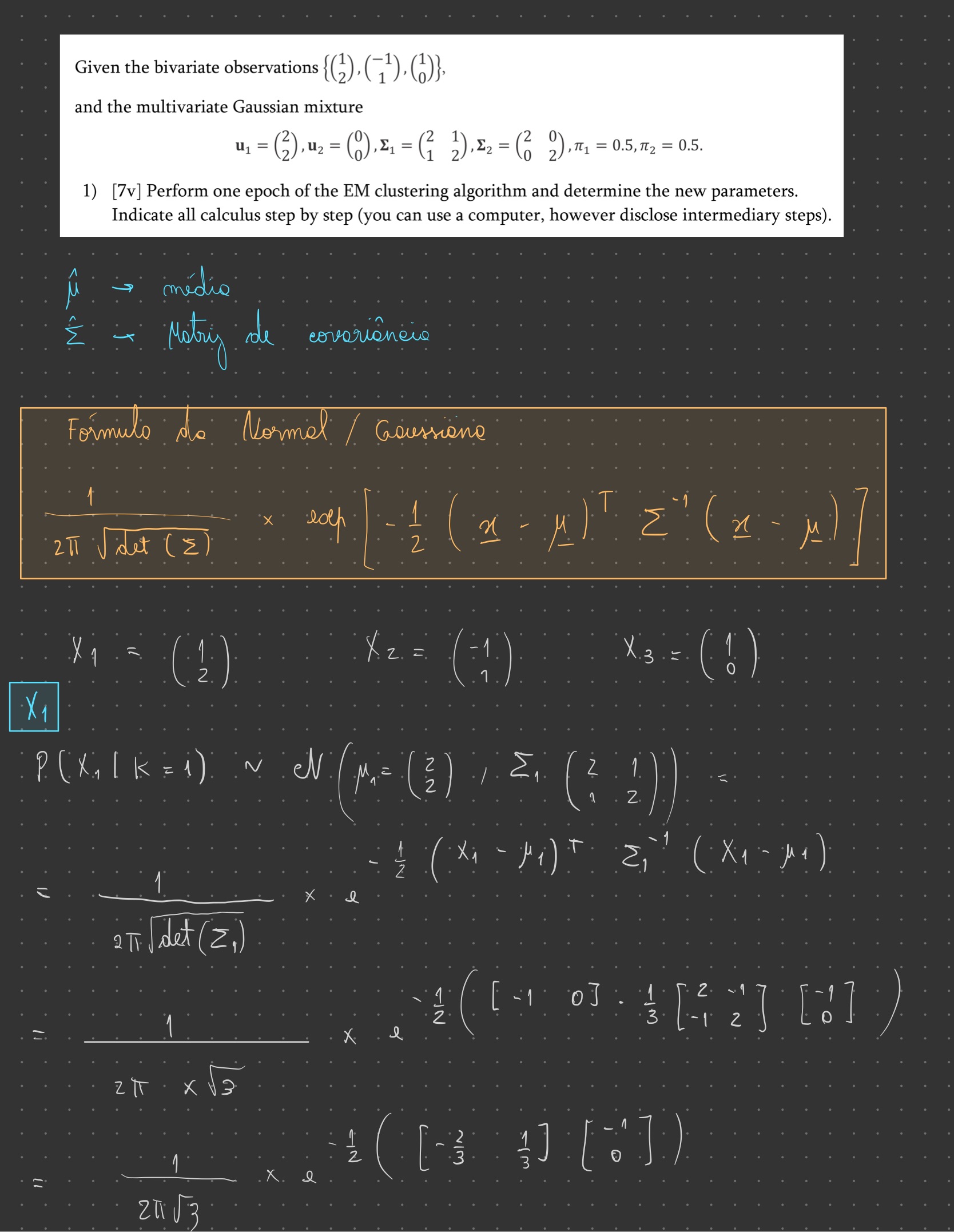
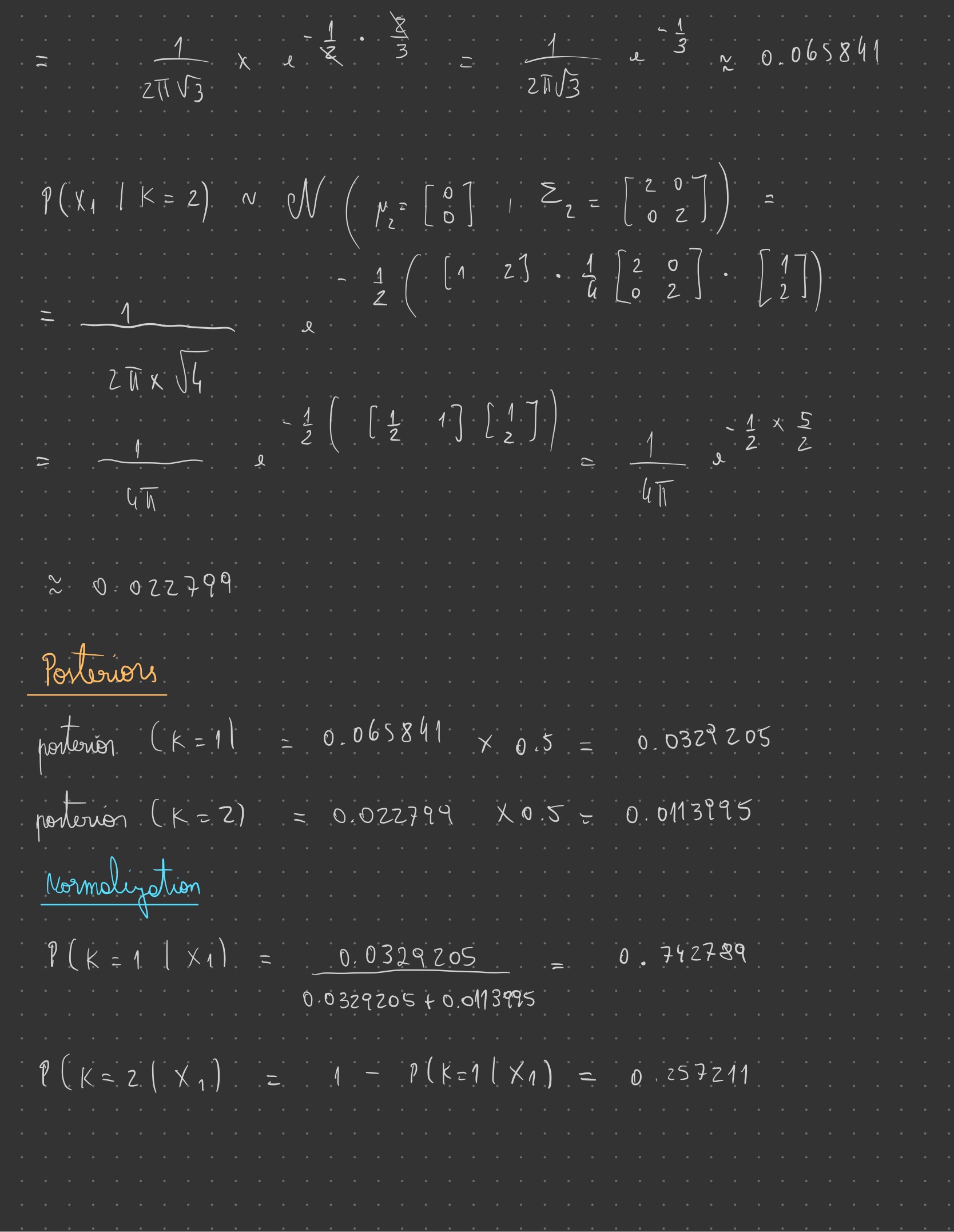
