, the expected BER at an \(E_b/N_0\) of 1.76 dB can be read from Figure 3.25 as around \(2 \times 10^{-4}\). As before, we note from Figure 5.9 a higher BER value for the rate third part of the code. This is due to interleaving both rates with the same interleaver, so information bits coded rate half are mixed with information bits coded rate third.

From the last three graphs we also note that increasing the \(E_b/N_0\) value or the proportion of rate third coded bits in a frame, a better BER is obtained for the bits positioned at multiples of 24, the number of columns of the block helical interleaver.

We now study the performance of this combining method as a function of the block size B. We consider a block in which the first 0.5B information bits are encoded with a rate third code and the rest are encoded with a rate half code. For a SNR = 1.0 dB, the probability of error for an AWGN channel as a function of the position of the bit relative to the cross over point is plotted in Figures 5.10 and 5.11 for a block size B of 104 and 206 bits. In Figure 5.8 we plot the BER at a SNR = 1.0 dB and B = 312 bits.
Figure 5.10 – BER for a 1/2 ratio at SNR 1.0dB (B = 104)

Figure 5.11 – BER for a 1/2 ratio at SNR 1.0dB (B = 206)
As expected, the higher the interleaver size is, the better the performance we get. An explanation for the rugged shape of the BER curves can be the limited number of errors counted for each bit in the frame.

5.4.3 Same frame with different coding rates and two interleavers

A solution to avoid such a sharp degradation in performance is to use a two level interleaver. It is based on the following observations:

- For the MAP decoder, the information bits at the beginning and at the end of a transmitted block have a lower BER. This is explained by the fact that the encoder starts and ends in a well defined state, say state zero. This gives these bits a higher level of protection at the beginning and at the end of the frame.
- If a block of information bits is coded by a rate third code, it is necessary that the interleaved sequence of information bits contains only bits coded rate third. This implies using different interleavers, one for the rate half code and another for the rate third code.

The information bits are organised as in Figure 5.12. The bits included in blocks A1 and A2 will be encoded by a rate third code and interleaved by interleaver IA. The bits included in block B will be encoded by a rate half code and interleaved by interleaver IB. As long as the interleaved sequence of information bits follows the same pattern (i.e., belong to the same sequence) as the straight sequence of information bits, the two outputs of the two interleavers can be connected in series.

A computer simulation was performed for a block of 432 information bits at an $E_b/N_0$ of 0.0 dB. Block A1 had 72 bits, block B had 312 bits and block A2 had 48 bits. Block B was interleaved by a simile odd–even helical interleaver (IB) with 13 rows and 24 columns (312 bits). The two blocks A1 and A2 where interleaved by a simile odd–even helical interleaver (IA) with 5 rows and 24 columns (120 bits). The BER performance of the MAP decoder is plotted in Figure 5.13.

It is obvious that there is a performance improvement in using a two level interleaver compared to one level interleaving. There is no “spill over” effect when switching from one rate to another and each rate provides the expected error probability. The transition regions are comparable or even better than those for RCPC codes presented in [65].
Figure 5.13 – BER for a two level UEP with two level interleaving
5.5 Interleaver design for three dimensional turbo codes

A natural extension of turbo codes to more than two dimensions was introduced in [25]. The systematic three dimensional turbo encoder (3D–TC) generates three coded sequences which are multiplexed, and shown in Figure 5.14.

In [67], using a different terminology, encoders with multiple codes were described. A trellis termination method was presented and unequal rate component codes were introduced to obtain better performance.

In order to use a MAP decoding algorithm the initial and the final state of each one dimensional encoder should be the same for all three coded sequences. This could be achieved by appending three different “tails”, one for each coded sequence, which will reduce the bit rate. A method similar to the one described in the previous section will be used to create a “simile” interleaver for a 3D–TC which needs only one tail to be appended to the information sequence.

We denote the encoder memory size of each one dimensional encoder as \( v \). We can rearrange the whole block of \( N \) information bits into \( \text{mod} (v + 1) \) sequences. The important advantage in doing this is that, from the point of view of the final encoder state, the order of the individual bits in each sequence does not matter as long as they belong to the same sequence. The “simile” interleaver must perform the interleaving of the bits within each particular sequence in order to drive the encoder into the same state as the state without interleaving. For 3D–TC the information sequence is stored row-wise and the two interleaved sequences start from the left corners as described in Figure 5.15; bottom left corner for interleaver \( I^a \) and top left corner for interleaver \( I^b \). The number of columns, \( \text{COLS} \), must be a multiple of \( (v + 1) \) in order to output symbols which belong to the same sequence pattern when the column counter wraps around.

If puncturing is needed, we can still use the interleaver in Figure 5.15. However, \( \text{COLS} \) must also be a multiple of the dimension order (three for 3D–TC). We can multiplex
the coded bits of the straight sequence whose index in time modulo three is zero with the interleaved I\textsuperscript{a} coded bits whose index in time modulo three is one and with the interleaved I\textsuperscript{b} coded bits whose index in time modulo three is two. In this way all information bits have associated with them one and only one coded bit, so the coding power is uniformly distributed.

![Diagram](image)

Figure 5.15 – “Simile” interleaver for 3D–TC

The MAP algorithm must be modified in order to take into account the \textit{a priori} information given by the previous decoding stages. Each decoder should use as \textit{a priori} information the previous extrinsic information provided by the decoding in the other two dimensions. The soft outputs produced by one decoder described in relation (3.11) should now be modified as

$$L(d_k) = L_c x_k + L_a^*(d_k) + L_e(d_k)$$

(5.1)

where \(L_a^*(d_k)\) represents the sum of the extrinsic information produced by the other two decoders.

In [67] a parallel turbo decoder was used with all uni–dimensional decoders operating in parallel at any given time as in Figure 5.16. The input arrows to each block represent the \textit{a priori} information. The output arrows represent the extrinsic information generated by that particular block.

![Diagram](image)

Figure 5.16 – Parallel structure for a 3D–TC decoder
In the following sections we use a serial turbo decoder as described in Figure 5.17.

