Name:

Note:

- You have 2 h 45 min to work at the exam.
- The problems can be solved in any order.
- The code will be evaluated according to the usual criteria, namely correctness, speed, form, and readability. Short comments that allow us to follow what you are doing will improve readability.
- Start by downloading from Moodle and unzipping the file `final_2014`. Every problem for which you are asked to write MATLAB code has a folder inside `final_2014`.
- At the end of the exam, upload your MATLAB files as a single archive.

Problem 1. (3p) *(Short Questions)* (Paper and Pencil)

(a) (1.5p) Assume that the C/A code for a GPS satellite has 2046 samples. When you turn on a GPS receiver, what is the minimum number of samples required, to be sure that you received a complete C/A code?

(b) (1.5p) What is the minimum number of satellites required to estimate the position of a GPS receiver? Justify your answer.

Problem 2. (6p) *(Third Degree Polynomial)* (MATLAB)

Complete the MATLAB function `PolyDeg3.m` that takes a vector $x = (x_0, x_1, x_2, x_3)$ and a vector $y = (y_0, y_1, y_2, y_3)$ and returns the vector $p = (p_0, p_1, p_2, p_3)$ of coefficients to the third degree polynomial $p(x) = p_0 + p_1 x + p_2 x^2 + p_3 x^3$ such that for $l = 0, 1, 2, 3$, $p(x_l) = y_l$. You are not allowed to use any toolbox function.

Problem 3. (11.5p) *(OFDM)* (MATLAB)

The vector `received_OFDM` (obtained by loading `ofdm_exam.mat`) contains the output of a finite-response channel over which we have transmitted one OFDM block. The OFDM parameters are:

- number of carriers = 256
- length of the cyclic prefix = 20.

In the OFDM block we inserted 16 reference symbols as follows: the i-th position of the block contains a reference symbol iff `mod(i,16)==1`. The variable `reference_symbols` contains the 16 (transmitted) reference symbols. Complete the script `ofdm.m` that implements the following tasks:

(a) (1.5p) Remove the cyclic prefix from the `received_OFDM` sequence and perform the FFT.

(b) (4p) Plot the absolute value of the frequency spectrum of the result, with the x-axis labeled in kHz. The sampling rate, F_s is 10^6 samples per second.

(c) (6p) Estimate the channel coefficients that correspond to the 16 positions of the reference symbols. Use the simplest method, i.e., assume that there is no noise.

Problem 4. (6p) *(Parity Check)* (MATLAB) The file `bits.mat` contains one page of data, organized in frames. The page is organized as a matrix of bits (0s and 1s), and each column represents a frame of length n. The first (top) $n-1$ bits of a frame are the data bits, and the last bit is a parity bit. The parity bit is the modulo 2 sum of the other bits of the frame.

Complete the MATLAB script `parityCheck.m`. It should display a vector, `parityResult`, indicating which frames pass the parity check. The number of elements of this vector is the same as the number of frames, and the ith element is 1 if the ith frame (ith column of `bits`) passes the parity check, and 0 otherwise.

Note: Code efficiency will be considered.

Problem 5. (7.5p) *(GPS)* (Paper and Pencil and MATLAB) Consider a simplified 2D GPS scenario with the Earth in the center of the reference system (Figure 1, where the squares are 1×1). Two "non-moving satellites" are at positions (-1,6) and (5,7), respectively. The satellite-to-receiver distance is 3 and 5, respectively.

(a) (1.5p) Complete the figure with the two satellites and all possible positions of the receiver (qualitatively).

(b) (6p) Complete the script `gps.m`, which returns the position of the receiver. (Notice the hints in the script.)