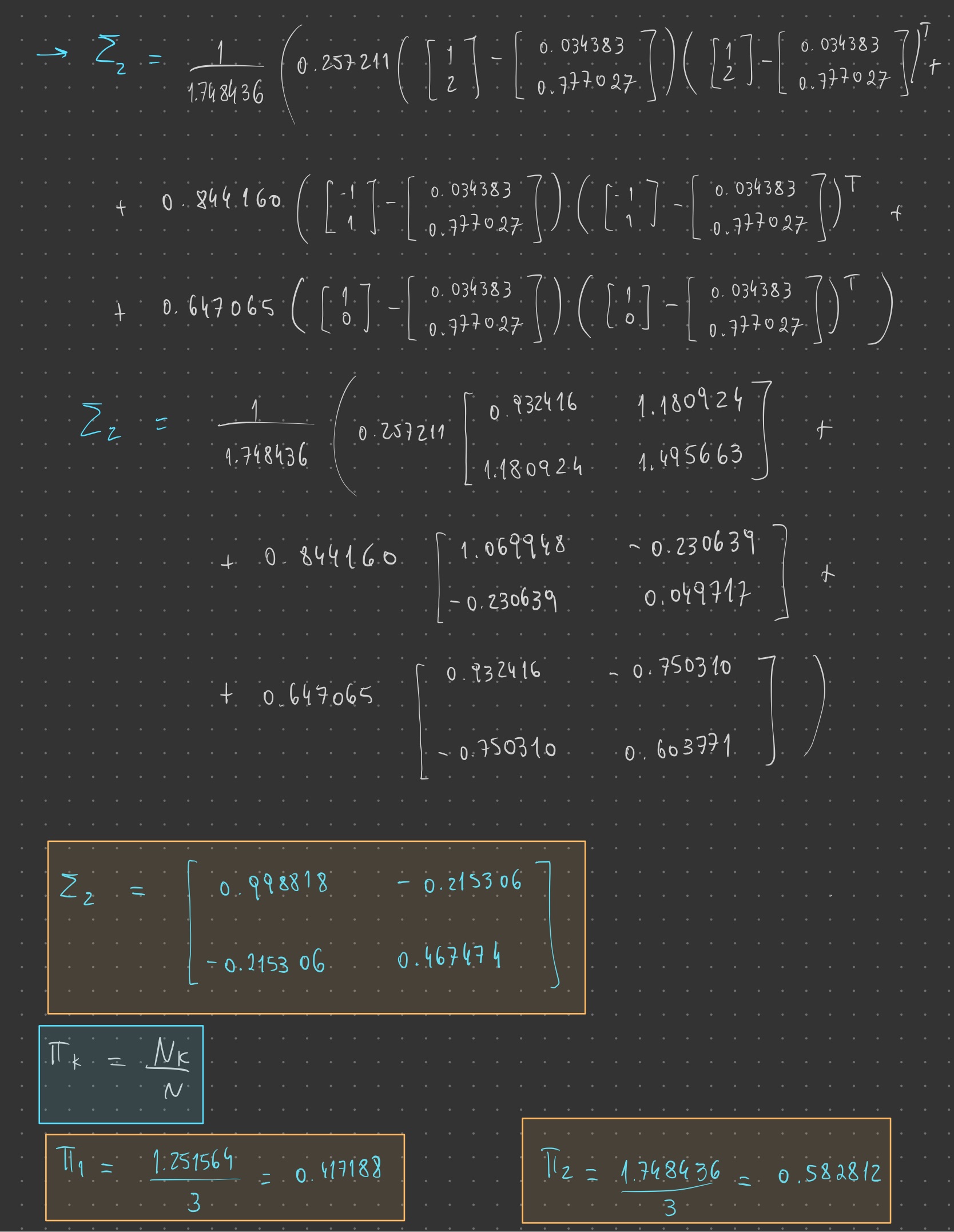
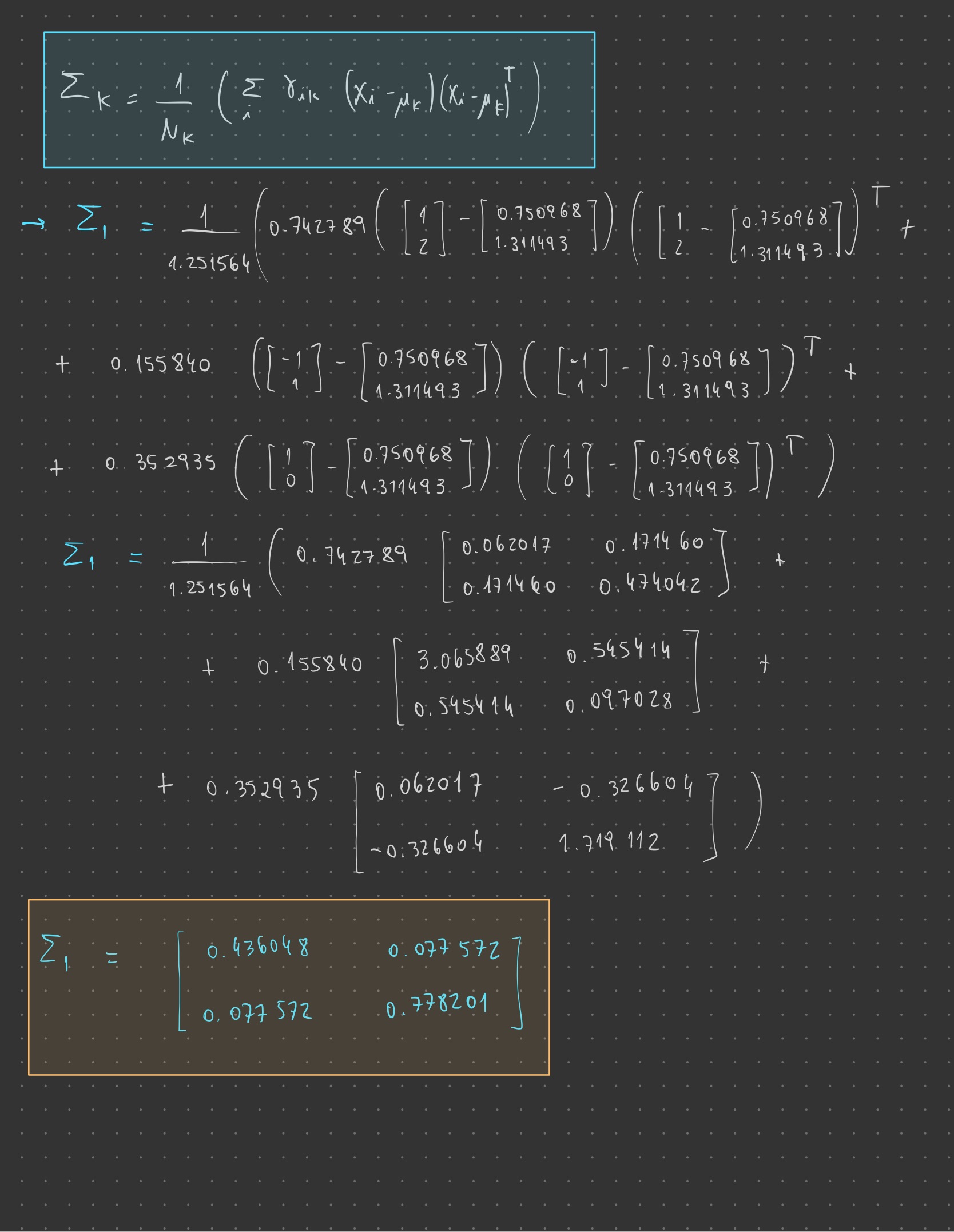
