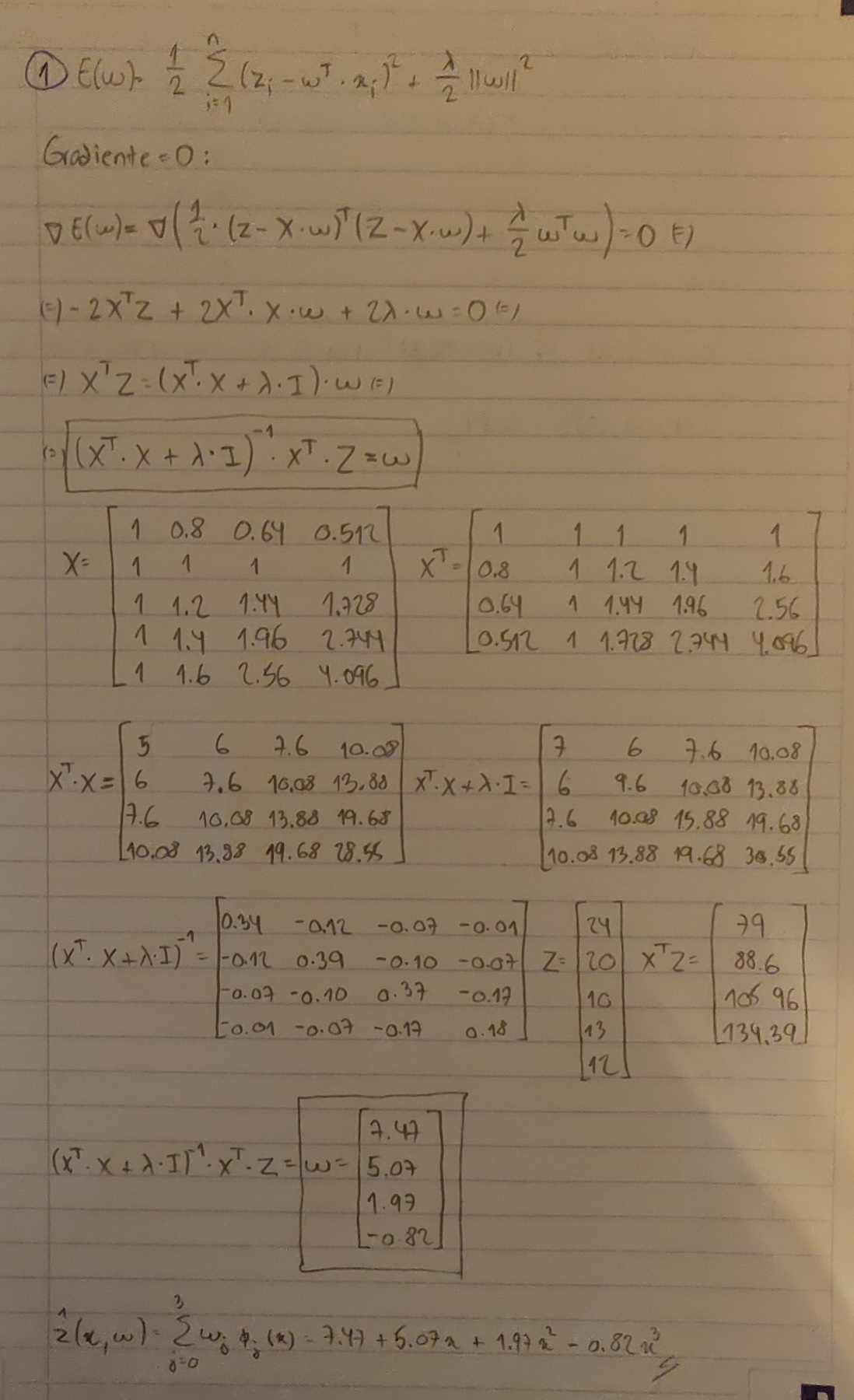
