Chapter 5 Low-Density Parity-Check Codes

- 5.1 Introduction of LDPC Codes
- 5.2 Tanner Graph Representation
- 5.3 Encoding of LDPC Code
- 5.4 Belief Propagation Decoding
- 5.5 The Sum-Product Algorithm
§ 5.1 Introduction of LDPC Code

• Introduction
  – Proposed by Robert Gallager in 1962 [1].
  – It was overlooked for over three decades until 1995, it was rediscovered by David Mackay [2].
  – It is a linear block code defined by its sparse parity-check matrix which is inherently good for the belief propagation decoding.
  – It can well approach the Shannon capacity with a decoding complexity that is quadratic in the dimension of the code.
  – Its potential applications include wireless communications and storage devices.

§ 5.1 Introduction of LDPC Codes

- LDPC code: A linear block code whose parity-check matrix \( H \) has sparse non-zero elements. For a binary LDPC code, its matrix \( H \) has sparse 1s.

- Column weight \( (w_c) \): Number of 1s in a column of \( H \).
Row weight \( (w_r) \): Number of 1s in a row of \( H \).

- Regular LDPC codes: Each column of \( H \) has the same column weight, and each row of the \( H \) has the same row weight. It is normally denoted as a \( (w_c, w_r, N) \) LDPC code, where \( N \) is the codeword length.

- Irregular LDPC codes: The parity-check matrix has varying column weights and row weights.

- In general, irregular codes have better performance than regular codes. But irregular codes are more difficult to implement.
§ 5.1 Introduction of LDPC Codes

Example 5.1 A regular LDPC code has a parity-check matrix of

\[
H = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}
\]

\(w_c = 3, w_r = 6, M = 5, N = 10.\)

\(M\): Number of parity-check equations. The above matrix implies

\[z_1 : c_1 + c_2 + c_3 + c_6 + c_7 + c_{10} = 0\]
\[z_2 : c_1 + c_3 + c_5 + c_6 + c_8 + c_9 = 0\]
\[z_3 : c_3 + c_4 + c_5 + c_7 + c_9 + c_{10} = 0\]
\[z_4 : c_2 + c_4 + c_5 + c_6 + c_8 + c_{10} = 0\]
\[z_5 : c_1 + c_2 + c_4 + c_7 + c_8 + c_9 = 0\]

- If all rows of \(H\) are independent, \(M = N - K\). Otherwise \(M > N - K\).
- Uniform row weight requires \(\frac{W_r}{N} = \frac{W_c}{M}\). If \(M = N - K\), then the code rate is \(R = \frac{K}{N} = 1 - \frac{M}{N} = 1 - \frac{W_c}{W_r}\). If \(M > N - K\), \(R > 1 - \frac{W_c}{W_r}\).
§ 5.1 Introduction of LDPC Codes

Example 5.2 Construct a (3, 4, 20) regular LDPC code.

Given a based matrix $A$ as:

$$A = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}$$

Let $\pi_i(A)$ denote a random permutation function that permutes the columns of $A$. 
§ 5.1 Introduction of LDPC Codes

The parity-check matrix of the $(3, 4, 20)$ regular LDPC code can be generated by

$$H = \begin{bmatrix}
A \\
\pi(A) \\
\pi_2(A)
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Since there are 13 independent rows, the code's dimension is $K = 20 - 13 = 7$. The rate of the code is $R = 0.35 > 1 - \frac{W_c}{W_r}$.

**Q:** Why is random permutation of columns of $A$ necessary?
§ 5.2 Tanner Graph Representation

- The parity-check matrix $H[h_{mn}]$ can be represented as a Tanner graph.
- The parity-check matrix $H$ of Example 5.1 can be shown as:

- The Tanner graph has two sets of nodes, the check nodes ($z_m$) and the bit nodes ($c_n$). There is a connection between $z_m$ and $c_n$ if $h_{mn} = 1$.
- Belief propagation decoding of a LDPC code is performed based on a Tanner graph: propagating soft information between the check nodes and the bit nodes through the established connections.
§ 5.2 Tanner Graph Representation

- $N_m = \{n : h_{mn} = 1\}$ — The set of bits that participate the check $z_m$.
  E.g., $N_1 = \{1, 2, 3, 6, 7, 10\}$, $N_3 = \{3, 4, 5, 7, 9, 10\}$.

- $N_{m \setminus n}$ — The set of bits except $c_n$ that participate check $z_m$.
  E.g., $N_{1 \setminus 3} = \{1, 2, 6, 7, 10\}$.

