ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

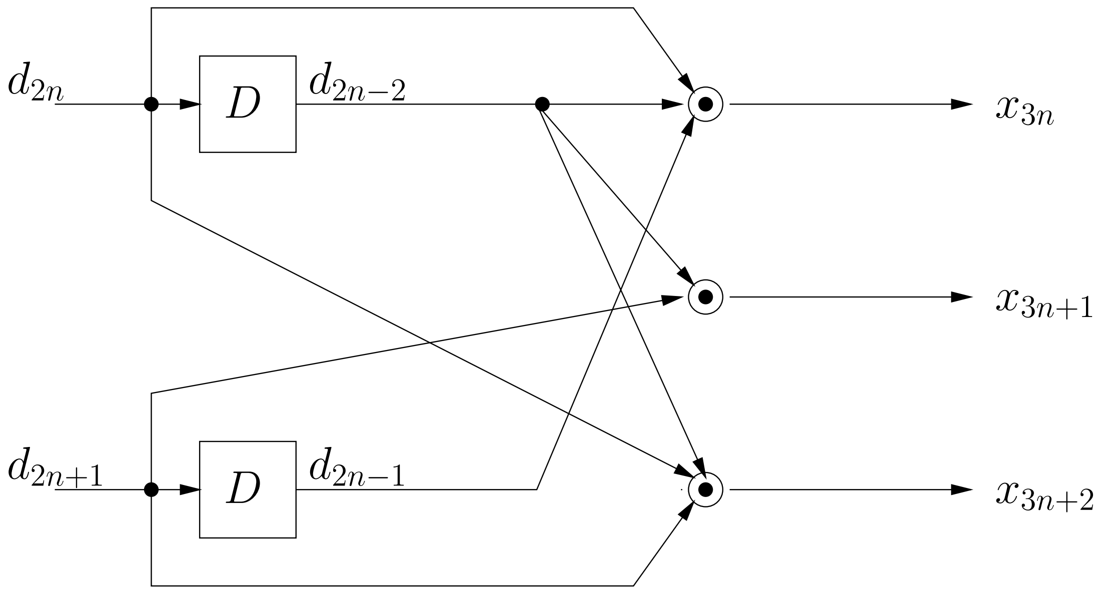
Principles of Digital Communications: Summer Semester 2012
Assignment date: May 16, 2012
Due date: May 23, 2012

Solution of Homework 12

Problem 1. It's about the project.

