Soft-Decision Decoding Algorithm for Low-density Parity-check Codes: Belief Propagation

NGUYEN Trong Cuong

Computer Communication Lab.,
The University of Aizu, Japan

January 31, 2023

Outline

I. Review of LDPC Codes

II. Decoding Algorithm: Belief Propagation

III. An Example of Belief Propagation Over BI-AWGN Channel

Linear Block Code

- Linear block code C (N, K) is the set of 2^K vectors of length N
  - Each vector in C is called a **codeword**
  - Any linear combination of codewords is also a codeword
- Linear block codes can be defined through **generator matrix** G or **parity check matrix** H
- **Generator matrix** G is a $K \times N$ matrix used to encode a K-bits message vector $\mathbf{m}$ to a N-bits codeword $\mathbf{c} \in C$

$$\mathbf{c} = \mathbf{m}G = \begin{pmatrix} m_1 & \cdots\cdots & m_K \end{pmatrix} \begin{pmatrix} g_{1,1} & \cdots\cdots & g_{1,N} \\ \vdots & \ddots & \vdots \\ g_{K,1} & \cdots\cdots & g_{K,N} \end{pmatrix}_{K \times N} = \begin{pmatrix} c_1 & \cdots\cdots & c_N \end{pmatrix}$$

Parity Check Matrix

- **Parity check matrix** H is a $(N-K)\times N$ matrix that satisfy

$$GH^T = \mathbf{0}$$

- Let $M = N - K$. A codeword $c \in C$ if and only if

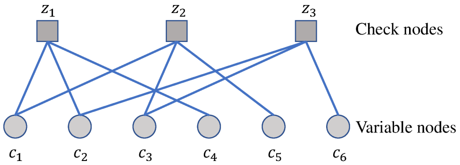
$$H\mathbf{c}^T = \begin{pmatrix} z_{1,1} & \cdots\cdots & z_{1,N} \\ \vdots & \ddots & \vdots \\ z_{M,1} & \cdots\cdots & z_{M,N} \end{pmatrix}_{M\times N} \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} \mathbf{z}_1\mathbf{c}^T \\ \vdots \\ \mathbf{z}_M\mathbf{c}^T \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

- The product $\mathbf{z}_i\mathbf{c}^T = 0$ is called a **check equation**
- **In other words**, a codeword $c \in C$ if and only if it passes all check equations

Low-density Parity-check Codes

- Low-density parity check (LDPC) codes are <u>linear block codes</u> that have a *very sparse* parity check matrix
  - A matrix is said to be **sparse** if *more than half of elements are zero*
- The parity check matrix of LDPC can be graphically presented by a **Tanner graph**
  - Each variable node represents a **bit in the codeword**
  - Each check node represent a **check equation**
  - Each **edge** connects a variable node to a check node

$$H = \begin{array}{c} \\ \\ \\ \end{array}\begin{array}{cl} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} & \begin{array}{c} z_1 \\ z_2 \\ z_3 \end{array} \\ \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} & \end{array}$$



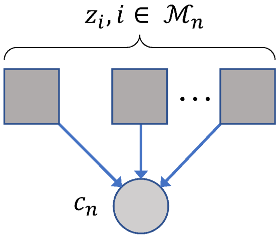
Notations

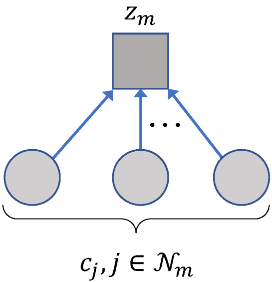
- Given a variable node c_n, let denote $\mathcal{M}_n$ is the set of **checks node** connecting to c_n

$$\mathcal{M}_n = \{m : H_{mn} = 1\}$$

- Given a check node z_m, let denote $\mathcal{N}_m$ is the set of **variable nodes** connecting to z_m