- $M_n = \{m : h_{mn} = 1\}$ — The set of checks in which bit $c_n$ is involved.
  E.g., $M_1 = \{1, 2, 5\}$, $M_{10} = \{1, 3, 4\}$.

- $M_{n \setminus m}$ — The set of checks except check $z_m$ in which bit $c_n$ is involved.
  E.g., $M_{1 \setminus 2} = \{1, 5\}$.
§ 5.2 Tanner Graph Representation

- For a regular LDPC code, every check node is connected to $|N_m|$ bit nodes where $|N_m| = w_r$, and every bit node is connected to $|M_n|$ check nodes where $|M_n| = w_c$.

- Girth: the shortest cycle in a Tanner graph and it is $\geq 4$. It is desirable to avoid a LDPC code whose Tanner graph has a girth of 4 as it would degrade the decoding performance. (In the above Tanner graph, the highlighted cycle is of length 4 and hence the LDPC code has a girth of 4.)
§ 5.3 Encoding of LDPC Codes

- By performing Gaussian elimination, a parity-check matrix $H$ can be transformed into

$$H = \left[ I_M \mid P \right]$$

where $I_M$ is a $M \times M$ identity matrix.

- Its corresponding generator matrix $G$ can be written as:

$$G = \left[ P^T \mid I_K \right]$$

where $I_K$ is a $K \times K$ identity matrix.

- Encoding of a $K$ dimensional message vector $\vec{m} = [m_1, m_2, \ldots, m_K]$ is done by

$$\vec{c} = \vec{m} \cdot G$$

$$= [c_1, c_2, \ldots, c_{N-K}, c_{N-K+1}, \ldots, c_N]$$

$$= [p_1, p_2, \ldots, p_{N-K}, m_1, \ldots, m_K].$$
§ 5.3 Encoding of LDPC Codes

**Example 5.3** By performing Gaussian elimination on the matrix \( H \) of *Example 5.1*, we have

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
\end{bmatrix}
\]

Hence, the generator matrix \( G \) is

\[
G = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

If the message vector is \( \bar{m} = [1 \ 0 \ 1 \ 0 \ 1] \), the codeword \( \bar{c} \) is generated as

\[
\bar{c} = \bar{m} \cdot G = [0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1]
\]
Belief Propagation (BP) decoding is performed based on the Tanner graph of the LDPC code.

Optimal decoding estimates a codeword by maximizing

\[ \Pr[\overline{c} \mid z_m = 0, \forall m] \]

Its complexity is \( O(2^K) \).

Suboptimal decoding estimates individual coded bit \( c_n \) by maximizing

\[ \Pr[c_n = \theta \mid z_m = 0, m \in M_n], \theta \in \{0,1\} \]

Its complexity is \( O(K^2) \).

BP decoding is a sub-optimal decoding algorithm.
BP decoding is to update the following two probabilities iteratively.

1. The probability of bit $c_n = \theta$ ($\theta \in \{0, 1\}$) conditioned on all its associated checks except $z_m$ are satisfied, i.e.,

$$q_{mn}(\theta) = \Pr[c_n = \theta \mid z_m' = 0, m' \in M_n \setminus m]$$  \hspace{1cm} (1)

2. The probability of check $z_m$ is satisfied conditioned on bit $c_n = \theta$, i.e.,

$$r_{mn}(\theta) = \Pr[z_m = 0 \mid c_n = \theta]$$  \hspace{1cm} (2)
§ 5.4 Belief Propagation Decoding

- Since there are $N$ coded bits and $M$ checks, $q_{mn}(\theta)$ and $r_{mn}(\theta)$ should be accommodated in matrices $Q$ and $R$, respectively. $Q$ and $R$ are of size $2M \times N$.

$$Q = \begin{bmatrix} q_{11}(0) & q_{12}(0) & \ldots & \ldots & q_{1N}(0) \\ q_{11}(1) & q_{12}(1) & \ldots & \ldots & q_{1N}(1) \\ q_{21}(0) & q_{22}(0) & \ldots & \ldots & q_{2N}(0) \\ q_{21}(1) & q_{22}(1) & \ldots & \ldots & q_{2N}(1) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ q_{M1}(0) & q_{M2}(0) & \ldots & \ldots & q_{MN}(0) \\ q_{M1}(1) & q_{M2}(1) & \ldots & \ldots & q_{MN}(1) \end{bmatrix} \quad R = \begin{bmatrix} r_{11}(0) & r_{12}(0) & \ldots & \ldots & r_{1N}(0) \\ r_{11}(1) & r_{12}(1) & \ldots & \ldots & r_{1N}(1) \\ r_{21}(0) & r_{22}(0) & \ldots & \ldots & r_{2N}(0) \\ r_{21}(1) & r_{22}(1) & \ldots & \ldots & r_{2N}(1) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ r_{M1}(0) & r_{M2}(0) & \ldots & \ldots & r_{MN}(0) \\ r_{M1}(1) & r_{M2}(1) & \ldots & \ldots & r_{MN}(1) \end{bmatrix}$$

