Estimating liquid level in a tank, located on a ship in stormy weather, by an Extended Kalman Filter

Specialization Project

Trondheim, autumn 2022

CANDIDATE (surname, first name):			
Bampoye Abubakar			
DATE	**Course**	**PAGE/ATTACHMENT**	**BIBL. NR: 28**
20.12.22	TKP458	34 / 29	
SUPERVISOR(S):			
Supervisor Johannes Jäschke & Co-supervisor Halvor Aarnes Krog			
TITLE:			
Estimating liquid level in a tank, located on a ship in stormy weather, by an Extended Kalman Filter			

Abstract:

This specialization project aims to implement a nonlinear Kalman Filter to estimate the values of unknown liquid volume in a dynamic system inside a tank. Given that we have two coordinate systems, one is defined by gravity (water surface of the ocean). The second is given by the boat. When there are waves, the boat rolls, and we get an angle theta (θ), between these coordinate systems. If the tank is filled with liquid on the boat, the liquid will follow the coordinate system defined by gravity, while the tank and its measuring instruments follow the ship's reference system. The angle theta (θ) will therefore also affect the level measurements in the tank, and if this is not corrected, we will measure the wrong liquid volume in the tank. We want to make this correction using an Extended Kalman filter. The Extended Kalman filter gave reasonable RMS values for different process disturbance $Q = 1$ gave $RMS = 0.35$, $Q = 10$ gave $RMS = 0.016$, and $Q = 100$ gave $RMS = 0.025$.

Contents

1 Introduction **5**

2 Theoretical Background **7**
2.1 PID control 7
2.1.1 Time discrete PID controller 7
2.1.2 Tuning of PID controllers 9
2.2 Stochastic signals 11
2.2.1 General statistics 11
2.2.2 White noise 13
2.3 The Kalman Filter 13
2.3.1 Observability of discrete-time systems 14
2.3.2 The process model 16
2.3.3 Meauserment model 17
2.3.4 The steps of Kalman Filter algorithm 18
2.3.5 Kalman Filter tuning 20

3 A mathematical model applied on a tank **21**
3.0.1 Derivation of the mathematical model 21
3.0.2 Applying the mathematical model inside the KF algorithm 22

4 Results & Discussions **25**

5 Conclusion **28**

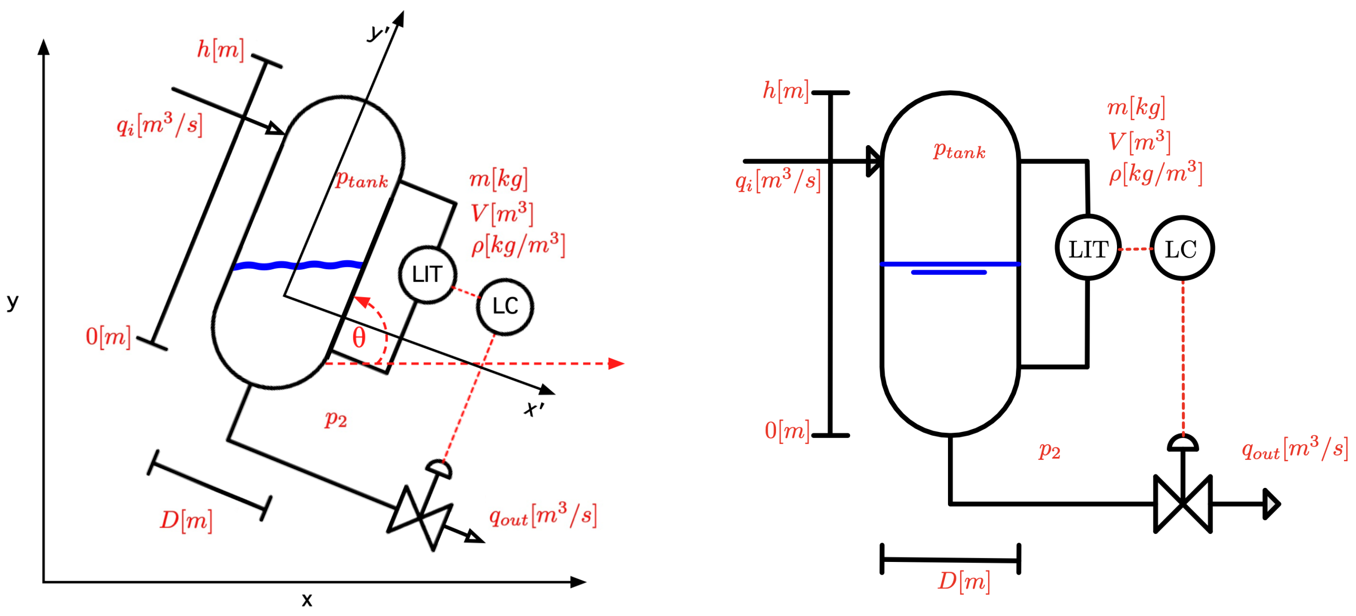
A Code listing **30**

List of abbreviations

DCS	distributed control system
EKF	Extended Kalman Filter
KF	Kalman Filter
MPC	Model predictive controller
MV	Manipulated values
PID	Proportional-integral-derivative
PDF	Probability distribution function
PV	Process values
RMS	Root mean square
SP	Set-points

1 Introduction

In process control, we aim to keep process values close to their desired set points for safety reasons, reduce cost or increase profitability. A classic control challenge is to control the liquid level in a tank close to the desired set point. A control algorithm, such as a PID controller, must know the value of the process variable to keep it close to the desired set point. For land-based tanks, the volume of the liquid in the tank is proportional to the fluid height. A sensor that monitors the fluid height (dP), is proportional to the density (ρ) of the liquid, gravitational constant (g), and the height (h) given as follows: $dP = \rho g h$. On land, the sensor provides reasonable control over the fluid volume in the tank. If the tank is on a boat, this becomes more complicated. The boat will roll and be affected by the waves, and thus, the liquid height h at the given location the sensor is placed will change with the angle θ. This will affect the measured volume of liquid, but not the real volume of liquid. But how can we solve this problem?



(a) Depiciton of the tank on a boat, with a change in the angle θ

(b) Depiciton of the tank on land.

Figure 1: The Figure depicts our tank from different angles.

One can solve the problem by using a PID regulator, but the results will be oscillations at the measured liquid level because of the waves. Since the process value (PV) deviates from the set-point (SP), the PID regulator will use (Manipulated values) MV all the time, and the valve will oscillate even in a steady state. This causes the valve to wear out.

The current solution to the problem is to use moving average measurements. The problem with this is that if we get a real disturbance which is not a wave, then we get a measurement delay. We, therefore, need a better method for determining the volume of liquid in the tank, which could be state estimation.

State estimators use the framework of combining a process model with a measurement model to predict the condition at the next time step[7]. The measurement from the system is used to correct this prediction and thus achieves an "optimal state estimate". State estimation can be used to improve existing measurements by removing measuring noise, to estimate unmeasured process values, or estimating unknown parameters and disturbances in a process.

The Kalman filter is perhaps the most widely used state estimator[8]. One reason for its popularity is because mathematically, the Kalman Filter is a set of equations that provides an efficient computational (recursive) mean to estimate the state of a process[1]. The main idea is to minimize the mean of the squared error. The Kalman Filter is a commonly used algorithm to estimate values of unknown state variables of a dynamic system[5]. The Kalman Filter is composed of a process simulator, and a sensor simulator which uses the process model, which it assumes is a true representation of the process, and a sensor model to estimate the states. The difference between the actual and simulated (estimated) measurements is used to correct the present estimate. That difference is called the innovation variable or the innovation "process". The correction is proportional to the innovation variable. The proportional gain, which is a matrix of time-varying elements, is called the Kalman Filter gain, K_k. The Kalman Filter uses process knowledge in terms of models and measurements to calculate the state estimate.

Why do we seek to investigate if a physical process like the fluid volume inside a tank can be maintained sufficiently close to a set point, and what do we obtain? One reason for seeking a reasonably small error between the set-point and the control error might be safety, such that we can guarantee security for humans and equipment[3]. Also, it may be required to keep the liquid volume within a specific limit. Thus, the variable must be controlled. Therefore by implementing an automatic control solution, we can accomplish tedious and dangerous operations for the benefit of human operators. Also, automation may reduce costs, thereby indirectly reducing product prices to the customer's benefit.