![Diagram](image.png)

Figure 5.17 – Serial structure for a 3D–TC decoder

A mathematical analysis of turbo codes is presented in [51]. The analytical approach was based on the definition of a uniform interleaver which is a probabilistic device which maps a given input sequence of Hamming weight $w$ into all distinct permutations of it with the same probability. We consider such an uniform interleaver of length $N$ and a turbo code with two identical convolutional constituent codes. We define $e_w$ as an error event with weight $h = w + z$, where $w$ is the number of information bits and $z$ the number of parity–check bits. We also define $w_{\text{min}}$ to be the minimum $w$ in any finite–weight error event and $e_{w_{\text{min}}}$ the corresponding error events.

In [51] it is shown that the probability of error events with lowest weight produced by error events $e_{w_{\text{min}}}$ is

$$P = \frac{N}{\left(\begin{array}{c} N \\ w_{\text{min}} \end{array}\right)} = N^{(1-w_{\text{min}})} w_{\text{min}}!$$  \hspace{1cm} (5.2)

For systematic non–recursive convolutional encoders and linear block codes, $w_{\text{min}}$ is equal to 1. Thus the performance of these codes is independent of the interleaver length $N$. For recursive encoders, $w_{\text{min}}$ is always greater than 1 and in particular for rate $1/n$ recursive encoders it is equal to 2, which explains the increased coding gain for increased interleaver sizes. The analysis in [51] was performed for two dimensional turbo codes. Following the same line of reasoning, the 3D–TC probability of error events with lowest weight produced by error events $e_{w_{\text{min}}}$ is

$$P = \frac{N^2}{\left(\begin{array}{c} N \\ w_{\text{min}} \end{array}\right)^2} = N^{2(1-w_{\text{min}})} (w_{\text{min}}!)^2$$  \hspace{1cm} (5.3)

Thus, the probability of error will be proportional to $N^{-2}$ and in the general case of $M$ dimensional turbo codes (MD–TC) it will be proportional with $N^{1-M}$. This explains the improvement in performance of multi–dimensional turbo codes.

With this interleaving technique we plot in the following two sections the BER for four state and sixteen state turbo codes for different interleaver sizes. The flooring effect for the rate half and the rate third turbo codes appears after eight iterations. For the rate quarter turbo codes we need a higher number of iterations, as will be explained later.
5.5.1 Three dimensional four state turbo code

Figures 5.18 to 5.21 shows the BER results for a four state turbo code, eight iterations, interleaver sizes of 102, 204, 306 and 3660 bits and rate half (y and y\textsuperscript{a} are punctured), rate third (y and y\textsuperscript{a} transmitted) and rate quarter (y, y\textsuperscript{a} and y\textsuperscript{b} transmitted).

### Figure 5.18
BER for a four state rate half, third and quarter codes and interleaver size of 102 bits (AWGN channel)

### Figure 5.19
BER for a four state rate half, third and quarter codes and interleaver size of 204 bits (AWGN channel)
From the previous results some interesting conclusions can be drawn. The rate third code is always better than the rate half code. The rate quarter code for very low $E_b/N_0$ is slightly worse than the rate third code. For increased $E_b/N_0$ the gain of the rate quarter code becomes very significant. It is important to observe that the cross over point changes for different interleaver sizes: 1.2 dB for 102 bits, 0.7 dB for 204 bits, 0.3 dB for 306 bits and below 0 dB for 3660 bits. Overall the rate quarter turbo code gives the best results with up to
2 dB improvement over the rate third code.

The BER as a function of the number of iterations is plotted in Figure 5.22 for the rate half code at $E_b/N_0 = 0$ dB, 1 dB and 2 dB. In Figure 5.23 we plot the BER for the rate third code and a similar interleaver size.

![Figure 5.22 – BER for four state rate half code at $E_b/N_0 = 0$ dB, 1 dB and 2 dB (interleaver size of 3660 bits, AWGN channel)](image1)

![Figure 5.23 – BER for four state rate third code at $E_b/N_0 = 0$ dB, 1 dB and 2 dB (interleaver size of 3660 bits, AWGN channel)](image2)

For the rate half and the rate third codes and interleaver sizes less than 400 bits, eight iterations were sufficient to achieve most of the coding gain, as was seen in Sections 3.7.1
and 3.7.2. For larger interleaver sizes, for example 3660 bits or more, the number of iterations needed depends on the value of $E_b/N_0$.

Figures 5.24 to 5.25 show the BER results for a four state rate quarter turbo code as a function of the number of iterations at an $E_b/N_0 = 0$ dB. This is specific only for larger interleaver sizes and at low values of $E_b/N_0$.

Figure 5.24 – BER for a four state rate quarter code at $E_b/N_0 = 0$ dB (interleaver size of 3660 bits, AWGN channel)

Figure 5.25 – BER for a four state rate quarter code at $E_b/N_0 = 0$ dB (interleaver size of 16500 bits, AWGN channel)
5.5.2 Three dimensional sixteen state turbo code

Figures 5.26 to 5.29 show the BER for a sixteen state turbo code, eight iterations, interleaver sizes of 110, 220, 420 and 930 bits and different rates: rate half (y and y^a are punctured), rate third (y and y^a transmitted) and rate quarter (y, y^a and y^b transmitted).

Figure 5.26 – BER for a sixteen state rate half, third and quarter codes and interleaver size of 110 bits (AWGN channel)

Figure 5.27 – BER for a sixteen state rate half, third and quarter codes and interleaver size of 220 bits (AWGN channel)
Figure 5.28 – BER for a sixteen state rate half, third and quarter codes and interleaver size of 420 bits (AWGN channel)

Figure 5.29 – BER for a sixteen state rate half, third and quarter codes and interleaver size of 930 bits (AWGN channel)
From the previous graphs we note a different scenario for the sixteen state turbo code when compared with the four state turbo code. Surprisingly the rate quarter code performs worse than the rate half code at low $E_b/N_0$ and always worse than the rate third code. These results could be due to the sixteen state code performing worse at lower $E_b/N_0$ on its own. Another reason could be that due to the longer constraint length of the sixteen state code, the block helical interleaver used produces highly correlated sequences. This means that whatever interleaver type is used for small blocks of data, we expect bad performance for long constraint length codes. A pseudo–random or random interleaver should be used for larger interleavers.

