Name: SOLUTIONS

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.02 Fall 2010

Quiz II

November 2, 2010

"×" your section	Section	Time	Room	Recitation Instructor
☐	1	10-11	36-112	Tania Khanna
☐	2	11-12	36-112	Tania Khanna
☐	3	12-1	36-112	George Verghese
☐	4	1-2	36-112	George Verghese
☐	5	2-3	26-168	Alexandre Megretski
☐	6	3-4	26-168	Alexandre Megretski

There are **17 questions** (some with multiple parts) and **12 pages** in this quiz booklet. Answer each question according to the instructions given. You have **120 minutes** to answer the questions.

If you find a question ambiguous, please be sure to write down any assumptions you make. **Please be neat and legible.** If we can't understand your answer, we can't give you credit! *And please show your work for partial credit.*

Use the empty sides of this booklet if you need scratch space. You may also use them for answers, although you shouldn't need to. *If you use the blank sides for answers, make sure to say so!*

Please write your name CLEARLY in the space at the top of this page. NOW, please!

One two-sided "crib sheet" allowed. No other notes, books, calculators, computers, cell phones, PDAs, information appliances, carrier pigeons carrying messages, etc.!

Do not write in the boxes below

1-2 (x/18)	3-5 (x/14)	6-9 (x/18)	10-12 (x/15)	13-14 (x/15)	15-17 (x/20)	Total (x/100)

I Linear Block Codes

1. [6 points]: For each of the three codes below, circle whether the code is a linear block code over $\mathbb{F}_2$ or not. Also fill in the rate of the code.

A. $\{111, 100, 001, 010\}$. Linear / Not linear. Not linear. Code rate = 2/3.

B. $\{00000, 01111, 10100, 11011\}$. Linear / Not linear. Linear. Code rate = 2/5.

C. $\{00000\}$. Linear / Not linear. Linear. Code rate = 0.

Solution: Part A is "Not linear" because the 000 codeword is missing—because the sum of any two codewords must be a codeword for a linear code, the absence of 000 makes the code nonlinear by simple inspection. The others are linear because the sum of any two codewords is indeed a codeword.

The code rate of a block code with 2^k codewords is k/n, where n is the number of bits in each codeword. That's because we can identify each codeword uniquely with k bits, if we have 2^k total codewords. Although most students answered this question correctly, some were confused by the definition of the code rate and answered (wrongly) that it was $1/2^k$ (i.e., the reciprocal of the number of code words in the code). Some students also wrongly answered that the code rate for Part (C) was non-zero (e.g., 1/5). The code rate is 0: if a code has only one element, *no useful information* can get sent because the receiver already knows *exactly* what it's getting!

2. [12 points]: Recall that a block code takes a set of k-bit messages and produces n-bit codewords, with a minimum Hamming distance of d between any two codewords. For each (n, k, d) combination below, state whether a linear block code with those parameters exists or not. *Please provide a brief explanation for each case*: **if such a code exists, give an example; if not, you may rely on a suitable necessary condition.**

A. $(31, 26, 3)$: **Yes** / **No** (circle one)
Solution: Yes. A Hamming code with these parameters exists, as can be easily verified.

B. $(32, 27, 3)$: **Yes** / **No** (circle one) Solution: No. $2^{32-27} = 32 < 32 + 1 = 33$, so such a code is impossible.

C. $(43, 42, 2)$: **Yes** / **No** (circle one) Solution: Yes. The parity code constructed over all 42-bit messages has these parameters.

D. $(27, 18, 3)$: **Yes** / **No** (circle one) Solution: Yes. The rectangular code with $r = 6$ rows and $c = 3$ columns (or vice versa). $n = rc + r + c$ and $k = rc$. The code can correct all single bit errors, as seen in Lab 4.

E. $(11, 5, 5)$: **Yes** / **No** (circle one)
Solution: No. $d = 5$ means **all patterns of up to two** bit errors can be corrected, but $2^{11-5} = 64 < \binom{11}{2} + 11 + 1 = 67$.