$$\mathcal{N}_m = \{n : H_{mn} = 1\}$$

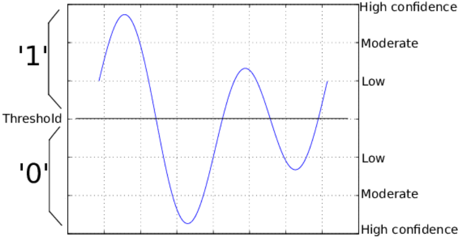




Outline

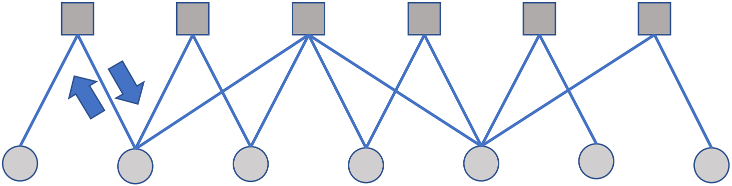
Hard-decision & Soft-decision Decoder

- **Hard-decision decoder** operates on data that take on a *fixed set of possible values (commonly 0 and 1)*
- **Soft-decision decoder** takes on a *whole range of values in between*
  - Extra information provides the reliability of each input and thus help the decoder make the decision better



Belief Propagation

- **Belief propagation** is a soft-decision iterative decoding algorithm for linear block codes.
- The main idea of belief propagation is the **information update** between variable nodes and check nodes in each iteration
  - Each variable node sends a message to each connected check node
  - Each check node sends a message to each connected variable node
- The updated information are **log-likelihood ratios (LLRs)**

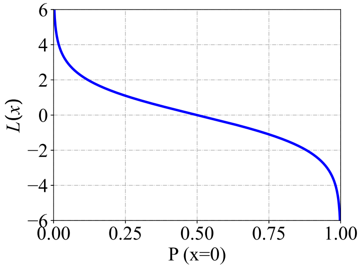


Log Likelihood Ratio

- Let $\tilde{x}$ be the binary-valued random variable taking values on set $\{0, 1\}$. The log-likelihood ratio (LLR) of $\tilde{x}$ is

$$L(\tilde{x}) = \ln \frac{P(\tilde{x} = 1)}{P(\tilde{x} = 0)}$$

- $|L(\tilde{x})|$ measures the **reliability** of $\tilde{x}$
- If $P(\tilde{x} = 0) \to 0$, $|L(\tilde{x})| \to \infty$
- If $P(\tilde{x} = 0) = P(\tilde{x} = 1) = 1/2$, $|L(\tilde{x})| = 0$



Sum of Log Likelihood Ratios

- If $\tilde{x_1}$ and $\tilde{x_2}$ are statistically indepedent

$$L\left(\tilde{x_1} \oplus \tilde{x_2}\right) = \ln \frac{1 + e^{L(\tilde{x_1})} e^{L(\tilde{x_2})}}{e^{L(\tilde{x_1})} + e^{L(\tilde{x_2})}},$$

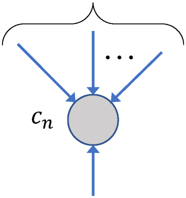
where $\oplus$ is the addition operation in GF(2).

- Let define the operator $\boxplus$ as

$$\boxed{L\left(\tilde{x_1}\right) \boxplus L\left(\tilde{x_2}\right) \stackrel{\Delta}{=} L\left(\tilde{x_1} \oplus \tilde{x_2}\right)}$$

A Posteriori Probability

- **Target of the algorithm:** Evaluate the probability of a bit c_n given the receiver vector $\mathbf{r}$ and the parity constraints in each iteration
- The probability is called *a posterior probability* (APP)



$$L_{\text{APP}}(n) = \ln \frac{P\left(c_n = 1 | \{z_m = 0 \text{ for all } m \in \mathcal{M}_n\}, \mathbf{r}\right)}{P\left(c_n = 0 | \{z_m = 0 \text{ for all } m \in \mathcal{M}_n\}, \mathbf{r}\right)}$$

- Applying Bayes' rule, the APP can be expressed as

$$L_{\text{APP}}(n) = \underbrace{\ln \frac{P\left(c_n = 1 | r_n\right)}{P\left(c_n = 0 | r_n\right)}}_{\text{intrinsic}} + \underbrace{\ln \frac{P\left(\{z_m = 0 \text{ for all } m \in \mathcal{M}_n\} | c_n = 1, \mathbf{r}\right)}{P\left(\{z_m = 0 \text{ for all } m \in \mathcal{M}_n\} | c_n = 0, \mathbf{r}\right)}}_{\text{extrinsic}}$$

