Written Exam
Information Transmission - EITA30/EIT100

Department of Electrical and Information Technology
Lund University

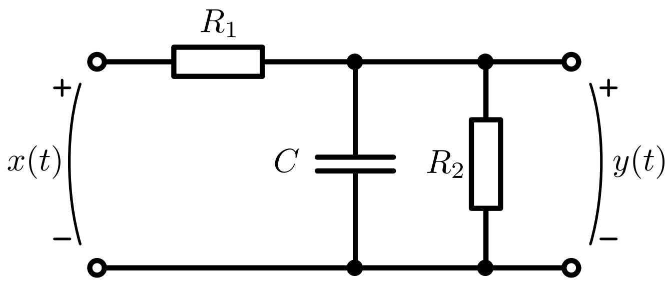
2020-06-01
14.00 – 19.00

***** 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. Consider this circuit, built up by two resistors, R_1 [Ω] and R_2 [Ω], and a capacitor, C [F]. The input signal is $x(t)$ [V] and the output signal $y(t)$ [V].



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 frequency function by dividing the voltage across the resistace R_1 and the capacitance C in parallel with resistance R_2. Using the $j\omega$ method, we calculate the output

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

where we identify the frequency function

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

(b) Assume that the input is a sinus-signal with angular frequency ω and amplitude A, i.e., $x(t) = A\sin(\omega t)$. 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 $\omega \to 0$

$$\lim_{\omega \to 0} |H(f)| = \frac{R_2}{R_1 + R_2}$$

and the output amplitude becomes $A\frac{R_2}{R_1+R_2}$.
When $\omega \to \infty$ we get ($R_1 + R_2$ in denominator becomes insignificant)

$$\lim_{\omega \to \infty} |H(f)| = 0$$

and the output amplitude becomes 0.

(c) For some reason, a critical requirement is that no frequency component of the input signal $x(t)$ is delayed more than 10 ms by the circuit. Let's approach this in two steps.

Hint: *With input $x(t) = \sin(\omega t)$ the output will be on the form $y(t) = |H(f)| \sin(\omega t + \phi(\omega))$, which can be re-written as $y(t) = |H(f)| \sin(\omega(t + \phi(\omega)/\omega))$.*

i. Determine the frequency-dependent time delay $\Delta(\omega)$ (in seconds) imposed by the circuit at angular frequency ω, in terms of R_1, R_2 and C. (2 p)

ii. Use $\Delta(\omega)$ to find an expression (that includes R_1, R_2, and C), that ensures a maximum delay of 10 ms (at any frequency). (1 p)

Solution:

i. The time delay of a sinusoidal signal, in seconds, at angular frequency ω is determined by the phase-angle of the frequency response at that particular frequency, as

$$\Delta(\omega) = -\frac{\angle H(f)}{\omega},$$

where

$$\angle H(f) = \angle\left(\frac{R_2}{R_1 + R_2 + j\omega R_1 R_2 C}\right) = -\arctan\left(\frac{\omega R_1 R_2 C}{R_1 + R_2}\right),$$

which gives

$$\Delta(\omega) = \frac{\arctan\left(\frac{\omega R_1 R_2 C}{R_1+R_2}\right)}{\omega}$$

ii. The delay expression $\Delta(\omega)$ is a decreasing function of ω and its maximum can be found as (we cannot divide by 0)

$$\Delta_{\max} = \lim_{\omega\to 0^+} \Delta(\omega) = \frac{R_1 R_2 C}{R_1 + R_2}.$$

Since the maximum permitted delay is 10 ms, the requirement on R_1, R_2, and C, becomes

$$\Delta_{\max} \leq 10 \cdot 10^{-3} \Longrightarrow \frac{R_1 R_2 C}{R_1 + R_2} \leq 10 \cdot 10^{-3}.$$

(d) This filter was designed by someone who wanted to remove some high-frequency disturbances from $x(t)$, making $y(t)$ a "cleaned" version of $x(t)$. The wanted part of $x(t)$ has lower frequencies than the unwanted disturbance. To get "as much as possible" of the wanted (low-frequency) signal to go through to $y(t)$, the designer decided to set $R_1 = 0$. Is this a good move? Motivate your answer clearly. (2 p)

Solution: Setting $R_1 = 0$ leads to a frequency function

$$H(f) = \frac{R_2}{0 + R_2 + j\omega \cdot 0 \cdot R_2 C} = \frac{R_2}{R_2} = 1,$$

which will let the entire $x(t)$ to go through to $y(t)$, unaffected. This is clearly NOT a good move if the intention was to remove high-frequency disturbances from $x(t)$.

