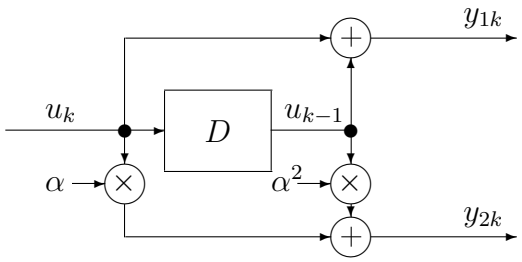
Final Exam Solutions

Problem F.1 (60 points)

In this problem we consider a convolutional code C over the quaternary field $\mathbb{F}_4$. The elements of $\mathbb{F}_4$ may be denoted as $\{00, 01, 10, 11\}$ (additive representation) or as $\{0, 1, \alpha, \alpha^2\}$ (multiplicative representation), where α is a primitive element of $\mathbb{F}_4$ and a root of x^2+x+1. You might wish to jot down the addition and multiplication tables of $\mathbb{F}_4$.

The convolutional code C is generated by the encoder shown below.



The input u_k at time k is an element of $\mathbb{F}_4$, and the delay element (denoted by D) stores the previous input u_{k-1}. There are two $\mathbb{F}_4$ outputs at each time k, whose equations are

$$\begin{aligned} y_{1k} &= u_k + u_{k-1}; \\ y_{2k} &= \alpha u_k + \alpha^2 u_{k-1}. \end{aligned}$$

(a) Show that the convolutional code C is linear over $\mathbb{F}_4$.

If $\{(y_{1k}, y_{2k})\} \in C$ and $\{(y'_{1k}, y'_{2k})\} \in C$ are the output sequences corresponding to the input sequences $\{u_k\}$ and $\{u'_k\}$, respectively, then the input sequence $\{u_k + u'_k\}$ generates the output sequence $\{(y_{1k} + y'_{1k}, y_{2k} + y'_{2k}\}$, since

$$\begin{aligned} y_{1k} + y'_{1k} &= u_k + u'_k + u_{k-1} + u'_{k-1}; \\ y_{2k} + y'_{2k} &= \alpha(u_k + u'_k) + \alpha^2(u_{k-1} + u'_{k-1}). \end{aligned}$$

Thus if $\{(y_{1k}, y_{2k})\}$ and $\{(y'_{1k}, y'_{2k})\}$ are in C, then $\{(y_{1k} + y'_{1k}, y_{2k} + y'_{2k}\}$ is in C. Also, for any $\beta \in \mathbb{F}_4$, the input sequence $\{\beta u_k\}$ generates the output sequence $\{\beta(y_{1k}, y_{2k})\}$, since

$$\begin{aligned} \beta y_{1k} &= \beta u_k + \beta u_{k-1}; \\ \beta y_{2k} &= \alpha(\beta u_k) + \alpha^2(\beta u_{k-1}). \end{aligned}$$

Thus if $\{(y_{1k}, y_{2k})\} \in C$, then $\{\beta(y_{1k}, y_{2k})\} \in C$. So C is a vector space over $\mathbb{F}_4$.

Alternatively, after doing part (b), we can verify that if $\mathbf{y}(D) = u(D)\mathbf{g}(D)$ and $\mathbf{y}'(D) = u'(D)\mathbf{g}(D)$ are in C, then $\mathbf{y}(D) + \mathbf{y}'(D) = (u(D) + u'(D))\mathbf{g}(D)$ is in C, and so is $\beta\mathbf{y}(D) = \beta u(D)\mathbf{g}(D)$.

(b) Let $u(D), y_1(D)$ and $y_2(D)$ be the D-transforms of the sequences $\{u_k\}, \{y_{1k}\}$ and $\{y_{2k}\}$, respectively. Give expressions for $y_1(D)$ and $y_2(D)$ in terms of $u(D)$.

It is straightforward to verify that $y_1(D) = (1+D)u(D)$ and $y_2(D) = (\alpha+\alpha^2 D)u(D)$, since $\{y_{1k}\}$ and $\{y_{2k}\}$ are the convolution of $\{u_k\}$ with the finite sequences $(g_{10} = 1, g_{11} = 1)$ and $(g_{10} = \alpha, g_{11} = \alpha^2)$, respectively.

