KALMAN FILTERS

THE OPTIMALS
EIE 516 PROJECT

Table of Contents

INTRODUCTION

What Is The Kalman Filter?

Brief History

WHAT IS THE KALMAN FILTER?

Kalman filtering is an algorithm that provides estimates of some unknown variables by using a series of measurements observed over time.

The estimates of the unknown variables using the series of observed measurements tend to be more accurate than those based on a single measurement alone.

The Kalman filter provides optimal estimates of variables of interests when they cannot be measured directly using linear models with additive Gaussian noises.

BRIEF HISTORY

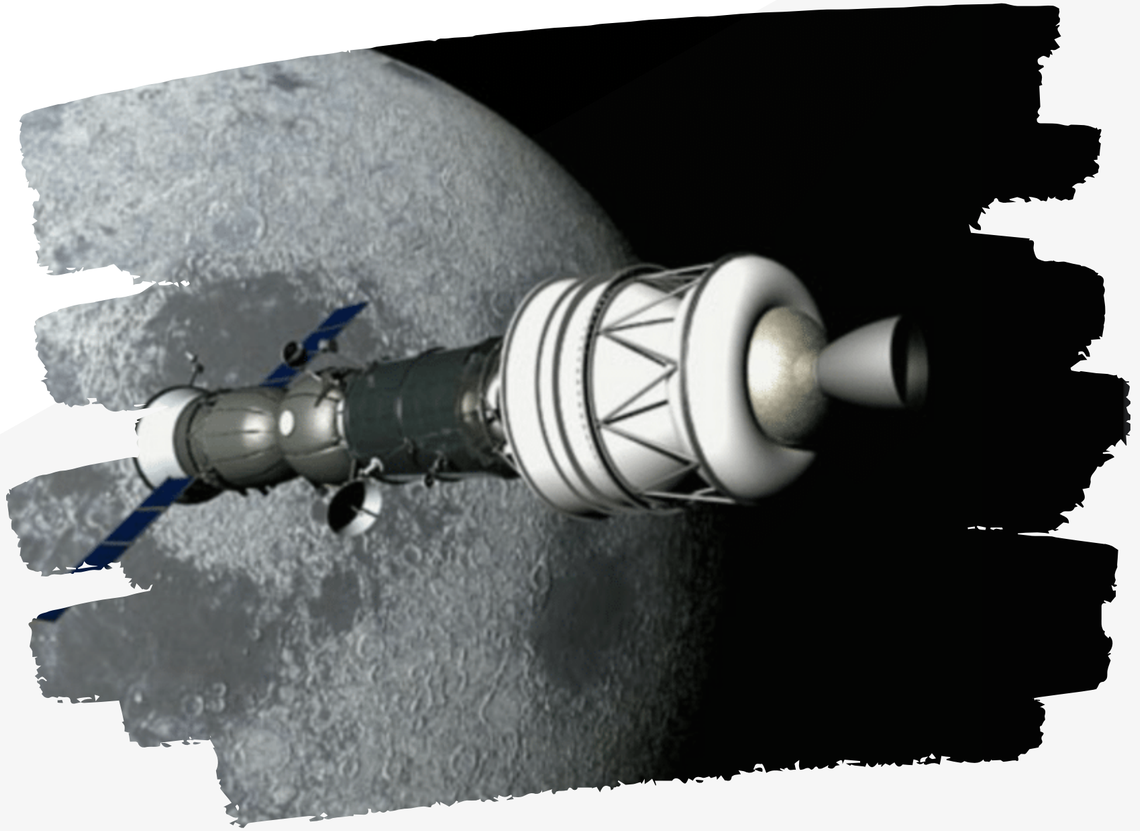
The Kalman filter was pioneered by Rudolf Emil Kalman in 1959, originally designed and developed to solve the navigation problem in the Apollo Project.

The filter was named after him.
In 1959, Kálmán published his famous paper describing a recursive solution to the discrete- data linear filtering problem.

BRIEF HISTORY
(AFTER KALMAN)

Dr Schmidt and other researchers who were working on navigation and guidance for the circumlunar expedition were exposed to its known publication by Dr Kalman in 1960.

The proposed filter was modified and employed as a remedy for the problems faced during the Apollo project.



TYPES OF KALMAN FILTERS

Unscented Kalman Filter

Linear Kalman Filter

Extended Kalman Filter

LINEAR KALMAN FILTER (LKF)

The linear Kalman filter is simply the most basic form of the Kalman filter also known as linear quadratic estimation (LQE).

It is an algorithm that uses a series of measurements observed over time, including statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe.

APPLICATION OF THE LKF

- A common application is for guidance, navigation, and control of vehicles, particularly aircraft, spacecraft, and ships positioned dynamically.

