Serial Concatenated Convolutional Codes (SCCCs) with Iterative Decoding

EECS 869: Error Control Coding

Fall 2013

Complete the following tasks. You should submit an e-mail with three .m file attachments (use the e-mail address esp@eecs.ku.edu).

1. **Specify the Trellis Look-up Tables for the Differential Encoder.** In this project, we will be using the differential encoder (DE) shown in Figure 1. This encoder will serve as the "inner code" of a serial concatenated convolutional code (SCCC). The SCCC encoder is shown in Figure 2 (the symbol Π denotes an interleaver). The "outer code" for the SCCC system is the now-familiar (5,7) convolutional code (CC). We already have the trellis look-up tables for the outer code from a previous project. We need this information for the inner code as well. In the file `RunScccSimulationTemplate.m`, you will find a "TODO" for specifying the trellis of the DE.

2. **Convert r into P(c;I).** In order to do this, you need to specify the values of N_0 and $L_c = 2\sqrt{E_c}/\sigma^2$ in your code. With these quantities defined, we have

$$P_{nt+j}(c_t^{(j)};\mathrm{I}) = \frac{1}{1+\exp(-L_c a_t^{(j)} r_{nt+j})}$$

   There is a "TODO" in the file `RunScccSimulationTemplate.m` where you should do this.

3. **Implement the SISO Algorithm.** A block diagram of the SCCC decoder is shown in Figure 3. In the e-mail distribution, you already have this implemented in the file `ScccSisoDecoderSoln.m`. Your task then is implement just the SISO algorithm itself, in the file `SisoCcXXX.m`, where XXX is your name. You do not need to run a full BER simulation; however, the template simulation file is set up to do so. You should verify that your full SCCC decoder works properly. An example BER point is that an information block size of 4096 (code block size of 8192) with 10 decoding iterations has a BER of approx. 3×10^{-3} at $E_b/N_0 = 1.4$ dB.

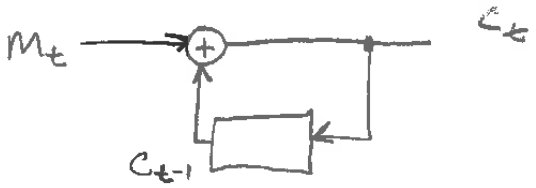

Figure 1: Differential encoder.

Figure 2: Serial concatenated convolutional encoder.

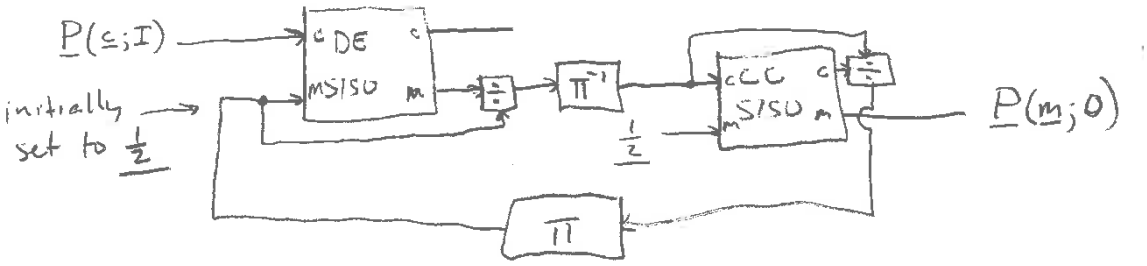

Figure 3: Serial concatenated convolutional decoder.

Algorithm 1 The Soft-Input Soft-Output (SISO) algorithm.

