4.8 Low-Density Parity-Check Codes

Low-density parity-check (LDPC) codes are a class of linear block codes with near capacity performance, which was invented by Gallager in the early 1960's in his Ph.D. thesis and were rediscovered in late 1990's. They are constructed using sparse bipartite graph (hence the name low-density), and can be decoded iteratively based on sum-product algorithm. LDPC codes have proved capable of approaching the Shannon limit more closely than any other class of codes.

Ever since their rediscovery, design, construction, decoding, efficient encoding, performance analysis, and applications of LDPC codes in digital communication and storage systems have become focal points of research.

Note: Some of material presented in this section is based on Prof. Shu Lin's lecture notes and S. J. Johnson's notes.

4.8.1 Definitions and Basic Concepts

Although LDPC codes can be generalized to nonbinary alphabets, we will consider only binary codes in this section for the sake of simplicity.

A binary LDPC code is a special class of (n, k) linear block codes whose parity-check matrix $\mathbf{H} = [h_{ij}]$ contains mostly 0's and relatively few 1's; i.e., **H** is sparse. This sparsity renders low complexity decoding and leads to simple implementations. In particular, an (n, d_v, d_c) *regular LDPC* code is a binary linear block code of length n, whose parity-check matrix **H** contains exactly d_v 1's in each column and d_c 1's in each row. This means that every coded bit participates in exactly d_v parity-check equations and that every such check equation involves exactly d_c coded bits. A formal definition is as follows.

<u>*Definition 4.8.1:*</u> A binary regular LDPC code is defined as the null space of a sparse parity-check matrix **H** over GF(2) with the following structural properties:

1) Each row has constant weight d_c ;
2) Each column has constant weight d_v ;
3) No two rows (or two columns) have more than one 1-component in common; and
4) Both d_c and d_v are small compared to code length and the number of rows in **H**.

The **H** with these properties will be said to be (d_v, d_c)-regular, and the code generated by such **H** will be called a (d_v, d_c)-regular LDPC code.

Property 3) is referred to as the *row and column (RC) constraint*. The RC-constraint on the rows and columns of **H** ensures that no d_v or fewer columns of **H** can be added to zero. As a result, the minimum distance of the LDPC code given by the null space of **H** is at least d_v+1.

Because both d_c and d_v are small compared to code length and the number of rows in the matrix, **H** therefore has a low density of 1's. In literature, the density r of the parity-check

matrix **H** is defined as the ratio of the total number of 1's in **H** to the total number of entries in **H**. Let m be the number of rows in **H** (i.e., suppose that **H** is an $m \times n$ matrix). We readily see that

$$r = \frac{d_c}{n} = \frac{d_v}{m}$$

Since d_v and d_c are related by $d_c m = d_v n$, the nominal code rate R_c can be computed by

$$R_c = \frac{n-m}{n} = 1 - \frac{d_v}{d_c} \tag{4.1}$$

The actual rate of the code specified by **H** may be higher since these m rows of **H** (m check equations) might not all be independent.

<u>*Definition 4.8.2:*</u> If **H** is sparse, but the number of 1's in each column or in each row is not constant, the resulting code is called an *irregular LDPC code*.

In general, irregular LDPC codes have a better performance than regular LDPC codes.

Recall that a square matrix is called a circulant matrix (simply a *circulant*) if each row is a cyclic-shift (one place to the right or the left) of the row above it and the first row is the cyclic-shift of the last row.

<u>*Definition 4.8.3:*</u> An LDPC code is said to be *cyclic* if its parity-check matrix consists of a column of circulants. An LDPC code is said to be *quasi-cyclic* (QC) if its parity-check matrix consists of an array of circulants.

The encoding of a cyclic or a QC-LDPC code can be implemented with simple shift-register(s).

■ <u>Example 4.8.1:</u> Consider the matrix given below:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

This matrix has both column and row weights 3. We can easily check that there are no two rows (or two columns) have more than one 1-component in common. Therefore the matrix satisfies the RC-constraint. It is a (3, 3)-regular matrix. It is also seen that **H** is a circulant. The null space of **H** gives a (7, 3) cyclic-LDPC code with minimum distance $d_{\min} = 4$.

4.8.1.2 Graphical Representation of LDPC codes

We have seen in Section 4.6 that a linear block code can also be represented by a Tanner graph (or factor graph, normal graph), which actually provides a graphical representation of the parity-check matrix **H**. The n variable nodes represent the columns of **H**, and the m constraint (check) nodes represent the rows of **H**. There exists an edge between the check node i and the variable node j if and only if the entry $h_{ij} = 1$. Thus, for an (n, d_v, d_c) regular LDPC code, there are $nd_v = md_c$ edges in the graph, d_v edges incident to each variable node and d_c edges incident to each check node. As an example, Fig. 4.8.1 shows the Tanner graph for a (10, 5) linear block code with $d_v = 3$, $d_c = 6$, and the parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{4.3}$$

Let **x** denote a codeword. From $\mathbf{x}\mathbf{H}^T = \mathbf{0}$, we can see that for each check node, the sum of all adjacent variable nodes is equal to zero. That is, check nodes are zero-sum constraints.

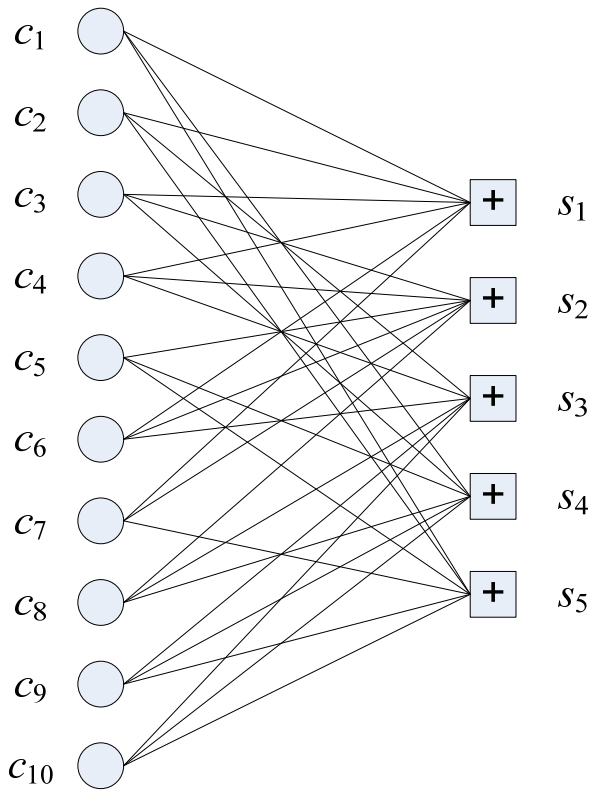


Figure 4.8.1 Tanner graph for a (10, 5) linear block code. Note that each variable node is of degree 3 and each check node is of degree 6.

The parity-check matrix **H** of a code is actually the incidence matrix of its Tanner graph.

Therefore, we also refer to the Tanner graph of a code as the Tanner graph of its parity-check matrix **H**.

Definition 4.8.4 [Cycle and Girth]: A cycle of length l in a Tanner graph is a path comprised of l edges which closes back on itself. The minimum cycle length of the graph is called the girth of the graph.
Clearly, the shortest possible cycle in a bipartite graph is a length-4 cycle, which manifest themselves in the **H** matrix as four 1's that lie an the corners of a submatrix of **H**.

$$\mathbf{H} = \begin{bmatrix} & \cdots & \\ 1 & & 1 \\ & \cdots & \\ 1 & & 1 \\ & \cdots & \end{bmatrix}$$

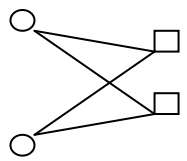


Figure 4.8.2

As stated in Section 4.7, the cycles, particularly short cycles will degrade the performance of the iterative decoding algorithm used for LDPC codes. The cycles that affect the decoding performance the most are the cycles of length 4. Therefore, in constructing LDPC codes, cycles of length 4 must be avoided.

If the parity-check matrix **H** of a binary linear block code satisfies the RC-constraint, then no two coded bits can be checked simultaneously by two parity-check sums. As a result, the Tanner graph of the code is free of cycles of length 4 and has a girth of at least 6. Since we require that the matrix **H** of an LDPC code, regular or irregular, satisfies the RC-constraint, the Tanner graph of an LDPC code has a girth of at least 6.

■ Example 4.8.2: The Tanner graph of the (7, 3, 3) LDPC code given in Example 1 is shown in Fig. 4.8.3.

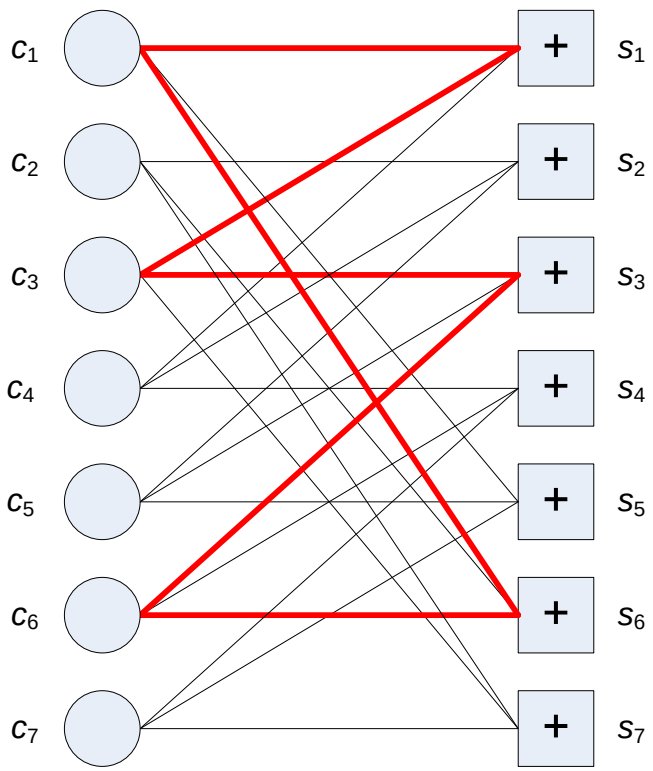


Figure 4.8.3 Tanner graph of a (7, 3) linear block code

LDPC codes can also be represented by normal graphs, see Fig. 4.8.3-2. Figure 4.8.4 depicts a generic normal graph for an LDPC code.

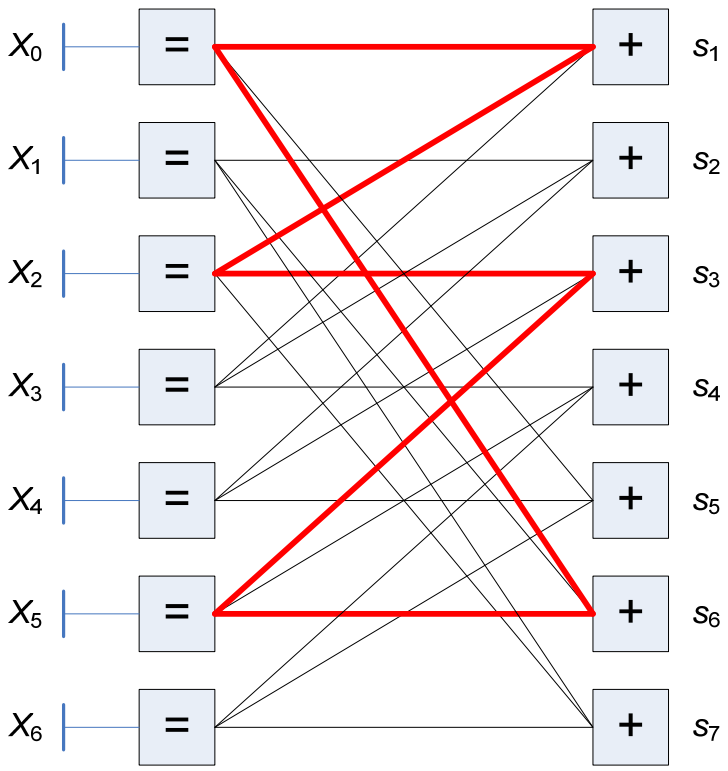


Figure 4.8.3-2

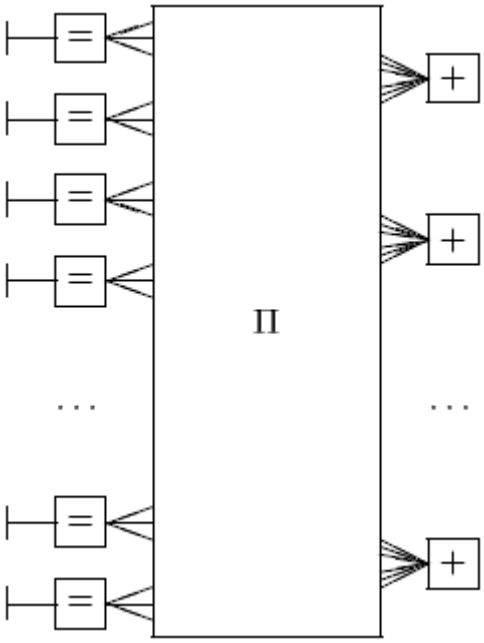


Figure 4.8.4 Normal graph of a regular LDPC code with $d_v = 3$ and $d_c = 6$.

4.8.2 Encoding of LDPC Codes

如前所述，LDPC码是一种线性分组码，其编码可以按照一般线性分组码的编码原理进行，即首先通过Gauss-Jordan消去法将给定的校验矩阵**H**变换为如下形式：

$$\mathbf{H} = \begin{bmatrix} \mathbf{P} & \mathbf{I}_{n-k} \end{bmatrix}$$

where **P** is a $(n-k)\times k$ binary matrix and $\mathbf{I}_{n-k}$ is the identity matrix of size n-k. Then the generator matrix in systematic form is obtained

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_k & \mathbf{P}^T \end{bmatrix}$$

Thus, the encoding of the code defined by **H** can be performed via

$$\mathbf{c} = \mathbf{uG} = \begin{bmatrix} \mathbf{u} & \mathbf{uP}^T \end{bmatrix} \tag{4.4}$$

<u>Example 4.8.3:</u> Consider the encoding of the length-10 rate-1/2 LDPC code [Johnson, pp.15]

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

<u>First, we put **H** into *row-echelon form*</u> (i.e. so that in any two successive rows that do not consist entirely of zeros, the leading 1 in the lower row occurs further to the right than the leading 1 in the higher row).

The matrix **H** is put into this form by applying *elementary row operations* in GF(2),

which are: interchanging two rows or adding one row to another modulo 2. From linear algebra we know that by using only elementary row operations the modified parity-check matrix will have the same codeword set as the original.

The 1-st and 2-nd columns of **H** already have ones on the diagonal and entries in these columns below the diagonal are removed by replacing the 4-th row with the modulo-2 sum of the 1-st and 4-th rows. The 3-rd column of H does not have a one on the diagonal but this can be obtained by swapping the 3-rd and 5-th rows. Finally, replacing the 5-th row with the modulo-2 sum of the 5-th and 4-th rows gives $\mathbf{H}_r$ in row-echelon form:

$$\mathbf{H}_r = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

<u>Next the parity-check matrix is put into *reduced* row-echelon form</u> (i.e. so that any column that contains a leading one has zeros everywhere else). The 1-st column is already correct and the entry in the 2-nd column above the diagonal is removed by replacing the 1-st row with the modulo-2 sum of the 1-st and 2-nd rows. Similarly the entry in the 3-nd column above the diagonal is removed by replacing the 2-nd row with the modulo-2 sum of the 2-nd and 3-rd rows. To clear the 4-th column the 1-st row is replace with the modulo-2 sum of the 1-st and 4-th rows. Finally, to clear the 5-th column involves adding the 5-th row to the 1-st, 2-nd and 4-th rows gives $\mathbf{H}_r$ in reduced row-echelon form:

$$\mathbf{H}_r = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

<u>Lastly, using column permutations we put the parity-check matrix into standard form</u>:

$$\mathbf{H}_{\text{std}} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

from which a generator **G** for the code (with parity-check matrices $\mathbf{H}_{\text{std}}$) can be obtained by the relation stated above.

In this final step column permutations have been used and so the codewords of $\mathbf{H}_{\text{std}}$ will be permuted versions of the codewords corresponding to **H**. A solution is to use a permutation vector to keep track of the column permutation used to create $\mathbf{H}_{\text{std}}$, which in this case is

$$\pi = \begin{bmatrix} 6 & 7 & 8 & 9 & 10 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

and apply the inverse permutation to each $\mathbf{H}_{\text{std}}$ codeword before it is transmitted. 这样，在接收端我们可仍然使用**H**进行译码。

Alternatively, if the channel is memoryless, and so the order of codeword bits is unimportant, a far easier option is to apply π to the original **H** to give a parity-check matrix

$$\mathbf{H}' = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

with the same properties as **H** but which shares the same codeword bit ordering as $\mathbf{H}_{\text{std}}$. 在接收端，我们使用**H'**对由**G**编出的码字进行译码。

All of this processing can be done off-line and just the matrices **G** and **H**′ provided to the encoder and decoder respectively.

The drawback of this approach is that, unlike **H**, the matrix **G** will most likely not be sparse and so the matrix multiplication in (4.4) have complexity in the order of n^2 operations. 随着码长n的增加，编码复杂度会变得非常高。Richarson and Urbanke 于2001年提出了一种直接基于**H**矩阵的编码方法[Richarson2001, IT]，它具有近似线性编码复杂度。另一种降低编码复杂度的方法是基于代数或几何方法构造LDPC码，这种结构化的校验矩阵通常能够允许编码器采用简单的移位寄存器实现。下面我们首先介绍R-U的方法，具有准循环**H**矩阵结构的QC-LDPC码编码方法见[19, Li-Lin2006]。

4.8.2.1 (Almost) Linear-Time Encoding of LDPC Codes

Rather than finding a generator matrix for **H**, an LDPC code can be encoded using the parity-check matrix directly by performing some preprocessing prior to encoding (transforming it into upper triangular form and using back substitution).

Firstly, we perform the following preprocessing steps. By row and column permutations, we bring **H** into the approximate lower triangular form as indicated in Fig. 4.8.5, where the upper right comer can be identified as a lower triangular matrix. Because it is obtained only by permutations, the **H** matrix is still sparse. We denote the permutation decomposition as

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \mathbf{C} & \mathbf{D} & \mathbf{E} \end{bmatrix}$$

where **T** is a $(m-g)\times(m-g)$ lower triangular matrix with ones along the diagonal and hence is invertible. If **H** is full rank, the matrix **B** is of size $(m-g)\times g$ and **A** is of size $(m-g)\times k$. The g rows of **H** left in **C**, **D**, and **E** are called the *gap* of this approximate representation. The smaller g the lower the encoding complexity for the LDPC code.