- Kalman filtering is a concept much applied in time series analysis used for topics such as signal processing and econometrics.

EXTENDED KALMAN FILTER (EKF)

The most real systems that require the use of the Kalman filtering process are non-linear.

The evolved equations of filters that are primarily designed for linear functions are needed. These evolved equations are called the Extended Kalman filter (EKF) and it is the nonlinear arrangement of the Kalman filter.

The Extended Kalman Filter (EKF) has become a standard technique used in several nonlinear estimations and machine learning applications. These include estimating the state of a nonlinear dynamic system, estimating parameters for nonlinear system identification (e.g., learning the weights of a neural network), and dual estimation

APPLICATION OF THE EKF

- Application in dynamic X-ray tomography. With this, the X-ray image is reconstructed through a low-dimensional pool of parameters. This has been shown to optimize the computational speed of the process.

- Extended Kalman Filters have extensively contributed to the optimization of robotic movements, tracking, and localization using GPS and IMU Sensors

UNSCENTED KALMAN FILTER (UKF)

The UKF addresses the approximation issues of the EKF.

The state distribution is again represented by a Gaussian random variable but is now specified using a minimal set of carefully chosen sample points. These sample points completely capture the true mean and covariance of the Gaussian random variable, and when propagated through the true non-linear system, capture the posterior mean and covariance accurately to the 3rd order (Taylor series expansion) for any nonlinearity.

APPLICATION OF THE UKF

- The UKF was originally designed for the state-estimation problem and has been applied in nonlinear control applications requiring full-state feedback

- UKF has largely replaced the EKF in many nonlinear filtering and control applications, including for underwater, ground and air navigation

- The UKF is used to minimize measuring and modelling errors for geometric and thermal errors of machine tools

WORKING OF THE FILTER

Operational Principle

Applications and Uses

Matlab Code

Algorithm and Calculations

OPERATIONAL PRINCIPLE

The Kalman filter consists of two distinct processes, namely, the prediction process(aka Propagation) and the measurement process(aka Update/Correction stage).

Both processes are combined and operated in a recursive manner to achieve an optimal Kalman filtering process.

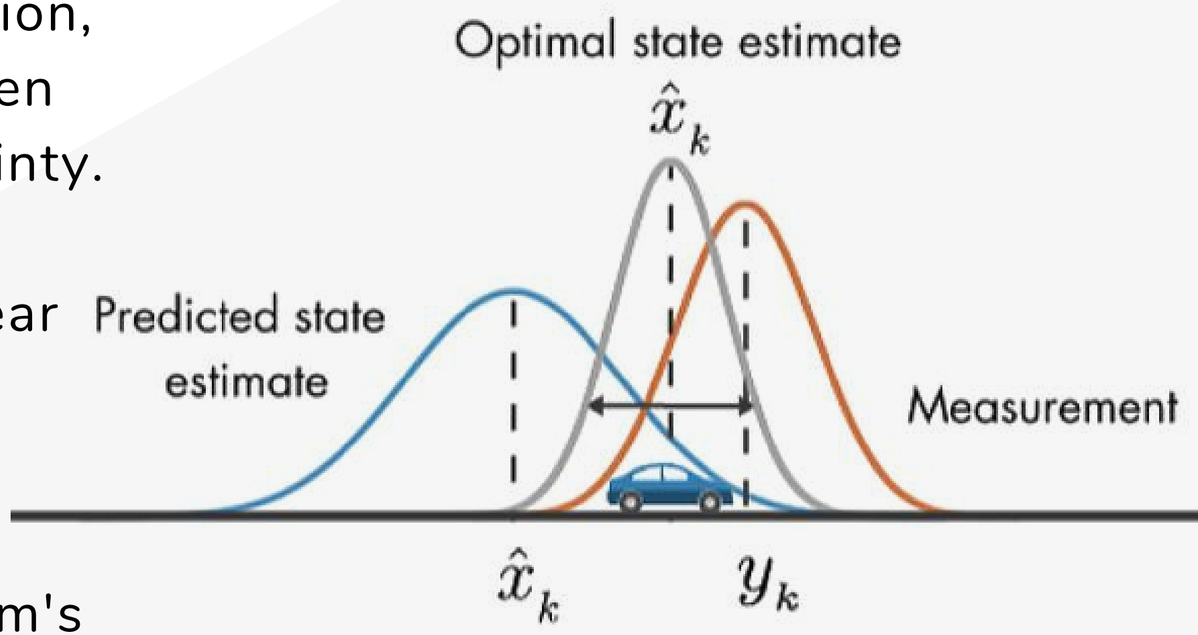
The Kalman filtering algorithm starts from the prediction process by estimating the prediction state based on the derived state space equation (state transition equation).