We could also expand the two dimensional interleaver to a three dimensional interleaver. The bits are written at location $(z, y, x)$ with $x$ the fastest index. Reading can start for the first interleaver from the $(Z, Y, 0)$ corner decrementing $z$ and $y$ and incrementing $x$. For the second interleaver, one can start from the $(Z, 0, 0)$ corner decrementing $z$ and incrementing $x$ and $y$. The advantage of using a three dimensional interleaver over a two dimensional one is an increased “distance” in time between two consecutive symbols in the interleaved sequences.

The extension to a rate $1/n$ turbo code is straightforward as shown in Figure 5.30. Depending of the multiplexing rule, this encoder can provide a variable coding rate, from rate one to rate $1/n$.

![Figure 5.30 – A n–dimensional turbo encoder](image)

This could be a future topic for investigation to decrease the bit error probability. However, we must emphasise that the coding gain which can be achieved by increasing the number of dimensions $n$ depends on the type and depth of the interleaver. Also the greater $n$ is, the more correlated the interleaved sequences are and the gain will diminish accordingly.

If we compare the performance of the four state and the sixteen state turbo code, it is clear that at low $E_b/N_0$, the four state code performs better. This observation can be used to build a “mixed–state” turbo code as described in Figure 5.31. The turbo decoder will
decode for the first N iterations only the four state rate third code \((x_k, y_k, z_k)\). This will “clean” the signal which is equivalent to an improvement of the SNR. At this increased SNR the sixteen state code is more powerful than the four state code. This is why in the next M iterations the turbo decoder should start decoding the more complex rate quarter code \((x_k, y_k, z_k, w_k)\). This “mixed–state” turbo code can be compared with an inner code and an outer code in a standard concatenated system with different correction capabilities [8]. This is another interesting area to be investigated in the future.

![Turbo Encoder Diagram](image)

**Figure 5.31 – A “mixed–state” turbo encoder**

### 5.6 Conclusions

This chapter described how turbo codes can be used in adaptive rate systems. Variable rate and variable error protection can be achieved with turbo codes.

The same decoder structure can be used for rate half and rate third codes. The two rates can be transmitted in separate blocks to achieve the best performance, or can be mixed together inside the same block. In the last case, there is a compromise between the expected BER values for the two rates.

The particular interleaver we used in our simulations, the block helical interleaver, has an interesting property which allows some bits to have a higher protection than others. Each time the column counter wraps around, the previous output bit in the interleaved
sequence is preceded and followed by bits at a larger distance than the interleaver depth. This causes a more reliable soft output from the turbo decoder which means a lower BER. If in a frame there are only a few bits which need a higher protection, we could place those bits in the last column of the block helical interleaver.

A better approach to obtain higher coding gains is to use multi–dimensional turbo codes. The decoder can have a parallel or serial structure. In our simulations we used a serial structure because we used only one MAP decoder working at any time. This serial structure also has the advantage of using the very last outputs of the uni–dimensional decoders which could be an advantage in a “mixed–state” decoder. The “mixed–state” decoder decodes the most powerful code first, followed by lesser powerful codes. A characteristic of multi–dimensional codes is their higher error correction capability which is proportional with \( N^{1-M} \), where \( N \) is the interleaver size and \( M \) is the dimension of the code. The number of iterations needed to reach the flattening area of BER curves is in the order of tens, even hundreds, which is much higher than for a two dimensional turbo code.

Multi–dimensional codes can also be used for encryption purposes as described in Section 3.10.

The high coding gain of the three–dimension turbo code using small interleaver sizes, (BER of \( 1.2 \times 10^{-5} \) at an Eb/N0 = 2 dB and interleaver size of 420 bits), makes it suitable for speech transmissions.

The multi–dimensional turbo code can be used for different rates, with only one encoder/decoder pair. The only overhead being the synchronisation of the multiplexer at the transmitter with the demultiplexed at the receiver.

The unequal error protection capability of turbo codes is very useful for mobile radio systems where the transmission conditions are non–stationary. In these systems the channel quality is usually monitored in order to vary the power of the signal. The same control signals could be used to vary the coding rate.
6 ITERATIVE DECODING OF PRODUCT CODES

6.1 Introduction

The turbo code structure is similar to that of product codes [52]. Both product codes and turbo codes encode twice the same information bits. In this chapter we compare the performance of these two classes. We examine two construction schemes for product codes and compare these to turbo codes with equivalent parameters. The most important result of this analysis is that the performance of product codes does not improve significantly with the increase of the interleaver size. The mathematical justification of this observation comes from [50].

6.2 Review of different product coding schemes

The use of block codes to construct a product code came naturally due to their easily visualised construction. There are a number of interesting approaches to designing product codes which we summarise below. In [35], the iterated product of parity check codes is presented as a means of designing good long codes. The conventional criterion of minimum distance is shown to be irrelevant for long block codes. A long code is defined as a code for which np \gg 1 where n is the code length and p is the error probability. In [23], the cross entropy minimisation principle is proposed as a better criteria. The cross entropy, or the relative entropy of two discrete distributions \( p = \{p_i\} \) and \( q = \{q_i\}, i = 1,2,...,n \) with \( \sum_{i=1}^{n} q_i = \sum_{i=1}^{n} p_i = 1 \) is defined as \( H(p, q) = \sum_{i=1}^{n} p_i \log \left( \frac{p_i}{q_i} \right) \). The minimisation is performed over the distribution \( p \), with fixed \textit{a priori} distribution \( q \), and some constraints on \( p \)’s moments. The \textit{a priori} distribution \( q \) considered is the binomial distribution of an \( n \)–tuple resulting from an independent equally distributed random choice of each of its symbols. Following [23] the best code would be a distribution \( p \) for which \( H(p, q) \) is minimum.

An example of an asymptotically good code is the product code of parity–check codes. A three dimensional product code is shown in Figure 6.1 (reproduced from [35]). Each code symbol is a cube, or a 3–D vector, and the code is a rectangular array of such cubes in the 3–dimensional space: 3 cubes in the \( x \) direction, 4 in the \( y \) direction and 5 in the \( z \) direction. The information symbols are denoted by \( i \) and the parity check symbols by \( C \). This model can be extended to an arbitrary number of dimensions.