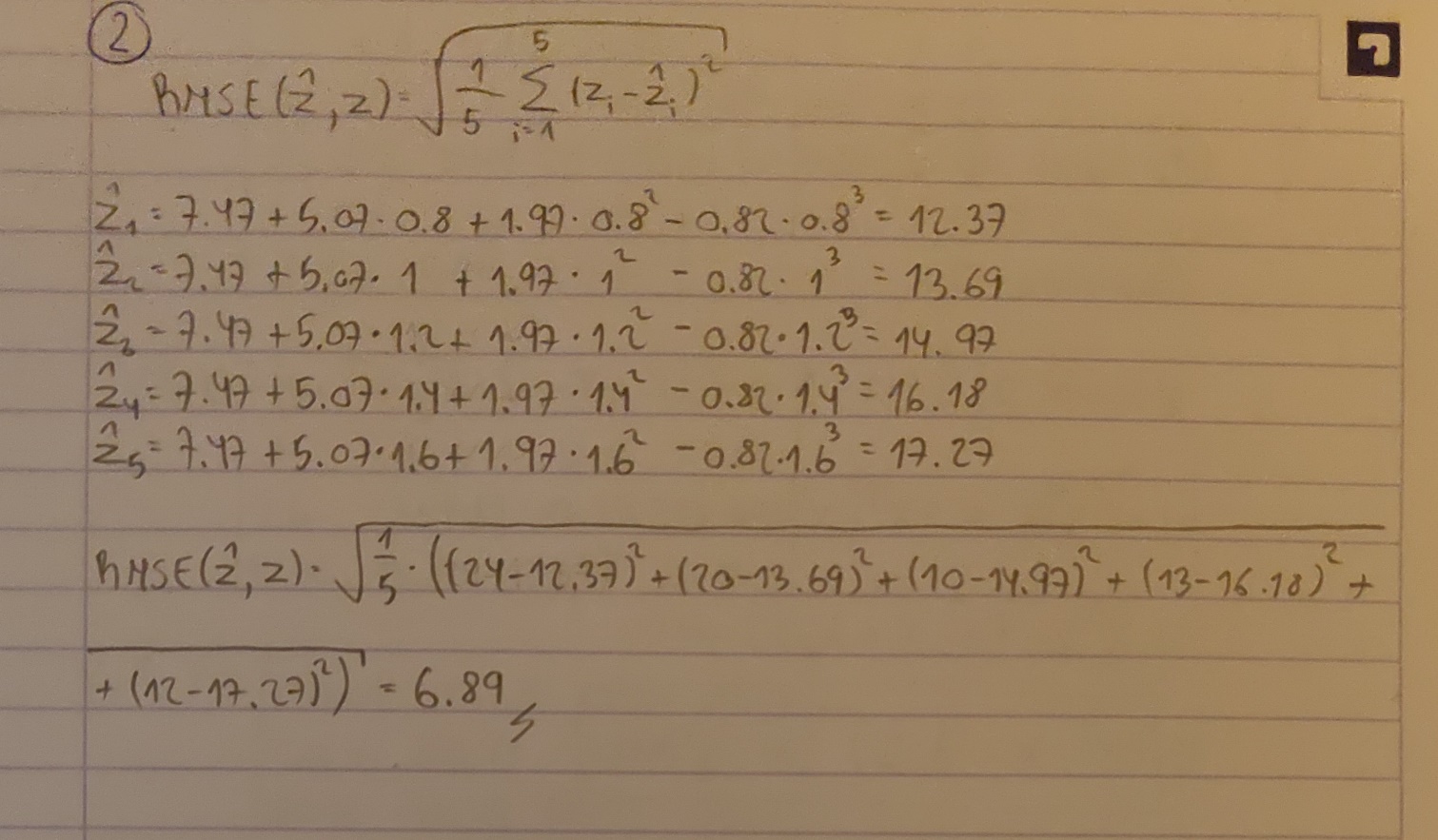
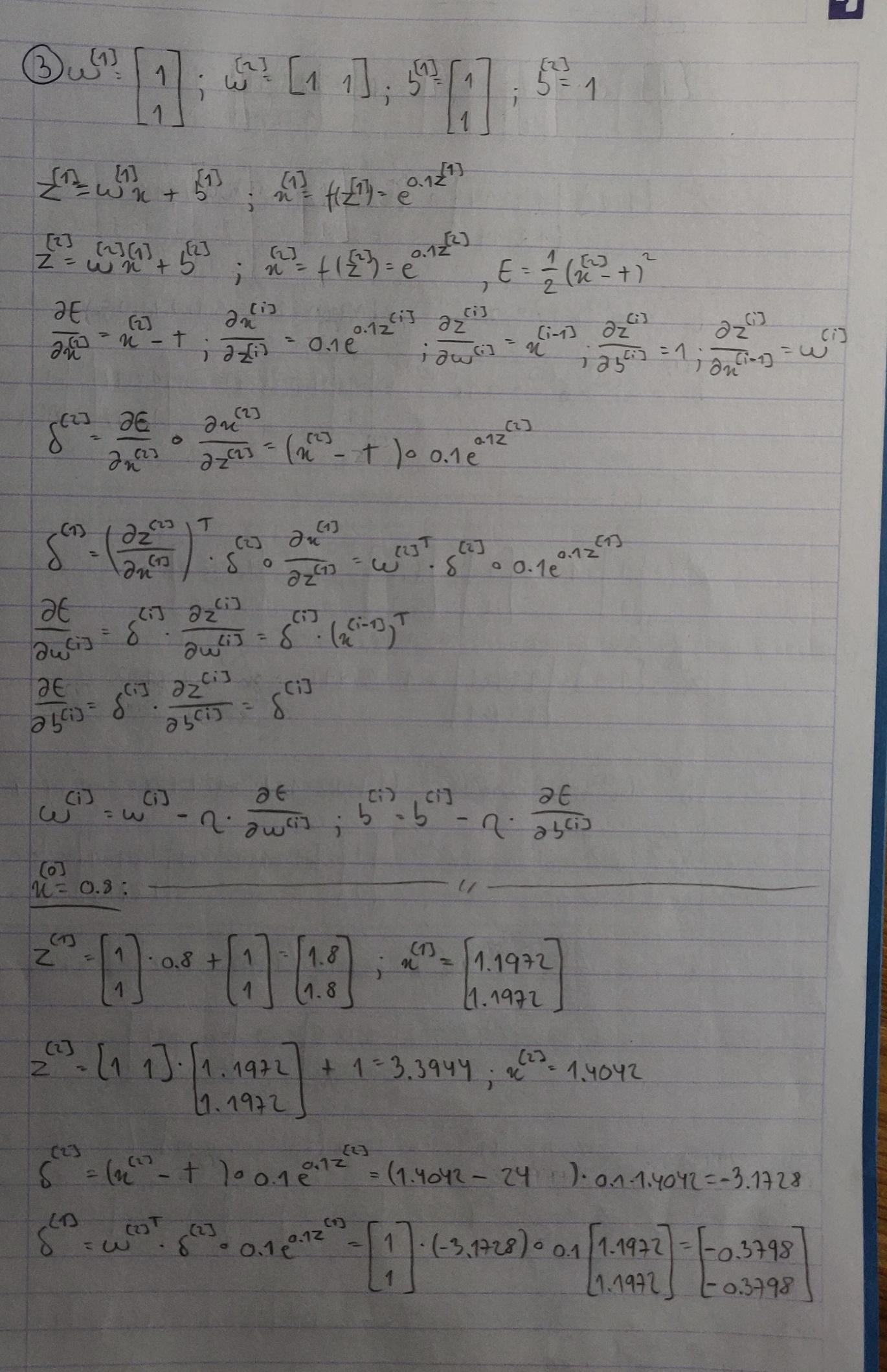
