1. **Linear innovation sequence.** This comes from the example on lecture notes 4-21. Let the observation sequence be $Y_i = X + Z_i$ for $i = 1, 2, \ldots, n$, where $X, Z_1, \ldots, Z_n$ are zero mean, uncorrelated r.v.s with $E(X^2) = P$ and $E(Z_i^2) = N$ for $i = 1, 2, \ldots, n$. Find the linear innovation sequence of $Y$.

2. **An innovations sequence and its applications.** Let $[Y_1 \ Y_2 \ Y_3]^\top$ be a zero-mean random vector with covariance matrix

\[
\begin{bmatrix}
1 & 0.5 & 0.5 & 0 \\
0.5 & 1 & 0.5 & 0.25 \\
0.5 & 0.5 & 1 & 0.25 \\
0 & 0.25 & 0.25 & 1
\end{bmatrix}.
\]

a. Let $\tilde{Y} = [\tilde{Y}_1 \ \tilde{Y}_2 \ \tilde{Y}_3]^\top$ be the innovations sequence of $Y = [Y_1 \ Y_2 \ Y_3]^\top$. Find the matrix $A$ such that

$$\tilde{Y} = AY.$$ 

b. Find the covariance matrix of $\tilde{Y}$ and the cross-covariance matrix of $X$ and $\tilde{Y}$.

c. Find the constants $a, b, c$ that minimize $E[(X - a\tilde{Y}_1 - b\tilde{Y}_2 - c\tilde{Y}_3)^2]$.

3. **Cellphone.** We aim to design a cellphone which is able to denoise a signal modeled as $Y_1 = V + Z$, where $V$ is a random variable representing the user’s voice and $Z$ is a random variable representing background noise. An extra microphone measures the background. However this measurement also includes some distorted voice signal. This is taken into account by modeling it as $Y_2 = Z + U$, where $U$ is a random variable representing the distortion. Assume that $V, Z$ and $U$ are all zero mean, both $(V, Z)$ and $(U, Z)$ are uncorrelated and $\text{Corr}(U, V) = \rho$. We also know that $\text{Var}(V) = P, \text{Var}(Z) = N$ and $\text{Var}(U) = Q$. We decide to obtain a linear estimate of $V$ from $Y_1$ and $Y_2$.

a. What is the innovation sequence $\tilde{Y}_1$ and $\tilde{Y}_2$ corresponding to $Y_1$ and $Y_2$?

b. What is the linear MMSE estimate of $V$ given the measurements expressed as a function of $\tilde{Y}_1, \tilde{Y}_2, P, Q$ and $N$?

c. What is the corresponding MSE in terms of $P, Q$ and $N$?

4. **Vector Kalman filter experiment.** Consider the state space model

$$X_{i+1} = \begin{bmatrix} 1 & 1 \\ 0 & 0.9 \end{bmatrix} X_i + U_i, \quad \text{for} \ i = 1, \ldots, n.$$
The first and second component of $X_i$ are the one-dimensional position and velocity of a moving object. In each step, the position and velocity evolve according to Newton’s laws of physics. Due to friction, the velocity is dampened by a factor of 0.9 in each time step. The state is disturbed by independent noise vectors $U_i$, distributed according to

$$U_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 10 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right), \quad \text{for } i = 1, \ldots, n.$$

The initial state is independent of the noise vectors $U_i$ and distributed according to

$$X_1 \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1000 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\right).$$

The observations are

$$Y_i = X_i + V_i, \quad \text{for } i = 1, \ldots, n + 1,$$

where the noise $V_i$ is distributed according to

$$V_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right), \quad \text{for } i = 1, \ldots, n + 1,$$

independent of the initial state $X_1$ and the state noise vectors $U_i$.

Download the posted MATLAB file `vectorKalman.m` and complete the code to compute the Kalman prediction filter $\hat{X}_{i+1|i}$ from the observations $Y^i$, for $i = 0, \ldots, n+1$. For a single realization with time horizon $n = 100$, plot the true state and its prediction over the time index $i$. In a separate figure, plot the prediction error over $i$. Hand in your MATLAB code and the plots.
5. **The filtering version of the Kalman filter**  We have studied the derivation of scalar Kalman filter to predict the next state in class. Now we are interested in estimating the current state. Derive the update equations and the error:

\[
\hat{X}_{i+1|i+1} = a_i(1 - k_i)\hat{X}_{i|i} + k_i Y_{i+1}
\]

where \(k_i = \frac{a_i^2 \sigma^2_{i|i} + Q_i}{a_i^2 \sigma^2_{i|i} + Q_i + N_{i+1}}\)

and \(\sigma^2_{i+1|i+1} = (1 - k_i)(a_i^2 \sigma^2_{i|i} + Q_i)\).