In [53] powerful codes are obtained by using simple block codes to construct multi–dimensional product codes. The decoding is done using a separable MAP filter which allows soft decisions to be passed from one iteration to the next. A one–dimensional MAP filter is used sequentially in each dimension. The probabilities of error obtained at the first step are further refined by another MAP filter used for the next dimension, which completes a single filtering cycle. After the first cycle, the resulting word may not be a valid codeword, but, by iterating the decoding operation a small number of times, the MAP
algorithm is be able to “capture” valid codewords.

In [54, 55] an improved version of this construction is presented. A “Partial factor MAP filtering” algorithm is used for each component code in a given dimension, but only for the refinement factors that correspond to the filtering passes in the other dimensions. This reduces the errors introduced by the independence assumption in the MAP processing. In [56] it is shown that MAP filtering follows from entropy optimization principles.

In [61], a study of iterative decoding of product codes was performed for different interleaving strategies based on combinatorial configurations. A near optimum method of iterative decoding of product codes was also presented in [62].

The first paper to propose an analytical solution to the problem of bounding the performance of parallel concatenated block codes (PCBC) was [63]. The average performance of a PCBC is investigated based on the input–redundancy weight enumerating function using an “uniform interleaver”. An uniform interleaver is defined as a probabilistic device which maps a given input word of weight w into all distinct permutations \( \binom{k}{w} \) with equal probability \( \binom{k}{w}^{-1} \).

In [64] it was shown that soft decision maximum likelihood decoding of any (n,k) linear block code over GF(q) can be accomplished using the Viterbi algorithm applied to a trellis with no more than \( q^{(n-k)} \) states. The trellis starts and ends in the same zero state. This characteristic allows the use of the MAP decoding algorithm without the need of appending any tail to the information sequence in order to drive the encoder to the zero state.
6.3 Turbo coding the (15, 11) binary cyclic code

Here we consider the same (15, 11) binary cyclic code with generator polynomial \( g(x) = x^4 + x + 1 \) as in [64]. This is a Hamming code having a minimum Hamming distance of 3, so it is a single error correcting code. The encoder design is given in Figure 6.2. Initially the switch is in position 1 and the gate is open. The first eleven information bits are entered in the shift register and at the same time are sent through the channel. Then the switch changes to 2 and the gate closes; the four bits of the shift register are shifted out one at a time.

![Figure 6.2 – Encoder for binary (15,11) code with \( g(x) = x^4 + x + 1 \) ](image)

The contents of the shift register represents the state of the encoder. The encoder starts in state zero and ends in state zero. The path through the trellis for the codeword 101001011101101 is shown in Figure 6.3.

![Figure 6.3 – Path in the trellis for codeword 101001011101101 ](image)

We analyse the performance of this code used in a product/turbo code scheme for two different interleaver constructions and different interleaver sizes.
6.3.1 Construction I

A two dimensional turbo encoder is shown in Figure 6.4 where each uni-dimensional encoder is the (15, 11) binary cyclic block code. For a clear presentation of the multiplexing which is performed at the output of the encoder, we ignore the delays and the random order of the coded bits in the vertical dimension after interleaving. After the eleven information bits are sent through the channel, the next four bits transmitted are the contents of the shift register of ENC^-1. They are followed by four bits which are the contents of the shift register of ENC. The decoder must follow the same demultiplexing pattern. The equivalent rate of the turbo code is \( \frac{11}{19} \approx 0.58 \), slightly higher than for the original rate half turbo code.

![Figure 6.4 – Turbo encoder used for block codes to produce a product code](image)

The output of the turbo encoder is as shown in Figure 6.5.

<table>
<thead>
<tr>
<th>TIME</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>( d_1 )</td>
<td>( d_2 )</td>
<td>( d_3 )</td>
<td>( d_4 )</td>
<td>( d_5 )</td>
<td>( d_6 )</td>
<td>( d_7 )</td>
<td>( d_8 )</td>
<td>( d_9 )</td>
<td>( d_{10} )</td>
<td>( d_{11} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIME</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>( Y_{12} )</td>
<td>( Y_{13} )</td>
<td>( Y_{14} )</td>
<td>( Y_{15} )</td>
<td>( Y_{i} )</td>
<td>( Y_{i+1} )</td>
<td>( Y_{i+2} )</td>
<td>( Y_{i+3} )</td>
</tr>
</tbody>
</table>

![Figure 6.5 – Turbo encoder output](image)

The equivalent model of the product code is shown in Figure 6.6. This is identical to Construction I from [61]. The encoding process is as follows:

- a block of \( m \times k \) information bits is stored in the DATA array as \( m \) rows and \( k \) columns (in this case \( n = 15, k = 11 \) and \( m \) is an integer number)
- each row of the DATA array is encoded “horizontally” to produce the array PH of parity bits of size \( m \times (n-k) \).
- the DATA array is interleaved and encoded “vertically” to produce the array PV of parity bits of size \( m \times (n-k) \).
The darker areas represent the transmitted bits. The rate of the product code is
\[ \frac{k}{2n-k} = \frac{11}{2 \times 15 - 11} = 11/19 \approx 0.58. \]

At the receiver end, the decoding algorithm is as follows:

- the “horizontal” decoder produces the extrinsic information for the received DATA array
- the received DATA array is interleaved and input to the “vertical” decoder together with the interleaved extrinsic information which is used as the \textit{a priori} input to the “vertical” decoder
- the “vertical” decoder produces the new extrinsic information which will be deinterleaved and used as \textit{a priori} input in the “horizontal” decoder in the next iteration, and so on.

The results are shown in Figure 6.7 for a helical block interleaver of 132 bits. Most of the gain is achieved after the first four iterations. Note that the \textit{a priori} information is provided only for the information bits in the DATA array. The decoding of the parity bits does not use any \textit{a priori} information.

The performance of this scheme is compared with that of a turbo code based on a sixteen state convolutional encoder as shown in Figure A.3 Each uni–dimensional convolutional code has a minimum distance of five. First we consider the same interleaver size of 110 bits. The bit error ratios for the product code and the turbo code are plotted in Figure 6.8. Due to a larger minimum distance, the turbo code performs better.