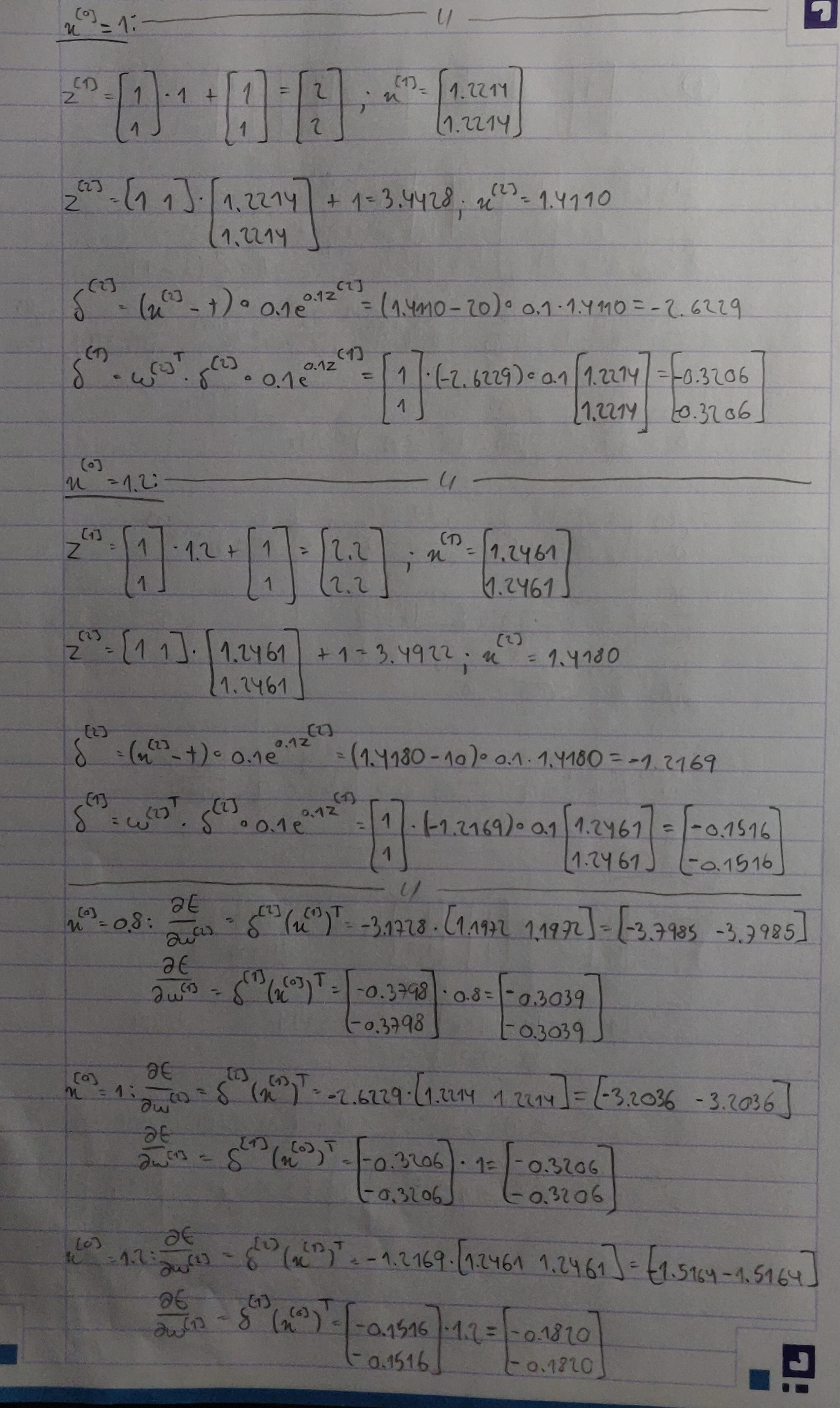
