1 Background Material

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. Implement a Function to Create Look-up Tables for the Trellis for Convolutional Codes. You may assume that we are limited to rate $1/n$ feedforward non-systematic codes. You need to create look-up tables that are sorted according to the right-hand edge index, $e^r$; tables are needed for the left-hand edge index, $e^l$, the starting state, $s^s$, the message bit, $m(e)$, the code bits, $c(e)$, and the ending state, $s^e$. You should implement this as a MATLAB function with the following syntax:

\[
[eL, sS, me, ce, sE] = \text{CreateCcTrellisXXX}(G);
\]

where $G$ is a $(v+1) \times n$ matrix that specifies $n$ encoder functions with a constraint length (memory order) of $v$. For the (5,7) code, we have a transfer function matrix

\[
G(D) = \begin{bmatrix} 1 + D^2 & 1 + D + D^2 \end{bmatrix}
\]

and so we specify $G$ in MATLAB as

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

As for the output arguments, each is an $N_e \times 1$ MATLAB vector (you should implement ce as an $N_e \times n$ matrix), where $N_e \times 1$ is the number of edges in the trellis. Because these elements are in order of increasing right-hand edge index, the specification for $e^r$ is implied and does not need to be stated explicitly.

2. Implement the Hard-Decision Viterbi Algorithm for Convolutional Codes. You should implement this as a MATLAB function with the following syntax:

\[
m_{\text{hat}} = \text{VaCcHdXXX}(r, \text{TERM}, sS, me, ce);
\]

where $r$ is a $1 \times nL$ MATLAB vector containing hard decisions from the BSC (0's and 1's) and TERM is a Boolean flag (i.e., a 0 or a 1) to indicate if the trellis is terminated. The metric increment for the hard-decision VA is

\[
\gamma_t(e) = d_h(r, c(e))
\]

where $d_h(\cdot, \cdot)$ denotes Hamming distance. For the (5,7) code, the metric increment is 0, 1, or 2. The objective of the VA is to minimize this metric. The algorithm you need to implement is shown in Algorithm [2]

3. Implement the Soft-Decision Viterbi Algorithm for Convolutional Codes. You should implement this as a MATLAB function with the following syntax:

\[
m_{\text{hat}} = \text{VaCcSdXXX}(r, \text{TERM}, sS, me, ae);
\]
BER Plot for the (5,7) CC with Hard/Soft Decision Decoding

\[ P_b (\text{BER}) \]

- \( P_b \) BPSK (theoretical)
- \( P_b \) HD (simulation)
- \( P_b \) SD (simulation)

\[ \gamma_t(e) = \sum_{i=0}^{n-1} r_t^{(i)} a^{(i)}(e) = a(e)r_t^T \]

where \( r \) is a \( 1 \times nL \) MATLAB vector containing soft decisions from the AWGN channel (noisy +1’s and -1’s). The metric increment for the soft-decision VA is

where \( a(e) \) is the antipodal version of \( c(e) \) and \((\cdot)^T\) is the transpose operator. Over time, the increments \( r_t a^T(e) \) add up and result in the correlation between \( r \) and \( a \). The objective of the VA is to maximize the correlation, so the appropriate changes must be made to the VA operations. The algorithm you need to implement is shown in Algorithm 5.

4. **BER Simulation.** Verify the correct operation of your VA implementations by running a BER simulation (using the “wrapper file” provided) for the (5,7) code. The expected result is shown above. You should run your simulation over the range of \( E_b/N_0 \) values from 0 to 5 dB.
Algorithm 1 The Viterbi algorithm (VA) for hard-decision inputs.

1: **Input**: The received signal \( r = \{ r_0, r_1, \ldots, r_{L-1} \} \).
2: **Input**: A Boolean flag \( \text{TERM} \) to indicate if the trellis is terminated.
3: **Input**: Trellis look-up tables indexed by the generic edge index \( e: s^k(e), m(e), \) and \( c(e) \).
4: **Output**: The estimated symbol sequence \( \hat{m} = \{ \hat{m}_0, \hat{m}_1, \ldots, \hat{m}_{L-1} \} \).

5: **Initialization**:
6: if \( \text{TERM} == \text{TRUE} \) then,
7: \( A_{-1}(0) = 0; \)
8: \( A_{-1}(s) = +\infty \) for \( 1 \leq s \leq N_s - 1; \)
9: else
10: \( A_{-1}(s) = 0 \) for \( 0 \leq s \leq N_s - 1; \)
11: end if;
12: **Main Algorithm**:
13: // Forward recursion with metric and survivor updates
14: for \( t = 0, 1, \ldots, L - 1 \) do
15: for \( s = 0, 1, \ldots, N_s - 1 \) do
16: \( A_t(s) = \min_{e \in s^k(e) = s} \{ A_{t-1}(s^k(e)) + y_t(e) \}; \)
17: \( T_t(s) = \arg \min_{e \in s^k(e) = s} \{ A_{t-1}(s^k(e)) + y_t(e) \}; \)
18: where \( y_t(e) = d_h(r_t, c(e)); \)
19: end for
20: end for
21: // Final global survivor at the end of the transmission
22: if \( \text{TERM} == \text{TRUE} \) then,
23: \( \hat{s}_{L-1}^F = 0; \)
24: else
25: \( \hat{s}_{L-1}^F = \arg \min_{0 \leq s \leq N_1} \{ A_{L-1}(s) \}; \)
26: end if;
27: // Traceback operation to identify output sequence
28: for \( t = L - 1, \ldots, 1, 0 \) do
29: \( \hat{m}_t = m(T_t(\hat{s}_t^F)); \)
30: \( \hat{s}_{t-1}^F = s^k(T_t(\hat{s}_t^F)); \)
31: end for
Algorithm 2 The Viterbi algorithm (VA) for hard-decision inputs using indexes on the right-hand side of the trellis.

