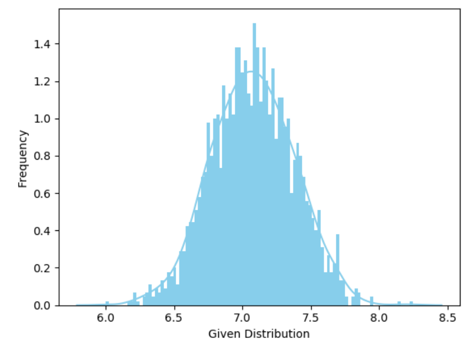
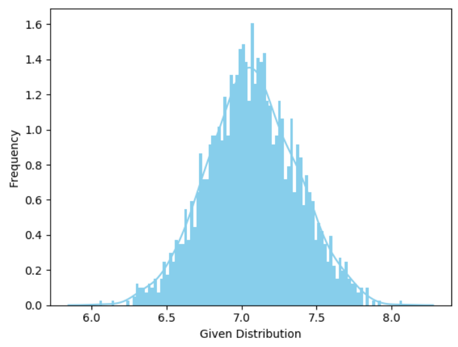
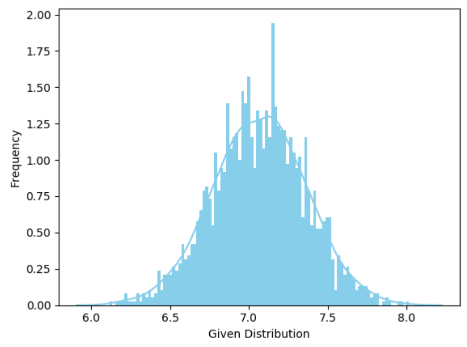
2 Maximum Likelihood Estimation of Model Parameters

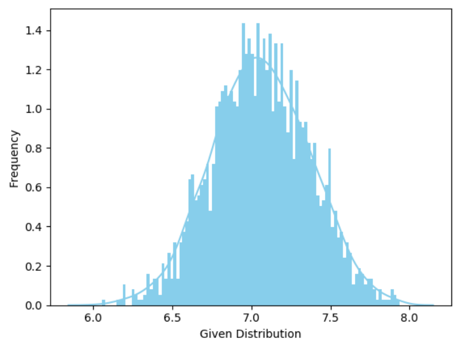
2.1 First scenario

For the first scenario we plotted the frequency of the given distribution of the points in the reference_measurement_1 so that we could analyze its distribution. The graphs for the various anchors obtained are the following:





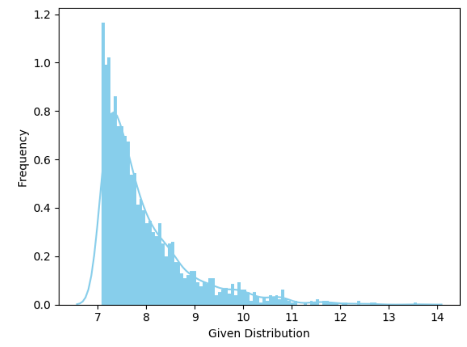


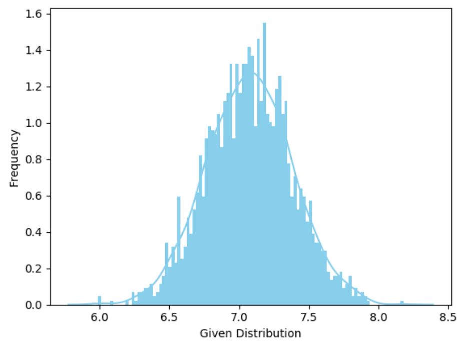


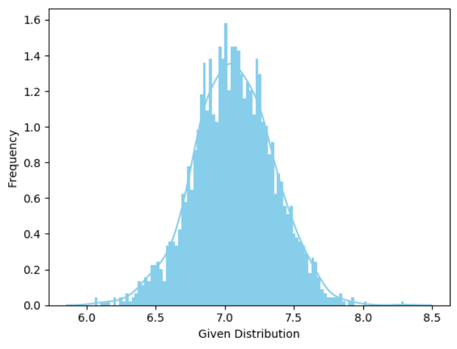
As we can see the distribution is a typical Gaussian distribution.