This specialization project aims to implement a nonlinear Kalman Filter to estimate the values of unknown liquid volume in a dynamic system inside a tank. It is divided into five sections, the theoretical background in chapter 2, a mathematical model applied on a tank 3, the results and discussions in chapter 4, and closed off by the conclusion in chapter 5.

2 Theoretical Background

This chapter contains the theoretical framework required for this project. PID regulation is described in sub-section 2.1. The remaining sub-sections are required for describing the state estimator. Sub-section 2.2 is about Stochastic signals, and 2.3 is about The Kalman Filter.

2.1 PID control

The PID - proportional-integral-derivative - controller function has become a standard for technical and industrial applications[4]. The PID controller is implemented with a discrete-time algorithm that calculates the control signal at a discrete point in time. A control system of a computer-based PID is depicted in Figure 2.

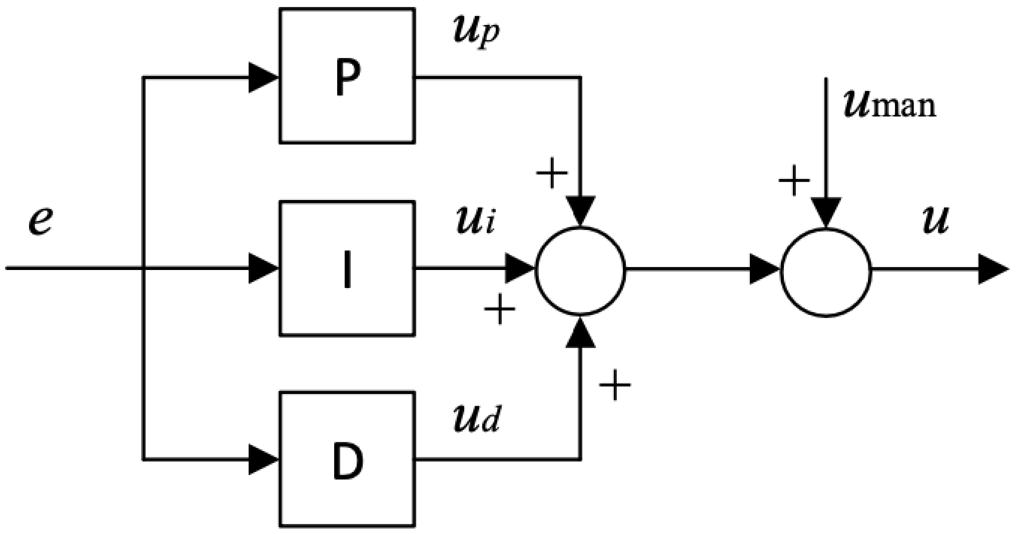


Figure 2: The Figure shows a PID control system.

Source: The image is from the book "*Modeling, Simulation and Control*" by F. A. Haugen.

2.1.1 Time discrete PID controller

We need the continuous-time PID controller as our basis to develop a discrete-time PID algorithm[4]. Which is given by:

$$\begin{aligned} u &= u_{man} + \underbrace{K_p e}_{P-term, u_p} + \underbrace{\frac{K_p}{T_i}\int_0^t e d\tau}_{I-term, u_i} + \underbrace{K_p T_d e}_{D-term, u_d} \\ &= u_{man} + u_p + u_i + u_d \end{aligned} \tag{1}$$

Where u is the control signal and is calculated by the regulator, u_{mann} is the nominal value of the control variable and is the control signal available for adjustment by the operator while the controller is in manual mode. Since u_{man} is constant, and hence “passive” when the controller is in automatic mode. Its contribution in automatic mode is to provide kind of a reasonable initial value at the moment of switching from manual to automatic mode. e is the control error given by Equation 2:

$$e_k = y_{sp,k} - y_{mf,k} \tag{2}$$

The second term is u_p, which is the proportional term and contributes with a term in the total control signal, u, which is proportional to the control error, e. Which brings some speed to the control. However, assuming u_{man} is not “perfect” to give zero control error, i.e., $e = 0$, the P term by itself can not ensure $e = 0$, either this is because with $e = 0$, $u_p = 0$, which mean no contribution from the P term. In other words, the P controller can ensure zero error in a steady state.

The third term is u_i, which is the integral term calculated from when the regulator was put into operation to the current time. It's an important part of the PID controller because it ensures zero steady-state control error, i.e., $e_s = 0$. That's because as long as e is different from zero, u_i will change. In other words, e is an “improvement term”. Or, e drives the control. This change continues until e has become zero, and then u_i and u are kept constant until some disturbance or setpoint changes causes e to become nonzero.

The last term is u_d, which is the derivative term. It stabilizes the control system, which otherwise can not be stabilized with a P or PI controller. It amplifies the random measurement noise, causing large variations in the control signal. These variations will be reduced with a lowpass filter acting on the process measurement.

The controller parameters in Equation 1 are:

- Proportional gain K_p
- Integral time T_i
- Derivative time gain T_d

We obtain the PID algorithm by discretizing the terms from Equation 1.

The discretization of the P term results into:

$$u_{p,k} = K_p e_k \tag{3}$$

The discretization of the I term results into:

$$\begin{aligned} u_{i,k} &= \frac{K_p}{T_i}\int_0^{t_k} e d\tau \\ &= \frac{K_p}{T_i}\Big(T_s e_1 + \ldots + T_s e_{k-1} + T_s e_k\Big) \\ &= \frac{K_p T_s}{T_i}\Big(e_1 + \ldots + e_{k-1} + e_k\Big) \\ &= \underbrace{\frac{K_p T_s}{T_i}\Big(e_1 + \ldots + e_{k-1} + e_k\Big)}_{u_{i,k-1}} + \frac{K_p T_s}{T_i} e_k \\ &= u_{i,k-1} + \frac{K_p T_s}{T_i} e_k \end{aligned} \tag{4}$$

The discretization of the D term results into:

$$u_{d,k} = K_p T_d \frac{e_k - e_{k-1}}{T_s} \tag{5}$$

2.1.2 Tuning of PID controllers

In our model, we will tune the controller with the Skogestad tuning method[9]. It's a model-based tuning method where the controller parameters are expressed as functions of the process model parameters. A block diagram of the scheme is illustrated in Figure 3.

The transfer function $H_p(s)$ is a combined transfer function of the process, the sensor, and the measurement filter, and it's also referred to as the "process transfer function" although it is a combined transfer function. The block diagram shows a disturbance acting on the process, we are not going to use that in our tuning in this project. Still, it's recommended to test the tuning on a simulation to see how the control system compensates for a process disturbance. The design principle of the Skogestads method is as follows. The transfer function from the setpoint to the filtered process measurement is specified as a first-order transfer function with delay:

$$T(s) = \frac{y(s)}{y_{sp}(s)} = \frac{1}{T_c s + 1} e^{-\tau} \tag{6}$$

Where T_c is the time constant of the control system, which the user must specify, and τ

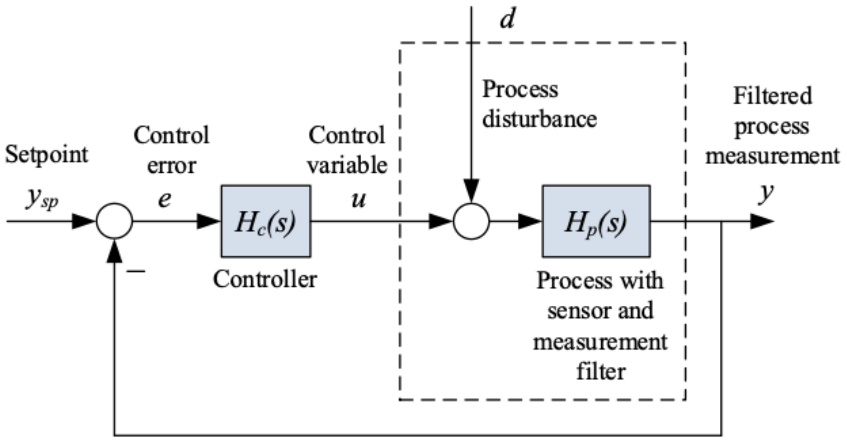


Figure 3: The Figure shows a block diagram of the control system in the Skogestad controller tuning method.

