Written Exam
Information Transmission - EITA30/EIT100

Department of Electrical and Information Technology
Lund University

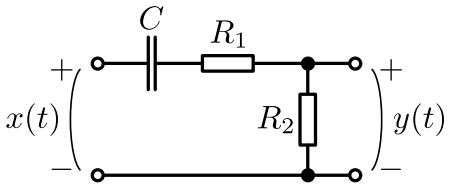
2018-05-28
8.00 – 13.00

*** SOLUTION ***

The exam consists of five problems. 20 of 50 points are required to pass.
Permitted aids: Pocket calculator without any programs, scripts or files stored, formula collection without any notes.

- Write your personal identifier on each page.
- Each solution must be written on separate sheets.
- Your solutions must clearly reveal your method of solution.

1. Given this circuit (a filter attenuating signals at some frequencies more than others),



where $x(t)$ is the input signal and $y(t)$ the output signal, answer the following questions:

(a) What is the frequency function $H(f)$ of the circuit, expressed in capacitance C and resistances R_1 and R_2? (3 p)

Solution: We can calculate the by dividing the voltage across the capacitor and the two resistances, where we get the output across the second resistor. Using the $j\omega$ method, we calculate the output

$$y(t) = \frac{R_2}{R_1 + R_2 + \frac{1}{j\omega C}} x(t)$$

where we identify

$$H(f) = \frac{R_2}{R_1 + R_2 + \frac{1}{j\omega C}} = \frac{j\omega C R_2}{1 + j\omega C(R_1 + R_2)}$$

(b) Assume that the input is a sinus signal with angular frequency ω and unit amplitude, i.e., $x(t) = \sin(\omega t)$. How large is the amplitude of the output $y(t)$ when the frequency is

i. low, i.e., when $\omega \to 0$, and (1 p)

ii. high, i.e., when $\omega \to \infty$? (1 p)

Solution: When $\omega \to 0$

$$\lim_{\omega \to 0} |H(f)| = \frac{0}{1+0} = 0$$

and the output amplitude is 0.
When $\omega \to \infty$ we get (1+ in denominator becomes insignificant)

$$\lim_{\omega \to \infty} |H(f)| = \frac{|j\omega C R_2|}{|j\omega C(R_1 + R_2)|} = \frac{R_2}{(R_1 + R_2)}$$

and the output amplitude is $R_2/(R_1 + R_2)$.

(c) Assume that you want as much as possible of your signal to go through the filter at high frequencies and therefore set $R_1 = 0$ Ohm.

i. At what frequency f (in Hz) is the amplitude of the output $1/\sqrt{2}$ of the input amplitude? (Note: This is called the *cut-off frequency* of the filter.) (3 p)

ii. Given that you have a capacitor with $C = 10$ μF, what value on the resistor R_2 (in Ohms) do you need to set the above cut-off frequency to 100 Hz? (1 p)

Solution: This happens when $|H(f)| = 1/\sqrt{2}$ or, equivalently and more conveniently, when $|H(f)|^2 = 1/2$. Let's use the latter in our calculations. We want to know when

$$|H(f)|^2 = \frac{|j\omega CR_2|^2}{|1 + j\omega C(0 + R_2)|^2} = \frac{(\omega CR_2)^2}{1^2 + (\omega CR_2)^2} = \frac{1}{2},$$

which is satisfied for $\omega CR_2 = 1$ (for positive frequencies) and therefore the frequency we ask for is $\omega = \frac{1}{CR_2}$ radians/sec or $f = \frac{1}{2\pi CR_2}$ Hz.
With a 10 μF capacitor, the resistor needs to be $R_2 = 1/(f \cdot 2\pi \cdot C) = 1/(100 \cdot 2\pi \cdot 10 \cdot 10^{-6}) = 159$ Ohm to set the cut-off frequency to 100 Hz.

(d) Given what you now know about the circuit/filter above, would you call it a low-pass filter (letting through low-frequency signals) or a high-pass filter (letting through high-frequency signals)? Motivate your answer, in a single sentence. (1 p)

Solution: Since the filter is letting through high-frequency signals better than low-frequency ones, it is a high-pass filter.

