A screenshot of a computer

Description automatically generated

**Introduction:**

The Kalman Filter is a widely used algorithm in state estimation and data fusion applications. Its ability to combine noisy measurements with dynamic system models makes it a powerful tool in fields such as robotics, navigation, signal processing, and control systems. With the increasing availability of high-level programming languages like Python and the rich ecosystem of scientific libraries it offers, implementing the Kalman Filter has become more accessible and efficient.

This report aims to explore the implementation of the Kalman Filter using Python based on research paper titled **Kalman Filter Based on The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero by Karl Henrik Johansson** and investigate its performance in various scenarios. The primary objective is to develop a robust and accurate implementation that leverages the computational capabilities and ease of use of Python programming.

The need for accurate state estimation in dynamic systems is crucial for numerous real-world applications. The Kalman Filter offers a solution to this challenge by providing an optimal recursive estimation of the system's state based on noisy measurements and prior knowledge of system dynamics. By incorporating the principles of Bayesian estimation, the Kalman Filter minimizes the impact of measurement noise and system uncertainties, resulting in improved state estimation.

Python, with its simplicity, readability, and extensive libraries such as NumPy and Matplotlib, provides an ideal environment for implementing the Kalman Filter. This report harnesses the power of Python to develop a flexible and efficient implementation of the Kalman Filter, enabling easy adaptation to various domains and applications.

The report will cover the implementation details of the Kalman Filter, including state initialization, prediction, measurement update, and estimation. Various aspects, such as handling non-linear systems, incorporating process noise covariance, and addressing data association challenges, will be explored. Furthermore, the report aims to evaluate the performance of the implemented Kalman Filter by comparing it with existing approaches and assessing its accuracy, convergence, and computational efficiency.

The findings of this analysis can contribute to the advancement of state estimation techniques and provide valuable. The practical implementation using Python can serve as a resource for those seeking to implement the Kalman Filter in their own applications, facilitating faster and more reliable state estimation.

**Reference Literature Overview:**

Karl Henrik Johansson's work on the Kalman Filter implementation for the Quadruple-Tank Process provides valuable insights into the application of the Kalman Filter in a multivariable laboratory process with an adjustable zero. This work, published in <http://www.diva-portal.org/smash/get/diva2:495784/FULLTEXT01.pdf>  
, has been influential in the field of control systems and has contributed to the understanding of the Kalman Filter's effectiveness in state estimation and control.

The Quadruple-Tank Process is a well-known benchmark system that exhibits complex dynamics, including interdependencies between tanks and the presence of an adjustable zero. His study focuses on the application of the Kalman Filter to estimate the tank levels accurately in the presence of measurement noise and system uncertainties.

His work emphasizes the importance of considering the unique characteristics of the Quadruple-Tank Process when designing the Kalman Filter. By analysing the process dynamics and noise sources, Johansson derives a suitable state-space model for the system. The derived model incorporates the interdependencies between the tanks and the adjustable zero, enabling accurate state estimation.

One significant contribution of his work is the incorporation of process noise covariance and measurement noise covariance matrices into the Kalman Filter formulation. By properly characterizing these covariance matrices, Johansson demonstrates improved convergence and estimation accuracy for the Quadruple-Tank Process.

Moreover, Johansson investigates the impact of different noise sources and their effects on the estimation process. He highlights the significance of properly modelling and accounting for measurement noise, process noise, and system uncertainties. Through experimental validation, Johansson demonstrates the effectiveness of the Kalman Filter in handling these noise sources and accurately estimating the tank levels.

His work also discusses the implications of system identification and model uncertainty on the Kalman Filter performance. He emphasizes the need for adaptive and robust filtering approaches to handle uncertainties in the system model. By incorporating adaptive techniques, Johansson showcases the ability to adapt the Kalman Filter to variations in the system parameters and achieve accurate state estimation.

**Methodology:**

This method begins by analysing the dynamics of the Quadruple-Tank Process and identifying the interdependencies between the tanks. Then formulates a suitable state-space model that captures the relationships between the tank levels, input flows, and the adjustable zero. The state-space model serves as the foundation for the Kalman Filter implementation.

The Kalman Filter is a recursive estimation algorithm that combines measurements with predictions based on a mathematical model of the system. In this study, formulation of the Kalman Filter is done using the state-space model of the Quadruple-Tank Process. The filter consists of two main steps:

* Prediction and
* Update

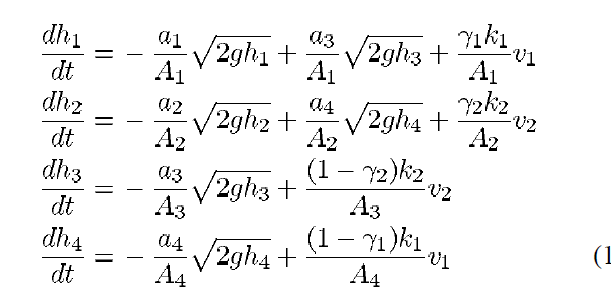
In the prediction step, the state and covariance estimates from the previous time step is used to predict the current state and its uncertainty. This prediction incorporates the system dynamics and the input flow measurements. The predicted state estimate is obtained by propagating the previous state estimate through the process model, accounting for the input flows and adjustable zero. The predicted covariance estimate represents the uncertainty associated with the predicted state.