Source: The image is from the book "*Modeling, Simulation and Control*" by F. A. Haugen.

is the process time delay that the process model gives. This model is also applicable for processes without time delay. Another way of deriving the transfer function is from the block diagram by using the feedback rule in Figure 3:

$$T(s) = \frac{H_c(s)H_p(s)}{1 + H_c(s)H_p(s)} \tag{7}$$

But $H_c(s)$ is unknown in Equation 7, so by solving for $H_c(s)$ result into:

$$H_c(s) = \frac{T(s)}{(1 - T(s))H_p(s)} \tag{8}$$

$H_c(s)$ becomes a PID controller or a PI controller for the assumed transfer by letting Equation 6 equal to Equation 7, and applying some proper simplifications to approximate the time delay term.

Our process is an integrating process, and the model of an integrator with time delay is given by:

$$\dot{y}(t) = K_i u(t - \tau) \tag{9}$$

Where the corresponding transfer function is:

$$\frac{y(s)}{u(s)} = H_p(s) = \frac{K_i}{s} e^{-\tau s} \tag{10}$$

The controller settings are as follows:

$$\begin{aligned} K_p &= \frac{1}{K_i(T_c + \tau)} \\ T_i &= 4(T_c + \tau) \\ T_d &= 0 \end{aligned} \tag{11}$$

2.2 Stochastic signals

The Kalman filter uses the characteristics of assumed random process disturbances, and random measurement noise[6]. To reflect the real-world behaviour of the system that we are filtering, which will vary more or less randomly. Even measurement signals contain random noise; the sensors are noisy, and we cannot rely on sensors to give us perfect information. The process disturbances as well have some random components. Consequently, control signals and controlled variables, i.e. process output variables, have some random behaviour. The future value of a random signal can not be predicted precisely, i.e. such signals are non-deterministic. Thus this subsection will give us an insight into how random signals can be described by statistical measures, which are typically expectation value (mean value), variance, and standard deviation.

2.2.1 General statistics

But how do we characterize a stochastic process? A stochastic process may be characterized by its mean given by Equation 12, standard deviation given by Equation 16, or variance given by Equation 15. The stochastic process can be observed via one or more realizations of the process in the form of a sequence or time series of samples. Another realization of the same stochastic process will undoubtedly show different sample values, but the mean value and the variance will be almost the same; the longer the realization sequence is, the more equal the mean values and the variance will be.

$$m_x = \frac{1}{N} = \sum_{k=0}^{N-1} x_k \tag{12}$$

Where m_x and x_k can be expressed in vector form as follows:

$$x_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ \vdots \\ x_{n,k} \end{bmatrix} \tag{13}$$

$$m_x = \begin{bmatrix} m_{x,1} \\ m_{x,2} \\ \vdots \\ x_{x,n} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} = \sum_{k=0}^{N-1} x_{1,k} \\ \frac{1}{N} = \sum_{k=0}^{N-1} x_{2,k} \\ \vdots \\ \frac{1}{N} = \sum_{k=0}^{N-1} x_{n,k} \end{bmatrix} \tag{14}$$

$$\begin{aligned} \sigma^2 = Var(x) &= E[x_k - m_x]^2 \\ &= \frac{1}{N-1}\sum_{k=0}^{N-1}[x_k - m_x]^2 \end{aligned} \tag{15}$$

$$\sigma = \sqrt{Var(x)} \tag{16}$$

In our case, our stochastic signal will vary with time. Therefore it can be useful to express it by the auto-covariance, which is given by:

$$R_x(L) = E\{[x_{k+L} - m_x][x_k - m_x]\} \tag{17}$$

Where L is the lag, and by observation, one can see that the argument of the auto-covariance function is the lag L. If the mean is zero, we can express the auto-covariance as follow in a higher dimension:

$$\begin{aligned} R_x(L) &= E\{[x_{k+L}][x]^T]\} \\ &= E\left\{\begin{bmatrix} x_{1,k+L} \\ x_{2,k+L} \end{bmatrix} \quad [x_{1,k}x_{2,k}]\right\} \\ &= \begin{bmatrix} E[x_{1,k+L} \cdot x_{1,k}] & E[x_{1,k+L} \cdot x_{2,k}] \\ E[x_{2,k+L} \cdot x_{1,k}] & E[x_{2,k+L} \cdot x_{2,k}] \end{bmatrix} \end{aligned} \tag{18}$$

The auto-covariance results in the covariance on the diagonal if $L = 0$:

$$R_x(0) = \begin{bmatrix} \underbrace{E\{[x_{1,k}^2]\}}_{=Var(x_1)} & E[x_{1,k} \cdot x_{2,k}] \\ E[x_{2,k \cdot x_{1,k}}] & \underbrace{E\{[x_{2,k}]^2\}}_{=Var(x_2)} \end{bmatrix} \tag{19}$$

If we have two different signals, the cross-covariance is given by:

$$\begin{aligned} R_{x,y}(L) &= E\{[x_{k+L} - m_x][y_k - m_y]\} \\ &= S\textstyle\sum_{k=0}^{N-1-|L|}[x_{k+L} - m_x][y_k - m_y], \quad L = 0, 1, 2, ... \\ &= S\textstyle\sum_{k=0}^{N-1-|L|}[y_L - m_y][x_k - m_x] = R_{y,x}(-L), \quad L = -1, -2, ... \end{aligned} \tag{20}$$

Where S is a scaling factor. If there is a correlation between the time it is coloured noise.

2.2.2 White noise

An essential type of stochastic signal is the so-called white noise signal or processes. “White” is because the noise contains approximately equally of all frequency components, analogously to white light, which contains all colours. The reason that it’s important is that the random noise, which is always present in measurements, can be represented by white noise. White noise has zero mean value, and there is no co-variance or relation between sample values at different time-indexes; hence the auto-covariance is zero for all lags L except for $L = 0$. Thus, the auto-covariance is the pulse function depicted in Figure 4.

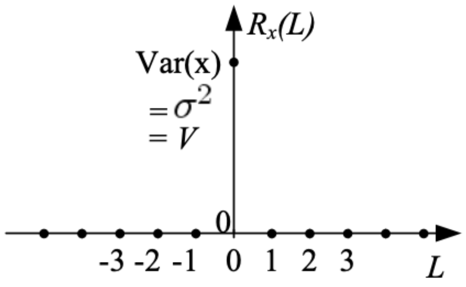


Figure 4: The Figure shows white noise, which has an auto-covariance function like a pulse function.

Mathematically the auto-covariance function of white noise is given by:

$$R_x(L) = Var(x)\delta(L) = \sigma^2(x)\delta(L) = V\delta(L) \tag{21}$$

Where we introduce V as the short-hand symbol of the variance. $\delta(L)$ is the unit pulse defined as follows.

$$\delta(L) = \begin{cases} 1 & \text{if } L = 0 \\ 0 & \text{else} \end{cases} \tag{22}$$

2.3 The Kalman Filter

This subsection provides us with the right tools to implement and design a Kalman Filter(KF), and Figure 5 illustrates the principle behind the KF[5].