Intrinsic Term

- **The intrinsic term** represents the reliability of the received bit based on channel output

$$L_{\text{ch}}(n) \triangleq \ln \frac{P(c_n = 1|r_n)}{P(c_n = 0|r_n)}$$

- **The intrinsic term** depends on channel model
- **E.g., binary-input additive white Gaussian noise (BI-AWGN)** where the channel input belong to the set $\{-A, A\}$, the noise variance is σ^2

$$L_{\text{ch}}(n) = \ln \frac{\exp\left[-\frac{1}{2\sigma^2}(r_n - A)^2\right]}{\exp\left[-\frac{1}{2\sigma^2}(r_n + A)^2\right]} = \frac{2 r_n A}{\sigma^2}$$

Extrinsic Term (1)

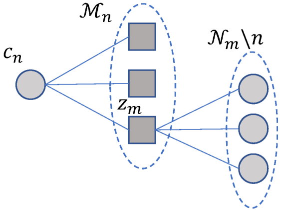
- **The extrinsic term** represents the reliability of received bits given the observation of the code structure

$$L_{E_v}(m,n) \triangleq \ln \frac{P\left(\{z_m = 0 \text{ for all } m \in \mathcal{M}_n\} | c_n = 1, \mathbf{r}\right)}{P\left(\{z_m = 0 \text{ for all } m \in \mathcal{M}_n\} | c_n = 0, \mathbf{r}\right)}$$

- Let denote $z_{m,n}$ as the computation of m-th parity check excluding bit c_n

$$z_{m,n} = \bigoplus_{j \in \mathcal{N}_m \backslash n} c_j.$$

- If $c_n = 1$, $z_{m,n} = 1$ for all check $m \in \mathcal{M}_n$
- If $c_n = 0$, $z_{m,n} = 0$ for all check $m \in \mathcal{M}_n$



Extrinsic Term (2)

- **The extrinsic term** can be rewritten as

$$
\begin{aligned}
L_{E_v}(m,n) &= \ln \frac{P\left(\{z_m = 0 \text{ for all } m \in \mathcal{M}_n\} | c_n = 1, \mathbf{r}\right)}{P\left(\{z_m = 0 \text{ for all } m \in \mathcal{M}_n\} | c_n = 0, \mathbf{r}\right)} \\
&= \ln \frac{P\left(z_{m,n} = 1 \text{ for all } m \in \mathcal{M}_n | \mathbf{r}\right)}{P\left(z_{m,n} = 0 \text{ for all } m \in \mathcal{M}_n | \mathbf{r}\right)} \\
&= \ln \frac{\prod_{m \in \mathcal{M}_n} P\left(z_{m,n} = 1 | \mathbf{r}\right)}{\prod_{m \in \mathcal{M}_n} P\left(z_{m,n} = 0 | \mathbf{r}\right)} \\
&= \sum_{m \in \mathcal{M}_n} \ln \frac{P\left(z_{m,n} = 1 | \mathbf{r}\right)}{P\left(z_{m,n} = 0 | \mathbf{r}\right)} \\
&= \boxed{\sum_{m \in \mathcal{M}_n} L\left(z_{m,n} | \mathbf{r}\right)}
\end{aligned}
$$

- The LLR terms are messages from **check nodes to variable nodes**

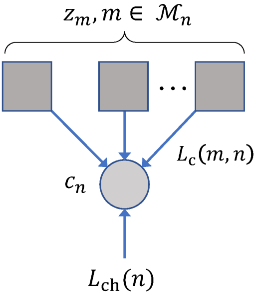
A Posteriori Probability (cont.)

- Let denote the message from m-th check node to n-th variable node as

$$L_{\mathrm{c}}(m,n) \triangleq L\left(z_{m,n}|\mathbf{r}\right)$$

- The APP of variable nodes is

$$L_{\mathrm{APP}}(n) = L_{\mathrm{ch}}(n) + \sum_{m \in \mathcal{M}_n} L_{\mathrm{c}}(m,n).$$

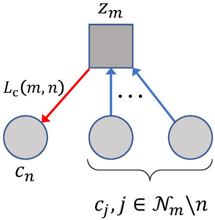


- **Next question:** How to compute the LLR messages from check nodes to variable nodes?

