Giorgio Mendoza

RBE595-S24-S01

Dr. Brodovsky

Particle Filter Report

Task 1:

Objective: Task 1 involves the development of a particle filter that enables a quadrotor's state estimation through predictive modeling and measurement updating. The particle filter's role is to accurately forecast the quadrotor's future state using the current state, control inputs, and noise models.

1. **State Prediction:** The prediction step of the particle filter evolves the state of each particle using the process model defined by the differential equations:

$$\dot{\mathbf{x}} = \mathbf{G}(\mathbf{q})^{-1}\boldsymbol{\omega} + \mathbf{g} + \mathbf{R}(\mathbf{q})\mathbf{a}$$

- Where:
- **ẋ** is the state derivative vector,
- **G(q)** is the rotation matrix transforming angular velocities from the world frame to the body frame,
- **ω** represents angular velocities,
- **g** denotes the gravitational acceleration,
- **R(q)** is the rotation matrix from the body frame to the world frame,
- **a** stands for linear accelerations.

2. **State Integration:** The new state is obtained by integrating the rate of change over time, Δt, leading to the discrete update equation below. This step also includes the application of noise directly to the state vector, reflecting both process noise and measurement noise characteristics.

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}\Delta t$$

3. **Measurement Update:** After receiving new measurements, the state estimate is refined using the observed data. A measurement matrix C maps the true state space into the measured space. The covariance matrix associated with this process encapsulates the sensor noise characteristics.

4. **Weight Calculation:** The weights of the particles are calculated using a Gaussian likelihood function, reflecting the error between the predicted and observed states. The weight of each particle is updated according to:

$$W_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_i - \mathbf{x}_{\text{obs}})^2}{\sigma^2}\right)$$

5. **Particle Resampling:** A resampling step follows to focus computational resources on particles with higher weights. This ensures diversity among particles and avoids degeneracy. Particles are resampled proportional to their weights, and the particle set is regenerated, concentrating on the more likely states.

The first plot below shows the quadrotor's trajectory in three-dimensional space. Each dot represents the quadrotor's position at a different time, with the color coding indicating ground truth versus the particle filter's estimates. The closeness of the two sets of points indicates how well the particle filter's estimates align with the actual trajectory. It looks like there's a decent match, although some divergence is evident, which is normal in such estimations due to noise and other uncertainties.
The second set of plots represents the yaw, pitch, and roll orientation angles over time. These plots compare the ground truth data with the estimated values from the particle filter. A few observations can be made from these plots:

- Yaw: The estimation seems to follow the ground truth closely, with the estimated values almost overlapping the true values. This indicates that the particle filter is accurately tracking the yaw angle of the quadrotor over time.
- Pitch: Similar to yaw, the pitch estimation is quite close to the ground truth. There are areas where the estimate deviates slightly, which could be due to the inherent noise in the sensor data or the approximation of the particle filter.
- Roll: The roll estimation also shows a good overlap with the ground truth. Some discrepancies are visible, which, like in pitch, may result from various error sources.

The number particles used for each mat file wasn't the same. After running the filter multiple taps on every mat file, it was observed that datasets with lower data points performed better when the number of particles were increased. On the other hand, datasets with lots of data points performed better with a lower number of particles.

