# The Viterbi Algorithm
## EECS 869: Error Control Coding
## Fall 2017

## 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 left-hand edge index, $e^{\mathrm{L}}$; tables are needed for the starting state, $s^{\mathrm{S}}(e^{\mathrm{L}})$, the message bit, $\mathbf{m}(e^{\mathrm{L}})$, the code bits, $\mathbf{c}(e^{\mathrm{L}})$, the ending state, $s^{\mathrm{E}}(e^{\mathrm{L}})$, and the right-hand edge index, $e^{\mathrm{R}}(e^{\mathrm{L}})$. Additionally, one look-up table is required to be sorted according to the *right-hand* edge index; this table is the left-hand edge index, $e^{\mathrm{L}}(e^{\mathrm{R}})$ (so that the algorithm can trace backwards).

   You should implement this as a MATLAB function with the following syntax:

   ```
   [sS, me, ce, sE, eR, eL_eR] = CreateCcTrellisXXX(G);
   ```

   where $\mathbf{G}$ 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

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

   and so we specify $\mathbf{G}$ in MATLAB as $\mathbf{G} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$.

   As for the output arguments, each is an $N_{\mathrm{E}} \times 1$ MATLAB vector (except for `ce`, which is an $N_{\mathrm{E}} \times n$ matrix), where $N_{\mathrm{E}}$ is the number of edges in the trellis. Because these elements are in order of increasing left-hand edge index, the specification for $e^{\mathrm{L}}$ 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_hat = VaCcHdXXX(r, TERM, sS, me, ce, eR, eL_eR);
   ```

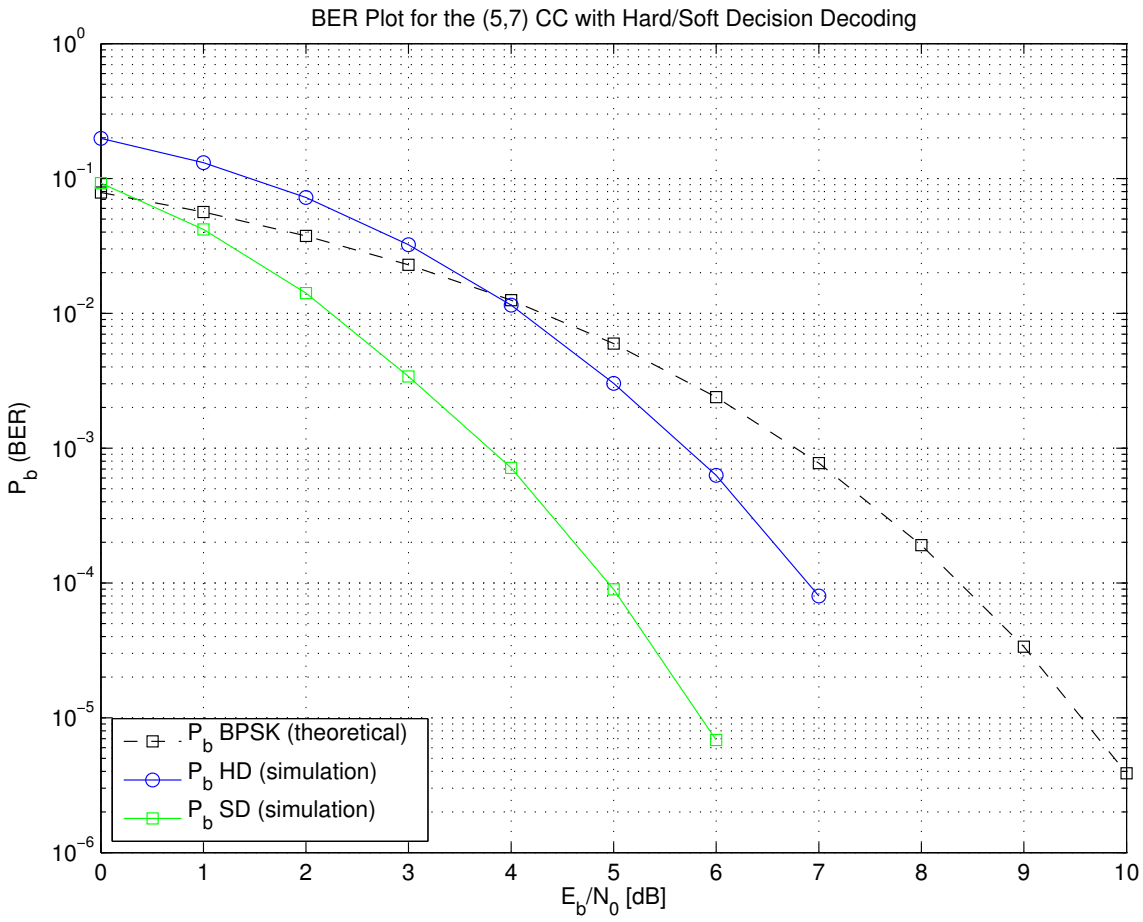
   where $\mathbf{r}$ is a $1 \times nL$ MATLAB vector containing hard decisions from the BSC (i.e., 0's and 1's) and TERM is a Boolean flag (i.e., a 0 or a 1) to indicate if the trellis is terminated. The metric increment for the hard-decision VA is

   $$\gamma_t(e) = d_{\mathrm{H}}\big(\mathbf{r}_t, \mathbf{c}(e)\big)$$

   where $d_{\mathrm{H}}(\cdot,\cdot)$ 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 3.