Message From Check Node

- The LLR message from m-th check node to n-th variable node is obtained from *all variable nodes of the check node, excluding n-th variable node*

$$
\begin{aligned}
L_{\mathrm{c}}(m,n) &= L\left(z_{m,n}|\mathbf{r}\right) \\
&= L\left(\bigoplus_{j\in\mathcal{N}_{m,n}} c_j|\mathbf{r}\right) \\
&= \underset{j\in\mathcal{N}_m\backslash n}{\boxed{+}} L\left(c_j|\mathbf{r}\right)
\end{aligned}
$$

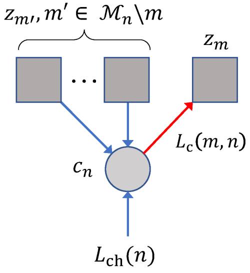


- The LLR terms are messages from **variable nodes to check nodes**

Message From Variable Node

- The message from a variable node to a check node is computed **by taking into account of other check nodes**
- Let denote the message from n-th variable node to m-th check node as

$$L_{\mathrm{v}}(m,n) \triangleq L\left(c_n \middle| \{z_{m'}=0 \text{ for all } m' \in \mathcal{M}_n \backslash m\}, \mathbf{r}\right)$$
$$= L_{\mathrm{ch}}(n) + \sum_{m' \in \mathcal{M}_n \backslash m} L_{\mathrm{c}}(m', n)$$



Summary: Belief Propagation

Input: The # of iterations L, set $L_{\mathrm{v}}(m,n) = L_{\mathrm{ch}}(n)$ for all (m,n) with $H(m,n) = 1$, **decoding** = False

for *each iteration* **do**

  for *each* (m,n) *with* $H(m,n) = 1$ **do** check node update

$$L_{\mathrm{c}}(m,n) = \underset{j \in \mathcal{N}_{m,n}}{\boxed{+}} L_{\mathrm{v}}(m,j)$$

  end

  for *each* (m,n) *with* $H(m,n) = 1$ **do** variable node update

$$L_{\mathrm{v}}(m,n) = L_{\mathrm{ch}}(n) + \sum_{i \in \mathcal{M}_{n,m}} L_{\mathrm{c}}(i,n)$$

  end

  for *each* n **do**

$$L_{\mathrm{APP}}(n) = L_{\mathrm{ch}}(n) + \sum_{i \in \mathcal{M}_{n}} L_{\mathrm{c}}(i,n)$$

   if $L_{APP}(n) < 0$ **then** set $\tilde{c_n} = 0$;

   else set $\tilde{c_n} = 1$;

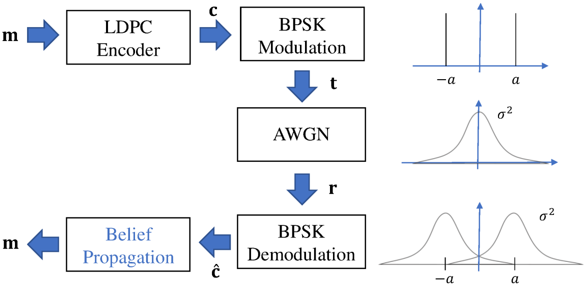
  end

  if $H\tilde{\boldsymbol{c}}^T = \boldsymbol{0}$ **then** set **decoding** = True and break;

end

Outline

An Example



- Consider a transmission model with signal amplitude $A = 1$, noise variance of AWGN channel $\sigma^2 = 1$

$$
\begin{aligned}
\mathbf{c} &= (\ 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1\ ) \\
\mathbf{t} &= (\ -1 \quad -1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1\ ) \\
\mathbf{r} &= (-0.63 \quad -0.83 \quad -0.73 \quad -0.04 \quad 0.1 \quad 0.95 \quad -0.76 \quad 0.66 \quad -0.55 \quad 0.58) \\
\hat{\mathbf{c}} &= (\ 0 \quad 0 \quad 0 \quad \underline{0} \quad \underline{1} \quad 1 \quad 0 \quad 1 \quad 0 \quad 1\ )
\end{aligned}
$$