- BP decoding iterations $Q \xleftarrow{\text{Horizontal update}} \xrightarrow{\text{Vertical update}} R$.

- After a number of iterations, the decision on all the bits $c_n$ is made based on $Q$ by

$$q_n(\theta) = \Pr[c_n = \theta \mid z_m = 0, m \in M_n] \quad (3)$$
§ 5.4 Belief Propagation Decoding

- **Initialization:**

Given a received symbol vector \( \bar{y} = (y_1, y_2, ..., y_N) \), one could obtain the channel observations for all the coded bits as

\[
\begin{align*}
  f_1(0) &= \Pr(y_1|c_1 = 0) \\
  f_2(0) &= \Pr(y_2|c_2 = 0) \\
  f_N(0) &= \Pr(y_N|c_N = 0) \\
  f_1(1) &= \Pr(y_1|c_1 = 1) \\
  f_2(1) &= \Pr(y_2|c_2 = 1) \\
  f_N(1) &= \Pr(y_N|c_N = 1)
\end{align*}
\]

Before decoding, we assume \( \Pr(c_n = 0) = \Pr(c_n = 1) = \frac{1}{2}, \ \forall n \).

Hence,

\[\Pr(c_n = \theta|y_n) = \Pr(y_n|c_n = \theta) = f_n(\theta), \ \theta \in \{0, 1\}, \forall n.\]

Matrix \( Q \) is initialized by

\[q_{mn}(\theta) = f_n(\theta) \cdot h_{mn}, \ \forall m, n.\]


5.4 Belief Propagation Decoding

- **Horizontal update:** update \( R(r_{mn}(\theta)) \) by \( Q(q_{mn}(\theta)) \).

\[
\begin{align*}
  r_{mn}(\theta) &= \Pr[z_m = 0 | c_n = \theta] \\
  &= \sum_{\{c_{n'}, \Sigma \theta_{n'} = \theta\}} \Pr[z_m = 0, \{c_{n'} = \theta_{n'}, n' \in N_m \setminus n\} | c_n = \theta] \\
  &= \sum_{\{c_{n'}, \Sigma \theta_{n'} = \theta\}} \Pr[z_m = 0 | \{c_{n'} = \theta_{n'}, n' \in N_m \setminus n\}, c_n = \theta] \cdot \Pr[\{c_{n'} = \theta_{n'}, n' \in N_m \setminus n\}] \\
  &= \sum_{\{c_{n'}, \Sigma \theta_{n'} = \theta\}} \Pr[z_m = 0 | \{c_{n'} = \theta_{n'}, n' \in N_m \setminus n\}, c_n = \theta] \cdot \prod_{n' \in N_m \setminus n} \Pr(c_{n'} = \theta_{n'}) \\
  &= \sum_{\{c_{n'}, \Sigma \theta_{n'} = \theta\}} \prod_{n' \in N_m \setminus n} \Pr(c_{n'} = \theta_{n'}) \quad q_{mn'}(\theta)
\end{align*}
\]

**Remark:** In (4), it is assumed that all codes bits \( c_n \) are independent. Moreover, for \( \{c_{n'}, n' \in N_m \setminus n\} \), if \( \Sigma \theta_n' = \theta \), \( \Pr[z_m = 0 | \{c_{n'} = \theta_{n'}, n' \in N_m \setminus n\}, c_n = \theta] = 1 \). Otherwise, \( \Pr[z_m = 0 | \{c_{n'} = \theta_{n'}, n' \in N_m \setminus n\}, c_n = \theta] = 0 \).
§ 5.4 Belief Propagation Decoding

- **Horizontal update:** update $R$ by $Q$.

\[
 r_{mn}(\theta) = \sum_{\theta = \sum_{n' \in N_m \setminus n} \theta_{n'}} \prod_{n' \in N_m \setminus n} q_{mn'}(\theta)
\]

