• For memoryless channel ML decoding is reduced to computing and comparing additive metrics of codewords. Metric formula depends on the channel model (Hamming distance, scalar product, etc.)

• Trellis representation of the code reduces ML decoding to search of the shortest path over the graph.

• Minimal trellis of the code, is unique.

• ML decoding is implemented by using the Viterbi algorithm.
Assume that the message is encoded by two or more systematic encoders. At the channel output soft decisions from demodulator enter the decoder of the first code. First decoder’s decisions...
Assume that the message is encoded by two or more systematic encoders. At the channel output soft decisions from demodulator enter the decoder of the first code. First decoder’s decisions are sent to the second decoder, then again to the first decoder as it would be output of another channel, probably, with less noise since the first decoder corrected some errors ...

In practice, tens of such iterations can be used. This approach is called (belief propagation).
Assume that the message is encoded by two or more systematic encoders. At the channel output soft decisions from demodulator enter the decoder of the first code. First decoder’s decisions are sent to the second decoder, then again to the first decoder as it would be output of another channel, probably, with less noise since the first decoder corrected some errors ...

In practice, tens of such iterations can be used. This approach is called (belief propagation). To apply this method we need SISO (soft input, soft output) decoders. The best SISO decoders are maximum a posteriori probability (MAP) decoders.
A posteriori probability

Let events $H_1, \ldots, H_M$ be non-intersecting, $H_i \cap H_j = \emptyset$, and $P(\bigcup_{m=1}^{M} H_m) = 1$. Then they are considered as a set of hypothesis with respect to event $A$. Law of total probability

$$P(A) = \sum_{m=1}^{M} P(A|H_m)P(H_m)$$

and Bayes’ a posteriori probability formula

$$P(H_j|A) = \frac{P(A, H_j)}{P(A)} = \frac{P(A|H_j)P(H_j)}{\sum_{m=1}^{M} P(A|H_m)P(H_m)}.$$
Let $C = \{c_m, m = 1, \ldots, M\} \subseteq \{0, 1\}^n$, be a binary block code and $y = (y_1, \ldots, y_n) \in Y^n$ is a channel output sequence. A posteriori probability of $c \in \{0, 1\}$ at position $t$ is

$$p(c_t = c | y) = \frac{p(c_t = c, y)}{p(y)},$$

where

$$p(c_t = c, y) = \sum_{c \in C_t(c)} p(c, y),$$

and $C_t(c)$ is a set of codewords which have $c$ at position $t$. 
Let $C = \{c_m, m = 1, \ldots, M\} \subseteq \{0, 1\}^n$, be a binary block code and $y = (y_1, \ldots, y_n) \in Y^n$ is a channel output sequence. A posteriori probability of $c \in \{0, 1\}$ at position $t$ is

$$p(c_t = c|y) = \frac{p(c_t = c, y)}{p(y)},$$

where

$$p(c_t = c, y) = \sum_{c \in C_t(c)} p(c, y),$$

and $C_t(c)$ is a set of codewords which have $c$ at position $t$.

Question: What is complexity of computing $p(c_t = c|y)$ in general?
Computing probabilities by using trellises

Hint: compute probabilities for one section taking into account both “past” and “future” sections.
Computing probabilities by using trellises

Hint: compute probabilities for one section taking into account both “past” and “future” sections.

\[ \sigma_t(m', m) = \alpha_t(m') \gamma_t(m', m) \beta_t(m), \quad m, m' \in \{0, 1, 2, 3\} \]

\[ \lambda_t = \log \sigma_t(0, 0) + \sigma_t(1, 2) + \sigma_t(2, 3) + \sigma_t(3, 1) + \sigma_t(0, 1) + \sigma_t(1, 0) + \sigma_t(2, 1) + \sigma_t(3, 3) \]

Computational complexity is determined by the trellis complexity (not by the size of the code)
Computing probabilities by using trellises

Let $s_t$ be a node at layer $t$.

$$
\Pr (s_{t-1} = m'; s_t = m | y) = \frac{\Pr (s_{t-1} = m'; s_t = m, y)}{p(y)}.
$$