Initialization

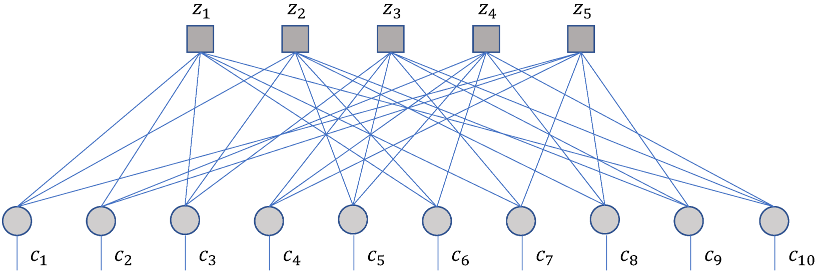
$$\mathbf{L}_{\mathrm{ch}} = \begin{pmatrix} -1.3 & -1.7 & -1.5 & -0.08 & 0.2 & 1.9 & -1.5 & 1.3 & -1.1 & 1.2 \end{pmatrix}$$

$$\mathbf{L}_{\mathrm{v}} = \begin{pmatrix} -1.3 & -1.7 & -1.5 & 0 & 0 & 1.9 & -1.5 & 0 & 0 & 1.2 \\ -1.3 & 0 & -1.5 & 0 & 0.2 & 1.9 & 0 & 1.3 & -1.1 & 0 \\ 0 & 0 & -1.5 & -0.08 & 0.2 & 0 & -1.5 & 0 & -1.1 & 1.2 \\ 0 & -1.7 & 0 & -0.08 & 0.2 & 1.9 & 0 & 1.3 & 0 & 1.2 \\ -1.3 & -1.7 & 0 & -0.08 & 0 & 0 & -1.5 & 1.3 & -1.1 & 0 \end{pmatrix}$$

$$\mathbf{L}_{\mathrm{c}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

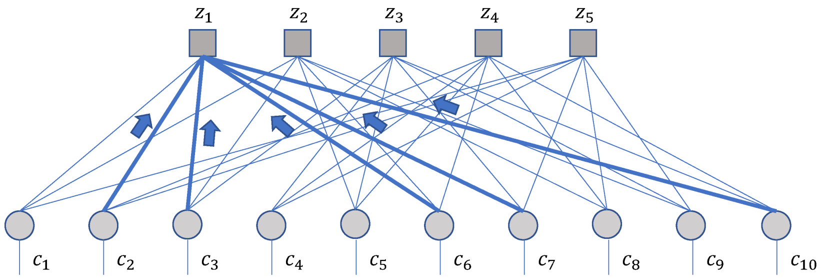
Initialization

$$
\begin{array}{cc}
H = \begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix} & \begin{array}{c} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{array} \\
\begin{array}{cccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} \end{array} &
\end{array}
$$



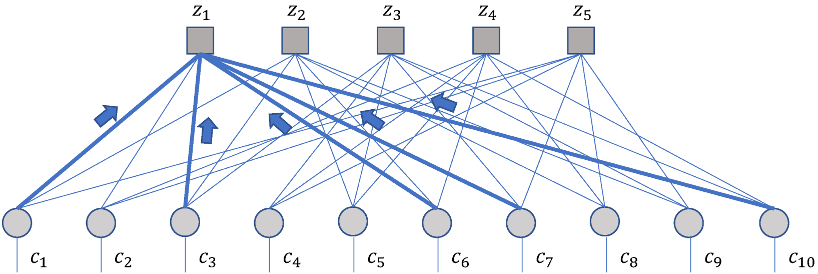
Check Node Update

$$\mathbf{L_c} = \begin{pmatrix} \boxed{-0.21} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



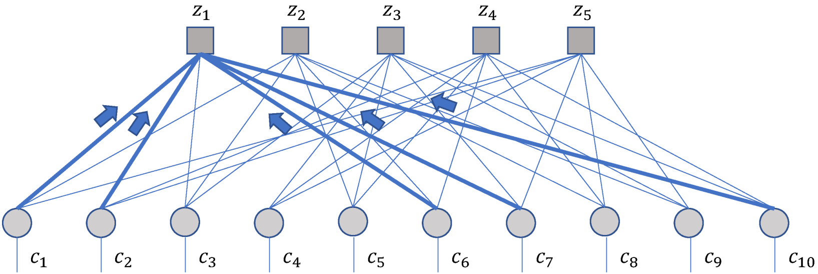
Check Node Update

$$\mathbf{L_c} = \begin{pmatrix} -0.21 & \boxed{-0.17} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