We now increase the interleaver size to 220 bits to compare the performance of both product and turbo code. The bit error ratio is plotted in Figure 6.9.
Figure 6.7 – BER for a product code based on the (15, 11) cyclic code and 11×12 bit helical interleaver using construction I (rate = 0.58)

Figure 6.8 – Comparison between a product code and a turbo code (16 states, 110 bit helical interleaver)
Note that if there is an improvement in the bit error rate for the turbo code, the performance remains almost the same for the product code. Computer simulations for increased interleaver size up to 1000 bits gave similar results for the product code shown in Figure 6.9. This will be explained in Section 6.3.3.

We could apply a similar puncturing technique to increase the coding rate. Instead of sending eight parity bits, we send only four: $Y_{i+2}$, $Y_{i+7}$, $Y_{i+14}$, $Y_{i+19}$. The coding rate becomes $11/15 \approx 0.73$ and the bit error ratio increases significantly as shown in Figure 6.10.

Figure 6.10 – BER for a product code based on the $(15, 11)$ cyclic code 11×12 bit helical interleaver with punctured parity bits (rate = 0.73)
6.3.2 Construction II

We examine the performance of a more complex encoder which can provide *a priori* information for the parity bits as well. We consider two block codes $C_1(n_1, k_1)$ and $C_2(n_2, k_2)$. The encoding process is as follows, with $m$, $t$, and $s$ chosen appropriately:

- a block of $m \times k_1$ information bits is stored in the DATA array as $m$ rows and $k_1$ columns
- each row of the DATA array is encoded “horizontally” to produce the array $P_1$ of parity bits of size $m \times (n_1 - k_1)$.
- the (DATA + P1) array of size $m \times n_1$ is interleaved, reorganised as $t \times k_2$ bits and encoded “vertically” to produce the array $P_2$ of size $t \times (n_2 - k_2)$.
- the $P_2$ array is interleaved and reorganised as $s \times k_1$ bits and encoded “horizontally” to produce the array $P_3$.

The transmitted data is shown as the darker areas in Figure 6.11.

![Figure 6.11 – Construction II of a product code](image)

The rate of the product code is

$$\text{rate} = \frac{mk_1}{mn_1 + s(n_1 - k_1) + t(n_2 - k_2)} = \frac{1}{\frac{n_1}{k_1} \left( \frac{n_1 n_2}{k_1 k_2} + \frac{n_1}{k_1} + 1 \right)}.$$

This construction is identical to Construction II in [61] for the first three steps. Here, we have added an extra step which encodes the interleaved $P_2$ array to produce the $P_3$ array. In [61] it was possible to compute the *a priori* information for $P_1$ only. With our construction both parity arrays $P_1$ and $P_2$ can provide the *a priori* information. We consider the particular case when the two codes are identical to the $(15, 11)$ binary cyclic code, $m = 121$, $t = 165$ and $s = 60$. The number of information bits in the DATA array is $m \times k_1 = 121 \times 11 = 1331$ bits.
The rate is computed with the previous formula where \( n_1 = n_2 = 15 \) and \( k_1 = k_2 = 4 \) and gives 0.49.

At the receiver end, the decoding algorithm is as follows:

- the “horizontal” decoder decodes all \((m + s)\) codewords and produces the extrinsic information for the received DATA array, the received P1 array and the received interleaved P2 array.
- the extrinsic information for the DATA and the P1 arrays is interleaved
- the extrinsic information for the interleaved P2 array is deinterleaved
- both above extrinsic informations are used as \textit{a priori} information in the “vertical” decoder which has as an input the interleaved \((DATA + P1)\) array and the deinterleaved P2 array.
- the “vertical” decoder decodes all \(t\) codewords and produces the extrinsic information for the interleaved \((DATA + P1)\) array and P2 array
- the extrinsic information for the DATA and the P1 arrays is deinterleaved
- the extrinsic information for the P2 array is interleaved
- both above extrinsic informations are used as \textit{a priori} information in the next iterative decoding step

The performance of construction II is plotted in Figure 6.12.

![Figure 6.12 – BER comparison for a product code based on the (15, 11) cyclic code using construction I (rate = 0.58) and construction II (rate = 0.49)](image-url)
In Figure 6.12, we also plot the performance of the product code based on the same (15, 11) binary cyclic code and a similar number of information bits, \( m \times k_1 = 120 \times 11 = 1320 \) bits using Construction I.

Comparing the bit error rates for Construction II and Construction I for the same code and the same interleaver size we find that Construction I gives slightly better performance. Construction II, which is also more complex, can be compared with a turbo code which can provide not only extrinsic information for the information bits but also extrinsic information for the coded bits. These two quantities are almost identical because they are produced as a function of the received sequence less the current symbol. From the information point of view there is no extra gain in using the same information twice.

Increasing the interleaver size does not produce a significantly better performance for product codes. Most of the coding gain can be achieved with a block interleaver with the number of columns equal to the number of rows which in turn is equal to the information size of the codeword. A further increase in the interleaver size will not improve the performance of the product codes, as is explained in the next section.

### 6.3.3 Limiting factors for product codes

In Section 3.9 we explained why the performance of turbo codes increases with the increase in interleaver size. In this section we use a similar approach to explain why the performance of product codes should not increase with the interleaver size.

For a block code, the extrinsic information \( E_i \) associated with the information bit \( d_i \) is produced by the MAP algorithm using a forward and backward recursion applied to one codeword at a time as shown in Figure 6.13.

![Transmitted codeword](image)

![Received codeword](image)

![Extrinsic information output](image)

**Figure 6.13 – Extrinsic information for product codes**

If we consider the code word of length \( B+1+A \) as shown in Figure 6.13, the forward recursion will only use the \( B \) symbols received before the information bit \( d_i \) and the backward recursion will only use the \( A \) symbols received after the information bit \( d_i \). This makes the estimation of the extrinsic information less reliable due to the small number of symbols used (limited by the size of the codeword) when compared with turbo codes.
The best we can expect after interleaving is to produce a codeword which contains the received symbol $R_i$ for the transmitted information bit $d_i$ and all other $(B + A)$ symbols to be different. This can be achieved using a block type of interleaver where the number of rows equals the number of columns which in turn equals the number of information bits. In this way each column-wise produced codeword will only have one bit in common with the row-wise produced codeword. Assuming random information bits, if we increase the interleaver size, we just get a different set of bits which are as “uncorrelated” as in the initial case. We can not get a “less correlated” sequence due to the limited size of the codeword.