Only numerator is important for evaluating log-likelihood ratio

$$
\sigma_t(m', m) = \Pr(s_{t-1} = m', s_t = m, y).
$$

Denote $y^j_i = (y_i, y_{i+1}, \ldots, y_j)$
Decompose \((s_{t-1} = m', s_t = m, y)\) to past, current and future as

\[
\sigma_t(m', m) = \Pr \left( (s_{t-1} = m', y_{t-1}^t); (s_t = m, y_t); (y_{t+1}^n) \right).
\]

For memoryless channel from
\[
\Pr(A_1 A_2 A_3) = \Pr(A_1) \Pr(A_2 | A_1) \Pr(A_3 | A_1 A_2)
\]
follows

\[
\begin{align*}
\Pr(A_1) &= \Pr(s_{t-1} = m', y_{1}^{t-1}); \\
\Pr(A_2 | A_1) &= \Pr(s_t = m, y_t | s_{t-1} = m', y_{1}^{t-1}) = \\
&= \Pr(s_t = m, y_t | s_{t-1} = m'); \\
\Pr(A_3 | A_1 A_2) &= \Pr(y_{t+1}^n | s_{t-1} = m', s_t = m, y_1^t) \\
&= \Pr(y_{t+1}^n | s_t = m).
\end{align*}
\]
Decompose \((s_{t-1} = m', s_t = m, y)\) to past, current and future as

\[
\sigma_t(m', m) = \Pr \left( (s_{t-1} = m', y_1^{t-1}); (s_t = m, y_t); (y_{t+1}^n) \right).
\]

For memoryless channel from

\[
\Pr(A_1A_2A_3) = \Pr(A_1) \Pr(A_2|A_1) \Pr(A_3|A_1A_2)
\]

follows

\[
\begin{align*}
\Pr(A_1) &= \Pr(s_{t-1} = m', y_1^{t-1}); \\
\Pr(A_2|A_1) &= \Pr(s_t = m, y_t|s_{t-1} = m', y_1^{t-1}) = \\
&= \Pr(s_t = m, y_t|s_{t-1} = m'); \\
\Pr(A_3|A_1A_2) &= \Pr(y_{t+1}^n|s_{t-1} = m', s_t = m, y_1^t) \\
&= \Pr(y_{t+1}^n|s_t = m).
\end{align*}
\]
New notations:

\[ \alpha_t(m) = \Pr(A_1) = \Pr(s_t = m, y_t^t); \]
\[ \gamma_t(m', m) = \Pr(s_t = m, y_t | s_{t-1} = m'); \]
\[ \beta_t(m) = \Pr(y_{t+1}^n | s_t = m) \]

In these terms

\[ \sigma_t(m', m) = \alpha_{t-1}(m') \gamma_t(m', m) \beta_t(m). \]
Computing probabilities by using trellises

We need to know $\alpha_t$, $\gamma_t$, и $\beta_t$. Due to law of total probability

$$\alpha_t(m) = \sum_{m'} \Pr(s_{t-1} = m', y_{t-1}^{t-1}) \Pr(s_t = m, y_t | s_{t-1} = m', y_{1}^{t-1}).$$

Condition $y_1^{t-1}$ not needed given $s_{t-1}$. Thereby, we have recursion

$$\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m') \gamma_t(m', m).$$

with initial conditions

$$\alpha_0(m) = \begin{cases} 1, & m = 0; \\ 0, & m \neq 0. \end{cases}$$
Computing probabilities by using trellises

Similarly, in inverse direction

$$\beta_t(m) = \sum_{m'} \Pr(s_{t+1} = m', y_{t+1}^n | s_t = m) =$$

$$= \sum_{m'} \Pr(s_{t+1} = m', y_{t+1}, y_{t+2}^n | s_t = m) =$$

$$= \sum_{m'} \Pr(s_{t+1} = m', y_{t+1} | s_t = m, y_{t+2}^n) \times$$

$$\times \Pr(y_{t+2}^n | s_t = m, s_{t+1} = m', y_{t+1}) =$$

$$= \sum_{m'} \underbrace{\Pr(s_{t+1} = m', y_{t+1} | s_t = m)}_{\gamma} \underbrace{\Pr(y_{t+2}^n | s_{t+1} = m')}_{\beta}.$$  