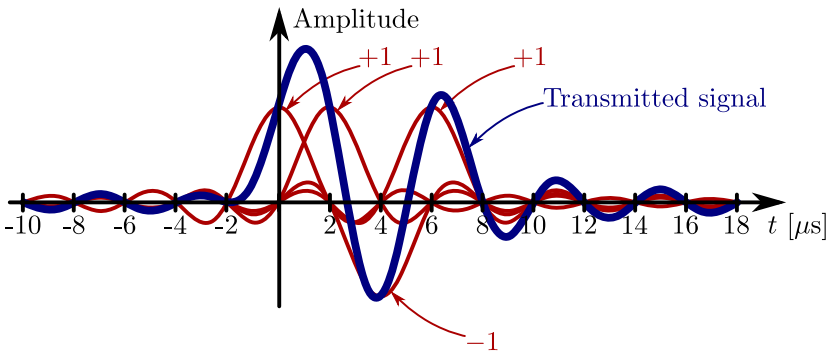
2. Sinc pulses are to be used to transmit a binary sequence of ± 1 at a rate of 500 kbit/s by means of 2-ASK. Transmission takes place in the base-band, i.e., *without* up-conversion to (modulation on) a carrier frequency.

(a) Sketch as accurately as possible the transmitted signal produced by the transmitted sequence +1, +1, -1, +1. (3 p)

Solution: To transmit at 500 kbit/sek, using binary 2-ASK signaling, we need to send one symbol every $T_s = 1/(500 \cdot 10^3) = 2 \cdot 10^{-6} = 2$ μs. This means that our sinc pulses have to be

$$p(t) \propto \text{sinc}\left(\frac{t}{T_s}\right) = \text{sinc}\left(\frac{t}{2 \cdot 10^{-6}}\right),$$

and plotting the sequence of pulses, according to the transmitted sequence, gives (both individual pulses and the transmitted signal shown)



(b) Sketch the spectrum of the transmitted signal as accurately as you can assuming that random independent symbols are transmitted. (3 p)

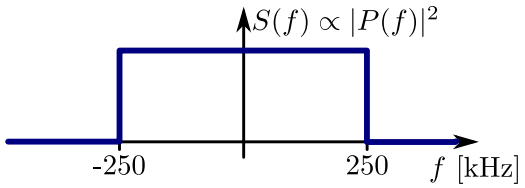
Solution: The Fourier transform of the transmission pulse determines the spectrum of the signal. It is (using Property 5, time scaling, and Fourier transform pair (i))

$$P(f) \propto \text{rect}(fT_s) = \begin{cases} 1 & \text{, if } -250 \cdot 10^3 < f < 250 \cdot 10^3 \\ 0 & \text{, otherwise} \end{cases}.$$

The spectrum of the signal scales proportionally to the magnitude-squared of the pulse Fourier transform, i.e.,

$$S(f) \propto |P(f)|^2$$

The spectrum therefore looks something like this (where the strength of the spectrum depends on the transmit power used).



(c) If 4-ASK (four amplitude levels, e.g. –3, –1, +1, and +3) is used instead of 2-ASK, also with sinc pulses, what is the symbol time given that the bit rate remains unchanged? How is the spectrum affected by this choice of modulation compared to the case of 2-ASK? (4 p)

Solution: Since each transmitted pulse carries 2 bits ($4 = 2^2$ alternatives, gives 2 bits), the symbols can be transmitted *two times slower*, while still maintaining the bit rate. This means that the symbol time with 4-ASK becomes $T_s = 2 \cdot 2 \cdot 10^-6 = 4$ μs. As a consequence, the spectrum becomes half as wide, i.e. instead of a 250 kHz bandwidth, we only need 125 kHz.