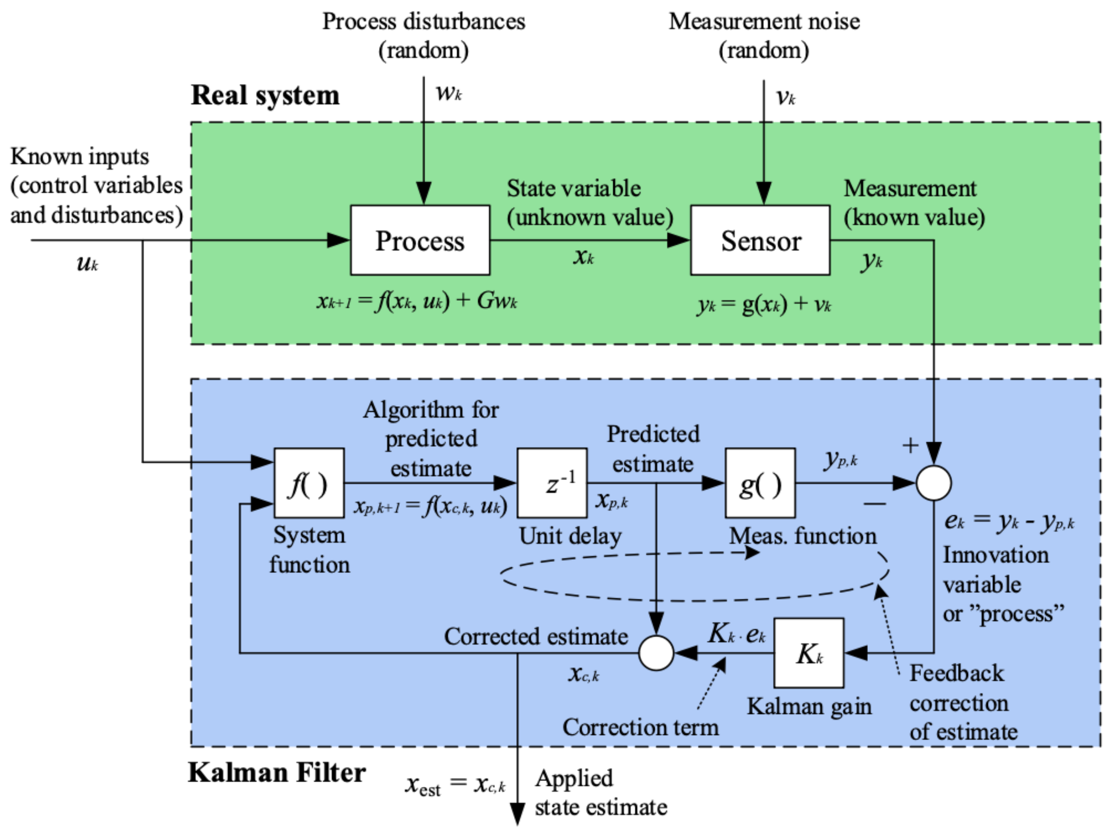


Figure 5: The Figure shows the principle of the Kalman Filter, represented by a block diagram.

Source: The image is from the book "*Modeling, Simulation and Control*" by F. A. Haugen.

2.3.1 Observability of discrete-time systems

A necessary condition for the KF to work correctly is that the system for which the states are to be estimated is observable and can be checked numerically. It's advised to check for observability before applying the KF. The observability presented here applies only to linear state space models, which can be derived by linearizing a nonlinear model. The observability of discrete-time systems can be defined as follows:

$$x_{k+1} = Ax_k + Bu_k \tag{23}$$

$$y_k = Cx_k + Du_k \tag{24}$$

The discrete-time system is observable if there is a finite number of time steps k so that sequence $u_0, ..., u_{k-1}$ and the output sequence $y_0, ..., y_{k-1}$ is sufficient to determine the initial state of the system, x_0. But we need a criterion for the system to be observable. Since the influence of input u on state x is known from the model, let us, for simplicity, assume that $u_k = 0$ Equation 23 and 24 result into:

$$x_{k+1} = Ax_k \tag{25}$$

$$y_k = Cx_k \tag{26}$$

By substituting Equation 25 inside 26 the result are as follows:

$$\begin{gathered} y_k = CAx_0 \\ \vdots \\ y_{n-1} = CA^{n-1}x_0 \end{gathered} \tag{27}$$

Which can be expressed as:

$$\underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}}_{M_{obs}} x_0 \tag{28}$$

Let's define the M_{obs} as our observability matrix:

$$M_{obs} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{29}$$

Thus our system based on Equation 23 and 24 is observable if and only if the observability matrix 29 has a rank n, and n is the order of the system model which is also equal to the number of state variables. The rank can be checked by calculating the determinant of M_{obs}. If the determinant is non-zero, the rank is full; hence, the system is observable. If the determinant is zero, the system is non-observable. But what are the consequences of a non-observable system other than that the KF may not work correctly? Some state variables or linear combinations of state variables may not respond to the estimated measurement; therefore, the innovation process cannot correct their estimates. Another consequence may be that the value of the estimator may diverge.

2.3.2 The process model

Let us assume the following discrete-time state space model:

$$x_{k+1} = \underbrace{Ax_k + Bu_k}_{f(x_k,u_k)} + Gw_k \tag{30}$$

The model contains the following state variables given in Equation 31, and state inputs given in Equation 32:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{31}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \tag{32}$$

The value of u is assumed to be known, including control variables and known disturbances. The system vector function is given by Equation 33, and random white disturbance is given by Equation 34:

$$f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{33}$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_q \end{bmatrix} \tag{34}$$

With an auto-covariance Q, which we assume to be diagonal since each of the process disturbances typically are assumed to act on their respective state independently, given as:

$$Q = \begin{bmatrix} Q_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q_{nn} \end{bmatrix} \tag{35}$$

The number of q of process disturbance is assumed to equal the number n of state variables. The noise gain matrix G, which relates the noise of the state variables, is also assumed to have the same dimension, which makes G square.

$$G = \begin{bmatrix} G_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & G_{nn} \end{bmatrix} \tag{36}$$

Commonly the element of G is assumed to be one, which makes G an identity matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_n \tag{37}$$

2.3.3 Meauserment model

The measurement model is given by:

$$y_k = \underbrace{Cx_k + Du_k}_{g(x_k,u_k)} + v_k \tag{38}$$

The measurement vector y is given by Equation 39, and the measurement function g is given by Equation 40:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} \tag{39}$$

$$g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_r \end{bmatrix} \tag{40}$$

Which contains r elements. The random measurement noise vector is given by v:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_q \end{bmatrix} \tag{41}$$

With an auto-covariance R, which is typically assumed to be diagonal, given as:

$$R = \begin{bmatrix} R_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R_{nn} \end{bmatrix} \tag{42}$$

Let's define the estimation error vector to be given by:

$$e_{x,k} = x_{est,k} - x_k \tag{43}$$

Where x_k is the assumed actual state vector, and x_{est} is the state estimate. The estimate of the KF is the minimal value of the expectation value of the sum of the estimation errors. Which is the sum of squared errors:

$$E[e_{x,k}^T e_k] = E[e_{x_{1,k}}^2 + \ldots + e_{x_{n,k}}^2] \tag{44}$$

Since this estimation technique assumes that the model is linear, it will only give an approximate result for nonlinear models.

2.3.4 The steps of Kalman Filter algorithm

The first step is the initialization step, which is executed only once before the loop start. Where the initial value $x_{p,0}$ of the predicted state estimate x_p is set equal to the initial guess:

$$x_{p,0} = x_{init} \tag{45}$$

We also need to initialize the auto-covariance matrix of the predicted state estimation error:

$$P_{p,k} = E[(x_k - m_{x_{p,k}})(x - m_{x,pk})^T] \tag{46}$$

Where the initial value is set to $P_{p,k} = P_{p,init}$, and is an educated guess as to how sure we are of the initial guessing $x_{p,0}$.

The second step is to calculate the Kalman Gain K_k, which is given by:

$$K_k = P_{p,k} C^T [C P_{p,k} C^T + R]^{-1} \tag{47}$$

Here, C is the measurement gain matrix of the linearized model of the original nonlinear model given by Equation 38 calculated at the most recent operating point, which is ($x_{p,k}$, u_k):

$$C = \left.\frac{\partial g}{\partial x}\right|_{op=(x_{x,p},u_k)} \tag{48}$$

The third step is the implementation of the estimation loop which is composed of first reading the measurement, $y_{m,k}$ from the sensor and calculating the predicted measurement from the predicted state according to the sensor model:

$$y_{p,k} = g(x_{p,k}) \tag{49}$$

The reason for not including the measurement noise v_k as in Equation 38 is because it's not known or not predictable since it's assumed to be white noise. Thus we can compute the innovation variable, which is step three, and is the difference between the accurate measurement, $y_{m,k}$ and the accurate measurement given by Equation 49:

$$e_k = y_{m,k} - y_{p,k} \tag{50}$$

Step four is to calculate the corrected state estimate $x_{c,k}$, also referred to as the *posterior* estimate, because it is calculated after the present measurement is taken. It is also denoted the *measurement-updated*. The computation is done by adding the corrective term $K_k e_k$ to the predicted state estimate $x_{p,k}$:

$$x_{c,k} = x_{p,k} + K_k e_k \tag{51}$$

Where K_k is the Kalman Gain given by Equation 47.