Recursion for $\beta$

$$\beta_t(m) = \sum_{m'} \beta_{t+1}(m') \gamma_{t+1}(m', m)$$

with initial conditions

$$\beta_n(m) = \begin{cases} 1, & m = 0; \\ 0, & m \neq 0. \end{cases}$$
Computing \( \gamma \):

\[
\gamma_t(m', m) = \sum_{c_t} \Pr(s_t = m, c_t, y_t | s_{t-1} = m') = \sum_{c_t} p(c_t | m, m') p(y_t | c_t)
\]

and \( \gamma_t(m', m) = 0 \) for those \((m, m')\), which are not connected in the trellis. For bit-wise trellis

\[
\gamma_t(m', m) = p(c_{t,m,m'}) p(y_t | c_{t,m,m'})
\]

where \( c_{t,m,m'} \) is the code symbol associated, with edge \( m' \rightarrow m \) at layer \( t \). Let \( S_t(c) \) be the set of pairs \((m, m')\), to which \( c_{t,m,m'} = c \) is assigned.

\[
p(c_t = c | y) = \frac{\sum_{(m', m) \in S_t(c)} \sigma_t(m', m)}{p(y)}.
\]

Similarly, a posteriori probability for information bits can be written.
Computing probabilities by using trellises

Usually, \( p(y) \) are not needed since the SISO decoder output are bit LLRs:

\[
\lambda_t = \ln \frac{p(c_t = 1|y)}{p(c_t = 0|y)} = \frac{\sum_{(m', m) \in S_t(1)} \sigma_t(m', m)}{\sum_{(m', m) \in S_t(0)} \sigma_t(m', m)}.
\]

\[
\sigma_t(m', m) = \alpha_{t-1}(m') \gamma_t(m', m) \beta_t(m), \ m, m' \in \{0, 1, 2, 3\}
\]

\[
\lambda_t = \log \frac{\sigma_t(0,1) + \sigma_t(1,0) + \sigma_t(2,1) + \sigma_t(3,3)}{\sigma_t(0,0) + \sigma_t(1,2) + \sigma_t(2,3) + \sigma_t(3,1)}
\]
**Input:** Channel output sequence $y$, a priori probability $u_1, \ldots, u_k$ or codeword symbols $c_1, \ldots, c_n$.

**Output:** LLRs $\lambda_1, \ldots, \lambda_n$.

**Initialization:** For all $t = 1, 2, \ldots, n$ and pairs $(m', m)$ compute $\gamma_t(m', m)$.

1. **Forward pass:** Starting from layer 0, for all layers and all nodes at each layer compute $\alpha_t(m)$.

2. **Backward pass:** Starting from layer $n$, moving backwards for all layers and all nodes at each layer compute $\beta_t(m)$.

3. Compute all $\sigma_t(m', m)$.

4. Compute all $\lambda_t, t = 1, 2, \ldots, n$. 
BCJR algorithm

**Input:** Channel output sequence $y$, a priori probability $u_1, \ldots, u_k$ or codeword symbols $c_1, \ldots, c_n$.

**Output:** LLRs $\lambda_1, \ldots, \lambda_n$.

**Initialization:** For all $t = 1, 2, \ldots, n$ and pairs $(m', m)$ compute $\gamma_t(m', m)$.

1. **Forward pass:** Starting from layer 0, for all layers and all nodes at each layer compute $\alpha_t(m)$.

2. **Backward pass:** Starting from layer $n$, moving backwards for all layers and all nodes at each layer compute $\beta_t(m)$.