With $c_n = \theta$, $\theta = \sum \theta_{n'}$ for $n' \in N_m \setminus n$ ensures check $z_m$ is satisfied, i.e.,

\[
 z_m = \sum_{n \in N_m} c_n = \theta + \sum_{n' \in N_m \setminus n} \theta_{n'} = 0
\]

- **Example 5.4** For the LDPC code of Example 5.1, if we want to update $r_{11}(1) = \Pr[z_1 = 0 | c_1 = 1]$, we need the remaining bits of $z_1$ satisfy $c_2 + c_3 + c_6 + c_7 + c_{10} = 1$. Bits $c_2, c_3, c_6, c_7, c_{10}$ have the following 16 permutations:

  10000, 01000, 00100, 00010, 00001, 11100, 01110, 00111,
  11001, 11010, 01101, 10101, 10011, 01011, 10110, 11111.

Hence, $r_{11}(1)$ is updated by summing the following 16 products.

\[
 q_{12}(1)q_{13}(0)q_{16}(0)q_{17}(0)q_{10}(0) : 16
\]

\[
 q_{12}(1)q_{13}(1)q_{16}(1)q_{17}(1)q_{10}(1)
\]
§ 5.4 Belief Propagation Decoding

- **Horizontal update:** update $R$ by $Q$

$$r_{mn}(\theta) = \sum_{\theta = \sum_{n' \in N_m \setminus n} \theta_{n'} \sum_{n' \in N_m \setminus n} q_{mn'}(\theta)} \prod_{n' \in N_m \setminus n} q_{mn'}(\theta)$$

- Tanner graph reflection.
  - The update of $r_{11}(1)$ of **Example 5.4** can be seen as

$$r_{11}(1) = \Pr[z_1 = 0|c_1 = 1].$$

The red edges provide information to calculate probability of the black edge.
§ 5.4 Belief Propagation Decoding

- **Vertical update**: update $Q(q_{mn}(\theta))$ by $R(r_{mn}(\theta))$.

$$q_{mn}(\theta) = \Pr[c_n = \theta | z_{m'} = 0, m' \in M_n \setminus m]$$

$$= \frac{\Pr[z_{m'} = 0, m' \in M_n \setminus m | c_n = \theta] \cdot \Pr[c_n = \theta]}{\Pr[z_{m'} = 0, m' \in M_n \setminus m]}$$

$$= \frac{\prod_{m' \in M_n \setminus m} \Pr[z_{m'} = 0 | c_n = \theta] \cdot \Pr[c_n = \theta]}{\Pr[z_{m'} = 0, m' \in M_n \setminus m]} \cdot r_{m'n(\theta)}$$

$$= \alpha_{mn} \cdot \prod_{m' \in M_n \setminus m} \Pr[z_{m'} = 0 | c_n = \theta] \cdot \Pr[c_n = \theta] \cdot f_{n}(\theta)$$

- In (5), it is assumed that all cheeks are independent.
- In (6), $\alpha_{mn}$ is a normalization factor that ensures $q_{mn}(0) + q_{mn}(1) = 1$. 
§ 5.4 Belief Propagation Decoding

- **Vertical update:** update $Q$ by $R$.

\[ q_{mn}(\theta) = \alpha_{mn} \cdot f_n(\theta) \cdot \prod_{m' \in M_n \setminus m} r_{m'n}(\theta) \]

$\alpha_{mn}$ is a normalization factor that ensures $q_{mn}(0) + q_{mn}(1) = 1$, i.e.,

\[ \alpha_{mn} = \left[ \sum_{\theta \in \{0,1\}} f_n(\theta) \cdot \prod_{m' \in M_n \setminus m} r_{m'n}(\theta) \right]^{-1} \]

- **Example 5.5** (Continue from **Example 5.4**), if we want to update

\[ q_{11}(\theta) = \Pr[c_1 = \theta \mid z_{m'} = 0, m' \in M_{1 \setminus 1}] \]

we need to calculate

\[ q_{11}(0) = \alpha_{11} \cdot f_{1}(0) \cdot (r_{21}(0) \times r_{51}(0)) \]
\[ q_{11}(1) = \alpha_{11} \cdot f_{1}(1) \cdot (r_{21}(1) \times r_{51}(1)) \]
§ 5.4 Belief Propagation Decoding

- **Vertical update:** update $Q$ by $R$

\[
q_{mn}(\theta) = \alpha_{mn} \cdot f_n(\theta) \cdot \prod_{m' \in Mn \setminus m} r_{m'n}(\theta)
\]