3. Peter wants to send messages to Petra using as efficient source coding as possible. Peter sends one word at the time and knows only 6 words: Hello, Love, Banana, Me, Apple, Pen. The probability of transmission of the different words are the following:

$word$	$P(word)$
Hello	0.12
Love	0.08
Banana	0.30
Me	0.33
Apple	0.10
Pen	0.07

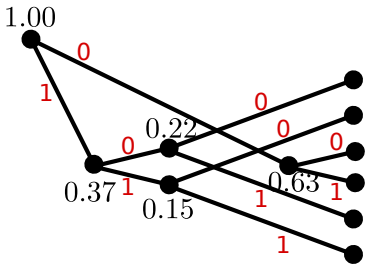
(a) What is the uncertainty of each transmission? (3 p)

Solution: The uncertainty is given by the entropy

$$H(word) = -0.12\log_2 0.12 - 0.08\log_2 0.08 - 0.30\log_2 0.30 - 0.33\log_2 0.33 - 0.10\log_2 0.10 - 0.07\log_2 0.07 \approx 2.31 \text{ bit}$$

(b) Derive an efficient bit representation for the case when many words are sent after another using using as few bits as possible on the average for the transmission. Different number of bits lengths are allowed if this increases the efficiency. (5 p)

Solution: Huffman coding gives the most efficient prefix-free veriable length code:



$P(word)$	*word*	Code word
0.12	Hello	1 0 0
0.08	Love	1 1 0
0.30	Banana	0 0
0.33	Me	0 1
0.10	Apple	1 0 1
0.07	Pen	1 1 1

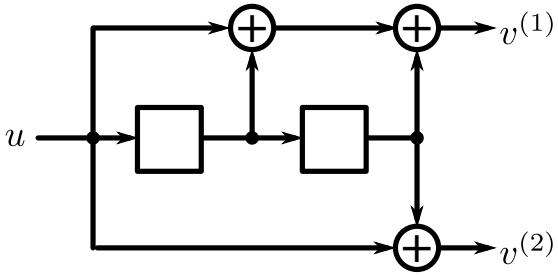
(c) What is the average number of bits per transmission using your bit representation and how far are you from the optimum representation? (2 p)

Solution: The most efficient way to calculate the average code-word lenth (when we have the Huffman construction tree available) is to sum all probabilities of the internal nodes, resulting in

$$W = 1.00 + 0.37 + 0.22 + 0.15 + 0.63 = 2.37 \text{ bit/code word.}$$

This is only 2.5% longer than the optimum representation (which would be 2.31 bit/code word).

4. Consider the rate $R = 1/2$ convolutional encoder shown below.



Hint: After the problems, there are two pages with pre-drawn trellises (only transitions, no outputs) for the type of encoder shown in the figure. Using these may simplify the problem solving a bit.

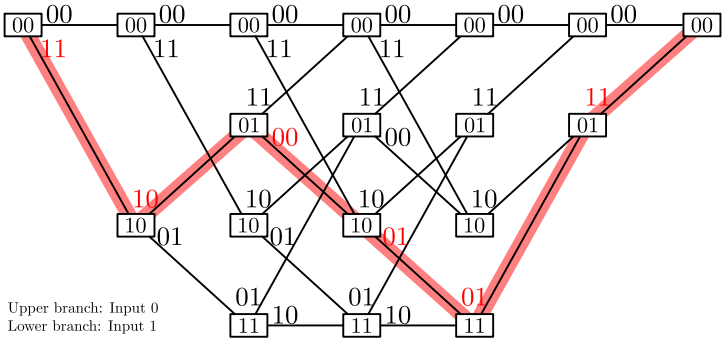
Solution: Before doing anything else, let's figure out the state transitions and encoder outputs, since we are ghoing to need them in more than one sub-problem. Let's list all combinations of inputs and encoder states in a table, and calculate corresponding outputs and next states:

Input u	Current state	Output $v^{(1)}v^{(2)}$	Next state
0	00	00	00
1	00	11	10
0	01	11	00
1	01	00	10
0	10	10	01
1	10	01	11
0	11	01	01
1	11	10	11

(a) Given the input sequence $\boldsymbol{u}$ = 1 0 1 1 0 0, what is the corresponding codeword? (The encoder starts in the all-zero state and the last two zeros

in the sequence are termination bits used to force the encoder back to the all-zero state.) (3 p)