The fifth set Step is to calculate the auto-covariance of the measured corrected state estimate error:

$$P_{c,k} = [I - K_k C]P_{p,k} \tag{52}$$

Step six is to calculate the predicted state estimate for the next time step, $x_{p,k+1}$. This is also called the *prior* estimate because it is calculated before the present measurement is taken. It is also referred to as the *time-updated estimate*. The computation is done by using the current state estimate and the known input u_k in the process model:

$$x_{p,k+1} = f(x_{c,k}, u_k) \tag{53}$$

Where K_k is the Kalman Gain given by Equation 47, and C is the measurement gain given by Equation 48. In the first iteration of the estimation loop, $P_{p,k}$ is known from the initialization 46. In a subsequent iteration, $P_{p,k}$ is known from its predicted value.

The seventh step is to calculate the predicted state estimate error in the next iteration, and the auto-covariance of the state estimate error in the next iteration is defined as follows:

$$P_{p,k+1} = AP_{c,k}A^T + GQG^T \tag{54}$$

Where A is the transition matrix of a linearized model:

$$A = I + T_s A = I + T_s \frac{\partial f}{\partial x}\bigg|_{op=(x_{c,k},u_k)} \tag{55}$$

The last step is the index-shift to prepare for the next iteration:

$$\begin{aligned} x_{p,k} &= x_{p,k+1} \\ P_{p.k} &= P_{p,k+1} \end{aligned} \tag{56}$$

2.3.5 Kalman Filter tuning

The R matrix is typically from the datasheet of the instrument provider, and the dominant factor in tuning the Kalman filter is the relationship between Q and R, which is important. If $Q \gg R$, it means that the model is much more uncertain than the measurement, so then KF relies more on the measurements than the model and vice versa. A large Q gives great uncertainty in prior prediction, which can be observed by Equation 54, which is the process disturbance auto-covariance. A large Q tells us the variations in the actual state variables are relatively large, which will give a larger Kalman Gain K_k, giving a stronger updating of the estimates. This results in more measurement noise being added to the estimates because the measurement noise is a term in the innovation process e, which is multiplied by K_k:

$$\begin{aligned} x_{c,k} &= x_{x,p} + K_k e_k \\ &= x_{p,k} + K_k[g(x_k) + v_k - g(x_{p,k})] \end{aligned} \tag{57}$$

The Q matrix can be selected by setting all the diagonal elements to one if we don't have any idea about the numerical values of the process, and hence Q is:

$$Q = Q_0 \begin{bmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{58}$$

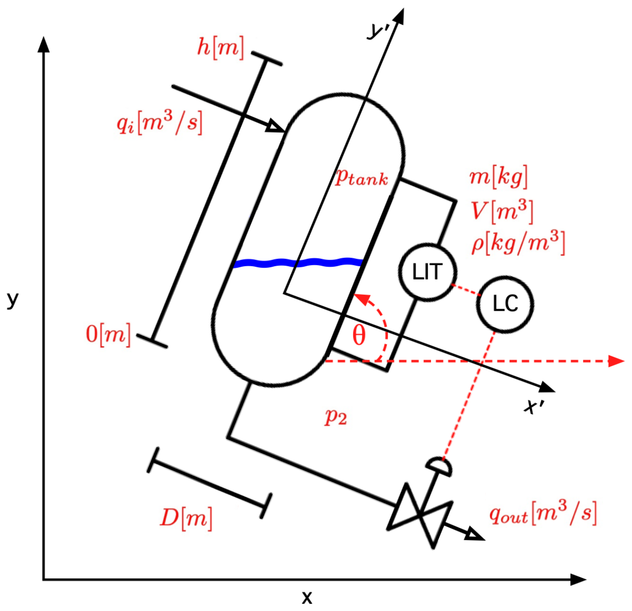
Where Q_0 is the only tuning parameter, there are methods to tune Q systematically, but in this project, we assume that Q is diagonal. We adjust the diagonal elements to fine-tune the process.

3 A mathematical model applied on a tank

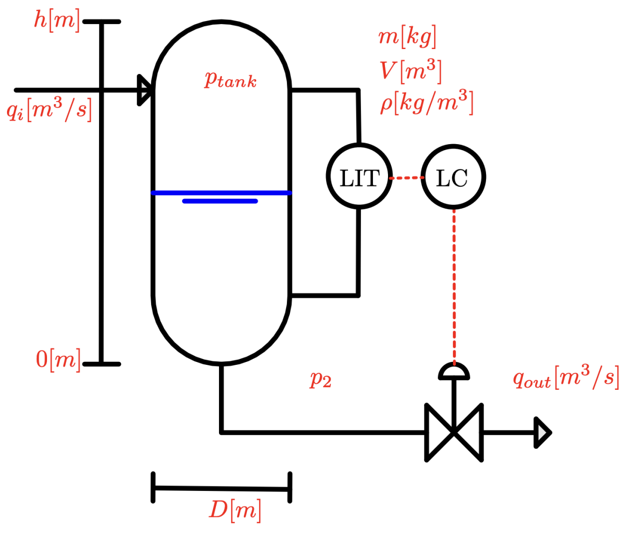
Our project aims to estimate the liquid volume in a tank located on a ship in stormy weather by a nonlinear Kalman Filter. Figure 6a depict that when the ship has a tilting movement, it results in an angle θ. This is a bit complicated to estimate, so we need to break it down to answer this question. This breaking down process involves fundamental activities: "We have a question, and we don't know the answer, so we ask another question, and the other question will be something that we judge to be easier to answer, and something that will shed light to the first question.". That's the core aim of this chapter. Therefore we are only looking at a tank that is on land, depicted in Figure 6b. Due to this simplification, we estimate fluid height and inflow, not fluid volume.

3.0.1 Derivation of the mathematical model

We can derive a mathematical model based on the first principle of the tank given in Figure 6b[2].



(a) Depiciton of the tank on a boat, with a change in the angle θ.



(b) Depiciton of the tank on land.

Figure 6: The Figure depicts our tank from different angles.

The rate of change in mass per unit of time is equal to net mass outflow. Mathematically it's given as:

$$\frac{dm(t)}{dt} = \sum \omega_{in}(t) - \sum \omega_{out}(t) + \sum \omega_{generated}(t) \tag{59}$$

Where $m[kg]$ is the mass, and $\omega[kg/s]$ is the mass flow, $t[sec]$ is the time argument. In our mathematical model of the tank, we assume no reaction is appearing inside the tank; thus $\sum \omega_{generated}(t) = 0$:

$$\begin{aligned}
\frac{dm(t)}{dt} &= \sum \omega_{in}(t) - \sum \omega_{out}(t) + \cancelto{0}{\sum \omega_{generated}(t)} \\
\frac{dm(t)}{dt} &= \sum \omega_{in}(t) - \sum \omega_{out}(t) \\
\frac{m(t)}{dt} &= \rho q_{in}(t) - \rho q_{out}(t)
\end{aligned} \tag{60}$$

Which is a differential equation for m. We need some conditions to mimic the real world. For instance, the mass can't be negative, thus $m \geq 0$. In our case, we are more interested in how the volume V will vary, and the relation between V and m is given by:

$$m = \rho V = Ah(t) \tag{61}$$

By substituting Equation 61 inside 60 result into:

$$\frac{\rho Ah(t)}{dt} = \rho q_{in}(t) - \rho q_{out}(t) \tag{62}$$

By assuming the density ρ is the same everywhere, in the inlet and outlet of the tank, and the cross-sectional area is constant, we can extract them outside the differential:

$$\begin{aligned}
\cancel{\rho} A\frac{h(t)}{dt} &= \cancel{\rho} q_{in}(t) - \cancel{\rho} q_{out}(t) \\
\frac{h(t)}{dt} &= \dot{h}(t) = \frac{1}{A}(q_{in}(t) - q_{out}(t))
\end{aligned} \tag{63}$$

3.0.2 Applying the mathematical model inside the KF algorithm

With conditions $h_{min} \leq h \leq h_{max}$, q_{out} is the outflow demanded by the level controller. Thus $q_{out} = u$ is our control signal:

$$\dot{h}(t) = \dot{q}_{in}(t) - u \equiv f_1 \tag{64}$$

Another assumption to simplify our model is by letting q_{in} be almost constant, which implies it changes slowly, and the results will be:

$$\dot{q}_{in} = 0 \equiv f_2 \tag{65}$$

The height h is measured hence:

$$h \equiv g \tag{66}$$

Assuming white measurement noise v_k with variance R_{11}, the measurement equation result into:

$$h_{m,k} = h_k + v_k \tag{67}$$

With the given measurement covariance matrix:

$$R = R_{11} = (\sigma)^2 = 3.33 \cdot 10^{-5} \tag{68}$$

Which were computed by the script in the Appendix A.

