Diploma Thesis

Development of a LDPC channel coding system for space communications

Ανάπτυξη κωδίκων LDPC για συστήματα διαστημικής επικοινωνίας

By

Adam Eleni

Department of Electrical and Computer Engineering

University of Thessaly

Supervisor:
Dr. Antonios Argyriou

Volos, 2016

ABSTRACT

The purpose of this thesis is the development of a Low-Density Parity-Check (LDPC) channel coding system for space communications. Specifically, the implementation of the encoder and a belief-propagation decoder of the following LDPC codes: the C2 LDPC code that is a modification of a regular (4,32) code, which has a code rate of approximately 7/8 and a set of nine LDPC codes that belong to the Accumulate, Repeat-by-4, and Jagged Accumulate (AR4JA) family codes with selected code rates 1/2, 2/3 and 4/5. A simulation of the aforementioned codes has been conducted in a BPSK AWGN channel to evaluate their performance. These codes are to be used in a real space channel in order to face the phenomenon of Superior Solar Conjunction, where any signals from and to a spacecraft travel through the solar corona.

ΠΕΡΙΛΗΨΗ

Ο σκοπός αυτής της διπλωματικής εργασίας είναι η ανάπτυξη κωδίκων Low-Density Parity-Check (LDPC) για συστήματα διαστημικής επικοινωνίας. Συγκεκριμένα, η υλοποίηση του κωδικοποιητή και ενός belief-propagation αποκωδικοποιητή των παρακάτω LDPC κωδίκων: ο C2 LDPC κώδικας που είναι μια τροποποίηση του ενός (4,32) κώδικα, με ρυθμό κώδικα κατά προσέγγιση 7/8 και ένα σετ από εννιά LDPC κώδικες που ανήκουν στην οικογένεια των Accumulate, Repeat-by-4, and Jagged Accumulate (AR4JA) κωδίκων με επιλεγμένους ρυθμούς κώδικα 1/2, 2/3 και 4/5. Έχει υλοποιηθεί μια προσομοίωση των ανωτέρω κωδίκων σε BPSK AWGN κανάλι για να αξιολογηθεί η επίδοσή τους. Οι κώδικες θα χρησιμοποιηθούν σε ένα αληθινό διαστημικό κανάλι προκειμένου να αντιμετωπίσουν το φαινόμενο της ανώτερης ηλιακής συνόδου, όπου τα σήματα από και προς ένα διαστημόπλοιο περνάνε μέσα από το ηλιακό στέμμα.

ACKNOWLEDGEMENTS

With the completion of my thesis, I would like to express my sincere gratitude to my supervisor Dr. Antonios Argyriou for inspiring me with this project and guiding me with his valuable ideas through this interesting time of research.
I would also like to acknowledge Dr. Athanasios Korakis for his collaboration.
Moreover, I would like to thank my friends for their company and assistance even when difficult times came.
I am especially grateful to my family for giving me the opportunity to follow my dreams and encouraging me throughout all these years. This thesis is dedicated to you, because without your support none of this would have been possible.

To my family,

Diana, George, Anni & Kyriakos

CONTENTS

1. Introduction

Since the beginning of time, humanity has been interested in exploring and understanding the Universe. It is easily seen throughout history that people observed the physical phenomena, developed mathematical models and constructed powerful systems in order to discover the solar system and everything that lies beyond. Nowadays, the ability to travel in space has given many answers to those questions, as well as raised new ones, making it of even greater need to continue these missions.

During a mission in space, a spaceship remains in constant communication with its base on earth, so as to exchange information about its status and findings. However, there is a period called Superior Solar Conjunction, when the spaceship is on the opposite side of the sun from earth and the radio interference from the sun makes the communications impossible. As a result, for a number of days the spaceship has to be placed on a safe mode until the connection can be restored [1].

This thesis focuses on solving this particular problem with channel coding, so that the connection between the earth and the spaceship is maintained at all times. The proposed channel coding schemes for the reduction of BER (Bit Error Rate) in such conditions are the Low-Density Parity-Check (LDPC) codes [2]. The LDPC codes are error-correction binary block codes with large codeword length, designed to achieve a greater coding gain.

Two sets of LDPC codes are presented in this thesis, one for transmissions that take place near the earth and one for those in the deep space [3]. The first one is the C2 code, a quasi-cyclic high-rate LDPC code which is the modification of a regular (4, 32) code with size (8160, 7136) and a rate of approximately $7/8$. The second, is a set of nine LDPC codes of the AR4JA (Accumulate, Repeat-by-4 and Jagged Accumulate) family with selected block lengths 1024, 4096, 16384 and code rates of $1/2$, $2/3$ and $4/5$.

The aforementioned codes are employed in a system that can be represented in figure 1.1. The system consists of a transmitter that sends a message which is initially encoded into a codeword and then sent through the channel. The channel may affect the bits of the transmitted codeword with noise, but the receiver will use an error-correcting decoder to detect the erroneous bits and correct them to get the right message.

Figure 1.1: The system model

In the next sections the implementation of this system will be described in detail. In section 2, the encoding algorithm of the LDPC codes is presented, the belief propagation decoder can be found in section 3 and a simulation of these codes for the case of a BPSK AWGN channel in section 4, providing the corresponding experimental results. In section 5, future research on this subject is discussed.

2. The Encoder

An encoder is the device that takes as input the information bits and returns the corresponding codeword as the output.

A. The Matrices

A LDPC code is described by its parity check matrix H, i.e. the parity check equations, which is an important tool of the encoder. The parity check matrix consists of circulant submatrices, which are determined by their first-rows, calculating the rest of the rows as a one-bit right shift of the previous row. The resulting block-circulant parity check matrix is of low density, hence the naming of the codes. In more detail, the encoder transforms this matrix into its dense generator matrix G.