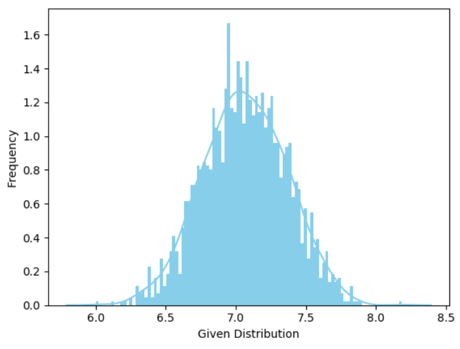
2.2 Second Scenario

We plotted the frequency of the distribution as in the first scenario with the following output:





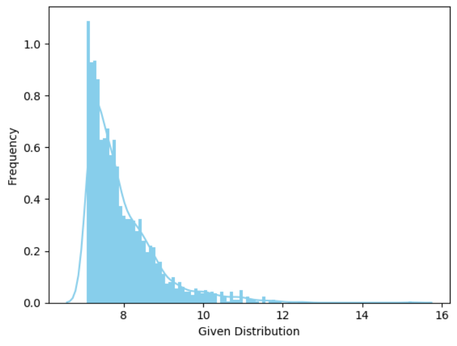


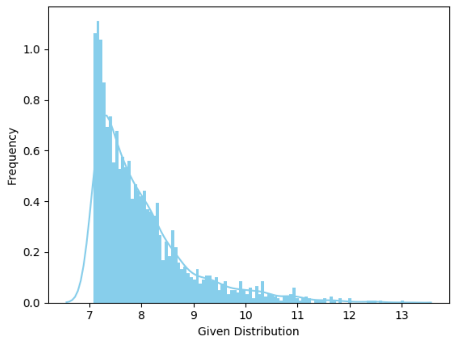


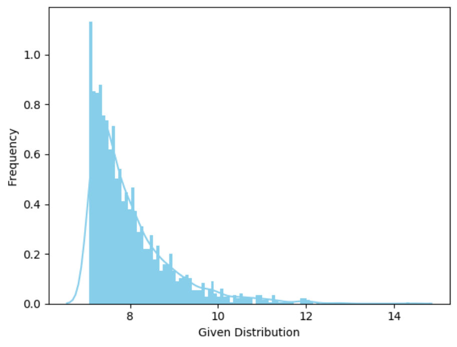
By looking at the plots it’s pretty clear that the measurements of the first anchor are following the Exponential model and the other 3 the Gaussian as in the first scenario.

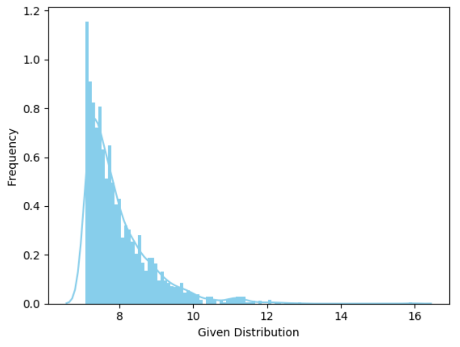
2.3 Third Scenario

As in the others scenarios we first looked at the frequency of the distribution in the plots:









As for the measurements for the first anchor in scenario 2 the distribution of all the measurements in scenario three follow the Exponential model.

2.4 Estimation of parameter

Now that we know the model of each anchor in each scenario we can calculate the parameter θ (λ_i or σ_i depending on the model) for each anchor. For the Guassian model we calculate the parameter using the numpy command np.cov wich take the measuraments for the anchor and return the covariance.

For the Exponential model we calculated the parameter using the Maximum Likelihood Estimation method which steps are shown below.