The discretized model of Equation:

$$h_{p,k+1} = h_{p,k} + \frac{T_s}{A}(q_{in,c,k} + u_k) \tag{69}$$

$$q_{in,p,k+1} = q_{in,c,k} \tag{70}$$

Here we have applied the Euler forward and added the white disturbance noise w_1 and w_2, which are independent and uncorrelated process disturbances with assumed variances Q_{11} and Q_{22}. The process disturbance matrix is then:

$$Q = \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & Q_{22} \end{bmatrix} \tag{71}$$

Where Q_{22} is going to be our main tuning factor of the KF, we will change it by “trial-and-error” to give a reasonably fast and smooth estimate of q_{in}.

The innovation variable in the simulation loop will be:

$$e_k = h_{m,k} - h_{p,k} \tag{72}$$

where $h_{m,k}$ is the simulated measurement, and $h_{p,k}$ is the predicted measurement. The Kalman gain from Equation 47 will have a measurement gain matrix as follows:

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{73}$$

Using the auto-covariance of measured corrected state estimate error from Equation 52 and the predicted state estimation error in the next iteration from Equation 54. The result of the disturbance gain, the discrete-time transition matrix:

$$G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{74}$$

$$A = I + T_s A = I + T_s \begin{bmatrix} \frac{\partial f_1}{\partial x_1} = 0 & \frac{\partial f_1}{\partial x_2} = \frac{1}{A_{areas}} \\ \frac{\partial f_2}{\partial x_1} = 0 & \frac{\partial f_2}{\partial x_2} = 0 \end{bmatrix} \Bigg|_{x_{p,k}, u_k} = \begin{bmatrix} 1 & \frac{T_s}{A_{areas}} \\ 0 & 1 \end{bmatrix} \tag{75}$$

4 Results & Discussions

The PID controller and the Kalman Filter implementation are given in the Python script from Appendix A. The result of implementing a PID controller is shown in Figure 7, where we have applied a step change in the SP after 2000 seconds. We observe the PID controller is decreasing the outflow to compensate for the increase in inflow, indicating the PID controller is working.

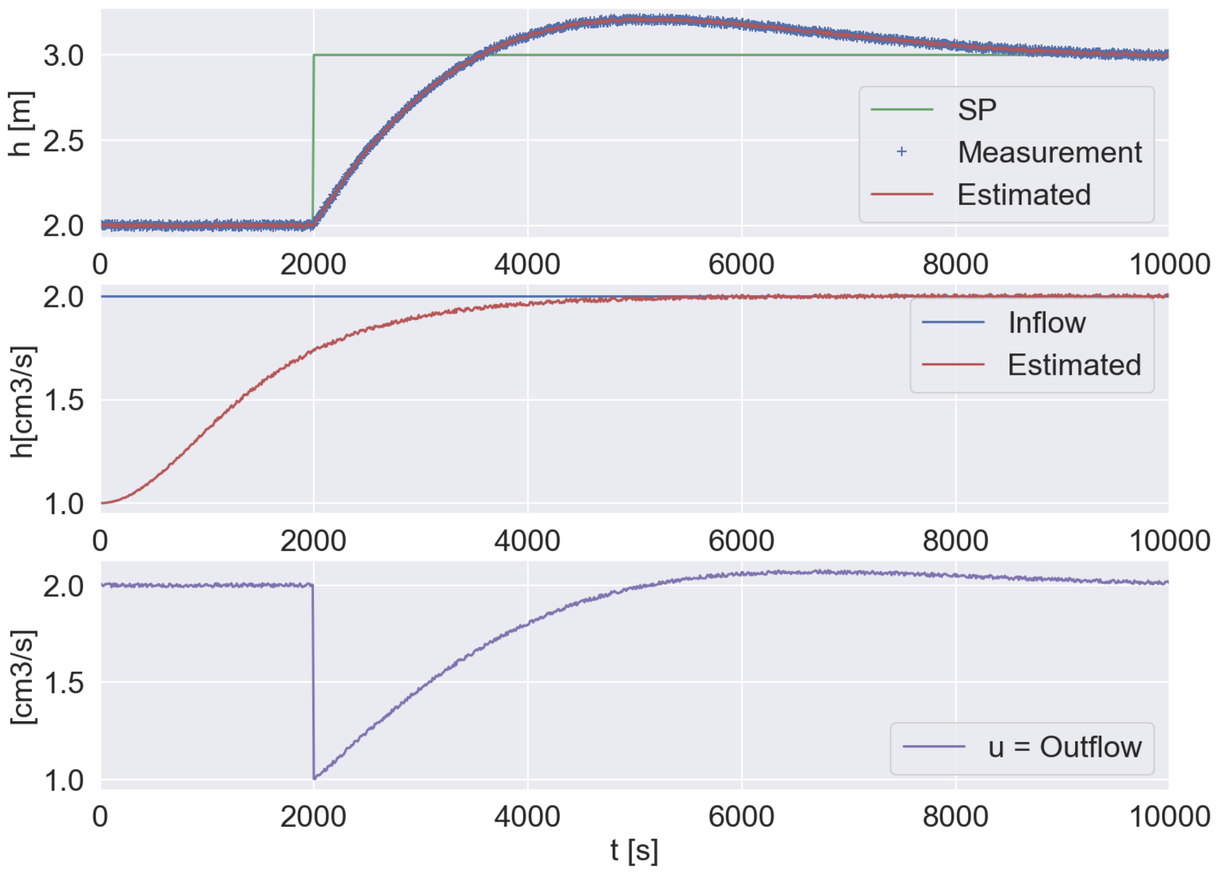


Figure 7: Illustrates of implementing a PID controller-

Figure 8, 9 and 10 show the simulated responses and the estimation of $q_{in} = q_{in,est}$ by the Kalman filter, and we changed the tuning factor Q, which is the process disturbance auto-covariance while keeping the disturbance variance unchanged. We observe that q_{in} is less noisy and slower for the case when $Q = 1$, which is depicted in Figure 8. But the model doesn't manage to predict the inflow accurately. From Table 1 we observe that the $RMS = 0.35$, which is calculated by taking between the deviation of the inflow q_{in} and the estimated value $q_{in,est}$.

When we change the tuning factor into $Q = 10$ depicted in Figure 9, more measurement noise is being added to the estimate but also faster. This makes sense since a larger Q tells us that the variance in the actual variables is relatively large. Thus it gives a larger Kalman gain. We also observe that the model manages to predict the inflow more

Table 1: The different RMS values by changing the process disturbance auto-covariance.

Process disturbance auto-covariance	RMS
Q_1	0.35
Q_{10}	0.016
Q_{100}	0.025

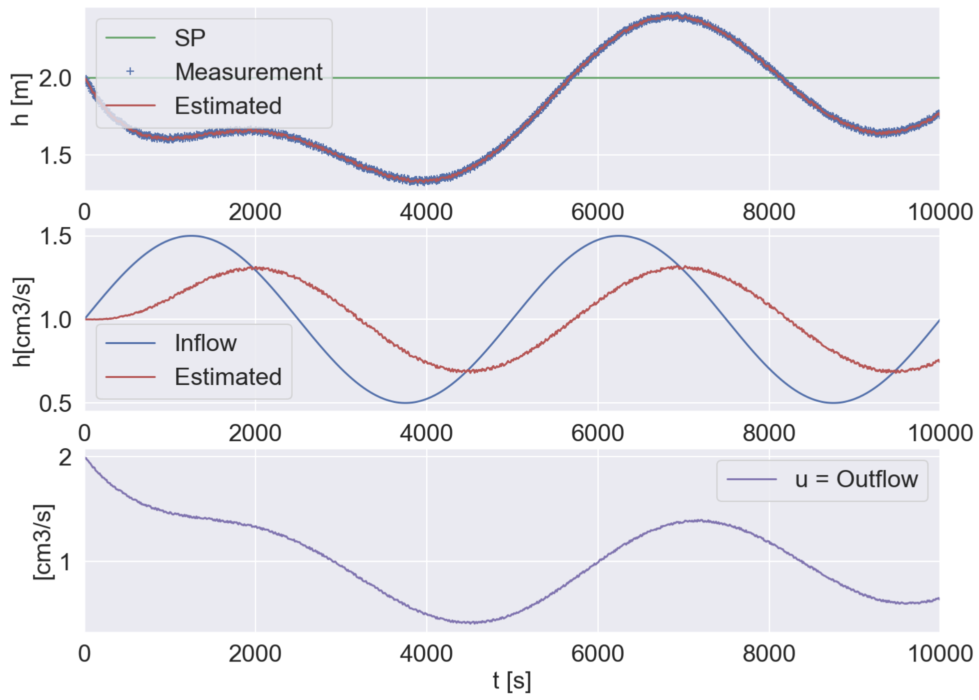


Figure 8: The Figure illustrates a simulated tank with PI level control and a Kalman Filter with process disturbance $Q_{22} = 1$.