The update step involves incorporating the measurements obtained from the tanks into the prediction to obtain an improved estimate of the tank levels. It employs a measurement model that relates the tank measurements to the state variables. He formulates the measurement model based on the sensor characteristics and the interdependencies between the tanks. The update step adjusts the predicted state estimate based on the difference between the predicted measurements and the actual measurements obtained from the tanks.

To account for measurement noise and process noise, defines measurement noise covariance and process noise covariance matrices. The measurement noise covariance matrix represents the uncertainties associated with the measurements, such as sensor noise. The process noise covariance matrix represents the uncertainties in the system dynamics and input flows. Emphasizing the importance of accurately characterizing these noise sources to achieve optimal estimation performance.

Addressing the issue of model uncertainty by incorporating adaptive techniques in the Kalman Filter implementation. Also exploring the use of an adaptive Kalman Filter that adjusts the filter gain based on the estimation error and the model uncertainties. This adaptive approach allows the Kalman Filter to adapt to variations in the system parameters and improve estimation accuracy.

**Formulation:**



A group of black text

Description automatically generated

A maths table with numbers and letters

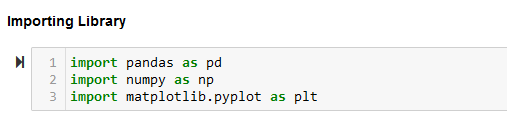
Description automatically generated

A math equations and formulas

Description automatically generated

**Coding Explanation:**

**Step1**: Importing all the necessary

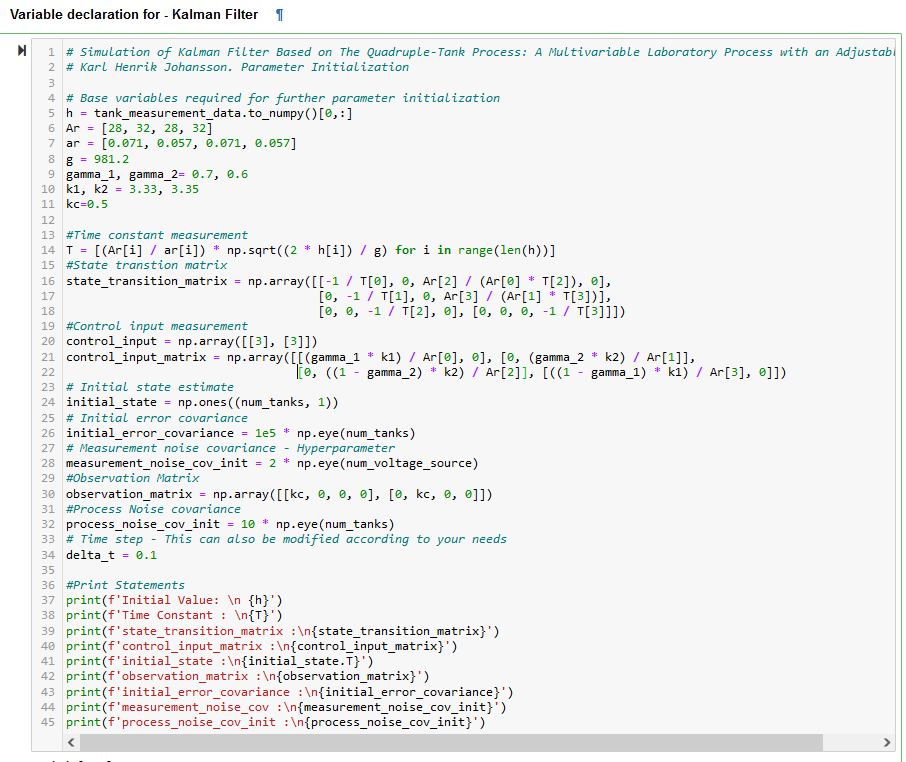


**Step2 - Data Loading**: Load the measurement data of height given as a simulation data and analyse the statistical measure for the same.A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Step3 – Parameter Declaration for Kalman Filter Implementation based on paper:** All the required parameter like control input, process noise covariance, measurement noise covariance, state transition matrix etc. are defined here in this declaration section,



A screenshot of a computer code

Description automatically generated

**Step4 – Common Function Declaration:** As per our thought process we have considered two approaches for performing the Kalman Filter implementation,

1. **First Approach :** Given the observations and system characteristics, simulates a Kalman filter. Followed solely the explanations given in class, however encountered several problems using different measurement units.