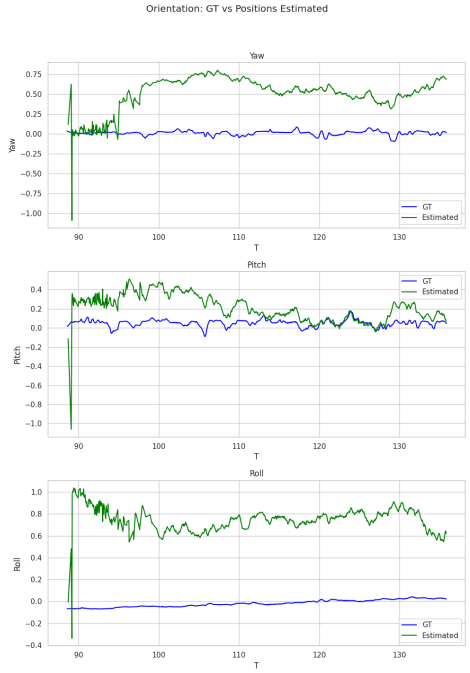

Figure 1: MatFile0 Orientation Plots

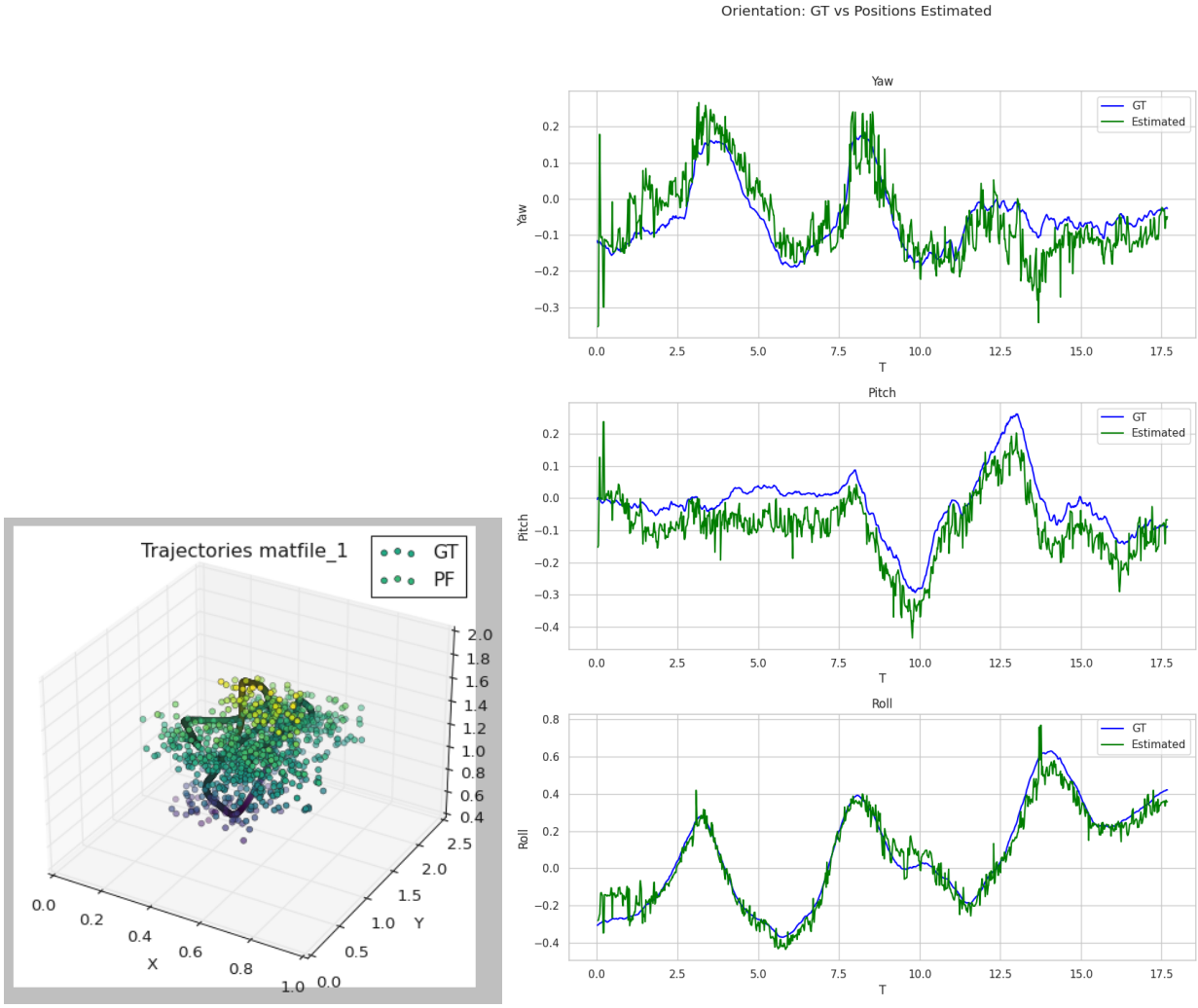

Figure 2, 3: MatFile1 Trajectories & Orientation Plots

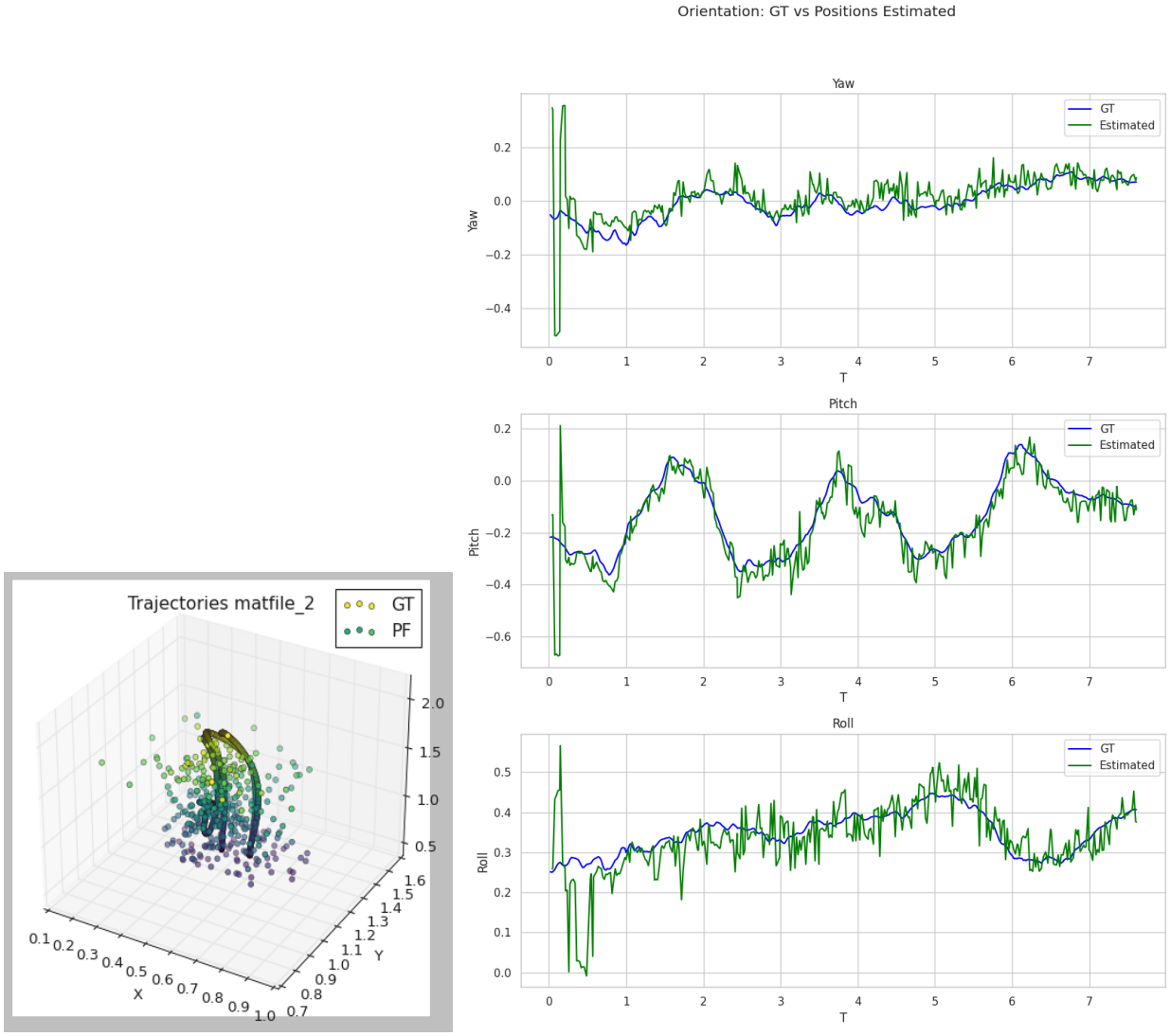

Figure 4, 5: MatFile2 Trajectories & Orientation Plots

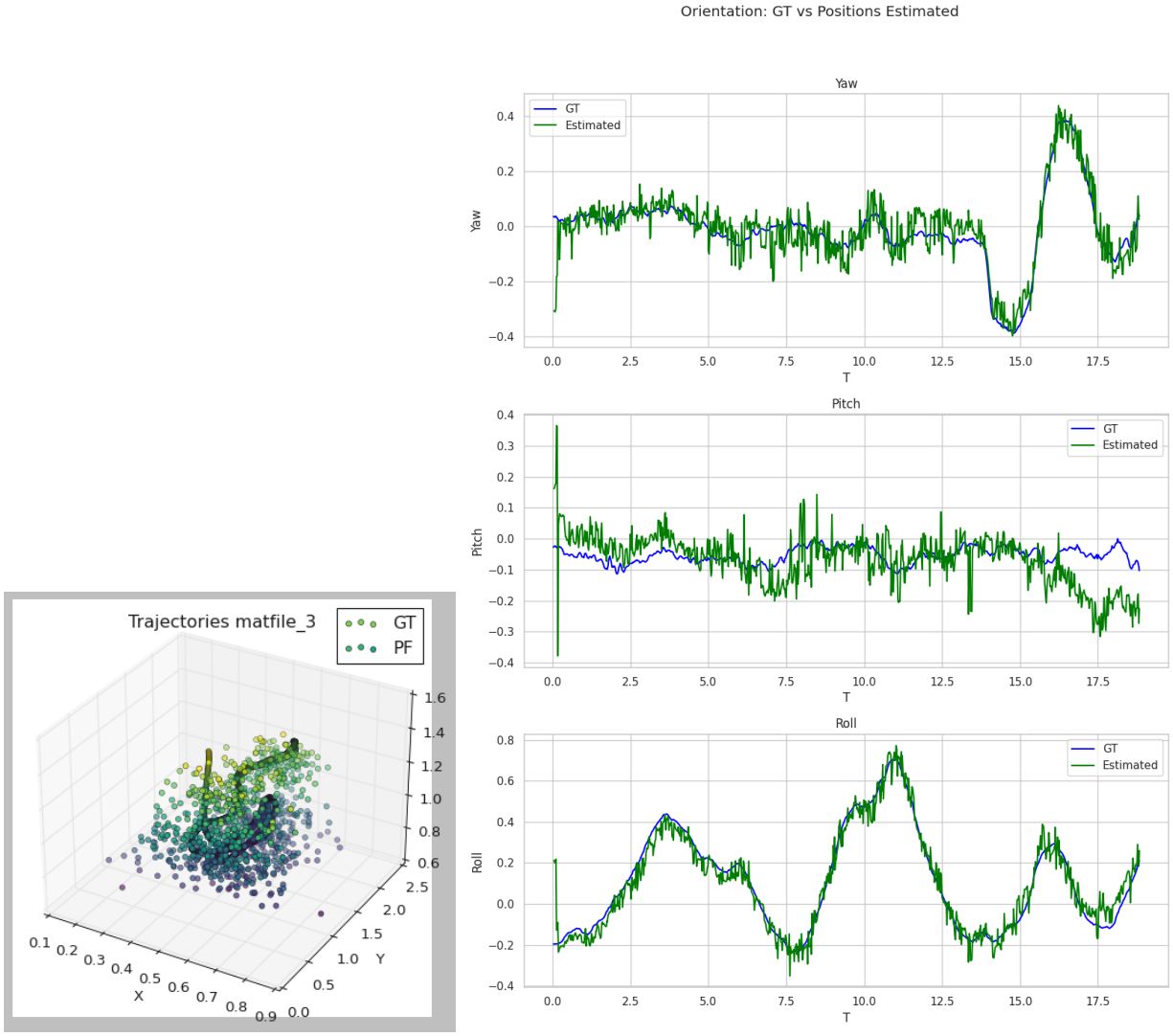