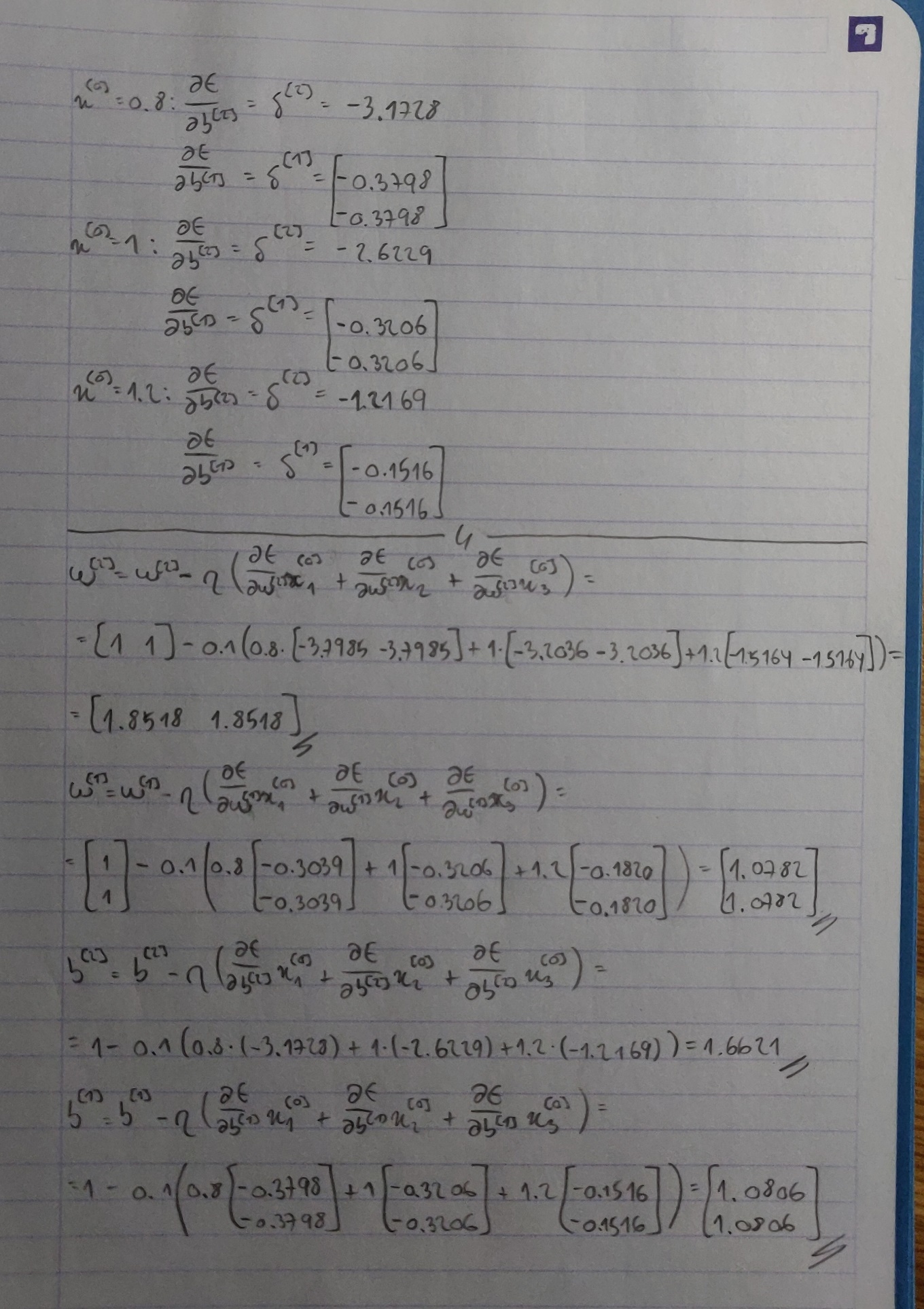
**I. Pen-and-paper**