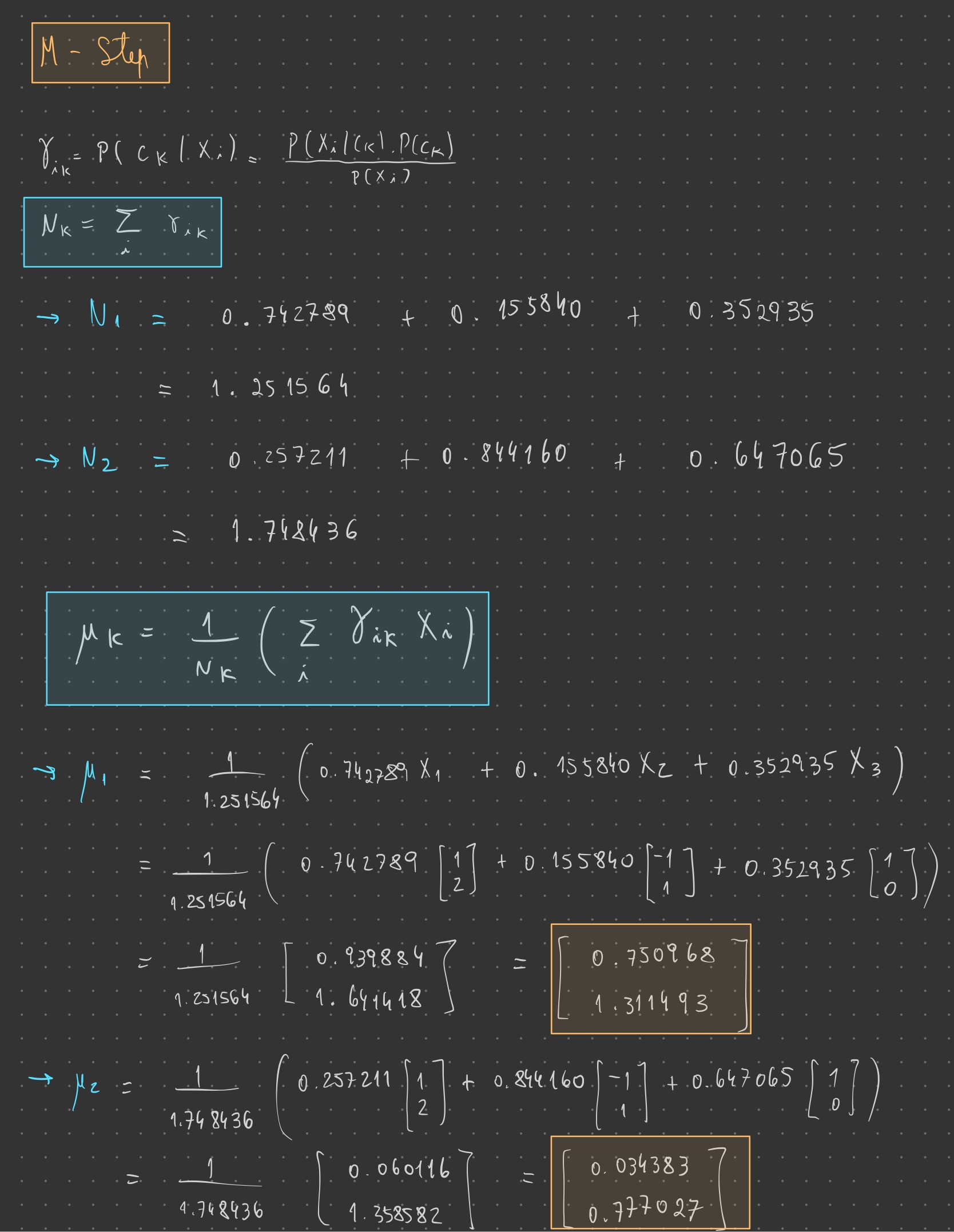
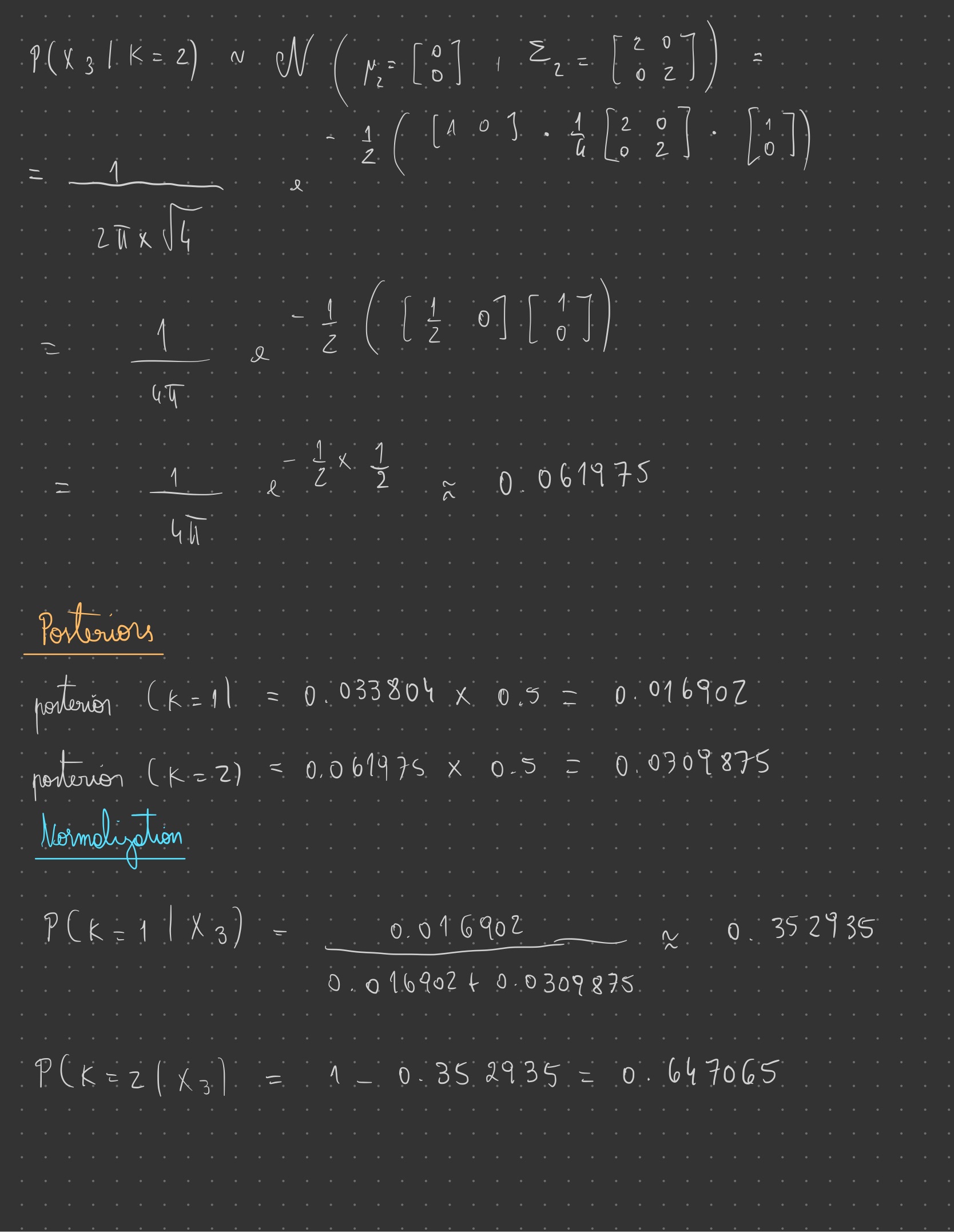