OPERATIONAL PRINCIPLE

Kalman filter is an iterative mathematical process that uses a set of equations and consecutive data inputs to quickly estimate the true value, position, velocity e.t.c of the object being measured when the measured values contain errors or uncertainty.

They are used to estimate states based on linear dynamical systems in state space format.

The Kalman filter produces an estimate of the state of the system as an average of the system's predicted state and of the new measurement using a weighted average.



KALMAN FILTER ALGORITHM

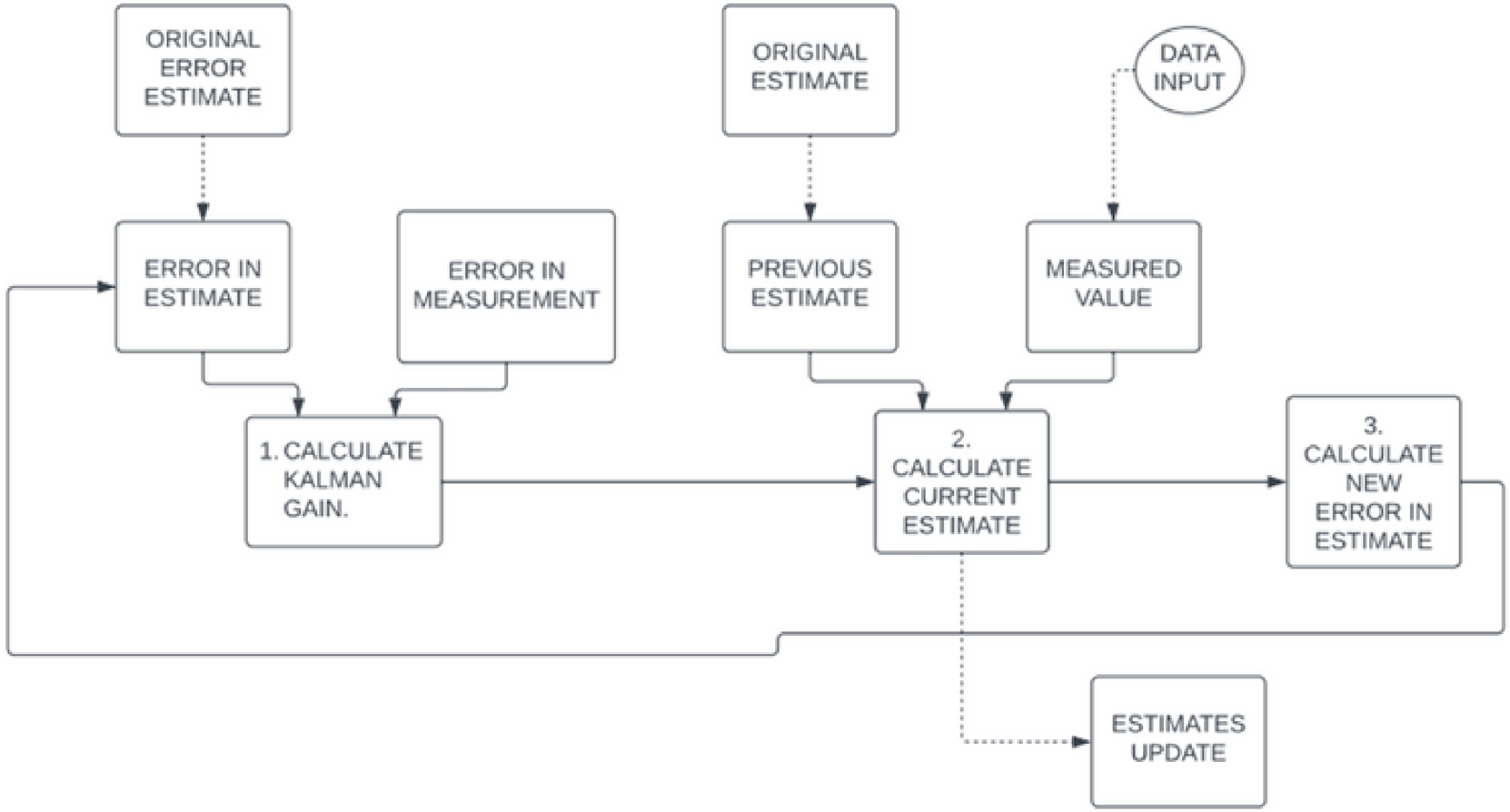
Prediction

- PREDICTED STATE ESTIMATE:
  = F $k-1$ + B k+ WK-1
- PREDICTED ERROR COVARIANCE:
  Pk = $FPk-1$ F T + QK

Correction

- MEASUREMENT RESIDUAL:
  ~ = − H k
- KALMAN GAIN:
  Kk = $PkHT$ (R + HPk HT) −1
- UPDATED STATE ESTIMATE:
  +1 = + Kk (~ - H)
- UPDATED ERROR COVARIANCE:
  Pk+1 = (I − HKk) Pk