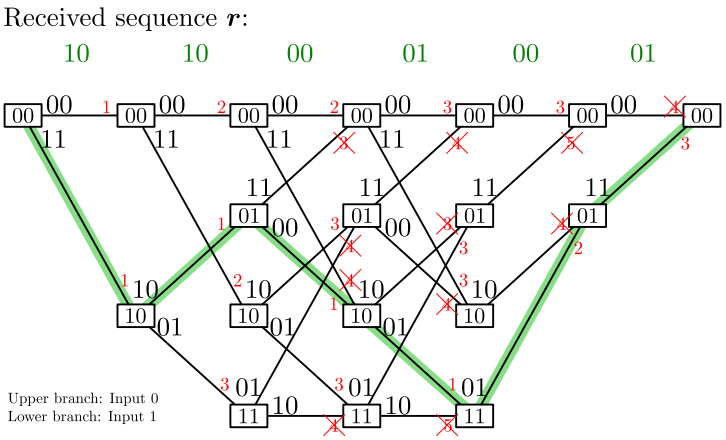
Solution: There are several ways of performing this encoding. It is possible to execute the encoding according to the figure directly, use the table vi calculated with inputs, states and outputs, or we can create a trellis (it will come in handy in the next sub-problem as well). A six-stage trellis (which is the length we need), with the path corresponding to our message $\boldsymbol{u}$ traced out, looks like this:



The codeword corresponding to $\boldsymbol{u}$ is (marked red in the trellis transitions) $\boldsymbol{v} = 11\ 10\ 00\ 01\ 01\ 11$.

(b) Use the Viterbi algorithm to decode the received sequence $\boldsymbol{r} = 10\ 10\ 00\ 01\ 00\ 01$ and show clearly how you get your answer. (5 p)

Solution: The Viterbi algorithm is best executed on the trellis, accumulating metrics along each path (counting bit errors) and eliminating the path with more bit errors when two paths collide. In case of equal metric, toss a fair coin. Executing the Viterbi algorithm for the given received sequence gives:

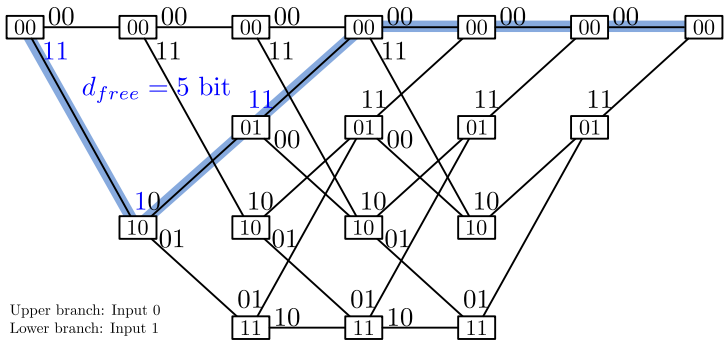


Accumulated metrics are shown in red and the surviving path is marked in green. The decoded code word is $\boldsymbol{v} = 11\ 10\ 00\ 01\ 01\ 11$ and the corresponding information sequence is $\boldsymbol{u} = 1\ 0\ 1\ 1\ 0\ 0$, where the last two 0's are the two termination bits (not really a part of the information message).

(c) How many bit errors can for sure be corrected using this code? Motivate your answer. (2 p)

Solution: To know how many bit errors the code can correct, we need to know its free distance d_{free}. Again, we can use the trellis to find our answer. Finding the codeword, path through the trellist starting in state 00 and

ending up in state 00 again, with the fewest 1's gives us the free distance. A quick search in the trellis gives:



where we identify $d_{free} = 5$ bits.
We can now calculate how many bit-errors the code can for sure correct as

$$\left\lfloor \frac{d_{free} - 1}{2} \right\rfloor = \left\lfloor \frac{5 - 1}{2} \right\rfloor = 2 \text{ bit errors.}$$

If the errors are spread out enough, the code can correct more than two errors. This is beyond the course.