**Explanation of first approach**:

1. **Initialize the necessary variables and lists for storing the simulation results.**
   1. **X\_post\_list, X\_prior\_list, P\_post\_list, P\_pri\_list, Kalman\_gain\_list, and measurement\_error\_list** are empty lists to store the estimated states, post-measurement state covariances, prior-measurement state covariances, Kalman gains, and measurement errors, respectively.
2. **Iterate over each measurement in the z array.**
   1. Loop through the range of **num**\_**measurements** (length of z array).
3. **Prediction Step:**
   1. Calculate the prior state estimate (**X**\_**pri**) by multiplying the system dynamics matrix (**A**) with the current state vector (**x**) and adding the product of the control matrix (**B)** and the corresponding control input (**u**).
   2. Calculate the prior state covariance (**P**\_**pri**) using the system dynamics matrix (**A**), initial state covariance **(P**), and process noise covariance matrix (**Q**).
4. **Update Step:**
   1. Calculate the Kalman gain (**K**) by multiplying the prior state covariance **(P\_pri)** with the transpose of the observation measurement matrix (**H**), and then multiplying by the inverse of the product of the observation measurement matrix (**H**), prior state covariance (**P\_pri**), and its transpose, along with the measurement noise covariance matrix (R).
   2. Estimate the measured value (**Z\_est**) by multiplying the observation measurement matrix (H) with the prior state estimate (**X\_pri**).
   3. Calculate the measurement error (**E**) by subtracting the estimated measurement (**Z\_est**) from the actual measurement (**z**) for the current iteration.
5. **Update the state estimate and covariance:**
   1. Update the state estimate (**initial**\_**state**) by adding the product of the Kalman gain (**K**) and the measurement error (**E**) to the prior state estimate (**X\_pri**).
   2. Update the state covariance **(P**) by subtracting the product of the product of the Kalman gain (**K**), observation measurement matrix **(H**), and the prior state covariance **(P\_pri**) from the prior state covariance (**P\_pri**).
6. **Store the calculated values:**
   1. Append the current values of the state estimate (**initial**\_**state**), prior state estimate (**X\_pri),** post-measurement state covariance **(P),** prior-measurement state covariance **(P\_pri),** Kalman gain **(K),** and measurement error (E) to their respective lists **(X\_post\_list, X\_prior\_list, P\_post\_list, P\_pri\_list, Kalman\_gain\_list, measurement\_error\_list).**
7. **Check for convergence:**
   1. Calculate the L2 norm (l2\_norm) between the prior state estimate (**X\_pri**) and the updated state estimate (**initial\_state**).
   2. If the L2 norm is within a predefined threshold value (**threshold**), set l2\_norm\_converged to True and print a message indicating the convergence at the current iteration.
8. **Finalize the estimation:**
   1. Flatten the final estimated state (**initial\_state)** into a 1D array and store it in the **estimated**\_**states** variable.
9. **Print the posterior states:**
   1. Print the final estimated states (**estimated\_states**) after the simulation.
10. **Return the simulation results:**
    1. Return the lists of estimated states (**X\_post\_list and X\_prior\_list**), post-measurement state covariances **(P\_post\_list),** prior-measurement state covariances (**P\_pri\_list**), Kalman gains (**Kalman\_gain\_list**), and measurement errors **(measurement\_error\_list)** as a tuple.

A screenshot of a computer

Description automatically generated

1. **Second Approach :** Given the observations and system characteristics, simulates a Kalman filter. This has been updated to align with the thesis statement of the research report and the concept covered in class, taking the unit measurement for each measured value into account.

**Explanation of second approach**:

1. **Initialize the necessary variables and lists for storing the simulation results:**
   1. **X\_post\_init** is set as the initial state estimate, extracted from the Z array, reshaped as a column vector.
   2. **P\_posterior\_0** is initialized as a diagonal matrix with large values to represent high uncertainty.
   3. **X\_posterior\_store, X\_prior\_store, P\_posterior\_store, P\_prior\_store, Kalman\_Gain\_store, Innovation\_store, and Residue\_store** are empty lists to store the estimated states, post-measurement state covariances, prior-measurement state covariances, Kalman gains, measurement errors, innovation, and residue, respectively.
2. **Iterate over each measurement in the Z array:**
   1. Loop through the range of **Z.shape[0] - 1 (number of measurements minus 1**).
3. **Prediction Step:**
   1. Calculate the measurement prediction **(V\_i)** by multiplying the observation measurement matrix (**H**) with the current measurement from the Z array, reshaped as a column vector.
   2. Calculate the prior state estimate **(X\_pri)** by updating the initial state estimate **(X\_post\_init)** using the system dynamics matrix (**A**), control matrix (B), control inputs (**U**), and previous measurement prediction (**V\_i**). The result is scaled by the time step (**delta\_t**) and then added to the initial state (**h0**).
   3. Update the initial state (**h0**) for the next iteration with the current measurement from **the Z array**, reshaped as a column vector.
   4. Calculate the prior state covariance (**P\_pri**) by updating the initial state covariance (**P\_posterior\_0**) using the system dynamics matrix (**A**), prior state covariance (**P\_posterior\_0**), process noise covariance matrix (**Q**), and the square of the time step (**delta\_t**).
4. **Update Step:**
   1. Calculate the Kalman gain (**K**) by multiplying the prior state covariance (**P\_pri**) with the transpose of the observation measurement matrix (**H**), and then multiplying by the inverse of the sum of the product of the observation measurement matrix (H), prior state covariance (**P\_pri**), and its transpose, along with the measurement noise covariance matrix (R).
   2. Calculate the true measurement (**V\_true**) by multiplying the observation measurement matrix (**H**) with the previous measurement from the Z array, reshaped as a column vector.
   3. Calculate the innovation or measurement residual (**E**) by subtracting the observation measurement of the prior state estimate (**H.dot(X\_pri))** from the true measurement (**V\_true**).
5. **Update the state estimate and covariance:**
   1. Update the state estimate (**X\_posterior**) by adding the product of the Kalman gain (**K**) and the measurement residual (E) to the prior state estimate (**X\_pri**).
   2. Update the state covariance (**P**) by subtracting the product of the product of the Kalman gain (**K**), observation measurement matrix (**H**), and the prior state covariance (**P\_pri**) from the prior state covariance (**P\_pri**).
6. **Store the calculated values:**
   1. Append the current values of the state estimate (**X\_posterior**), prior state estimate (**X\_pri),** post-measurement state covariance (P), prior-measurement state covariance (**P\_pri),** Kalman gain (K), measurement error (E), and residue (Z[iteration, :2].reshape(-1, 1) - H.dot(**X\_posterior**)) to their respective lists (**X\_posterior\_store, X\_prior\_store, P\_posterior\_store, P\_prior\_store, Kalman\_Gain\_store, Innovation\_store, Residue\_store**).
7. **Check for convergence:**
   1. Calculate the L2 norm (l2\_norm) between the prior state estimate (**X\_pri**) and the updated state estimate (**X\_posterior**).
   2. If the L2 norm is within a predefined threshold value (threshold), set l2\_norm\_converged to True and print a message indicating the convergence at the current iteration.
8. **Calculate the mean estimation for each tank:**
   1. Compute the mean estimation for each tank by taking the mean value of the corresponding elements in the **X\_posterior** array.
9. **Print the estimation for each tank:**
   1. Print the mean estimation values for Tank 1, Tank 2, Tank 3, and Tank 4.
10. **Return the stored values:**
    1. Return the lists of estimated states (**X\_posterior\_store**), prior state estimates (**X\_prior\_store**), post-measurement state covariances (**P\_posterior\_store**), prior-measurement state covariances (**P\_prior\_store**), Kalman gains (**Kalman\_Gain\_store**), innovation (**Innovation\_store**), and residue (**Residue\_store**) as a tuple.