Consequently, the matrices of the aforementioned codes presented in [2] are: For the C2 code, the encoder uses the generator matrix of the systematic (8176, 7154) subcode illustrated in figure 2.2, which derives from the parity check matrix of the basic (8176, 7156) code illustrated in figure 2.1. The $A_{i,j}$s represent the parity check matrix circulants of size 511×511, whilst the $B_{i,j}$s represent the generator matrix circulants, both are specified by their first rows as described in table 2.1 and table 2.2 respectively.

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & A_{1,5} & A_{1,6} & A_{1,7} & A_{1,8} & A_{1,9} & A_{1,10} & A_{1,11} & A_{1,12} & A_{1,13} & A_{1,14} & A_{1,15} & A_{1,16} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} & A_{2,5} & A_{2,6} & A_{2,7} & A_{2,8} & A_{2,9} & A_{2,10} & A_{2,11} & A_{2,12} & A_{2,13} & A_{2,14} & A_{2,15} & A_{2,16} \end{bmatrix}$$

Figure 2.1: Base Parity Check Matrix of the Basic LDPC Code
Source: [2]

$$\begin{bmatrix} \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{1,1} & \mathrm{B}_{1,2} \\ 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{2,1} & \mathrm{B}_{2,2} \\ 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{3,1} & \mathrm{B}_{3,2} \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{4,1} & \mathrm{B}_{4,2} \\ 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{5,1} & \mathrm{B}_{5,2} \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{6,1} & \mathrm{B}_{6,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{7,1} & \mathrm{B}_{7,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{8,1} & \mathrm{B}_{8,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & \mathrm{B}_{9,1} & \mathrm{B}_{9,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & \mathrm{B}_{10,1} & \mathrm{B}_{10,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & \mathrm{B}_{11,1} & \mathrm{B}_{11,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & \mathrm{B}_{12,1} & \mathrm{B}_{12,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & \mathrm{B}_{13,1} & \mathrm{B}_{13,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & \mathrm{B}_{14,1} & \mathrm{B}_{14,2} \end{bmatrix}$$

Figure 2.2: Generator Matrix of the Systematic Subcode
Source: [2]

Table 2.1: Parity Check Matrix Circulants
Source: [2]

Circulant	1's position in 1st row of circulant
$A_{1,1}$	0, 176
$A_{1,2}$	12, 239
$A_{1,3}$	0, 352
$A_{1,4}$	24, 431
$A_{1,5}$	0, 392
$A_{1,6}$	151, 409
$A_{1,7}$	0, 351
$A_{1,8}$	9, 359
$A_{1,9}$	0, 307
$A_{1,10}$	53, 329
$A_{1,11}$	0, 207
$A_{1,12}$	18, 281
$A_{1,13}$	0, 399
$A_{1,14}$	202, 457
$A_{1,15}$	0, 247
$A_{1,16}$	36, 261
$A_{2,1}$	99, 471
$A_{2,2}$	130, 473
$A_{2,3}$	198, 435
$A_{2,4}$	260, 478
$A_{2,5}$	215, 420
$A_{2,6}$	282, 481
$A_{2,7}$	48, 396
$A_{2,8}$	193, 445
$A_{2,9}$	273, 430
$A_{2,10}$	302, 451
$A_{2,11}$	96, 379
$A_{2,12}$	191, 386
$A_{2,13}$	244, 467
$A_{2,14}$	364, 470
$A_{2,15}$	51, 382
$A_{2,16}$	192, 414

Table 2.2: Generator Matrix Circulants
Source: [2]