accurately. Therefore, the RMS value between the real and estimated inflow has decreased. In Figure 10 we increased Q, even more, to observe if that resulted in a better prediction of the inflow. But that's not the case, even tho it manages to follow the same pattern as the inflow, the measurement noise also increases. Thus the RMS value increased compared to the simulation of $Q = 10$. The common thing for all the tuning is that we observe the output gets a sinusoidal shape, which implies that the valve will be closed and open more frequently, and as time passes, it will cause wear and tear on the valve.

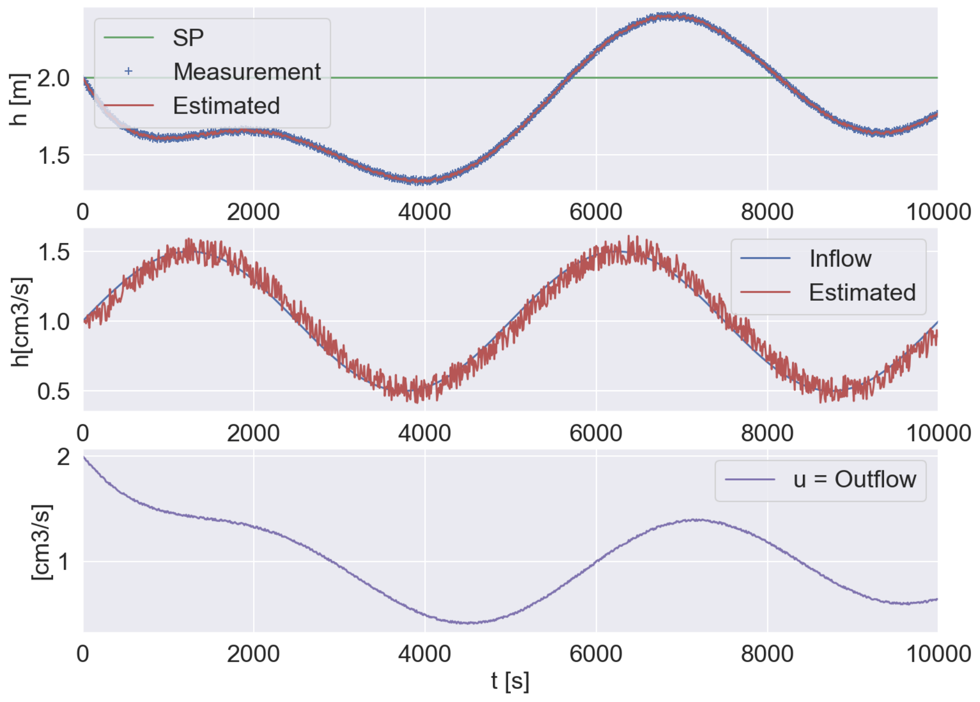


Figure 9: The Figure illustrates a simulated tank with PI level control and a Kalman Filter with process disturbance $Q_{22} = 10$.

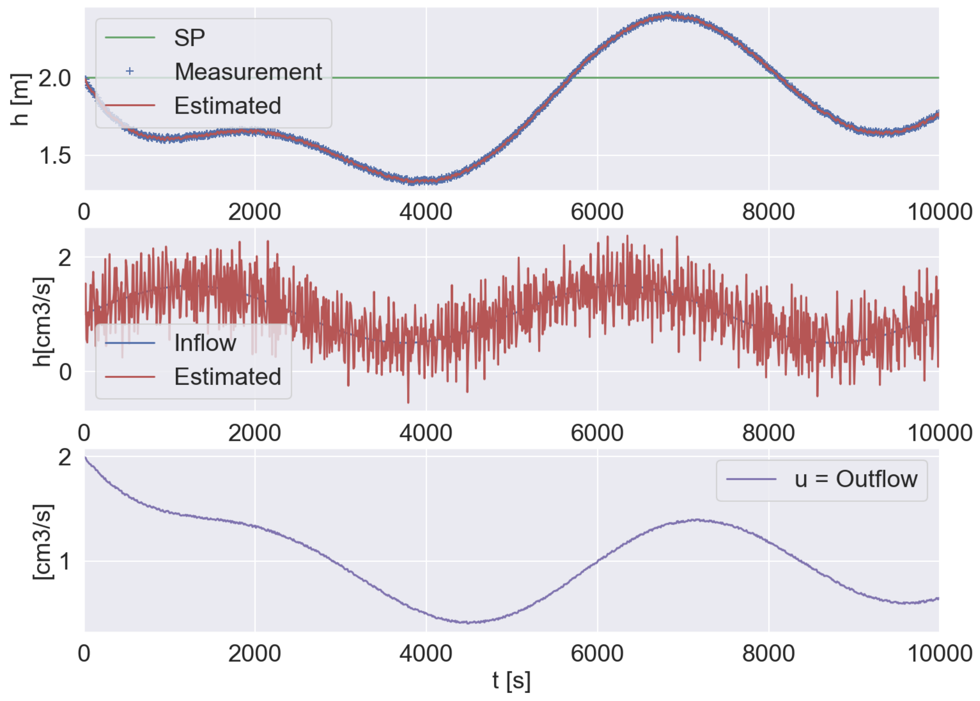


Figure 10: The Figure illustrates a simulated tank with PI level control and a Kalman Filter with process disturbance $Q_{22} = 100$.

5 Conclusion

The simulation results are reasonable, and the reason for that is because we used a linear model. But in reality, we know that the valve equation that represents the out-stream is nonlinear, and we simplified it by assuming it's linear. This is something one should have in mind, thus, a better filter is needed if we want to capture that aspect of the system.

One problem with the filter is that it assumes that the model is linear, and most physical system is nonlinear. This can be solved by implementing other state estimators like the Unscented Kalman filter, which is more accurate than the EKF but has a higher computational load. Another problem with the filter is that sometimes the proper initial value. Which is composed of two values the average value of $x_{p,0}$ and the auto-covariance $P_{p,k}$. This is needed to make the simulation converge, so knowledge of the system is preferred to make a proper initial value. Normally we have an intuition of the initial state of the tank. We may have insight if the tank is empty $x_{p,0} = 0$, or half empty $x_{p,0} = 50$. But the biggest problem is to reflect the physical model that it's representing, if the model deviates too much from the actual system (process) the solution won't reflect that system's evolution over time.

In this project, we assumed that the tank is on land and made the system linear. In the future, we should assume it is on a boat and that boat is rolling side by side. This is going to make the estimation problem significantly more difficult. Therefore, we also possibly need a more accurate estimator as the UKF.

References

[1] G. B. Greg Welch. An Introduction to the Kalman Filter. *Industrial &amp; Engineering Chemistry Research*, 7 2006.

[2] F. A. Haugen. Matematisk modellering. In *Dynamsike systemer*, pages 13–46. FAGBOKFORLAGET, 2016.

[3] F. A. Haugen. Introduction to automatic control. In *Modeling, Simulation and Control*, pages 22–49. FAGBOKFORLAGET, 2021.

[4] F. A. Haugen. PID Control. In *Modeling, Simulation and Control*, pages 252–276. FAGBOKFORLAGET, 2021.

[5] F. A. Haugen. State estimation with Kalman Filter. In *Modeling, Simulation and Control*, pages 561–587. FAGBOKFORLAGET, 2021.