We start with the definition of exponentially distributed.

$$p\left(\widetilde{d_n}(a_i,\boldsymbol{p})\middle|\boldsymbol{p}\right) = \begin{cases}\lambda_i \cdot e^{-\lambda_i \cdot \widetilde{d_n}(a_i,\boldsymbol{p}) - d_n(a_i,\boldsymbol{p})}\ if\ \widetilde{d_n}(a_i,\boldsymbol{p}) \geq d_n(a_i,\boldsymbol{p}) \\ 0\ else\end{cases}$$

Its likelihood function is

$$\mathcal{L}(\lambda, a_i) = \prod_N p\left(\widetilde{d_n}(a_i,\boldsymbol{p})\middle|\boldsymbol{p}\right) = \prod_N \lambda_i \cdot e^{-\lambda_i \cdot d_n(a_i,\boldsymbol{p})} = \lambda_i{}^N\ e^{-\lambda_i \cdot \sum_N d_n(a_i,\boldsymbol{p})}$$

The derivative with respect to λ_i must equal zero

$$\frac{d\ \ln\left(\mathcal{L}\left(\lambda_i, d_n(a_i,\boldsymbol{p})\right)\right)}{d\lambda_i} = 0$$

To calculate the rate we solve for λ

$$\frac{d\ \ln\left(\mathcal{L}\left(\lambda_i, d_n(a_i,\boldsymbol{p})\right)\right)}{d\lambda_i} = \frac{d\ \ln\left(\lambda_i{}^N\ e^{-\lambda_i \cdot \sum_N d_n(a_i,\boldsymbol{p})}\right)}{d\lambda_i} = \frac{N}{\lambda_i} - \sum_N d_n(a_i,\boldsymbol{p})$$

Finally we get

$$\lambda_i = \frac{N}{\sum_1^N d(a_i,\boldsymbol{p})}$$

2.5 Parameters

- *Scenario 1*

$${\sigma^2}_{A_1} = 0.0909398$$

$${\sigma^2}_{A_2} = 0.08840547$$

$${\sigma^2}_{A_3} = 0.08708974$$

$${\sigma^2}_{A_4} = 0.09243711$$

- *Scenario 2*

$$\lambda_{A_1} = 0.12557937$$

$${\sigma^2}_{A_2} = 0.08374108$$

$${\sigma^2}_{A_3} = 0.0933894$$

$${\sigma^2}_{A_4} = 0.08989452$$

- *Scenario 3*

$$\lambda_{A_1} = 0.12504257$$

$$\lambda_{A_2} = 0.12516356$$

$$\lambda_{A_3} = 0.12545039$$

$$\lambda_{A_4} = 0.12577921$$

3 Estimation of the Position

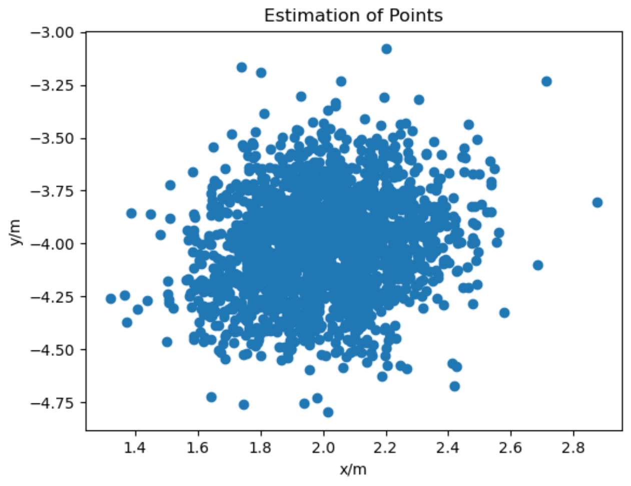
3.1 Least-Squares Estimation of the Position