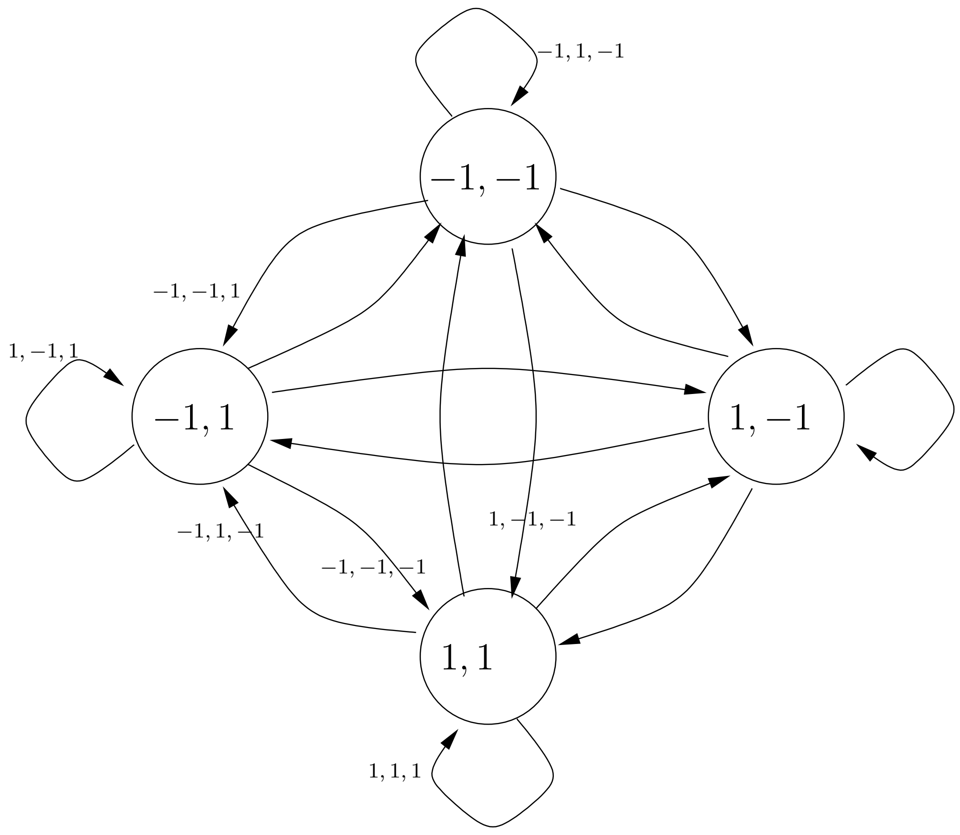
Problem 2. *(Convolutional Code)*

1. An implementation of the encoder will be as follows:



2. The rate of this code is 2/3: Two inputs produce three outputs.

3. The state diagram is very similar to the one considered in class. The difference is that now, *every* state is accessible from every other state. We use the following terminology: the state label is a, b, where a is the "state of the even sub-sequence", i.e. contains d_{2n-2}, and b is the "state of the odd sub-sequence", i.e. contains d_{2n-1}. On the arrows, we only mark the outputs; the input required to make a particular transition is simply the next state, therefore we omitted it. Take for example the arrow linking state $1, 1$ to state $-1, 1$. It is clear that the inputs have to be $d_{2n} = -1$ and $d_{2n+1} = 1$. The corresponding output in order $x_{3n}, x_{3n+1}, x_{3n+2}$ are marked on the branch: $-1, 1 - 1$. We only give a few of the labels in the figure below; it should be easy to find the rest.



4. Almost everything remains the same. The formula as derived in class is an upper bound on the number of errors per section of the trellis (which you can verify by going through the derivation). When there is only one data bit per section of the trellis, this is the same as the bit error probability. However, when there are multiple bits per section of the trellis, the formula is

$$P_b \;\leq\; \frac{1}{K_0}\,\frac{\delta T(D,I)}{\delta I}\bigg|_{I=1,D=Z}, \tag{1}$$

where $Z = e^{-\frac{E_s}{N_0}}$ and K_0 is the number of inputs per section of the trellis. In our example, $K_0 = 2$. Since there are three channel symbols per two source symbol, we find that $E_s = 2E_b/3$.

Problem 3. *(Convolutional Encoder, Decoder and Error Probability)*

1. Convolutional Encoder

   (a) The rate is

$$R = \frac{\#\text{ input bits}}{\#\text{ output bits}} = \frac{1}{2}.$$

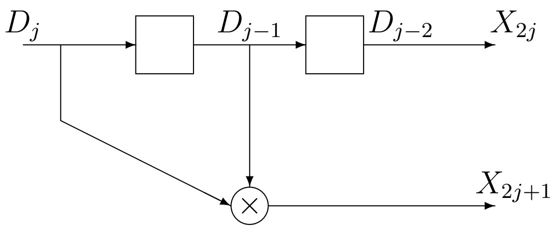
(b) Since the state is (D_{j-1}, D_{j-2}), we need two shift registers. From the finite state machine, we can derive a table that relates the state (D_{j-1}, D_{j-2}) and the current input D_j with the two outputs (X_{2j}, X_{2j+1}).