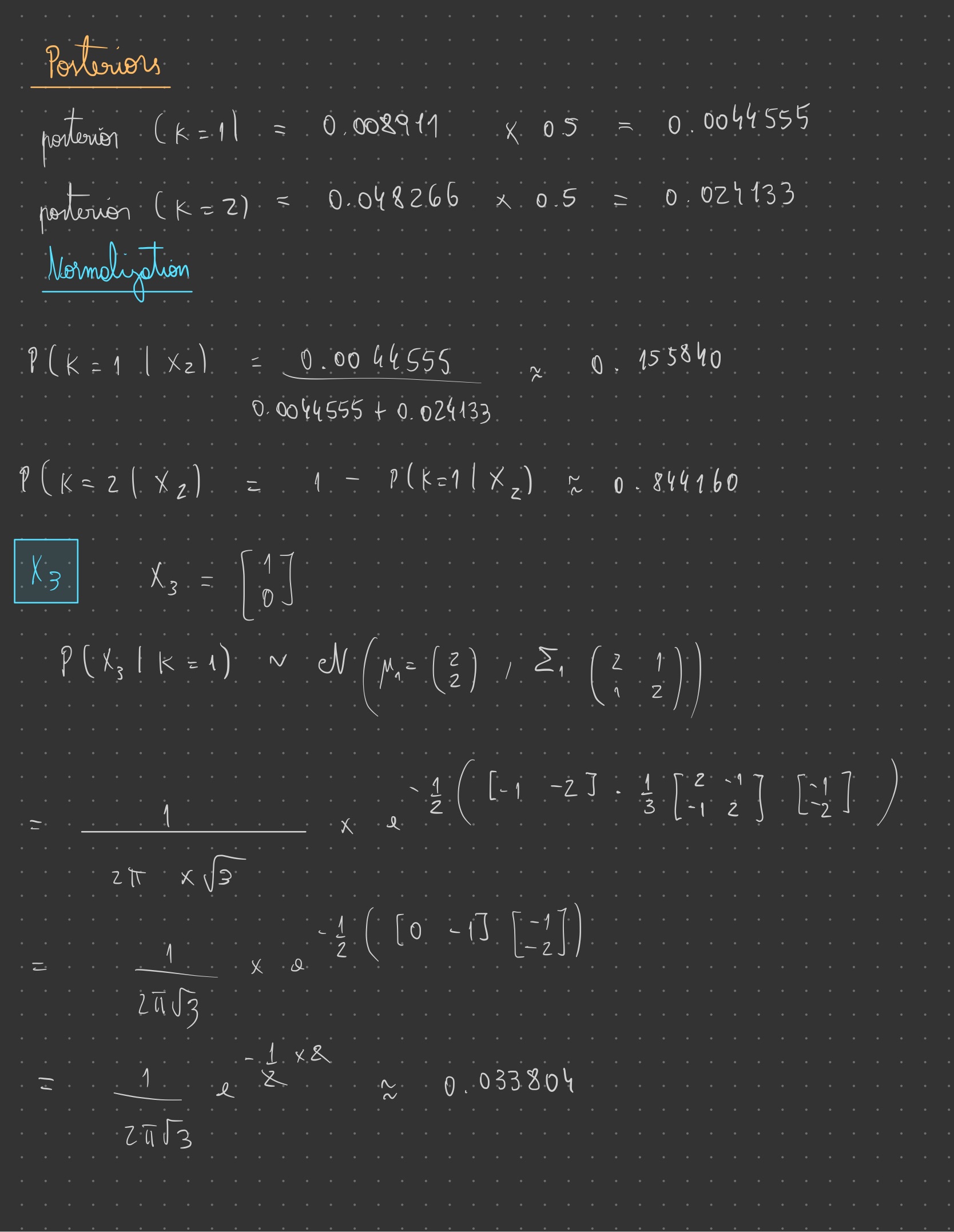
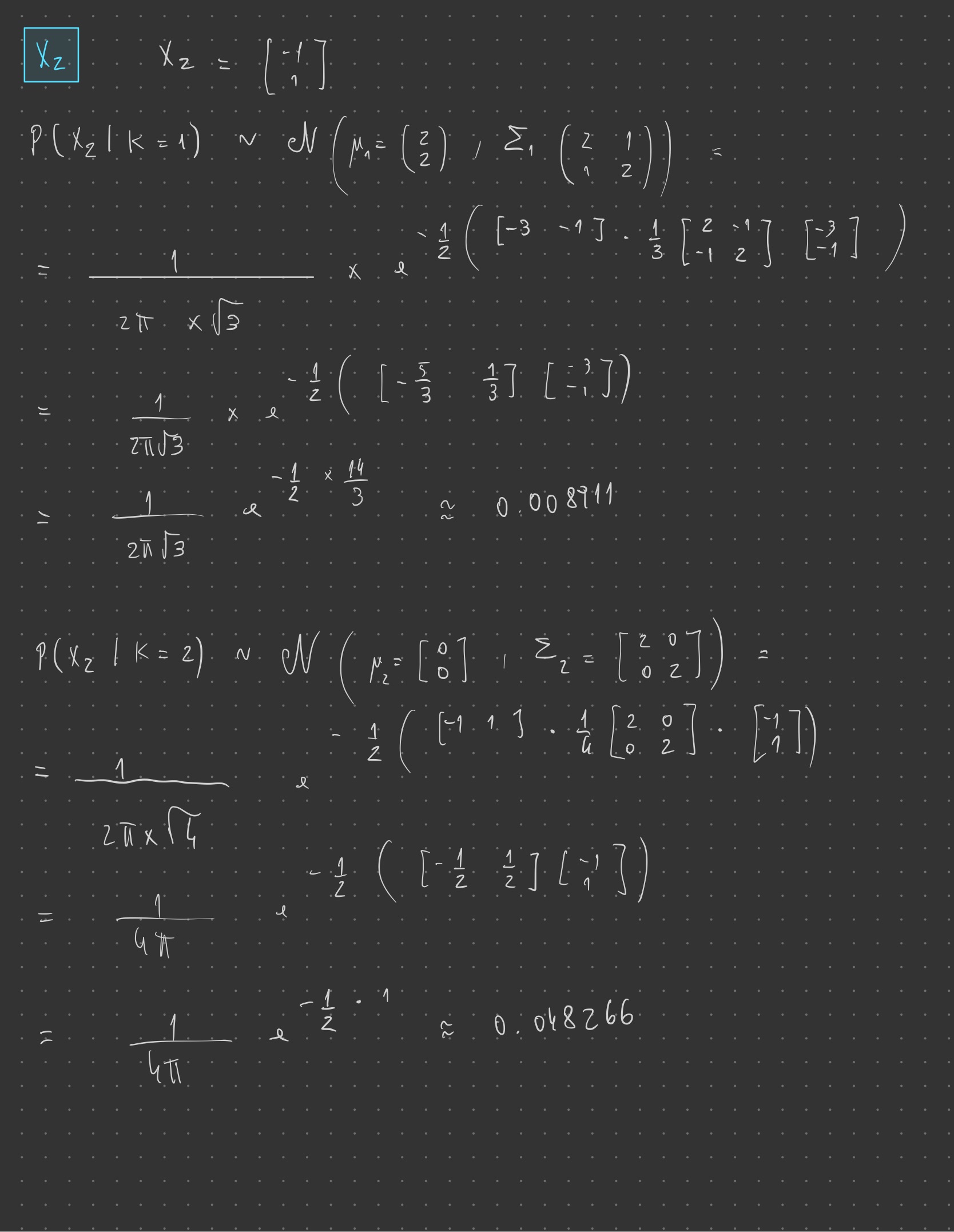