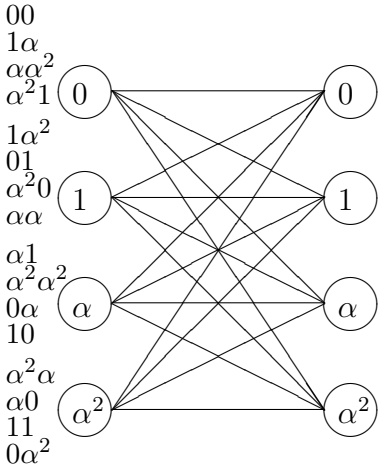
In other words, $\mathbf{y}(D) = u(D)\mathbf{g}(D)$, where

$$\mathbf{g}(D) = (1 + D, \alpha + \alpha^2 D).$$

(c) Specify the number of states in this encoder. Draw a single section of a trellis diagram for C, labelling each branch with a quaternary 2-tuple $(y_{1k}, y_{2k}) \in (\mathbb{F}_4)^2$.

The encoder has 4 states, corresponding to the 4 possible values of u_{k-1}.

Every state transition is possible, so there are 16 branches. The equations of the encoder determine the output 2-tuple associated with each branch. A trellis section for this encoder is therefore as follows:



(d) Show that this encoder for C is noncatastrophic.

The simplest way to show that no infinite input sequence can lead to a finite output sequence is to observe that there is only one branch labeled with 00, namely the branch from the zero state to the zero state, so any nonzero input u_k must give a nonzero output.

Noncatastrophicity also follows algebraically from the fact that $1 + D$ and $\alpha + \alpha^2 D$ have no common factors.

(e) Find the minimum Hamming distance $d_{\text{free}}(C)$, and the average number of nearest neighbors $K_{\min}(C)$ per unit time.

Since C is linear, it suffices to find the minimum-weight nonzero code sequences. Since it is noncatastrophic, the finite-weight code sequences are those generated by finite input sequences.

The impulse response $\mathbf{g}(D) = (1+D, \alpha+\alpha^2 D)$ is a code sequence with Hamming weight 4, and so are $\alpha\mathbf{g}(D)$ and $\alpha^2\mathbf{g}(D)$. All finite code sequences start with a weight-2 2-tuple and end with a weight-2 2-tuple, and have 2-tuples of weight at least 1 in between. Therefore the nonzero scalar multiples of $\mathbf{g}(D)$ are the only nonzero minimum-weight sequences, the free distance is $d_{\text{free}}(C) = 4$, and $K_{\min}(C) = 3$.

Now define the binary image *of C as the binary convolutional code C' obtained by mapping the outputs $y_{jk} \in \mathbb{F}_4$ into the additive representation $\{00, 01, 10, 11\}$, where each representative is a pair of elements of $\mathbb{F}_2$.*

(f) Repeat parts (a)-(e) for C', replacing $\mathbb{F}_4$ by $\mathbb{F}_2$ where appropriate. (For part (b), map $u_k \in \mathbb{F}_4$ to its binary image.)

It is easy to verify that the binary image map $f : \mathbb{F}_4 \to (\mathbb{F}_2)^2$ is linear; *i.e.*, $f(\beta + \gamma) = f(\beta) + f(\gamma)$. Therefore if $f(\mathbf{y}(D))$ and $f(\mathbf{y}'(D))$ are code sequences in C', then so is $f(\mathbf{y}(D)) + f(\mathbf{y}'(D)) = f(\mathbf{y}(D) + \mathbf{y}'(D))$, since $\mathbf{y}(D) + \mathbf{y}'(D) \in C$. So C' is a vector space over $\mathbb{F}_2$.

The trellis for C' is the same as that for C with quaternary labels mapped to binary labels. In other words, C' is a rate-2/4, 4-state binary linear convolutional code. The encoder is still noncatastrophic because there are no branches other than the zero branch that have the all-zero label 0000.