BLOCK DIAGRAM DISPLAYING KALMAN FILTER ALGORITHM



EXAMPLES

EXAMPLE 1

Knowing the true temperature of a room is 72o, design a system that can adjust a device that measures the temperature to be 75o with an error measurement of 4o each time it is used. Making an initial estimate of 68O and the estimate error 2o

EXAMPLE 1 SOLUTION

From Question,

True temperature= 72°; Initial Estimate= 68°; Initial Error in estimate= 2°;

initial Measurement = 75°; Error in measurement= 4°

Using Formula:

The Kalman Gain, $K_G = \frac{Error\ in\ Estimate, E_{est}}{Error\ in\ Estimate, E_{est} + Error\ in\ Measurement, E_{mea}}$

Current Estimate, EST_t = Previous Estimate, EST_{t-1} + Kalman Gain, K_G (Measured Value, MEA- Previous Estimate, EST_{t-1})

$Error\ in\ Estimate, E_{est}$ = [1 − Kalman Gain, K_G] * Error in Previous Estimate, E

1ST Iteration:

Kalman Gain, $K_G = \frac{2}{2+4} = 0.33$

Current Estimate, $EST_t = 68 + 0.33(75-68) = 70.33$

$Error\ in\ Estimate, E_{est} = (1-0.33)\ (2) = 1.33$

EXAMPLE 1 SOLUTION CONT'D

Time (t)	Measured (MEA)	Error in Measurement	Kalman Estimates	Error in Estimates	Kalman Gain
t-1			68	2	
t	75	4	70.33	1.33	0.33
t+1	71	4	70.497	0.9982	0.2495
t+2	70	4	70.4678	0.9396	0.0587
t+3	74	4	71.1389	0.761076	0.19
t+4	74	4	71.596676	0.6393	0.16

EXAMPLE 2

Given the data, Noise densities: ~ (0,0.05), ~ (0, 0.1). Assume we are able to determine the vehicle position directly using something like a GPS.
^ ~ ([0, 5]; [0.01 0, 0 1]); Uo=-2 m/s^2
Y1=2.2m ; H= [1 0]

EXAMPLE 2 SOLUTION

Prediction Stage:

$\hat{x}_k = F\hat{x}_{k-1} + Bu_k + w_k$

$P_k = FP_{k-1}F^T + Q_{k-1}$

From question:

$\hat{x}_{k-1} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$; Since it is a vehicle, therefore $F = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$

$B = \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix} = \begin{bmatrix} \frac{1}{2} * 0.5^2 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.5 \end{bmatrix}$; u_0=-2 m/s^2; $P_{k-1} = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$

$F^T = \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix}$; From w_k, Q_{k-1}=0.1= $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$.

Then,

$\hat{x}_k = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 0.125 \\ 0.5 \end{bmatrix}(-2) = \begin{bmatrix} 2.25 \\ 4 \end{bmatrix}$

$P_k = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix} + \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$

Correction Stage:

$K_k = P_k H^T (R + HP_k H^T)^{-1}$

$\hat{x}_{k+1} = \hat{x}_k + K_k(\tilde{y}_k - H\hat{x}_k)$

$P_{k+1} = (I - HK_k)P_k$

$H^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$; From v_k, R_{k-1}=0.05; $\tilde{y}_k$=2.2

Then,

$K_k = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} (0.05 + [1\ 0] \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix})^{-1}$

$= \begin{bmatrix} 0.88 \\ 1.22 \end{bmatrix}$

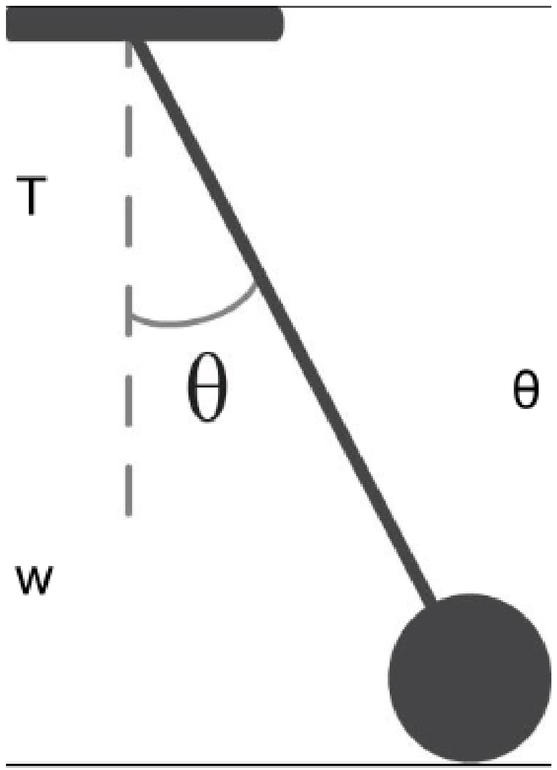
$\hat{x}_{k+1} = \begin{bmatrix} 2.25 \\ 4 \end{bmatrix} + \begin{bmatrix} 0.88 \\ 1.22 \end{bmatrix}(2.2 - [1\ 0] \begin{bmatrix} 2.25 \\ 4 \end{bmatrix}) = \begin{bmatrix} 2.206 \\ 3.939 \end{bmatrix}$