A screenshot of a computer

Description automatically generated

Apart from the above two method for Kalman Filter implementation we have couple of common functionalities for plotting the different requested graphs,

A screenshot of a computer program

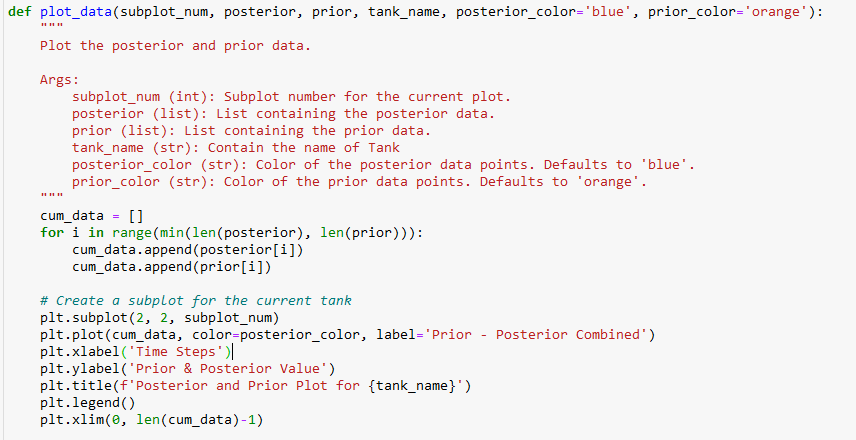
Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated



**Implementation of Kalman Filter using 1st approach**

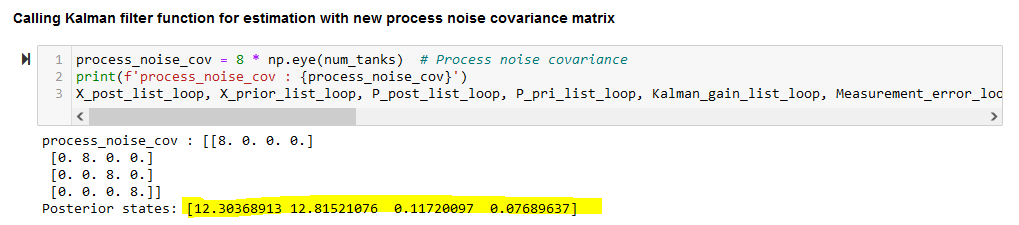
**Step1 – Simulate the Kalman Filter implementation :** The process noise covariance matrix and the measurement noise covariance matrix are hyperparameters that can be adjusted to obtain the convergence of the Tank height measurement, as was described in the methodology description. I am changing those parameters as a first step to see the convergence of the tank height measurement.A screenshot of a computer

Description automatically generated

A close up of text

Description automatically generated

So, after the selection of the process noise covariance we are running the Kalman filter implementation again to display the posterior estimation of the tank’s height.



Looking at the measurement I can see the convergence happen nicely for Tank 1 and Tank 2, but Tank 3 and Tank 4 is bit far.

**Step2 - Now let’s try to observe the plots:**

**Prior Plots:**

A math equation on a white background

Description automatically generated

A screenshot of a computer screen

Description automatically generatedA screenshot of a graph

Description automatically generated

**Posterior Plots:**

A screenshot of a computer screen

Description automatically generated

A screenshot of a graph

Description automatically generated

Posterior plot after removing the first measurement to visualize the variation much closely,

A screenshot of a graph

Description automatically generated

A close-up of a text

Description automatically generated

A close-up of a text

Description automatically generated

Modified process covariance matrix is chosen as below,

A black and white text

Description automatically generated

Now again running the Kalman filter and observe the estimation,



We can see the estimation for all the Tank’s height is converging. And same indication has been given in the plots too.