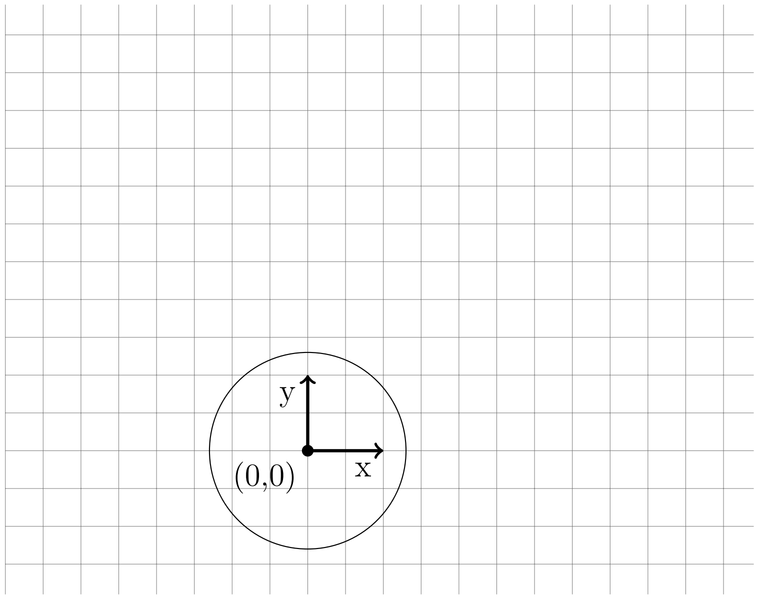


Figure 1: 2D GPS

Problem 6. (12p) *(LDPC-H)* (MATLAB) Complete the script `ldpc_h.m` that implements the following tasks:

(a) (2p) By loading the file `H_exam.mat`, you obtain a sparse parity check matrix `H` consistent with the form.

$$\begin{array}{c|c|c|c|} & n-m & g & m-g \\ \hline m-g & A & B & T \\ \hline g & C & D & E \\ \hline \end{array}$$

T lower triangular with 1s on the diagonal.

Determine the code rate (i.e., the ratio between the number of information bits and the codeword length).

(b) (2p) Determine the value of g.

(c) (8p) Create a matrix named `Hcols` that describes the locations of the 1s of `H` according to the following format. The size of the matrix is n x c, where n is the number of column of `H` and c is the maximum number of 1s that there is in a column of `H`. In row i of `Hcols`, write the position of the 1s in the ith column of `H`, followed by `NaN` to fill the row as needed.

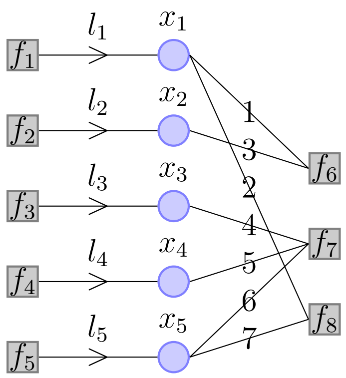
Hint1: you may fill one row of `Hcols` at a time using a `for` loop.

Hint2: certain operations applied to a sparse matrix return a sparse matrix, and the result can be converted into normal form via `full`. For instance, to obtain the element i, j of the sparse matrix `H` you do `full(H(i,j))`.

Hint3: if you do not manage to find c for a general binary matrix `H`, use the fact that for the given `H`, the first column has the largest number of 1s. Using this hint will cost you 2 points.

Problem 7. (14p) *(LDPC-Decoder)* (Paper and Pencil) Consider a binary parity check code described by the factor graph below. The values l_i are the observed log likelihood ratios.

(a) (2p) Write a parity check matrix H for this code.



(b) (3p) For the first iteration of the sum-product algorithm, determine the messages from the variable nodes to the check nodes, $\tilde{V}_{i\to j}$. (The tilde reminds us that these are log likelihood ratios.) Messages from the factor nodes on the right are initialized to 0.

$\tilde{V}_{1\to 6} =$

$\tilde{V}_{1\to 8} =$

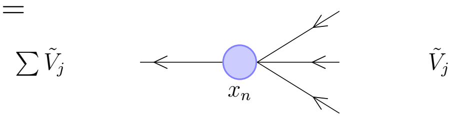
$\tilde{V}_{2\to 6} =$

$\tilde{V}_{3\to 7} =$

$\tilde{V}_{4\to 7} =$

$\tilde{V}_{5\to 7} =$

$\tilde{V}_{5\to 8} =$

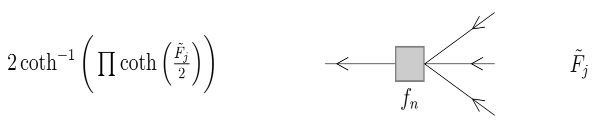


(c) (3p) For the first iteration of the sum-product algorithm, determine the following messages from the check nodes to the variable nodes:

$\tilde{F}_{6\to 1} =$

$\tilde{F}_{7\to 4} =$

$\tilde{F}_{8\to 5} =$



(d) (6p) After how many iterations does the algorithm converge? Justify your answer. (One variable node to check node computation counts for one iteration. The same is true for one check node to variable node computation.)