$P_{k+1} = (1 - [1\ 0] \begin{bmatrix} 0.88 \\ 1.22 \end{bmatrix}) \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} = \begin{bmatrix} 0.04 & 0.06 \\ 0.06 & 0.13 \end{bmatrix}$

CODE IMPLEMENTATION OF THE KALMAN FILTER

APPLICATION OF KALMAN FILTER IN A PENDULUM SYSTEM

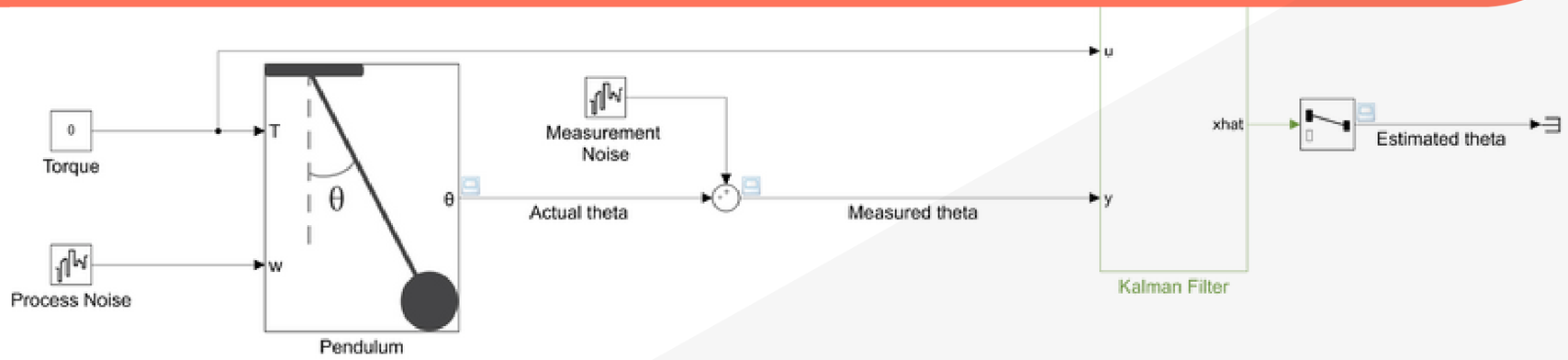
We will see a practical approach on how to use the **Kalman filter in simulink** to estimate the angular position of a pendulum.