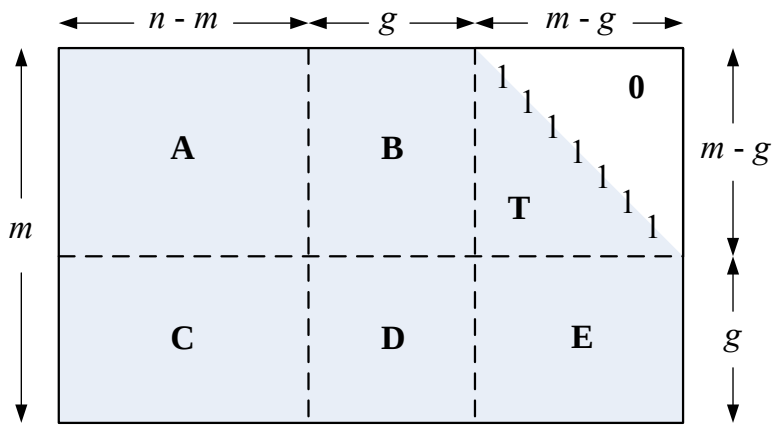


Figure 4.8.5

Then Gauss-Jordan elimination is applied to clear **E**, which is equivalent to multiplying **H** on the left by the matrix

$$\begin{bmatrix} \mathbf{I}_{M-g} & \mathbf{0} \\ -\mathbf{E}\mathbf{T}^{-1} & \mathbf{I}_g \end{bmatrix}$$

This gives

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{I}_{M-g} & \mathbf{0} \\ -\mathbf{E}\mathbf{T}^{-1} & \mathbf{I}_g \end{bmatrix} \mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ -\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C} & -\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D} & \mathbf{0} \end{bmatrix}$$

Note that $\tilde{\mathbf{H}}$ is the parity check matrix for an equivalent code.

To encode a message vector **u** of length k using $\tilde{\mathbf{H}}$, we write the codeword **c** as

$$\mathbf{c} = \begin{bmatrix} \mathbf{u} & \mathbf{p}_1 & \mathbf{p}_2 \end{bmatrix}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ contain the first g parity bits and the remaining parity bits, respectively. The parity-check equation $\mathbf{c}\tilde{\mathbf{H}}^T = \mathbf{0}$ gives rise to two equations,

$$\mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{p}_1 + \mathbf{T}\mathbf{p}_2 = 0 \tag{4.6}$$

and

$$\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C}\right)\mathbf{u} + \left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}\right)\mathbf{p}_1 = \mathbf{0} \tag{4.7}$$

Let $\tilde{\mathbf{C}} = -\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C}$ and $\tilde{\mathbf{D}} = -\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}$. If $\tilde{\mathbf{D}}$ is nonsingular (invertible), we have from (4.7)

$$\mathbf{p}_1 = -\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{C}}\mathbf{u} \tag{4.8}$$

If $\tilde{\mathbf{D}}$ is not invertible, the columns of $\tilde{\mathbf{H}}$ can be permuted to obtain a non-singular $\tilde{\mathbf{D}}$. The matrix $-\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{C}}$ can be pre-computed and saved, so that $\mathbf{p}_1$ can be computed with a complexity of $O\left(g(n-m)\right)$.

Once $\mathbf{p}_1$ is known, $\mathbf{p}_2$ can be found from (4.6) by

$$\mathbf{p}_2 = -\mathbf{T}^{-1}(\mathbf{Au} + \mathbf{Bp}_1) \tag{4.9}$$

Since $\mathbf{T}^{-1}$ is lower triangular, $\mathbf{p}_2$ can be found by using back substitutuin.

Note that column permutations were used to obtain approximate lower triangular **H** from the original parity-check matrix **H**', so either **H**, or 实施相同列置换之后的 **H**'（**H**' with the same column permutation applied）, will be used at the decoder.

The process of computing $\mathbf{p}_1$ and $\mathbf{p}_2$ constitutes the encoding process. The steps for the computation as well as their computational complexity are outlined here (assuming that the preprocessing steps have already been accomplished). For the sake of clarity, intermediate variables are used to show the steps which may not be necessary in a final implementation.

- Steps to compute $\mathbf{p}_1 = -\tilde{\mathbf{D}}^{-1}\left(-\mathbf{ET}^{-1}\mathbf{A} + \mathbf{C}\right)\mathbf{u}$:

Operation	Comment	Complexity
$\mathbf{x}_1 = \mathbf{Au}$	Multiplication by a sparse matrix	$O(n)$
$\mathbf{x}_2 = \mathbf{T}^{-1}\mathbf{x}_1$	Solve $\mathbf{Tx}_2 = \mathbf{x}_1$ by backsubstitution	$O(n)$
$\mathbf{x}_3 = -\mathbf{Ex}_2$	Multiplication by a sparse matrix	$O(n)$
$\mathbf{x}_4 = \mathbf{Cu}$	Multiplication by a sparse matrix	$O(n)$
$\mathbf{x}_5 = \mathbf{x}_3 + \mathbf{x}_4$	Addition	$O(n)$
$\mathbf{p}_1 = -\tilde{\mathbf{D}}^{-1}\mathbf{x}_5$	Multiplication by dense $g \times g$ matrix	$O(g^2)$

- Steps to compute $\mathbf{p}_2 = -\mathbf{T}^{-1}(\mathbf{Au} + \mathbf{Bp}_1)$:

Operation	Comment	Complexity
$\mathbf{x}_1 = \mathbf{Au}$	Multiplication by a sparse matrix (already done)	0
$\mathbf{x}_6 = \mathbf{Bp}_1$	Multiplication by a sparse matrix	$O(n)$
$\mathbf{x}_7 = \mathbf{x}_1 + \mathbf{x}_6$	Addition	$O(n)$
$\mathbf{p}_2 = -\mathbf{T}^{-1}\mathbf{x}_7$	Solve $\mathbf{Tp}_2 = \mathbf{x}_7$ by backsubstitution	$O(n)$

The overall algorithm is $O(n + g^2)$. Clearly, the smaller g can be made, the lower the

complexity of the algorithm.

<u>Example 4.8.4:</u> [Johnson]

4.8.3 Decoding of LDPC codes

In his Ph.D. thesis of 1961, Gallager also presented two iterative decoding algorithms for LDPC codes. One is for hard-decision decoding, and another for soft-decision decoding. Since that time, other researchers have independently discovered that algorithm and related algorithms, albeit sometimes for different applications. Now, there are various methods for decoding LDPC codes, ranging from low to high decoding complexities and from reasonably good to very good error performance. These methods include:

- One-step majority logic (OSML) decoding;
- Bit-flipping (BF) decoding;
- Weighted BF decoding;
- Iterative decoding based on message passing algorithm (IDMPA). (The term "message passing algorithm" usually refers to the sum-product algorithm, the belief propagation algorithm and their approximations.)

Among these methods, OSML and BF decodings are hard-decision decodings. Weighted BF decoding is a reliability-based decoding method. IDMPA is a soft-in soft-out (SISO) decoding method, which is actually the sum-product algorithm working on the code's Tanner graph, as discussed in Section 4.7. The OSML decoding is the simplest in terms of decoding complexity. IDMPA provides the best performance but requires higher computational complexity. In the following, we will first outline two hard-decision decoding algorithms.

4.8.3.1 OSML Decoding of regular LDPC codes

Consider a (d_v, d_c)-regular LDPC code $\mathcal{C}$ of length n given by the null space of a $q \times n$ parity-check matrix $\mathbf{H}$ with column and row weights d_v and d_c, respectively. Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_q$ be the rows of $\mathbf{H}$. For each code bit position l, due to the RC-constraint, there are a set

$$\mathbf{A}_l = \left\{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \ldots, \mathbf{h}_{d_v}^{(l)}\right\}$$

of d_v rows in $\mathbf{H}$ such that: (1) each row in $\mathbf{A}_l$ has a 1-component at position l; and (2) no two rows in $\mathbf{A}_l$ have a common 1-component at any other position. See the figure below.

$$\begin{array}{c} \text{the } l\text{th column} \\ \downarrow \\ \mathbf{A}_l = \begin{bmatrix} \cdots & 0 & 0 & 1 & 1 & 0 & \cdots \\ & & 0 & 0 & 1 & 1 & \\ & & \vdots & \vdots & \vdots & \vdots & \\ \cdots & & 1 & 0 & 1 & 0 & \cdots \end{bmatrix} \end{array}$$

Clearly, the rows in $\mathbf{A}_l$ are orthogonal on the bit position l.

Suppose a codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ in $\mathcal{C}$ is transmitted and

$$\mathbf{z} = (z_0, z_1, ..., z_{n-1})$$

is the hard-decision received vector. Then **z** = **c** + **e**, where

$$\mathbf{e} = (e_0, e_1, ..., e_{n-1})$$

is the error pattern in **z**. The syndrome of **z** is given by

$$\mathbf{s} = (s_1, s_2, ..., s_q) = \mathbf{z}\mathbf{H}^T = (\mathbf{c}+\mathbf{e})\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

where

$$s_i = \mathbf{e}\cdot\mathbf{h}_i = e_0 h_{i,0} + e_1 h_{i,1} + \cdots + e_{n-1} h_{i,n-1}, \text{ for } 1 \le i \le q \tag{4.10}$$

Eq. (4.10) gives a set of q equations that relate the error digits to the q computed syndrome bits.

Remark: $h_{i,l} = 1$ indicates that the code bit c_l is checked by s_i .

Consider the following set of d_v syndrome bits computed based on the rows of $\mathbf{A}_l$:

$$\mathcal{S}_l = \left\{ s_i^{(l)} = \mathbf{e}\cdot\mathbf{h}_i^{(l)} \mid \mathbf{h}_i^{(l)} \in \mathbf{A}_l \right\} \tag{4.11}$$

where

$$s_i^{(l)} = \mathbf{e}\cdot\mathbf{h}_i^{(l)} = e_0 h_{i,0}^{(l)} + e_1 h_{i,1}^{(l)} + \cdots + e_{n-1} h_{i,n-1}^{(l)}, \text{ for } 1 \le i \le d_v \tag{4.12}$$

It follows from the structural properties of $\mathbf{A}_l$ that every syndrome in $\mathcal{S}_l$ checks (or contains) the error bit e_l and no other error bit in **e** is checked by more than one syndrome in $\mathcal{S}_l$. Thus, the syndromes in $\mathcal{S}_l$ are said to be orthogonal on the error bit e_l. From the discussion in Chapter ? on majority logic decoding of linear block codes, e_l can be corrected by using one-step majority logic decoding method. Furthermore, it can be show that if the number of errors in **e** is $\lfloor d_v / 2 \rfloor$ or fewer, the errors in **e** can be correctly determined.

The decoding algorithm, called one-step majority-logic (OSML) decoding, is stated as follows. The error digit e_l is decoded as 1 if a clear majority of the syndromes orthogonal on e_l is 1; otherwise, e_l is decoded as 0. Mathematically, it can be formulated below.

OSML decoding: For $0 \le l < n$,

- Decode $e_l = 1$ if $\sum_{s_j^{(l)} \in \mathcal{S}_l} \left(2s_j^{(l)} - 1\right) > 1$;
- Decode $e_l = 0$ if $\sum_{s_j^{(l)} \in \mathcal{S}_l} \left(2s_j^{(l)} - 1\right) \le 0$

The sum $\sum_{s_j^{(l)} \in \mathcal{S}_l} \left(2s_j^{(l)} - 1\right)$ is called the OSML decision function. Fig. 4.8.6 is an illustration of OSML decoding.

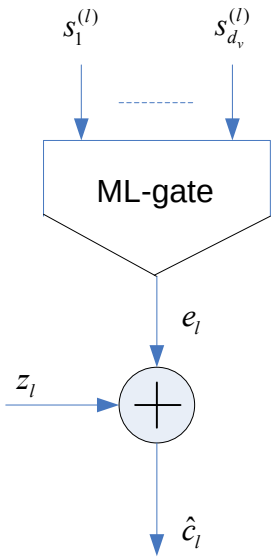


Figure 4.8.6

OSML decoding can be implemented in parallel such that all the n error bits are corrected simultaneously.

With the above OSML decoding of a (d_v, d_c)-regular LDPC code, correcting decoding is guaranteed if there are $\lfloor d_v/2 \rfloor$ or fewer errors in the received vector z. Since the code can correct $\lfloor d_v/2 \rfloor$ or fewer errors, it must have a minimum distance $d_{\min} \geq 2\lfloor d_v/2 \rfloor + 1 = d_v + 1$.

Clearly, OSML decoding of a (d_v, d_c)-regular LDPC code is effective only if d_v is reasonably large; i.e., the column weight of the parity-check matrix of the code is reasonably large.

4.8.3.2 BF Decoding

BF decoding was devised by Gallager in 1962. It is a very simple iterative hard-decision decoding scheme designed for LDPC codes.

In BF decoding of an LDPC code $\mathcal{C}$, the decoder first computes the syndrome $\mathbf{s} = (s_1, s_2, ..., s_q)$ of the hard-decision vector **z** based on (4.10). If **s** = **0**, then **z** is a codeword and decoding stops. If **s** ≠ **0**, then there are parity failures. The number of parity failures is equal to the number of syndrome bits in **s** that are not equal to 0. Let f_l be the number of check-sums in $\mathcal{S}_l = \left\{ s_i^{(l)} = \mathbf{e} \cdot \mathbf{h}_i^{(l)} \mid \mathbf{h}_i^{(l)} \in \mathbf{A}_l \right\}$ that are equal to 1 (parity-failure). If f_l is greater than some fixed number δ of failed parity-check sums, then flip the bit z_l in **z**. Using the modified received vector z', the decoder recomputes q parity-check sums based on (4.10), checks parity failures and performs bit flippings. The above process continues until all the

parity-check sums (syndrome bits) are equal to 0, i.e., no parity failure.

The parameter δ is called the *flipping threshold*, and should be chosen to optimize the error performance while minimizes the number of decoding iterations.

The simplest version of BF decoding is the one without threshold δ, which is described as follows.

1) Compute the q parity-check sums based on (4.10). If all the parity-check sums are zero, stop the decoding.
2) Find the number of failed parity-check sums for each received bit, denoted by f_i, $0 \le i < n$.
3) Identify the set $\mathcal{F}$ of received bits for which f_i is the largest.
4) Flip the bits in $\mathcal{F}$.
5) Repeat steps 1) to 4) until all the parity-check sums are zero or a preset maximum number of iterations is reached.

4.8.3.3 Hard-Decoding Based on Message-Passing Algorithm

The above BF decoding can also be described using code's Tanner graph, resulting in the hard-decision version of message-passing algorithm (MPA). To do so, we first introduce some notation. Denote by $q_{i,j}$ the message sent from a variable node c_i to a check node s_j, and by $\sigma_{j,i}$ the message sent from a check node s_j to a variable node c_i. Both $q_{i,j}$ and $\sigma_{j,i}$ take values in the message alphabet $\mathcal{M}$. For hard-decision decoding, $\mathcal{M}$={0, 1}. For soft-decision decoding, $\mathcal{M}$=(-∞, ∞) (in the log domain); and $\mathcal{M}$=(0, 1) (in the probability domain).

Besides, we use the following notation. Let

$\mathcal{N}_v(i) = \{ j \mid h_{ji} \ne 0, 1 \le j \le q \}$表示与 variable node c_i相连的 check 节点索引集合,

$\mathcal{N}_c(j) = \{ i \mid h_{ji} \ne 0, 1 \le i \le n \}$表示与 check node s_j相连的 variable 节点索引集合.

并且，为记号简单起见，我们将使用$\mathcal{N}_v(i) \setminus j$来表示$\mathcal{N}_v(i) \setminus \{j\}$，即排除 check node s_j 之外，与 variable node c_i相连的 check 节点索引集合。(the set of check nodes that connect to variable node c_i, excluding check node s_j.) Similarly, let $\mathcal{N}_c(j) \setminus i$ denote the set of variable nodes that connect to check node s_j, excluding variable node c_i.

For a variable node c_i (a check node s_j), we use $E_v(i)$ ($E_c(j)$) to denote the set of edges that connect c_i (s_j) to nodes in $\mathcal{N}_v(i)$ ($\mathcal{N}_c(j)$). Clearly, $|\mathcal{N}_v(i)| = |E_v(i)|$. (similarly, $|\mathcal{N}_c(j)| = |E_c(j)|$.)

With these notations, we now present the hard-decision MPA. Consider a binary-input

memoryless channel. Let $y_i \in \mathcal{M}=\{0, 1\}$ be the received value corresponding to the ith variable node. Gallager's hard-decoding algorithm may be described as follows.

- <u>Gallager's hard-decoding [also called Gallager's algorithm B]:</u>
  Do the following alternatively:

  1) Update for $q_{i,j}$: For all edges $e=(c_i, s_j)$, $1 \le i \le n$, $1 \le j \le q$, do the following in parallel:

     If this is the zeroth round, then set $q_{i,j} = y_i$. Otherwise compute $q_{i,j}$ as follows.

     - If more than $\delta(|E_v(i)|-1)$ of incoming messages along edges in $E_v(i)\backslash\{e\}$ were equal to the same value $b \in \{0, 1\}$ in the previous round, then set $q_{i,j} = b$.
     - Otherwise set $q_{i,j} = y_i$.

  2) Update for $\sigma_{j,i}$: For all edges $e=(c_i, s_j)$, $1 \le i \le n$, $1 \le j \le q$, do the following in parallel:
     The check node s_j sends the variable node c_i the XOR of the values it received in this round along edges in $E_c(j)\backslash\{e\}$; i.e., $\sigma_{j,i} = \sum_{i' \in \mathcal{N}_c(j)\backslash\{i\}} q_{i',j} \pmod 2$.

Note that the message passed contain only extrinsic information; i.e., the value of $q_{i,j}$ depends only on the value $\sigma_{j',i}$, where j' runs over all check nodes incident on c_i other than s_j. (Similarly, for $\sigma_{j,i}$.)

Decoding is stopped after a preset number of rounds is reached, or before that, provided $\mathbf{zH}^T = \mathbf{0}$. (Each variable node can determine its most likely value based on its neighbors.) If $\mathbf{zH}^T \neq \mathbf{0}$ after a preset number of rounds is reached, then we say that the decoding has failed.

Note: All message-passing algorithms must respect the following rule, which was introduced with turbo codes.
<u>Rule [Extrinsic information principle]:</u> A message sent from a node along an edge e cannot depend on any message previously received on edge e.
This message passing is to prevent correlation between consecutive decoding iterations.

4.8.3.4 Weighted BF Decoding

The simple BF decoding given in Section 4.8.3.2 can be improved by including reliability information of the received symbols in their decoding decisions.

Consider the soft-decision received sequence $\mathbf{y} = (y_0, y_1, ..., y_{n-1})$ at the output of the

receiver matched filter. For BPSK transmission over the AWGN channel, a simple measure of the reliability of a received symbol y_l is its magnitude $|y_l|$: the larger the magnitude, the larger the reliability of the hard-decision bit z_l. This reliability measure has been used in many reliability-based algorithm for decoding linear block codes.