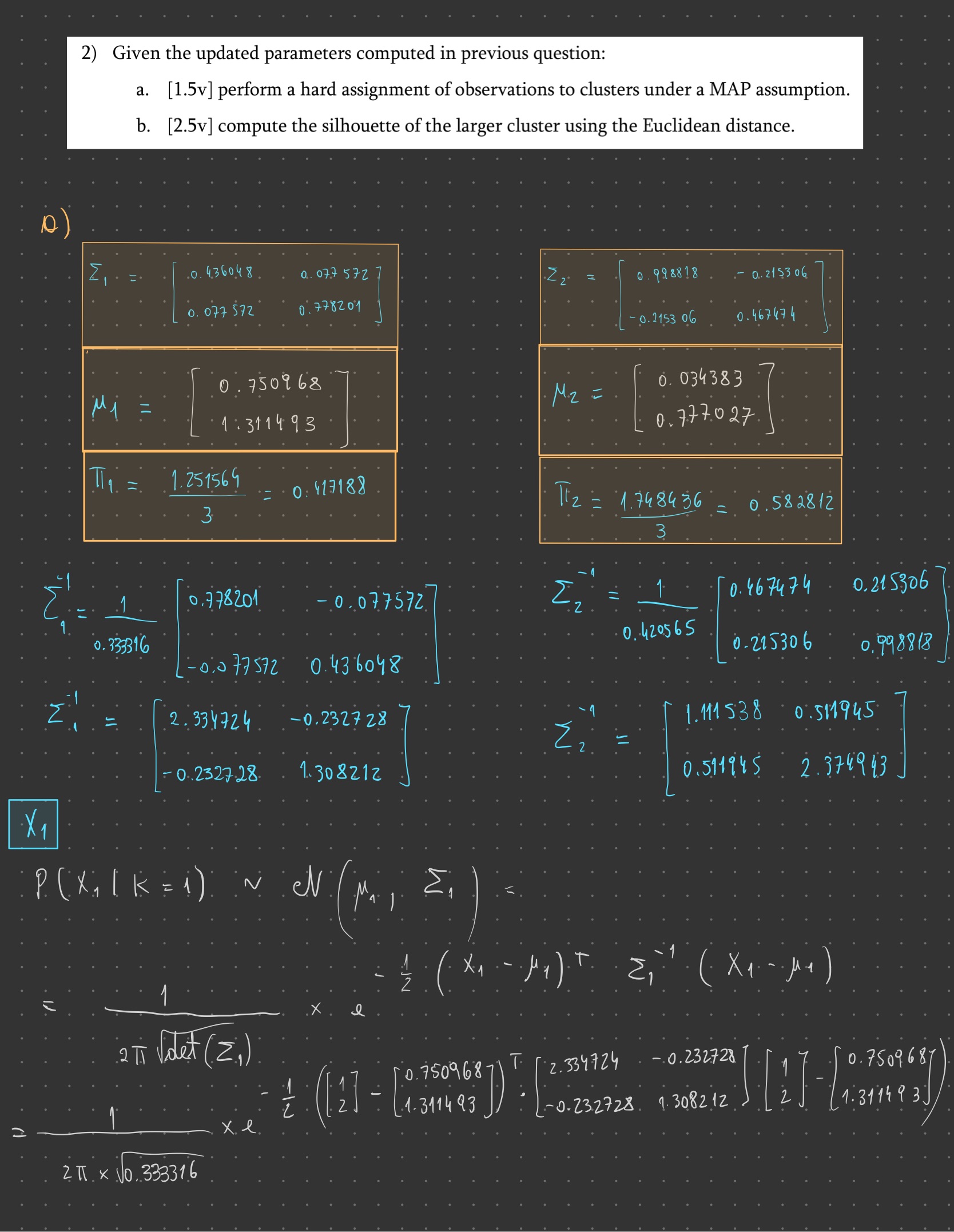
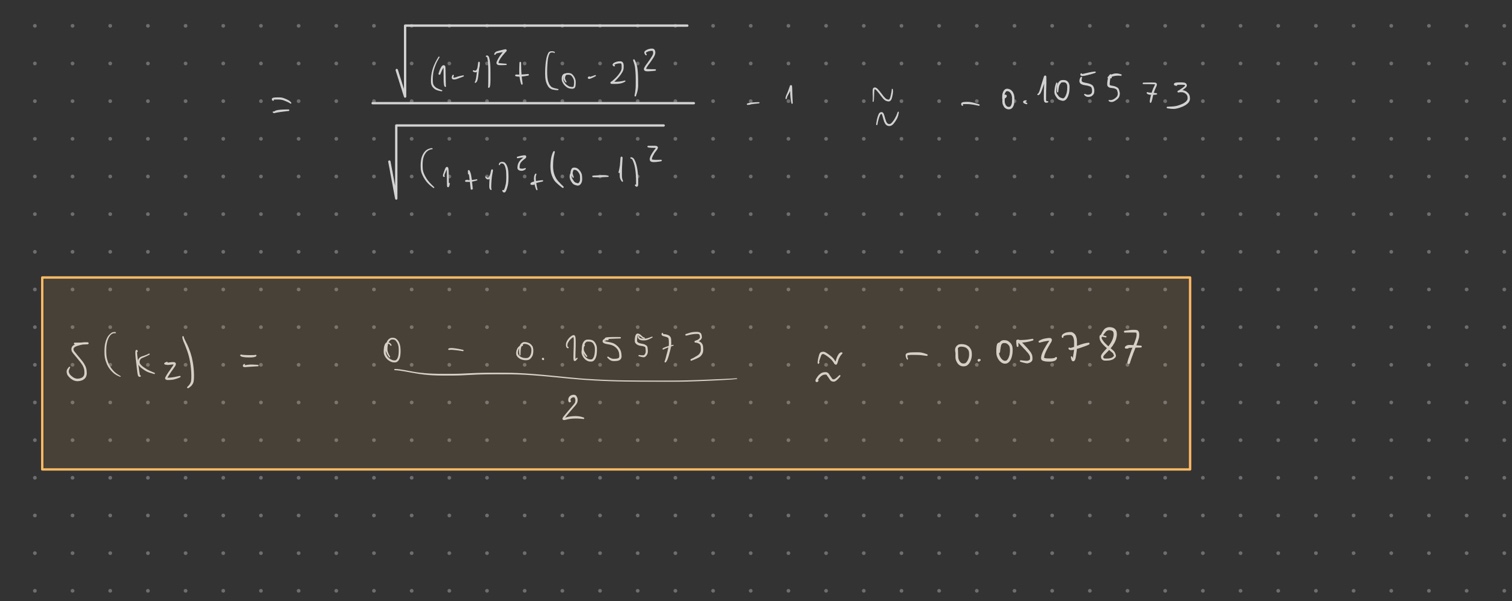