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

   ```
   m_hat = VaCcSdXXX(r, TERM, sS, me, ae, eR, eL_eR);
   ```




where $\mathbf{r}$ is a $1 \times nL$ MATLAB vector containing soft decisions from the AWGN channel (i.e., noisy +1's and −1's). The metric increment for the soft-decision VA is

$$\gamma_t(e) = \sum_{i=0}^{n-1} r_t^{(i)} a^{(i)}(e) = \mathbf{a}(e)\mathbf{r}_t^T$$

where $\mathbf{a}(e) = 2\mathbf{c}(e) - 1$ is the antipodal version of $\mathbf{c}(e)$ and $(\cdot)^T$ is the transpose operator. Over time, the increments $\mathbf{r}_t\mathbf{a}^T(e)$ add up and result in the correlation between $\mathbf{r}$ and $\mathbf{a}$. 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.

5. **Channel Capacity.** Plot the (5,7) code performance by adding two data points to your capacity plot, one for hard decision decoding and one for soft decision decoding.

**Algorithm 1** The Viterbi algorithm (VA) for hard-decision inputs.

**Input:** The hard-decision received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{T-1}\}$.
**Input:** A Boolean flag TERM to indicate if the trellis is terminated.
**Output:** The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_{T-1}\}$.
**Initialization:**
**if** TERM == TRUE **then**,
    $A_{-1}(0) = 0$; and $A_{-1}(s) = +\infty$ for $1 \le s \le N_{\mathrm{S}} - 1$;
**else**
    $A_{-1}(s) = 0$ for $0 \le s \le N_{\mathrm{S}} - 1$;
**end if**;
**Main Algorithm:**
The edge metric is $\gamma_t(e) = d_{\mathrm{H}}\big(\mathbf{r}_t, \mathbf{c}(e)\big)$ for all $t \in \{0, 1, \ldots, T-1\}$ and $e \in \{0, 1, \ldots, N_{\mathrm{E}} - 1\}$
**for** $t = 0, 1, \ldots, T-1$ **do** // Forward recursion with metric and survivor updates
    **for** $s = 0, 1, \ldots, N_{\mathrm{S}} - 1$ **do**
        $A_t(s) = \min\limits_{e: s^{\mathrm{E}}(e) = s} \big\{A_{t-1}(s^{\mathrm{S}}(e)) + \gamma_t(e)\big\}$;
        $T_t(s) = \arg\min\limits_{e: s^{\mathrm{E}}(e) = s} \big\{A_{t-1}(s^{\mathrm{S}}(e)) + \gamma_t(e)\big\}$;
    **end for**
**end for**
**if** TERM == TRUE **then**, // Final global survivor at the end of the transmission
    $\hat{s}^{\mathrm{E}}_{T-1} = 0$;
**else**
    $\hat{s}^{\mathrm{E}}_{T-1} = \arg\min\limits_{0 \le s \le N_{\mathrm{S}} - 1} \{A_{T-1}(s)\}$;
**end if**;
**for** $t = T-1, \ldots, 1, 0$ **do** // Traceback operation to identify output sequence
    $\hat{\mathbf{m}}_t = \mathbf{m}\big(T_t(\hat{s}^{\mathrm{E}}_t)\big)$;
    $\hat{s}^{\mathrm{E}}_{t-1} = s^{\mathrm{S}}\big(T_t(\hat{s}^{\mathrm{E}}_t)\big)$;
**end for**

