# CS4070 - MULTIVARIATE DATA ANALYSIS - PART 2

ASSIGNMENT 1

**Konstantinos Krachtopoulos**

**Student Number: 5472539**

**Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)**

**Technical University Delft**

December 2021

## 1 Exercise 1

In order to find the distribution of $\theta|y$ we calculate its posterior probability from Bayes Rule:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{1}$$

We already know the distributions of $p(y|\theta)$ and $p(\theta)$ from the problem description. Moreover, the marginal distribution $p(y)$ does not depend on $\theta$, and thus can be dropped. Then, we have:

$$\begin{gathered} p(\theta|y) \propto p(y|\theta)p(\theta) = \\ (2\pi\sigma^2)^{-\frac{n}{2}} \exp(-\frac{1}{2}(y - H\theta)^T(\sigma^2 I_n)^{-1}(y - H\theta))\times \\ (2\pi P_0)^{-\frac{n}{2}} \exp(-\frac{1}{2}(y - m_0)^T P_0^{-1}(y - m_0)) \end{gathered} \tag{2}$$

Additionally, all terms that are independent from $\theta$ can be dropped, as they are constants to our problem:

$$\begin{aligned} p(y|\theta)p(\theta) &\propto \exp(-\frac{1}{2}(y - H\theta)^T(\sigma^2 I_n)^{-1}(y - H\theta)) \times \exp(-\frac{1}{2}(y - m_0)^T P_0^{-1}(y - m_0)) \\ &\propto \exp(-\frac{1}{2\sigma^2}(-2y^T H\theta + \theta^T H^T H\theta) - \frac{1}{2}(\theta^T P_0^{-1}\theta - 2m_0^T P_0^{-1}\theta)) \end{aligned} \tag{3}$$

Afterwards, we know that the distribution of $\theta|y$ will be in the form of $N_2 \sim (\nu, C)$, meaning that its respective probability will be in the form of:

$$p(\theta|y) = (2\pi C)^{-\frac{n}{2}} \exp(-\frac{1}{2}(\theta - \nu)^T(C)^{-1}(\theta - \nu)) \tag{4}$$

and by dropping the terms that are independent from $\theta$ as before, we get:

$$\begin{aligned} p(\theta|y) &\propto \exp(-\frac{1}{2}(\theta - \nu)^T(C)^{-1}(\theta - \nu)) \\ &\propto \exp(-\frac{1}{2}(\theta^T\theta C^{-1} - 2\theta\nu^T C^{-1})) \end{aligned} \tag{5}$$

Finally, in order to compute the values of $\nu$ and $C$, we set polynomial equations (3) and (5) to be equal with respect to $\theta$. Hence we have, for the quadratic part:

$$\begin{gathered} \exp(-\frac{1}{2\sigma^2}\theta^T H^T H\theta - \frac{1}{2}\theta^T P_0^{-1}\theta) = \exp(-\frac{1}{2}(\theta^T\theta C^{-1})) \Rightarrow \\ C^{-1} = H^T\sigma^{-2}H + P_0^{-1} \end{gathered} \tag{6}$$

For the linear term:

$$\begin{gathered} \exp(\frac{1}{2\sigma^2}2y^T H\theta + \frac{1}{2}2m_0^T P_0^{-1}\theta) = \exp(\frac{1}{2}2\theta\nu^T C^{-1}) \Rightarrow \\ \frac{2y^T H}{\sigma^2} + 2m_0^T P_0^{-1} = 2\nu^T C^{-1} \Rightarrow \\ \nu^T = C(y^T H\sigma^{-2} + m_0^T P_0^{-1}) \Rightarrow \\ \nu = C(H^T\sigma^{-2}y + P_0^{-1}m_0) \end{gathered} \tag{7}$$

## 2 Exercise 2

The Woodbury matrix identity is

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{8}$$

where, $U, A, C, V$ are comformable matrices, where the inverse of matrix $(A + UCV)$ is computed by doing a correction to the original matrix$A$. We can apply Woodbury's formula by using equation (1) from the exercise description. There, matrix $P$ is updated from the original matrix in of the previous step. Hence, we have:

- $A = P_{k-1}^{-1}$
- $U = H_k^T$
- $C = \sigma^{-2}$
- $V = H_k$
- $A + UCV = P_k^{-1}$

By simple substitution, we get the expected outcome:

$$P_k = P_{k-1} - P_{k-1}H_k^T(H_kP_{k-1}H_k^T + \sigma^2)^{-1}H_kP_{k-1} \tag{9}$$

The Woodbury identity is advantageous over equation (1), since it provides a faster and more convenient way of calculating matrix $P_k$. Namely, in order to derive matrix $P_k$, one needs to calculate the inverse of matrix $P_k$, and then invert it. Inversion is a costly calculation, especially when the dimension $p$ of $P$ is large. On the other side, Woodbury identity only requires the inversion matrix $P_k$ is directly calculated, and hence, no extra inversion is required.