- Tanner graph reflection.
  - The update of $q_{11}(\theta)$ of **Example 5.5** can be seen as

\[
q_{11}(1) = f_1(1) \cdot r_{21}(1) \cdot r_{51}(1)
\]

Again, the red edges provide information to update probability of the black edge.
\section*{5.4 Belief Propagation Decoding}

- After each horizontal-vertical iteration, we can calculate \( q_n(\theta) \) of (3) by
  \[
  q_n(\theta) = \alpha_n \cdot f_n(\theta) \cdot \prod_{m \in M_n} r_{mn}(\theta)
  \]
  \( \alpha_n \) is a normalization factor that ensures \( q_n(0) + q_n(1) = 1 \).
  \[
  \alpha_n = \left[ \sum_{\theta \in \{0,1\}} f_n(\theta) \cdot \prod_{m \in M_n} r_{mn}(\theta) \right]^{-1}
  \]

- Decision on bit \( c_n \)
  \[
  \begin{cases} 
  c_n = 0, & \text{if } q_n(0) > q_n(1) \\
  c_n = 1, & \text{if } q_n(0) < q_n(1) 
  \end{cases}
  \]

- After decisions are made on all the coded bits, we can obtain an estimated codeword \( \hat{c} \). The iteration will be terminated if \( \hat{c} \) is a valid codeword, i.e., \( \hat{c} \cdot H^T = 0 \). Otherwise, the iterative horizontal-vertical updates continue until \( \hat{c} \cdot H^T = 0 \) is satisfied, or the designed maximal iteration number is reached.

- The BP decoding algorithm is also called the \textbf{Sum-Product algorithm}. 
§ 5.4 Belief Propagation Decoding

Why low density of $H$ is important for BP decoding?

- The Horizontal update computation of $\prod_{n' \in N_{m \setminus n}} q_{mn'}(\theta)$ assumes that all the coded bits are independent. Similarly, the Vertical update of $\prod_{m' \in M_{n \setminus m}} r_{m'n}(\theta)$ assumes all checks are independent.

- However, once cycles exist in the Tanner graph, the independence will disappear. For example, when two coded bits are involved in the same two checks, a cycle of length 4 will exist in the Tanner graph.

- A low density $H$ inherits less cycles especially the cycles of length 4. The BP decoding would favour this type of code — low-density parity-check codes.
§ 5.4 Belief Propagation Decoding

Why low density of $H$ is important for BP decoding?

**Example 5.6** Let us look at BP decoding of the LDPC code of Example 5.1.

- By examining the Tanner graph, we can see coded bits $c_1$ and $c_2$ are involved in both checks $z_1$ and $z_5$, yielding a cycle of length 4.

- Horizontal update:

  - $r_{11}(\theta) \leftarrow q_{12}(\theta), q_{13}(\theta), q_{16}(\theta), q_{17}(\theta), q_{1,10}(\theta)$
  - $r_{12}(\theta) \leftarrow q_{11}(\theta), q_{13}(\theta), q_{16}(\theta), q_{17}(\theta), q_{1,10}(\theta)$
  - $\vdots$
  - $r_{51}(\theta) \leftarrow q_{52}(\theta), q_{54}(\theta), q_{57}(\theta), q_{58}(\theta), q_{59}(\theta)$
  - $r_{52}(\theta) \leftarrow q_{51}(\theta), q_{54}(\theta), q_{57}(\theta), q_{58}(\theta), q_{59}(\theta)$

- Vertical update:

  - $q_{11}(\theta) \leftarrow r_{21}(\theta), r_{51}(\theta)$
  - $q_{12}(\theta) \leftarrow r_{42}(\theta), r_{52}(\theta)$

- Observations:
  1) $\overline{1} \rightarrow \overline{2}$ process, bits $c_1$ and $c_2$ start to correlate.
  2) $\overline{1} \rightarrow \overline{2} \rightarrow \overline{3}$ process, part of the information used to update $r_{mn}(\theta)$ comes for $c_n$ itself.
Example 5.7 (Continue from Example 5.3). If the LDPC codeword
\( \overline{c} = [0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1] \) is transmitted to a memoryless channel, with the received symbol vector \( \overline{y} \), we obtain the channel observation matrix \( F \) as

\[
F = \begin{bmatrix}
0.78 & 0.84 & 0.81 & 0.52 & 0.45 & 0.13 & 0.82 & 0.21 & 0.75 & 0.24 \\
0.22 & 0.16 & 0.19 & 0.48 & 0.55 & 0.87 & 0.18 & 0.79 & 0.25 & 0.76 \\
\end{bmatrix}
\]