**Algorithm 2** The Viterbi algorithm (VA) for soft-decision inputs.

**Input:** The soft-decision received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{T-1}\}$.
**Input:** A Boolean flag TERM to indicate if the trellis is terminated.
**Output:** The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_{T-1}\}$.
**Initialization:**
**if** TERM == TRUE **then**,
    $A_{-1}(0) = 0$; and $A_{-1}(s) = -\infty$ for $1 \le s \le N_{\mathrm{s}} - 1$;
**else**
    $A_{-1}(s) = 0$ for $0 \le s \le N_{\mathrm{s}} - 1$;
**end if**;
**Main Algorithm:**
The edge metric is $\gamma_t(e) = \sum_{i=0}^{n-1} r_t^{(i)} a^{(i)}(e) = \mathbf{a}(e)\mathbf{r}_t^T$ for all $t \in \{0, 1, \ldots, T-1\}$ and $e \in \{0, 1, \ldots, N_{\mathrm{E}} - 1\}$
**for** $t = 0, 1, \ldots, T-1$ **do** // Forward recursion with metric and survivor updates
    **for** $s = 0, 1, \ldots, N_{\mathrm{s}} - 1$ **do**
        $A_t(s) = \max\limits_{e: s^{\mathrm{E}}(e) = s} \left\{ A_{t-1}(s^{\mathrm{s}}(e)) + \gamma_t(e) \right\}$;
        $T_t(s) = \arg\max\limits_{e: s^{\mathrm{E}}(e) = s} \left\{ A_{t-1}(s^{\mathrm{s}}(e)) + \gamma_t(e) \right\}$;
    **end for**
**end for**
**if** TERM == TRUE **then**, // Final global survivor at the end of the transmission
    $\hat{s}_{T-1}^{\mathrm{E}} = 0$;
**else**
    $\hat{s}_{T-1}^{\mathrm{E}} = \arg\max\limits_{0 \le s \le N_{\mathrm{s}} - 1} \{ A_{T-1}(s) \}$;
**end if**;
**for** $t = T-1, \ldots, 1, 0$ **do** // Traceback operation to identify output sequence
    $\hat{\mathbf{m}}_t = \mathbf{m}(T_t(\hat{s}_t^{\mathrm{E}}))$;
    $\hat{s}_{t-1}^{\mathrm{E}} = s^{\mathrm{s}}(T_t(\hat{s}_t^{\mathrm{E}}))$;
**end for**

**Algorithm 3** The VA for hard-decision inputs using indexes for the left- and right-hand sides of the trellis.

**Input:** The hard-decision received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{T-1}\}$.
**Input:** A Boolean flag TERM to indicate if the trellis is terminated.
**Output:** The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_{T-1}\}$.
**Initialization:**
**if** TERM == TRUE **then**,
    $A_{-1}(0) = 0$; and $A_{-1}(s^{\mathrm{E}}) = +\infty$ for $1 \le s^{\mathrm{E}} \le N_{\mathrm{S}} - 1$;
**else**
    $A_{-1}(s^{\mathrm{E}}) = 0$ for $0 \le s^{\mathrm{E}} \le N_{\mathrm{S}} - 1$;
**end if**;
**Main Algorithm:**
The edge metric is $\gamma_t(e^{\mathrm{L}}) = d_{\mathrm{H}}\left(\mathbf{r}_t, \mathbf{c}(e^{\mathrm{L}})\right)$ for all $t \in \{0, 1, \ldots, T-1\}$ and $e \in \{0, 1, \ldots, N_{\mathrm{E}} - 1\}$
**for** $t = 0, 1, \ldots, T-1$ **do** // Forward recursion with metric and survivor updates
    **for** $e^{\mathrm{L}} = 0, 1, \ldots, N_{\mathrm{E}} - 1$ **do** // Compute the edge metrics, store them in right-edge order
        $M_t(e^{\mathrm{R}}(e^{\mathrm{L}})) = A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{L}})) + \gamma_t(e^{\mathrm{L}})\}$;
    **end for**
    **for** $s^{\mathrm{E}} = 0, 1, \ldots, N_{\mathrm{S}} - 1$ **do** // Determine survivors at the ending states