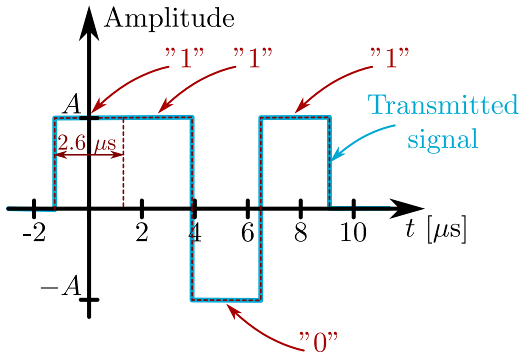
2. Rectangular basis-pulses are used to transmit a binary sequence, where amplitude $-A$ represents "0" and amplitude $+A$ represents "1". This constitutes 2-ASK transmission and the transmission rate is set to 384 kbit/s. Transmission takes place in the base-band, i.e., *without* up-conversion to (modulation on) a carrier.

(a) Sketch, as accurately as possible, the transmitted signal produced by the binary sequence $\mathbf{b} = (1\,1\,0\,1)$. The sketch should include accurate amplitude and time scales. (For control purposes, enter the max/min values of the resulting signal and the symbol time in the corresponding text boxes below.) (3 p)

Solution: To transmit at 384 kbit/sek, using binary 2-ASK signaling, we need to send one symbol every $T_s = 1/(386 \cdot 10^3) = 2.6 \cdot 10^{-6} = 2.6\ \mu$s. This means that our rectangular pulses should be (they can be shorter, but then we use more spectrum than necessary)

$$p(t) = \pm A \text{ rect}\left(\frac{t}{T_s}\right) = \pm A \text{ rect}\left(\frac{t}{2.6 \cdot 10^{-6}}\right),$$

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. The frequency scale and general shape of the spectrum are important to get right, not absolute amplitude levels. (For control purposes, enter the one-sided bandwidth, in terms of the frequency of the first spectrum zero, in the text box below.) (4 p)

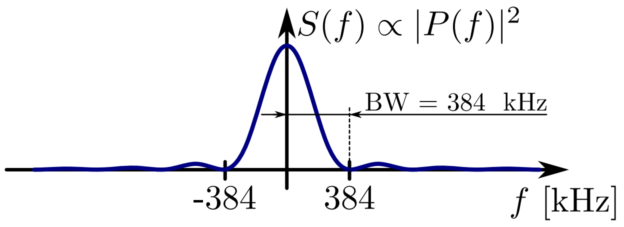
Solution: The Fourier transform of the transmission pulse determines the spectrum of the signal. It is (using Property 5, time scaling, and Fourier transform pair (h))

$$P(f) = \pm T_s A \text{ sinc}(T_s f) = \pm 2.6 \cdot 10^{-6} A \text{ sinc}(2.6 \cdot 10^{-6} f)$$

The spectrum of the signal is proportional to the magnitude-squared of the pulse Fourier transform (the absolute level depends on the load resistance, which is not given in the problem), i.e.,

$$S(f) \propto |P(f)|^2 = (2.6 \cdot 10^{-6} A)^2 \text{ sinc}^2(2.6 \cdot 10^{-6} f)$$

The spectrum therefore looks something like this,



and the bandwidth (as defined in the problem) is BW = 384 kHz.

(c) Now, let's assume that 8-ASK (eight different amplitude levels, e.g. $\pm A$, $\pm 3A$, $\pm 5A$ and $\pm 7A$) is used, instead of 2-ASK, also with rectangular pulses. What is the new symbol time, given that the bit rate remains unchanged? How is the spectrum bandwidth affected by this choice of modulation, compared to the 2-ASK case? Motivate clearly. (3 p)

Solution: Since each transmitted pulse carries 3 bits ($8 = 2^3$ alternatives, gives 3 bits), the symbols can be transmitted *tree times slower*, while still maintaining the bit rate. This means that the symbol time with 8-ASK becomes $T_s = 3 \cdot 2.6 \cdot 10^{-6} = 7.8$ μs. As a consequence, the spectrum becomes one-third as wide, i.e., instead of a 384 kHz bandwidth (as defined above), we only need 128 kHz.