Solution: Some students were confused about the difference between the necessary condition for a code to exist and the existence of a code just because a necessary condition holds. Specifically: *if* a linear block code has the ability to correct all single bit errors, *then* its parameters, n and k must satisfy the inequality $2^{n-k} \geq n+1$. This result was proved in the lecture notes. Moreover, we showed that a Hamming code exists for specific values of n of the form $2^m - 1$. However, based on what we studied, we cannot conclude that *if* the relation $2^{n-k} \geq n+1$ holds for a code, then that particular code can correct all single bit errors. So, for example, for Part (D), one needed to show an example of a code with the specified parameters; likewise for Part (A). Part (E) was the trickiest of the problems because it asks about a code capable of correcting *up to two* bit errors, but is a direct application of a problem in the lecture notes and the review problem set. For Part (C), $d = 2$, so the inequality above does not apply. One just needs to observe that there always exists an $(n, n-1, 2)$ code—adding a parity bit to each bit string!

II "Pairwise" Codes

Pairwise Communications has developed a linear block code over $\mathbb{F}_2$ with three data and three parity bits:

$$\begin{aligned} P_1 &= D_1 + D_2 \quad \text{(Each } D_i \text{ is a data bit; each } P_i \text{ is a parity bit.)} \\ P_2 &= D_2 + D_3 \\ P_3 &= D_3 + D_1 \end{aligned}$$

3. [4 points]: Fill in the values of the following three attributes of this code:

1. Code rate = $3/6 = 1/2$.

2. Number of 1s in a minimum-weight non-zero codeword = 3. (Explain your answer.)
Note: The *weight* of a codeword is the number of 1s in it.

Solution: Assuming without loss of generality that a codeword has the form $D_1D_2D_3P_1P_2P_3$, the seven non-zero codewords for the code are 001011, 010110, 011101, 100101, 101110, 110011, and 111000. It is easy to see that the minimum weight of these codewords is 3.

3. Minimum Hamming distance of code = 3.

Solution: That's because the desired quantity is always equal to the weight of the minimum-weight non-zero codeword for any linear code.

4. [6 points]: The receiver computes three syndrome bits from the (possibly corrupted) received data and parity bits: $E_1 = D_1 + D_2 + P_1, E_2 = D_2 + D_3 + P_2$, and $E_3 = D_3 + D_1 + P_3$. The receiver performs maximum likelihood decoding using the syndrome bits. For the combinations of syndrome bits in the table below, state what the maximum-likelihood decoder believes has occured: no errors, a single error in a specific bit (state which one), or multiple errors.

$E_3E_2E_1$	**Error pattern** ["No errors" / "Error in bit ..." (specify the bit) / "Multiple errors"]
0 0 0	No errors.
0 1 0	Error in P_2.
1 0 1	Error in D_1.
1 1 1	Multiple errors.

5. [4 points]: Alyssa P. Hacker extends Pairwise's code by adding an *overall parity bit*. That is, she computes $P_4 = \sum_{i=1}^{3}(D_i + P_i)$, and appends P_4 to each original codeword to produce the new set of codewords. What improvement in error correction or detection capabilities, if any, does Alyssa's extended code show over Pairwise's original code? **(Explain your answer in the space below.)**

Solution: Adding a parity bit to each codeword increases the minimum Hamming distance from 3 to 4. That does **not** change the error correction capability in the worst case, because there are patterns of 2-bit errors that cannot be corrected, but it increases the error detection capability from all 2-bit errors to all 3-bit errors. Several students were tripped up on this question, which applies a property we did study (in a different context): adding a parity bit to every codeword in a code whose d is odd makes it a code with Hamming distance $d + 1$, an even number, and makes it possible to detect one more error than before. But the worst-case error correcting capability remains unchanged.

III Convolutional Coding

Consider a convolutional code whose parity equations are

$$\begin{aligned} p_0[n] &= x[n] + x[n-1] + x[n-3] \\ p_1[n] &= x[n] + x[n-1] + x[n-2] \\ p_2[n] &= x[n] + x[n-2] + x[n-3] \end{aligned}$$

6. [3 points]: What is the rate of this code? How many states are in the state machine representation of this code as discussed in 6.02?

Code rate = 1/3.

Number of states in the state machine representation = $2^3 = 8$.

7. [7 points]: Suppose the decoder reaches the state "110" during the forward pass of the Viterbi algorithm with this convolutional code.

A. How many predecessor states (i.e., immediately preceding states) does state "110" have?
Solution: 2.

B. What are the bit-sequence representations of the predecessor states of state "110"?
Solution: "100" and "101". Note: we assumed (as in the rest of 6.02) that the most recent bit is on the left and the least recent on the right. We tried to be careful in giving full credit to students who had the reverse interpretation, *as long as it was consistent throughout the questions.*

C. What are the expected parity bits for the transitions from each of these predecessor states to state "110"? Specify each predecessor state and the expected parity bits associated with the corresponding transition below.