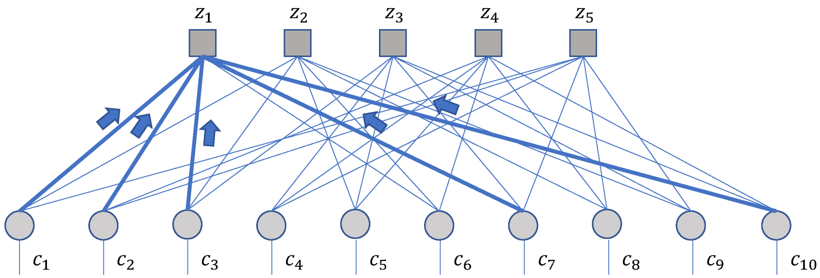
Check Node Update

$$\mathbf{L_c} = \begin{pmatrix} -0.21 & -0.17 & \boxed{-0.19} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



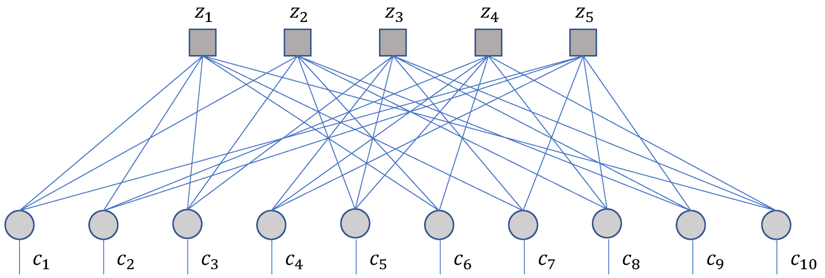
Check Node Update

$$\mathbf{L}_c = \begin{pmatrix} -0.21 & -0.17 & -0.19 & 0 & 0 & \boxed{0.16} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



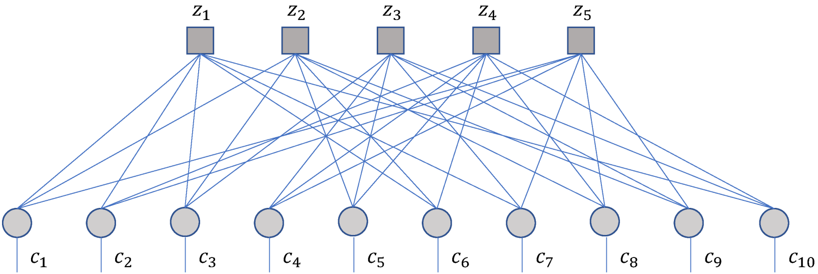
Check Node Update - Finish

$$\mathbf{L}_c = \begin{pmatrix} -0.21 & -0.17 & -0.19 & 0 & 0 & 0.16 & -0.18 & 0 & 0 & 0.22 \\ 0.027 & 0 & 0.024 & 0 & -0.15 & -0.02 & 0 & -0.026 & 0.03 & 0 \\ 0 & 0 & -0.0013 & -0.021 & 0.0083 & 0 & -0.0013 & 0 & -0.0017 & 0.0016 \\ 0 & -0.0018 & 0 & -0.03 & 0.012 & 0.0016 & 0 & 0.0021 & 0 & 0.0023 \\ 0.01 & 0.0083 & 0 & 0.14 & 0 & 0 & 0.0088 & -0.0097 & 0.011 & 0 \end{pmatrix}$$



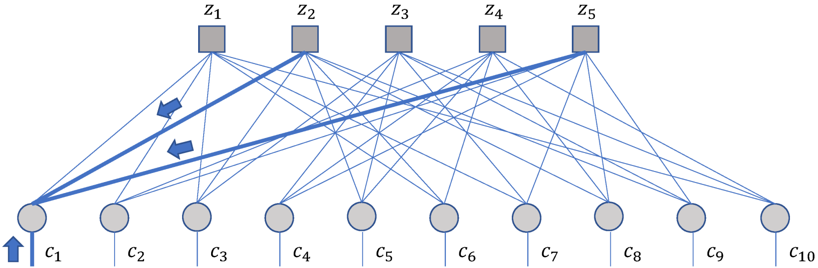
Variable Node Update

$$\mathbf{L}_{\mathrm{v}} = \begin{pmatrix} -1.3 & -1.7 & -1.5 & 0 & 0 & 1.9 & -1.5 & 0 & 0 & 1.2 \\ -1.3 & 0 & -1.5 & 0 & 0.2 & 1.9 & 0 & 1.3 & -1.1 & 0 \\ 0 & 0 & -1.5 & -0.08 & 0.2 & 0 & -1.5 & 0 & -1.1 & 1.2 \\ 0 & -1.7 & 0 & -0.08 & 0.2 & 1.9 & 0 & 1.3 & 0 & 1.2 \\ -1.3 & -1.7 & 0 & -0.08 & 0 & 0 & -1.5 & 1.3 & -1.1 & 0 \end{pmatrix}$$