// TODO

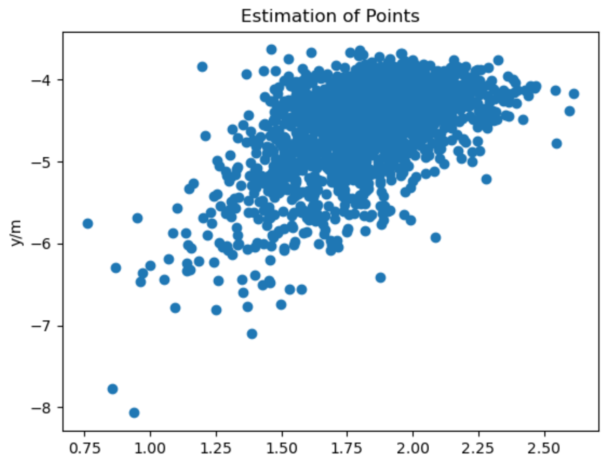
3.2 Gauss-Newton Algorithm for Position Estimation

First of all we have calculated the derivate for the Jacobian Matrix. $d_n(a_i, p)$ is a costant and so in the derivative is canceled. Because of that the result is only the distance between the coordinates of the measured point and the guessed p_start which is randomly calculated based on the uniform distribution within the square spanned by the anchor points. Now that we have the Jacobian Matrix and the formula given in the assignment sheet we can calculate $p^{(t+1)}$. We calculate $p^{(t+1)}$ until tolerance or the maximum number of interaction is reached and then return the founded point to the calling function. We've chosen a tolerance of 10^(-4) for having a good precision but not too many steps. With this tolerance our function never reach more than 9 interaction, we've chosen 8 as max_iter for maintaining a good precision but letting the function exit because of reaching the maximum number of interaction only a few times. After we've estimated one point for each measurement and saved all the points into an array we've calculated the squared errors between the founded point and p_true. We saved all the errors into a list and with the statistics functions stastistics.mean() and statistics.variance() we've calculated the mean and variance of the errors for each scenario. After the calculations have been completed we called the function ecdf and printed out the graph of the CDF-function and we printed out different type of graph for analyzing our results. Our plots and our analysis is in the pages below. Finally we performed all these steps again excluding the first anchor from the calculus for the second scenario.

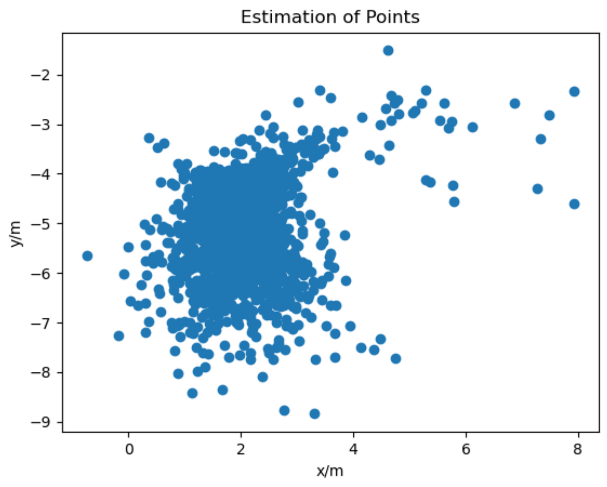
These are the resulting plots of all the estimated points for each scenario:



Scenario 1: As we can see the distribution of the results of the least squares estimation for the first scenario looks pretty homogeneous distributed. This is probably due to the Gaussian model adopted