Circulant	1st row of circulant (Leftmost number: octal, Following numbers: hexadecimal)
$b_{1,1}$	55BF56CC55283DFEEFEA8C8CFF04E1EBD9067710988E25048D67525426939E206 8D2DC6FCD2F822BEB6BD96C8A76F4932AAE9BC53AD20A2A9C86BB461E43759C
$b_{1,2}$	6855AE08698A50AA3051768793DC238544AF3FE987391021AAF6383A6503409C3 CE971A80B3ECE12363EE809A01D91204F1811123EAB867D3E40E8C652585D28
$b_{2,1}$	62B21CF0AEE0649FA67B7D0EA6551C1CD194CA77501E0FCF8C85867B9CF679C18B CF7939E10F8550661848A4E0A9E9EDB7DAB9EDABA18C168C8E28AACDDEAB1E
$b_{2,2}$	64B71F486AD57125660C4512247B229F0017BA649C6C11148FB00B70808286F1A 9790748D296A593FA4FD2C6D7AAF7750F0C71B31AEE5B400C7F5D73AAF00710
$b_{3,1}$	681A8E51420BD8294ECE13E491D618083FFBBA830DB5FAF330209877D801F92B5 E07117C57E75F6F0D873B3E520F21EAFD78C1612C6228111A369D5790F5929A
$b_{3,2}$	04DF1DD77F1C20C1FB570D7DD7A1219EAECEA4B2877282651B0FFE713DF338A63 263BC0E324A87E2DC1AD64C9F10AAA585ED6905946EE167A73CF04AD2AF9218
$b_{4,1}$	35951FEE6F20C902296C9488003345E6C5526C5519230454C556B8A04FC0DC642 D682D94B4594B5197037DF15B5817B26F16D0A3302C09383412822F6D2B234E
$b_{4,2}$	7681CF7F278380E28F1262B22F40BF3405BFB92311A8A34D084C086464777431D BFDDD2E82A2E6742BAD6533B51B2BDEE0377E9F6E63DCA0B0F1DF97E73D5CD8
$b_{5,1}$	188157AE41830744BAE0ADA6295E08B79A44081E111F69BBE7831D07BEEBF7623 2E065F752D4F218D39B6C5BF20AE5B8FF172A7F1F680E6BF5AAC3C4343736C2
$b_{5,2}$	5D80A6007C175B5C0DD88A442440E2C29C6A136BBCE0D95A58A83B48CA0E7474E 9476C92E33D164BFF943A61CE1031DFF441B0B175209B498394F4794644392E
$b_{6,1}$	60CD1F1C282A1612657E8C7C1420332CA245C0756F78744C807966C3E13264388 78BD2CCC83388415A612705AB192B3512EEF0D95248F7B73E5B0F412BF76DB4
$b_{6,2}$	434B697B98C9F3E48502C8DBD891D0A0386996146DEBEF11D4B833033E05EDC28 F808F25E8F314135E6675B7608B66F7FF3392308242930025DDC4BB65CD7B6E
$b_{7,1}$	766855125CFDC804DAF8DBE3660E8686420230ED4E049DF11D82E357C54FE256E A01F5681D95544C7A1E32B7C30A8E6CF5D0869E754FFDE6AEFA6D7BE8F1B148
$b_{7,2}$	222975D325A487FE560A6D146311578D9C5501D28BC0A1FB48C9BDA173E869133 A3AA9506C42AE9F466E85611FC5F8F74E439638D66D2F00C682987A96D8887C
$b_{8,1}$	14B5F98E8D55FC8E9B4EE453C6963E052147A857AC1E08675D99A308E7269FAC5 600D7B155DE8CB1BAC786F45B46B523073692DE745FDF10724DDA38FD093B1C
$b_{8,2}$	1B71AFFB8117BCF8B5D002A99FEEA49503C0359B056963FE5271140E626F6F8FC E9F29B37047F9CA89EBCE760405C6277F329065DF21AB3B779AB3E8C8955400
$b_{9,1}$	0008B4E899E5F7E692BDCE69CE3FAD997183CFAEB2785D0C3D9CAE510316D4BD6 5A2A06CBA7F4E4C4A80839ACA81012343648EEA8DBBA2464A68E115AB3F4034
$b_{9,2}$	5B7FE6808A10EA42FEF0ED9B41920F82023085C106FBBC1F56B567A14257021BC 5FDA60CBA05B08FAD6DC3B0410295884C7CCDE0E56347D649DE6DDCEEB0C95E
$b_{10,1}$	5E9B2B33EF82D0E64AA2226D6A0ADCD179D5932EE1CF401B336449D0FF775754C A56650716E61A43F963D59865C7F017F53830514306649822CAA72C152F6EB2
$b_{10,2}$	2CD8140C8A37DE0D0261259F63AA2A420A8F81FECB661DBA5C62DF6C817B4A61D 2BC1F068A50DFD0EA8FE1BD387601062E2276A4987A19A70B460C54F215E184
$b_{11,1}$	06F1FF249192F2EAF063488E267EEE994E7760995C4FA6FFA0E4241825A7F5B65 C74FB16AC4C891BC008D33AD4FF97523EE5BD14126916E0502FF2F8E4A07FC2
$b_{11,2}$	65287840D00243278F41CE1156D1868F24E02F91D3A1886ACE906CE741662B40B 4EFDFB90F76C1ADD884D920AFA8B3427EEB84A759FA02E00635743F50B942F0
$b_{12,1}$	4109DA2A24E41B1F375645229981D4B7E88C36A12DAB64E91C764CC43CCEC188E C8C5855C8FF488BB91003602BEF43DBEC4A621048906A2CDC5DBD4103431DB8
$b_{12,2}$	2185E3BC7076BA51AAD6B199C8C60BCD70E8245B874927136E6D8DD527DF0693D C10A1C8E51B5BE93FF7538FA138B335738F4315361ABF8C73BF40593AE22BE4
$b_{13,1}$	228845775A262505B47288E065B23B4A6D78AFBDDB2356B392C692EF56A35AB4A A27767DE72F058C6484457C95A8CCDD0EF225ABA56B7657B7F0E947DC17F972
$b_{13,2}$	2630C6F79878E50CF5ABD353A6ED80BEACC7169179EA57435E44411BC7D566136 DFA983019F3443DE8E4C60940BC4E31DCEAD514D755AF95A622585D69572692
$b_{14,1}$	7273E8342918E097B1C1F5FEF32A150AEF5E11184782B5BD5A1D8071E94578B0A C722D7BF49E8C78D391294371FFBA7B88FABF8CC03A62B940CE60D669DFB7B6
$b_{14,2}$	087EA12042793307045B283D7305E93D8F74725034E77D25D3FF043ADC5F8B5B1 86DB70A968A816835EFB575952EAE7EA4E76DF0D5F097590E1A2A978025573E

For the nine AR4JA codes, the encoder uses the generator matrices that derive from the parity check matrices computed by the following equations. The first equation calculates the parity check matrix for code rate $1/2$, the second for $2/3$ and the third for $4/5$:

$$H_{1/2} = \begin{bmatrix} 0_M & 0_M & I_M & 0_M & I_M \oplus \Pi_1 \\ I_M & I_M & 0_M & I_M & \Pi_2 \oplus \Pi_3 \oplus \Pi_4 \\ I_M & \Pi_5 \oplus \Pi_6 & 0_M & \Pi_7 \oplus \Pi_8 & I_M \end{bmatrix}$$

$$H_{2/3} = \left[\begin{array}{ccc} 0_M & 0_M & \\ \Pi_9 \oplus \Pi_{10} \oplus \Pi_{11} & I_M & \left| H_{1/2} \right. \\ I_M & \Pi_{12} \oplus \Pi_{13} \oplus \Pi_{14} & \end{array}\right]$$

$$H_{4/5} = \left[\begin{array}{cccc} 0_M & 0_M & 0_M & 0_M & \\ \Pi_{21} \oplus \Pi_{22} \oplus \Pi_{23} & I_M & \Pi_{15} \oplus \Pi_{16} \oplus \Pi_{17} & I_M & \left| H_{2/3} \right. \\ I_M & \Pi_{24} \oplus \Pi_{25} \oplus \Pi_{26} & I_M & \Pi_{18} \oplus \Pi_{19} \oplus \Pi_{20} & \end{array}\right]$$

The I_M is the $M \times M$ identity matrix, the 0_M is the $M \times M$ zero matrix and Π_K is an $M \times M$ permutation matrix with entries of 1 in row i and column $\pi_K(i)$ as specified by the equation $\pi_k(i) = \frac{M}{4}\big((\theta_k + \lfloor 4i/M \rfloor) mod 4\big) + \big(\varphi_k(\lfloor 4i/M \rfloor, M) + i\big) mod \frac{M}{4}$, where $i\ \varepsilon\ \{0, M-1\}$ and functions θ_k, φ_k are defined in tables 2.3 and 2.4.
The submatrix size M which is determined by the length of the information block as well as each code's rate can be found in table 2.5.