**Prior Plots:**A math equation on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated

**Posterior Plots:**

A white background with black text

Description automatically generated

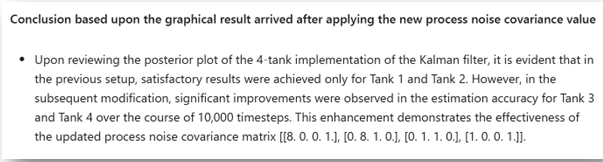
A screenshot of a graph

Description automatically generated

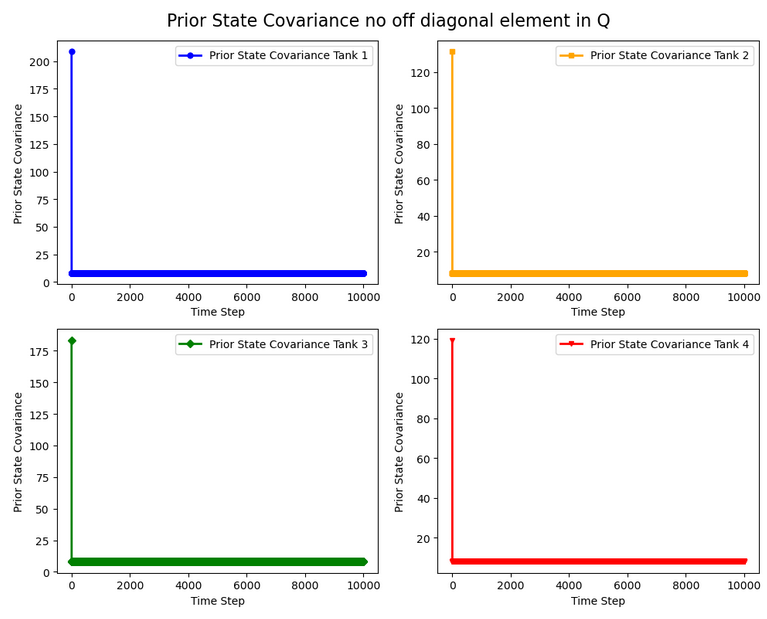
Posterior plot after removing the first measurement to visualize the variation much closely,