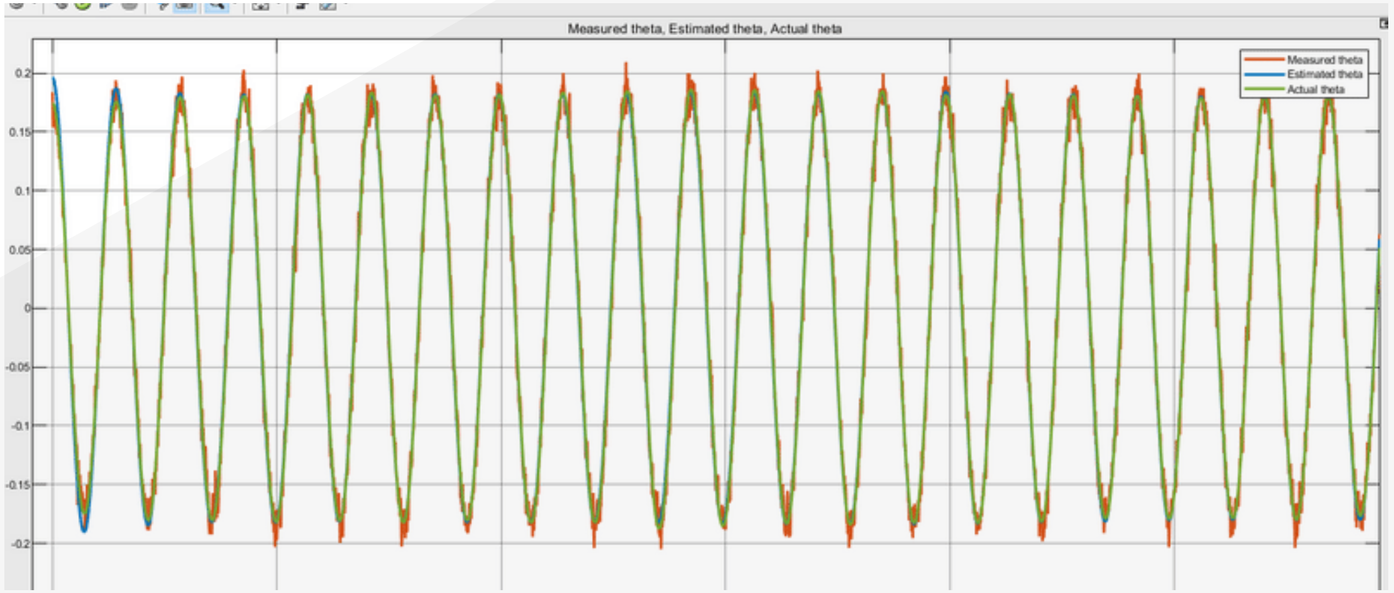


MATLAB CODE

```
% Pendulum model
%Gravity
g= 9.81; %[m/s^2]
% Pendulum mass
m=1; %[kg]
% Pendulum length
l=0.5; %[m]
%State space representation
A=[0 1 ; -g/1 0];
B=[0; 1/(m*l^2)];
C=[1 0];
D= 0;
% Process Noise Covariance
Q= 1e-3;
%Measurement noise covariance
R= 1e-4;
%Sampling time
Ts= 0.01; %[s]
```

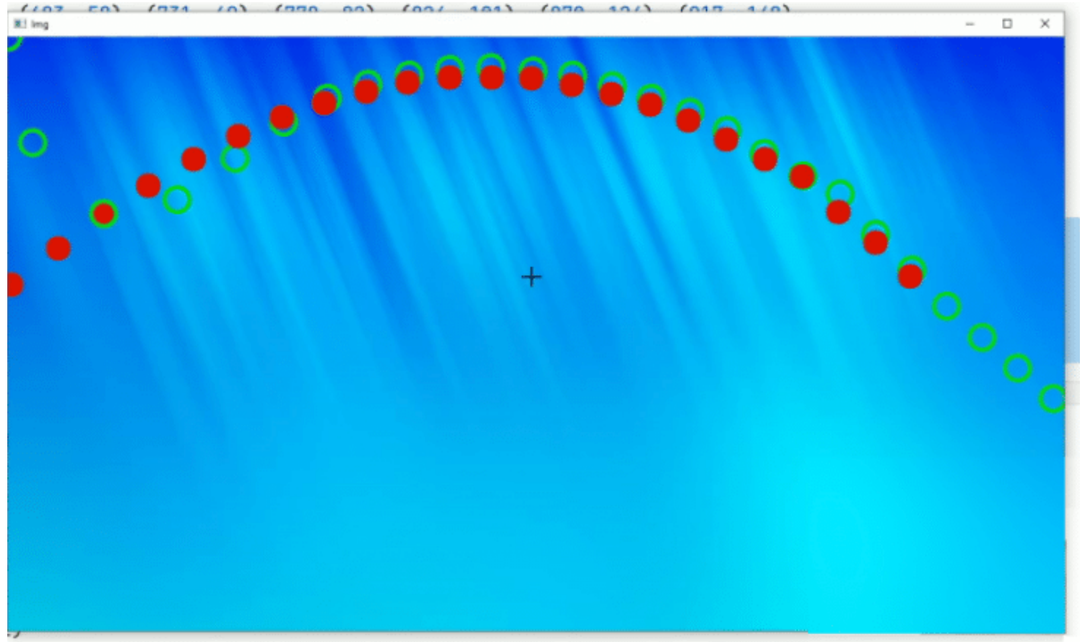
BLOCK DIAGRAM OF PENDULUM SYSTEM AND GRAPH





PREDICTING THE TRAJECTORY OF AN OBJECT WITH KALMAN FILTER

We will see a practical approach on how to use the **Kalman filter** to track and predict the trajectory of an object.This is used in many fields such as sensors, GPS, to predict the position in case of signal loss for a few seconds and this is what we will also see in computer vision.