Table 2.3: Description of $\boldsymbol{\theta_k}, \boldsymbol{\varphi_k(0,M)}, \boldsymbol{\varphi_k(1,M)}$
Source: [2]

		$\phi_k(0,M)$							$\phi_k(1,M)$						
k	θ_k	$M = 2^7 \dots 2^{13}$							$M = 2^7 \dots 2^{13}$						
1	3	1	59	16	160	108	226	1148	0	0	0	0	0	0	0
2	0	22	18	103	241	126	618	2032	27	32	53	182	375	767	1822
3	1	0	52	105	185	238	404	249	30	21	74	249	436	227	203
4	2	26	23	0	251	481	32	1807	28	36	45	65	350	247	882
5	2	0	11	50	209	96	912	485	7	30	47	70	260	284	1989
6	3	10	7	29	103	28	950	1044	1	29	0	141	84	370	957
7	0	5	22	115	90	59	534	717	8	44	59	237	318	482	1705
8	1	18	25	30	184	225	63	873	20	29	102	77	382	273	1083
9	0	3	27	92	248	323	971	364	26	39	25	55	169	886	1072
10	1	22	30	78	12	28	304	1926	24	14	3	12	213	634	354
11	2	3	43	70	111	386	409	1241	4	22	88	227	67	762	1942
12	0	8	14	66	66	305	708	1769	12	15	65	42	313	184	446
13	2	25	46	39	173	34	719	532	23	48	62	52	242	696	1456
14	3	25	62	84	42	510	176	768	15	55	68	243	188	413	1940
15	0	2	44	79	157	147	743	1138	15	39	91	179	1	854	1660
16	1	27	12	70	174	199	759	965	22	11	70	250	306	544	1661
17	2	7	38	29	104	347	674	141	31	1	115	247	397	864	587
18	0	7	47	32	144	391	958	1527	3	50	31	164	80	82	708
19	1	15	1	45	43	165	984	505	29	40	121	17	33	1009	1466
20	2	10	52	113	181	414	11	1312	21	62	45	31	7	437	433
21	0	4	61	86	250	97	413	1840	2	27	56	149	447	36	1345
22	1	19	10	1	202	158	925	709	5	38	54	105	336	562	867
23	2	7	55	42	68	86	687	1427	11	40	108	183	424	816	1551
24	1	9	7	118	177	168	752	989	26	15	14	153	134	452	2041
25	2	26	12	33	170	506	867	1925	9	11	30	177	152	290	1383
26	3	17	2	126	89	489	323	270	17	18	116	19	492	778	1790

Table 2.4: Description of $\theta_k, \varphi_k(2, M), \varphi_k(3, M)$
Source: [2]

k	θ_k	$\phi_k(2,M)$ $M = 2^7 \dots 2^{13}$							$\phi_k(3,M)$ $M = 2^7 \dots 2^{13}$						
1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	12	46	8	35	219	254	318	13	44	35	162	312	285	1189
3	1	30	45	119	167	16	790	494	19	51	97	7	503	554	458
4	2	18	27	89	214	263	642	1467	14	12	112	31	388	809	460
5	2	10	48	31	84	415	248	757	15	15	64	164	48	185	1039
6	3	16	37	122	206	403	899	1085	20	12	93	11	7	49	1000
7	0	13	41	1	122	184	328	1630	17	4	99	237	185	101	1265
8	1	9	13	69	67	279	518	64	4	7	94	125	328	82	1223
9	0	7	9	92	147	198	477	689	4	2	103	133	254	898	874
10	1	15	49	47	54	307	404	1300	11	30	91	99	202	627	1292
11	2	16	36	11	23	432	698	148	17	53	3	105	285	154	1491
12	0	18	10	31	93	240	160	777	20	23	6	17	11	65	631
13	2	4	11	19	20	454	497	1431	8	29	39	97	168	81	464
14	3	23	18	66	197	294	100	659	22	37	113	91	127	823	461
15	0	5	54	49	46	479	518	352	19	42	92	211	8	50	844
16	1	3	40	81	162	289	92	1177	15	48	119	128	437	413	392
17	2	29	27	96	101	373	464	836	5	4	74	82	475	462	922
18	0	11	35	38	76	104	592	1572	21	10	73	115	85	175	256
19	1	4	25	83	78	141	198	348	17	18	116	248	419	715	1986
20	2	8	46	42	253	270	856	1040	9	56	31	62	459	537	19
21	0	2	24	58	124	439	235	779	20	9	127	26	468	722	266
22	1	11	33	24	143	333	134	476	18	11	98	140	209	37	471
23	2	11	18	25	63	399	542	191	31	23	23	121	311	488	1166
24	1	3	37	92	41	14	545	1393	13	8	38	12	211	179	1300
25	2	15	35	38	214	277	777	1752	2	7	18	41	510	430	1033
26	3	13	21	120	70	412	483	1627	18	24	62	249	320	264	1606

Table 2.5: Values of Submatrix Size M for the nine AR4JA codes
Source: [2]

Information block length k	Submatrix size M		
	Rate $1/2$	**Rate $2/3$**	**Rate $4/5$**
1024	512	256	128
4096	2048	1024	512
16384	8192	4096	2048

B. The Encoding Algorithm

For the description of the algorithm, we define the following variables: u represents the message (the uncoded word) with k bits length, c is the resulting codeword with n bits length, H the parity check matrix and G the generator matrix of H as determined above.

To calculate the codeword, the encoder performs matrix multiplication between the message and the generator matrix:

$$c = uG$$

Based on this operation, a faster method can be used. For that reason, instead of using the whole generator matrix G, the right side of the matrix that contains the circulants, i.e. the last $n - k$ columns of G includes the information needed, this new matrix is called B.