1: Input: The received signal $r = \{r_0, r_1, \ldots, r_{L-1}\}$.
2: Input: A Boolean flag TERM to indicate if the trellis is terminated.
3: Input: Trellis look-up tables indexed by the right-hand edge index $e^R$: $s^R(e^R)$, $m(e^R)$, and $c(e^R)$.
4: Output: The estimated symbol sequence $\hat{m} = \{\hat{m}_0, \hat{m}_1, \ldots, \hat{m}_{L-1}\}$.
5: Initialization:
6: if TERM == TRUE then
7: $A_{-1}(0) = 0$;
8: $A_{-1}(s) = +\infty$ for $1 \leq s \leq N_s - 1$;
9: else
10: $A_{-1}(s) = 0$ for $0 \leq s \leq N_s - 1$;
11: end if;
12: Main Algorithm:
13: // Forward recursion with metric and survivor updates
14: for $t = 0, 1, \ldots, L - 1$ do
15: for $s = 0, 1, \ldots, N_s - 1$ do
16: $A_t(s) = \min_{2^k \leq e^R \leq 2^k + 2^k - 1} \{A_{t-1}(s + e^R) + y_t(e^R)\}$;
17: $T_t(s) = \arg \min_{2^k \leq e^R \leq 2^k + 2^k - 1} \{A_{t-1}(s + e^R) + y_t(e^R)\}$;
18: where $y_t(e^R) = d_H(r_t, c(e^R))$;
19: end for
20: end for
21: // Final global survivor at the end of the transmission
22: if TERM == TRUE then
23: $\hat{s}_{L-1}^R = 0$;
24: else
25: $\hat{s}_{L-1}^R = \arg \min_{0 \leq s \leq N_s - 1} \{A_{L-1}(s)\}$;
26: end if;
27: // Traceback operation to identify output sequence
28: for $t = L - 1, \ldots, 1, 0$ do
29: $\hat{m}_t = m(T_t(\hat{s}_t^R))$;
30: $\hat{s}_{t-1}^R = s^R(T_t(\hat{s}_t^R))$;
31: end for
Algorithm 3 The Viterbi algorithm (VA) for soft-decision inputs using indexes on the right-hand side of the trellis.

1: Input: The received signal \( r = \{ r_0, r_1, \cdots, r_{L-1} \} \).
2: Input: A Boolean \( \text{TERM} \) to indicate if the trellis is terminated.
3: Input: Trellis look-up tables indexed by the right-hand edge index \( e^R \): \( s^E(e^R) \), \( m(e^R) \), and \( a(e^R) \).
4: Output: The estimated symbol sequence \( \hat{\mathbf{m}} = \{ \hat{m}_0, \hat{m}_1, \cdots, \hat{m}_{L-1} \} \).
5: Initialization:
6: if \( \text{TERM} == \text{TRUE} \) then,
7: \( A_{-1}(0) = 0 \);
8: \( A_{-1}(s) = -\infty \) for \( 1 \leq s \leq N_s - 1 \);
9: else
10: \( A_{-1}(s) = 0 \) for \( 0 \leq s \leq N_s - 1 \);
11: end if;
12: Main Algorithm:
13: // Forward recursion with metric and survivor updates
14: for \( t = 0, 1, \cdots, L - 1 \) do
15: for \( s = 0, 1, \cdots, N_s - 1 \) do
16: \( A_t(s) = \max_{2^k s \leq e^R \leq 2^k(s+1)-1} \{ A_{t-1}(s^E(e^R)) + \gamma_t(e^R) \} \);
17: \( T_t(s) = \arg \max_{2^k s \leq e^R \leq 2^k(s+1)-1} \{ A_{t-1}(s^E(e^R)) + \gamma_t(e^R) \} \);
18: where \( \gamma_t(e^R) = \sum_{i=0}^{N_s-1} r_t^{(i)} a^{(i)}(e^R) = a(e^R)e_t^T \);
19: end for
20: end for
21: // Final global survivor at the end of the transmission
22: if \( \text{TERM} == \text{TRUE} \) then,
23: \( \hat{s}_{L-1}^E = 0 \);
24: else
25: \( \hat{s}_{L-1}^E = \arg \max_{0 \leq s \leq N_s - 1} \{ A_{L-1}(s) \} \);
26: end if;
27: // Traceback operation to identify output sequence
28: for \( t = L - 1, \cdots, 1, 0 \) do
29: \( \hat{m}_t = m(T_t(\hat{s}_t^E)) \);
30: \( \hat{s}_{t-1}^E = s^E(T_t(\hat{s}_t^E)) \);
31: end for