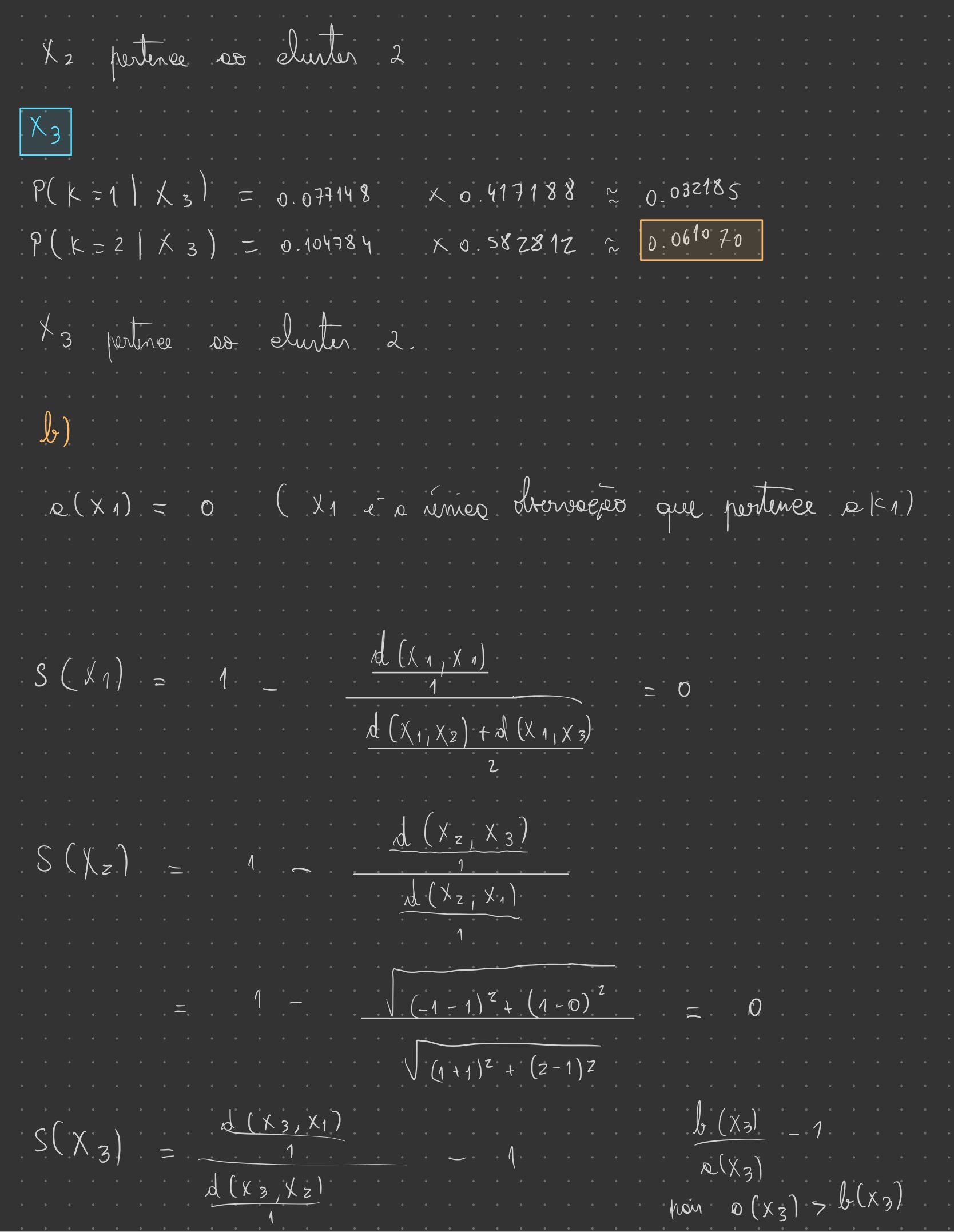
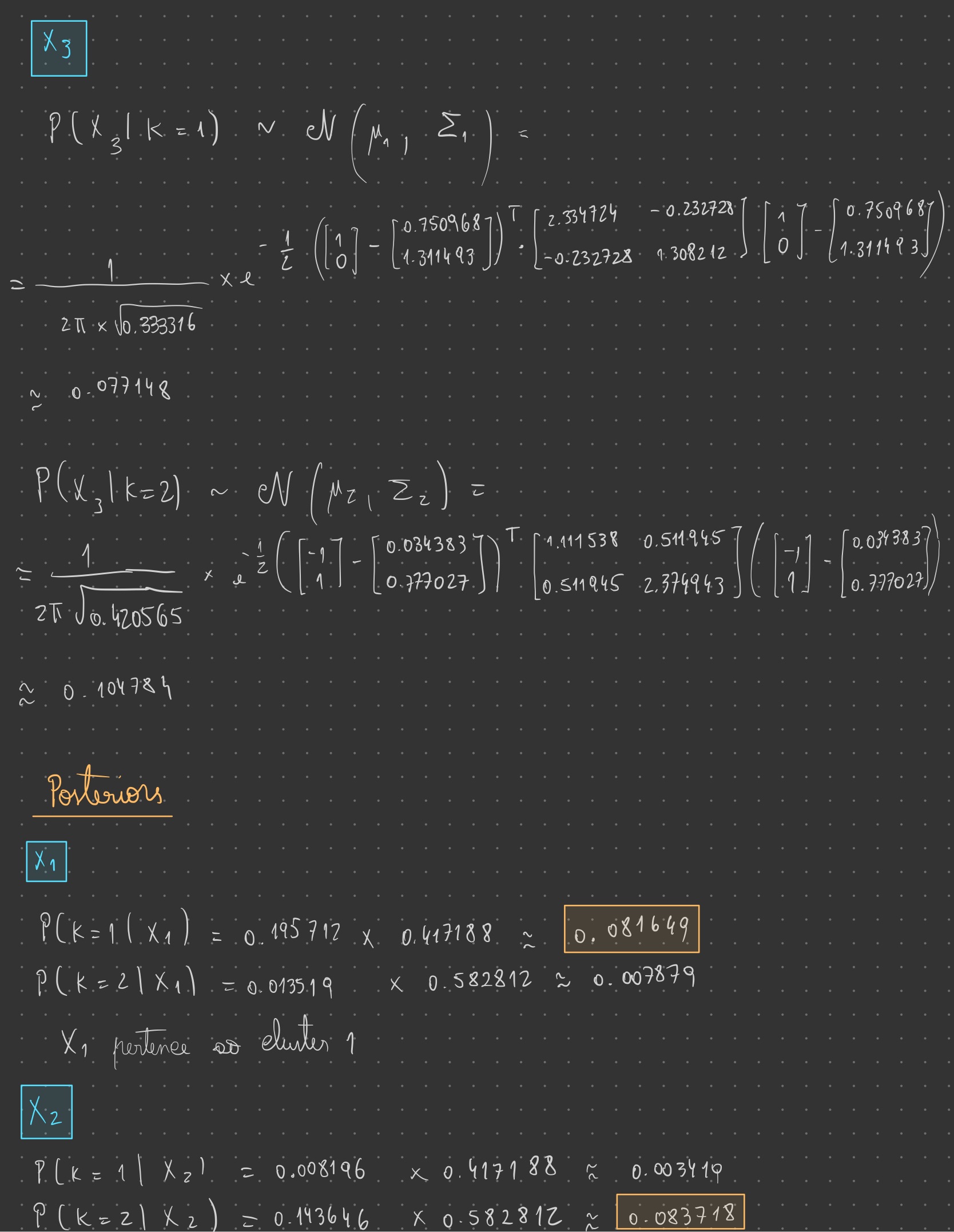