PYTHON CODE FOR KALMAN FILTER

kalmanfilter.py > ...

```python
import cv2
import numpy as np


class KalmanFilter:
    kf = cv2.KalmanFilter(4, 2)
    kf.measurementMatrix = np.array([[1, 0, 0, 0], [0, 1, 0, 0]], np.float32)
    kf.transitionMatrix = np.array([[1, 0, 1, 0], [0, 1, 0, 1], [0, 0, 1, 0], [0, 0, 0, 1]], np.float32)


    def predict(self, coordX, coordY):
        ''' This function estimates the position of the object'''
        measured = np.array([[np.float32(coordX)], [np.float32(coordY)]])
        self.kf.correct(measured)
        predicted = self.kf.predict()
        x, y = int(predicted[0]), int(predicted[1])
        return x, y

```

PYTHON CODE FOR EXAMPLE

main.py > ...

```python
from kalmanfilter import KalmanFilter
import cv2

# Kalman Filter
kf = KalmanFilter()

img = cv2.imread("blue_background.webp")

ball1_positions = [(50, 100), (100, 100), (150, 100), (200, 100), (250, 100), (300, 100), (350, 100), (400, 100), (450, 10

ball2_positions = [(4, 300), (61, 256), (116, 214), (170, 180), (225, 148), (279, 120), (332, 97),
                   (383, 80), (434, 66), (484, 55), (535, 49), (586, 49), (634, 50),
                   (683, 58), (731, 69), (778, 82), (824, 101), (870, 124), (917, 148),
                   (962, 169), (1006, 212), (1051, 249), (1093, 290)]

for pt in ball2_positions:
    cv2.circle(img, pt, 15, (0, 20, 220), -1)
    predicted = kf.predict(pt[0], pt[1])

    cv2.circle(img, predicted, 15, (20, 220, 0), 4)

for i in range(10):
    predicted = kf.predict(predicted[0], predicted[1])
    cv2.circle(img, predicted, 15, (20, 220, 0), 4)

    print(predicted)


cv2.imshow("Img", img)
cv2.waitKey(0)
```

orange_prediction.py > (e) y

```python
import cv2
from orange_detector import OrangeDetector
from kalmanfilter import KalmanFilter

cap = cv2.VideoCapture("ball.MOV")

# Load detector
od = OrangeDetector()

# Load Kalman filter to predict the trajectory
kf = KalmanFilter()

while True:
    ret, frame = cap.read()
    if ret is False:
        break

    orange_bbox = od.detect(frame)
    x, y, x2, y2 = orange_bbox
    cx = int((x + x2) / 2)
    cy = int((y + y2) / 2)

    predicted = kf.predict(cx, cy)
    #cv2.rectangle(frame, (x, y), (x2, y2), (255, 0, 0), 4)
    cv2.circle(frame, (cx, cy), 20, (0, 0, 255), 4)
    cv2.circle(frame, (predicted[0], predicted[1]), 20, (255, 0, 0), 4)

    cv2.imshow("Frame", frame)
    key = cv2.waitKey(100)
    if key == 27:
        break
```

Grid Synchronization

APPLICATION TO POWER SYSTEMS

Load Forcasting

GRID SYNCHRONIZATION

The world is evolving towards the use of green, renewable and sustainable energy. This energy is gotten for various sources such as solar farm, wind farms, Nuclear plants amongst others. This newers plants are used together with the traditional fossil fuel plants

The problem with this system is with matching the frequency, Phase, and amplitude of the several newer plants to the tune of the older plants on the power grid.

The Kalman filter is used after generating an estimate of the phase values and measuring phase values such as amplitude, frequency, phase on the line the Kalman filter, the values are compared, and the filter generates the optimal estimate which is used by the system to make or adjust the output from the inverter.

LOAD FORECASTING

One of the first applications of the KF in the power system area had been to forecast the total load demanded by a multi-node system

Temporal load data, collected by the various agents in the power system administration, are used to predict load conditions, and the KF is frequently a fundamental part of the
algorithm.

Normally, the collected data are hourly based, and the prediction algorithm must yield short-term results, that would be useful in scheduling the actual system to supply the daily demand, and medium and long-term results, what would be useful in, for example, expansion planning and annual maintenance scheduling

OTHER USES AND APPLICATIONS

Computer Vision Object Tracking

Guidance, Navigation, and Control

Body Weight Estimate on Digital Scale

Radar sensors for position awareness used in Self driving vehicles

Time Series analysis such as signal processing and econometrics

MORE ON KALMAN FILTERS

Advantages of Kalman Filter

Drawbacks of Kalman Filter

Alternative to the Kalman Filter

ALTERNATIVES TO THE KALMAN FILTER

Kalman filters belong to a general group of filters called **Bayes Filters**. Bayes Filtering is the general term used to discuss the method of using a predict/update cycle to estimate the state of a dynamical system from sensor measurements. As mentioned, two types of Bayes Filters are **Kalman filters** and **particle filters.**

COMPARISON BETWEEN PARTICLE AND KALMAN FILTERS

- Kalman filter can be used for linear or linearized processes and measurement system, the particle filter can be used for nonlinear systems.

- Also, the uncertainty of Kalman filter is restricted to Gaussian distribution, while the particle filter can deal with non-Gaussian noise distribution.

ADVANTAGES OF THE KALMAN FILTER

- Kalman filters are ideal for systems which are continuously changing. They have the advantage that they are light on memory), and they are very fast, making them well-suited for real-time problems and embedded systems.