3. Variable-length prefix-free source codes can often be made more efficient (perform closer to what is optimal in terms of average number of bits/symbol for a given source) by encoding several source symbols together, rather than encoding them one-by-one. This, however, brings about additional "complexity" when creating the codewords (there are more of them). Let's investigate this in some detail, under the assumption that our source (stochastic variable) U has M different outcomes/symbols $\{U = u_i\}$, with respective probabilities $P_U(u_i)$, for $i = 1, 2, \ldots, M$. The sequences generated by the source, $\mathbf{U} = U_1U_2U_3\ldots$, are composed of independently drawn outcomes from U.

For this problem we use only $M = 4$ different symbols, but have two different probability distributions, $P_U^{(A)}(u_i)$ and $P_U^{(B)}(u_i)$, indicated by superscripts (A) and (B), respectively:

U	$P_U^{(A)}(u_i)$	$P_U^{(B)}(u_i)$
u_1	0.500	0.500
u_2	0.239	0.250
u_3	0.161	0.125
u_4	0.100	0.125

(a) What is the uncertainty of the source, for each of the two probability distributions, (A) and (B)? (3 p)

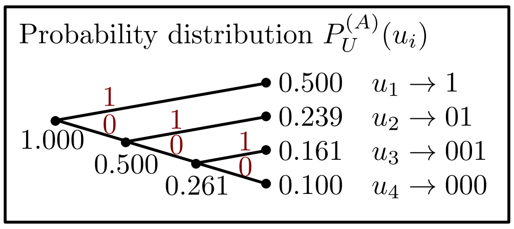
Solution: The uncertainty is given by the entropy, for the two cases,

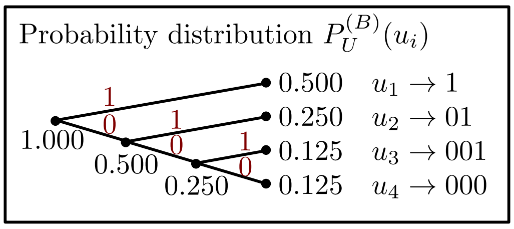
$$\begin{aligned}
H^{(A)}(U) &= -\sum_{i=1}^{4} P_U^{(A)}(u_i)\log_2 P_U^{(A)}(u_i) \\
&= -0.500\log_2 0.500 - 0.239\log_2 0.239 - 0.161\log_2 0.161 \\
&\quad - 0.100\log_2 0.100 \approx 1.75 \text{ bit} \\
H^{(B)}(U) &= -\sum_{i=1}^{4} P_U^{(B)}(u_i)\log_2 P_U^{(B)}(u_i) \\
&= -0.500\log_2 0.500 - 0.250\log_2 0.250 - 0.125\log_2 0.125 \\
&\quad - 0.125\log_2 0.125 \approx 1.75 \text{ bit}
\end{aligned}$$

(b) Derive optimal variable-length prefix-free binary code words, for each of the two probability distributions, (A) and (B), when encoding symbol-by-symbol. Calculate the corresponding average code-word lengths, $\overline{W}^{(A)}$ and

$\overline{W}^{(B)}$, and comment on how close they are to what is ultimately possible and if grouping more and more symbols together potentially can improve the efficiency? (4 p)

Solution: Huffman codes are optimal prefix-free variable-length codes:





Both sets of code words are "equal" in the sense that they have the same code-word lengths (and equal code words, if branches in the Huffman trees are labeled the same way), but the average code-word lengths (here adding internal node probabilities)

$$\overline{W}^{(A)} = 1.000 + 0.500 + 0.261 = 1.761 \text{ bit/symbol}$$
$$\overline{W}^{(B)} = 1.000 + 0.500 + 0.250 = 1.750 \text{ bit/symbol}$$

are different. For probability distribution (A) the average code-word length is about 0.11 bit/symbol away from what is ultimately possible, while for (B) the average code-word length equals the uncertainty and cannot be improved by grouping symbols.