Let **H** be the parity-check matrix of a (d_v, d_c)-regular LDPC code $\mathcal{C}$ of length n. For $0 \le l < n$, consider the set

$$\mathbf{A}_l = \left\{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, ..., \mathbf{h}_{d_v}^{(l)}\right\}$$

of d_v rows in **H** that are orthogonal on the bit position l. For $1 \le j < d_v$, consider the jth row in $\mathbf{A}_l$,

$$\mathbf{h}_j^{(l)} = \left(h_{j,0}^{(l)}, h_{j,1}^{(l)}, ..., h_{j,n-1}^{(l)}\right)$$

For simplicity of notation, we will use ρ to denote d_c. Let $0 \le i_1, i_2, ..., i_\rho < n$ be the locations where $h_{j,0}^{(l)} = h_{j,1}^{(l)} = ... = h_{j,n-1}^{(l)} = 1$. Then $\mathbf{h}_j^{(l)}$ checks the code bits, $c_{i_1}, c_{i_2}, ..., c_{i_\rho}$. Then the syndrome bit computed from **z** and $\mathbf{h}_j^{(l)}$ is

$$\begin{aligned} s_j^{(l)} = \mathbf{z} \cdot \mathbf{h}_j^{(l)} &= z_{i_1} h_{j,i_1}^{(l)} + z_{i_2} h_{j,i_2}^{(l)} + \cdots + z_{i_\rho} h_{j,i_\rho}^{(l)} \\ &= e_{i_1} h_{j,i_1}^{(l)} + e_{i_2} h_{j,i_2}^{(l)} + \cdots + e_{i_\rho} h_{j,i_\rho}^{(l)} \end{aligned}$$

We measure the reliability of the received symbol z_{i_k} with $|y_{i_k}|$. For $0 \le l < n$ and $1 \le j < d_v$, define

$$|y_j|_{\min}^{(l)} = \min\left\{|y_{i_k}|:\ 1 \le k \le \rho\right\} \tag{4.13}$$

The number $|y_j|^{(l)}$ is used as a measure of the reliability of the syndrome bit $s_j^{(l)}$.

Define the following sum

$$E_l \triangleq \sum\nolimits_{s_j^{(l)} \in \mathcal{S}_l} \left(2s_j^{(l)} - 1\right) |y_j|_{\min}^{(l)} \tag{4.14}$$

Then E_l is simply a weighted check-sum that is orthogonal on the bit position l. The weighted check-sums can be incorporated into the BF decoding algorithm to improve the decoding performance. Such a BF decoding is called *weighted BF decoding*. It is carried out as follows.

1) Compute the parity-check sums based on (4.10). If all the parity-check sums are zero, stop the decoding.
2) Compute the weighted check-sums E_l based on (4.14) for $0 \le l < n$.
3) Find the bit position l for which E_l is the largest.
4) Flip the received bit z_l.
5) Repeat steps 1) through 4). This processing of bit flipping continues until all the

parity-check sums are zero or a preset maximum number of iterations is reached.

4.8.3.5 The Sum-Product Algorithm for LDPC Decoding

We now present the soft-decision version of the message-passing algorithm, beginning with the introduction of the following notation.

Consider an LDPC code of length n given by the null space of a sparse $q \times n$ parity-check matrix $\mathbf{H}$ with rows $\mathbf{h}_1$, $\mathbf{h}_2$, …, $\mathbf{h}_q$. In the code's Tanner graph, the local constraints (check nodes) are zero-sum constraints; and the q check-sums constitute actually the syndrome vector. Denote by s_j the check sum of the jth check node (computed from the jth parity-check equation). Let

$$q_i(b) = P\left(c_i = b \mid \mathbf{y}, \text{ all checks involving } c_i \text{ are satisfied}\right)$$

$$= P\left(c_i = b \mid \mathbf{y}, \left\{s_j = 0, j \in \mathcal{N}_v(i)\right\}\right), \quad b \in \{0, 1\}$$

be the *pseudo-posteriori* probability that is used to make the decisions about the decoded bits.

$q_{ij}(b) = P\left(c_i = b \mid y_i, \left\{s_{j'} = 0, j' \in \mathcal{N}_v(i) \setminus j\right\}\right)$ denote the conditional probability that the transmitted code bit c_i has value $b \in \{0, 1\}$, given the check-sums computed from the parity-check equations corresponding to $\mathcal{N}_v(i) \setminus \{j\}$, which is the message to be passed from the variable node c_i to the check node s_j.

Let $M_v(\sim i) =$ {messages from all variable nodes except the node c_i}, and

$$\sigma_{ji}(b) = P\left(\text{check-sum } s_j = 0 \mid c_i = b, \mathbf{y}, M_v(\sim i)\right)$$

$$= P\left(\text{check-sum } s_j = 0 \mid c_i = b, \{q_{i'j}, i' \in \mathcal{N}_c(j) \setminus i\}\right)$$

denote the conditional probability of the jth check equation being satisfied, given that $c_i = b \in \{0, 1\}$ and the other code bits in $\mathcal{N}_c(j) \setminus \{i\}$ have a separable distribution $\{q_{i'j}\}_{i' \neq i}$. This is the message to be passed from the check node s_j to the variable node c_i.

At the beginning of iterative decoding, $q_{ij}(b)$ is initialized to the prior probabilities $P(c_i = b)$. Then from the discussion in Section 4.7 (cf. (4.17)), we have the following rule for message-passing in the lth iteration.

Update rule for check nodes:

$$\sigma_{ji}^{(l)}(b) = \sum_{c_{i'}: i' \in \mathcal{N}_c(j) \setminus i} P\left(s_j = 0 \mid c_i = b, \left\{c_{i'}, i' \in \mathcal{N}_c(j) \setminus i\right\}\right) \prod_{i' \in \mathcal{N}_c(j) \setminus i} q_{i'j}^{(l)}(c_{i'}) \tag{4.15}$$

The conditional probabilities in the above summation are either zero or one, depending on whether $\left\{c_i, i \in \mathcal{N}_c(j)\right\}$ constitute codewords of the jth local constraint code. Notice that in

the derivation we have used the separable distribution assumption. So for the graph without cycles, it is exact; but for the graph containing cycles, it is approximate.

The computed values of $\sigma_{ji}^{(l)}(b)$ are then used to update the values of $q_{ij}^{(l+1)}(b)$ as follows.

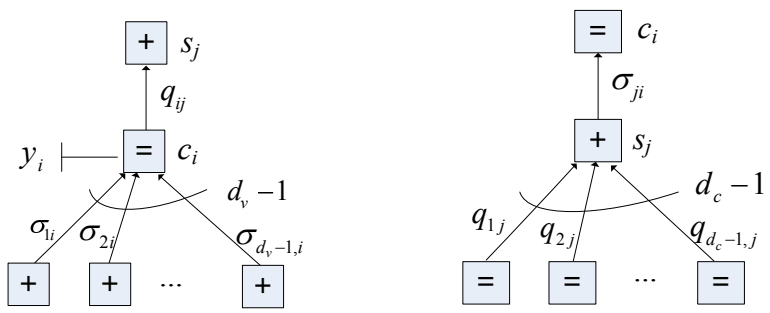
Update rule for variable nodes:

$$q_{ij}^{(l+1)}(b) = \alpha_{ij}^{(l+1)} P(c_i = b \mid y_i) \prod_{j' \in \mathcal{N}_v(i) \setminus j} \sigma_{j'i}^{(l)}(b) \tag{4.16}$$

where $\alpha_{ij}^{(l+1)}$ is a scale factor which is chosen such that

$$q_{ij}^{(l+1)}(0) + q_{ij}^{(l+1)}(1) = 1$$

式(4.16)的推导见附录。



(a) (b)

Figure 4.8.7. Flow for message update for variable and check nodes

At the *l*th iteration step, the pseudo-posteriori probabilities are given by

$$q_i^{(l)}(b) = \alpha_i^{(l)} P(c_i = b \mid y_i) \prod_{j \in \mathcal{N}_v(i)} \sigma_{ji}^{(l-1)}(b) \tag{4.17}$$

where $\alpha_i^{(l)}$ is chosen such that

$$q_i^{(l)}(0) + q_i^{(l)}(1) = 1$$

Based on these probabilities, we can make hard decisions and form the following vector as the decoded candidate,

$$\mathbf{z}^{(l)} = \left(z_0^{(l)}, z_1^{(l)}, \ldots, z_{n-1}^{(l)} \right)$$

with

$$z_i^{(l)} = \begin{cases} 1, & \text{if } q_i^{(l)}(1) > q_i^{(l)}(0) \\ 0, & \text{otherwise} \end{cases}$$

If $\mathbf{z}^{(l)}\mathbf{H}^T = \mathbf{0}$ or the maximum iteration number is reached, then stop the decoding iteration process and output $\mathbf{z}^{(l)}$ as the decoded codeword. Moreover, if $\mathbf{z}^{(l)}\mathbf{H}^T \neq \mathbf{0}$, a decoding error is declared.

For the AWGN channel model $y_i = x_i + n_i$, where $n_i \sim N(0, \sigma^2)$ and $x_i = (-1)^{c_i}$ is equally likely to be +1 or –1, it is ease to verify that

$$P(c_i \mid y_i) = P(x_i \mid y_i) = \frac{e^{x_i y_i / \sigma^2}}{e^{y_i/\sigma^2} + e^{-y_i/\sigma^2}} = \frac{1}{1 + e^{-2 y_i x_i / \sigma^2}}$$

With the assignment $q_{ij}^{(0)}(b) = P(c_i = b \mid y_i)$, the SPA can be summarized as follows.

- Initialization: Set l = 0, and maximum number of iteration to $l_{\max}$. For every (i, j) such that $h_{ji} = 1$ with $1 \le j \le q$ and $0 \le i \le n$, set

$$q_{ij}^{(0)}(0) = P(x_i = +1 \mid y_i) = \frac{1}{1 + e^{-2 y_i / \sigma^2}}$$

$$q_{ij}^{(0)}(1) = P(x_i = -1 \mid y_i) = \frac{1}{1 + e^{2 y_i / \sigma^2}}$$

- Do the following for all edges in parallel:

  1) compute the probabilities $\sigma_{ji}^{(l)}(0)$ and $\sigma_{ji}^{(l)}(1)$ according to (4.15).

  2) compute the values $q_{ij}^{(l+1)}(0)$ and $q_{ij}^{(l+1)}(1)$ according to (4.16) with $\alpha_{ij}^{(l+1)}$=1.

  Then normalize: $q_{ij}^{(l+1)}(0) = \dfrac{q_{ij}^{(l+1)}(0)}{q_{ij}^{(l+1)}(0) + q_{ij}^{(l+1)}(1)}$

  $$q_{ij}^{(l+1)}(1) = 1 - q_{ij}^{(l+1)}(0)$$

  3) Compute the values $q_i^{(l+1)}(0)$ and $q_i^{(l+1)}(1)$ for all i. Form the decision vector $\mathbf{z}^{(l+1)}$ and test $\mathbf{z}^{(l+1)}\mathbf{H}^T$. If $\mathbf{z}^{(l+1)}\mathbf{H}^T = \mathbf{0}$ or the maximum iteration number $l_{\max}$ is reached, go to next step; Otherwise, set $l = l + 1$ and go back to 1).

- Output $\mathbf{z}^{(l+1)}$ as the decoded codeword and stop the decoding process..

An example:

4.8.3.6 The Sum-Product Algorithm in the Log Domain

We now present a log-domain version of the sum-product algorithm above, which has lower complexity. First we define the following log-likelihood ratio (LLR):

$$L(c_i) \equiv \ln \frac{\Pr(c_i = 0 \mid y_i)}{\Pr(c_i = 1 \mid y_i)} = \ln \frac{\Pr(x_i = +1 \mid y_i)}{\Pr(x_i = -1 \mid y_i)}$$

$$L(q_{ij}) \equiv \ln \frac{q_{ij}(0)}{q_{ij}(1)}, \quad L(\sigma_{ji}) \equiv \ln \frac{\sigma_{ji}(0)}{\sigma_{ji}(1)}, \quad L(Q_i) \equiv \ln \frac{q_i(0)}{q_1(1)}$$

In addition, by letting $\lambda = \ln(P_0 / P_1)$, the following relationship can be shown easily:

$$\tanh\left(\frac{\lambda}{2}\right) = \frac{e^{\lambda} - 1}{e^{\lambda} + 1} = P_0 - P_1 = 1 - 2P_1 \tag{4.18}$$

■ With these definitions, the initialization step becomes (for the BI-AWGN channel)

$$L(q_{ij}) = L(c_i) = \ln \frac{\dfrac{e^{y_i/\sigma^2}}{e^{y_i/\sigma^2} + e^{-y_i/\sigma^2}}}{\dfrac{e^{-y_i/\sigma^2}}{e^{y_i/\sigma^2} + e^{-y_i/\sigma^2}}} = 2y_i / \sigma^2$$

■ To obtain the log version of (4.15), we first introduce the following lemma.

Lemma 1 (Gallager): Consider a sequence of m independent binary digits $\boldsymbol{a} = (a_1, a_2, ..., a_m)$ in which $\Pr(a_k = 1) = p_k$. Then the probability that **a** contains an even number of 1's is

$$\Pr\left(\sum_{k=1}^{m} a_k = 0\right) = \frac{1}{2} + \frac{1}{2}\prod_{k=1}^{m}(1 - 2p_k) \tag{4.19}$$

Correspondingly, the probability that **a** contains an odd number of 1's is $\frac{1}{2} - \frac{1}{2}\prod_{k=1}^{m}(1 - 2p_k)$.

Proof: (By induction on m)

When m=2, $\Pr(even) = \Pr(a_1 \oplus a_2 = 0) = p_1 p_2 + (1 - p_1)(1 - p_2) = \frac{1}{2} + \frac{1}{2}(1 - 2p_1)(1 - 2p_2)$

Assume that 错误！未找到引用源。 holds for m=L-1. With $Z_L \equiv a_1 + ... + a_L$, we have

$$\begin{aligned}
\Pr(Z_L = 0) &= \Pr(Z_{L-1} \oplus a_L = 0) \\
&= \frac{1}{2} + \frac{1}{2}\left[1 - 2\Pr(Z_{L-1} = 1)\right](1 - 2p_L) \\
&= \frac{1}{2} + \frac{1}{2}\left[1 - 1 + \prod_{k=1}^{L-1}(1 - 2p_k)\right](1 - 2p_L) = \frac{1}{2} + \frac{1}{2}\prod_{k=1}^{L}(1 - 2p_k)
\end{aligned}$$

Then the lemma immediately follows.

Now we consider the calculation of $L(\sigma_{ji})$. By the definition,

$$\sigma_{ji}^{(l)}(1) = \Pr(\text{第 } j \text{ 个校验方程中除 } c_i \text{ 之外的其它 } d_c\text{-1 个比特包含奇数个 1})$$

From Lemma 1, the probability of an odd number of 1's in the other d_c-1 bits of the jth parity-check equation is given by

$$\frac{1}{2} - \frac{1}{2} \prod_{k \in N_c(j) \setminus \{i\}} (1 - 2P_{kj})$$

where P_{kj} denote the probability that the kth digit in the jth parity-check equation involving c_i is 1. Under the message-passing algorithm, we make the assignment $q_{kj}(1) \to P_{kj}$. Then we have

$$\sigma_{ji}^{(l)}(1) = \frac{1}{2} - \frac{1}{2} \prod_{i' \in N_c(j) \setminus \{i\}} \left(1 - 2q_{i'j}^{(l)}(1)\right) \tag{4.20}$$

$$\sigma_{ji}^{(l)}(0) = 1 - \sigma_{ji}^{(l)}(1) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in N_c(j) \setminus \{i\}} \left(1 - 2q_{i'j}^{(l)}(1)\right)$$

Thus,

$$L(\sigma_{ji}^{(l)}) = \log \frac{\sigma_{ji}^{(l)}(0)}{\sigma_{ji}^{(l)}(1)} = \log \left(\frac{\frac{1}{2} + \frac{1}{2} \prod_{i' \in N_c(j) \setminus \{i\}} \left(1 - 2q_{i'j}^{(l)}(1)\right)}{\frac{1}{2} - \frac{1}{2} \prod_{i' \in N_c(j) \setminus \{i\}} \left(1 - 2q_{i'j}^{(l)}(1)\right)} \right)$$

Take into account that (cf(4.18))

$$\tanh\left(\frac{1}{2} L(q_{ij})\right) = 1 - 2q_{ij}(1) \tag{4.21}$$

we have

$$L\left(\sigma_{ji}^{(l)}\right) = \log \left(\frac{1 + \prod_{i' \in N_c(j) \setminus \{i\}} \tanh\left(\frac{L(q_{i'j}^{(l)})}{2}\right)}{1 - \prod_{i' \in N_c(j) \setminus \{i\}} \tanh\left(\frac{L(q_{i'j}^{(l)})}{2}\right)} \right) \tag{4.22}$$

On the other hand, from (4.18),

$$\tanh\left(\frac{1}{2} L(\sigma_{ji})\right) = 1 - 2\sigma_{ji}(1)$$

Substituting (4.20) into this equation yields

$$\tanh\left(\frac{1}{2}L(\sigma_{ji}^{(l)})\right) = \prod_{i'\in N_c(j)\setminus\{i\}} \left(1-2q_{i'j}^{(l)}(1)\right) = \prod_{i'\in N_c(j)\setminus\{i\}} \tanh\left(\frac{1}{2}L(q_{i'j}^{(l)})\right) \tag{4.23}$$

Therefore, the update rule in (4.22) for check nodes can also be expressed as

Update rule for check nodes:

$$L\left(\sigma_{ji}^{(l)}\right) = 2\tanh^{-1}\left[\prod_{i'\in N_c(j)\setminus\{i\}} \tanh\left(\frac{1}{2}L\left(q_{i'j}^{(l)}\right)\right)\right] \tag{4.24}$$

Following Gallager, we factor $L(q_{ij})$ into $L(q_{ij}) = \alpha_{ij}\beta_{ij}$, where

$$\alpha_{ij} \triangleq \operatorname{sign}\left(L(q_{ij})\right) \text{ and } \beta_{ij} \triangleq \left|L(q_{ij})\right|,$$

respectively. Then (4.24) can be rewritten as

$$\begin{aligned} L(\sigma_{ji}) &= \prod_{i'\in N_c(j)\setminus\{i\}} \alpha_{i'j} \cdot 2\tanh^{-1}\left[\prod_{i'\in N_c(j)\setminus\{i\}} \tanh\left(\frac{1}{2}\beta_{i'j}\right)\right] \\ &= \prod_{i'\in N_c(j)\setminus\{i\}} \alpha_{i'j} \cdot 2\tanh^{-1}\log^{-1}\left[\sum_{i'\in N_c(j)\setminus\{i\}} \log\tanh\left(\frac{1}{2}\beta_{i'j}\right)\right] \\ &= \left(\prod_{i'\in N_c(j)\setminus\{i\}} \alpha_{i'j}\right) f\left[\sum_{i'\in N_c(j)\setminus\{i\}} f\left(\beta_{i'j}\right)\right] \end{aligned} \tag{4.25}$$

where we have defined

$$f(x) \triangleq -\ln\tanh\left(\frac{x}{2}\right) = \ln\frac{e^x+1}{e^x-1} \tag{4.26}$$

and used the fact that $f^{-1}(x) = f(x)$ when $x > 0$:

$$f\left(f(x)\right) = \ln\frac{e^{f(x)}+1}{e^{f(x)}-1} = x$$

Notice that the function $f(x)$ is fairly well behaved, as shown in Fig. 4.8.8, and so may be implemented via a look-up table.