        $A_t(s^{\mathrm{E}}) = \min_{2^k s^{\mathrm{E}} \le e^{\mathrm{R}} \le 2^k(s^{\mathrm{E}}+1)-1} \{M_t(e^{\mathrm{R}})\}$;
        $T_t(s^{\mathrm{E}}) = \arg\min_{2^k s^{\mathrm{E}} \le e^{\mathrm{R}} \le 2^k(s^{\mathrm{E}}+1)-1} \{M_t(e^{\mathrm{R}})\}$;
    **end for**
**end for**
**if** TERM == TRUE **then**, // Final global survivor at the end of the transmission
    $\hat{s}^{\mathrm{E}}_{T-1} = 0$;
**else**
    $\hat{s}^{\mathrm{E}}_{T-1} = \arg\min_{0 \le s^{\mathrm{E}} \le N_{\mathrm{S}}-1} \{A_{T-1}(s^{\mathrm{E}})\}$;
**end if**;
**for** $t = T-1, \ldots, 1, 0$ **do** // Traceback operation to identify output sequence
    $\hat{e}^{\mathrm{L}}_t = e^{\mathrm{L}}(T_t(\hat{s}^{\mathrm{E}}_t))$;
    $\hat{\mathbf{m}}_t = \mathbf{m}(\hat{e}^{\mathrm{L}}_t)$;
    $\hat{s}^{\mathrm{E}}_{t-1} = s^{\mathrm{S}}(\hat{e}^{\mathrm{L}}_t)$;
**end for**

**Algorithm 4** The VA for soft-decision inputs using indexes for the left- and right-hand sides of the trellis.

**Input:** The soft-decision received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{T-1}\}$.
**Input:** A Boolean flag TERM to indicate if the trellis is terminated.
**Output:** The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_{T-1}\}$.
**Initialization:**
**if** TERM == TRUE **then**,
    $A_{-1}(0) = 0$; and $A_{-1}(s^{\mathrm{E}}) = -\infty$ for $1 \le s^{\mathrm{E}} \le N_{\mathrm{S}} - 1$;
**else**
    $A_{-1}(s^{\mathrm{E}}) = 0$ for $0 \le s^{\mathrm{E}} \le N_{\mathrm{S}} - 1$;
**end if**;
**Main Algorithm:**
The edge metric is $\gamma_t(e^{\mathrm{L}}) = \sum_{i=0}^{n-1} r_t^{(i)} a^{(i)}(e^{\mathrm{L}}) = \mathbf{a}(e^{\mathrm{L}})\mathbf{r}_t^T$ for all $t \in \{0, 1, \ldots, T-1\}$ and $e \in \{0, 1, \ldots, N_{\mathrm{E}}-1\}$
**for** $t = 0, 1, \ldots, T-1$ **do** // Forward recursion with metric and survivor updates
    **for** $e^{\mathrm{L}} = 0, 1, \ldots, N_{\mathrm{E}} - 1$ **do** // Compute the edge metrics, store them in right-edge order
        $M_t(e^{\mathrm{R}}(e^{\mathrm{L}})) = A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{L}})) + \gamma_t(e^{\mathrm{L}})\}$;
    **end for**
    **for** $s^{\mathrm{E}} = 0, 1, \ldots, N_{\mathrm{S}} - 1$ **do** // Determine survivors at the ending states
        $A_t(s^{\mathrm{E}}) = \max_{2^k s^{\mathrm{E}} \le e^{\mathrm{R}} \le 2^k(s^{\mathrm{E}}+1)-1} \{M_t(e^{\mathrm{R}})\}$;
        $T_t(s^{\mathrm{E}}) = \arg\max_{2^k s^{\mathrm{E}} \le e^{\mathrm{R}} \le 2^k(s^{\mathrm{E}}+1)-1} \{M_t(e^{\mathrm{R}})\}$;
    **end for**
**end for**
**if** TERM == TRUE **then**, // Final global survivor at the end of the transmission
    $\hat{s}_{T-1}^{\mathrm{E}} = 0$;
**else**
    $\hat{s}_{T-1}^{\mathrm{E}} = \arg\max_{0 \le s^{\mathrm{E}} \le N_{\mathrm{S}}-1} \{A_{T-1}(s^{\mathrm{E}})\}$;
**end if**;
**for** $t = T-1, \ldots, 1, 0$ **do** // Traceback operation to identify output sequence
    $\hat{e}_t^{\mathrm{L}} = e^{\mathrm{L}}(T_t(\hat{s}_t^{\mathrm{E}}))$;
    $\hat{\mathbf{m}}_t = \mathbf{m}(\hat{e}_t^{\mathrm{L}})$;
    $\hat{s}_{t-1}^{\mathrm{E}} = s^{\mathrm{S}}(\hat{e}_t^{\mathrm{L}})$;
**end for**