This is why when we use the $(15, 11)$ cyclic code, the performance of the product code does not increase significantly with the interleaver size as would be expected for a turbo code based on convolutional codes. Even if we use Construction II, increasing the interleaver size cannot produce better results. The mathematical proof for this was presented in [50, 51] and repeated briefly in Section 5.5.

### 6.4 Conclusions

This chapter compared turbo codes with product codes. From the encoder point of view, the difference between the two is that turbo codes use recursive convolutional codes while product codes are based on systematic block codes.

The recursive property of convolutional codes is the reason why the probability of error decreases when the interleaver size is increased. This property does not apply to product codes as explained in [50, 51]. The power of turbo codes can be increased by increasing the interleaver size and the “randomness” of the interleaver output. For product codes the smallest block interleaver size, which is the square of the number of information bits in a codeword, is as good as any larger interleaver.

Two construction schemes were presented. The difference between them is that one construction can provide a priori information for the parity bits too. This does not lead to a better BER because from the information point of view it does not provide more information.

The main conclusion is that turbo codes offer better performance and more ways to vary the coding gain as a function of the restrictions imposed by the communication system (complexity, delay, etc).
7 ITERATIVE DECODING OF CONCATENATED CODES

7.1 Introduction

In this chapter we present a novel approach to iterative decoding of standard concatenated schemes where the outer code is a Reed–Solomon (RS) code. The advantage of this technique is that it can be used in real time for existing hardware with minimum modifications. It gives about 0.8 dB coding gain after only one iteration.

The originality of this work consists of the way the information from the outer decoder is passed back to the inner decoder for the next iteration. This is explained in the following sections and allows improved decoding of the convolutional code which in turn allows a lower word error rate for the RS decoder. This idea was first discussed by the authors of [68].

7.2 Review of decoding techniques

A standard concatenated coding system using an outer RS outer code and an inner convolutional code is shown in Figure 7.1. Using a classical approach, both decoders work independently of each other; the information provided by the inner decoder, usually a Viterbi decoder, is passed to the outer decoder, the RS decoder in our case.

Iterative decoding techniques used for turbo codes were applied to concatenated codes almost 20 years ago [69]. A MAP decoder was used instead of a Viterbi decoder so that the reliability information about the estimated symbol could be passed from the inner decoder to the outer decoder. A very small improvement of 0.05 to 0.1 dB was achieved. Transfer of information from the outer decoder to the inner one was also present: when the outer decoder detects and corrects the errors made by the inner decoder, the inner decoder is forced to choose specific paths which include these estimates. A further improvement of 0.3 to 0.4 dB was achieved by this forced–state decoding technique.

In [70] a new decoding technique based on repeated decoding trials and exchange of information between the two decoders was introduced. Using the fact that the error correction capability of the RS–decoder can be increased if an error is transformed to an erasure, the RS decoder is allowed to perform repeated trials on the Viterbi decoded data, operating as an error–and–erasure decoder. This decoding technique is described by the
following steps: we assume that at least one RS codeword is correctly decoded; this means that we know the positions of the former errors and we can also assume that the neighbouring symbols in the interleaved sequence were in error. By erasing those symbols, there is a chance that the RS codewords have less errors and more erasures, so the error correction capability of the RS decoder is improved. A coding gain of 0.3–0.4 dB can be obtained. An improvement of 0.5–0.6 dB compared to the standard coding system was obtained with repeated Viterbi decoding trials with forced states [69].

Trellis pruning and the use of an outer RS code with non-uniform correcting capability was described in [71, 72]. The drawback of this method is that the optimal code profile depends on the $E_b/N_0$ used.

In [73, 74] a soft output Viterbi decoder is presented which can provide reliability information to the outer RS decoder while accepting reliability information from a previous RS decoding step in an interleaved scheme. This can be explained using Figure 7.2 as in [74]. We assume that after the first decoding pass two RS codewords are successfully decoded. For the second iteration, the soft output Viterbi decoder is forced through known correct states. This reduces the BER for the inner code which in turn will lead to more RS codewords correctly decoded and so on until the whole block is error free.

![Convolutional code](image)

**Figure 7.2** – Iterative decoding using non-uniform RS codes

A similar approach is taken in [75] where the RS codewords have four different levels of redundancy. The set of codewords with the highest redundancy is decoded first after which the decoded symbols are fed back to the Viterbi decoder. The next highest redundancy codewords are decoded by the RS decoder and so on.

The above decoding steps can be further improved using erasure declarations: a “double-sided” erasure is when all symbols between two known erroneous symbols spaced at less than the interleaver depth are erased; a “single-sided” erasure is when the symbol immediately adjacent to a known erroneous symbol is erased.

All these decoding techniques assume that at least one RS codeword is correctly decoded. In the next section we will need a few consecutive RS codewords to be successfully decoded. This is a function of the constraint length of the feed forward convolutional code.
7.3 Regenerate Transmitted Symbol Algorithm (RTS)

The algorithm described in this section can be applied to any concatenated scheme as shown in Figure 7.1 where the inner code is a feed forward convolutional code.

The approach we take here is that if we have a certain number of RS codewords decoded successfully in a row, we can then substitute certain symbols in the original received sequence and perform a second Viterbi decoding operation over a ‘cleaner’ sequence. We consider the system model in Figure 7.3. Let us assume that the encoder memory order of the inner convolutional code is \( m \) and the interleaver depth used is \( d \). For a feed forward encoder which initially is in an unknown state, after \( m + 1 \) cycles, the encoder state and output will be defined by the last \( m + 1 \) inputs. Now consider the block deinterleaver in Table 7.1 where \( d = 6 \).