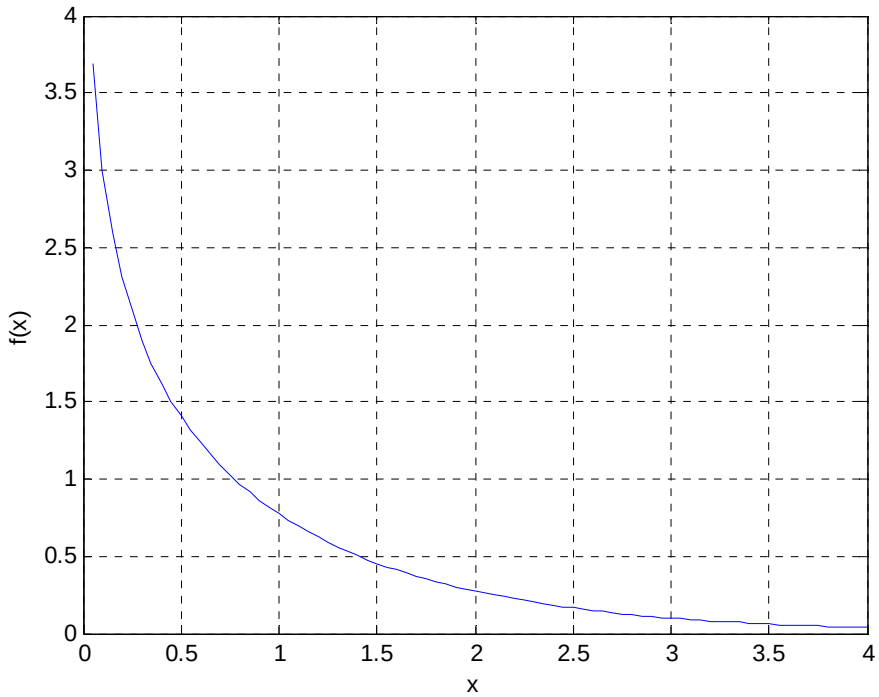


Figure 4.8.8 $f(x)$ function

- For the log version of (4.16), it is seen that

Update rule for variable nodes

$$L\left(q_{ij}^{(l+1)}\right)=L(c_i)+\sum_{j'\in N_v(i)\setminus\{j\}}L\left(\sigma_{j'i}^{(l)}\right) \tag{4.27}$$

- Step (3) is similarly modified:

$$L\left(Q_i^{(l)}\right)=L(c_i)+\sum_{j\in N_v(i)}L\left(\sigma_{ji}^{(l-1)}\right)$$

- Finally, for all $1\le i\le n$,

$$z_i^{(l)}=\begin{cases}1, & \text{if } L\left(Q_i^{(l)}\right)<0\\ 0, & \text{otherwise}\end{cases}$$

Note: 在对数域 SPA 算法的实际编程实现中，对于校验节点的消息更新，我们更喜欢采用对每两个变量计算对数似然比的方法（box-plus）而不是式（4.24）。设二元随机变量 $Z=X\oplus Y$，其中 $X=\sum_{i=1}^{n}\oplus X_i$，$Y=\sum_{i=1}^{m}\oplus Y_i$（$\oplus$ 为异或运算）。假定我们已经获得变量 X 和 Y 的对数似然比值，并记

$$L(X)=\log\frac{P(X=0)}{P(X=1)},\qquad L(Y)=\log\frac{P(Y=0)}{P(Y=1)}$$

则有

$$L(Z)=\log\frac{P(X\oplus Y=0)}{P(X\oplus Y=1)}$$

$$= \log \frac{P(X=0)P(Y=0)+P(X=1)P(Y=1)}{P(X=0)P(Y=1)+P(X=1)P(Y=0)}$$

$$= \log \frac{\frac{P(X=0)}{P(X=1)} \cdot \frac{P(Y=0)}{P(Y=1)} + 1}{\frac{P(X=0)}{P(X=1)} + \frac{P(Y=0)}{P(Y=1)}}$$

$$= \frac{e^{L(X)+L(Y)}+1}{e^{L(X)}+e^{L(Y)}} \tag{4.28}$$

对每一个校验节点（方程 $Z_L \equiv a_1 \oplus ... \oplus a_L$），递归应用式(4.28)：

$$Z_1 = a_1, Z_2 = Z_1 \oplus a_2, Z_3 = Z_2 \oplus a_3, ..., Z_L = Z_{L-1} \oplus a_L ,$$

即可获得关于校验和的 LLR 值。显然，这种算法本质上是工作于 Trellis 图上的 SPC 码的前向后向递归算法。

4.8.3.7 The Min-Sum Algorithm (MSA)

Consider the update rule in (4.25) for check codes:

$$L(\sigma_{ji}) = \left(\prod_{i' \in N_c(j) \setminus \{i\}} \alpha_{i'j} \right) f\left[\sum_{i' \in N_c(j) \setminus \{i\}} f\left(\beta_{i'j}\right) \right]$$

Notice the property of $f(x)$, we have

$$f\left[\sum_{i' \in N_c(j) \setminus \{i\}} f\left(\beta_{i'j}\right) \right] \approx f\left[f\left(\min_{i'} \beta_{i'j} \right) \right] = \min_{i'} \beta_{i'j} \tag{4.29}$$

Thus, with step (1) replaced by $L(\sigma_{ji}) = \left(\prod_{i' \in N_c(j) \setminus \{i\}} \alpha_{i'j} \right) \cdot \min_{i'} \beta_{i'j}$, we obtain the min-sum algorithm.

4.8.3.8 修正 Min-Sum Algorithm

在 LDPC 码的实际应用中，SPA 算法由于其性能严重地依赖于信道参数（如噪声方差），它的应用范围就受到了限制。实用中，更多地是采用修正 Min-Sum 算法，其性能不依赖于信道噪声方差的变化，并且通过加权或比例因子修正可以达到近似和积算法的性能。

- Normalized MSA

从前面的讨论中可知，MSA 算法可以通过除以一个大于 1 的常量来对 $L(\sigma_{ji})$ 进行进一步改进，从而得到较精确的软信息，这就是 Normalized-MSA 算法。此时检验节点的更新公式为：

$$L(\sigma_{ji}) \approx \left(\prod_{i' \in \mathcal{N}_c(j) \backslash i} sign\left(L(q_{i'j})\right) \right) \times \frac{\min\limits_{i' \in \mathcal{N}_c(j) \backslash i} \left| L(q_{i'j}) \right|}{\alpha}.$$

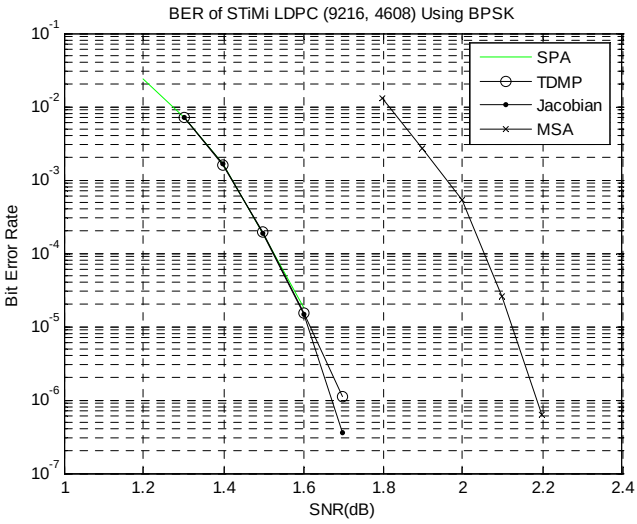
理论上，α应该随不同的信噪比(SNR)和迭代次数而改变以得到最优的结果，但为了简单起见α保持为一个常量，估计α的一个较好的方法是采用密度进化(DE)。

- Offset MSA

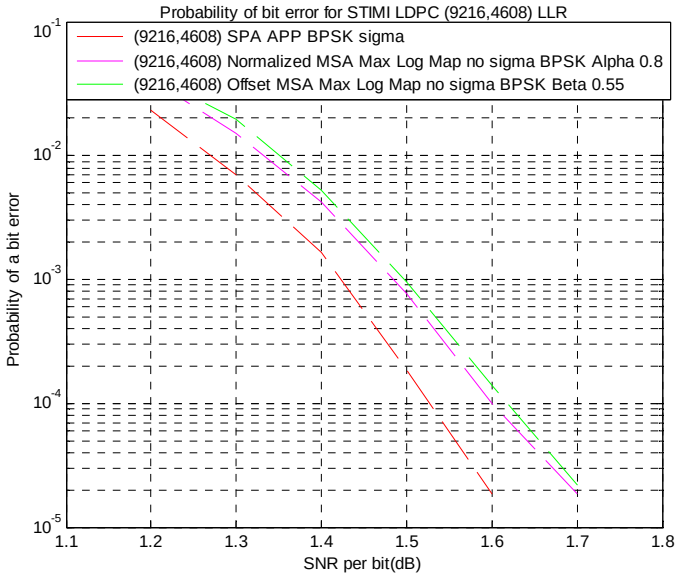
$$L(\sigma_{ji}) \approx \left(\prod_{i' \in \mathcal{N}_c(j) \backslash i} sign\left(L(q_{i'j})\right) \right) \times \max\left\{ \min_{i' \in \mathcal{N}_c(j) \backslash i} \left| L(q_{i'j}) \right| - \beta, 0 \right\}$$

这种方法与Normalized-MSA算法的不同之处在于，模值小于β的LLR信息值被置为零，也就是忽略了其对下一轮变量节点更新的影响。

图 4.8.8-1 给出了不同译码算法的性能比较。LDPC 码是移动电视标准 STiMi 中的 LDPC(9216, 4608)码，信道环境为 AWGN。仿真结果表明，当 alpha=$1/\alpha = 0.8$ 时，采用 Noamalized-MSA 算法可以获得较好的译码性能，与标准 BP 算法相比只有 0.1dB 的性能损失。



(a)



(b)

图 4.8.8-1　LDPC(9216,4608)码在 BPSK 调制下不同译码算法的性能比较

4.8.3.9 Iterative Soft Reliability-Based Decoding Algorithm

An iterative soft reliability-based (ISRB) majority-logic decoding (MLGD) algorithm and a modified version by replacing the soft reliability measures of received bits by some type of hard reliability measures (the resulting algorithm is called an iterative hard reliability-based MLGD-algorithm) 见 [Huang-Lin-2010, TCom]。

4.8.4 Density Evolution (Analysis of LDPC Codes on Symmetric Binary Input Channels)

Density evolution (DE) is an analytical technique which has been used to understand the performance of iterative decoding of long LDPC codes. It also provides a tool which can be used in the design of families of LDPC codes, since their performance can be predicted using density evolution much more rapidly than the performance can be simulated. Density evolution introduces the idea of a channel threshold, above which the code performs well and below which the probability of error is non-negligible. This provides a single parameter characterizing code performance which may be used to gauge the performance compared to the ultimate limit of the channel capacity. Usually, the maximum level of channel noise which is likely to be corrected by a particular ensemble of codes using the message-passing algorithm (MPA) is used to indicate this threshold. 我们通过跟踪迭代译码过程中消息的概率分布（密度）函数的演进来确定这个噪声阈值。

In density evolution, we make a key assumption that the block length $n \to \infty$, under which it may be assumed there are no cycles in the factor graph. Furthermore, 假定信道是无记忆的。这两个假设能够使得每一个节点处的输入消息是相互独立的。我们还假定信道是输出对称的：

$$p(y_k = q \mid x_k = 1) = p(y_k = -q \mid x_k = -1)$$

Since the code is linear, and we have assumed a symmetric channel, it suffices for this analysis to assume that the all-zero codeword **c** = **0** is sent.

注意：DE 分析的是(d_v, d_c)-code ensemble 在迭代译码下的平均性能（收敛性分析）

4.8.4.1 DE on the binary erasure channel (BEC)

■ The BEC

The binary erasure channel (BEC) models a memoryless channel with two inputs {0,1} and three outputs, {0, 1, *e*}, where "*e*" is an "erasure symbol." The probability that any transmitted bit will be received correctly is $1-\varepsilon$, that it will be erased is ε, and that it will be received incorrectly is zero. See Fig. 4.8.9.

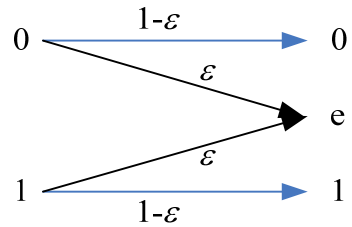


Figure 4.8.9 A binary erasure channel

■ (Iterative) Message-passing decoding of LDPC codes on the BEC

On the BEC, a transmitted bit is either received correctly or completely erased with some probability ε. Since the bits which are received are always completely correct, the task of the decoder is to determine the value of the unknown bits. So iterative decoding using the sum-product algorithm reduces simply to the propagation of unerased variables through the code graph:

- A variable node sends the same outgoing message *M* to each of its connected check nodes. This message declares the value of the bit '1', '0' if it is known or '*e*' if it is erased.
- If a check node receives only one '*e*' message, it can calculate the value of the unknown variable/bit by setting the value to the mod-2 sum of other inputs; otherwise, the outgoing messages from the check node remain unchanged.
- The check nodes send back different messages to each of their connected variable nodes. The message from the *j*th check node to the *i*th variable node declares the value of the *i*th bit '1', '0' or 'e' as determined by the *j*th check node.
- If the variable node of an erased bit receives an incoming message which is '1' or '0' the bit node changes its value to the value of the incoming message.
- This process is repeated until all of the bit values are known, or until some maximum number of decoder iterations has passed and the decoder gives up.

Example 4.8.6:

■ Analysis of regular LDPC codes

Consider the ensemble $\mathcal{C}(d_v, d_c)$ which consists of all regular LDPC codes with variable nodes of degree d_v and check nodes of degree d_c. Let $q_{v\to c}^{(l)}$ and $q_{c\to v}^{(l)}$ denote,

respectively, the probability that at iteration l a variable-to-check message is an 'e' and the probability that at iteration l a check-to-variable message is an 'e'.

The check to variable message on an edge is 'e' if one or more of the incoming messages on the other ($d_c - 1$) edges into that check node is an 'e'. With the assumption that at each node, the incoming messages are independent of each other, the probability that none of the other $d_c - 1$ incoming message to the check node is 'e' is simply the product of the probabilities, $\left(1 - q_{v\to c}^{(l)}\right)$, that each individual message is not 'e'. So the probability that one or more of the other incoming messages are 'e' is

$$q_{c\to v}^{(l)} = 1 - \left(1 - q_{v\to c}^{(l)}\right)^{d_c - 1} \tag{4.30}$$

At iteration l the variable to check message will be 'e' if the original message from the channel was an erasure, which occurs with probability ε, and all of the incoming messages at iteration $l - 1$ are erasures, which each have probability $q_{c\to v}^{(l)}$. Again we make the assumption that all of the incoming messages are independent of each other, and so the probability that the variable to check message is an 'e' is the product of the probabilities that the other $d_v - 1$ incoming messages to the variable node, and the original message from the channel, were erased. That is,

$$q_{v\to c}^{(l)} = \varepsilon \left(q_{c\to v}^{(l-1)}\right)^{d_v - 1} \tag{4.31}$$

Substituting (4.30) into (4.31) yields

$$q_{v\to c}^{(l)} = \varepsilon \left(1 - \left(1 - q_{v\to c}^{(l-1)}\right)^{d_c - 1}\right)^{d_v - 1} \tag{4.32}$$

Prior to decoding, 我们置 $q_{v\to c}^{(l)}$ 的初始值为the probability that the channel erased a codeword bit:

$$q_{v\to c}^{(0)} = \varepsilon$$

Thus for a (d_v, d_c)-regular ensemble

$$\boxed{q_{v\to c}^{(0)} = \varepsilon, \qquad q_{v\to c}^{(l)} = \varepsilon \left(1 - \left(1 - q_{v\to c}^{(l-1)}\right)^{d_c - 1}\right)^{d_v - 1} \qquad (4.33)}$$

The recursion in (4.33) describes how the erasure probability of message-passing decoding evolves as a function of the iteration number l for (d_v, d_c)- regular LDPC codes. Applying this recursion we can determine for which erasure probabilities the message-passing decoder is likely to correct the erasures.

Example 4.8.7 [Johnson, pp.40]:

A code from the (3, 6)-regular ensemble is to be transmitted on a binary erasure channel with erasure probability ε = 0.3 and decoded with the message-passing algorithm. The probability that a code bit will remain erased after l iterations of message-passing decoding (if the code Tanner graph is cycle free) is given by the recursion:

$$q_{v\to c}^{(0)} = \varepsilon, \quad q_{v\to c}^{(l)} = q_{v\to c}^{(0)}\left(1-\left(1-q_{v\to c}^{(l-1)}\right)^5\right)^2,$$

Applying this recursion for 7 iterations gives the sequence of bit erasure probabilities,

$$q_{v\to c} = [0.3,\ 0.2076,\ 0.1419,\ 0.0858,\ 0.0392,\ 0.0098,\ 0.0007,\ 0.0000]$$

Thus the erasure probability in a codeword from a 4-cycle free (3, 6)-regular LDPC code transmitted on a BEC with erasure probability 0.3 will approach zero after seven iterations of message-passing decoding.

We now want to know which erasure probabilities the codes from a (3, 6)-regular ensemble are likely to be able to correct. Applying the recursion (4.32) to a range of channel erasure probabilities we see, in Fig. 4.8.10, that for values of $\varepsilon \geq 0.43$ the probability of remaining erasures does not decrease to zero even as l gets very large, whereas, for values of $\varepsilon \leq 0.42$ the probability of error does approach zero as $l \to \infty$. The transition value of ε, between these two outcomes is the called the *threshold* of the (3, 6)-regular ensemble. Again applying the recursion (4.32), it can be verified that the threshold for a (3, 6)-regular ensemble on the binary erasure channel is between 0.4293 and 0.4294.