These codes are systematic in their first symbols, i.e. the first k bits of the codeword are the message bits, while the remaining $n - k$ parity bits are computed in the following way:

The first message bit is multiplied by the first row of B and the result is placed in an accumulator. Then, the next message bit is multiplied by the second row of B and the result is added to the accumulator. This process continues until all message bits are used and afterwards, it is repeated for the remaining rows of B. The resulting codeword is a concatenation of the k message bits and the $n-k$ bits of the accumulator.

Regarding the C2 code and the set of AR4JA codes, the encoder is employed according to the following:

The input given to the C2 encoder is the 7136-bit message. This message is extended by a prefix of 18 zeros so as to form a 7154-bit vector, which can be multiplied by the generator matrix of the (8176, 7154) subcode when inserted in the encoder algorithm. The result of this multiplication will be an 8176-bit vector consisting of 18 prefixed zeros, 7136 message bits and 1022 parity bits. From these bits, the 18 zeros are discarded and 2 zeros are appended, producing a C2 codeword of 8160 bits seen in figure 2.3.

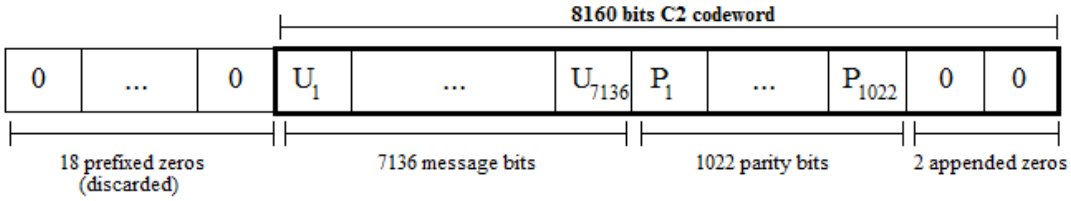

Figure 2.3: C2 Codeword

Considering the set of nine AR4JA codes, there are three cases of input: the 1024, 4096 and 16384-bit message. The message is inserted in the encoder algorithm without any modification and according to which code rate is chosen between $1/2$, $2/3$ and $4/5$, the suitable generator matrix is used for the multiplication. The last M bits of the product are punctured, (the value of M being specified in table 2.5) and the resulting codeword can be seen in figure 2.4. For each case the encoder returns a codeword of different length n, displayed in table 2.6.

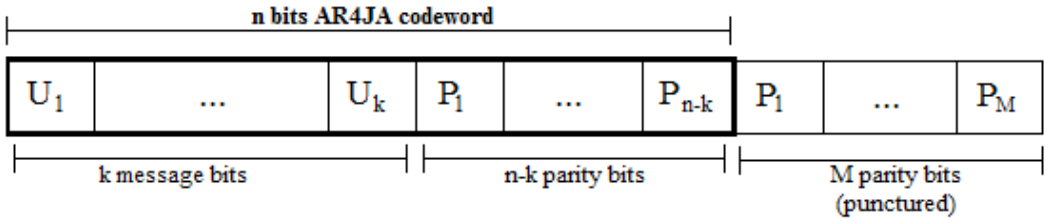

Figure 2.4: AR4JA Codeword

Table 2.6: Size (n, k) of each AR4JA code
Source: [2]

Information block length k	Codeword length n		
	Rate $1/2$	**Rate $2/3$**	**Rate $4/5$**
1024	2048	1536	1280
4096	8192	6144	5120
16384	32768	24576	20480

3. The Decoder

A decoder is the device that takes as input the received codeword and reconstructs it to its original form even if it has been corrupted by the noise of the channel.

A. Message-Passing Decoding

The algorithms used to decode LDPC codes are called message-passing decoding algorithms. The term derives from the fact that decoding takes place on a Tanner graph, where messages are iteratively exchanged between the nodes. A specific message-passing algorithm is belief propagation, also known as the sum-product algorithm. In this case, the messages are probabilities, often represented by log likelihood ratios, which show the level of belief a node has regarding the value of a codeword bit.

A Tanner graph is a bipartite graph consisting of two sets of nodes, the variable nodes that represent the codeword bits and the check nodes that represent the parity check equations, the structure can be seen in figure 3.1.

There are two types of messages passed along the Tanner graph edges:

The message passed from a variable node i to a check node j, which is the probability that i has a certain value, considering the observed value of that node and the values given to i in the previous iteration by all the corresponding check nodes connected to it except for j.

The message passed from a check node j to a variable node i, which is the probability that i has a certain value, considering all the messages given to j in the previous iteration by all the corresponding variable nodes except for i.

These messages are passed along the edges until the valid codeword is found or the maximum number of iterations is reached.

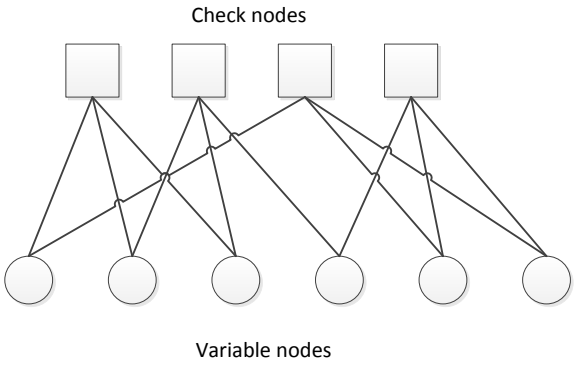

Figure 3.1: Tanner Graph

B. The Decoding Algorithm

The decoder introduced in this thesis uses the sum-product algorithm [4], an iterative soft-decision message-passing decoding algorithm, which takes as input the probability of each received codeword bit (a priori probabilities) and returns the maximum a posteriori probabilities in the form of log-likelihood ratios. Within each iteration, the algorithm checks whether all the parity check conditions are satisfied and if not, it continues until they are or until the maximum number of iterations is reached.