Matrix \( Q \) is initialized as:

\[
Q = \begin{bmatrix}
0.78 & 0.84 & 0.81 & 0 & 0 & 0.13 & 0.82 & 0 & 0 & 0.24 \\
0.22 & 0.16 & 0.19 & 0 & 0 & 0.87 & 0.18 & 0 & 0 & 0.76 \\
0.78 & 0 & 0.81 & 0 & 0.45 & 0.13 & 0 & 0.21 & 0.75 & 0 \\
0.22 & 0 & 0.19 & 0 & 0.55 & 0.87 & 0 & 0.79 & 0.25 & 0 \\
0 & 0 & 0 & 0.81 & 0.52 & 0.45 & 0 & 0.82 & 0 & 0.75 \\
0 & 0 & 0 & 0.19 & 0.48 & 0.55 & 0 & 0.18 & 0 & 0.25 \\
0 & 0.84 & 0 & 0.52 & 0.45 & 0.13 & 0 & 0.21 & 0 & 0.24 \\
0 & 0.16 & 0 & 0.48 & 0.55 & 0.87 & 0 & 0.79 & 0 & 0.76 \\
0.78 & 0.84 & 0 & 0.52 & 0 & 0.82 & 0.21 & 0.75 & 0 & 0 \\
0.22 & 0.16 & 0 & 0.48 & 0 & 0.18 & 0.79 & 0.25 & 0 & 0 \\
\end{bmatrix}
\]
§ 5.4 Belief Propagation Decoding

After the 1st Horizontal-Vertical iteration, we have

\[
R = \begin{bmatrix}
0.551914 & 0.542753 & 0.546890 & 0 & 0 & 0.460714 & 0.545425 & 0 & 0 & 0.444092 \\
0.448086 & 0.457247 & 0.453110 & 0 & 0 & 0.539286 & 0.454575 & 0 & 0 & 0.555908 \\
0.493347 & 0 & 0.493991 & 0 & 0.537255 & 0.505034 & 0 & 0.506423 & 0.493347 & 0 \\
0.506653 & 0 & 0.506009 & 0 & 0.462745 & 0.494966 & 0 & 0.493577 & 0.507451 & 0 \\
0 & 0 & 0.500333 & 0.505158 & 0.497937 & 0 & 0.500322 & 0 & 0.500413 & 0.499603 \\
0 & 0 & 0.499667 & 0.494842 & 0.502063 & 0 & 0.499678 & 0 & 0.499587 & 0.500397 \\
0 & 0.500446 & 0 & 0.507588 & 0.496965 & 0.499590 & 0 & 0.499477 & 0 & 0.499416 \\
0 & 0.499554 & 0 & 0.492412 & 0.503035 & 0.500410 & 0 & 0.500523 & 0 & 0.500584 \\
0.497476 & 0.497921 & 0 & 0.464662 & 0 & 0 & 0.497791 & 0.502437 & 0.497173 & 0 \\
0.502524 & 0.502079 & 0 & 0.535338 & 0 & 0 & 0.502209 & 0.497563 & 0.502827 & 0
\end{bmatrix}
\]
Hence, the *a posteriori* probability matrix $Q'$ is:

$$Q' = \begin{bmatrix}
0.773636 & 0.839121 & 0.806481 & 0 & 0 & 0.132106 & 0.818884 & 0 & 0 & 0.239285 \\
0.226364 & 0.160879 & 0.193519 & 0 & 0 & 0.867894 & 0.181116 & 0 & 0 & 0.760715 \\
0.812140 & 0 & 0.837461 & 0 & 0.444958 & 0.113039 & 0 & 0.211273 & 0.748185 & 0 \\
0.187860 & 0 & 0.162539 & 0 & 0.555042 & 0.886961 & 0 & 0.788727 & 0.251815 & 0 \\
0 & 0 & 0.833978 & 0.492203 & 0.484126 & 0 & 0.844187 & 0 & 0.742212 & 0.201076 \\
0 & 0 & 0.166022 & 0.507797 & 0.515874 & 0 & 0.155813 & 0 & 0.257788 & 0.798924 \\
0 & 0.860727 & 0 & 0.489773 & 0.485097 & 0.115241 & 0 & 0.215940 & 0 & 0.201196 \\
0 & 0.139273 & 0 & 0.5110227 & 0.514903 & 0.884759 & 0 & 0.784060 & 0 & 0.798804 \\
0.809608 & 0.861934 & 0 & 0.532711 & 0 & 0 & 0.845514 & 0.213942 & 0.744684 & 0 \\
0.190392 & 0.138066 & 0 & 0.467289 & 0 & 0 & 0 & 0.154486 & 0.786058 & 0.255316 \\
\end{bmatrix}$$