Table 7.1 – Block deinterleaver

<table>
<thead>
<tr>
<th>X0</th>
<th>X6</th>
<th>X12</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>X7</td>
<td>X13</td>
</tr>
<tr>
<td>X2</td>
<td>X8</td>
<td>X14</td>
</tr>
<tr>
<td>X3</td>
<td>X9</td>
<td></td>
</tr>
<tr>
<td>X4</td>
<td>X10</td>
<td></td>
</tr>
<tr>
<td>X5</td>
<td>X11</td>
<td></td>
</tr>
</tbody>
</table>

The decoded data produced by the Viterbi decoder represented by the \( \{ x \} \) sequence is written column–wise in the block deinterleaver. Each row of the deinterleaver represents one Reed–Solomon (RS) codeword. Suppose that the RS decoder is able to correct \( m + 1 \) successive RS codewords. Let us consider the encoder memory order \( m = 2 \) and let us assume that the first three RS codewords in Table 7.1 are corrected. This means that from the RS decoded sequence \( \{ u \} \), the following symbols are correctly decoded:

\[ u_0, u_1, u_2, \ldots, u_6, u_7, u_8, \ldots, u_{12}, u_{13}, u_{14}, \ldots \]

If we re–encode this sequence with the same encoder with \( m = 2 \), we will know for sure that the following outputs are correct:

\[ \ldots, v_2, \ldots, v_8, \ldots, v_{14}, \ldots \]

At this point we can start a second decoding process which instead of the received sequence \( \{ r \} \) will use as an input a ‘cleaner’ sequence as described below:

\[ r_0, r_1, v_2, r_3, r_4, r_5, r_6, v_7, r_8, r_9, r_{10}, r_{11}, r_{12}, v_{13}, v_{14}, r_{15}, \ldots \]

The above sequence was produced by substituting in the received sequence the corrected symbols which hopefully will help the Viterbi decoder to produce less errors than in the first decoding process.

The system model in Figure 7.3 is simplified for computer simulations: the darker area is replaced with an RS Simulator block and the lighter areas are removed.
Figure 7.3 – Regenerate Transmitted Symbol Decoder
7.4 Computer simulation results

We are interested in simulating a particular concatenated scheme which was used in a 155.52 Mbit/s codec described in [68]. The coding scheme uses an outer (255,239) RS code and a 16 state, rate 8/9 six–dimensional 8PSK rotationally invariant trellis code as the inner code.

In our computer simulation we simplified the system model from Figure 7.3 to the one shown in Figure 7.4 to test the BER improvement after only one iteration.

![Simulated model](image)

Figure 7.4 – Simulated model

The RS Simulator block is the darker block in Figure 7.4. It simulates a deinterleaver block, an RS decoder block and an interleaver block. The RS decoder is able to correct up to 8 bytes per RS codeword. Inside the RS Simulator block, data is stored in a three dimensional array. The x dimension represents the number of bits in a word, usually 8 bits. The y dimension represents the depth of the interleaver which can be increased up to 16. The z dimension is the length of the RS codeword (by default 255). The maximum number of errors which can be corrected by the RS decoder can also be selected (by default 8). For a block interleaver, the best bit error rate of the output was achieved for an interleaver depth of 16. The rate 8/9 6–D 8PSK Trellis encoder is shown in Figure 7.5. This is a feed forward encoder with encoder memory order of two. It requires an $E_b/N_0$ of 7.5 dB for a bit error ratio of $8 \times 10^{-4}$ in order to have a concatenated bit error ratio of $10^{-10}$ for an ideal interleaver. With the RTS scheme we want to achieve the required BER of $8 \times 10^{-4}$ at a lower value of $E_b/N_0$. 

131
Figure 7.5 – Encoder for rate 8/9 6-D 8PSK Trellis Code

Differential Encoder

Non-systematic Convolutional Encoder

Multi-D Signal Set Mapper

logical XOR

logical AND

mod-4 adder

mod-8 adders
The BER is plotted in Figure 7.6. Note that the BER after one iteration starts to decrease after $E_b/N_0 = 6.3$ dB due to the RS decoder, which begins to clean up the signal. Before that point, the performance is the same as without any iteration.

Figure 7.6 – BER for the rate 8/9 6–D 8PSK Trellis Code before and after one iteration
An important observation is that the RTS decoder does not have any information about the phase. This requires that the differential encoding/decoding to be turned off which could add a further 0.1 to 0.2 dB coding gain but would require more effort in the synchronisation of the decoder.

In our simulations we made the assumption that the RS decoder is able to decode any codewords with \( E = 8 \) or less errors. It is well known that the undetected error rate is bound by \( \frac{1}{E!} \) times the detected error rate [75]. In our case, this means that for each 40320 corrected errors there will be one undetected error. This small decrease in performance must be taken in account in order to have a more realistic estimate of the RTS performance.

A further improvement could be made by taking advantage of the correct decoding of each RS codeword. The corrected symbols would be then used in a determinate state convolutional decoder which forces particular state transitions in a conventional convolutional decoder [76]. However, this path decision forcing algorithm is not currently available in commercial Viterbi decoders.

### 7.5 Conclusions

The RTS algorithm is a simple method to lower the \( E_b/N_0 \) needed to achieve the desired BER for a standard concatenated scheme. For the particular case presented in this chapter, the target BER for the inner code is \( 8 \times 10^{-4} \). With only one Viterbi decoder this can be achieved at an \( E_b/N_0 \) of 7.5 dB. With the RTS algorithm, the same BER can be achieved at an \( E_b/N_0 \) close to 6.7 dB. This gives a 0.8 dB coding gain. The main advantage of the RTS algorithm is that iterative decoding can be performed in real time. In the particular case which was presented, the bit rate is 155.52 Mbit/s. The extra hardware needed is two more decoders, an encoder for the inner code, a multiplexer and some synchronising logic. This is inserted between two standard decoder stages for each iteration. Another advantage is that if the threshold for the RS decoder to start working is not reached, the iterative decoder will perform as well as a single decoder.

The drawback of this algorithm is that it needs a number of codewords equal to the memory order of the inner encoder to be corrected. The complexity of the decoder also increases, in proportion to the number of iterations.

The gain of 0.8 dB is significant when each tenth of a dB means a significant cut in the cost of a communication system.
8 CONCLUSIONS

This thesis discussed iterative decoding techniques for concatenated coding schemes in order to approach channel capacity with lower complexity than maximum likelihood decoding algorithms.