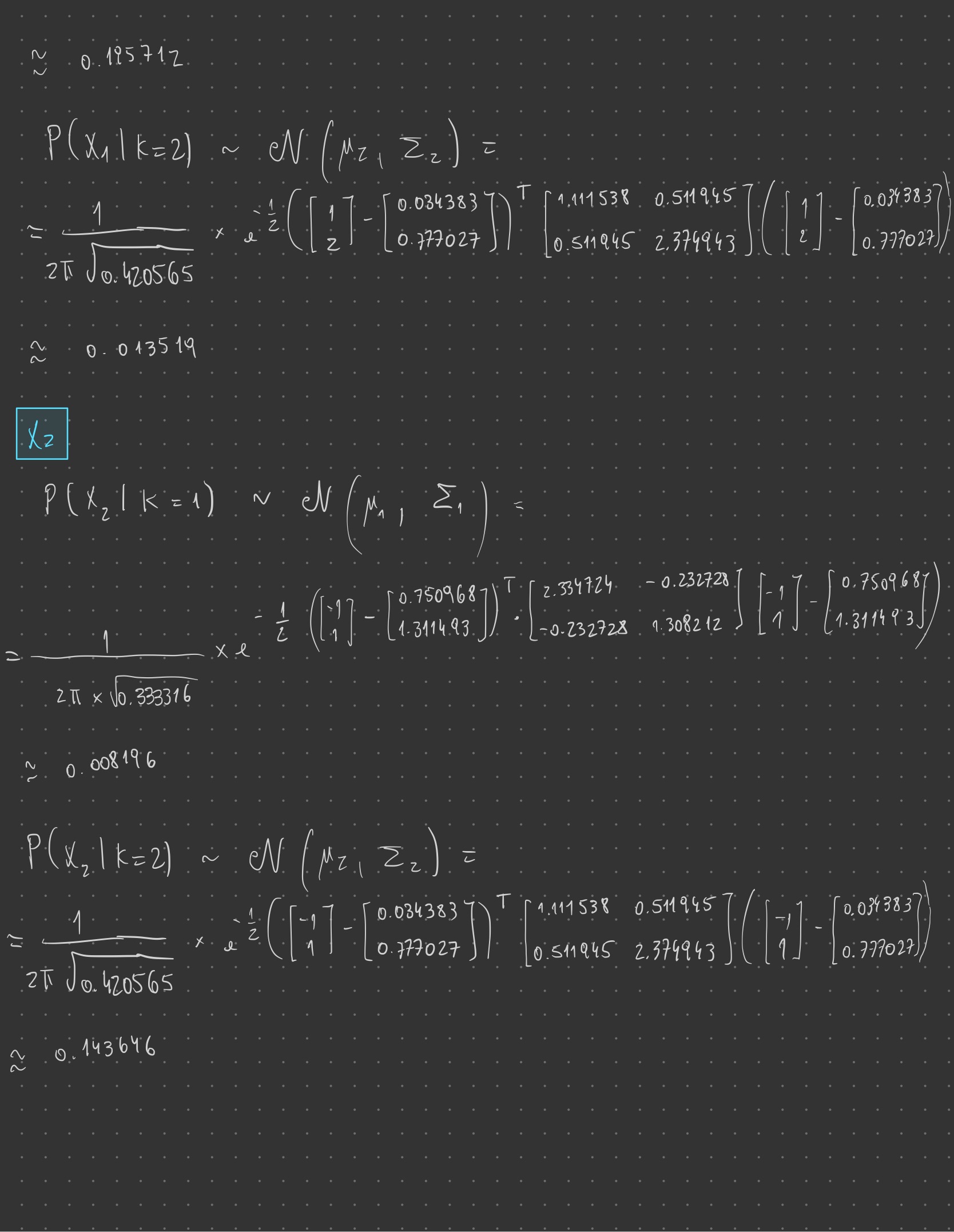
**I. Pen-and-paper**