[6] F. A. Haugen. Stochastic signals. In *Modeling, Simulation and Control*, pages 542–55. FAGBOKFORLAGET, 2021.

[7] R. R. . L. Jr. The g-h Filter. In *Kalman and Bayesian Filters in Python*, pages 17–49. 5 2020.

[8] D. Simon. Matematisk modellering. In *Optimal State Estimation*. John Wiley & Sons, Inc, 2016.

[9] S. Skogestad and C. Grimholt. The SIMC Method for Smooth PID Controller Tuning. In *PID Control in the Third Millennium*, pages 147–175. Springer, 2012.

A Code listing

```
# Importing the required modules
#-------------------------------------------------------------------------------
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
import seaborn as sns
sns.set_theme(style="darkgrid")
plt.rc('axes', labelsize=16)
plt.rc('xtick', labelsize=16)
plt.rc('ytick', labelsize=16)
plt.rc('font', size=16)


#-------------------------------------------------------------------------------

# Defining a function of PI controller:
def pi_controller(y_sp_k, y_m_k, u_man, u_i_km1,
                      Kc_LC, Ti_LC, Ts,u_min, u_max,
                      u_i_min, u_i_max):

    e_k = y_sp_k - y_m_k  # Control error
    u_p_k = Kc_LC*e_k     # P term

    # Anti windup by limiting integral term:
    u_i_k_tempor = u_i_km1 + (Kc_LC/Ti_LC)*Ts*e_k
    u_i_k        = np.clip(u_i_k_tempor, u_i_min, u_i_max)

    u_k_tempor   = u_man + u_p_k + u_i_k
    u_k          = np.clip(u_k_tempor, u_min, u_max)

    return (u_k, u_i_k)


# Defining a function for the PI tuning:
def pi_tuning(Ki, Tc):

    Kc = 1/(Ki*Tc)  # [(m3/s)/m] Controller gain
    Ti = 2*Tc       # [s] Integrator gain

    return (Kc, Ti)


# Time settings:
```

```
Ts        = 10     # Sim time−step [s]
t_start = 0.0    # [s]
t_stop  = 10000  # [s]
N_sim   = int((t_stop−t_start)/Ts) + 1

# Defining arrays for plotting:
t          = np.zeros(N_sim)
h          = np.zeros(N_sim)
h_meas     = np.zeros(N_sim)
h_sp       = np.zeros(N_sim)
h_est      = np.zeros(N_sim)
u          = np.zeros(N_sim)
q_in       = np.zeros(N_sim)
q_in_est = np.zeros(N_sim)
wave       = 0.5∗np.sin(np.linspace(−2∗np.pi, 2∗np.pi, N_sim)) + 1 # q_in i

# Model parameters:
A_area    = 1000   # [m2]
h_measin = 0       # [m] Max level
h_measax = 4       # [m] Min level

# Level meas noise param:
ampl_h_meas_noise = 0.01 # [m] Ampl of unif random meas noise

# Controller parameters:
Tc         = 1000         # [s] Specified closed loop time constant
Ki         = −1/A_area   # Process integrator gain
(Kc, Ti) = pi_tuning(Ki, Tc)
u_man      = 1  # [m3/s]
u_min      = 0  # [m3/s]
u_max      = 8  # [m3/s]
u_i_min    = −8 # [m3/s]
u_i_max    = 8  # [m3/s]

# Initialization of Kalman Filter:
h_pred_k     = 2.0  # [m] # Initial state
q_in_pred_k = 1.0  # [m3/s]
n_x           = 2    # Number of states

# Settings of Kalman Filter:
P_pred_k          = np.diag([1.0∗∗2, 1.0∗∗2])
std_Q_22          = 10∗∗1 # Tuning factor
Q                 = np.diag([1.0∗∗2, std_Q_22∗∗2])
std_h_meas_noise = ampl_h_meas_noise/np.sqrt(3)  # [m]
R                 = np.diag([std_h_meas_noise∗∗2])
```

```
# Initilizing the states:
h_k      = 2     # [m]
u_i_km1 = 0.0   # [m3/s]

# For−loop for simulation:
for k in range(0, N_sim):

    t_k = k*Ts

    # Selecting Inputs:
    if (t_k < 4000):
        h_sp_k = 2
        q_in_k = wave[k]
    elif (t_k >= 4000):
        h_sp_k = 2
        q_in_k = wave[k]

    # Adding uniformly distributed meas noise:
    h_meas_noise_k = np.random.uniform(−ampl_h_meas_noise,
                                        ampl_h_meas_noise, 1)[0]
    h_meas_k = h_k + h_meas_noise_k


    # Level PI controller:
    (u_k, u_i_k) = pi_controller(
        h_sp_k, h_meas_k,
        u_man, u_i_km1,
        Kc, Ti, Ts,
        u_min, u_max,
        u_i_min, u_i_max)

    # Kalman Filter for estim h and q_in using meas of h:

    # Matrices in linearized model:
    A_cont = np.array([[0, 1/A_area], [0, 0]])
    C_cont = np.array([[1, 0]])
    A = A_disc = np.eye(n_x) + Ts*A_cont
    C = C_disc = C_cont

    # Calculating the Kalman gain:
    K_k = ((P_pred_k @ C.T)
           @ (np.linalg.inv(C @ P_pred_k @ (C.T) + R)))

    # Calculating the innovation process:
```

```
    e_innov_k = h_meas_k - h_pred_k

    # Meas-corrected estimates are used as applied estim:
    h_corr_k = h_pred_k + K_k[0, 0]*e_innov_k
    q_in_corr_k = q_in_pred_k + K_k[1, 0]*e_innov_k

    # Predicted estimate:
    dh_corr_dt_k = (1/A_area)*(q_in_corr_k - u_k)
    dq_in_corr_dt_k = 0
    h_pred_kp1 = h_corr_k + Ts*dh_corr_dt_k
    q_in_pred_kp1 = (q_in_corr_k + Ts*dq_in_corr_dt_k)

    # Auto-covariance of error of meas-corrected estimate:
    P_corr_k = (np.eye(n_x) - K_k @ C) @ P_pred_k

    # Auto-covariance of error of predicted estimate:
    P_pred_kp1 = A @ P_corr_k @ (A.T) + Q

    # Process simulation:
    dh_dt_k       = (1/A_area)*(q_in_k - u_k)
    h_kp1_tempor = h_k + dh_dt_k*Ts
    h_kp1         = np.clip(h_kp1_tempor, h_measin, h_measax)

    # Defining the arrays for plotting:
    t[k]          = t_k
    h_sp[k]       = h_sp_k
    h[k]          = h_k
    h_meas[k]     = h_meas_k
    h_est[k]      = h_corr_k
    u[k]          = u_k
    q_in[k]       = q_in_k
    q_in_est[k] = q_in_corr_k

    # Index-shift to prepare for the next iteration:
    u_i_km1       = u_i_k
    h_k           = h_kp1
    h_pred_k      = h_pred_kp1
    q_in_pred_k = q_in_pred_kp1
    P_pred_k      = P_pred_kp1

# Calculating the root mean squared error
print(f"""The root mean squared error is: {np.sqrt((np.sum(q_in_est - q_i
**2)/len(q_in_est))}""")

# Plotting:
```

```
plt.close("all")
plt.figure(num=1, figsize=(12, 9))

print(h_meas)
plt.subplot(3, 1, 1)
plt.plot(t, h_sp, 'g')
plt.plot(t, h_meas, 'b+')
plt.plot(t, h_est, 'r')
plt.grid('minor')
plt.xlim(t_start, t_stop)
plt.ylabel('h [m]')
plt.legend(('SP', 'Measurement', 'Estimated'))

plt.subplot(3, 1, 2)
plt.grid('minor')
plt.xlim(t_start, t_stop)
plt.ylabel('h[cm3/s]')
plt.plot(t, q_in, 'b')
plt.plot(t, q_in_est, 'r')
plt.legend(('Inflow', 'Estimated'))

plt.subplot(3, 1, 3)
plt.grid('minor')
plt.xlim(t_start, t_stop)
plt.ylabel('[cm3/s]')
plt.plot(t, u, 'm')
plt.legend(('u = Outflow',))
plt.xlabel('t [s]')

plt.savefig('KF.png')
plt.show();
```