If the input sequence is zero except for 10 (α) at time zero, then the output sequence is $(\alpha\alpha^2, \alpha 1) = (1011, 1001)$; *i.e.*, the impulse response is $(1 + D, 0, 1, 1 + D)$. Similarly the impulse response to an input 01 at time zero is $(0110, 0111) = (0, 1 + D, 1 + D, D)$. Thus the generator matrix of this binary rate-2/4 code is

$$G'(D) = \begin{bmatrix} 1+D & 0 & 1 & 1+D \\ 0 & 1+D & 1+D & D \end{bmatrix}$$

Each of the two generator sequences has Hamming weight 5, and their sum has weight 6. By noting that every finite sequence starts and ends with a 4-tuple of weight at least 2 and examining the low-weight continuations through the trellis, we can quickly verify that these are the only weight-5 sequences in the code. Therefore $d_{\text{free}}(C') = 5$ and $K_{\min}(C) = 2$.

(g) Compute the nominal spectral efficiency $\rho(C')$ and the nominal coding gain $\gamma_c(C')$, and estimate the effective coding gain $\gamma_{\text{eff}}(C')$ using our usual rule of thumb. Compare the performance of C' to that of the best rate-$1/n$ binary linear convolutional code with the same spectral efficiency and number of states (see tables above).

The nominal spectral efficiency is $\rho(C') = 2k/n = 1$ b/2D. The nominal coding gain is $\gamma_c(C') = d_{\text{free}}k/n = 5/2$ (3.98 dB). The number of nearest neighbors per information bit is $K_b(C') = K_{\min}(C')/k = 1$. Therefore the effective coding gain is the same as the nominal coding gain.

In fact, these parameters are precisely the same as those of the best rate-1/2 4-state binary linear convolutional code, namely our standard example code. Therefore the two codes will have the same performance, to the accuracy of the union bound estimate.

Now define another binary convolutional code C'' as the code obtained by mapping the outputs $y_{jk} \in \mathbb{F}_4$ into the codewords $\{000, 011, 101, 110\}$ in the $(3, 2, 2)$ binary SPC code, where each representative is now a 3-tuple of elements of $\mathbb{F}_2$.

(h) Repeat parts (a)-(e) for C'', replacing $\mathbb{F}_4$ by $\mathbb{F}_2$ where appropriate. (For part (b), map $u_k \in \mathbb{F}_4$ to its binary image.)

Again, it is easy to verify that this map $g : \mathbb{F}_4 \rightarrow (\mathbb{F}_2)^3$ is linear, and thereby to prove that C'' is linear.

The trellis for C'' is the same as that for C with quaternary 2-tuples mapped to binary 6-tuples. In other words, C'' is a rate-2/6, 4-state binary linear convolutional code. The encoder is still noncatastrophic because there are no branches other than the zero branch that have the all-zero label 000000.

The impulse response to an input 10 at time zero is now $(101110, 101011)$. Similarly, the impulse response to an input 01 at time zero is now $(011101, 011110)$. Thus the generator matrix of this binary rate-2/6 code is

$$G''(D) = \begin{bmatrix} 1+D & 0 & 1+D & 1 & 1+D & D \\ 0 & 1+D & 1+D & 1+D & D & 1 \end{bmatrix}$$

Note that the map $g : \mathbb{F}_4 \rightarrow (\mathbb{F}_2)^3$ maps every nonzero element of $\mathbb{F}_4$ into a binary 3-tuple of Hamming weight 2. Therefore the weight of any binary image sequence is just twice the weight of the corresponding quaternary sequence. Therefore, using our previous results for C in part (e), we have $d_{\text{free}}(C'') = 8$ and $K_{\min}(C'') = 3$.

(i) Compute $\rho(C'')$ and $\gamma_{\text{c}}(C'')$, and estimate $\gamma_{\text{eff}}(C'')$. Compare the performance of C'' to that of the best rate-$1/n$ binary linear convolutional code with the same spectral efficiency and number of states (see tables above).

The nominal spectral efficiency is $\rho(C'') = 2k/n = 2/3$ b/2D. The nominal coding gain is $\gamma_{\text{c}}(C'') = d_{\text{free}}k/n = 8/3$ (4.26 dB). The number of nearest neighbors per information bit is $K_b(C'') = K_{\min}(C'')/k = 3/2$. Therefore the effective coding gain is about 0.1 dB less than the nominal coding gain; *i.e.*, $\gamma_{\text{eff}}(C'') \approx 4.15$ dB.

By comparing with Table 2, we see that the nominal coding gain is precisely the same as that of the best rate-1/3 4-state binary linear convolutional code. Moreover, K_b is actually a factor of 2 better, so the effective coding gain of C'' is about 0.2 dB better.