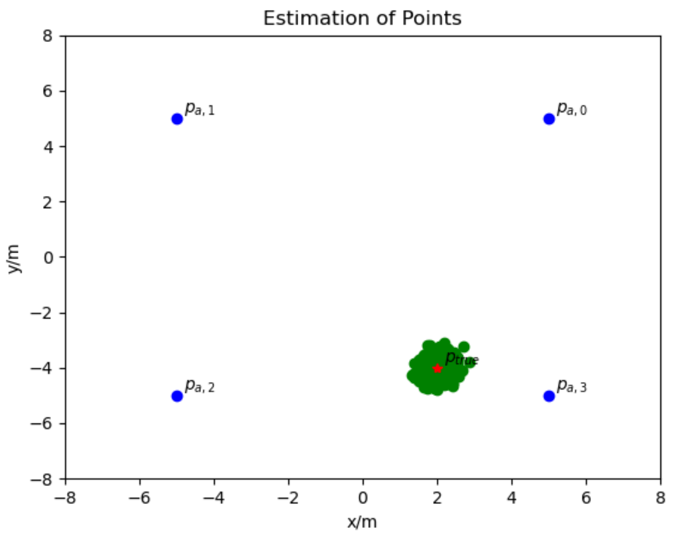


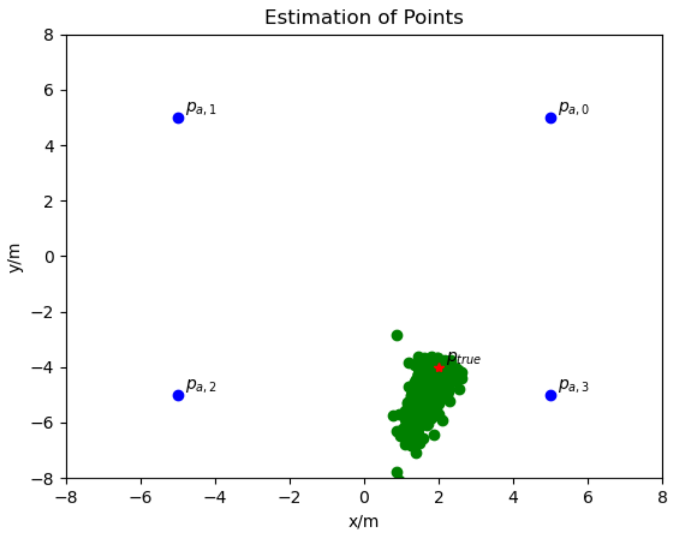
Scenario 2: As in the first scenario the points are pretty homogeneous distributed except for some points at the bottom left of the graph. This is probably due the errors in measurements of the first anchor (the one who follow the Exponential model).

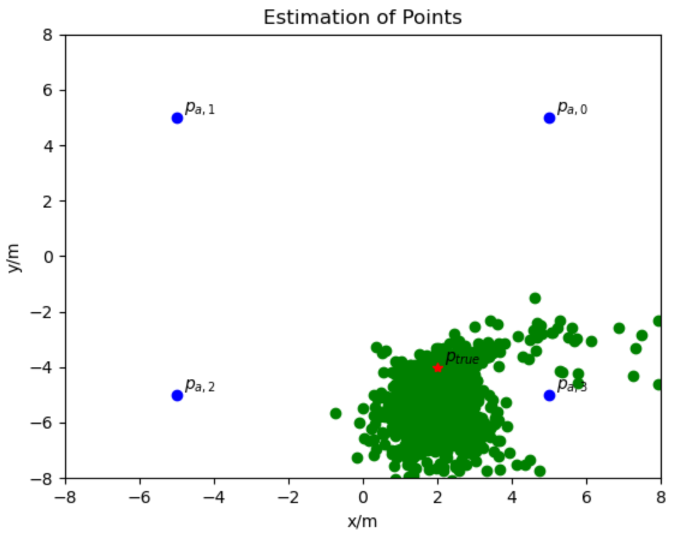


Scenario 3: The points here are grouped in a position the isn't the position of p_true (2, -4) and there are points that are pretty far away from the others. This is probably due the Exponential model used in this scenario

Here we have another perspective of the resulting points of the least squares estimation of position compared to p_true and the anchors (Please note that not all the resulting points are shown in the graphs for having a better perspective, the distribution of all points is shown in the graphs above):