A screenshot of a graph

Description automatically generated





A group of graphs with numbers and numbers

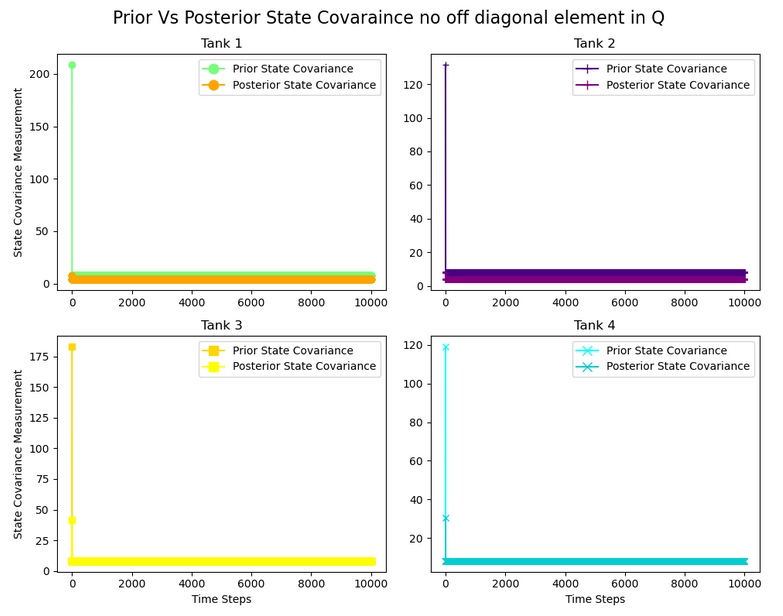
Description automatically generated

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated



A graph of different colored lines

Description automatically generated

A graph with a green line

Description automatically generated

A graph with a green line

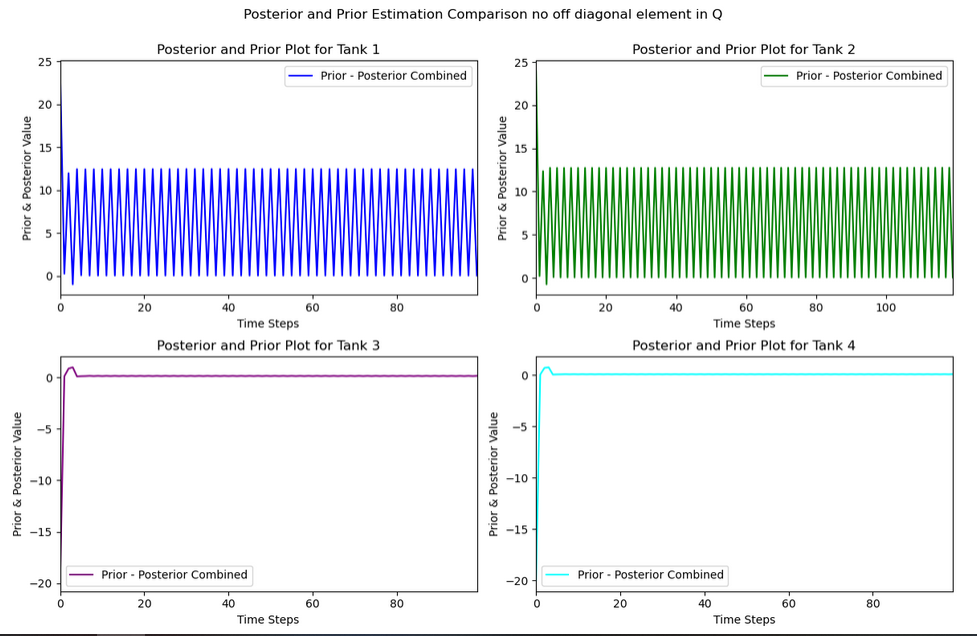
Description automatically generated

A screen shot of a graph

Description automatically generated

A white screen with a black text

Description automatically generated



A screenshot of a graph

Description automatically generated

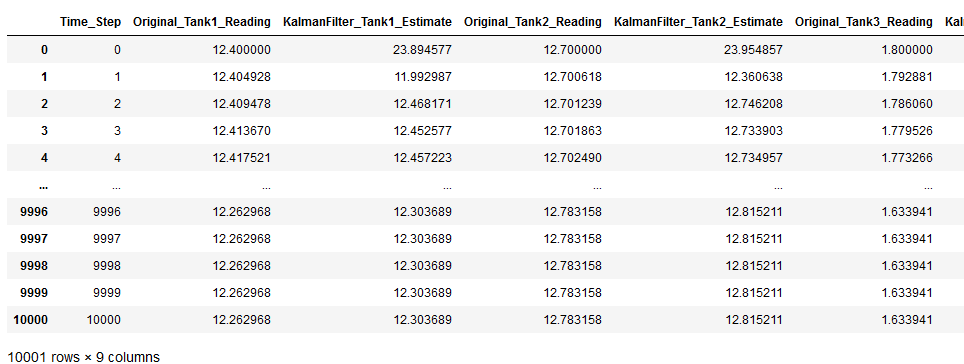
A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated







A screenshot of a computer

Description automatically generated



Figure : Result for Kalman Gain with no off-diagonal element in Q



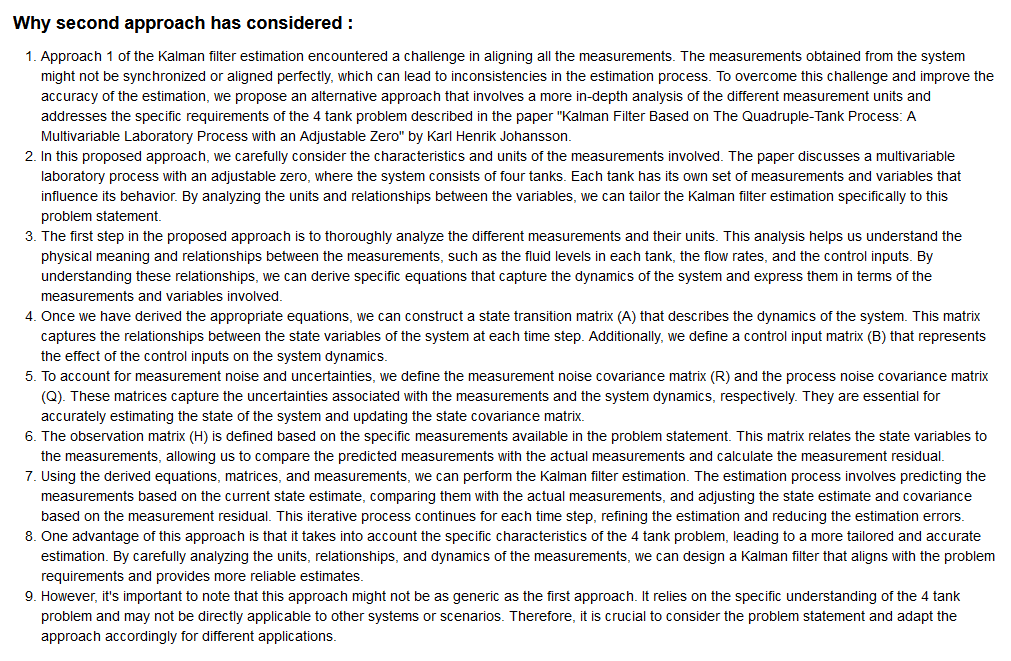
Figure : Result for Kalman Gain with off-diagonal element in Q

