convolutional codes: introduction

Prof. Luiz DaSilva
dasilval@tcd.ie
+353 1 896-3660

block codes …

• take k input bits and produce n output bits
  – in practice, n and k are large

• there is no data dependency between blocks

• useful for latency-tolerant communications
  (e.g., data communications)
... and convolutional codes

• take a small number of input bits and produces a small number of output bits for each time period

• data passes through the coder in a continuous stream

• useful for low-latency communications

(n,k) convolutional codes

• typically, n and k are small (1 ≤ k ≤ 3, 2 ≤ n ≤ 6)
  – and, often, k = 1

• output bits depend not only on the current k input bits but also on past input bits
  – number of time slots on which output depends is the constraint length K (the “memory” of the code)
  – distance properties (and coding gain) increases with K
k = 1, n = 2, K = 3 convolutional code example

\[ r = \text{rate} = \frac{1}{2} \]

\[ k = 1, n = 2, K = 3 \text{ convolutional code example, redrawn} \]

assume both flip-flops initially cleared
$k = 1, n = 2, K = 3$ convolutional code example, timing

\[ \begin{align*}
\text{INPUT CLOCK} & : \quad & \\
\text{INPUT BITS} & : & \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \\
\text{LEFT FF OUTPUT} & : \\
\text{RIGHT FF OUTPUT} & : \\
\text{UPPER XOR OUTPUT} & : \\
\text{LOWER XOR OUTPUT} & : \\
\text{SYMBOL CLOCK} & : \\
\text{OUTPUT SYMBOLS} & : & \quad 0 \quad 3 \quad 1 \quad 1 \quad 1 \quad 0 
\end{align*} \]

$k = 2, n = 3, K = 2$ convolutional code example

\[ \begin{align*}
\text{input} \quad & \quad (2 \text{ bits at a time}) \\
\text{output bit 1} \\
\text{output bit 2} \\
\text{output bit 3} \\
\text{r = 2/3} 
\end{align*} \]
representation of a convolutional code

1. encoder block diagram (previous examples)
2. generator representation
   • represents the relationship between input/stored bits and output bits
3. trellis representation
4. state diagram representation

convolutional code generators

- there is one generator vector for each of the n output bits
  - the length of the generator vector for a rate-r code, \( r = k/n \), with constraint length \( K \) is \( K*k \)
- the bits in the generator vector, from left to right, represent connections in the encoder circuit
  - a ‘1’ represents a link to the shift register, and a ‘0’ represents no link
  - often represented in octal
$k = 1, n = 2, K = 3$ convolutional code

example: generator vectors

\[ g_1 = (101) = (5)_8 \]

\[ g_2 = (111) = (7)_8 \]

octal representation

$k = 2, n = 3, K = 2$ convolutional code

example: generator vectors

\[ g_1 = (1011) = (13)_8 \]

\[ g_2 = (1101) = (15)_8 \]

\[ g_3 = (1010) = (12)_8 \]
**state diagram representation**

- contents of shift register make up “state” of the code
  - most recent input is most significant state bit
- arcs connecting states represent allowable transitions
  - arcs labeled with resulting output bits

\[ g_1 = (101) \quad g_2 = (111) \]

\[ \begin{align*}
0/00 & \rightarrow 0/11 & 0/01 & \rightarrow 0/10 \\
0/10 & \rightarrow 1/00 & 1/00 & \rightarrow 1/01 \\
1/01 & \rightarrow 1/11 & 1/11 & \rightarrow 1/10
\end{align*} \]

input bit / output bits
**trellis representation**

- state diagram unfolds as \( f(\text{time}) \)
  - time increases towards the right

- contents of shift register determine the state of the code (most recent = MSB)

- allowable transitions denoted by connections between states
  - transitions labeled with output bits
  - allowable transitions determine which output bits are possible from a given state

**k = 1, n = 2, K = 3 convolutional code example: trellis**
**k = 1, n = 2, K = 3 encoding example**

start at state 00

<table>
<thead>
<tr>
<th>input</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>00</td>
<td>11</td>
<td>01</td>
<td>00</td>
<td>10</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

**distance structure of convolutional codes**

- the Hamming distance between two distinct code sequences is \( c_1, c_2 \in C \) is the number of bits by which they differ

\[
d_H(c_1, c_2) = \sum_{i=-\infty}^{\infty} c_{1,i} \oplus c_{2,i}
\]
**minimum free distance**

- the minimum free distance of a convolutional code is the smallest Hamming distance separating two distinct code sequences (i.e., two paths through the trellis)

\[ d_{\text{free}} = \min_{i \neq j} \{ d_H(c_i, c_j) \} \]

**searching for good codes**

- would like codes w/ large free distance
  - must avoid catastrophic codes - finite number of errors may cause an infinite number of decoding errors

- generators often found via computer search
  - search is constrained to codes with regular structure
  - search is simplified because of linearity and structure
**identifying catastrophic codes**

- A state diagram having a loop in which a nonzero information sequence corresponds to an all-zero output sequence identifies a catastrophic convolutional code

Examples:

- Rate $\frac{1}{2}$ codes

<table>
<thead>
<tr>
<th>$K$</th>
<th>Generators</th>
<th>$d_{\text{free}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>53</td>
<td>75</td>
</tr>
<tr>
<td>7</td>
<td>133</td>
<td>171</td>
</tr>
<tr>
<td>8</td>
<td>247</td>
<td>371</td>
</tr>
<tr>
<td>9</td>
<td>561</td>
<td>753</td>
</tr>
</tbody>
</table>

Free distance increases with $K$
### rate 1/3 codes

<table>
<thead>
<tr>
<th>( K )</th>
<th>Generators (In Octal)</th>
<th>( d_{\text{free}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5  7  7</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>13 15 17</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>25 33 37</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>47 53 75</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>133 145 171</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>225 331 367</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>557 663 711</td>
<td>18</td>
</tr>
</tbody>
</table>

*free distance increases w/ \( K \)*

### rate 2/3 codes

<table>
<thead>
<tr>
<th>( K )</th>
<th>Generators (In Octal)</th>
<th>( d_{\text{free}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>17 06 15</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>27 75 72</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>236 155 337</td>
<td>7</td>
</tr>
</tbody>
</table>

*free distance increases w/ \( K \)*
summary

- convolutional codes are useful for real-time applications because data can be continuously encoded / decoded
  - frame-based encoding/decoding also possible
- can represent convolutional codes as generators, block diagrams, state diagrams, and trellis diagrams
- want to design convolutional codes that maximize free distance while maintaining non-catastrophic performance