- Kalman Filter is very robust to measurement noise and does not depend on good initial conditions (after finite time the filter will converge to the correct system state) and by calculating the Kalman gain and estimating the state-covariance a measure for the confidence in the estimate is available.

- It can be fed multiple measurements of different natures at any time.

- Good results in practice due to optimality and structure.
- Easy to formulate and implement given a basic understanding.

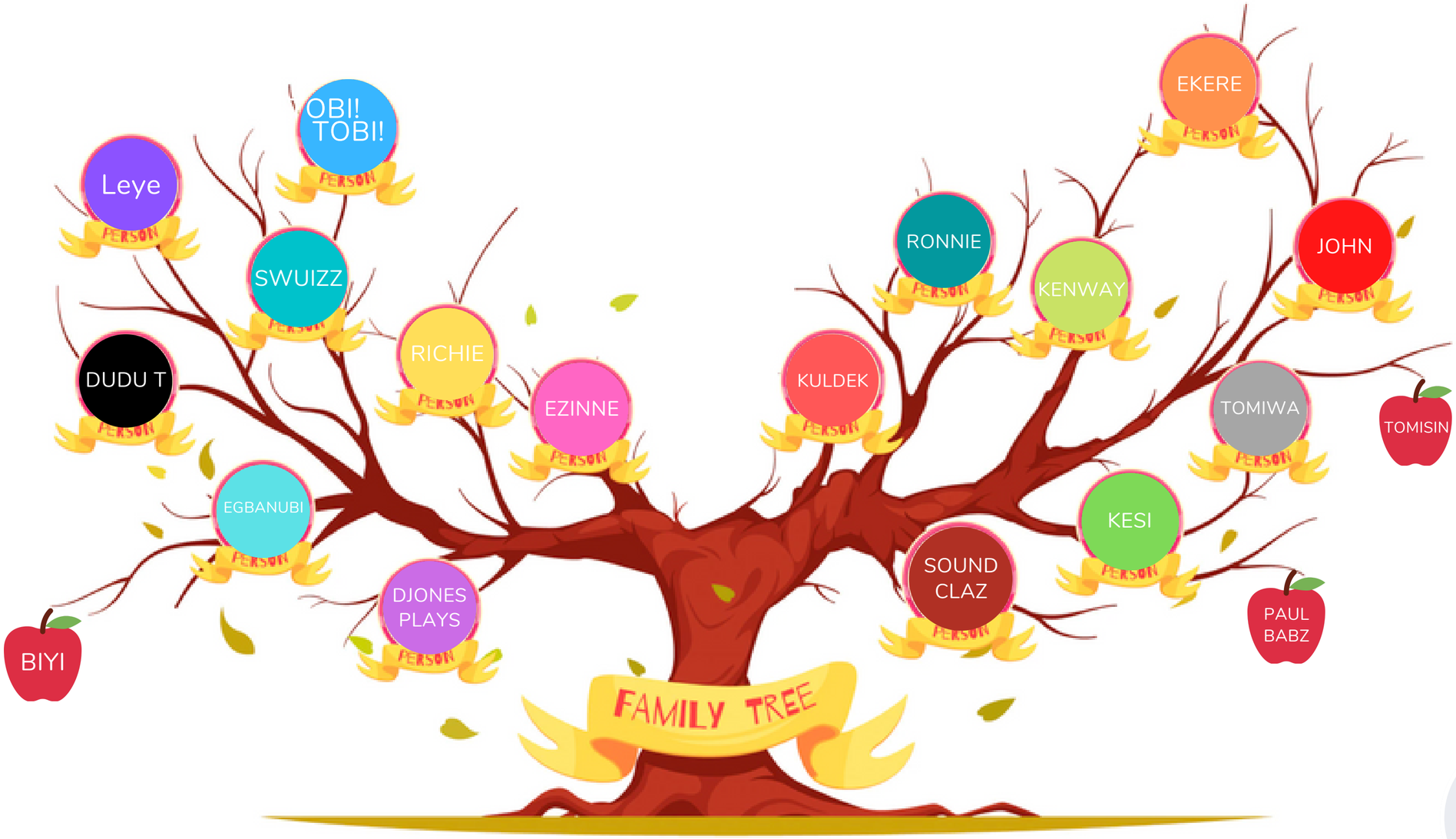
DRAWBACKS OF THE KILMAN FILTER

- In rapid change situation, sometimes Kalman filter has a slow reaction speed.
- Kalman filter does not include model uncertainty, hence the model has to be known.
- It assumes that the state belief is Gaussian distributed.

CONCLUSION

The aim of the project was to give a detailed explanation of the Kalman filter.

THE OPTIMAL TEAM



THANK YOU