**Algorithm 5** The VA for soft-decision inputs with finite traceback.

**Input:** The soft-decision received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{T-1}\}$.
**Input:** An integer $\delta$ to specify the finite traceback length.
**Input:** A Boolean flag TERM to indicate if the trellis is terminated.
**Output:** The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_{T-1}\}$.
**Initialization:**
**if** TERM == TRUE **then**,
    $A_{-1}(0) = 0$; and $A_{-1}(s^{\mathrm{E}}) = -\infty$ for $1 \le s^{\mathrm{E}} \le N_{\mathrm{S}} - 1$;
**else**
    $A_{-1}(s^{\mathrm{E}}) = 0$ for $0 \le s^{\mathrm{E}} \le N_{\mathrm{S}} - 1$;
**end if**;
**Main Algorithm:**
The edge metric is $\gamma_t(e^{\mathrm{L}}) = \sum_{i=0}^{n-1} r_t^{(i)} a^{(i)}(e^{\mathrm{L}}) = \mathbf{a}(e^{\mathrm{L}})\mathbf{r}_t^T$ for all $t \in \{0, 1, \ldots, T-1\}$ and $e \in \{0, 1, \ldots, N_{\mathrm{E}}-1\}$
**for** $t = 0, 1, \ldots, T-1$ **do** // Forward recursion with metric and survivor updates
    **for** $e^{\mathrm{L}} = 0, 1, \ldots, N_{\mathrm{E}} - 1$ **do** // Compute the edge metrics, store them in right-edge order
        $M_t(e^{\mathrm{R}}(e^{\mathrm{L}})) = A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{L}})) + \gamma_t(e^{\mathrm{L}})\}$;
    **end for**
    **for** $s^{\mathrm{E}} = 0, 1, \ldots, N_{\mathrm{S}} - 1$ **do** // Determine survivors at the ending states
        $A_t(s^{\mathrm{E}}) = \max_{2^k s^{\mathrm{E}} \le e^{\mathrm{R}} \le 2^k(s^{\mathrm{E}}+1)-1} \{M_t(e^{\mathrm{R}})\}$;
        $T_t(s^{\mathrm{E}}) = \arg\max_{2^k s^{\mathrm{E}} \le e^{\mathrm{R}} \le 2^k(s^{\mathrm{E}}+1)-1} \{M_t(e^{\mathrm{R}})\}$;
    **end for**
    **if** $t \ge \delta$ **then**, // Traceback operation to identify the output symbol with delay $\delta$
        $\hat{s}_t^{\mathrm{E}} = \arg\max_{0 \le s^{\mathrm{E}} \le N_{\mathrm{S}}-1} \{A_t(s^{\mathrm{E}})\}$;
        **for** $t' = t, t-1, \ldots, t-\delta+1$ **do**
            $\hat{s}_{t'-1}^{\mathrm{E}} = s^{\mathrm{S}}(e^{\mathrm{L}}(T_{t'}(\hat{s}_{t'}^{\mathrm{E}})))$;
        **end for**
        $\hat{\mathbf{m}}_{t-\delta} = \mathbf{m}(e^{\mathrm{L}}(T_{t-\delta}(\hat{s}_{t-\delta}^{\mathrm{E}})))$;
    **end if**;
**end for**
**if** TERM == TRUE **then**, // Final global survivor at the end of the transmission
    $\hat{s}_{T-1}^{\mathrm{E}} = 0$;
**else**
    $\hat{s}_{T-1}^{\mathrm{E}} = \arg\max_{0 \le s^{\mathrm{E}} \le N_{\mathrm{S}}-1} \{A_{T-1}(s^{\mathrm{E}})\}$;
**end if**;
**for** $t = T-1, T-2, \ldots, T-\delta$ **do** // Complete the traceback up to the most recent symbol
    $\hat{e}_t^{\mathrm{L}} = e^{\mathrm{L}}(T_t(\hat{s}_t^{\mathrm{E}}))$;
    $\hat{\mathbf{m}}_t = \mathbf{m}(\hat{e}_t^{\mathrm{L}})$;
    $\hat{s}_{t-1}^{\mathrm{E}} = s^{\mathrm{S}}(\hat{e}_t^{\mathrm{L}})$;
**end for**