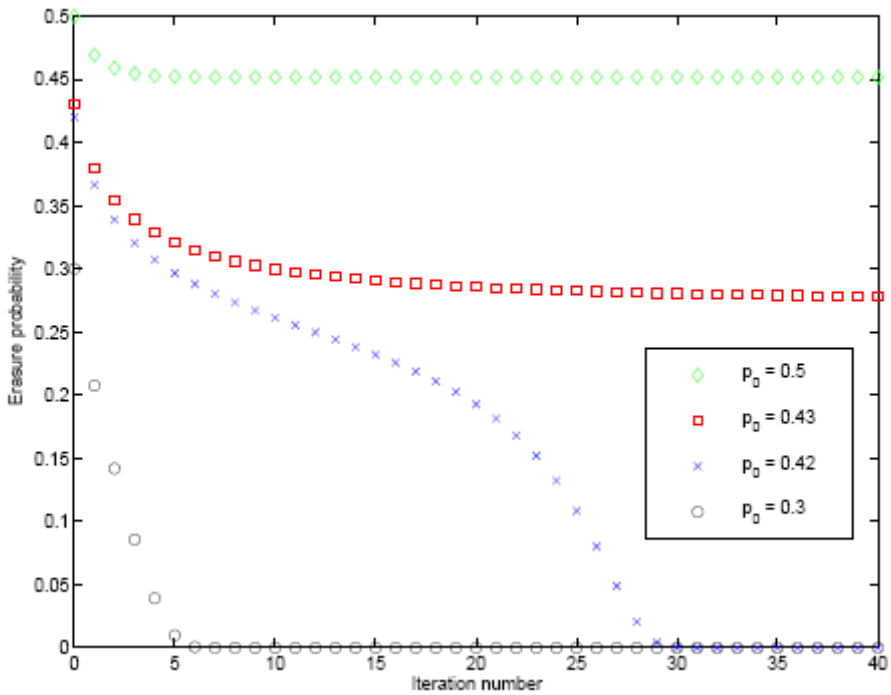


Figure 4.8.10 The erasure probabilities calculated in Example 4.8.7

■ Analysis of irregular LDPC codes

Irregular LDPC codes can provide improved performance by allowing for the degrees of (variable and check) nodes widely varying. A good amount of mathematical theoretic analysis has been developed [Luby01]. Following are some basic knowledge for construction of

irregular codes.

Consider an irregular bipartite graph. Assume that the LDPC code has length n and that the number of variable nodes of degree d is L_d, so that $\sum_d L_d = n$. In the same manner, denote the number of check nodes of degree d by R_d, so that $\sum_d R_d = n(1-R_c)$, where R_c is the nominal code rate. Let E be the total number of edges in the graph. Denote by l_d the number of edges that connect to variable nodes of degree d, and r_d the number of edges that connect to check nodes of degree d. Then

$$l_d = dL_d, \quad r_d = dR_d$$

$$E = \sum_d l_d = \sum_d r_d \tag{4.34}$$

For convenience, we introduce the following polynomials

$$L(x) = \sum_{d=1}^{d_v} L_d x^d, \quad R(x) = \sum_{d=1}^{d_c} R_d x^d \tag{4.35}$$

$$l(x) = \sum_{d=1}^{d_v} l_d x^{d-1}, \quad r(x) = \sum_{d=1}^{d_c} r_d x^{d-1} \tag{4.36}$$

where d_v and d_c are the maximum variable degree and maximum check degree, respectively. From (4.35),

$$L'(x) = \frac{dL(x)}{dx} = \sum_{d=1}^{d_v} dL_d x^{d-1} = \sum_{d=1}^{d_v} l_d x^{d-1} = l(x), \tag{4.37}$$

$$R'(x) = \frac{dR(x)}{dx} = \sum_{d=1}^{d_c} dR_d x^{d-1} = \sum_{d=1}^{d_c} r_d x^{d-1} = r(x) \tag{4.38}$$

From these definitions, we can see that

$$L(1) = n, \; R(1) = n(1-R_c), \; L'(1) = R'(1) = E, \text{ and } R_c = 1 - R(1)/L(1) \tag{4.39}$$

We will call $L(x)$ and $R(x)$ the *left and right (or variable and check) degree distributions from a node perspective*.

On the other hand, from the viewpoint of asymptotic analysis, it will be more convenient to take on an edge perspective. Define

$$\lambda(x) = \frac{l(x)}{E} = \frac{l(x)}{l(1)} = \sum_{d=1}^{d_v} \lambda_d x^{d-1} \tag{4.40}$$

$$\rho(x) = \frac{r(x)}{E} = \frac{r(x)}{r(1)} = \sum_{d=1}^{d_c} \rho_d x^{d-1} \tag{4.41}$$

which will be called the *left and right (or variable and check) degree distributions from a edge perspective*. Note that λ_d is equal to the fraction of edges that connect to variable nodes of degree d and ρ_d is equal to the fraction of edges that connect to check nodes of degree d.

For example, for the (3, 6) regular code, we have $\lambda(x) = x^2$ and $\rho(x) = x^5$.

From (4.37) and (4.40), the inverse relationships read

$$L(x) = \sum_{d=1}^{d_v} \left(l_d / d\right) x^d = \int_0^x l(z)dz = E\int_0^x \lambda(z)dz \tag{4.42}$$

Similarly, from (4.38) and (4.41) we have

$$R(x) = \sum_{d=1}^{d_c} \left(r_d / d\right) x^d = \int_0^x r(z)dz = E\int_0^x \rho(z)dz \tag{4.43}$$

Using these functions, we can describe many important parameters of the graph.

- The number of variable nodes of degree d can be expressed as $L_d = E\lambda_d / d$
- The number of variable nodes $n = \sum_d L_d = E\sum_d \lambda_d / d$
- The average variable degree is $\bar{d}_v = \dfrac{E}{n} = \dfrac{1}{\sum_d \lambda_d / d} = \dfrac{1}{\int_0^1 \lambda(x)dx}$
- The average check degree is $\bar{d}_c = \dfrac{E}{n(1-R_c)}$

Analogous assertion hold for $\rho(x)$. Assuming that all check equations are linearly independent. Then the design rate is given by

$$R_c = \frac{n - E\sum_d \rho_d / d}{n} = 1 - \frac{\sum_d \rho_d / d}{\sum_d \lambda_d / d} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \tag{4.44}$$

The analysis of iterative decoding of irregular LDPC codes may be carried out nicely in terms of these generating functions. From (4.30) we know that, at the lth iteration, the probability that a check-to-variable message is 'e' is

$$q_{c\to v}^{(l)} = 1 - \left(1 - q_{v\to c}^{(l)}\right)^{d-1}$$

for an edge connected to a degree-d check node. For an irregular graph, the probability that an edge is connected to a degree-d check node is ρ_d. Thus, the average probability that a check-to-bit message is in error is given by

$$q_{c\to v}^{(l)} = \sum_d \rho_d \left(1 - \left(1 - q_{v\to c}^{(l)}\right)^{d-1}\right) = 1 - \sum_d \rho_d \left(1 - q_{v\to c}^{(l)}\right)^{d-1}$$

By the definition of ρ(x) in (4.41), this becomes

$$q_{c\to v}^{(l)} = 1 - \rho\left(1 - q_{v\to c}^{(l)}\right) \tag{4.45}$$

Similarly, the average probability that a bit-to-check message is in error at the lth iteration is given by

$$q_{v\to c}^{(l)} = \varepsilon \sum_d \lambda_d \left(q_{c\to v}^{(l-1)}\right)^{d-1} = \varepsilon\lambda\left(q_{c\to v}^{(l-1)}\right) \tag{4.46}$$

where we have used the definition of λ(x) in (4.40).
Finally, combining (4.45) and (4.46), we have

$$q_{v\to c}^{(l)} = \varepsilon\lambda\left(1-\rho\left(1-q_{v\to c}^{(l-1)}\right)\right)$$

Thus, for irregular LDPC codes we have the recursion:

$$\boxed{q_{v\to c}^{(0)} = \varepsilon\,, \qquad q_{v\to c}^{(l)} = \varepsilon\lambda\left(1-\rho\left(1-q_{v\to c}^{(l-1)}\right)\right) \qquad (4.47)}$$

■ Threshold

The aim of DE is to determine for which channel erasure probability, ε, the iterative decoder is likely to correct all of the erased bits. Using (4.47) we have a means of approximating this as an average over all LDPC codes with a given degree distribution λ and ρ, by assuming that the graphs are cycle free.

Define the function

$$f(q,\varepsilon) = \varepsilon\lambda\left(1-\rho(1-q)\right)$$

wehre $q \in [0,1]$ and $\varepsilon \in [0,1]$. The erasure probability at iteration l is then

$$q^{(l)}(\varepsilon) = f\left(q^{(l-1)},\varepsilon\right)$$

Note that $f(q,\varepsilon)$ is a strictly increasing function in q for $\varepsilon > 0$. Thus if $q^{(l)} > q^{(l-1)}$, then

$$q^{(l+1)} = f\left(q^{(l)},\varepsilon\right) \geq f\left(q^{(l-1)},\varepsilon\right) = q^{(l)}$$

So $q^{(l)}(\varepsilon)$ is a monotone sequence which is lower-bounded at q=0 by

$$f\left(0,\varepsilon\right) = \varepsilon\lambda\left(1-\rho(1)\right) = \varepsilon\lambda(1-1) = 0$$

and upper-bounded at q=1 by

$$f\left(1,\varepsilon\right) = \varepsilon\lambda\left(1-\rho(1-1)\right) = \varepsilon\lambda(1) = \varepsilon$$

Therefore, we have

$$0 \leq f\left(q,\varepsilon\right) \leq \varepsilon$$

Thus $q^{(l)}$ converges to an element $q^{(\infty)} \in [0,\varepsilon]$. It is shown that there is a value ε^* called the threshold such that for values of ε below ε^*, $q^{(l)} \to 0$ as $l \to \infty$ while for values of ε above ε^*, it does not. The threshold ε^* for (λ, ρ) is defined as the supremum of ε for which $q^{(l)}(\varepsilon) \to 0$:

$$\varepsilon^*(\lambda,\rho)=\sup\left\{\varepsilon\in[0,1]:q^{(l)}(\varepsilon)|_{l\to\infty}\to 0\right\} \tag{4.48}$$

4.8.4.2 DE on the binary input AWGN channel (BI-AWGN)

下面我们以 AWGN 信道为例，分析一般无记忆信道中 LDPC 码的迭代译码性能。假定采用对数域上的和积译码算法，传递的消息是对数似然比（LLR）。由于这些 LLR 是连续取值的，因此我们将用概率密度函数来描述一个 LLR 取某个特定值的概率。为叙述方便起见，我们用$V=\log\left(\sigma(0)/\sigma(1)\right)$表示迭代译码过程中从变量节点到校验节点的 LLR 消息，用$U=\log\left(q(0)/q(1)\right)$表示从校验节点到变量节点传递的 LLR 消息。Under sum-product decoding, a degree-d_v variable node implements the MAP decoding function of a [d_v, 1] repetition code, 其消息更新规则为：

$$V=\sum_{i=0}^{d_v-1}U_i \tag{4.49}$$

where $U_i, i=1,...,d_v-1$ are the incoming messages from the neighbors of the variable node, except the check node to which message V is sent, and U_0 is the channel message associated with the variable node.

如同对于 BEC 信道的分析，这里我们仍然假定所有输入消息是相互独立的（即假定信道是无记忆的并且码的因子图上没有环存在）。式(4.49)表示 d_v 个随机变量之和，因此 V 的概率密度 can be easily calculated by convolution of densities of the U_i's. 令 $p_X(x)$ 表示随机变量 x 的概率密度函数，则我们有

$$p_V=\bigotimes_{i=0}^{d_v-1}p_{U_i} \tag{4.50}$$

其中，“$\otimes$”表示卷积。显然，上式可以通过应用 Fourier 变换技术来高效计算。

对于非规则 LDPC 码，我们在变量节点的度分布 $\lambda(x)$ 上取平均，有

$$p_V=p_{U_0}\otimes\sum_d\lambda_d\bigotimes_{i=1}^{d}p_{U_i}$$

A degree-d_c check node implements the MAP decoding function of a [d_c, d_c-1] SPC code, 其消息更新规则为：

$$\tanh\left(\frac{U}{2}\right)=\prod_{j=1}^{d_c-1}\tanh\left(\frac{V_j}{2}\right) \tag{4.51}$$

where $V_j, j=1,...,d_c-1$ are the incoming messages from the neighbors of the check node, except the variable node to which the message U is sent.

To apply the Fourier 变换技术 to (4.51), 我们需要在(4.51)式两边取对数，将乘积运算转换为求和运算：

$$\left(\operatorname{sgn}(U), \log\left|\tanh\frac{U}{2}\right|\right) = \sum_{j=1}^{d_c-1}\left(\operatorname{sgn}(V_j), \log\left|\tanh\frac{V_j}{2}\right|\right) \tag{4.52}$$

其中 sgn(x)表示符号运算。这里，我们定义

$$\operatorname{sgn}(x) = \begin{cases} 0, & \text{if } x \geq 0 \\ 1, & \text{otherwise} \end{cases}$$

并且(4.52)中符号的求和运算 is performed in $\mathbb{Z}_2$ and the sum for the magnitude is the ordinary sum in $\mathbb{R}$.

Let $\gamma(x)$: $[-\infty, +\infty] \to \{0,1\} \times [0, \infty]$ be the function defined by

$$\gamma(x) = \left(\gamma_1(x), \gamma_2(x)\right) = \left(\operatorname{sgn}(x), \log\left(\tanh\left|\frac{x}{2}\right|\right)\right)$$

Then (4.52) can be written as

$$\gamma\left(U^{(l)}\right) = \sum_{j=1}^{d_c-1}\gamma\left(V_j^{(l-1)}\right)$$

for the *l*th iteration. Hence, 校验节点到变量节点传递的消息可以表示为：

$$U^{(l)} = \gamma^{-1}\left(\sum_{j=1}^{d_c-1}\gamma\left(V_j^{(l-1)}\right)\right)$$

这样，校验节点的 DE can be done numerically in the Fourier domain. 用分布函数描述的 DE 见[Moon, pp.666; Richardson 2001].

In [Chung01a], Chung et al. 提出了一种 DE 的高效数值计算方法，称为 discretized density evolution (DDE)，其基本思想是将消息的分布进行量化，然后在变量节点用 FFT 计算更新消息的分布，而在校验节点用查表的方法计算。这对应于量化的 MPA, which is suboptimal compared to exact message-passing. Any decoding threshold thus obtained is therefore a lower bound on the actual threshold. In [Chung01a], this method has been used to design rate-1/2 LDPC codes which has a threshold within 0.0045dB of the BPSK bound. Simulations of actual codes with code length of 10e+7 show that the performance within 0.04dB of BPSK bound can be achieved.

■ Gaussian approximation (GA)

上述连续空间上的 DE，需要跟踪无限维空间上消息密度的进化，因此计算复杂度很高。为了简化分析与计算，人们提出了高斯近似方法 for DE over BI-AWGN channel，即假定因子图上传递的消息具有近似高斯分布，这样消息密度迭代计算中的无限维问题便简化为跟踪高斯分布的单个参数变化这样一个一维问题。高斯近似方法有多种形式：

- A method based on the mean value of the message was suggested in [Chung01b];
- A method based on the SNR was used in analysis of turbo codes in [Gamal01];
- A method based on BER for analysis of LDPC and product codes was proposed in [Lehmann03];

- A method based on the mutual information measure was suggested in [tenBrink, Sharon06]; This method uses EXIT functions of constituent decoders in the iterative decoding scheme.
- A MSE-based method was suggested in [Bhattad07], which use MSE charts instead of EXIT charts. This is based on the recent results of Guo, Shamai and Verdu [Guo05] about the relationship between mutual information and MMSE for the AWGN channel.

下面我们首先介绍最简单的基于消息均值的方法[Moon05]。

Assume that the all-zero codeword **c** = **0** is sent. With mapping $x_k = \sqrt{E_s}\left(1-2c_k\right)$, the received signal is given by

$$y_k = \sqrt{E_s} + n_k$$

where $n_k \sim \mathcal{N}(0,\sigma_n^2)$. The initial LLR channel message is then

$$U_{0,k} = \frac{2\sqrt{E_s}}{\sigma_n^2} y_k = \frac{2\sqrt{E_s}}{\sigma_n^2}\left(\sqrt{E_s} + n_k\right)$$

which is Gaussian. The mean and variance are, respectively,

$$m_U = \mathsf{E}[U_{0,k}] = \frac{2E_s}{\sigma_n^2}, \quad \sigma_U = \mathrm{var}[U_{0,k}] = \frac{4E_s}{\sigma_n^2} = 2m_U,$$

Thus, $U_0 \sim \mathcal{N}\left(m_U, 2m_U\right)$. A Gaussian r.v. having the property that its variance is equal to twice its mean is said to be consistent. Consistent r.v.'s are convenient because they are described by a single parameter, the mean.

Consider the computation of V in (4.49). Clearly, U_0 is Gaussian, but other U_i's are not Gaussian. As the message V is computed as the sum of other random variables, a central limit argument can be made that it should tend toward Gaussian. Furthermore, numerical experiments confirm that they are fairly Gaussian. The messages U sent by check nodes is non-gaussian, but again numerical experiments confirm that they can be approximately represented by Gaussians. In the interest of analytical tractability, we assume that all messages are not only Gaussian, but consistent. The density evolution analysis tracks the parameters of these Gaussian random variables through the decoding process.

Let $m_U^{(l)} = \mathsf{E}[U_i^{(l)}]$ and $m_V^{(l)} = \mathsf{E}[V_i^{(l)}]$ 表示第 l 次迭代中传递消息的均值。We assume that both the $U_i^{(l)}$ and $V_i^{(l)}$ are consistent r.v.'s, so

$$U_i^{(l)} \sim \mathcal{N}\left(m_U^{(l)}, 2m_U^{(l)}\right), \qquad V_i^{(l)} \sim \mathcal{N}\left(m_V^{(l)}, 2m_V^{(l)}\right)$$

Taking the expectation of both sides of (4.49), we have

$$m_V^{(l)} = \frac{2E_s}{\sigma_n^2} + (d_v - 1)m_U^{(l-1)} \tag{4.53}$$

对于校验节点更新方程(4.51)，taking the expectation of both sides, we have

$$\mathsf{E}\left[\tanh\left(\frac{U^{(l)}}{2}\right)\right]=\mathsf{E}\left[\prod_{j=1}^{d_c-1}\tanh\left(\frac{V_j^{(l)}}{2}\right)\right] \tag{4.54}$$

Now define the function

$$\Psi(x)=\mathsf{E}\left[\tanh\left(y/2\right)\right],\quad y\sim\mathcal{N}\left(x,2x\right)$$

Then (4.54) can be written as （用高斯变量 y 来近似 U, V）

$$\Psi\left(m_U^{(l)}\right)=\left(\Psi\left(m_V^{(l)}\right)\right)^{d_c-1} \tag{4.55}$$

Substituting (4.53) into (4.55), we obtain

$$\Psi\left(m_U^{(l)}\right)=\left(\Psi\left(\frac{2E_s}{\sigma_n^2}+(d_v-1)m_U^{(l-1)}\right)\right)^{d_c-1}$$

or

$$m_U^{(l)}=\Psi^{-1}\left(\left(\Psi\left(\frac{2E_s}{\sigma_n^2}+(d_v-1)m_U^{(l-1)}\right)\right)^{d_c-1}\right)$$

The recursion is initialized with $m_U^{(0)}=0$. The dynamics are completely determined by the variable node degree d_v, check node degree d_c, and the SNR= E_s/σ_n^2.