Problem F.2 (40 points)

In this problem we consider graphical realizations and decoding of the $(32, 16, 8)$ *binary Reed-Muller code* $\mathrm{RM}(2, 5)$.

(a) Show that there is a partition of the 32 symbols of this code into four 8-tuples such that the projection of $\mathrm{RM}(2, 5)$ *onto any 8-tuple is the* $(8, 7, 2)$ *binary SPC code, and the subcode corresponding to each 8-tuple is the* $(8, 1, 8)$ *binary repetition code; moreover, the 8-tuples may be paired such that the projection onto each resulting 16-tuple is the* $(16, 11, 4)$ *extended Hamming code, and the subcode corresponding to each resulting 16-tuple is the* $(16, 5, 8)$ *biorthogonal code.*

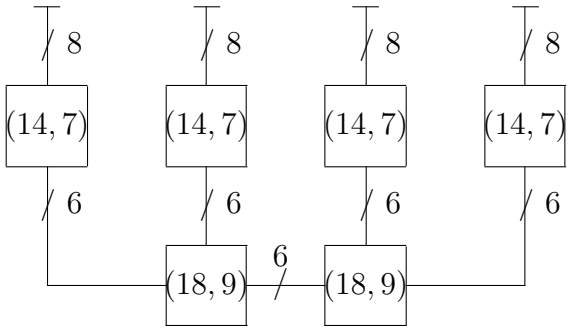
The $(32, 16, 8)$ code $\mathrm{RM}(2, 5)$ may be constructed by the $|u|u + v|$ construction from the $(16, 11, 4)$ code $\mathrm{RM}(2, 4)$ and the $(16, 5, 8)$ code $\mathrm{RM}(1, 4)$ as follows:

$$\mathrm{RM}(2, 5) = \{(\mathbf{u}, \mathbf{u} + \mathbf{v}) \mid \mathbf{u} \in \mathrm{RM}(2, 4), \mathbf{v} \in \mathrm{RM}(1, 4)\}.$$

Thus the projection onto either 16-tuple is $\mathrm{RM}(2, 4)$ (since $\mathrm{RM}(1, 4)$ is a subcode of $\mathrm{RM}(2, 4)$). A codeword has the form $(\mathbf{u}, \mathbf{0})$ if and only if $\mathbf{u} = \mathbf{v} \in \mathrm{RM}(1, 4)$, so the subcode of codewords equal to zero on the second 16-tuple is equal to $\mathrm{RM}(1, 4)$ on the first 16-tuple. Similarly, a codeword has the form $(\mathbf{0}, \mathbf{u} + \mathbf{v})$ if and only if $\mathbf{u} = \mathbf{0}$, which implies that the second 16-tuple is a codeword $\mathbf{v} \in \mathrm{RM}(1, 4)$.

Similarly, the $(16, 11, 4)$ code $\mathrm{RM}(2, 4)$ may be constructed by the $|u|u + v|$ construction from the $(8, 7, 2)$ code $\mathrm{RM}(2, 3)$ and the $(8, 4, 4)$ code $\mathrm{RM}(1, 3)$; by an argument similar to that above, this shows that the projection onto any corresponding 8-tuple is $\mathrm{RM}(2, 3)$. Also, the $(16, 5, 8)$ code $\mathrm{RM}(1, 4)$ may be constructed by the $|u|u + v|$ construction from the $(8, 4, 4)$ code $\mathrm{RM}(1, 3)$ and the $(8, 1, 8)$ code $\mathrm{RM}(0, 3)$; by an argument similar to that above, this shows that the subcode corresponding to any 8-tuple is $\mathrm{RM}(0, 3)$.

(b) Using part (a), show that there is a normal realization of $\mathrm{RM}(2, 5)$ *whose graph is as follows:*



[Tip: to find the constraint code dimensions, you may use the fact (not proved in 6.451) that the constraint codes in a cycle-free realization of a self-dual code are self-dual.]

If the 32 symbols are partitioned into 8-tuples as in part (a), then the projection and subcode corresponding to each such 8-tuple are the $(8, 7, 2)$ and $(8, 1, 8)$ code, respectively. By the state space theorem, the dimension of the state space corresponding to a partition of the time axis into one of these 8-tuples and the remaining 24-tuple is equal to the

difference of the dimensions between this projection and subcode, which is 6. Similarly, the dimension of the state space corresponding to the partition of the time axis into two 16-tuples is the difference of the dimensions of the projection $(16, 11, 4)$ and the subcode $(16, 5, 8)$, which is also 6. This accounts for the dimensions of all state spaces shown in the normal graph above.