Input: The *a priori* probability distribution for $\mathbf{c} = \{\mathbf{c}_0, \mathbf{c}_1, \cdots, \mathbf{c}_{L-1}\}$, $\mathbf{P}(\mathbf{c}; \mathrm{I})$.
Input: The *a priori* probability distribution for $\mathbf{m} = \{\mathbf{m}_0, \mathbf{m}_1, \cdots, \mathbf{m}_{L-1}\}$, $\mathbf{P}(\mathbf{m}; \mathrm{I})$.
Input: A Boolean flag TERM to indicate if the trellis is terminated.
Input: Trellis look-up tables indexed by the generic edge index e: $s^{\mathrm{S}}(e)$, $\mathbf{m}(e)$, $\mathbf{c}(e)$, and $s^{\mathrm{E}}(e)$.
Outputs: The *a posteriori* probability distributions $\mathbf{P}(\mathbf{c}; \mathrm{O})$ and $\mathbf{P}(\mathbf{m}; \mathrm{O})$.
Initialization:
if TERM == TRUE **then**,
    $A_{-1}(0) = 1$; and $A_{-1}(s) = 0$ for $1 \le s \le N_{\mathrm{s}} - 1$;
    $B_{L-1}(0) = 1$; and $B_{L-1}(s) = 0$ for $1 \le s \le N_{\mathrm{s}} - 1$;
else
    $A_{-1}(s) = \frac{1}{N_{\mathrm{s}}}$ for $0 \le s \le N_{\mathrm{s}} - 1$;
    $B_{L-1}(s) = \frac{1}{N_{\mathrm{s}}}$ for $0 \le s \le N_{\mathrm{s}} - 1$;
end if;
Main Algorithm:
for $t = 0, 1, \cdots, L-2$ **do** // Forward recursion
    for $s = 0, 1, \cdots, N_{\mathrm{s}} - 1$ **do**
        $A_t(s) = \sum_{e: s^{\mathrm{E}}(e) = s} A_{t-1}(s^{\mathrm{S}}(e)) P_t(\mathbf{m}(e); \mathrm{I}) P_t(\mathbf{c}(e); \mathrm{I})$;
    end for
    Normalize $\{A_t(s)\}_{s=0}^{N_{\mathrm{s}}-1}$ so that it sums to unity;
end for
for $t = L-2, \cdots, 1, 0$ **do** // Backward recursion
    for $s = 0, 1, \cdots, N_{\mathrm{s}} - 1$ **do**
        $B_t(s) = \sum_{e: s^{\mathrm{S}}(e) = s} B_{t+1}(s^{\mathrm{E}}(e)) P_{t+1}(\mathbf{m}(e); \mathrm{I}) P_{t+1}(\mathbf{c}(e); \mathrm{I})$;
    end for
    Normalize $\{B_t(s)\}_{s=0}^{N_{\mathrm{s}}-1}$ so that it sums to unity;
end for
for $t = 0, 1, \cdots, L-1$ **do** // Completion Step
    for $j = 0, 1, \cdots, n-1$ **do** // Compute $\mathbf{P}(\mathbf{c}; \mathrm{O})$
        $P_{nt+j}(0; \mathrm{O}) = \sum_{e: c^{(j)}(e) = 0} A_{t-1}(s^{\mathrm{S}}(e)) P_t(\mathbf{m}(e); \mathrm{I}) P_t(\mathbf{c}(e); \mathrm{I}) B_t(s^{\mathrm{E}}(e))$;
        $P_{nt+j}(1; \mathrm{O}) = \sum_{e: c^{(j)}(e) = 1} A_{t-1}(s^{\mathrm{S}}(e)) P_t(\mathbf{m}(e); \mathrm{I}) P_t(\mathbf{c}(e); \mathrm{I}) B_t(s^{\mathrm{E}}(e))$;
        Normalize $\{P_{nt+j}(c^{(j)}; \mathrm{O})\}_{c^{(j)}=0}^{1}$ so that it sums to unity;
    end for
    for $j = 0, 1, \cdots, k-1$ **do** // Compute $\mathbf{P}(\mathbf{m}; \mathrm{O})$
        $P_{kt+j}(0; \mathrm{O}) = \sum_{e: m^{(j)}(e) = 0} A_{t-1}(s^{\mathrm{S}}(e)) P_t(\mathbf{m}(e); \mathrm{I}) P_t(\mathbf{c}(e); \mathrm{I}) B_t(s^{\mathrm{E}}(e))$;
        $P_{kt+j}(1; \mathrm{O}) = \sum_{e: m^{(j)}(e) = 1} A_{t-1}(s^{\mathrm{S}}(e)) P_t(\mathbf{m}(e); \mathrm{I}) P_t(\mathbf{c}(e); \mathrm{I}) B_t(s^{\mathrm{E}}(e))$;
        Normalize $\{P_{kt+j}(u^{(j)}; \mathrm{O})\}_{m^{(j)}=0}^{1}$ so that it sums to unity;
    end for
end for

Algorithm 2 The Soft-Input Soft-Output (SISO) algorithm with detailed look-up tables.

Input: The *a priori* probability distribution for $\mathbf{c} = \{\mathbf{c}_0, \mathbf{c}_1, \cdots, \mathbf{c}_{L-1}\}$, $\mathbf{P}(\mathbf{c}; \mathrm{I})$.

Input: The *a priori* probability distribution for $\mathbf{m} = \{\mathbf{m}_0, \mathbf{m}_1, \cdots, \mathbf{m}_{L-1}\}$, $\mathbf{P}(\mathbf{m}; \mathrm{I})$.

Input: A Boolean flag TERM to indicate if the trellis is terminated.

Input: Trellis look-up tables: $s^{\mathrm{S}}(e^{\mathrm{R}})$, $\mathbf{m}(e^{\mathrm{R}})$, $\mathbf{c}(e^{\mathrm{R}})$, $s^{\mathrm{E}}(e^{\mathrm{R}})$; $\mathbf{m}(e^{\mathrm{L}})$, $\mathbf{c}(e^{\mathrm{L}})$, $s^{\mathrm{E}}(e^{\mathrm{L}})$; and $\{c_1^{(j)}\}_{j=0}^{n-1}$, $\{c_0^{(j)}\}_{j=0}^{n-1}$, $\{m_1^{(j)}\}_{j=0}^{k-1}$, $\{m_0^{(j)}\}_{j=0}^{k-1}$, which contain values of e^{R}.