Predecessor state → "110"	Expected parity bits
"100" → "110"	001
"101" → "110"	100

Suppose we want to send a message M using the convolutional code from the previous page over some channel. Let P be the sequence of parity bits produced by the encoder using the convolutional code. The receiver gets a sequence of voltage samples. It digitizes each sample to produce a sequence of received parity bits, R. It decodes this sequence using the Viterbi decoder with the Hamming distance between the expected and received parity bits as the branch metric. Let D denote the decoder's final result (i.e., the message output by the decoder). We find that the minimum path metric among all the states in the final stage of the trellis is 2010.

8. [5 points]: Please answer the following questions.

A. The Hamming distance between __P__ and __R__ is 2010.

Solution: We botched this question, though it's important to note that the above answer is the only reasonable one from the possibilities given. The correct answer is worth understanding in more detail though, so here goes. The answer stated above is correct as long as the decoded message, D, is the same as the original one, M; i.e., as long as the decoder successfully corrected all the errors.

But if not, the path metric corresponding to the final state of the most likely path in the Viterbi decoder is in fact the Hamming distance between P, the set of received parity bits, and `conv_encode`(D), the parity bit sequence that will be produced if we took the decoded final output, D, and passed that to the same convolutional encoder. That's because the path metric of the final state (from where the reverse pass begins) gives the aggregate Hamming distance between P and the parity sequence corresponding to the most likely path in the forward pass. But that sequence of states (and the corresponding parity bits) is precisely the bit sequence that will be produced by running the convolutional encoder on D!

So where did we botch the question? Well, we should have said to assume that the decoded message is identical to the message that needed to be sent.

In any case, most students got this question correct, and we assigned credit to anyone who said P (or equivalent) to one of the blanks and something not-too-egregious in the other.

B. Suppose the final state from which the Viterbi decoder begins its backward pass has the bit representation $b_1 b_2 \dots b_\ell$, where ℓ is the number of bits required to represent all the states.

Then, D *must* **start / end** with the bit sequence __________________________.

(*Circle "start" or "end" and fill in the blank so that this sentence is always true.*)

Solution: "end" is the right choice. The sequence is $b_\ell \dots b_2 b_1$.

9. [3 points]: To increase the rate of the given code, Lem E. Tweakit punctures the p_0 parity stream using the vector (1 0 1 1 0), which means that every second and fifth bit produced on the stream are *not sent*. In addition, she punctures the p_1 parity stream using the vector (1 1 0 1 1). She sends the p_2 parity stream unchanged. What is the rate of the punctured code?

(Explain your answer in the space below.)

Solution: The original rate is 1/3. The three punctured vectors send 12 of the 15 possible bits. So the rate of the punctured code is $\frac{1}{3} \cdot \frac{15}{12} = \frac{5}{12}$. A few students answered $\frac{5}{7}$ presumably because they forgot about the third parity stream (we gave them partial credit).

A few students gave a more imaginative, but alas still incorrect, answer. They suggested that the rate of the code is equal to $\frac{1}{3} \cdot \frac{3}{5} + \frac{1}{3} \cdot \frac{3}{5} + \frac{1}{3} \cdot \frac{5}{5}$ because 3/5 of the first parity stream carrying a third of the bits is sent, and similarly for the second parity stream, and all of the third stream is sent. This argument is incorrect: I think it's analogous to saying (wrongly) that if you drive in a car from A to B at a speed of 60 mph and from B to A at a speed of 40 mph, then the average speed for your trip is 50 mph...

IV Time Division Duplex (TDD) MAC Protocol

Bluetooth is a wireless technology found on many mobile devices, including laptops, mobile phones, GPS navigation devices, headsets, and so on. It uses a MAC protocol called *Time Division Duplex (TDD)*. In TDD, the shared medium network has 1 master and N slaves. You may assume that the network has already been configured with one device as the master and the others as slaves. Each slave has a unique identifier (ID) that serves as its address, an integer between 1 and N. Assume that no devices ever turn off during the operation of the protocol. Unless otherwise mentioned, assume that no packets are lost.

The MAC protocol works as follows. Time is slotted and each packet is one time slot long.

