Lab 1

1. Create a Matlab function `myCDF()` that estimates the cumulative distribution function (CDF) from a vector of data
   - the function requires two arguments and returns one value `p = myCDF(v,x)`
   - `v` is a vector, `x` and `p` are scalar numbers
   - the function computes how many elements from `v` are smaller or equal than `x`, divided to the total number of elements of `v`
2. Use the `myCDF()` function to compute the CDF of a random vector
   - generate a vector `data` with 1000 values from the normal distribution $\mathcal{N}(0, \sigma^2 = 2)$
   - apply `myCDF()` with `v = data` and with $x = -10, -9, -8, ...8, 9, 10$, and store the results in a vector `cdf`
   - plot the resulting vector `cdf` against the values of `n`

Lab 2

1. Simulate threshold-based detection with a single sample, as follows:
   - Generate a vector of 100000 values 0 or $A = 3$, with equal probability (hint: use `rand()` and compare to 0.5)
   - Add over it a random noise with normal distribution $\mathcal{N}(0, \sigma^2 = 1)$
   - Compare each element with $T = \frac{A}{2}$ to decide which sample is logical 0 or logical 1
   - Compare the decision result with the true original vector, and count how many correct detections and how may false alarms have been.
   - Estimate P(hit) and P(false alarm) by dividing the above numbers to the size of the vector

Lab 3

1. Simulate the BPSK sender and channel
   - Generate a vector **`data`** of 1000 values 0 or 1, with equal probability (hint: use `rand()` and compare to 0.5).
   - Generate a vector **`signal`** of 100000 values as follows:
     - for each bit 0 in **`data`**, put a 100-long sine $A \sin(2\pi f n)$ in **`signal`**
     - for each bit 1 in **`data`**, put a 100-long sine $-A \sin(2\pi f n)$ in **`signal`**
     - Use $A = 1$, $f = 1/100$.
   - Plot `signal`.
   - Generate a vector of white gaussian noise with distribution $\mathcal{N}(0, \sigma^2)$, the same length as `signal`, and $\sigma^2 = A/10$.
   - Add the noise to the signal, store result as **`signalplusnoise`**.
   - Plot the resulting signal `signalplusnoise`.

Lab 4

1. Simulate detection of a constant signal with two levels 0 and $A = 5$, based on two samples, as follows:
   - Generate a vector `data` of 1000 values 0 or 1, with equal probability (hint: use `rand()` and compare to 0.5).
   - Generate a matrix named `points`, of size 1000×2, defined as:
     - row i of `points` is $(0, 0)$ if `data(i)` is 0, or
     - row i of `points` is (A, A) if `data(i)` is 1.
   - Add over it a random noise with normal distribution $\mathcal{N}(0, \sigma^2 = 2)$ (use a noise matrix of same size 1000×2). The result should be saved as the matrix `M`.
   - Implement the Maximum Likelihood decision rule for each row i of the received samples.
     - first, create a vector `decision` of 1000 values equal to 0
     - then, for each row `i` from 1 to 1000 in matrix `received`:
       * if $M(i,1)^2 + M(i,2)^2 > [M(i,1) - A]^2 + [M(i,2) - A]^2$, then set `decision(i)` to 0
       * otherwise, set `decision(i)` to 1

Lab 5

1. Generate a 100-samples long sinusoidal signal with frequency $f_0 = 0.01$, and add over it normal noise with distribution $\mathcal{N}(0, \sigma^2 = 2)$. Name the resulting vector `data`. Plot the `data` vector.

2. Estimate the frequency $\hat{f}$ of the signal via Maximum Likelihood estimation:

   - Try all frequency values f_k going from 0 to 0.5, in 500 equally-spaced values (step 0.001), and compute for each f_k the likelihood value

   $$w(k) = -\sum(data - \sin(2\pi f_k n))$$

   - Maximum Likelihood: choose $\hat{f}$ as the value which maximizes the likelihood `w(k)` (use `max()`)
   - Display the estimate value $\hat{f}$