The lengths of the constraint codes are simply the sums of the dimensions of the incident variables, which are $8 + 6 = 14$ for the top constraint codes, and $6 + 6 + 6 = 18$ for the bottom constraint codes. Using the tip and the fact that the $(32, 16, 8)$ RM code is a self-dual code, the constraint codes are self-dual and therefore must have dimension equal to half their length.

(c) Using part (b), give a high-level description of an efficient algorithm for maximum-likelihood decoding of $\mathrm{RM}(2, 5)$ *on an arbitrary memoryless channel.*

Maximum-likelihood decoding of a code defined on a cycle-free graph over any memoryless channel may be performed by the max-product algorithm using likelihood weights, or equivalently by the min-sum algorithm using negative log likelihood weights.

A high-level description of how the max-product algorithm would work on the above graph is as follows. The inputs are the received likelihood vectors (2-tuples) for each of the 32 received symbols. For the top constraint codes, the max-product update rule amounts to taking the maximum of each pair of 8-tuple "intrinsic" likelihoods (which are each the product of the appropriate 8 received likelihoods) corresponding to each of the 64 states (cosets of the $(8, 1, 8)$ code in the $(8, 7, 2)$ code). Next, for the bottom constraint codes, the max-product update rule amounts to taking the products of two incoming state likelihoods corresponding to 12 of the 18 bits in each of the 512 codewords of the $(18, 9)$ constraint code, and then taking the maximum of each of the 8 such products that correspond to each of the 64 states specified by the third 6-tuple in each codeword. The result is a vector of 64 likelihoods, one for each state in the central state space.

In a generalized Viterbi algorithm, the "past" and "future" likelihoods of each central state could then be combined (multiplied), and the maximum selected. The result will be the maximum likelihood of any codeword. If the sequence of decisions leading to this maximum have been remembered, or are retraced, then the entire maximum-likelihood codeword can be reconstructed.

However, you could continue to straightforwardly apply the max-product algorithm. The max-product update rule can now be applied twice more to each bottom constraint code, resulting in "extrinsic" likelihood vectors for all 4 upper-level state spaces. At each of the top-level constraint codes, the max-product update rule amounts to fanning out each of these 64 "extrinsic" likelihoods, 2 at a time, to the 128 possible input 8-tuples. These likelihoods may then be combined (multiplied) with each of the corresponding "intrinsic" likelihoods, and the maximum of these 128 products chosen; this maximum corresponds to the ML-decoded 8-tuple. The four ML-decoded 8-tuples then form the ML-decoded codeword.

(d) Compare the performance (probability of error) and complexity (number of arithmetic operations, roughly) of the algorithm of part (c) to that of the Viterbi algorithm applied to an efficient trellis realization of $\mathrm{RM}(2, 5)$*. [Hint: start by finding a trellis-oriented*

generator matrix for RM$(2,5)$*, and then find an efficient sectionalization.]*

The max-product decoder and the Viterbi algorithm applied to a trellis are both exact ML decoding algorithms, so both will have the same performance (probability of error).

To compare complexity, we first need to find an efficient trellis realization for RM$(2,5)$. The standard coordinate ordering for RM codes yields an efficient trellis realization. A standard generator matrix corresponding to the standard coordinate ordering is obtained by taking the 16 rows of weight 8 or more from the "universal" 32×32 generator matrix $U_{32} = U_2^{\otimes 5}$, namely

$$G = \left[\begin{array}{cccc}
11111111 & 00000000 & 00000000 & 00000000 \\
11110000 & 11110000 & 00000000 & 00000000 \\
11001100 & 11001100 & 00000000 & 00000000 \\
10101010 & 10101010 & 00000000 & 00000000 \\
11111111 & 11111111 & 00000000 & 00000000 \\
11110000 & 00000000 & 11110000 & 00000000 \\
11001100 & 00000000 & 11001100 & 00000000 \\
10101010 & 00000000 & 10101010 & 00000000 \\
11111111 & 00000000 & 11111111 & 00000000 \\
11000000 & 11000000 & 11000000 & 11000000 \\
10100000 & 10100000 & 10100000 & 10100000 \\
11110000 & 11110000 & 11110000 & 11110000 \\
10001000 & 10001000 & 10001000 & 10001000 \\
11001100 & 11001100 & 11001100 & 11001100 \\
10101010 & 10101010 & 10101010 & 10101010 \\
11111111 & 11111111 & 11111111 & 11111111
\end{array}\right].$$