3. Compute all $\sigma_t(m', m)$.

4. Compute all $\lambda_t$, $t = 1, 2, \ldots, n$.

Steps 2-4 are performed at one backward pass. Values $\beta_t(m)$, are not kept in decoder memory. Only current values $\beta(m)$ are updated.
Consider BSC with $p_0 = 0.01$. Channel output $y = (0, 0, 1, 0, 1)$.

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\text{Information symbols } u_1, u_2 \quad \text{ } \quad c = (c_1, \ldots, c_5) \quad \text{ } \quad p(y|c) \quad \text{ } \quad p(c|y) = \frac{p(y|c)p(c)}{p(y)}$$

$y = (0, 0, 1, 0, 1)$

| Information symbols $u_1, u_2$ | $c = (c_1, \ldots, c_5)$ | $p(y|c)$ | $p(c|y)$ |
|-------------------------------|-------------------------------|----------|-----------|
| 00                            | 00000                         | $p_0^2(1-p_0)^3 = 0.00729$ | 0.45      |
| 01                            | 01011                         | $p_0^3(1-p_0)^2 = 0.00081$ | 0.05      |
| 10                            | 11100                         | $p_0^3(1-p_0)^2 = 0.00081$ | 0.05      |
| 11                            | 10111                         | $p_0^2(1-p_0)^3 = 0.00729$ | 0.45      |

$$p(y) = \sum_{m=0}^{3} p(y|c_m)p(c_m) = 0.00405$$

| Символы | Output LLRs $\ln \frac{p(1|y)}{p(0|y)}$ |
|---------|-------------------------------------|
| $u_1$   | 0                                   |
| $u_2$   | 0                                   |
| $c_1, c_3, c_4, c_5$ | 0                                   |
| $c_2$   | $-2.1972$                           |
y =

```
<table>
<thead>
<tr>
<th>0</th>
<th>01</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
</tbody>
</table>
```

a) The trellis and channel output sequence \( y \)

\[
\begin{align*}
\frac{1}{2}(1 - p_0) & \\
\frac{1}{2} p_0 & \\
\frac{1}{2} (1 - p_0) & \\
\frac{1}{2}(1 - p_0) p_0 & \\
(1 - p_0) & \\
\frac{1}{2}(1 - p_0)^2 & \\
\end{align*}
\]

b) Formulas for \( \gamma \)

\[
\begin{align*}
0.45 & \\
0.045 & \\
0.005 & \\
0.05 & \\
0.405 & \\
0.09 & \\
0.09 & \\
0.09 & \\
\end{align*}
\]

b) Counting \( \gamma \)
Example

\( \delta \) Counting \( \gamma \)

\( \theta \) Counting \( \alpha \)

\( \vartheta \) Counting \( \beta \)
Example

1

0.45

0.05

0.05

0.0225

0.09

0.00405

(1) Counting $\alpha$

0.00405

0.45

0.0045

0.045

0.09

1

0.0002025

0.0018225

0.002025

1 $\times$ 0.45 $\times$ 0.0045

=0.002025

(4) Counting $\beta$

0

0

0.0002025

0.0002025

0.0002025

0.0002025

1 $\times$ 0.05 $\times$ 0.0405

=0.002025

(5) Counting $\sigma$
Thick edges correspond to information symbol 1. Code symbols processing:
For example, $c_2 = 0$ is assigned to edges $0 \rightarrow 0$ and $1 \rightarrow 1$. Other two edges of section 2 correspond to $c_2 = 1$. Therefore,

$$\ln \frac{0.0002025 + 0.0002025}{0.0018225 + 0.0018225} = -2.1972,$$
• SISO decoding is used for exchange information between decoders used in different concatenated constructions: product codes, concatenated codes, turbo-codes, LDPC codes, etc.

• Optimal SISO decoder is symbol-MAP decoder.

• For codes represented by trellises symbol-MAP decoding is performed by using BCJR decoding algorithm

• Complexity of the BCJR decoding is 2 times higher than that of the Viterbi decoding. It makes 2 passes, forward and backward.

• Unlike ML decoding, BCJR decoding requires additions and multiplications. In practice, simplified versions operating in log domain are used.