For the description of the algorithm consider the following:

The information exchanged about each codeword bit, is its probability of being zero or one. In order to address both of these states, the log-likelihood ratio (LLR) form is used:

$$L(c_i) = \log_e \frac{p(c_i = 0)}{p(c_i = 1)}$$

When $p(c_i = 0) > p(c_i = 1)$ then $L(c_i) > 0$ and the codeword bit value is considered zero $c_i = 0$.
When $p(c_i = 0) < p(c_i = 1)$ then $L(c_i) < 0$ and the codeword bit value is considered one $c_i = 1$.
As a result, the bigger the absolute value of $L(c_i)$, the more reliable the decision towards the codeword bit value is.

INPUT

- The variable r_i represents the a priori probabilities of each codeword bit i given by the channel as input to the decoder:

$$r_i = \log_e \frac{p(c_i = 0)}{p(c_i = 1)}$$

- H represents the parity check matrix as specified in section 2.A and it is used in order to determine whether the computed codeword y is a valid one, hence providing the termination condition to the algorithm, through the verification of:

$$Hy^T = 0$$

- I_{max} represents the maximum number of iterations that the decoder algorithm is granted, making it the second termination condition.

MESSAGES

- The update message V_{ji} from variable node i to check node j is the LLR of the probability that bit i is a one.

Initially: $V_{ji} = r_i$
Afterwards: $V_{ji} = \sum_{j' \neq j} U_{j'i} + r_i$

- The update message U_{ji} from check node j to variable node i is the LLR of the probability that parity check j is satisfied by bit i.

$$U_{ji} = \log_e \frac{\frac{1}{2} + \frac{1}{2}\prod_{i' \neq i} \tanh \frac{V_{ji'}}{2}}{\frac{1}{2} - \frac{1}{2}\prod_{i' \neq i} \tanh \frac{V_{ji'}}{2}}$$

$$U_{ji} = \log_e \frac{1 + \prod_{i' \neq i} \tanh \frac{V_{ji'}}{2}}{1 - \prod_{i' \neq i} \tanh \frac{V_{ji'}}{2}}$$

$$U_{ji} = 2 \tanh^{-1} \left(\prod_{i' \neq i} \tanh \frac{V_{ji'}}{2} \right)$$

OUTPUT

- The variable L_i represents the calculated probability of each codeword bit i as a result of the decoding process.

SUM-PRODUCT ALGORITHM

Sum_Product($H, LLR(y), Imax$){

// A priori probabilities

$r = LLR(y)$

// Construction of D

For $j = 1: Hrows$

$k = 1$

For $i = 1: Hcols$

If $(H(j,i) = 1)$

$D\{j\}(k) = i$ // Check j includes bit i

$k = k + 1$

End

End

End

// Construction of S

For $i = 1: Hcols$

$k = 1$

For $j = 1: Hrows$

If $(H(j,i) = 1)$

$S\{i\}(k) = j$ // Bit i is included in check j

$k = k + 1$

End

End

End

// Variable messages initialization

For $i = 1: Hcols$

For $j \; \varepsilon \; S\{i\}$

$V_{ji} = r_i$

End

End

// Iterations

$iter = 1$

$valid = 1$

While $(iter \leq Imax \;\&\; valid \neq 0)$ // Termination conditions

// Check messages

For $j = 1: Hrows$

For $i \; \varepsilon \; D\{j\}$

$$U_{ji} = 2\tanh^{-1}\left(\prod_{\substack{i' \varepsilon D\{j\} \\ i' \neq i}} \tanh\left(\frac{V_{ji'}}{2}\right)\right)$$

End

End

// Codeword validation

For $i = 1: Hcols$

$L_i = \sum_{j \; \varepsilon \; S\{i\}} U_{ji} + r_i$

If $(L_i \leq 0)$

$y_i = 1$

Else

$y_i = 0$

End

End

$valid = Hy^T$ // Check constructed codeword

// Variable messages

If $(valid \neq 0)$

For $i = 1: Hcols$

For $j \; \varepsilon \; S\{i\}$

$$V_{ji} = \sum_{\substack{j' \varepsilon S\{i\} \\ j' \neq j}} U_{j'i} + r_i$$

End

End

End

$iter = iter + 1$

End

Return (L)

}

A priori probabilities:
The a priori probability of each received codeword bit is placed in variable r.

Construction of D and S:
Extract information from the parity check matrix H about the connections between parity check equations and bits, in order to place it within D and S.
D discloses the information of which bits are included in a parity check equation. Whereas, S includes the reverse, meaning within which parity check equations a bit is included.

Variable messages initialization:
Each variable node i sends the message V_{ji} to each connected check node j. V_{ji} contains the probability of bit i, initialized as the a priori probability r_i.

Iterations:
Since the initialization process is done, the iterations begin until one of the two termination conditions is true. The variable $iter$ represents the counter of the iterations with the limit of I_{max}. Whereas, the variable $valid$ initialized with the value of one, once it turns to zero, denotes the construction of a valid codeword and the iterations stop.

Check messages:
Each check node j returns the message U_{ji} to each connected variable node i. U_{ji} contains the calculated a posteriori probability for variable node i, based on the connected variable nodes except for i.

Codeword validation:
In order to check whether a valid codeword has been constructed, each variable node i adds its a priori probability and the received check messages to calculate the current LLR of each codeword bit. These probabilities correspond to a bit value of zero when the LLR is positive and a bit value of one when the LLR is negative, thereby establishing the resulting codeword y.
A codeword is valid when the equation $Hy^T = 0$ is true. So, if the result of the multiplication is zero, the codeword is found, the algorithm is terminated and the L_i probabilities are returned. Otherwise, the codeword is not valid and if the maximum number of iterations (I_{max}) is not exceeded, a new iteration starts.

Variable messages:
If a valid codeword is not yet found, each variable node i sends the message V_{ji} to each connected check node j. V_{ji} takes its new value, which contains the calculated probability of bit i, based on the connected check nodes except for j and the a priori probability known for bit i.