These generators already have distinct stopping times. Reducing G to trellis-oriented form, we obtain symmetrical starting times:

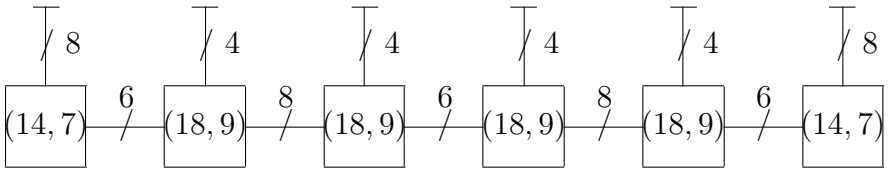
$$G' = \left[\begin{array}{cccc}
11111111 & 00000000 & 00000000 & 00000000 \\
00001111 & 11110000 & 00000000 & 00000000 \\
00110011 & 11001100 & 00000000 & 00000000 \\
01010101 & 10101010 & 00000000 & 00000000 \\
00000000 & 11111111 & 00000000 & 00000000 \\
00000000 & 00001111 & 11110000 & 00000000 \\
00000000 & 00110011 & 11001100 & 00000000 \\
00000000 & 01010101 & 10101010 & 00000000 \\
00000000 & 00000000 & 11111111 & 00000000 \\
00000011 & 00000011 & 11000000 & 11000000 \\
00000101 & 00000101 & 10100000 & 10100000 \\
00000000 & 00000000 & 00001111 & 11110000 \\
00010001 & 00010001 & 10001000 & 10001000 \\
00000000 & 00000000 & 00110011 & 11001100 \\
00000000 & 00000000 & 01010101 & 10101010 \\
00000000 & 00000000 & 00000000 & 11111111
\end{array}\right].$$

From this trellis-oriented generator matrix, we can read off the dimensions s_k and b_k of the state and branch spaces of an unsectionalized trellis, namely (first half only; the second half is symmetric):

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
s_k	0	1	2	3	4	5	6	7	6	7	8	9	8	9	8	7	6
b_k	1	2	3	4	5	6	7	7	7	8	9	9	9	9	8	7
	$<$	$<$	$<$	$<$	$<$	$<$	$<$	$>$	$<$	$<$	$<$	$>$	$<$	$>$	$>$	$>$

The last row of the table above shows starting times ($<$) and stopping times ($>$). From these times and either our heuristic clustering rule or the LV optimal clustering rule, we obtain section boundaries at $k = \{0, 8, 12, 16, 20, 24, 32\}$, state complexity profile $\{0, 64, 256, 64, 256, 64, 0\}$ and branch complexity profile $\{128, 512, 512, 512, 512, 128\}$ (very similar to the optimal sectionalization and profiles of the Golay code).

A normal graph of the resulting sectionalized trellis is as follows:



Comparing this graph to the earlier graph, we see that they differ only in the second and third sections (8-tuples), where the trellis realization has two $(18, 9)$ sections (constraint codes) in series, whereas the earlier realization has one $(14, 7)$ constraint code and one $(18, 9)$ constraint code.

Since the complexity of the max-product update rule or Viterbi algorithm update is roughly proportional to the constraint code (branch space) size, the total complexity of the Viterbi algorithm is roughly proportional to $2 \cdot 128 + 4 \cdot 512 = 2304$, whereas the complexity of the max-product algorithm with survivor memory (generalized Viterbi algorithm) on the earlier graph is roughly proportional to $4 \cdot 128 + 2 \cdot 512 = 1536$. (This assumes either that we use survivor memory in both algorithms to avoid decoding the same sections twice, or equivalently that we do not use survivor memory in either algorithm.) Thus decoding the earlier graph is roughly 2/3 as complex as decoding the optimum sectionalized trellis. Again, we see that optimal cycle-free graphs can be more efficient than optimal sectionalized trellises, but not by very much.