- In every *odd* time slot $(1, 3, 5, \ldots, 2t - 1, \ldots)$, the master sends a packet addressed to *some* slave for which it has packets backlogged, in round-robin order (i.e., cycling through the slaves in numeric order).
- In every *even* time slot $(2, 4, 6, \ldots, 2t, \ldots)$, the slave that received a packet from the master in the immediately preceding time slot gets to send a packet to the master, if it has a packet to send. If it has no packet to send, then that time slot is left unused, and the slot is wasted.

10. [3 points]: Alyssa P. Hacker finds a problem with the TDD protocol described above, and implements the following rule in addition:

> From time to time, in an odd time slot, the master sends a "dummy" packet addressed to a slave even if it has no other data packets to send to the slave (and even if it has packets for other slaves).

Why does Alyssa's rule improve the TDD protocol?

(Explain your answer in the space below.)

Solution: Because it will prevent a slave from being starved; without it, a slave that has packets to send will never send data packets if the master never has packets to send to it.

Henceforth, the term "TDD" will refer to the protocol described above, augmented with Alyssa's rule. Moreover, whenever a "dummy" packet is sent, that time slot will be considered a wasted slot.

11. [3 points]: Alyssa's goal is to emulate a round-robin TDMA scheme amongst the N slaves. Propose a way to achieve this goal by specifying the ID of the slave that the master should send a data or dummy packet to, in time slot $2t - 1$ (note that $1 \leq t < \infty$).

(Explain your answer in the space below.)

Solution: Send to the slave whose ID is $t \bmod N + 1$. This is almost exactly the same problem as in Lab 5 Task 1 (TDMA). The only difference is that the nodes begin with ID 1 here, not 0. We gave full credit to answers where the $+1$ was missing.

Henceforth, assume that the TDD scheme implements round-robin TDMA amongst the slaves.

12. [4+5=9 points]: Suppose the master always has data packets to send *only* to an arbitrary (but fixed) subset of the N slaves. In addition, a (possibly different) subset of the slaves always has packets to send to the master. Each subset is of size r, a fixed value. Answer the questions below (you may find it helpful to think about different subsets of slaves).

Solution: This question ended up being more confusing than it should have been. And it became tricky unintentionally because the two parts have the same answer, but the helpful hint about different subsets of slaves is a red herring that's really not relevant for TDMA (as several of you pointed out). On the + side, a vast number of students answered the question perfectly.

A. What is the *maximum possible utilization* of such a configuration?

(Explain your answer in the space below.)

Solution: The protocol is TDMA, so the master sends useful data packets in r of the $2N$ slots in an epoch, and receives useful packets in the r of the $2N$ slots. So the utilization is $\frac{r}{N}$. If one seeks to maximize that over r, it's clear that the maximum happens when $r = N$, giving us a maximum of 1. Our original intent was to ask a question about a non-TDMA version, where different subsets would lead to different utilizations: the maximum occurring when the subsets fully overlapped and the minimum occurring when the subsets minimally overlapped. But at some stage we added TDMA to the question and were remiss in not fixing this question. So the answer to this part and the next one became identical.

B. What is the *minimum possible utilization* (for a given value of r) of such a configuration? Assume that $r > N/2$. Note that if the master does not have a data packet to send to a slave in a round, it sends a "dummy" packet to that slave instead. (A dummy packet does not count toward the utilization of the medium.)

(Explain your answer in the space below.)

Solution: $\frac{r}{N}$. When minimizing over all r, the smallest value becomes $\frac{1}{2} + \frac{1}{N}$ when N is even and $\frac{1}{2} + \frac{1}{2N}$ when N is odd. (But we really didn't ask for that, though many students provided the correct answer, such as "slightly bigger than $\frac{1}{2}$" or "$\frac{1}{2} + \frac{1}{N}$".)

VI Filters

Suppose a causal linear time invariant (LTI) system, denoted H, is described by the difference equation relating input X to output Y,

$$y[n] = x[n] + \alpha x[n-1] + \beta x[n-2] + \gamma x[n-3]. \quad (1)$$

15. [7 points]: Please determine the values of α, β and γ so that the frequency response of system H is $H(e^{j\Omega}) = 1 - 0.5e^{-j2\Omega}\cos\Omega$.