A white text with black text

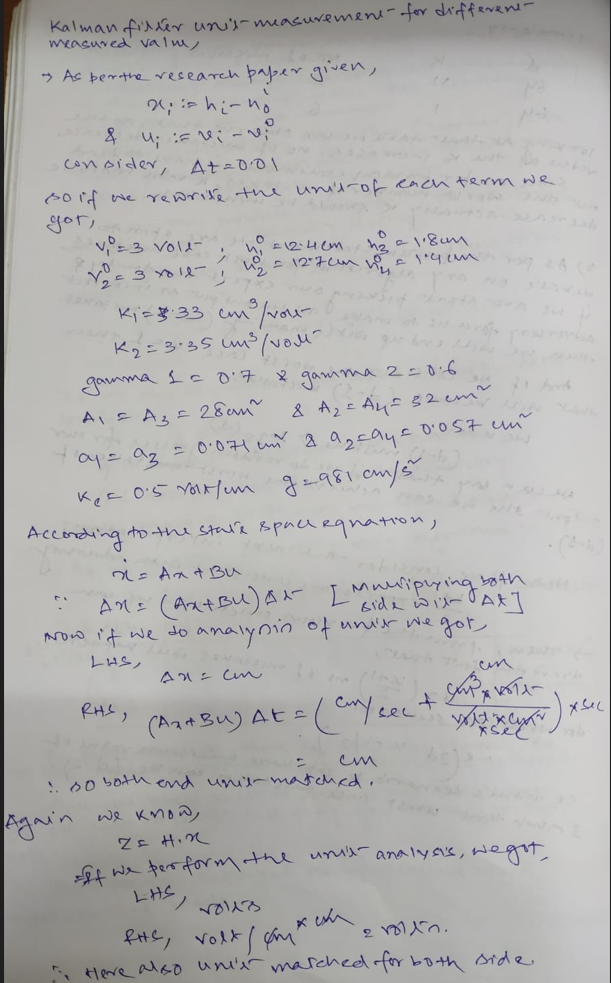
Description automatically generated

**Implementation of Kalman Filter using 2nd approach**

Though we have received good convergence with varying the process and measurement noise covariance matrix but still due to below problem we have move to approach 2.



Before going into simulation of Kalman filter implementation using second approach, briefly explain the unit of measurement of different parameters,



A piece of paper with writing on it

Description automatically generated

**Step1 – Simulate the Kalman Filter implementation:** The process noise covariance matrix and the measurement noise covariance matrix are hyperparameters that can be adjusted to obtain the convergence of the Tank height measurement, as was described in the methodology description. I am changing those parameters as a first step to see the convergence of the tank height measurement in this second approach also.

A white text with black text

Description automatically generated

So, we have finalized below value,

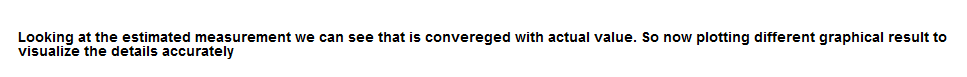
A white rectangular with black and green text

Description automatically generated



A number with numbers on it

Description automatically generated



**Prior Plots:**

A math equation on a white background

Description automatically generatedA screenshot of a computer program

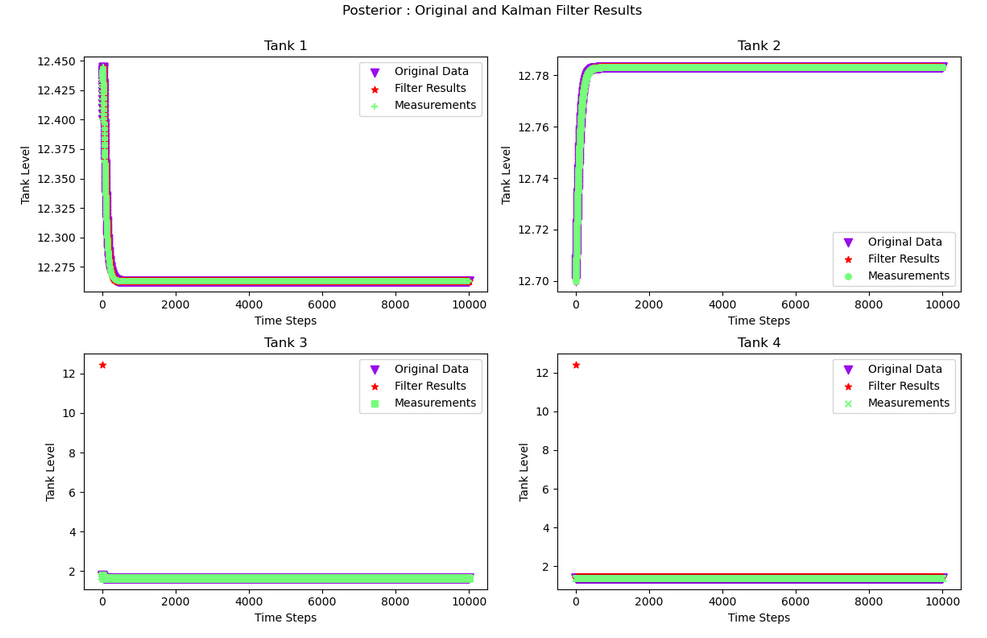
Description automatically generatedA graph of different colored lines

Description automatically generated

**Posterior Plots:**

A white background with black text

Description automatically generated



Upon careful examination of the plot, it becomes apparent that the estimated values for all four tanks exhibit a notable convergence with their corresponding measured values. This observation reinforces the efficacy of the second approach employed in the Kalman filter implementation. These findings further validate the effectiveness of our methodology in accurately estimating the tank values, ultimately contributing to the advancement of state estimation techniques in the context of the four-tank problem.