Regarding the C2 code and the set of AR4JA codes, the decoder is employed according to the following:
The input given to the C2 decoder is the codeword of 8160 bits. This codeword is extended by a prefix of 18 zeros, which transformed in LLRs will take a very big positive value and the last two bits of the codeword, corresponding to the appended zeros are removed, producing a vector of 8176 bits as input to the decoder algorithm. From the resulting codeword (according to figure 2.3) the 18 prefixed zeros are removed along with the 1022 last parity bits, in order to present the 7136-bit decoded message.
Considering the set of nine AR4JA codes, there are nine cases of input as seen in table 2.6. In every case, M zero LLRs are appended to the n-bit codeword according to table 2.5 in order to represent the punctured bits. This codeword is inserted in the decoder algorithm. From the resulting codeword, the first $(Hcols - M) \cdot code\,rate$ bits represent the decoded message.

C. Other Decoding Algorithms

The implementation of these LDPC codes with other decoding algorithms such as the Min-Sum, the SVS Min-Sum [5], GSVS Min-Sum [6] and the Quantized-BP [7] algorithms, showed that even though these are of lower complexity, they still do not perform as good as the Belief-Propagation (Sum-Product) algorithm described above.

4. Performance

A simulation of the aforementioned codes has been conducted in Matlab (Matrix Laboratory) to evaluate their performance in the AWGN (Additive White Gaussian Noise) channel.
The chosen modulation scheme is BPSK (Binary Phase-Shift Keying) and the metric of comparison used is BER (Bit Error Rate) in each selected SNR (Signal to Noise Ratio), defining the number of error bits that occur compared to the total bits within a transmitted codeword. The decoder algorithm iterations (I_{max}) are set to the optimal value of 200 and the number of data blocks sent varies from 1000 to 10000 according to the simulation, specifically 10000 for the $^1/_2$ 4096, $^1/_2$ 1024, $^2/_3$ 4096, $^2/_3$ 1024, $^4/_5$ 16384, $^4/_5$ 4096, $^4/_5$ 1024 AR4JA codes and 1000 for the $^1/_2$ 16384, $^2/_3$ 16384 AR4JA codes and $^7/_8$ 7136 C2 code.

The performance curves can be seen in figure 4.1. The AR4JA codes with lower code rate 1/2 and 2/3 are suitable for low SNR environments between 0.4db and 2.4db, where they can decode the received message establishing an error rate of almost 10^{-5}. Whereas, the 4/5 AR4JA codes along with the 7/8 C2 code, perform better within the SNRs of 2.4db and 4db, introducing the same error rate on the order of 10^{-5}.

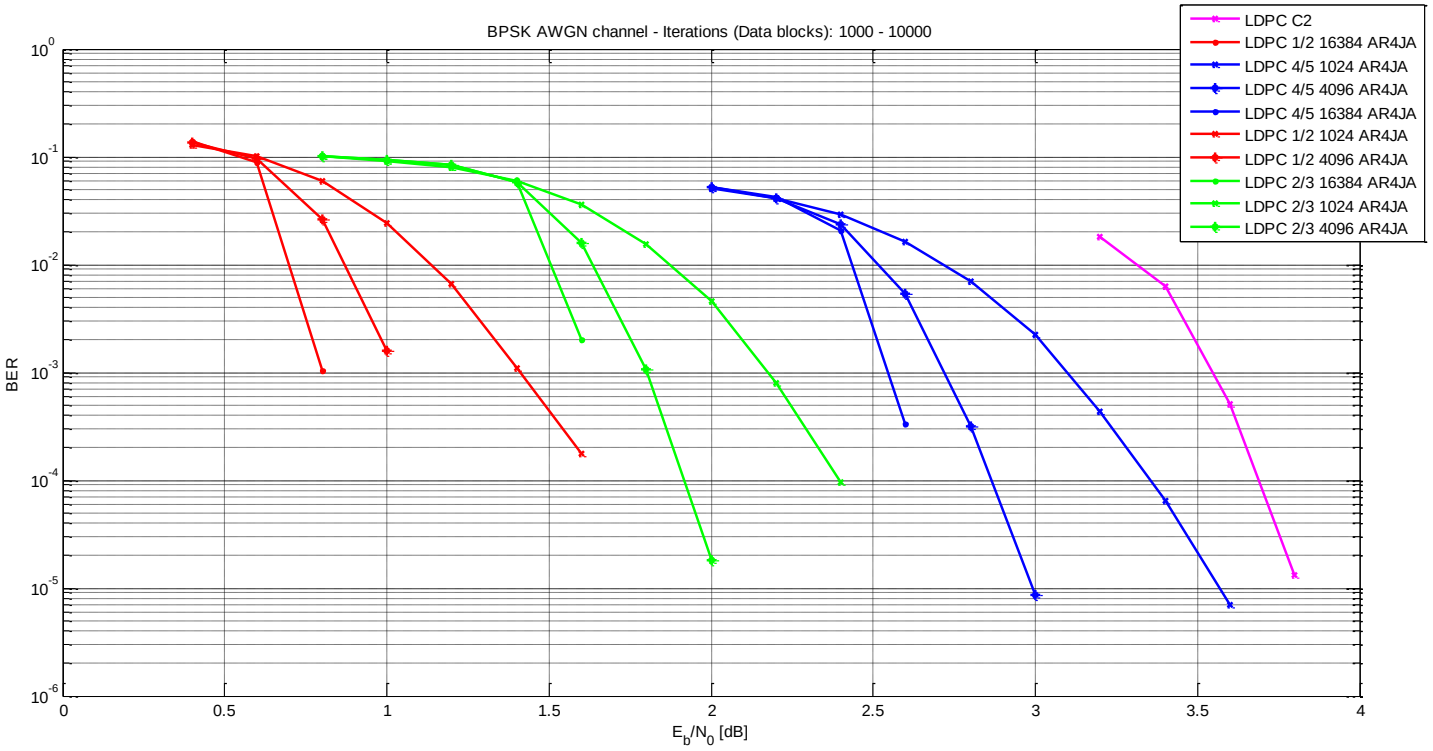

Figure 4.1: The BER for the set of AR4JA codes and the C2 code.