(c) Compare the expression for uncertainty/entropy, $H(U)$, with this expression,

$$\overline{W}^{(*)} = \sum_{i=1}^{M} P_U^{(*)}(u_i) w_i^{(*)},$$

for calculating average code-word length. Here (*) indicates either (A) or (B), and $w_i^{(*)}$ is the length of the code word for symbol u_i. What conclusion do you draw about which type of probability distributions lead to symbol-by-symbol Huffman codes that cannot be improved by grouping more symbols together? (3 p)

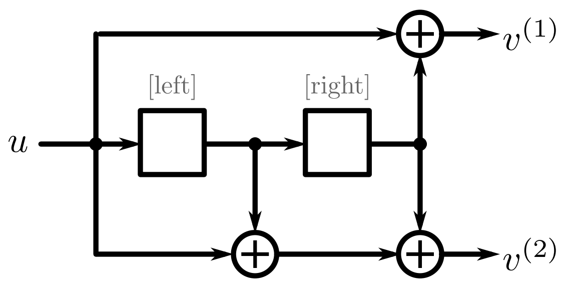
Hint: *Studying the two probability distributions, (A) and (B), and the outcome of (b) may help you to draw the right conclusion.*

Solution: If our code words have lengths $w_i^{(*)} = -\log_2 P_U^{(*)}(u_i)$, the average code word length becomes

$$\overline{W}^{(*)} = \sum_{i=1}^{M} P_U^{(*)}(u_i)(-\log_2 P_U^{(*)}(u_i)) = -\sum_{i=1}^{M} P_U^{(*)}(u_i) \log_2 P_U^{(*)}(u_i),$$

which equals the uncertainty $H(U)$ and no improvement can be achieved by grouping more symbols together. Since variable-length code words need to have an integer number of bits (cannot have fractional bits), $-\log_2 P_U^{(*)}(u_i)$ must be an integer, and this is only possible for this selection of code-word lenths if all probabilities in the distribution are on the form $P_U^{(*)}(u_i) = 2^{-\text{"some positive integer"}}$. This is the case for probability distribution (B), where the probabilities are $0.500 = 2^{-1}$, $0.250 = 2^{-2}$, $0.125 = 2^{-3}$, and $0.125 = 2^{-3}$ (and the corresponding code words have lengths 1, 2, 3, and 3).

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

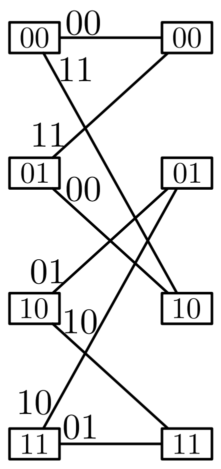


Solution: Before doing anything else, let's figure out the state transitions and encoder outputs, since we are going to need them in more than one sub-problem. Let's list all combinations of inputs and encoder states in a table, and calculate corresponding outputs and next states:

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	01	01
1	10	10	11
0	11	10	01
1	11	01	11

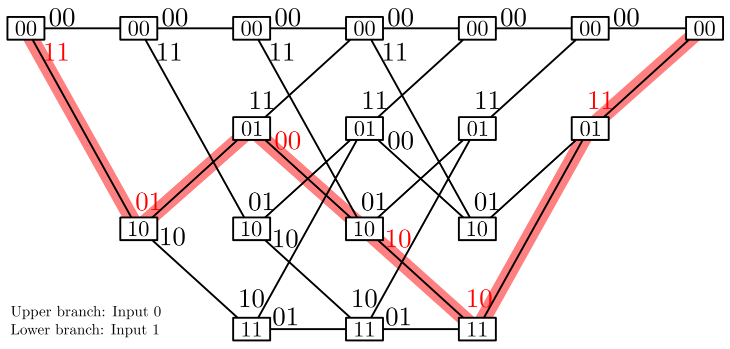
(a) Draw one complete trellis stage for the above encoder, with states [left][right] in the order 00, 01, 10, 11, from top to bottom. Include all state transitions, as well as the encoder output $v^{(1)}v^{(2)}$ at each transition. (For control purposes, enter the two encoder outputs when exiting state 11 in the text boxes below.) (3 p)

Solution: Based on the state transition table created above, the trellis stage becomes



(b) Given the input sequence $\boldsymbol{u} = (u_1\ u_2\ \dots\ u_6) = (1\ 0\ 1\ 1\ 0\ 0)$, what is the corresponding code word $\boldsymbol{v} = (v_1^{(1)}v_1^{(2)}\ v_2^{(1)}v_2^{(2)}\ \dots)$? (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.) (2 p)