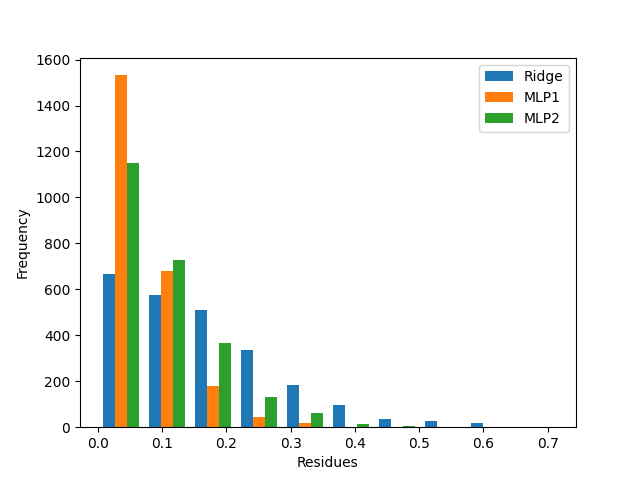
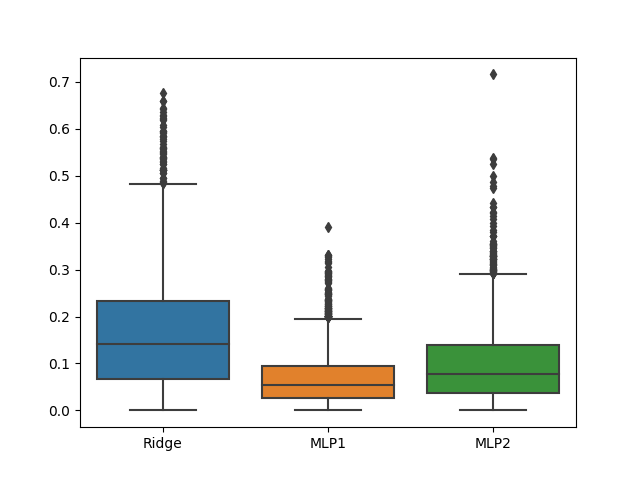