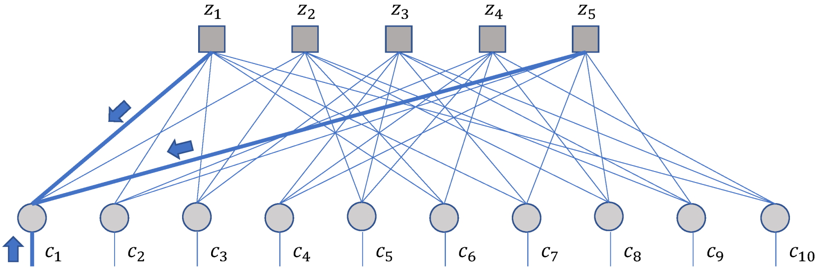
Variable Node Update

$$\mathbf{L}_v = \begin{pmatrix} \boxed{-1.26} & -1.7 & -1.5 & 0 & 0 & 1.9 & -1.5 & 0 & 0 & 1.2 \\ -1.3 & 0 & -1.5 & 0 & 0.2 & 1.9 & 0 & 1.3 & -1.1 & 0 \\ 0 & 0 & -1.5 & -0.08 & 0.2 & 0 & -1.5 & 0 & -1.1 & 1.2 \\ 0 & -1.7 & 0 & -0.08 & 0.2 & 1.9 & 0 & 1.3 & 0 & 1.2 \\ -1.3 & -1.7 & 0 & -0.08 & 0 & 0 & -1.5 & 1.3 & -1.1 & 0 \end{pmatrix}$$



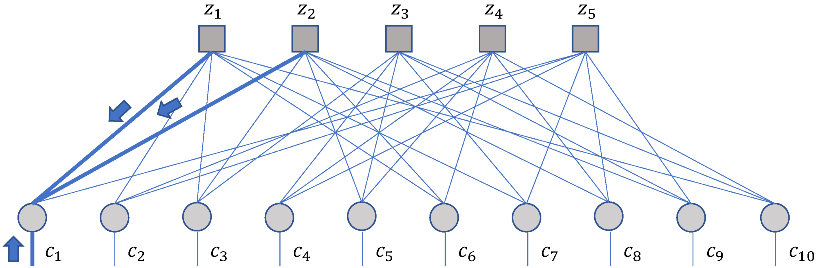
Variable Node Update

$$\mathbf{L}_\mathrm{v} = \begin{pmatrix} -1.26 & -1.7 & -1.5 & 0 & 0 & 1.9 & -1.5 & 0 & 0 & 1.2 \\ \boxed{-1.5} & 0 & -1.5 & 0 & 0.2 & 1.9 & 0 & 1.3 & -1.1 & 0 \\ 0 & 0 & -1.5 & -0.08 & 0.2 & 0 & -1.5 & 0 & -1.1 & 1.2 \\ 0 & -1.7 & 0 & -0.08 & 0.2 & 1.9 & 0 & 1.3 & 0 & 1.2 \\ -1.3 & -1.7 & 0 & -0.08 & 0 & 0 & -1.5 & 1.3 & -1.1 & 0 \end{pmatrix}$$