**1) Answer 1**





**2) Answer 2**



**II. Programming and critical analysis**

1. **Answer 1**

**Silhouettes and purities respectively:**

Seed = 0 🡪 0.11362027575179431; 0.7671957671957672

Seed = 1 🡪 0.11403554201377074; 0.7632275132275133

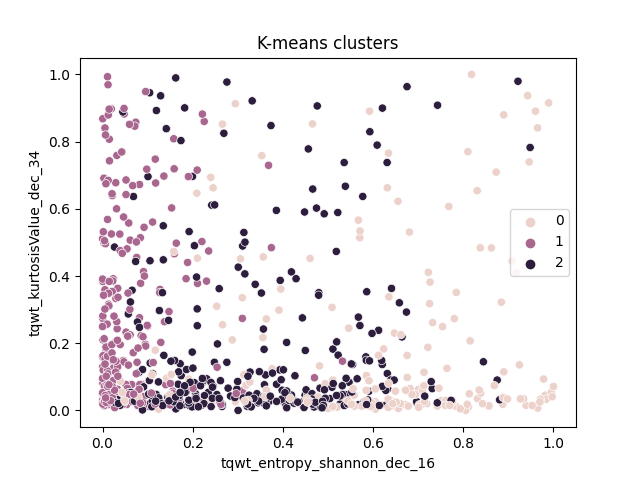
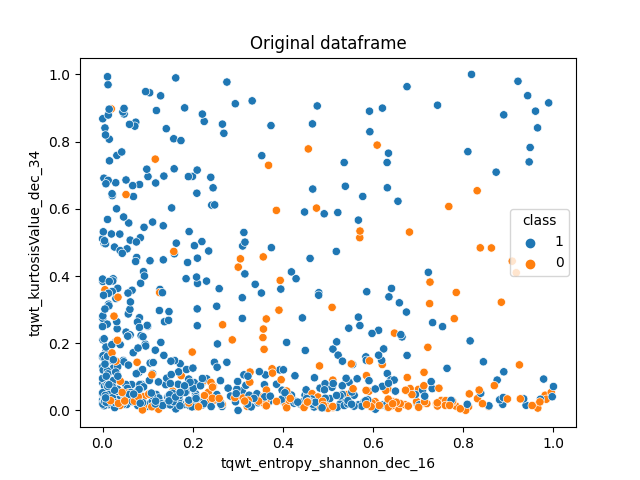
Seed = 2 🡪 0.11362027575179431; 0.7671957671957672



1. **Answer 2**

What is causing the non-determinism is the random initialization of the centroids of the clusters. For different seeds, the algorithm converges to different local minimums. Therefore, the obtained results are different, given enough decimal places (for seed = 0 and seed = 2 we obtain approximately the same result for 16 decimal places).

1. **Answer 3**



1. **Answer 4**

The number of principal components required to explain more than 80% of variability is 31.



**III. APPENDIX**

**import** pandas **as** pd

**from** scipy.io.arff **import** loadarff

**from** sklearn.model\_selection **import** train\_test\_split

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn **import** metrics, datasets

**from** sklearn.preprocessing **import** MinMaxScaler

**from** sklearn.cluster **import** KMeans

**from** sklearn **import** cluster

**import** numpy **as** np

**from** sklearn.feature\_selection **import** VarianceThreshold

**from** sklearn.decomposition **import** PCA

data **=** loadarff('pd\_speech.arff')

df **=** pd.DataFrame(data[**0**])

df['class'] **=** df['class'].str.decode('utf-8')

X **=** df.drop('class', **axis=1**)

scaler **=** MinMaxScaler()

scaledDF **=** scaler.fit\_transform(df)

scaledDF **=** pd.DataFrame(scaledDF, **columns=**df.columns)

**print**("SHAPE:", scaledDF.shape)

scaledDF **=** scaledDF.drop(**columns=**['class'])

labels **=** []

silhouettes **=** []

purity **=** []

**def** **purity\_score**(**y\_true**, **y\_pred**):

confusion\_matrix **=** metrics.cluster.contingency\_matrix(y\_true, y\_pred)

**return** np.sum(np.amax(confusion\_matrix, **axis=0**)) **/** np.sum(confusion\_matrix)

**for** i **in** **range**(**3**):

kmeans **=** KMeans(**n\_clusters** **=** **3**, **random\_state** **=** i).fit(scaledDF)

y\_pred **=** kmeans.labels\_

labels.append(y\_pred)

silhouettes.append(metrics.silhouette\_score(scaledDF, y\_pred, **metric=**'euclidean'))

purity.append(purity\_score(df["class"], y\_pred))

**print**("Silhouette scores: ", silhouettes)

**print**("Purity scores: ", purity)

variances **=** scaledDF.var()

variances **=** variances.sort\_values(**ascending=**False)

variances **=** variances[:**2**]

sns.scatterplot(**x** **=** scaledDF[variances.index[**0**]], **y** **=** scaledDF[variances.index[**1**]], **hue** **=** df["class"])

plt.title("Original dataframe")

plt.savefig("OG\_DF.png")

plt.show()

sns.scatterplot(**x** **=** scaledDF[variances.index[**0**]], **y** **=** scaledDF[variances.index[**1**]], **hue** **=** labels[**0**])

plt.title("K-means clusters")

plt.savefig("kmeans.png")

plt.show()

*#print(variances)*

*#print(len(df\_scaled["tqwt\_kurtosisValue\_dec\_34"]))*

*#print(len(df\_scaled["tqwt\_entropy\_shannon\_dec\_16"]))*

pca **=** PCA(**n\_components=0.8**)

pca.fit(scaledDF)

**print**("Number of principal components:", pca.n\_components\_)

**END**