Figure 6, 7: MatFile3 Trajectories & Orientation Plots

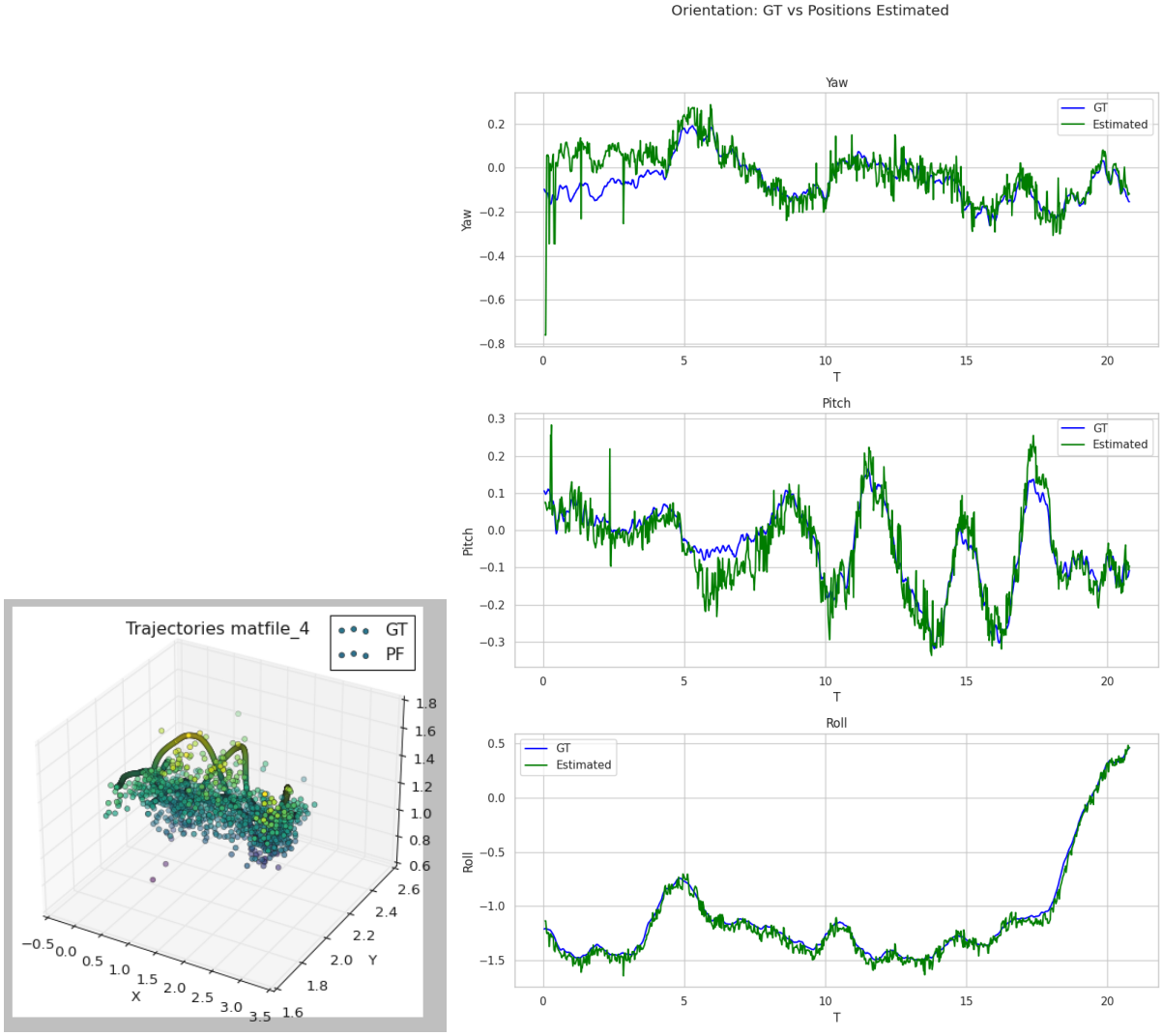

Figure 8, 9: MatFile4 Trajectories & Orientation Plots

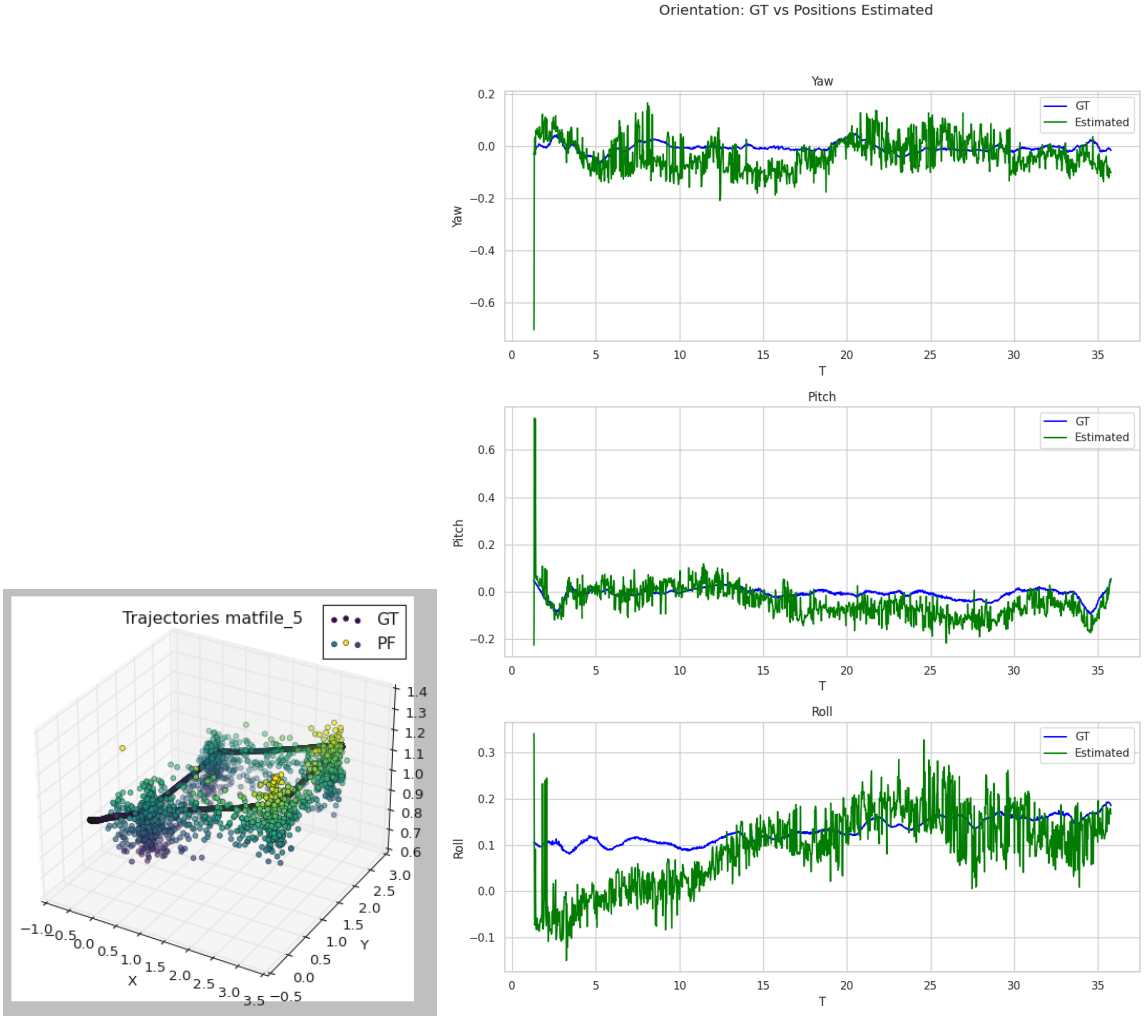

Figure 10, 11: MatFile5 Trajectories & Orientation Plots

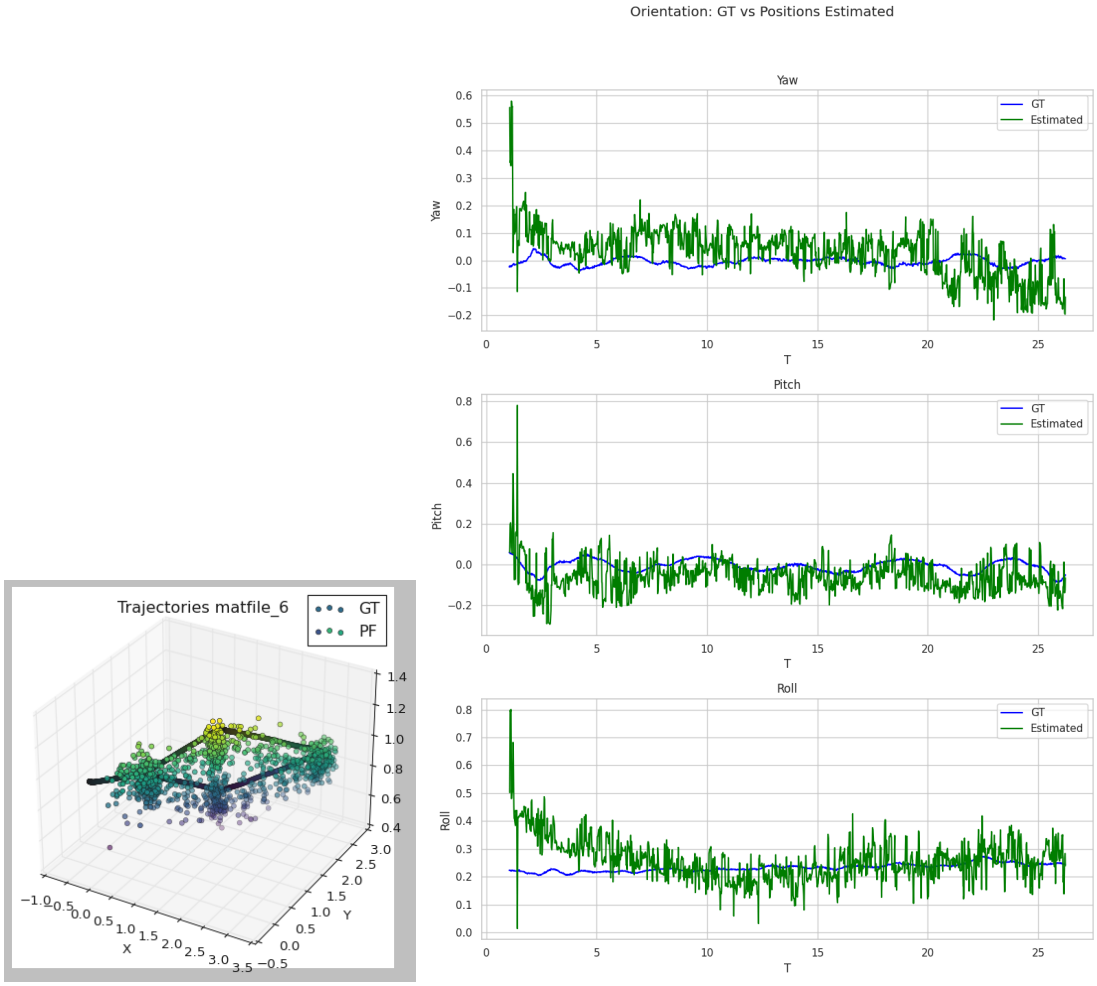

Figure 12, 13: MatFile6 Trajectories & Orientation Plots