A simplified MAP algorithm was presented in Chapter 2. This algorithm has only four times the complexity of the Viterbi algorithm and can be implemented in hardware [36]. It is a maximum likelihood decoding algorithm which minimises the probability of bit error and provides soft outputs. Comparisons with other implementations were made in order to emphasise the reduction in computations which can be achieved. We also noted the higher dynamic range of its outputs compared with those of a soft output Viterbi algorithm, although both have similar time domain patterns.

The time delay due to the necessity of receiving the whole block of data before the start of the decoding process can be eliminated by having a similar sliding window same as used in the Viterbi algorithm. The decoded bits at the beginning and at the end of the window are ignored due to the errors caused by the unknown starting state. This method increases the number of computations but eliminates the huge delays and memory requirements for larger blocks of data.

The MAP algorithm can be easily modified to accept a priori information which is very useful for some particular channels. In mobile communications, some a priori information can be obtained for different speech parameters such as line spectral pairs, pitch and energy. Using this a priori information leads to a significant improvement of the performance of the decoding algorithm.

The only significant drawback of this algorithm is the need for an accurate estimate of the noise variance. However, in some cases, this can be used to advantage in an encryption scheme as presented in Section 3.10.

In Chapter 3 we presented the principles of iterative decoding. A real time turbo decoder and a modified MAP algorithm to suit iterative decoding was described. An original bandwidth–efficient interleaver used to improve the performance of the MAP decoder was shown. This interleaver needs only one “tail” to be appended to drive all uni dimensional encoders of a multi–dimensional turbo code to state zero. It also produces a uniform distribution of the coding power of punctured turbo codes. The performance of four and sixteen state, rate half and rate third turbo codes, using this new interleaving technique was studied for AWGN and Rayleigh channels at low signal to noise ratios. The analysis was performed for small interleaver sizes and showed that turbo codes can be successfully used in mobile satellite communications.

This original interleaver can also be used in a possible application of turbo codes in the area of cryptography. The idea used is similar to that of spread spectrum techniques: burying the signal in noise and recovering it at the receiver end. This is possible with turbo
codes due to the low $E_b/N_0$ at which they operate. The assumption we made here is that we use a very low $E_b/N_0$ for which each individual sequence of information bits or coded bits is very hard to detect. Only with the high gain of the turbo decoder can one successfully recover the transmitted data from noise. The outputs of the turbo encoder are buried in noise whose variance can be changed in each frame or even in each interleaved sequence. The long pseudo–random generator which generates the starting states of each uni–dimensional encoder, the pseudo–random generators used for each interleaver, and the variance of the noise added at the transmitter are the keys to the proposed encryption system. We assume these keys to be secret and known at the receiver end. From the attacker’s point of view, assuming the coding scheme is known, it would be very difficult to find all the missing variables needed by a turbo decoder. This is an interesting area for future investigation.

In Chapter 4 an original diversity combining technique was applied to turbo codes which gave a 0.5 dB improvement over conventional combining techniques. The novelty of the proposed scheme is that the transmitted rate half turbo code is transformed at the receiver into a more powerful rate third turbo code. Both AWGN and Rayleigh channels were studied. Simulations using real data showed that turbo codes outperform other conventional coding schemes when used in real fading channels.

The original concept of rate compatible turbo codes was introduced in Chapter 5. They can offer a variable coding gain using the same turbo encoder/decoder structure. This unequal error protection capability of turbo codes is very useful for mobile radio systems where the transmission conditions are nonstationary. In these systems the channel quality is usually monitored in order to vary the power of the signal. The same control signals could be used to vary the coding rate. Different methods of combining two different rates were presented. Multi–dimensional turbo codes were introduced as an alternative to increase the performance of turbo codes when small interleaver sizes are used.

By comparing the performance of the four state and the sixteen state turbo code, it becomes clear that at low $E_b/N_0$ the four state code performs better. The “mixed–state” turbo code described in this chapter decodes the more powerful four state code first. This achieves a “cleaning” of the signal which is equivalent to an improvement of the SNR. At this increased SNR the sixteen state code is more powerful than the four state code. This is why in the following iterations the turbo decoder will start decoding the more complex sixteen state code. This “mixed–state” turbo code can be compared with an inner code and an outer code in a standard concatenated system with different correction capabilities. This is another interesting area to be investigated in the future.

Chapter 6 presented a comparison between the performance of turbo codes and product codes. Two construction methods are used for product codes. The difference between them is that one method provided a priori information for the parity bits, while the other did not. This did not improve the BER because from the information point of view since it did not provide more information.
The performance of product codes is not affected by the interleaver size. The interleaving factor comes into play only for recursive codes and does not apply to product codes. For product codes, the smallest block interleaver size, which is the square of the number of information bits in a codeword, is as good as any larger interleaver.

The main conclusion of this chapter is that turbo codes offer better performance and more ways to vary the coding gain as a function of the restrictions imposed by the communication system (complexity, delay, etc).

An iterative decoding technique for concatenated codes was presented in Chapter 7. This was specially designed to reuse existing hardware for decoding concatenated codes in real time. It is not an optimum method, but it is a cost effective way to increase the performance of existing decoders for this type of concatenation scheme.

There are many doors left open for further investigation. Turbo codes used in higher order modulation schemes (8PSK, 16QAM) have just started to be investigated. Promising results have been obtained when turbo codes are combined with trellis coded modulation schemes [43, 44].

Iterative techniques can also be applied in joint multi–user detector strategies in CDMA communication systems [77]. Turbo codes could be the ideal candidate when error control coding is needed. The soft outputs of the multi–user detector could be used as inputs to the turbo decoder. A joint multi–user detection/decoding scheme should be investigated.

Iterative decoding is a very successful method to approach channel capacity and its applications have just begun to be discovered.
APPENDIX

Turbo encoder Figures
Figure A.1 A four state rate half turbo encoder

Figure A.2 A four state rate third turbo encoder
Figure A.3 A sixteen state rate half turbo encoder

Figure A.4 A sixteen state rate third turbo encoder
REFERENCES


73. J. Hagenauer et al., “Improving the standard coding system for deep space missions”