令 $p_V^{(l)}(v)$ 表示第 l 次迭代后变量节点输出消息的概率密度函数，则在发送全'0'码字假设下，decoded bit in error 的概率等于 Pr(LLR is negative)，因此迭代译码器产生的译码错误概率为：

$$P_e^{(l)}=P\left(V^{(l)}\le 0\right)=\int_{-\infty}^{0}p_V^{(l)}(v)dv=\int_{-\infty}^{0}\frac{1}{\sqrt{4\pi m_V^{(l)}}}\exp\left\{-\frac{\left(v-m_V^{(l)}\right)^2}{4\pi m_V^{(l)}}\right\}dv$$

观察 $P_e^{(l)}$ 随着 l 与 SNR 的变化，即可确定译码门限。

- 基于外信息转移（EXIT）图的 LDPC 码迭代译码性能分析（**基于互信息测度的高斯近似**）

在对非规则 LDPC 码的高斯近似分析中，前述跟踪消息均值的方法会变得不够准确，我们需要考虑其它消息度量。一般地，基于互信息测度的 GA 能给出最准确的译码门限预测。因此，下面我们将详细介绍基于 EXIT 函数的 LDPC 码分析技术。

假设同上：一个节点的所有输入消息独立，且在 BI-AWGN 信道上发送的是全'0'码字（由于满足对称条件，译码性能独立于发送码字）；传递的消息假定具有 consistent 高斯密度。

LDPC 码的迭代译码可看作是由重复码和 SPC 码的一系列基本译码所组成，因此 LDPC 的分析将依赖于这些分量码的 EXIT 函数。LDPC 码的 EXIT 图分析模型如图 4.8.12 所示，在结构上它类似于一个由重复码和 SPC 码构成的串行级联编码系统，其中（一组）重复码用作内码，（一组）SPC 码用作外码。

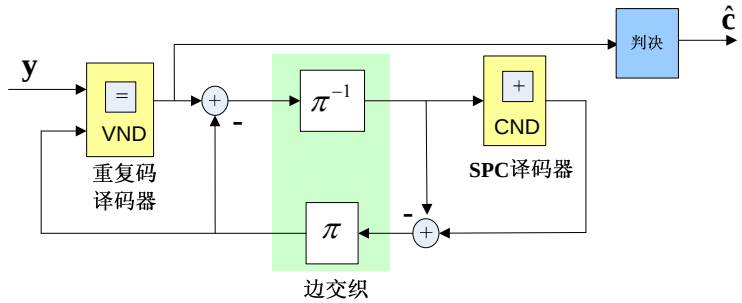


图 4.8.12 LDPC 码的迭代译码器结构

下面我们讨论变量节点（重复码）和校验节点（SPC 码）的消息更新及 EXIT 函数计算问题。首先考虑规则 LDPC 码。

Let $m_U^{(l)} = \mathrm{E}[U_i^{(l)}]$ for $i \neq 0$, $m_{U_0} = \mathrm{E}[U_0] = 2E_s/\sigma_n^2$, and $m_V^{(l)} = \mathrm{E}[V_i^{(l)}]$. Taking the expectation of both sides of (1), we get at the lth iteration

$$m_V^{(l)} = m_{U_0} + (d_v - 1)m_U^{(l-1)} \tag{4.56}$$

where $m_U^{(0)} = 0$. 变量节点的输入消息的统计独立性保证 V 的 consistency 条件 is maintained. 基于(4.56)，我们能够较容易地计算出 V 所对应的发送变量与 V 之间的互信息。

推导 update rule for $m_U^{(l)}$ 要复杂一些。即使我们假定校验节点的输入消息是独立的且具有 consistent 高斯密度，其输出消息 U which is given in (4.51) is not Gaussian, due to the 校验节点计算规则的非线性。在上一小节中，我们用一个高斯变量 U_g 来近似校验节点的输出 message U，该变量满足条件：

$$\mathrm{E}\left[\tanh\frac{U_g}{2}\right] = \mathrm{E}\left[\tanh\frac{U}{2}\right]$$

这里，我们用一个满足如下互信息准则的高斯变量 U_g 来替换 U [Sharon06]:

$$I(X;U) = I(X;U_g),$$

其中 X 是与消息 U 相对应的发送符号（比特）。然后，使用下列 update rule for m_U :

$$m_U^{(l)} = J^{-1}\left(I\left(X;U^{(l)}\right)\right) \tag{4.57}$$

其中函数 $J(m)$ 定义为

$$J(m) = I(X;Z)$$

$$= 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{4\pi m}} \exp\left(-\frac{(z-m)^2}{4m}\right) \log_2\left(1+\exp(-z)\right) dz \tag{4.58}$$

它表示一个 consistent 高斯变量 $(Z \mid X = 1) \sim \mathcal{N}(m, 2m)$ 和一个二元随机变 X (taking

values in {+1, -1} with equal probability)之间的互信息。

注：式(4.58)的推导参见（4.76）。这里参数 m 具有方差性质；而在通常的 $J(\sigma)$ 定义中，参数 sigma 是标准离差，不是方差。函数 $J(\cdot)$ and $J^{-1}(\cdot)$ 的近似计算见[ten Brink-comm04].

Let $T_V = \tanh\frac{V}{2}$ and $T_U = \tanh\frac{U}{2}$. Then it can be shown easily that $I(X;U) = I(X;T_U)$. 关于式(4.57)中 $I(X;U)$的计算，我们在后面进行讨论。

下面讨论变量节点和校验节点 EXIT 函数的计算。从信息理论的角度来看，变量节点的译码模型是一个对应于重复编码的并行信道传输问题，如图 4.8.13 所示，该并行信道由一个实际通信信道和 d_v-1 个虚拟信道所组成。变量节点的输入消息 U_i is modeled as 一个二进制输入对称无记忆信道（BI-SMC）（更具体地说，是一个高斯信道）的输出 LLR 值，该信道的输入是二元编码符号 X。

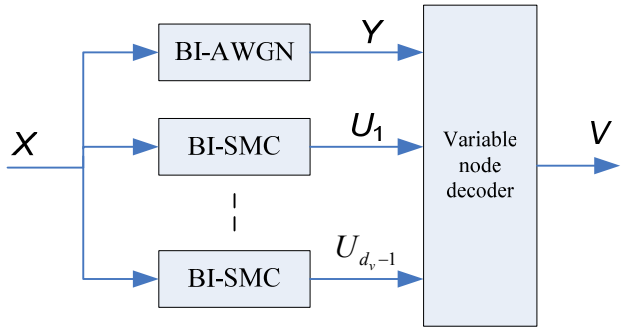


图 4.8.13 变量节点的译码模型（互信息计算模型）

For the 变量节点 of code symbol X, the mutual information

$$I_{A,v} = I(X;U_i)$$

is called the 先验（互）信息；the mutual information between the code symbol and the outgoing message

$$I_{E,v} = I(X;V) = I\left(X;Y,U_1,\ldots,U_{d_v-1}\right)$$

is called the extrinsic (mutual) information. As the value is the same for all outgoing messages, this is also the average extrinsic information.

The mapping from 先验信息 to 外信息 with the 信道信息 as a parameter,

$$\text{EXIT}_v : I_{A,v} \to I_{E,v}$$

is called the variable-node EXIT function.

For the check node connected to the variable node of the code symbol X, the mutual information

$$I_{A,c} = I(X_i;V_i)$$

is called the 先验（互）信息；The mutual information

$$I_{E,c} = I(X;U) = I\left(X;V_1,\ldots,V_{d_c-1}\right)$$

is called the extrinsic (mutual) information. 这也是平均外信息。校验节点的互信息计算模型如图 4.8.14 所示。

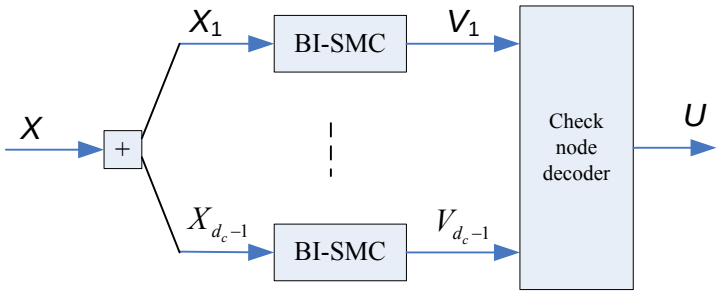


图 4.8.14 校验节点的译码模型

The mapping

$$\mathrm{EXIT}_c : I_{A,c} \to I_{E,c}$$

is called the check-node EXIT function.

在迭代译码中，变量节点的先验信息是校验节点的外信息，and vice versa, 因此 the following identities hold:

$$I_{A,c} = I_{E,v} \text{ and } I_{A,v} = I_{E,c}$$

The EXIT chart is a plot of both EXIT functions in a single diagram. 译码门限（threshold）is the SNR at which the two EXIT curves just touch.

通常，EXIT 函数 are determined by running the constituent decoders. 而对于 LDPC 码来说，they can be determined analytically. From [Sharon-comm, July 2006], we have

$$I(X;U) = \frac{1}{\ln 2}\sum_{i=1}^{\infty}\frac{1}{(2i)(2i-1)}\mathsf{E}\left[T_U^{2i}\right]$$

$$= \frac{1}{\ln 2}\sum_{i=1}^{\infty}\frac{1}{(2i)(2i-1)}\left(\mathsf{E}\left[T_V^{2i}\right]\right)^{d_c-1} \tag{4.59}$$

其中 $\mathsf{E}\left[T_V^{2i}\right]$ can be computed using the following function [Sharon-comm, July 2006]

$$\Phi_i(m) \approx \mathsf{E}\left[T_V^{2i}\right] = \int_{-1}^{+1}\frac{2t^{2i}}{(1-t^2)\sqrt{4\pi m}}\exp\left\{-\frac{\left(\ln\left((1+t)/(1-t)\right)-m\right)^2}{4m}\right\}dt$$

这样，式(4.57)给出的 $m_U^{(l)}$ 更新规则可写为

$$\boxed{m_U^{(l)} = J^{-1}\left(\frac{1}{\ln 2}\sum_{i=1}^{\infty}\frac{1}{(2i)(2i-1)}\left(\Phi_i\left(m_V^{(l)}\right)\right)^{d_c-1}\right) \qquad (4.60)}$$

为方便起见，我们也将变量节点的消息更新规则重复如下：

$$m_V^{(l)} = m_{U_0} + (d_v - 1) m_U^{(l-1)} \tag{4.61}$$

Using a finite number of terms in the series of (4.59), we can approximate $I(X;U)$ as accurately as we wish.

The function $J(m)$ and $\Phi_i(m)$, $i = 1,...,\infty$, can be computed using a pre-processed table, or by an approximation of the functions.

在 EXIT chart analysis of LDPC codes 中，我们将(4.60) and (4.61)看作是传递消息的方差更新公式，这样便可以基于（4.60）和（4.61）这两个消息更新规则，计算出前述的两个 EXIT 函数。

- 变量节点 EXIT 函数 $\mathrm{EXIT_v}$：

$$\begin{aligned} I_{E,v} = I(X;V) &= I\left(X;Y,U_1,...,U_{d_v-1}\right) \\ &= f\left(I(X;Y), I(X;U)\right) \\ &= f\left(I_{ch}, I_{A,v}\right) \\ &= J\left(J^{-1}(I_{ch}) + (d_v - 1) J^{-1}(I_{A,v})\right) \\ &= J\left(\frac{\sigma_n^2}{2} + (d_v - 1) J^{-1}(I_{A,v})\right) \end{aligned} \tag{4.62}$$

- 校验节点 EXIT 函数 $\mathrm{EXIT_c}$：

$$\begin{aligned} I_{E,c} = I(X;U) &= I\left(X;V_1,...,V_{d_c-1}\right) \\ &= f\left(I(X;V)\right) \\ &= f\left(I_{A,c}\right) \\ &= \frac{1}{\ln 2} \sum_{i=1}^{\infty} \frac{1}{(2i)(2i-1)} \left(\Phi_i\left(J^{-1}(I_{A,c})\right)\right)^{d_c-1} \end{aligned} \tag{4.63}$$

Fig. 4.8.15 shows an example of an EXIT chart for a (3, 6)-regular LDPC code over the BI-AWGN channel.

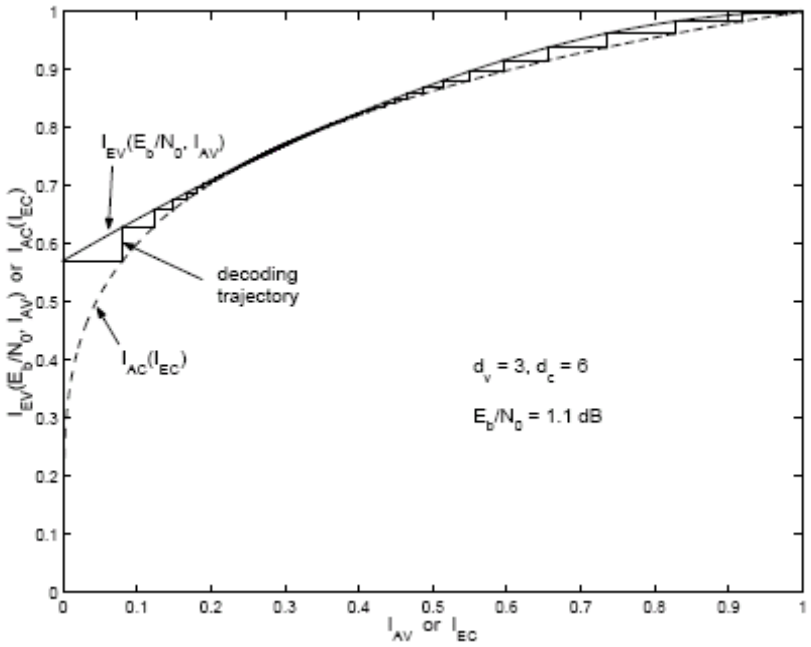


Figure 4.8.15 EXIT chart example for (d_v, d_c) = (3, 6) regular LDPC code.

推广到非规则 LDPC 码：

The generalized update rules for m_V and m_U are given by

$$m_V^{(l)} = J^{-1}\left(\sum_{d=1}^{d_v}\lambda_d J\left(\frac{2E_s}{\sigma_n^2}+(d-1)m_U^{(l-1)}\right)\right) \tag{4.64}$$

$$m_U^{(l)} = J^{-1}\left(\sum_{d=1}^{d_c}\rho_d \frac{1}{\ln 2}\sum_{i=1}^{\infty}\frac{1}{(2i)(2i-1)}\left(\Phi_i\left(m_V^{(l)}\right)\right)^{d-1}\right) \tag{4.65}$$

变量与校验节点的 EXIT 函数分别为：

$$I_{E,v} = \sum_{d=1}^{d_v}\lambda_d I_{E,v_d} = \sum_{d=1}^{d_v}\lambda_d J\left(\frac{E_s}{\sigma_n^2}+(d-1)J^{-1}\left(I_{A,v}\right)\right)$$

$$I_{E,c} = \sum_{j=1}^{d_c}\rho_j I_{E,c_j} = \sum_{d=1}^{d_c}\rho_d \frac{1}{\ln 2}\sum_{i=1}^{\infty}\frac{1}{(2i)(2i-1)}\left(\Phi_i\left(J^{-1}(I_{A,c})\right)\right)^{d-1}$$

$I_{E,c}$的简化（近似）计算：

$I_{E,c}$可通过解析或仿真方法确定，下面是一个基于对偶码原理的简单近似方法：

SPC 码和重复码是对偶码。For BEC, 码长为 d_c 的 SPC 码的 EXIT 曲线与长为 d_c 的重复码的 EXIT 曲线之间存在下列对偶关系：

$$I_{E,SPC}\left(I_A, d_c\right) = 1 - I_{E,REP}\left(1-I_A, d_c\right) \tag{4.66}$$

这个特性 is not exact for BI-AWGNC 先验输入，but it is very accurate. 为使用方便起见，将其写为

$$I_{E,c} \approx 1 - I_{E,REP}\left(1 - I_{A,c}, d_c\right)$$

$$= 1 - J\left((d_c - 1)J^{-1}(1 - I_{A,c})\right) \tag{4.67}$$

其逆函数为：

$$I_{A,c} \approx 1 - J\left(\frac{J^{-1}(1 - I_{E,c})}{d_c - 1}\right)$$

Remark: 由于 EXIT chart 给出的 threshold 只是一个近似值，为此一种更好的方法是给出一个 threshold 的取值范围，于是人们提出了 information combining 技术 [Land-Huber2003]，可用来确定 threshold 的上下界。

4.8.6 Design and Construction of LDPC codes -- Overview

We now proceed to consider the design and construction of LDPC codes. 像一般线性分组码一样，LDPC 码也分为二元码和多元码，在这一节中我们主要讨论二元 LDPC 码的设计与构造。我们的目标是希望设计具有低编码复杂度、好的 waterfall 区性能和低的 error floors 的 LDPC 码。

Clearly, the most obvious path to the construction of an LDPC code is via the construction of a low-density parity-check matrix with prescribed properties. A large number of design techniques exist in the literature, and we introduce some of the more prominent ones in this section.

给定码率等参数之后，LDPC 码的设计一般从选择节点度分布开始。首先采用 DE or EXIT chart 等技术选择度分布，使得所设计的码在迭代译码下具有好的收敛阈值；然后依据此分布采用随机或代数等方法构造校验矩阵 **H** (即 For a given pair $\lambda(x)$ and $\rho(x)$ of degree distributions, constructing a random bipartite graph with n variable nodes and $(n-k)$ check nodes which has this degree distribution)，在此过程中需要采用有关优化技术使所构造的 **H** 具有较大的围长（好的环分布），以便保证最终的码具有好的 error floor 性能。

（准）循环 LDPC 码与基于累加器结构的 LDPC 码在上述三个性能方面具有较好的折衷，因此目前获得了较广泛的应用。

4.8.6.1 选择度分布

We have seen that the threshold of an ensemble of codes with a given degree distribution can be found using density evolution or 以EXIT chart为代表的GA技术. The question for code designers is then which degree distributions will produce the best threshold.

Generally, the more irregular the bit degree distribution the better. The capacity approaching LDPC codes are both very long and very irregular. The famous LDPC ensemble with threshold 0.0045 dB from the Shannon limit are 10^7 bits long with nodes having degrees varying from 2 to 8000. The subset of the bits nodes with very high degree will very quickly converge to their solution, and once correct will pass high LLR values to their many connected nodes. Since the overall density of **H** needs to be low, a large proportion of degree-2 bit nodes are also required to reduce the average node degree. Thus a degree distribution with a good threshold will contain a few very high degree bit nodes, many degree

two nodes, but no more than allowed for by stability, and some nodes with degrees in between these. Irregularity in the check node degrees is not as essential and generally one or two check degrees, chosen to achieve the required average row weight is sufficient.

Trying every possible distribution that fits this general pattern is of course not practical and so optimization techniques are used to find the best degree distribution subject to the desired constraints. Optimizing over the density evolution algorithm is not straightforward, in particular because a gradient for the cost function is not defined. Nevertheless, two general optimization algorithms have been applied to finding the degree distributions of LDPC codes, *iterative linear programming* and the *differential evolution*.

对于二元码，it has been observed that 为了使变量节点和校验节点 EXIT 函数曲线具有好的匹配，变量节点的不同度的数量（the number of different VN degrees）大约只需要 3 或 4 种，and the number of different CN degrees need only be 1 or 2.