D_j	D_{j-1}	D_{j-2}	X_{2j}	X_{2j+1}
1	1	1	1	1
1	1	-1	-1	1
1	-1	1	1	-1
1	-1	-1	-1	-1
-1	1	1	1	-1
-1	1	-1	-1	-1
-1	-1	1	1	1
-1	-1	-1	-1	1

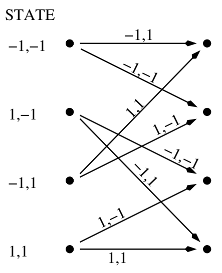
Now, we look for rows in the table that contain a single "-1". The second row has this property. Since in the second row, $D_{j-2} = -1$ is the only negative entry, the outputs that are "+1" (X_{2j+1} in this case) cannot contain D_{j-2} in their product. On the other hand, the outputs that are "-1" (X_{2j}) must contain D_{j-2} in their product. We repeat this argument, for instance using the third row, where $D_{j-1} = -1$ is the only negative entry. Looking at the outputs of the third row, we conclude that X_{2j} cannot contain D_{j-1}, but that X_{2j+1} must contain D_{j-1}. Repeating this argument once more for the fifth row, we find that

$$\begin{aligned} X_{2j} &= D_{j-2} \\ X_{2j} &= D_j \cdot D_{j-1}. \end{aligned}$$

We obtain the diagram below:

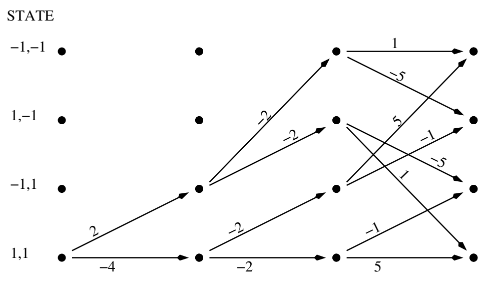


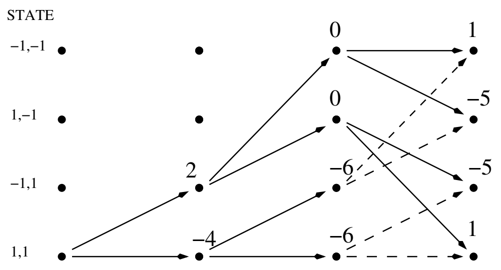
(c) The trellis section is as follows:

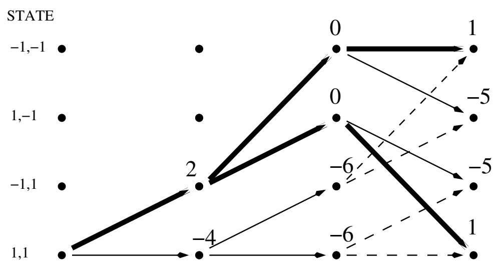


2. We have:

(a) We need a trellis of length $\frac{6}{2} = 3$. All the steps of the algorithm are given below.