Problem F.3 (50 points)

For each of the propositions below, state whether the proposition is true or false, and give a brief proof. If a proposition is false, the proof will usually be a counterexample. Full credit will not be given for correct answers without an adequate explanation.

(a) The Euclidean image of an (n, k, d) binary linear block code is an orthogonal signal set if and only if $k = \log_2 n$ and $d = n/2$.

FALSE. The Euclidean images of two binary words are orthogonal if and only if $d = n/2$, so all codewords must be at distance $d = n/2$ from each other. In a linear code, this happens if and only if all nonzero codewords have weight $d = n/2$. However, all that is specified is that the minimum distance is $d = n/2$, which is necessary but not sufficient. The smallest counterexample is a $(4, 2, 2)$ code; *e.g.*, $\{0000, 1100, 0011, 1111\}$.

(b) Every element $\beta \in \mathbb{F}_{32}$ is the root of a binary polynomial $f(x) \in \mathbb{F}_2[x]$ of degree less than or equal to 5.

TRUE. Every element $\beta \in \mathbb{F}_{32}$ is a root of $x^{32} + x$, which factors into the product of all binary irreducible polynomials whose degrees divide 5; *i.e.*, of degree 1 or 5. So β must be a root of a binary irreducible polynomial of degree 1 or 5.

(In fact, 0 and 1 are the roots of the two binary irreducible polynomials of degree 1, namely x and $x + 1$, so the remaining 30 elements of $\mathbb{F}_{32}$ must be roots of irreducible polynomials of degree 5, and there must be 6 such polynomials.)

(c) If codewords in an (n, k, d) binary linear block code with d even are transmitted equiprobably over an AWGN channel using a standard 2-PAM map and are optimally detected, then the minimum squared distance to any decision boundary is twice the minimum squared distance that is achieved if binary hard decisions are made first on each symbol and then the resulting binary received word is optimally decoded.

TRUE. With a standard 2-PAM map $\{0, 1\} \rightarrow \{\pm\alpha\}$, the minimum squared distance between codewords is $d^2_{\min} = 4\alpha^2 d$, so with optimum (minimum-distance) detection the minimum squared distance to any decision boundary is $(d_{\min}/2)^2 = \alpha^2 d$. On the other hand, if hard decisions are made first and there are $d/2$ noise components of magnitude α in coordinates where the transmitted codeword differs from another codeword at Hamming distance d (and 0 in all other coordinates), then a decoding error may be made, so such a vector reaches the decision boundary. The squared norm of such a noise vector is $(d/2)\alpha^2$.

(d) Capacity-approaching codes must have trellis complexity parameters that become arbitrarily large as the Shannon limit is approached arbitrarily closely.

MOST LIKELY TRUE. The average dimension bound shows that the sizes of the maximal branch and state spaces are essentially lowerbounded by 2^{γ_c}, which goes to infinity exponentially with n for any "good" sequence of codes. However, the effective coding gain needed to get to the Shannon limit is finite, so the question of whether the nominal coding gain γ_c has to become arbitrarily large to get arbitrarily close to the Shannon limit remains open. However, all empirical evidence indicates that it does. (Credit based on the quality of your discussion.)

(e) If the points $\mathbf{x}$ *in a lattice* Λ *are transmitted with unequal probabilities* $\{p(\mathbf{x}), \mathbf{x} \in \Lambda\}$ *over an AWGN channel and optimally detected, then* $\Pr(E) \approx K_{\min}(\Lambda) Q^{\surd}(d^2_{\min}(\Lambda)/4\sigma^2)$*, where* $d^2_{\min}(\Lambda)$ *is the minimum squared distance between points in* Λ*, and* $K_{\min}(\Lambda)$ *is the average number of nearest neighbors to each transmitted point.*

FALSE. The non-equiprobable condition does not exclude using only a subset $\mathcal{A}$ of the lattice points with minimum distance $d^2_{\min}(\mathcal{A}) > d^2_{\min}(\Lambda)$; *e.g.*, a sublattice $\Lambda' \subset \Lambda$. In this case the argument of the $Q^{\surd}(\cdot)$ function in the UBE will be at least $d^2_{\min}(\mathcal{A})/4\sigma^2$.

Also, with non-equiprobable signals, minimum-distance (Voronoi) decision regions are in general not optimum, so the argument that justifies the UBE no longer holds.