## 3 Exercise 3

Implemented an algorithm with the mentioned requirements. The algorithm can be found in the appendix.

The computed posterior mean and covariance matrices of $\theta$, based on the first 5 iterations are shown below:

$$m_5 = \begin{bmatrix} 0.865153 \\ 0.57819009 \\ 2.19007668 \end{bmatrix}, \quad P_5 = \begin{bmatrix} 31.17980408 & -21.39478959 & -2.02377133 \\ -21.39478959 & 31.56409375 & 3.80653564 \\ -2.02377133 & 3.80653564 & 27.82768465 \end{bmatrix} \tag{10}$$

The computed posterior mean and covariance matrices of $\theta$ for all iterations are shown below:

$$m_{50} = \begin{bmatrix} 1.21949428 \\ 0.17463734 \\ 2.48057594 \end{bmatrix}, \quad P_{50} = \begin{bmatrix} 7.20336779 & -0.534743514e & -0.0622388502 \\ -0.534743514 & 0.0545438385 & 0.00634836272 \\ -0.0622388502 & 0.00634836272 & 3.92230751 \end{bmatrix} \tag{11}$$

One can easily observe that the values in the diagonal are much smaller when all data points have been processed. This happens because the diagonal terms represent the uncertainty in the selection of each parameter. When only a few data points have been processed, the uncertainty is large. Conversely, when all 50 data points have been used for updating the posterior probability, the uncertainty of $\theta$ parameters becomes smaller.

The fitted curve when using all observations together with the observed data is presented in 1.

## 4 Exercise 4

In order to verify the formulas of the prediction step of the Kalman filter, we need to compute the distribution of $\theta_k|y_{1:k-1}$. We notice that this distribution can be obtained as the marginal distribution of $\theta_k, \theta_{k-1}|y_{1:k-1}$:

$$p(\theta_k, \theta_{k-1}|y_{1:k-1}) = p(\theta_k|\theta_{k-1}, y_{1:k-1})p(\theta_{k-1}|y_{1:k-1}) \tag{12}$$

However, we observe that $\theta_k = A\theta_{k-1} + q_{k-1}$ does not depend on $y_{1:k-1}$) given $\theta_{k-1}$, and thus (12) becomes:

$$p(\theta_k, \theta_{k-1}|y_{1:k-1}) = p(\theta_k|\theta_{k-1})p(\theta_{k-1}|y_{1:k-1}) \tag{13}$$

Moreover, we have:

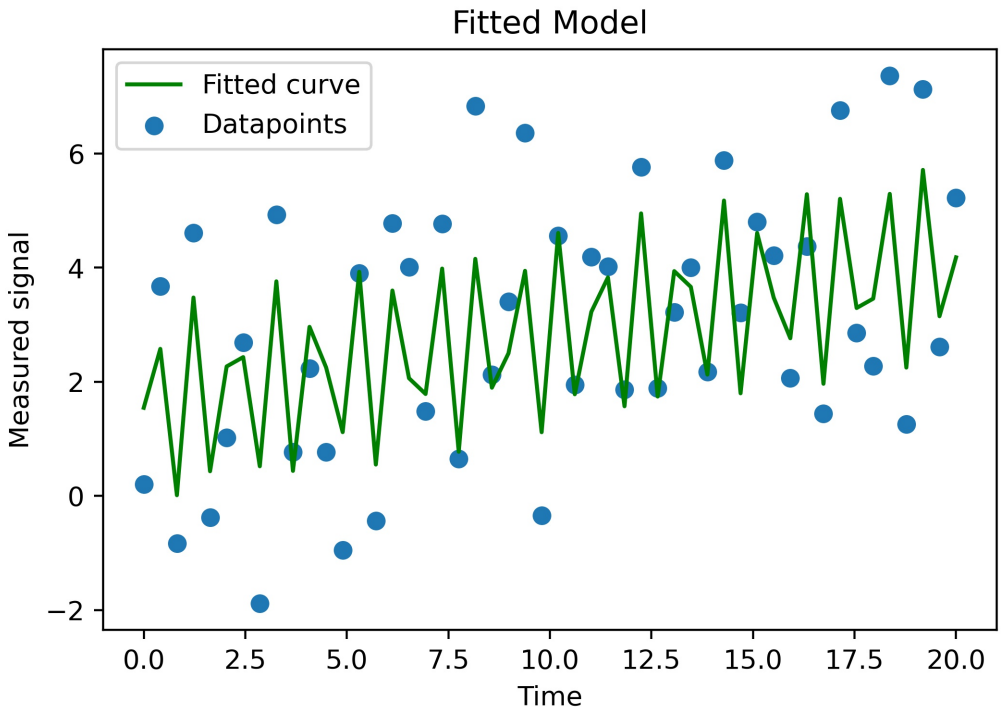


Figure 1: Fitted curve.

- $\theta_{k-1}|y_{1:k-1} \sim N(\nu, C)$
- $\theta_k|\theta_{k-1} \sim N(A\theta_{k-1} + q_{k-1}, Q)$

By using Lemma 1 of the problem description, setting $X = \theta_{k-1}|y_{1:k-1}$, and $\theta_k|\theta_{k-1}$, we get:

$$\begin{bmatrix} \theta_{k-1}|y_{1:k-1} \\ \theta_k|\theta_{k-1} \end{bmatrix} = N\left( \begin{bmatrix} m_{k-1} \\ Am_{k-1} + q_{k-1} \end{bmatrix}, \begin{bmatrix} P_{k-1} & P_{k-1}A^T \\ AP_{k-1} & AP_{k-1}A^T + Q \end{bmatrix} \right) \tag{14}$$

If we marginalize the above distribution over $\theta_{k-1}$, we get the wanted quantity, which is the distribution of component $\theta_k|\theta_{k-1}$:

$$\theta_k|\theta_{k-1} \sim N\left(Am_{k-1}, AP_{k-1}A^T + Q\right) \tag{15}$$

, while the expected value of $q_{k-1}$ is zero.

## 5 Appendix

The code used for computing the quantities of Exercise 3 can be seen below, together with the appropriate comments:

```
# basic imports
import numpy as np
import pandas as pd
from numpy.linalg import inv
import matplotlib.pyplot as plt

# Load the data as a pandas dataframe
df = pd.read_csv("periodic.csv")
```

```
# Load the data as a pandas dataframe
df = pd.read_csv("periodic.csv")

# Initialize the problem parameters
params = 3 # number of parameters
s0_sqr = 100
m = np.zeros((arr.shape[0] + 1, params, 1)) # Create buffer for m matrices with size 51x3x1
p = np.zeros((arr.shape[0] + 1, params, params)) # Create buffer for m matrices with size 51x1x1

# Initialize the apriori matrices
m[0, :, 0] = np.zeros(params) # Set m0
p[0] = s0_sqr*np.eye(params) # Set p0

# Iterate over timepoints:
for idx,t in enumerate(arr[:, 0]):
    # Calculate the h matrix
    h = np.array([[1, t, np.sin(2*np.pi*t)]])
    p[idx+1] = p[idx] - p[idx] @ h.T*inv(h @ p[idx] @ h.T + s0_sqr) @ h @ p[idx]
    m[idx+1] = p[idx+1] @ (h.T / s0_sqr * arr[idx, 1] + inv(p[idx]) @ m[idx])

# print wanted quantities
print(p[5], m[5])
print(p[-1], m[-1])

# plot graph
np.random.seed(15) # ensure that the same random result will always be produced
# Plot the data points, together with the fitted curve from all data
# Get the thetas from the fitted distribution
thetas = np.random.multivariate_normal(m[-1, :, 0], p[-1])
# make the scatter plot of the data points
x = arr[:, 0]
y = arr[:, 1]
plt.scatter(x, y, label="Datapoints")
plt.plot(x, thetas[0] + thetas[1]*x + thetas[2]*np.sin(2*np.pi*x) + np.random.normal(0, 1),
         label="Fitted curve", color="green")
# plot formatting
plt.xlabel("Time")
plt.ylabel("Measured signal")
plt.title("Fitted Model")
plt.legend()
```