关于 the optimal degree distribution 的进一步讨论，见 [Richardson01] and the references therein.

Example 4.8.8: [Liva06, Design of ldpc codes]

4.8.6.2 LDPC 码的随机构造

对于规则的 LDPC 码，Gallager 和 MacKay 给出了两种简单的随机构造方法。

4.8.6.2.1 Gallager codes

Let $k \geq 1$ be a positive integer. For a given choice of d_c and d_v, Gallager presented the following construction of LDPC codes.

Form a $kd_v \times kd_c$ matrix **H** that consists of d_v $k \times kd_c$ submatrices, $\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_{d_v}$. Each row of a submatrix has d_c 1's, and each column of a submatrix contains a single 1. Therefore, each submatrix has a total of kd_c 1's. For $1 \leq i \leq k$, the ith row of $\mathbf{H}_1$ contains all its d_c 1's in the columns $(i-1)d_c + 1$ to id_c. The other submatrices are merely column permutations of $\mathbf{H}_1$. Thus

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_{d_v} \end{bmatrix}$$

From this construction of **H**, it is clear that

- no two rows in a submatrix of **H** have any 1-component in common; and
- no two columns in a submatrix of **H** have more than one 1-component in common.

Because the total number of ones in **H** is kd_cd_v and the total number of entries in H is $k^2d_cd_v$, the density of **H** is $1/k$. If k is chosen much greater than 1, **H** has a very low density and is a sparse matrix. Whether **H** possesses the 3rd property of the parity-check matrix of an LDPC code defined in Definition 4.8.1 depends on the choice of $d_v - 1$ column permutations of the submatrix $\mathbf{H}_1$. Random permutations of columns of $\mathbf{H}_1$ to form the other submatrices result in a class of linear block codes that contains a subclass of LDPC codes defined by Definition

4.8.1.

Gallager showed that the ensemble of such codes has excellent distance properties provided $d_v \geq 3$ and $d_c > d_v$. However, Gallager did not provide a method for choosing column permutations of the submatrix $\mathbf{H}_1$ to form the other submatrices, $\mathbf{H}_2, \mathbf{H}_3, \cdots, \mathbf{H}_{d_v}$, such that the overall matrix $\mathbf{H}$ gives an LDPC code with good minimum distance and the structural properties required in Definition 4.8.1. Computer searches are need to find good LDPC codes, especially for long codes.

Gallager codes were generalized by Tanner in 1981.

■ Example 4.8.9: Let $k = 5$, $d_c = 4$, and $d_v = 3$. Using Gallager's construction, we can form a 15×20 matrix $\mathbf{H}$ as shown in Fig. 4.8.17, which consists of three 5×20 submatrices, $\mathbf{H}_1$, $\mathbf{H}_2$, and $\mathbf{H}_3$. Each row of $\mathbf{H}_1$ consists of four consecutive 1's, and no two rows have any 1-component in common. Submatrices $\mathbf{H}_2$ and $\mathbf{H}_3$ are obtained by two different permutations of columns of $\mathbf{H}_1$ such that no two columns (two rows) of $\mathbf{H}$ have more than 1-component in common. The null space of $\mathbf{H}$ gives a (3, 4)-regular LDPC code of length 20.

$$\mathbf{H} = \left[\begin{array}{cccccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{array}\right]$$

图 4.8.17 (20,3,4)低密度校验码的校验矩阵

4.8.6.2.2 Semi-random construction of LDPC codes: MacKay codes

MacKay had independently discovered the benefits of designing binary codes with sparse parity-check matrices and was the first to show the ability of these codes to perform near capacity limits [MacKay96]. He provided in [] algorithms to semi-randomly generate sparse $\mathbf{H}$ matrices. A commonly used method is as follows.

$\mathbf{H}$ is first generated with weight-d_v columns, weight-d_c row, and no two columns having overlap greater than one. Then short cycles are voided for code optimization. The detailed construction algorithm is listed below.

1) Set column counter $j = 0$.
2) Generate a weight-d_v columns vector and place in the column j of **H**.
3) If the weight of each row in **H** is $\leq d_c$ and all cycle lengths are $\geq L_{min}$, then $j=j+1$;
4) If $j=n$, stop; else, go to 2).

Note that the above construction may lead to **H** matrix that is not full rank. This is generally not a problem that can be seen as follows.

1) Once **H** is constructed, we will put it in the form $\mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}]$, via Gauss-Jordan elimination (and possible column swapping) so that we may encode via $\mathbf{G} = [\mathbf{I} \vdots \mathbf{P}]$.
2) If **H** is not full rank, Gauss-Jordan elimination will result in **H** of the form

$$\mathbf{H} = \begin{bmatrix} \tilde{\mathbf{P}}^T & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

   where $\tilde{\mathbf{P}}^T$ is of $r' \times (n-r')$ and $r' < r = n-k$.

3) In this latter case, we instead use the matrix $\tilde{\mathbf{H}} = [\tilde{\mathbf{P}}^T \mid \mathbf{I}]$ which is $r' \times n$ and corresponding to a higher code rate $(n-r')/n$.

4.8.6.2.3 Bit Filling or PEG 构造方法

In bit filling, variable nodes are added to the Tanner graph one at a time and edges connecting the new variable nodes to the graph are chosen to avoid cycles of size g. For each new variable node c_i, d_v check nodes are selected to join by an edge to c_i. The set of feasible check nodes are the nodes that are distance $g/2$ or more edges away from all of the check nodes already connected to c_i. From each set of feasible check nodes the check node chosen is the one which least used so far (i.e. with the lowest degree). Fig 4.8.18 shows this process.

In progressive edge growth, Tanner graphs edges are similarly added to the graph one at a time but instead of meeting some fixed girth requirement g the edge is added so as to maximize the local girth at the current variable node.

These techniques can also be applied to semi-structured codes, i.e. an RA code can be constructed by fixing the accumulator portion of the Tanner graph and applying bit filling to the remaining variable nodes.

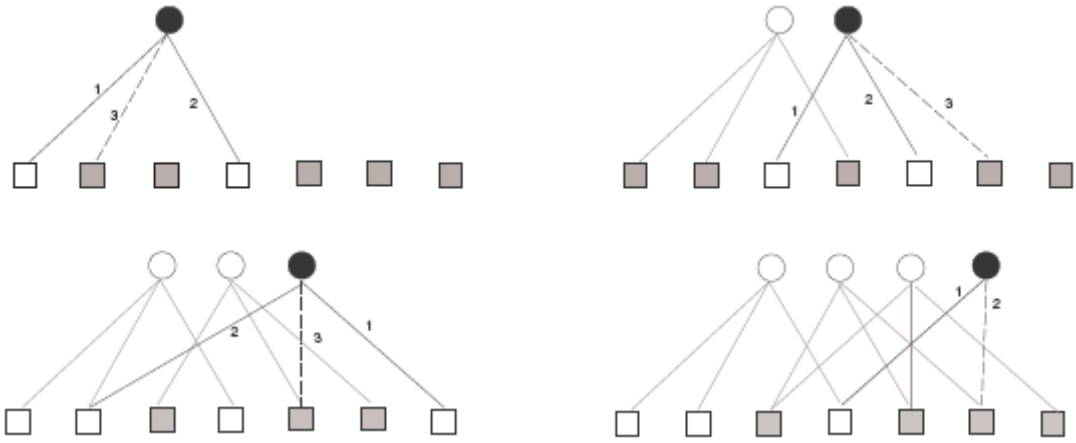


Figure 4.8.18

4.8.6.3 Structured LDPC Codes Based on Finite Geometries

The LDPC codes based on random construction require high-complexity encoders, since they lack sufficient structure to allow simple encoding, although Richardson have proposed a clever encoding algorithm whose complexity is approximately linear in the code length. Structured LDPC codes given by Lin, W.E. Ryan, et al. provide important alternatives to these codes.

In [Kou-Lin01], regular LDPC codes are designed using techniques based on finite geometries (FG). The resulting LDPC codes (known as FG LDPC codes) fall into the cyclic and quasi-cyclic classes of block codes and lend themselves to simple encoder implementation via shift-register circuits.

Finite geometries such as, **Euclidean and projective geometries**, were successfully used in early 1960's and early 1970's to construct majority-logic decodable codes. In the early 2000's, finite geometries were shown to be a powerful mathematical tool to construct structured LDPC codes that perform well with iterative decoding.

A finite geometry is a collection of "points," which are m-tuples over GF(q), and "lines," upon which sets of points lie. For constructing LDPC codes, the finite geometry $\mathcal{G}$ is employed, where $\mathcal{G}$ is a finite geometry with n points and m lines with the following structural properties:

(1) Each line consists of d_c points;

(2) Any two points are connected by exactly one line;

(3) Two lines either are parallel (having no point in common), or they intersect at exactly one point (having one point in common).

(4) Each point lies on d_v lines (i.e., each point $\boldsymbol{p}$ in $\mathcal{G}$ is intersected by a fixed number of d_v lines);

For any line $\mathcal{L}$ in such a geometry $\mathcal{G}$**,** we form a binary incidence vector defined by

$$\mathbf{v}_{\mathcal{L}} = (v_1, v_2, \ldots, v_n)$$

where v_i=1 if v_i corresponds to a point on $\mathcal{L}$, and v_i = 0 otherwise. The vector $\mathbf{v}_{\mathcal{L}}$ simply displays the points on $\mathcal{L}$ and it has weight d_c.

We form a $m \times n$ matrix $\mathbf{H}_G$ over GF(2) with the incidence vectors of all the lines in the FG $\mathcal{G}$ as rows. Then the columns and rows of $\mathbf{H}_G$ correspond to the points and lines of $\mathcal{G}$, respectively, with $h_{ij} = 1$ if and only if the ith line of $\mathcal{G}$ contains the jth point of $\mathcal{G}$**.** It follows that $\mathbf{H}_G$ has the following properties:

- Each row has constant weight d_c;
- Each column has constant weight d_v.;
- Two rows (or two columns) have at most one 1-component in common.

If d_c is much smaller than n, then $\mathbf{H}_G$ is a sparse (d_v, d_c)-regular matrix that satisfies the RC-constraint. The null space of $\mathbf{H}_G$ gives a (d_v, d_c)-regular LDPC code of length n with minimum distance at least d_v+1.

Based on this idea, parity check matrices for codes are constructed [Kou-Lin01] that offer the following potential advantages:

- Several different decoding algorithms exist, from one step majority logic decoders with low complexity, through the usual sum-product algorithm with higher complexity.
- The code can be extended by splitting each column of **H.** If done properly, performance within a few dB of capacity can be achieved.
- Codes derived from finite geometries may be cyclic or nearly cyclic, structure that enables them to be efficiently encoded.

The cyclic FG codes tend to have relatively large minimum, but the quasi-cyclic codes tend to have somewhat small minimum distances. Also, short LDPC codes (n on the order of 200 bits) designed using these techniques are generally better than short LDPC codes designed using pseudo-random **H** matrices.

The cyclic FG codes have the drawback that the parity-check matrix used in decoding is n by n instead of $(n-k) \times n$. The $n \times n$ matrix is circulant, with its first row equal to a certain incidence vector. Another drawback is that the values of d_v and d_c are relatively large which is undesirable since the complexity of the SPA is proportional to these values. 再一个不足之处是在码长与码率的选择上缺乏灵活性。

4.8.7 Algebraic Construction of LDPC codes Based on Finite Fields

见本讲义 Chapter 4-3b.

4.8.8 Accumulator-Based LDPC Codes

见本讲义 Chapter 4-3b.

4.8.9 Nonbinary LDPC codes

LDPC codes over GF(q), q>2.

Construction: 基于有限域构造的多元 LDPC 码

设α为伽逻华域 GF(q) (q>2)上的本原元，$\alpha^{-\infty} \triangleq 0, \alpha^0 = 1, \alpha, ..., \alpha^{q-2}$组成了 GF($q$)上的所有元素。对于每一个非零元素$\alpha^i$可构成 q -1 维向量

$$\mathbf{z}(\alpha^i) = (z_0, z_1, z_2, ..., z_{q-2}),$$

其中，第 i 个分量$z_i = \alpha^i$，其余 q-2 个分量均为零，该向量称为域元素α^i的 q 元位置向量。根据以上定义，我们采用基于本原元的方法，分别由以下几个步骤构造 GF(q)上的 LDPC 码：

1、定义 GF(q)上的一个$(q-1)\times(q-1)$矩阵 $\mathbf{W}$，向量$\mathbf{w}_0 = \left(0, \alpha-1, \alpha^2-1, ..., \alpha^{q-2}-1\right)$作为该矩阵的第一行，其余$q$–2行由向量 $\mathbf{w}_0$分别循环左移 1 次得到，即：

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_{q-2} \end{bmatrix} = \begin{bmatrix} 0 & \alpha-1 & \cdots & \alpha^{q-2}-1 \\ \alpha-1 & \alpha^2-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{q-2}-1 & 0 & \cdots & \alpha^{q-3}-1 \end{bmatrix},$$

其中$\mathbf{w}_{i+1}$由$\mathbf{w}_i$循环左移一位得到，且$0 \le i < q-1$。该 $\mathbf{W}$ 矩阵有以下四点特征：

(1) 任意两行中位置相同的项均不相同；
(2) 任意两列中位置相同的项均不相同；
(3) 每行或每列中的 q-1 项均是 GF(q)上的不同元素；
(4) 每行或每列有且只有一个零元素。

2、根据矩阵 $\mathbf{W}$ 中的每一个行向量$\mathbf{w}_i$，以$\mathbf{w}_i, \alpha\mathbf{w}_i, ..., \alpha^{q-2}\mathbf{w}_i$作为行构成一个$(q-1)\times(q-1)$矩阵

$$\mathbf{W}_i = \begin{bmatrix} \mathbf{w}_i & \alpha\mathbf{w}_i & \cdots & \alpha^{q-2}\mathbf{w}_i \end{bmatrix}^T;$$

再用 q 元位置向量替换 $\mathbf{W}_i$ 中的每一项，构成一个$(q-1)\times(q-1)^2$矩阵$\mathbf{H}_i = \begin{bmatrix} \mathbf{Q}_{i,0} & \mathbf{Q}_{i,1} & \cdots & \mathbf{Q}_{i,q-2} \end{bmatrix}$，其中 $\mathbf{Q}_{i,j}$由 $\mathbf{W}_i$中第 j (0<j<q-1)列的 q-1 个元素的 q 元位置向量组成。显然，$\mathbf{H}_i$的其中一个子矩阵是$(q-1)\times(q-1)$的零矩阵，其余 q-2 个子矩阵为单位重量的循环矩阵。

3、令$\mathbf{H}_{qc} = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_{q-2} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{Q}_{0,1} & \cdots & \mathbf{Q}_{0,q-2} \\ \mathbf{Q}_{0,1} & \mathbf{Q}_{0,2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{0,q-2} & \mathbf{0} & \cdots & \mathbf{Q}_{0,q-3} \end{bmatrix}$，

则$\mathbf{H}_{qc}$是一个行重、列重均为 q-2 的$(q-1)^2\times(q-1)^2$方阵。对于一个规则 q 元 LDPC 码，令γ, ρ分别表示其校验矩阵的列重量与行重量。对于整数对(γ, ρ)，$1 \le \gamma, \rho < q$，令$\mathbf{H}_{qc}(\gamma, \rho)$表示 $\mathbf{H}_{qc}$ 的一个$\gamma\times\rho$子阵，则由编码理论可知[22]，$\mathbf{H}_{qc}(\gamma, \rho)$的零化空间就形成一类 q 元准循环 LDPC 码。

Decoding: QSPA, FFT-based SPA, Extended MSA, 陈超-王雪鹏-白宝明的 hard message passing algorithm

4.8.10 Performance Analysis of ML-Decoded LDPC Codes

There are two types of results regarding the performance of LDPC codes. The first concerns the properties of these codes under the assumption of optimal (ML) decoding. The second relates to the properties of practical iterative decoding algorithms for these codes.

$$\text{Performance analysis}\begin{cases}\text{ML decoding}\begin{cases}\text{Bound on P(e)}\\ \text{Bound on rate}\end{cases}\\ \text{Iterative decoding}\end{cases}$$

在第 4.8.5 节我们已经讨论了 LDPC 码迭代译码的平均性能，下面简要说明 LDPC 码在最大似然译码下的性能。分析方法与分析 turbo 码性能的方法相同，下面只简要给出相关结果。

4.8.10.1 Ensemble of LDPC codes

For the analysis of LDPC codes under iterative decoding, it is convenient to introduce ensembles of codes. Rather than introducing such ensembles by a proper choice of the generator matrices or parity-check matrices, it will be preferable to define them via their Tanner graphs.

■ <u>The standard Ensemble LDPC (Λ, P):</u>

Given a degree distribution pair (Λ, P), define an ensemble of bipartite graph LDPC (Λ, P) in the following way. Each graph in LDPC (Λ, P) has Λ(1) left nodes and P(1) right nodes. Λ_i out of total Λ(1) left nodes have degree *i* and P_i out of total P(1) right nodes have degree *i*. A node of degree *i* has *i* sockets from which the *i* edges originates, so that in total there are Λ'(1) = P'(1) sockets on each side. Label the sockets on each side with the set E={1, 2, …, Λ'(1)} in some arbitrary but fixed way. Let π be a random permutation of E. Associate to π a bipartite graph by connecting the *i*th socket on the left to the π(i) socket on the right. (Note that this may potentially lead to nodes with multiple edges between them.) Let π run with uniform distribution over the set of permutation of E, then the resulting set of bipartite graphs is the *ensemble of LDPC (Λ, P)*. See Fig. 4.6.2. It remains to associate a code to every element of LDPC (Λ, P). We will do so by associating a parity check matrix to each graph. Because of possible parallel edges and since the encoding is done over GF(2), we define the parity-check matrix as a {0, 1} matrix in the form of $\mathbf{H}=[h_{ij}]_{P(1)\times\Lambda(1)}$, where h_{ij} = 1 iff the *i*th right node is connected to the *j*th left node an *odd number of times*.

It is sometimes more natural to work with matrices which contain the multiplicities of edges and therefore we define the extended parity-check matrix as $\tilde{\mathbf{H}}=[\tilde{h}_{ij}]$, where $\tilde{h}_{ij}=d$ iff the *i*th right node is connected to the *j*th left node by *d* edges. Clearly, $\mathbf{H}=\tilde{\mathbf{H}} \bmod 2$. We

sometimes refer to these two matrices as the *adjacency matrix* and the *extended adjacency matrix* of the bipartite graph.

Since (Λ, P) and (n, λ, ρ) contain equivalent information, we may freely switch between these two perspectives. We will often refer to the above ensemble as LDPC (n, λ, ρ). For the asymptotic analysis, it will be convenient to fix (λ, ρ) and to investigate the behavior of the ensemble LDPC (n, λ, ρ) as the block length $n\to\infty$. In such a case, it will happen that for some n, the corresponding (λ, ρ) will not be integral. We will then assume that the individual node distributions are rounded to the closest integer.