6. **Neural network estimator.** In this exercise, we explore a neural network’s ability to learn statistical estimators in communication problems. Specifically, we consider the point to point channel model. When symbol \(X\) is transmitted, a distorted symbol \(Y\) is received according to conditional probability \(P_{Y|X}\) which characterizes the channel.

As discussed in class, MMSE is a very popularly used method in estimating \(X\) from \(Y\) that would minimize the mean square error between the estimate, \(\hat{X}\) and \(X\). It is well-known that the best MMSE estimate is \(\hat{X}_{MMSE} = \mathbb{E}[X|Y]\), the conditional expectation of \(X\) given \(Y\). However, this is usually hard to compute. In many cases the linear MMSE estimator is used. In the following questions, you will implement the neural network estimators for different types of signal and noise models and compare some of them with linear MMSE estimators. The starter code for this problem is given in `neural_estimators.py`.

![Figure 2: Communication channel.](image)

a. First we consider a simple case where the channel noise is additive white Gaussian noise (AWGN). The input is a Gaussian random variable \(X\), with mean 0 and variance \(P\). The output of the channel is \(Y = X + Z\), where \(Z \sim \mathcal{N}(0, N)\) is an independent Gaussian distribution and \(N\) is the noise power.

i. Find the MMSE linear estimate for \(X\).

ii. Write the function `generate_data_AWGN` to generate samples for \(X\) and \(Y\).

iii. Train a neural network estimator for \(P = 10, N = 1\), which takes input \(Y\) and outputs an estimated \(\hat{X}\). Tune the parameters, including but not limited to depth, number of neurons, non-linearity, initialization and so on. What is a reasonable set of parameters you get?

iv. Compare the neural network with the linear estimator on a separate test dataset of 1000 samples. Report the mean-square error (MSE) for both estimators.
v. On the same plot using the test dataset, plot a scatter plot of the test data $(X,Y)$, a plot of the MMSE linear estimate computed in part i., and a scatter plot of the estimate from the neural network $(\hat{X},Y)$. Comment on the results.

vi. Keeping other parameters the same, now train your neural network estimator using different depths (you should need no more than 5 layers). Plot the MSE against the number of layers. How many layers do you need to achieve the lowest MSE? How does it compare with the MSE from the MMSE linear estimate? Note: Use `tensorflow.reset_default_graph()` before you create a new model if you have problems with reusing the variables.

b. Now suppose the input $X$ is a Laplacian random variable with mean 0 and rate $\lambda_X$. The output of the channel is $Y = X + Z$, where $Z \sim \text{Laplace}(\lambda_Z)$ independent of $X$.

i. Find the MMSE linear estimate for $X$.

ii. Write the function `generate_data_laplace` to generate samples for $X$ and $Y$. Note that the function `numpy.random.laplace` uses $\beta = \frac{1}{\lambda}$ as its scale parameter.

iii. Train a neural network estimator for $\lambda_X = 0.5, \lambda_Z = 1$, which takes input $Y$ and outputs an estimated $\hat{X}$. Tune the parameters, including but not limited to depth, number of neurons, non-linearity, initialization and so on. What is a reasonable set of parameters you get?

iv. Compare the neural network with the linear estimator on a separate test dataset of 1000 samples. Report the mean-square error (MSE) for both estimators.

v. On the same plot using the test dataset, plot a scatter plot of the test data $(X,Y)$, a plot of the MMSE linear estimate computed in part i., and a scatter plot of the estimate from the neural network $(\hat{X},Y)$. Comment on the results.

vi. Keeping other parameters the same, now train your neural network estimator using different depths (you should need no more than 5 layers). Plot the MSE against the number of layers. How many layers do you need to achieve the lowest MSE? How does it compare with the MSE from the MMSE linear estimate?

c. When $Y$ is not a linear function of $X$, even the linear MMSE estimate can be hard to compute. Suppose the input is a uniform random variable $X \sim U[-P,P]$, and the output is a cubic function of $X$ with noise i.e. $Y = X^3 + Z$, where $Z \sim U[-N,N]$ independent of $X$.

i. Find the MMSE linear estimate for $X$. You can express your answers in terms of moments ($E[X^k]$).

ii. Write the function `generate_data_unif_cube` to generate samples for $X$ and $Y$.

iii. Train a neural network estimator for $P = 4, N = 0.5$, which takes input $Y$
and outputs an estimated $\hat{X}$. Tune the parameters, including but not limited to depth, number of neurons, non-linearity, initialization and so on. What is a reasonable set of parameters you get?

iv. On the same plot using a separate test dataset of 1000 samples, plot a scatter plot of the test data $(X,Y)$ and the estimate from the neural network $(\hat{X},Y)$. Comment on the results.

v. (Optional) Compare the neural network with the linear estimator on the test dataset. Report the mean-square error (MSE) for both estimators.

vi. Keeping other parameters the same, now train your neural network estimator using different depths (you should need no more than 15 layers). Plot the MSE against the number of layers. How many layers do you need to achieve the lowest MSE?