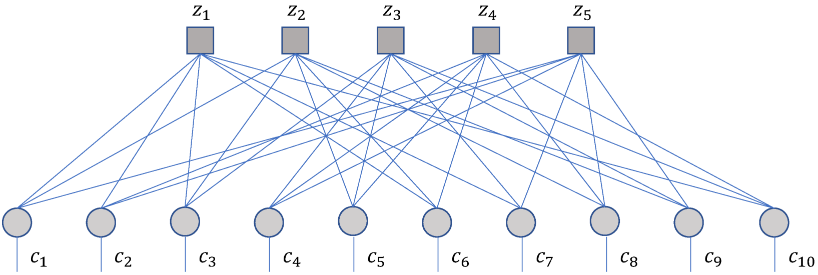
Variable Node Update

$$\mathbf{L}_{\mathrm{v}} = \begin{pmatrix} \boxed{-1.26} & -1.7 & -1.5 & 0 & 0 & 1.9 & -1.5 & 0 & 0 & 1.2 \\ -1.5 & 0 & -1.5 & 0 & 0.2 & 1.9 & 0 & 1.3 & -1.1 & 0 \\ 0 & 0 & -1.5 & -0.08 & 0.2 & 0 & -1.5 & 0 & -1.1 & 1.2 \\ 0 & -1.7 & 0 & -0.08 & 0.2 & 1.9 & 0 & 1.3 & 0 & 1.2 \\ \boxed{-1.48} & -1.7 & 0 & -0.08 & 0 & 0 & -1.5 & 1.3 & -1.1 & 0 \end{pmatrix}$$



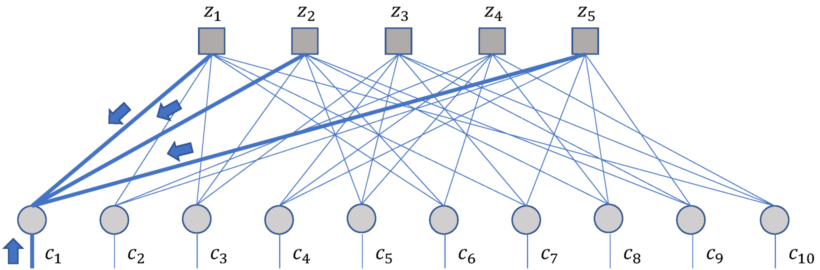
Variable Node Update - Finish

$$\mathbf{L}_v = \begin{pmatrix} -1.26 & -1.69 & -1.47 & 0 & 0 & 1.9 & -1.5 & 0 & 0 & 1.2 \\ -1.5 & 0 & -1.69 & 0 & 0.2 & 1.9 & 0 & 1.3 & -1.1 & 0 \\ 0 & 0 & -1.66 & 0.03 & 0.2 & 0 & -1.5 & 0 & -1.1 & 1.2 \\ 0 & -1.86 & 0 & 0.039 & 0.2 & 1.9 & 0 & 1.3 & 0 & 1.2 \\ -1.48 & -1.78 & 0 & -0.131 & 0 & 0 & -1.5 & 1.3 & -1.1 & 0 \end{pmatrix}$$



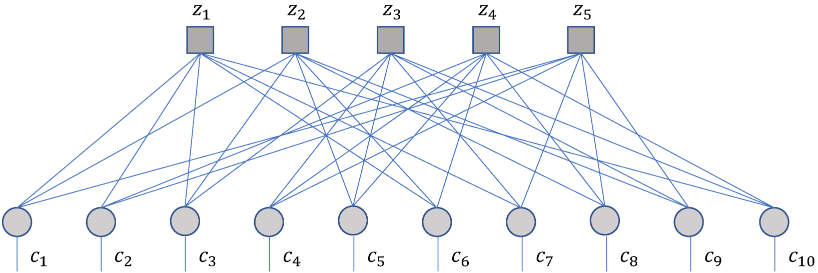
Computing the APP

$$\mathbf{L}_{\text{APP}} = \left(\boxed{-1.4} \quad -1.7 \quad -1.5 \quad -0.08 \quad 0.2 \quad 1.9 \quad -1.5 \quad 1.3 \quad -1.1 \quad 1.2 \right)$$



Computing the APP

$$\mathbf{L}_{\text{APP}} = \begin{pmatrix} -1.4 & -1.8 & -1.6 & 0.011 & 0.072 & 2 & -1.7 & 1.3 & -1.1 & 1.4 \end{pmatrix}$$



Parity Check

- The estimated codeword after the first iteration is

$$\tilde{\mathbf{c}} = \begin{pmatrix} 0 & 0 & 0 & 1 & \underline{1} & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

- The estimated codeword is **failed** the parity check. The procedure is repeated in the next iteration

Iteration 2:

$$\tilde{\mathbf{c}} = \begin{pmatrix} 0 & 0 & 0 & 1 & \underline{1} & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Iteration 3:

$$\tilde{\mathbf{c}} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

After two more iterations, the codeword is corrected