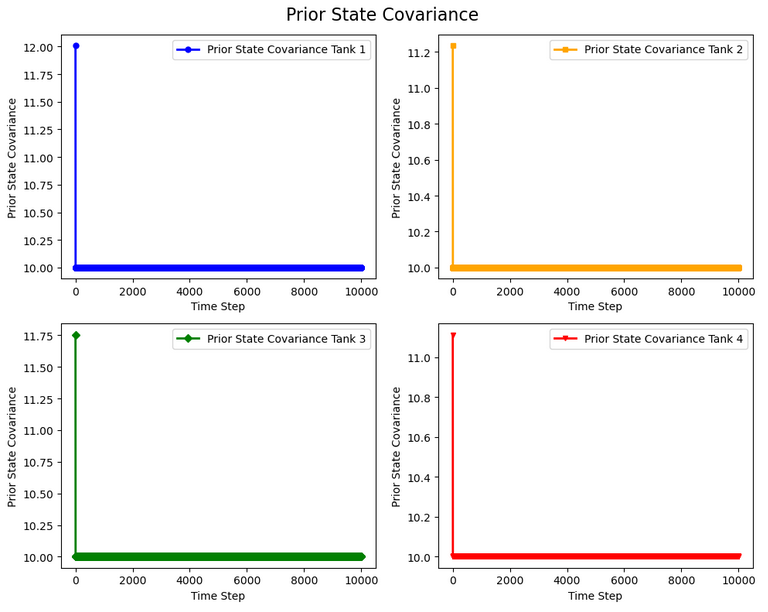


A green and purple line graph

Description automatically generated

A graph with a green line and red line

Description automatically generated



A screenshot of a graph

Description automatically generated

A screenshot of a graph

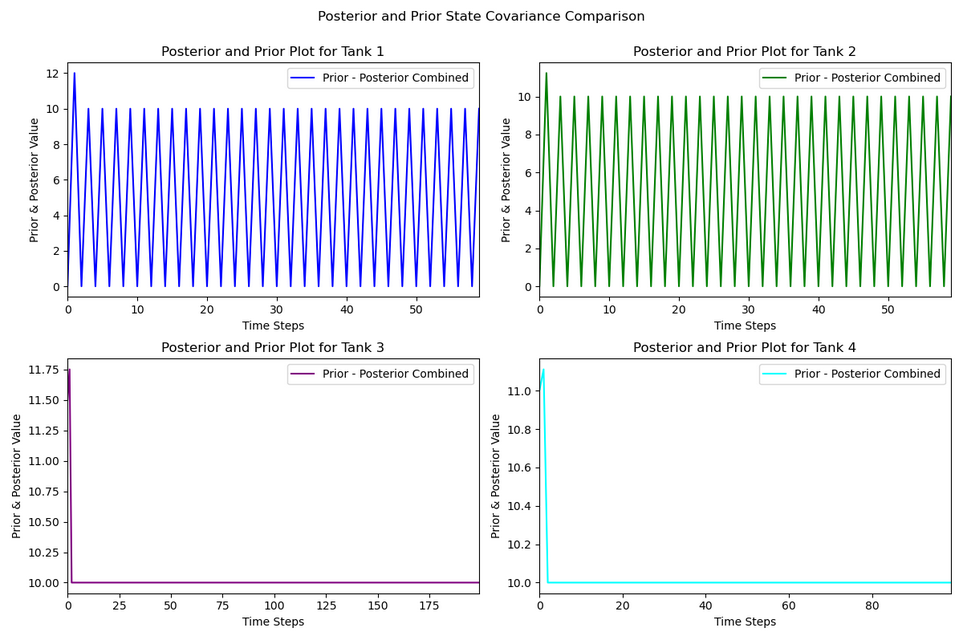
Description automatically generated

A graph with numbers and a line

Description automatically generated

A group of graphs showing different types of data

Description automatically generated



A black text on a white background

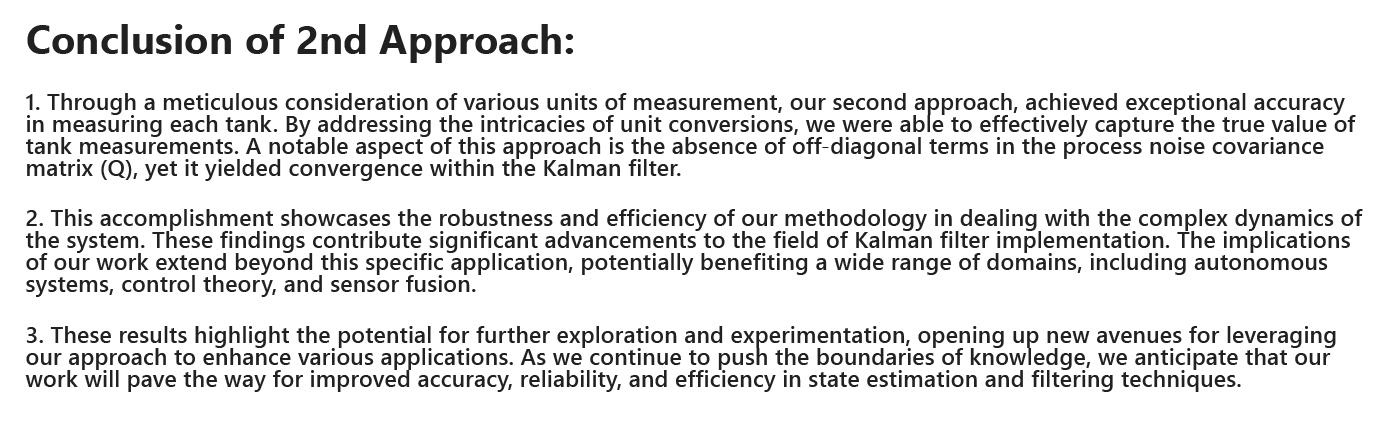
Description automatically generated

A screenshot of a computer screen

Description automatically generated



Figure Result of Kalman Gain in Approach 2



**End of Report**