**II. Programming and critical analysis**

MAE for Ridge Regression: 0.163;

MAE for MLP with early stopping: 0.068;

MAE for MLP without early stopping: 0.098.



MLP1 iterations: 452;

MLP2 iterations: 77.

O MLP1 é parametrizado com early stopping, enquanto o MLP2 não. O early stopping tem como função validar o modelo, pondo de parte 10% do training data e terminando a execução quando o resultado de validação for inferior a um determinado limiar pré-definido. Logo podemos concluir que o facto do MLP1 ser parametrizado com early stopping faz com que este execute menos iterações.

**III. APPENDIX**

#######         Importing required libraries          #######

from scipy.io.arff import loadarff

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.linear\_model import Ridge

from sklearn.neural\_network import MLPRegressor

from sklearn import metrics

import warnings

def warn(\*args, \*\*kwargs): pass

warnings.warn = warn

#######            Reading the ARFF file              #######

data = loadarff('kin8nm.arff')

df = pd.DataFrame(data[0])

#######      Creating the training-testing split      #######

X, y = df.drop('y', axis=1), df['y']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.7, random\_state=0)

#######     Creating and asserting the regressors     #######

# Creating the regressors

ridge = Ridge(alpha=0.1)

mlp1 = MLPRegressor(hidden\_layer\_sizes=(10,10), max\_iter=500, early\_stopping=True, random\_state=0, activation='tanh')

mlp2 = MLPRegressor(hidden\_layer\_sizes=(10,10), max\_iter=500, early\_stopping=False, random\_state=0, activation='tanh')

# Training the regressors

ridge.fit(X\_train, y\_train)

mlp1.fit(X\_train, y\_train)

mlp2.fit(X\_train, y\_train)

# Predicting the values

ridge\_pred = ridge.predict(X\_test)

mlp1\_pred = mlp1.predict(X\_test)

mlp2\_pred = mlp2.predict(X\_test)

#######     Calculating the MAE for each regressor    #######

ridge\_mae = metrics.mean\_absolute\_error(y\_test, ridge\_pred)

mlp1\_mae = metrics.mean\_absolute\_error(y\_test, mlp1\_pred)

mlp2\_mae = metrics.mean\_absolute\_error(y\_test, mlp2\_pred)

#######                      Ex 4                     #######

print("MAE for Ridge Regression: ", round(ridge\_mae, 3))

print("MAE for MLP with early stopping: ", round(mlp1\_mae, 3))

print("MAE for MLP without early stopping: ", round(mlp2\_mae, 3))

#######                      Ex 5                     #######

residues = [abs(y\_test - ridge\_pred), abs(y\_test - mlp1\_pred), abs(y\_test - mlp2\_pred)]

# Boxplot

sns.boxplot(residues)

plt.xticks([0, 1, 2], ['Ridge', 'MLP1', 'MLP2'])

plt.show()

# Histogram

plt.hist(residues, bins=10, label=['Ridge', 'MLP1', 'MLP2'])

plt.legend()

plt.xlabel('Residues')

plt.ylabel('Frequency')

plt.show()

#######                      Ex 6                     #######

print("MLP1 iterations: ", mlp1.n\_iter\_)

print("MLP2 iterations: ", mlp2.n\_iter\_)

**END**