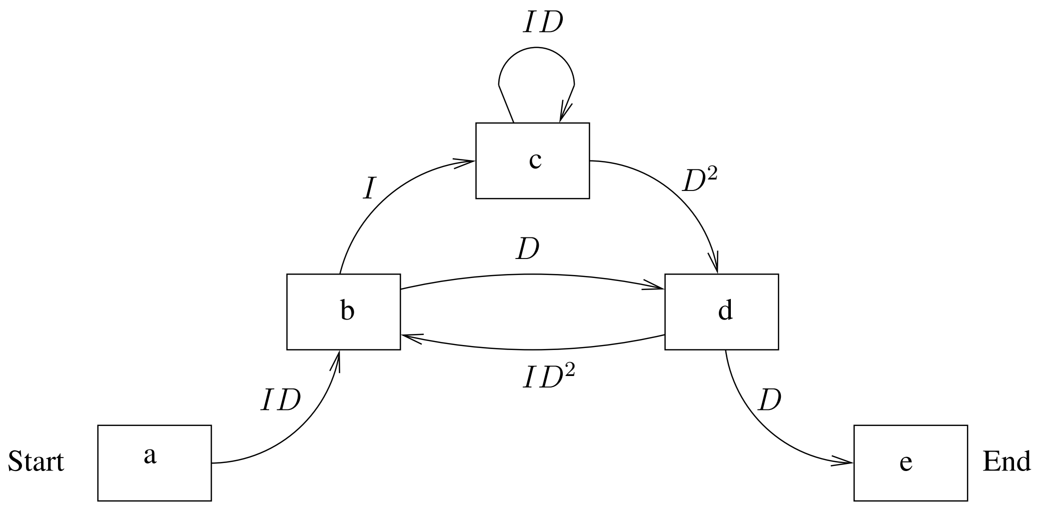




Note that there are two possible optimal paths. Hence, the decoder has to randomly decide for one of the paths. This means that the two corresponding input sequences $(-1, -1, -1)$ and $(-1, 1, 1)$ are equally likely given the output sequence $(-1, -3, -2, 0, 2, 3)$.

3. For the performance analysis of the code we have:

(a) The detour flow graph (with respect to the all-one sequence) is given below:



(b) Note that the generating function given here does not correspond to the trellis / detour flow graph considered in the rest of the problem.

$$\begin{aligned} T(I,D) &= \frac{ID^4}{1-3ID} \\ &= ID^4\big[1+3ID+(3ID)^2+\dots \\ &= ID^4+3I^2D^5+9I^3D^6+27I^4D^7+\dots \end{aligned}$$

Hence, $a(1,4)=1$, $a(2,5)=3$, $a(3,6)=9$, etc... Therefore, we see that

$$a(i,d)=\begin{cases} 3^{i-1} & \text{if } d=i+3 \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the value of the sum is

$$\begin{aligned} \sum_{i,d} i a(i,d) e^{-\frac{d}{2N_0}} &= \sum_{i=1}^{\infty} i 3^{i-1} e^{-\frac{i+3}{2N_0}} \\ &= \sum_{i=1}^{\infty} i \big(3e^{-\frac{1}{2N_0}}\big)^{i-1} e^{-\frac{4}{2N_0}} \\ &= \frac{1}{(1-3e^{-\frac{1}{2N_0}})^2} e^{-\frac{4}{2N_0}}, \end{aligned}$$

where the last step is true whenever $3e^{-\frac{1}{2N_0}} < 1$. This value is the bit error probability P_b when $E_s = \frac{1}{2}$.

Problem 4. *(Trellis With Antipodal Signals)*

1. To find the most likely path, we maximize the inner product between the received vector $\mathbf{y}$ and the path labels. We use the Viterbi algorithm to do so. The metrics are 8

on the upper-left branch and -2 on the lower-left branch. On the upper-right branch, the metric is -5, hence, the upper path has length 3. On the lower-right branch, the metric is 9, and hence, the lower path has length 7. Therefore, the lower path is more likely.

2. The metrics on the left are the same as above. The upper-right metric is now -9, and hence the upper path has length -1. The lower-right metric is 5, and hence, the lower path has length 3. Therefore, the lower path is more likely.

3. (a) The condition is $-a - c \geq -b + d$.

   (b) The condition is $-a - d \geq -b + c$.

   (c) The condition is $b - c \geq a + d$.

   (d) The condition is $b - d \geq a + c$.

   (e) Yes, the conditions are all equivalent. One can easily see this by bringing all the terms to one side of the inequality. In all four cases, we get $a + c + d - b \leq 0$.

   (f) The advantage is that to compute the most likely state σ_j, we only need to know the metrics of the trellis that are immediately to the left and to the right of depth j. To compute these metrics, we only need to know $y_{2j-1}, y_{2j}, y_{2j+1}, y_{2(j+1)}$. This means that even if we have not (yet) received the whole $\mathbf{y}$ sequence, we can find parts of the most likely path, and hence, we can decode parts of the transmitted message. This is a great advantage if we use the communication system for real-time applications (like audio or video), where the decoding-delay is important. Also, the decoding complexity is considerably lower than when we run the full Viterbi algorithm. We do not have to compute all the metrics, and no backtracking is needed in the end of the algorithm.