The estimated codeword is $\hat{c} = [0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1]$. It does not satisfy $\hat{c} \cdot H^T = 0$ and the iteration continues...
5.4 Belief Propagation Decoding

After the 3rd Horizontal-Vertical iteration, we have

\[
R = \begin{bmatrix}
0.549960 & 0.540086 & 0.544369 & 0 & 0 & 0.463092 & 0.542650 & 0 & 0 & 0.447890 \\
0.450040 & 0.459914 & 0.455631 & 0 & 0 & 0.536908 & 0.457350 & 0 & 0 & 0.552110 \\
0.493114 & 0 & 0.493650 & 0 & 0.545393 & 0.505532 & 0 & 0.507453 & 0.491301 & 0 \\
0.506886 & 0 & 0.506350 & 0 & 0.454607 & 0.494468 & 0 & 0.492547 & 0.508699 & 0 \\
0 & 0 & 0 & 0.499989 & 0.500176 & 0.502649 & 0 & 0.499989 & 0 & 0.499985 & 0.500012 \\
0 & 0 & 0 & 0.500011 & 0.499824 & 0.497351 & 0 & 0.500011 & 0 & 0.500015 & 0.499988 \\
0 & 0.499975 & 0 & 0.500415 & 0.503915 & 0.500023 & 0 & 0.500032 & 0 & 0.500030 \\
0 & 0.500025 & 0 & 0.499585 & 0.496085 & 0.49977 & 0 & 0.499968 & 0 & 0.499970 \\
0.496595 & 0.497094 & 0 & 0.457904 & 0 & 0 & 0.496955 & 0.503693 & 0.495668 & 0 \\
0.503405 & 0.502906 & 0 & 0.542096 & 0 & 0 & 0.503045 & 0.496307 & 0.504332 & 0
\end{bmatrix}
\]
§ 5.4 Belief Propagation Decoding

The \textit{a posteriori} probability matrix $Q'$ becomes:

$$Q' = \begin{bmatrix}
0.772854 & 0.838418 & 0.806053 & 0 & 0 & 0.132534 & 0.818189 & 0 & 0 & 0.240031 \\
0.227146 & 0.161582 & 0.193947 & 0 & 0 & 0.867466 & 0.181811 & 0 & 0 & 0.759969 \\
0.810391 & 0 & 0.835883 & 0 & 0.456507 & 0.114117 & 0 & 0.212482 & 0.746725 & 0 \\
0.189609 & 0 & 0.164117 & 0 & 0.543493 & 0.885823 & 0 & 0.787518 & 0.253275 & 0 \\
0 & 0 & 0.832375 & 0.478244 & 0.499266 & 0 & 0.842265 & 0 & 0.740099 & 0.230955 \\
0 & 0 & 0.167625 & 0.521756 & 0.500734 & 0 & 0.157735 & 0 & 0.259901 & 0.796045 \\
0 & 0.859034 & 0 & 0.478005 & 0.498000 & 0.116425 & 0 & 0.217493 & 0 & 0.203943 \\
0 & 0.140996 & 0 & 0.521995 & 0.502000 & 0.83575 & 0 & 0.782507 & 0 & 0.796057 \\
0.808242 & 0.860424 & 0 & 0.520590 & 0 & 0 & 0.843871 & 0.215010 & 0.743407 & 0 \\
0.191758 & 0.139576 & 0 & 0.479410 & 0 & 0 & 0.156129 & 0.784990 & 0.256593 & 0
\end{bmatrix}$$

The estimated codeword is $\hat{c} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$. It satisfies $\hat{c} \cdot H^T = 0$ and the decoding terminates.
§ 5.4 Belief Propagation Decoding

- AWGN channel, BPSK modulation
- Design code rate: 0.5
§ 5.5 The Sum-Product Algorithm

- The BP decoding algorithm can be simplified in logarithm domain.
- Regarding bit $c_n$, its probabilities $\Pr(c_n = 0)$ and $\Pr(c_n = 1)$ can be unified in log likelihood ratio (LLR) as
  \[ LLR(c_n) = \ln \frac{\Pr(c_n = 0)}{\Pr(c_n = 1)}. \]
  Inversely,
  \[ \Pr(c_n = 0) = \frac{1}{1 + e^{-LLR(c_n)}}, \Pr(c_n = 1) = \frac{1}{1 + e^{LLR(c_n)}}. \]
- In the Horizontal update
  \[ r_{mn}(\theta) = \sum_{\{c_{n'}: \sum \theta_{n'} = \theta\}} \prod_{n' \in N_m \setminus n} q_{mn'}(\theta). \]