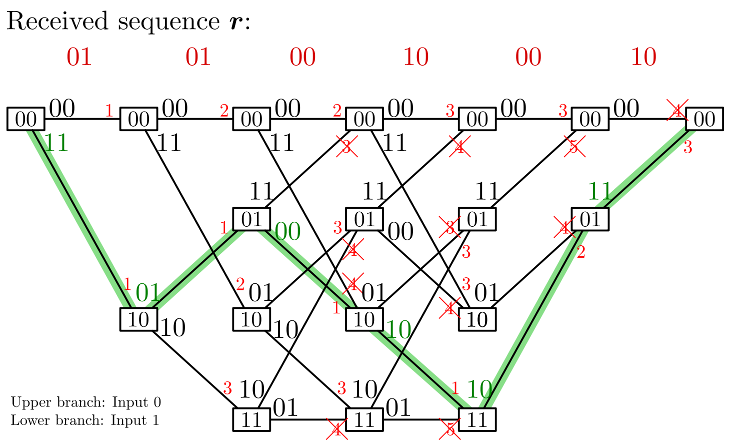
Solution: There are several ways of performing this encoding. It is possible to execute the encoding directly according to the encoder block diagram, use the state-transition table (with inputs, states, next states, and outputs), or we can create a complete trellis (using the trellis stage from (a)). 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 becomes $\boldsymbol{v} = (11\ 01\ 00\ 10\ 10\ 11)$ (marked red).

(c) Show clearly how you execute the Viterbi algorithm to decode the received sequence $\boldsymbol{r} = (01\ 01\ 00\ 10\ 00\ 10)$. What is the resulting decoded code-sequence $\hat{\boldsymbol{v}} = (\hat{v}_1^{(1)}\hat{v}_1^{(2)}\ \hat{v}_2^{(1)}\hat{v}_2^{(2)}\ \ldots)$? (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 and eliminate one of them randomly. Executing the Viterbi algorithm for the given received sequence gives:



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

5. Consider an RSA public key crypto system with the public encryption key $(n, e) = (697, 21)$. Find the plaintext P corresponding to the ciphertext $C = 421$. Motivate your answer carefully, by showing all calculation steps in detail. (10 p)

Hint: *One of the factors of n is $p = 41$.*

Solution: The first prime factor in n is given as $p = 41$. The other is $q = n/p = n/41 = 17$. 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 (mod $\Phi(n)$), 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:

$$\begin{aligned} 640 &= 30 \cdot 21 + 10 \\ 21 &= 2 \cdot 10 + 1 \\ 10 &= 10 \cdot 1 + 0 \leftarrow \text{ Not needed.} \end{aligned}$$

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

$$\begin{aligned} 1 &= 21 - 2 \cdot 10 \\ &\text{[Use that } 10 = 640 - 30 \cdot 21] \\ 1 &= 21 - 2 \cdot (640 - 30 \cdot 21) = 21 \cdot 61 - 2 \cdot 640 \end{aligned}$$

Here we identify $d = 61$ (the factor multiplied by $e = 21$).

Decryption can now be done as

$$P = C^d = 421^{61} \pmod{697}.$$

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

$$61 = 32 + 16 + 8 + 4 + 1$$

and we get

$$P = C^{61} = 421^{32} \cdot 421^{16} \cdot 421^{8} \cdot 421^{4} \cdot 421^{1} \pmod{697}.$$

Calculating these exponents of 421, successively, starting with 421^1, we get (the ones we need in **bold**)

$$\begin{aligned} 421^1 &\equiv \mathbf{421} \pmod{697} \\ 421^2 &\equiv 203 \pmod{697} \\ 421^4 = 203^2 &\equiv \mathbf{86} \pmod{697} \\ 421^8 = 86^2 &\equiv \mathbf{426} \pmod{697} \\ 421^{16} = 426^2 &\equiv \mathbf{256} \pmod{697} \\ 421^{32} = 256^2 &\equiv \mathbf{18} \pmod{697} \end{aligned}$$

We now recover the plaintext as

$$\begin{aligned} P &= 18 \cdot 256 \cdot 426 \cdot 86 \cdot 421 \pmod{697} \\ &= (18 \cdot 256 \pmod{697}) \cdot (426 \cdot 86 \pmod{697}) \cdot 421 \pmod{697} \\ &= 426 \cdot 392 \cdot 421 \pmod{697} \equiv \mathbf{30} \pmod{697} \end{aligned}$$