Outputs: The *a posteriori* probability distributions $\mathbf{P}(\mathbf{c}; \mathrm{O})$ and $\mathbf{P}(\mathbf{m}; \mathrm{O})$.

Initialization:

if TERM == TRUE **then**,

  $A_{-1}(0) = 1$; and $A_{-1}(s) = 0$ for $1 \le s \le N_{\mathrm{s}} - 1$;

  $B_{L-1}(0) = 1$; and $B_{L-1}(s) = 0$ for $1 \le s \le N_{\mathrm{s}} - 1$;

else

  $A_{-1}(s) = \frac{1}{N_{\mathrm{s}}}$ for $0 \le s \le N_{\mathrm{s}} - 1$;

  $B_{L-1}(s) = \frac{1}{N_{\mathrm{s}}}$ for $0 \le s \le N_{\mathrm{s}} - 1$;

end if;

Main Algorithm:

for $t = 0, 1, \cdots, L-2$ **do** // Forward recursion

  for $s = 0, 1, \cdots, N_{\mathrm{s}} - 1$ **do**

    $$A_t(s) = \sum_{e^{\mathrm{R}}=2^k s}^{2^k(s+1)-1} A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}}))P_t(\mathbf{m}(e^{\mathrm{R}}); \mathrm{I})P_t(\mathbf{c}(e^{\mathrm{R}}); \mathrm{I});$$

  end for

  Normalize $\{A_t(s)\}_{s=0}^{N_{\mathrm{s}}-1}$ so that it sums to unity;

end for

for $t = L-2, \cdots, 1, 0$ **do** // Backward recursion

  for $s = 0, 1, \cdots, N_{\mathrm{s}} - 1$ **do**

    $$B_t(s) = \sum_{e^{\mathrm{L}}=2^k s}^{2^k(s+1)-1} B_{t+1}(s^{\mathrm{E}}(e^{\mathrm{L}}))P_{t+1}(\mathbf{m}(e^{\mathrm{L}}); \mathrm{I})P_{t+1}(\mathbf{c}(e^{\mathrm{L}}); \mathrm{I});$$

  end for

  Normalize $\{B_t(s)\}_{s=0}^{N_{\mathrm{s}}-1}$ so that it sums to unity;

end for

for $t = 0, 1, \cdots, L-1$ **do** // Completion Step

  for $j = 0, 1, \cdots, n-1$ **do** // Compute $\mathbf{P}(\mathbf{c}; \mathrm{O})$

    $$P_{nt+j}(0; \mathrm{O}) = \sum_{e^{\mathrm{R}} \in c_0^{(j)}} A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}}))P_t(\mathbf{m}(e^{\mathrm{R}}); \mathrm{I})P_t(\mathbf{c}(e^{\mathrm{R}}); \mathrm{I})B_t(s^{\mathrm{E}}(e^{\mathrm{R}}));$$

    $$P_{nt+j}(1; \mathrm{O}) = \sum_{e^{\mathrm{R}} \in c_1^{(j)}} A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}}))P_t(\mathbf{m}(e^{\mathrm{R}}); \mathrm{I})P_t(\mathbf{c}(e^{\mathrm{R}}); \mathrm{I})B_t(s^{\mathrm{E}}(e^{\mathrm{R}}));$$

    Normalize $\{P_{nt+j}(c^{(j)}; \mathrm{O})\}_{c^{(j)}=0}^{1}$ so that it sums to unity;

  end for

  for $j = 0, 1, \cdots, k-1$ **do** // Compute $\mathbf{P}(\mathbf{m}; \mathrm{O})$

    $$P_{kt+j}(0; \mathrm{O}) = \sum_{e^{\mathrm{R}} \in m_0^{(j)}} A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}}))P_t(\mathbf{m}(e^{\mathrm{R}}); \mathrm{I})P_t(\mathbf{c}(e^{\mathrm{R}}); \mathrm{I})B_t(s^{\mathrm{E}}(e^{\mathrm{R}}));$$

    $$P_{kt+j}(1; \mathrm{O}) = \sum_{e^{\mathrm{R}} \in m_1^{(j)}} A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}}))P_t(\mathbf{m}(e^{\mathrm{R}}); \mathrm{I})P_t(\mathbf{c}(e^{\mathrm{R}}); \mathrm{I})B_t(s^{\mathrm{E}}(e^{\mathrm{R}}));$$

    Normalize $\{P_{kt+j}(u^{(j)}; \mathrm{O})\}_{m^{(j)}=0}^{1}$ so that it sums to unity;

  end for

end for