Solution: For the difference equation, the non-zero values of the unit sample response are $h[0] = 1$, $h[1] = \alpha$, $h[2] = \beta$, and $h[3] = \gamma$. The associated frequency response is, in general, $H(e^{j\Omega}) = \sum_{m=0}^{m=\infty} h[m]e^{-j\Omega m}$. In this case, the general formula simplifies to

$$H(e^{j\Omega}) = 1 + \alpha e^{-j\Omega} + \beta e^{-j2\Omega} + \gamma e^{-j3\Omega}.$$

Now let's multiply out the given frequency response in terms of complex exponentials:

$$H(e^{j\Omega}) = 1 - 0.5e^{-j2\Omega}\cos\Omega = 1 - 0.5e^{-j2\Omega}\left(0.5e^{j\Omega} + 0.5e^{-j\Omega}\right) = 1 - 0.25e^{-j\Omega} - 0.25e^{-j3\Omega}.$$

Matching the coefficients in the general form to the multiplied out form of the given frequency response yields

$\alpha =$ $-\frac{1}{4}$; $\beta =$ 0; $\gamma =$ $-\frac{1}{4}$.

16. [4 points]: Suppose that Y, the output of H, is used as the input to a second causal LTI system G, with output W, as shown below.

If $H(e^{j\Omega}) = 1 - 0.5e^{-j2\Omega}\cos\Omega$, what should the frequency response, $G(e^{j\Omega})$, be so that $w[n] = x[n]$ for all n?

Solution: For a series (cascade) of LTI systems, the frequency response is the **product** of the frequency responses of the individual LTI systems. So, the frequency response of this cascade is $G(e^{j\Omega})H(e^{j\Omega})$. For $w[n]$ to be equal to $x[n]$ for all n, the unit sample response of the cascade must be $\delta[n]$. The frequency response of the cascade is related the unit sample response by $G(e^{j\Omega})H(e^{j\Omega}) = \sum_{m=0}^{m=\infty} \delta[m]e^{-j\Omega m} = 1$. Since $G(e^{j\Omega})H(e^{j\Omega}) = 1$, and $H(e^{j\Omega})$ is never zero for $-\pi \le \Omega \le \pi$, then $G(e^{j\Omega}) = \frac{1}{H(e^{j\Omega})}$. Hence,

$G(e^{j\Omega}) =$ $1/(1 - 0.5e^{-j2\Omega}\cos\Omega)$.

The difference equation from the previous page, Eq. (1), is reproduced below for convenience:

$$y[n] = x[n] + \alpha x[n-1] + \beta x[n-2] + \gamma x[n-3].$$

17. [9 points]: Suppose $\alpha = 1$ and $\gamma = 1$ in the above equation for an H with a **different** frequency response than the one you obtained in Problem 15. For this different H, you are told that $y[n] = A\,(-1)^n$ when $x[n] = 1.0 + 0.5\,(-1)^n$ for all n. Please use this information to determine the value of β in Eq. (1) and the value of A in the formula for $y[n]$.

Solution: First, observe that we can rewrite the given $x[n]$ as

$$x[n] = e^{j0n} + 0.5e^{j\pi n}.$$

Then using the meaning of frequency response, and applying superposition, we can write

$$y[n] = 0e^{j0n} + Ae^{j\pi n} = H(e^{j0})e^{j0n} + H(e^{j\pi n})0.5e^{j\pi n}.$$

Matching terms, it then follows that $H(e^{j0}) = 0$ and $A = 0.5H(e^{j\pi n})$. For this difference equation, the frequency response for an arbitrary Ω is given by

$$H(e^{j\Omega}) = 1 + \alpha e^{-j\Omega} + \beta e^{-j2\Omega} + \gamma e^{-j3\Omega},$$

so for the special case of $\Omega = 0$,

$$H(e^{j0}) = 1 + \alpha + \beta + \gamma.$$

Given that $H(e^{j0}) = 0$, and $\alpha = \gamma = 1$, it must be true that $\beta = -3$. Then to determine A, consider that

$$H(e^{j\pi n}) = 1 + \alpha e^{-j\pi} + \beta e^{-j2\pi} + \gamma e^{-j3\pi} = 1 - \alpha + \beta - \gamma = 1 - 1 - 3 - 1 = -4.$$

Finally, $A = 0.5H(e^{j\pi n}) = 0.5(-4) = -2$.

$\beta = \underline{\quad -3 \quad}$; $A = \underline{\quad -2 \quad}$.

FIN