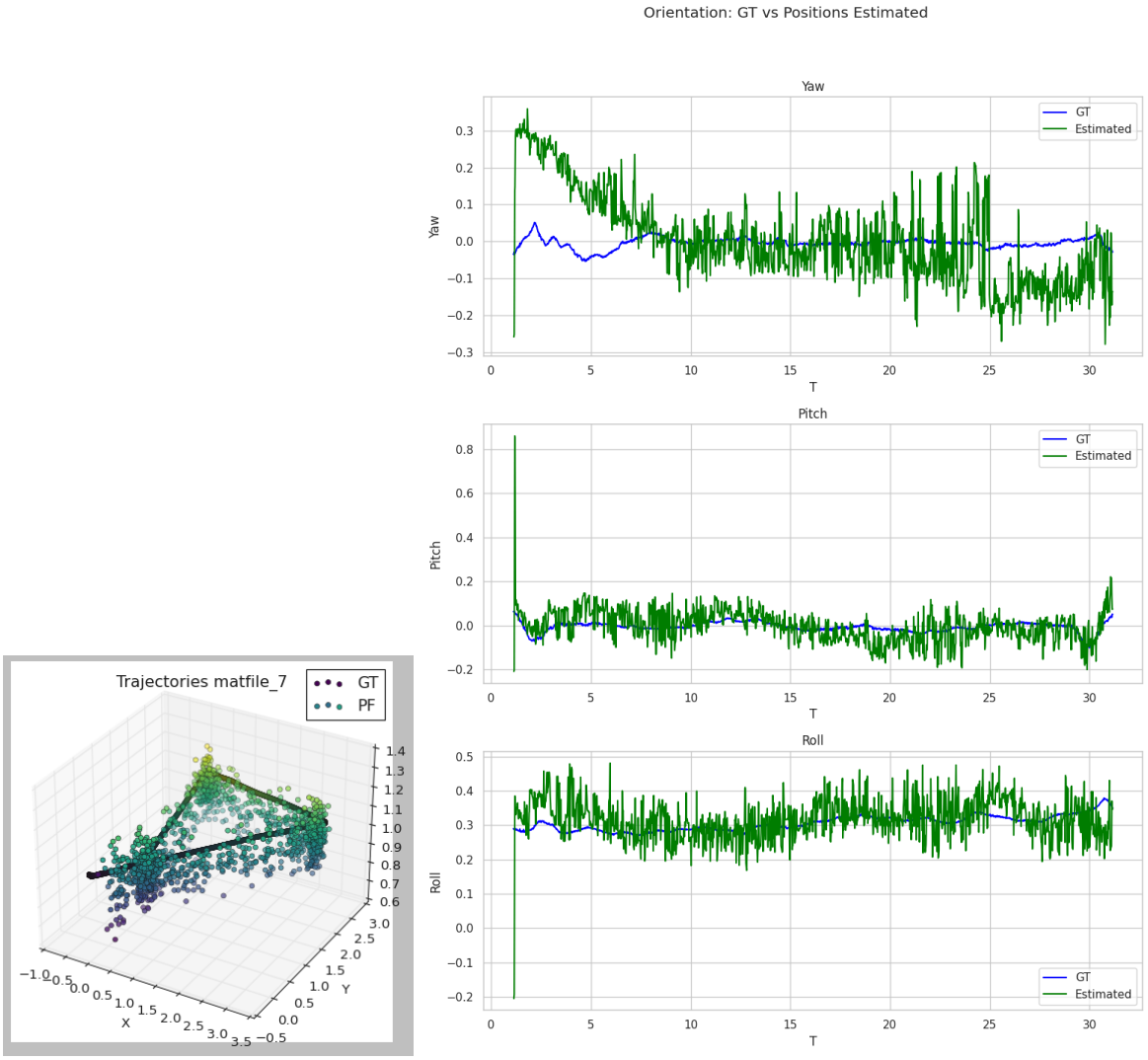

Figure 14, 15: MatFile7 Trajectories & Orientation Plots

Task 2:

In the analysis, the state estimation accuracy of the particle filter was compared using three distinct estimation methods: Average Weighted, Average, and Highest Weight. The evaluation through Root Mean Square Error metrics showed minimal variance