3.3 Variance and Mean of Errors

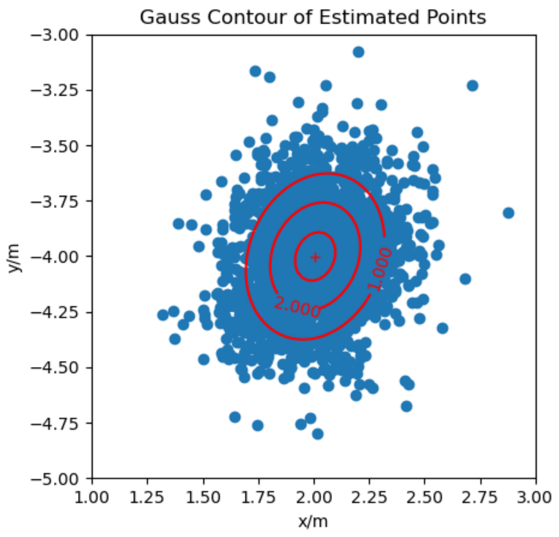
As we can see in the table below variance and mean of the errors is increasing from scenario to scenario. We can so assume that they are depending on the used model and that the Gaussian model present smaller error than the Exponential one.

	Scenario 1	Scenario 2	Scenario 3
Variance	0.021643086124266794	0.2753618900716004	0.7620324949504788
Mean	0.27785493253809157	0.643289440482222	1.3087527883528434

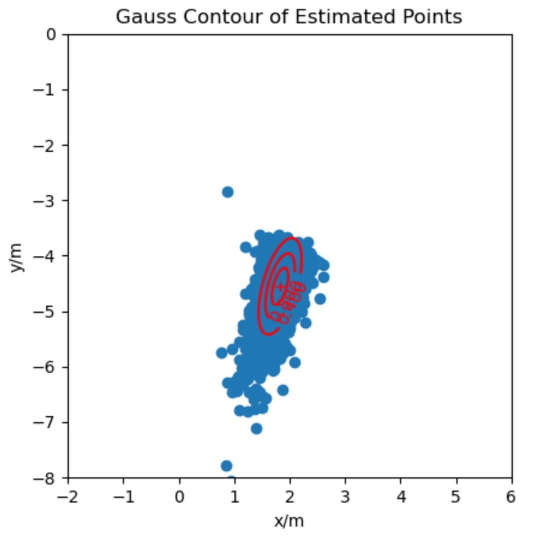
3.4 Gauss Contour of Estimated Points

For printing out the gauss contour graph of our points we've calculated the mean and covariance of our collection of points and we used the helper function plot_gauss_contour(mu,cov,xmin,xmax,ymin,ymax).

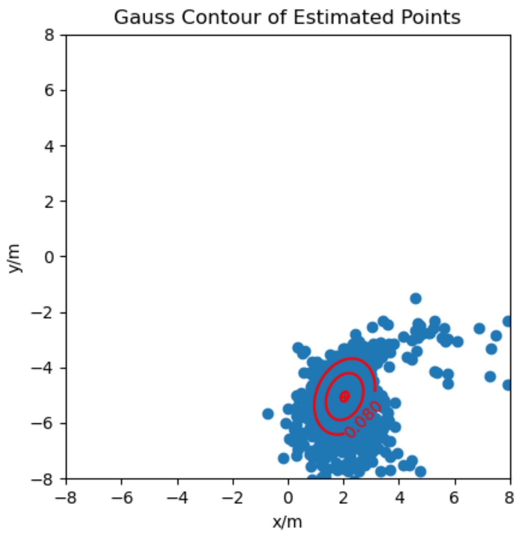
In the first scenario the points are distributed all around the expected area and that confirm the fact that the first scenario is following the Gaussian model. Also the second scenario looks Gaussian except of some point probably due the Exponential anchor. The graph of the third scenario has a lot of point that are far away from the expected area. This is clearly not following the Gaussian model. Our graphs are shown below:



Scenario 1



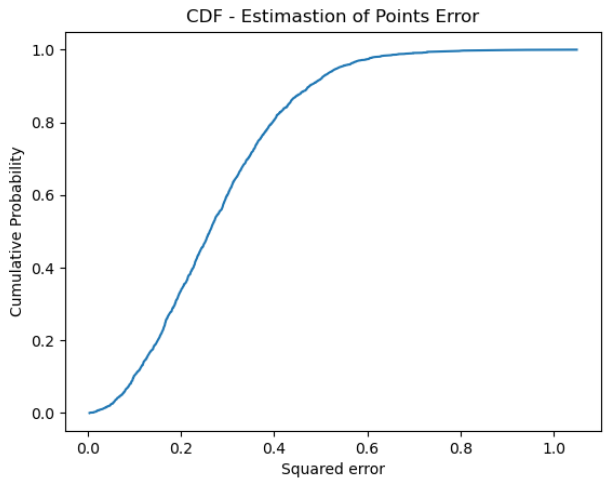
Scenario 2



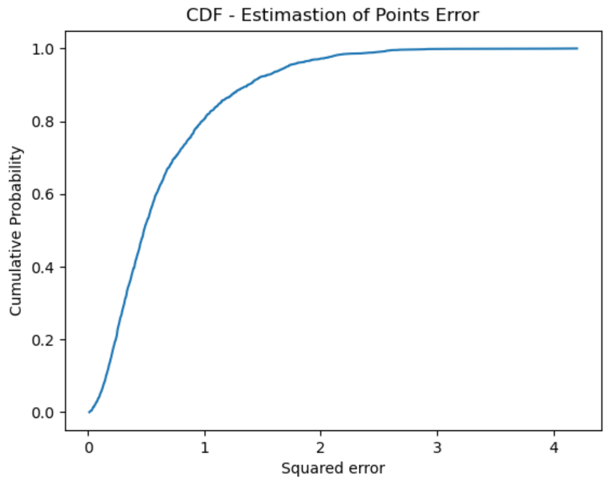
Scenario 3

3.5 CDF Function of errors

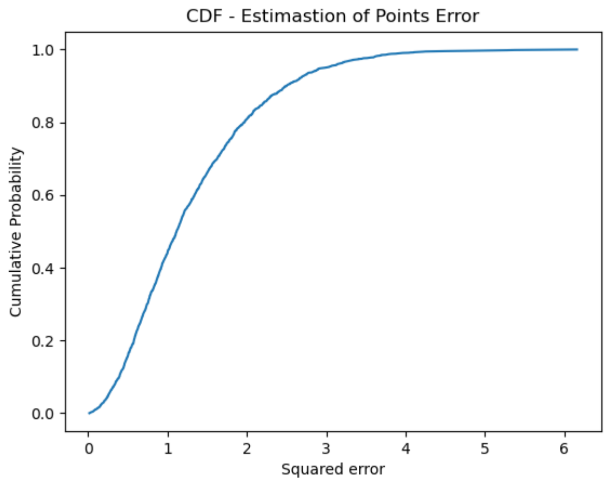
As said before for this part of the assignment we first calculated the squared error between the least squares estimation of point and the given true point. After that we called the function ecdf(realizations) with the squared error list as parameter and printed out the output. From the graphs we can clearly see the difference between the probabilities of big errors in different scenario.