5. Consider an RSA public key crypto system with the public parameters $n = 697$ and $e = 27$. Find the plaintext P corresponding to the the ciphertext $C = 190$. *Hint:* One of the factors of n is $p = 17$. (10 p)

Solution: The first prime factor in n is given as $p = 17$. The other is $q = n/p = n/17 = 41$. Now we can calculate Euler's totient function as $\Phi(n) = (q-1)(p-1) = 640$ and start the process of finding the decoding exponent d. Since e and $\Phi(n)$ are relatively prime ($gcd(e, \Phi(n)) = 1$) and we want d to be the multiplicative inverse of $e \pmod{\Phi(n)}$, i.e. $ed \equiv 1 \pmod{\Phi(n)}$, Bézout's identity says that we can write $de + t\Phi(n) = 1$ and the constant multiplying e must be the d we are looking for. Let's use Euclide's algorithm to calculate $gcd(e, \Phi(n))$ (which we already know is one) and then substitute back again to obtain the form $de + t\Phi(n) = 1$.
Euclide's algorithm for gcd:

$$\begin{aligned} 640 &= 23 \cdot 27 + 19 \\ 27 &= 19 + 8 \\ 19 &= 2 \cdot 8 + 3 \\ 8 &= 2 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 \end{aligned}$$

Going back, starting with the second to last row above, we get

$$
\begin{aligned}
1 &= 3 - 1 \cdot 2 \\
&\quad \text{[Use that } 2 = 8 - 2 \cdot 3\text{]} \\
1 &= 3 - 1 \cdot (8 - 2 \cdot 3) = 3 \cdot 3 - 1 \cdot 8 \\
&\quad \text{[Use that } 3 = 19 - 2 \cdot 8\text{]} \\
1 &= 3 \cdot (19 - 2 \cdot 8) - 1 \cdot 8 = 3 \cdot 19 - 7 \cdot 8 \\
&\quad \text{[Use that } 8 = 27 - 19\text{]} \\
1 &= 3 \cdot 19 - 7 \cdot (27 - 19) = 10 \cdot 19 - 7 \cdot 27 \\
&\quad \text{[Use that } 19 = 640 - 23 \cdot 27\text{]} \\
1 &= 10 \cdot (640 - 23 \cdot 27) - 7 \cdot 27 = 10 \cdot 640 - 237 \cdot 27.
\end{aligned}
$$

Here we identify d as -237, but we want d to fulfill $0 \leq d < 640$. We obtain d as $d = -237 \equiv 403 \pmod{640}$.

Decryption can now be done as

$$P = C^d = 190^{403} \pmod{697}.$$

To awoid numerical overflow, let's first write d as a sum of integer powers of 2, i.e.

$$403 = 256 + 128 + 16 + 2 + 1$$

and we get

$$P = C^d = 190^{256} \cdot 190^{128} \cdot 190^{16} \cdot 190^2 \cdot 190^1 \pmod{697}.$$

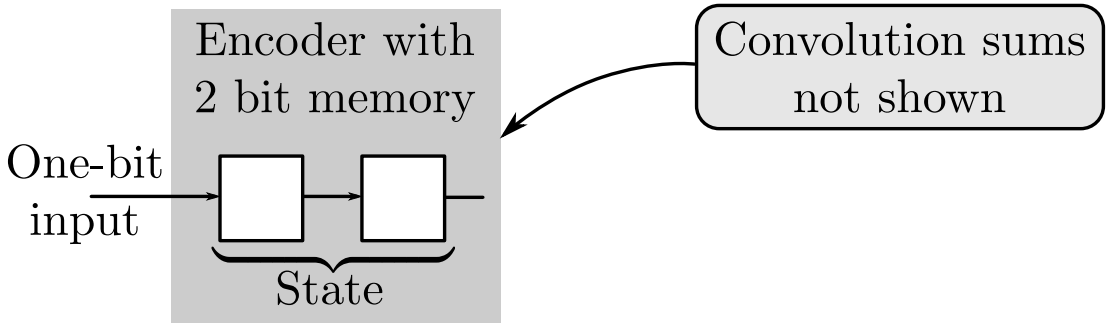
Calculating these exponents of 190 successively, starting with 190^1, we get (the ones we need in bold)