The results were compared to the ones presented in literature [3] and show that there is a distance lower than 0.1db between them, placing them within the acceptable limits and in most cases revealing an even better performance than those in theory suggest. The results can be seen in figure 4.2.

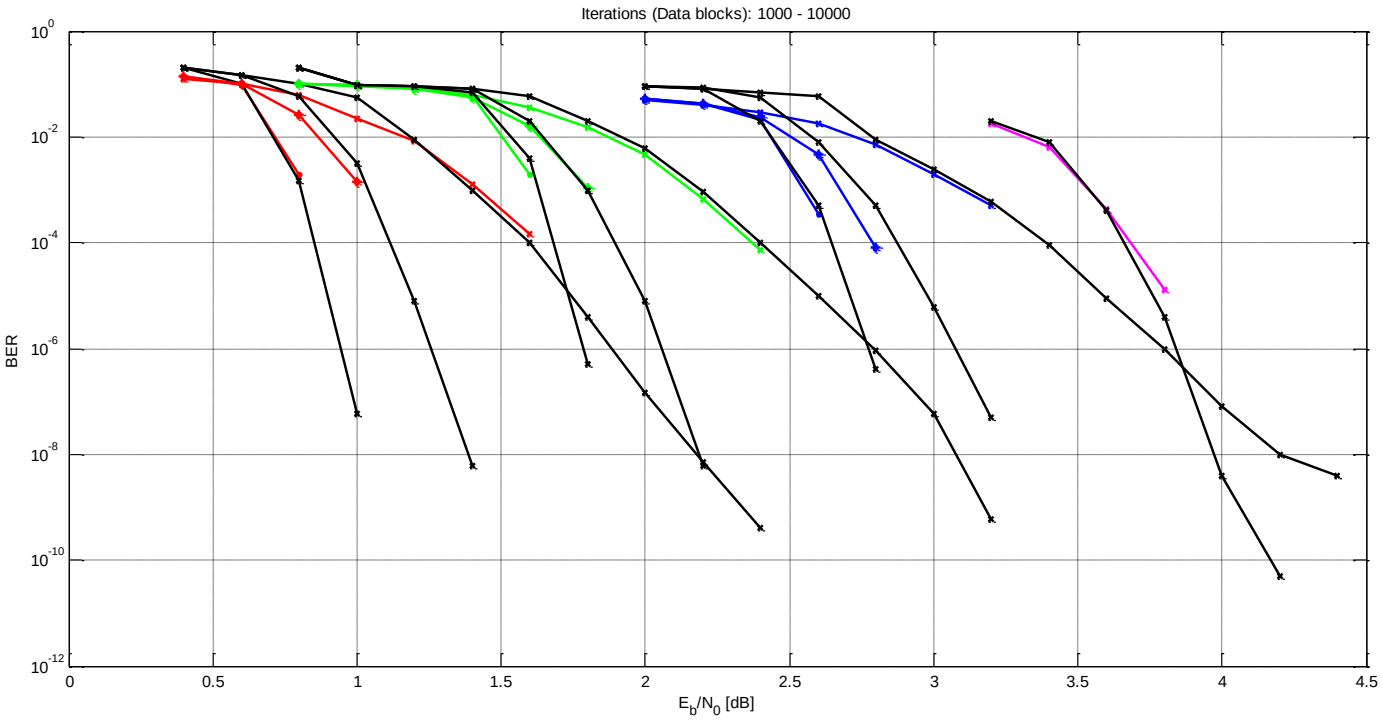

Figure 4.2: The BER compared to theory for the set of AR4JA codes and the C2 code.
The black curves represent the theoretical values. The colored curves from left to right: $^{1}/_{2}\,16384, ^{1}/_{2}\,4096, ^{1}/_{2}\,1024, ^{2}/_{3}\,16384, ^{2}/_{3}\,4096, ^{2}/_{3}\,1024, ^{4}/_{5}\,16384, ^{4}/_{5}\,4096, ^{4}/_{5}\,1024\ AR4JA\ codes,$ $^{7}/_{8}\,7136\ C2\ code.$

5. Conclusion

In this thesis the C2 LDPC code and a set of nine AR4JA LDPC codes have been proposed for space communications [3]. Specifically, the system containing the encoder and the belief-propagation decoder employed for these particular codes has been described in detail and the performance of the codes has been evaluated in Matlab.

The results of these simulations demonstrate that these codes can be effectively used in a deep space environment with a very low SNR.

As future research, the development of the encoder and decoder in a FPGA (Field-Programmable Gate Array) platform is recommended, with consideration to these LDPC codes.

6. References

[1]. *Planned Science Data During Reduced STEREO Science Operations* [Online] Available: http://stereo-ssc.nascom.nasa.gov/solar_conjunction_science.shtml

[2]. *TM Synchronization and Channel Coding*. Recommendation for Space Data System Standards, CCSDS 131.0-B-2. Blue Book. Issue 2. Washington D.C.: CCSDS, August 2011.

[3]. *TM Synchronization and Channel Coding–Summary of Concept and Rationale*. Report Concerning Space Data System Standards, CCSDS 130.1-G-2. Green Book. Issue 2. Washington, D.C.: CCSDS, November 2012.

[4]. S. J. Johnson, "Introducing Low-Density Parity-Check Codes" [Online] Available: http://sigpromu.org/sarah/SJohnsonLDPCintro.pdf

[5]. A. A. Emran and M. Elsabrouty, "Simplified variable-scaled min sum LDPC decoder for irregular LDPC codes," *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, 2014, pp. 518-523. doi: 10.1109/CCNC.2014.6940497

[6]. A. A. Emran and M. Elsabrouty, "Generalized simplified variable-scaled min sum LDPC decoder for irregular LDPC codes," *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Washington DC, 2014, pp. 892-896. doi: 10.1109/PIMRC.2014.7136292

[7]. J. K. S. Lee and J. Thorpe, "Memory-efficient decoding of LDPC codes," *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005., Adelaide, SA, 2005*, pp. 459-463. doi: 10.1109/ISIT.2005.1523376