Scenario 1



Scenario 2

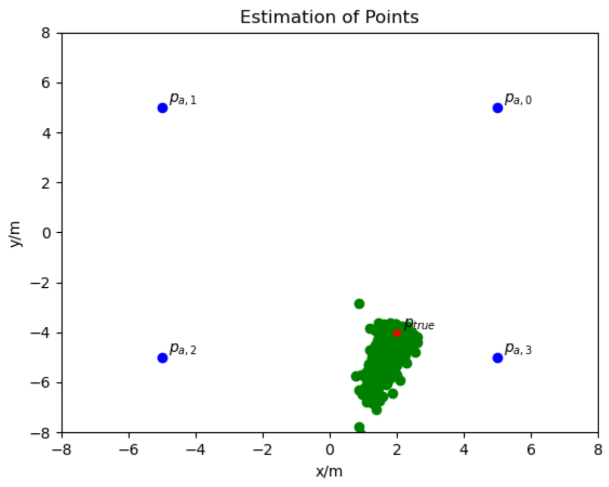


Scenario 3

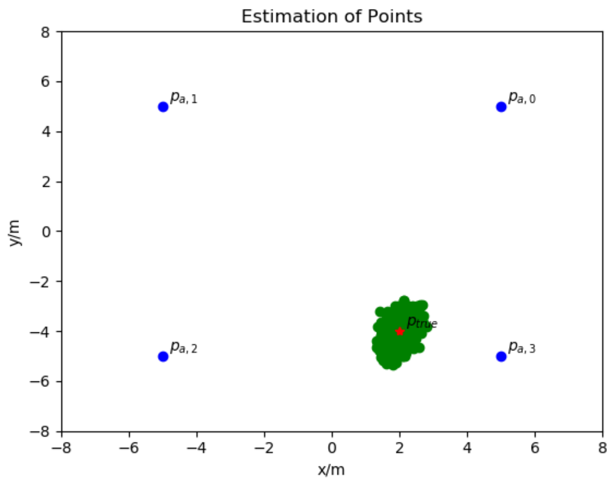
As we can see from these graphs and in particular the scale used on the horizontal axis (squared error) the probabilities of big error always increase from scenario 1 to scenario 3. By looking at the scale in the first scenario the maximum of the cumulative probability (= 1) is reached already with squared error of value 1. In other words we can say that the probability of having an error >1 tents to null and all errors are in the range of 0-1. In the scenario 2 the maximum is reached with 4 as squared error value and for scenario 3 with 6 as value. We can so assume that the Gaussian model is a lot more precise than the Exponential that presents pretty big errors.

3.6 Scenario 2 without Exponential Error

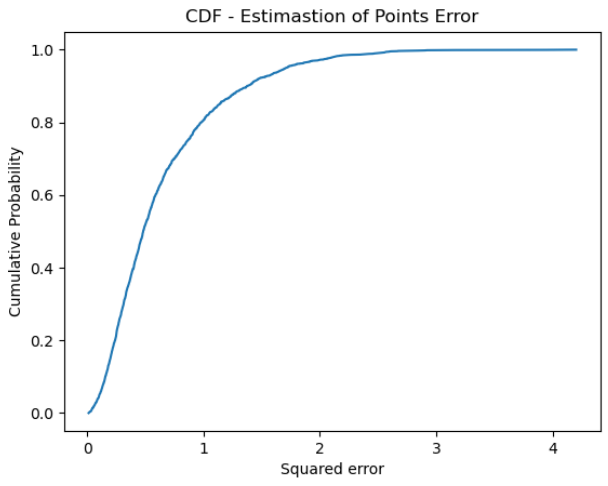
For this step we just performed the Estimation of point as before with the only exception that we excluded the 1 anchor (the exponential one) from our calculus. With this trick the result of the estimation of position is a lot more precise and the probability of big errors decreased a lot. Here below we give a comparison between the second scenario with and without the first anchor:



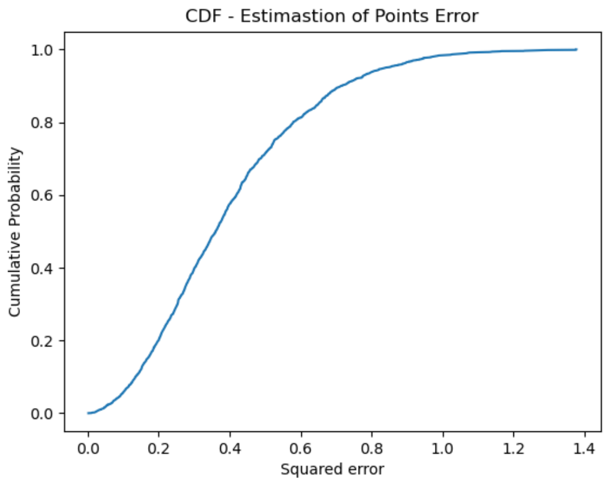
With Exponential Anchor



Without Exponential Anchor



With Exponential Anchor



Without Exponential Anchor