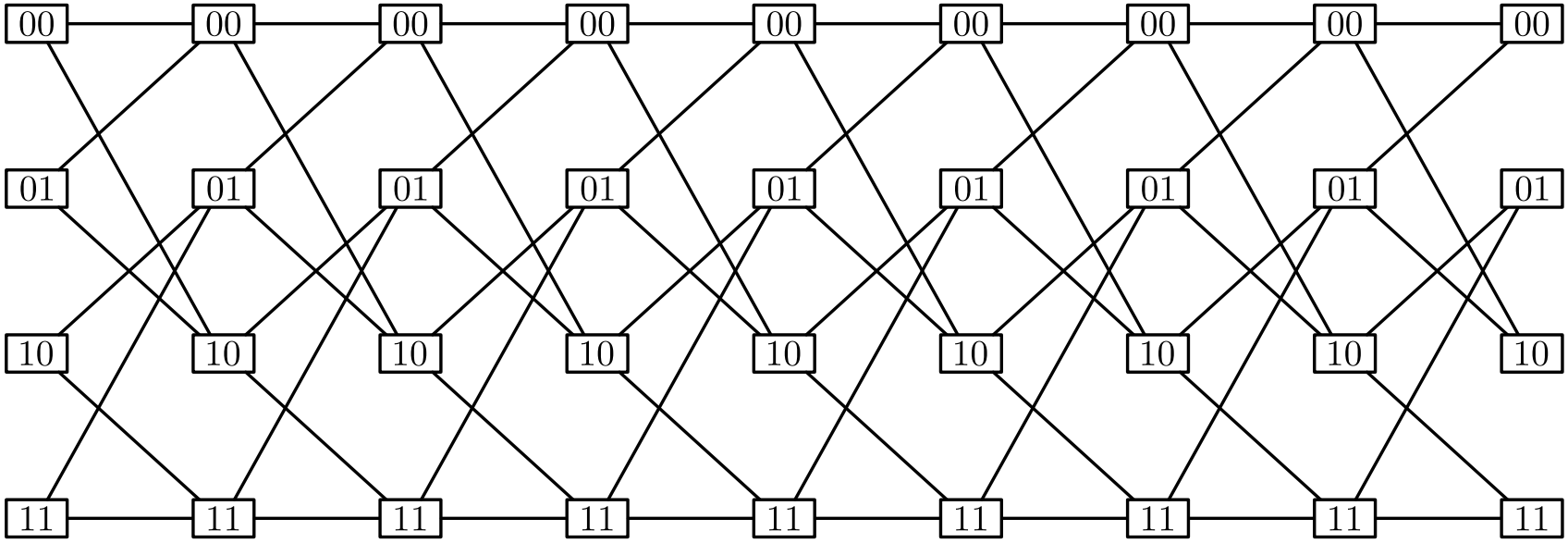
$$
\begin{aligned}
190^1 &\equiv \mathbf{190} \pmod{697} \\
190^2 = 190^2 &\equiv \mathbf{553} \pmod{697} \\
190^4 = 553^2 &\equiv 523 \pmod{697} \\
190^8 = 523^2 &\equiv 305 \pmod{697} \\
190^{16} = 305^2 &\equiv \mathbf{324} \pmod{697} \\
190^{32} = 324^2 &\equiv 426 \pmod{697} \\
190^{64} = 426^2 &\equiv 256 \pmod{697} \\
190^{128} = 256^2 &\equiv \mathbf{18} \pmod{697} \\
190^{256} = 18^2 &\equiv \mathbf{324} \pmod{697}
\end{aligned}
$$

We now recover the plaintext as

$$P = 324 \cdot 18 \cdot 324 \cdot 553 \cdot 190 \equiv 520 \pmod{697}.$$

State-transition trellis for a one-bit input, memory two, feed-forward convolutional encoder.





State-transition trellis for a one-bit input, memory two, feed-forward convolutional encoder.