Note: A socket refers to a point on a node to which an edge may be attached. For example, we say that a variable node has d_v sockets, meaning d_v edges may be attached to that node. There will be a total of nd_v sockets on variable nodes and R(1)d_c sockets on parity-check nodes.

A particular pattern of edge connections can be described as a permutation π from variable-node sockets to check-node sochets. An individual edge is specified by the pair (i, π(i)), which indicates that the ith variable node socket is connected to the π(i)th check-node socket.

Selecting a random code from the $C^n(d_v, d_c)$ therefore amounts to randomly selecting a permutation on nd_v elements.

4.8.10.2 Upper-Bound on the Rate of LDPC Codes over BSC

In his Ph.D. thesis, Gallager derived an upper-bound on the rate required for reliable communication when using an LDPC code over the BSC. In [], the bound is generalized to a memoryless binary-input symmetric-output channel.

Theorem [Gallager]: Let a parity-check code $\mathcal{C}$ of length n and rate R be used on a BSC with crossover probability p, and let the codewords be used with equal probability. Suppose that the parity-check matrix of $\mathcal{C}$ has a property that all its rows have a constant weight d. Then a necessary condition for reliable communication is

$$R \le \frac{H(p_d) - H(p)}{H(p_d)} = 1 - \frac{1-C}{H(p_d)} \tag{4.68}$$

where $p_d = \frac{1+(1-2p)^d}{2}$, $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the binary entropy function and C is the channel capacity.

Proof: To prove the theorem, we show that if (4.68) is not satisfied, then the probability of ML decoding error is bounded away from 0 by an amount independent of n. Let $\mathbf{X} = (X_1, ..., X_n)$ be a transmitted codeword and $\mathbf{Y} = (Y_1, ..., Y_n)$ a received sequence. The average mutual information is

$$I(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}\,|\,\mathbf{Y}) = H(\mathbf{Y}) - H(\mathbf{Y}\,|\,\mathbf{X}) \tag{4.69}$$

From (4.69), we have

$$H(\mathbf{X}\,|\,\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{Y}) + H(\mathbf{Y}\,|\,\mathbf{X}) \tag{4.70}$$

Now $H(\mathbf{X})=nR$ and

$$H(\mathbf{Y}\,|\,\mathbf{X}) = \sum_{i=1}^{n} H(Y_i\,|\,Y_i,...Y_{i-1},\mathbf{X}) = nH(Y\,|\,X) = nH(p) \tag{4.71}$$

where $X=X_1$ and $Y=Y_1$.
Thus, to lower-bound $H(\mathbf{X}|\mathbf{Y})$, we need to upper-bound $H(Y|X)$. Consider specifying **Y** by first specifying the parities of the $n(1-R)$ parity checks and then specifying the Y_i in some set of nR linearly independent positions in the code. The probability that a parity-check is satisfied is the probability that an even number of errors have occurred within the coded bits involved in the parity-check equation, which is

$$\sum \binom{d}{i} p^i (1-p)^{d-i} = \frac{1+(1-2p)^d}{2} \triangleq p_d \tag{4.72}$$

The uncertainty associated with each parity-check is thus $H(p_d)$. Since the entropy of each information bit is at most 1 bit, we have

$$H(\mathbf{Y}) \le n(1-R)H(p_d) + nR \tag{4.73}$$

Substituting (4.71) and (4.73) into (4.70) yields

$$H(\mathbf{X}\,|\,\mathbf{Y}) \ge n[H(p) - (1-R)H(p_d)] \tag{4.74}$$

If $R > 1 - H(p)/H(p_d)$, then from (4.74) we have

$$\frac{1}{n} H(\mathbf{X}\,|\,\mathbf{Y}) > 0$$

which implies that the decoding error probability is bounded away from 0. Therefore the theorem follows.

4.8.11 More on Encoding

While LDPC codes have an efficient decoding algorithm, with complexity linear in the code length, the *encoding* efficiency is quadratic in the block length, since it requires multiplication by the generator matrix which is not sparse. This complexity is in contrast to the turbo code case, which has linear encode complexity. However, as we present here, it is possible to encode with a reasonable complexity, provided that some preprocessing is performed prior to encoding.

For QC-LDPC codes: [Li-Lin06], CCSDS
For RA-based LDPC codes: 802.16e

Another class of approaches is based on iterative use of the ***decoding*** algorithm in the encoding process. This could allow for the same hardware to be used for both encoding and decoding. These approaches are described in [?].

4.8.12 Applications of LDPC Codes

目前，LDPC 码已被多个通信标准建议为候选信道编码方案，广泛应用于包括深空通信（CCSDS 建议：基于欧氏几何构造的(8176, 7156)准循环 LDPC 码 for near earth applications, ARA for deep space applications)、4G 移动通信、IEEE 802.16e、DVB-S2、中国地面数字电视（DMB-T）、手机电视（STiMi）等在内的通信与广播领域中。下表简要总结了这些应用方案。

标准	码长	码率	构造方法	
CCSDS	(8176, 7156), 缩短(8160,7136)		Euclidean geometry, H = 2*16 array of 511*511 circulants	for near earth applications
CCSDS	信息位长度 k=1024, 4096, and 16384 bits	1/2, 2/3, 4/5	AR4JA LDPCC	for deep space applications
ETSI DVB S2	64800 and 16200	1/4 to 9/10	IRA codes	
IEEE 802.16e 802.11n	2304 bits	1/2, 2/3, 3/4	IRA-influenced QC LDPC	
STiMi	9216 bits	1/2, 3/4		
DMB-T	(7493, 3048), (7493, 4572), (7493, 6096)	0.4, 0.6, 0.8	Finite field LDPCC based on (59, 2, 58) shortened RS code over GF(2^7)	
4G				

■ *Accumulator-Based LDPC Codes in Standards:*

IRA codes and IRA-influenced codes are being considered for several communication standards. The ETSI DVB S2 standard for digital video broadcast specifies two IRA code families with block lengths 64800 and 16200. The code rates supported by this standard range from 1/4 to 9/10, and a wide range of spectral efficiencies is achieved by coupling these LDPC codes with QPSK, 8-PSK, 16-APSK, and 32-APSK modulation formats. A further level of protection is afforded by an outer BCH code. The IEEE standards bodies are also considering IRA-influenced QC LDPC codes for 802.11n (wireless local-area networks) and 802.16e (wireless metropolitan-area networks). Rather than employing a tailing-biting accumulator (which avoids weight-one columns), these standards have replaced the last block-column in (?) with a weight-three block-column and moved it to the first column, as displayed below. Encoding is facilitated by this matrix since the sum of all block-rows gives the block-row $(I_0, 0, ..., 0)$, so that encoding is initialized by summing all of the block-rows

of **H** and solving for first Q parity bits using the resulting block-row.

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{I}_0 & \mathbf{I}_0 & & & & \\ & \mathbf{I}_0 & \mathbf{I}_0 & & & \\ & & \mathbf{I}_0 & \ddots & & \\ \mathbf{I}_0 & & & \ddots & \mathbf{I}_0 & \\ & & & & \mathbf{I}_0 & \mathbf{I}_0 \\ \mathbf{I}_0 & & & & & \mathbf{I}_0 \end{bmatrix}$$

ARA codes are being considered by the Consultative Committee for Space Data Systems (CCSDS) for high data-rate bandwidth-efficient space links. Very low floors are required for this applications because the scientific data (e.g., images) being transmitted from space to the ground are typically in a compressed format.

References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[2] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol.47, no.2, pp.585-598, Feb. 2001.

[3] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol.47, no.2, pp.585-598, Feb. 2001.

[4] W. E. Ryan, "An Introduction to LDPC Codes," *CRC Handbook for Coding and Signal Processing for Recording Systems*, Ed., B. Vasic and E. Kurtas, CRC Press, 2004.

[5] D. J. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol.45, pp.399-431, Mar. 1999.

[6] J. M. F. Moura, J. Lu, and H. Zhang, "Structured LDPC codes with large girth," *IEEE Signal Processing Mag.,* vol. 21, pp. 42–55, Jan. 2004.

[7] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.

[8] Shu Lin and D. J. Costello, Jr. *Error Control Coding: Fundamentals and Applications*. 2nd ed. Prentice-Hall, 2004.

[9] S. J. Johnson, *Introducing Low-Density Parity-Check Codes*. Lecture notes, 2006.

[10] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. NJ: John Wiley & Sons, Inc. 2005.

[11] G. Liva, S. Song, L. Lan, Y. Zhang, S. Lin, and W. E. Ryan, "Design of LDPC codes: A survey and new results," 2006.

[12] E. Sharon, A. Ashikhmin, and S. Litsyn, "Analysis of low-density parity-check codes," IEEE Trans. Commun., vol.54, no.8, August 2006.

[13] E. Sharon, A. Ashikhmin, and S. Litsyn, "EXIT functions for binary input memoryless symmetric channels," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1207–1214, Jul. 2006.

[14] S. ten Brink, G. Kramer, and A. Ashikmin, "Design of low-density parity-check codes

for modulation and detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, Apr. 2004.

[15] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2657-2673, Nov. 2004.

[16] H. El Gamal and A. R. Hammons, "Analyzing the Turbo decoder using the Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 671-686, Feb. 2001.

[17] S.-Y. Chung, T. J. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[18] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[19] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, no.1, pp. 71-81, Jan. 2006.

[20] Y. Kou, S. Lin, and M. Fossorier, "Low density parity check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inform. Theory,* vol. 47, no.11, pp. 2711-2736, Nov. 2001.

[21] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Information Theory*, vol. 47, pp. 619-637, Feb. 2001.

[22] L. Lan, L. -Q Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach," *IEEE Trans. Inform. Theory,* vol.53, no.7, pp.2429-2458, July 2007.

[23] L. Lan, Y. Y. Tai, L. Chen, Y. Y. Tai, L. Lan, L. Zeng, S. Lin, and K. Abdel-Ghaffar, "Algebraic construction of quasi-cyclic LDPC codes for the AWGN and erasure channels," *IEEE Trans. Commun.*, vol.54, pp.1765-1774, no.10, Oct. 2006.

[24] L.-Q. Zeng, L. Lan, Y.Y. Tai, B. Zhou, S. Lin, and K. Abdel-Ghaffar, "Construction of nonbinary cyclic, quasi-cyclic and regular LDPC codes: A finite geometry approach," *IEEE Trans. Commun*., vol.56, no.3, March 2008.

[25] L.-Q. Zeng, L. Lan, Y.Y. Tai, S. Lin, and K. Abdel-Ghaffar, "Construction of nonbinary quasi-cyclic LDPC codes: a finite field approach," *IEEE Trans. Commun*., vol.56, no.4, April 2008.

Suggested Reading

[1] W. E. Ryan and Shu Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.

[2] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.

Homework

Compare the performance of the SPA used to decode the (8, 4, 4) RM code with the performance of optimal decoding, performed by exhaustively searching the 2^4 codewords, and the performance of hard-decision decoding.

Appendix

■ **<u>式（4.16）的证明:</u>** By applying Bayes' rule, we have

$$
\begin{aligned}
q_{ij}(b) &= P\left(c_i = b \mid y_i, \left\{s_{j'} = 0, j' \in \mathcal{N}_v(i) \setminus j\right\}\right) \\
&= \frac{P\left(c_i = b, \left\{s_{j'} = 0, j' \in \mathcal{N}_v(i) \setminus j\right\} \middle| y_i\right)}{P\left(\left\{s_{j'} = 0, j' \in \mathcal{N}_v(i) \setminus j\right\} \middle| y_i\right)} \\
&= \frac{1}{P\left(\left\{s_{j'} = 0, j' \in \mathcal{N}_v(i) \setminus j\right\} \middle| y_i\right)} P\left(c_i = b \middle| y_i\right) P\left(\left\{s_{j'} = 0, j' \in \mathcal{N}_v(i) \setminus j\right\} \middle| c_i = b, y_i\right)
\end{aligned}
$$

Under the assumption that the checks are independent, the joint probability on checks can be factored, so that

$$
\begin{aligned}
q_{ij}(b) &= \frac{1}{P\left(\left\{s_{j'} = 0, j' \in \mathcal{N}_v(i) \setminus j\right\} \middle| y_i\right)} P\left(c_i = b \middle| y_i\right) \prod_{j' \in \mathcal{N}_v(i) \setminus j} P\left(s_{j'} = 0 \middle| c_i = b, y_i\right) \\
&= \frac{P\left(c_i = b \middle| y_i\right) \prod_{j' \in \mathcal{N}_v(i) \setminus j} P\left(s_{j'} = 0 \middle| c_i = b, y_i\right)}{\sum_{b'} P\left(c_i = b' \middle| y_i\right) \prod_{j' \in \mathcal{N}_v(i) \setminus j} P\left(s_{j'} = 0 \middle| c_i = b, y_i\right)}
\end{aligned}
$$

Clearly, 前式中的第一个因子只是一个归一化常量，我们将其记为 α_{ij}，它保证 $q_{ij}(b)$ is a probability mass function with respect to b.

■ **<u>式（4.15）的推导:</u>**

$$
\begin{aligned}
\sigma_{ji}(b) &= P\left(s_j = 0 \mid c_i = b, \mathbf{y}\right) \\
&= \sum_{c_{i'}: i' \in \mathcal{N}_c(j) \setminus i} P\left(s_j = 0, \{c_{i'}, i' \in \mathcal{N}_c(j) \setminus i\} \mid c_i = b, \mathbf{y}\right) \\
&= \sum_{c_{i'}: i' \in \mathcal{N}_c(j) \setminus i} P\left(s_j = 0 \mid c_i = b, \{c_{i'}, i' \in \mathcal{N}_c(j) \setminus i\}, \mathbf{y}\right) P\left(\{c_{i'}, i' \in \mathcal{N}_c(j) \setminus i\} \mid c_i = b, \mathbf{y}\right)
\end{aligned}
$$

Under the assumption that the bits $\{c_{i'}, i' \in \mathcal{N}_c(j)\}$ are independent – which is true only if there are no cycles in the graph – the above equation can be written as

$$
\sigma_{ji}(b) = \sum_{c_{i'}: i' \in \mathcal{N}_c(j) \setminus i} \left(P\left(s_j = 0 \mid c_i = b, \{c_{i'}, i' \in \mathcal{N}_c(j) \setminus i\}\right) \prod_{k \in \mathcal{N}_c(j) \setminus i} P\left(c_k = b' \mid \mathbf{y}\right) \right)
$$

Note: 式（4.15）的运算也可在 trellis 图上完成。By creating a trellis with more states, the LDPC decoding algorithm can be extended to linear codes over arbitrary fields.

■ <u>基于 Trellis 的校验节点消息更新算法：</u>

一个每行有 d_i 个 $1\,(i=1,...,m)$ 的 LDPC 码的校验矩阵 $\mathbf{H}_{m\times n}$ 可以用 m 个网格图表示，其中网格图分别对应于 (d_i, d_i-1) SPC 码。如下图所示：

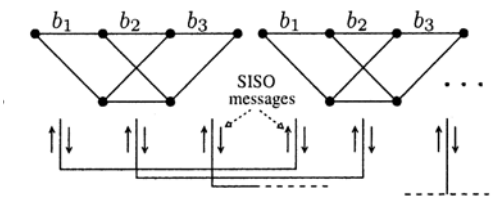


图 4A　奇偶校验码的 Trellis 图表示及 BCJR 算法的 SISO 信息传递

我们可以在 Trellis 图上计算输出外信息。令 $\alpha_k(x)$ 表示 Trellis 图上 k 时刻前向节点状态为 x 的概率，$\beta_k(x)$ 表示 k 时刻后向节点状态为 x 的概率，$\gamma_k(x)$ 表示 k 时刻分支(网格图的边)上的输入为 x 的先验概率，$x=0,1$。则有

$$\alpha_k(1)=\alpha_{k-1}(1)\times\gamma_k(0)+\alpha_{k-1}(0)\times\gamma_k(1),$$
$$\alpha_k(0)=\alpha_{k-1}(1)\times\gamma_k(1)+\alpha_{k-1}(0)\times\gamma_k(0)$$

$k=0,1,...,L$，其中 L 为网格图的长度。定义 $L(\alpha_k)=\ln(\alpha_k(1)/\alpha_k(0))$，$L(\beta_k)=\ln(\beta_k(1)/\beta_k(0))$，$L(\gamma_k)=\ln(\gamma_k(1)/\gamma_k(0))$。上面两式相除并取对数，可得

$$L(\alpha_k)=q\left(L(\alpha_{k-1}),L(\gamma_k)\right)$$

其中

$$q(x,y)=\ln(e^x+e^y)-\ln(e^{x+y}+1) \tag{A.1}$$

同理可得

$$L(\beta_k)=q\left(L(\beta_{k+1}),L(\gamma_{k+1})\right)$$

从而关于 k 时刻分支输入 b_k 的 LLR 为

$$L(b_k\,|\,\mathbf{y})=\ln\frac{P(b_k=1\,|\,\mathbf{y})}{P(b_k=0\,|\,\mathbf{y})}$$

$$=\ln\frac{\alpha_{k-1}(0)\gamma_k(1)\beta_k(1)+\alpha_{k-1}(1)\gamma_k(1)\beta_k(0)}{\alpha_{k-1}(0)\gamma_k(0)\beta_k(0)+\alpha_{k-1}(1)\gamma_k(0)\beta_k(1)}$$

$$= \ln \frac{\frac{\gamma_k(1)}{\gamma_k(0)}\left(\frac{\alpha_{k-1}(1)}{\alpha_{k-1}(0)} + \frac{\beta_k(1)}{\beta_k(0)}\right)}{1 + \frac{\alpha_{k-1}(1)}{\alpha_{k-1}(0)}\frac{\beta_k(1)}{\beta_k(0)}}$$

$$= \ln \frac{\exp\left(L(\alpha_{k-1})\right) + \exp\left(L(\beta_k)\right)}{1 + \exp\left(L(\alpha_{k-1}) + L(\beta_k)\right)} + L(\gamma_k)$$

$$= q\left(L(\alpha_{k-1}), L(\beta_k)\right) + L(\gamma_k)$$

最后计算的外信息为：

$$L_e(b_k) = q\left(L(\alpha_{k-1}), L(\beta_k)\right)$$

$L_e(b_k)$ 即为图 4A 中的 SISO 信息，在分量译码器间进行传递，用于 Turbo 迭代译码。为降低计算复杂度，式(A.1)可简化为

$$q(x, y) \approx q_0(x, y) = \max(x, y) - \max(x + y, 0)$$

■ <u>重复码的软判决译码：</u>

假定用 BPSK 调制：0 -->(+1), 1--> (-1)

则 LLR 为:

$$L(u) = \ln \frac{P(\mathbf{y} \mid u = 0)}{P(\mathbf{y} \mid u = 1)} = L_c \sum_{k=0}^{n-1} y_k$$

where $L_c = 2/\sigma^2$. Then we have

$$\hat{u} = \begin{cases} 0, & \text{if } \sum_{k=0}^{n-1} y_k > 0 \\ 1, & \text{otherwise} \end{cases}$$

and

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

It is the same as for uncoded transmission.