If $c_n = \theta = 1$, we need to consider the permutation of $\{c_{n'}\}$ in which there are odd number (#) of 1s, so that $z_m = c_n + \Sigma c_{n'} = 0$. Otherwise, if $c_n = \theta = 0$, we need to consider the permutation of $\{c_{n'}\}$ in which there are even (#) of 1s.
§ 5.5 The Sum-Product Algorithm

- **Lemma**  Given a binary sequence of length $N$ in which each bit is independent, the probability of bit $n$ being 1 is $p_n$. Then,

  $$\Pr[\text{there are even # of 1s}] = \frac{1}{2} + \frac{1}{2} \prod_{n=1}^{N} (1 - 2p_n),$$

  $$\Pr[\text{there are odd # of 1s}] = \frac{1}{2} - \frac{1}{2} \prod_{n=1}^{N} (1 - 2p_n).$$

- Applying the above lemma, the BP decoding becomes

  **Horizontal update**:

  $$r_{mn}(0) = \frac{1}{2} + \frac{1}{2} \prod_{n' \in N_m \setminus n} \left(1 - 2q_{mn'}(1)\right),$$

  $$r_{mn}(1) = \frac{1}{2} - \frac{1}{2} \prod_{n' \in N_m \setminus n} \left(1 - 2q_{mn'}(1)\right).$$  \hspace{1cm} (7)  \hspace{1cm} (8)

  **Vertical update**:

  $$q_{mn}(0) = \alpha_{mn} \cdot f_n(0) \cdot \prod_{m' \in M_n \setminus m} r_{m'n}(0),$$

  $$q_{mn}(1) = \alpha_{mn} \cdot f_n(1) \cdot \prod_{m' \in M_n \setminus m} r_{m'n}(1).$$  \hspace{1cm} (9)  \hspace{1cm} (10)
§ 5.5 The Sum-Product Algorithm

- Let us define the following LLR values

\[ l_n = \ln \frac{f(0)}{f(1)}, \quad u_{mn} = \ln \frac{q_{mn}(0)}{q_{mn}(1)}, \quad v_{mn} = \ln \frac{r_{mn}(0)}{r_{mn}(1)}, \quad l_{n,p} = \ln \frac{q_n(0)}{q_n(1)}. \]

- Equip with \( \tanh \frac{x}{2} = \frac{e^x - 1}{e^x + 1}, \ 2 \tanh^{-1} x = \ln \frac{1+x}{1-x} \).

- Horizontal update: \( u_{mn} \to v_{mn} \)

\[
\tanh \frac{u_{mn}}{2} = \frac{e^{u_{mn}} - 1}{e^{u_{mn}} + 1} = \frac{q_{mn}(0)}{q_{mn}(1)} - 1 = \frac{q_{mn}(0) - q_{mn}(1)}{q_{mn}(0) + q_{mn}(1)} = 1 - 2q_{mn}(1).
\]

(7) and (8) become

\[
\begin{align*}
r_{mn}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{n' \in N_m \setminus n} \tanh \frac{u_{mn'}}{2}, \\
r_{mn}(1) &= \frac{1}{2} - \frac{1}{2} \prod_{n' \in N_m \setminus n} \tanh \frac{u_{mn'}}{2}.
\end{align*}
\]

(11) \hspace{1cm} (12)

\[
v_{mn} = \ln \frac{(11)}{(12)} = 2 \tanh^{-1} \left( \prod_{n' \in N_m \setminus n} \tanh \frac{u_{mn'}}{2} \right).
\]
§ 5.5 The Sum-Product Algorithm

- Vertical update: $v_{mn} \rightarrow u_{mn}$

$$u_{mn} = \ln \frac{9}{10} = \ln \frac{f(0)}{f(1)} + \sum_{m' \in M_n \setminus m} \ln \frac{r_{m'n}(0)}{r_{m'n}(1)}$$

- Aposteriori LLR

$$l_{n,p} = \ln \frac{q_n(0)}{q_n(1)} = l_n + \sum_{m \in M_n} v_{mn}$$

- Decision on $c_n$

If $l_{n,p} \geq 0$, $\hat{c}_n = 0$; If $l_{n,p} < 0$, $\hat{c}_n = 1$. 