between these techniques which could indicate a filter with balanced weight distribution among the particles.

It was also observed that the Average Weighted method, which incorporates the probability of each particle, did not offer a substantial benefit over the simpler Average or Highest Weight methods. This tight clustering of RMSE values suggests that all three methods provide comparably precise estimations for the dataset in question.

The second plot illustrates the RMSE error trends across different particle counts for each dataset. The graphs demonstrate a tendency where increasing the number of particles from 250 to 5000 generally leads to a more compact distribution of RMSE values, suggesting an improvement in estimation accuracy. It was also noted that the error spikes decline as the particle count rises, indicating that a greater number of particles may lead to more stable and reliable state estimations. This could be attributed to the increased particle diversity, which allows for a better representation of the state space and improves the robustness of the particle filter against noise and uncertainty in the measurements.

However, the major drawback of utilizing a higher particle number was that the calculations required more computational resources.

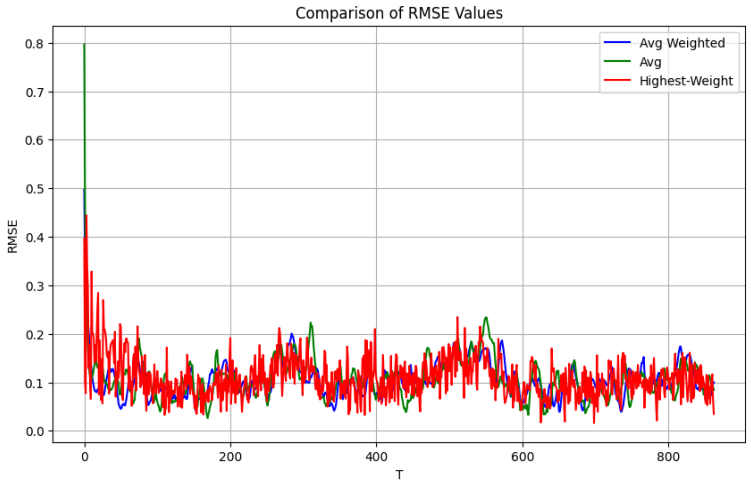

Figure 16: MatFile1 RMSE Comparison

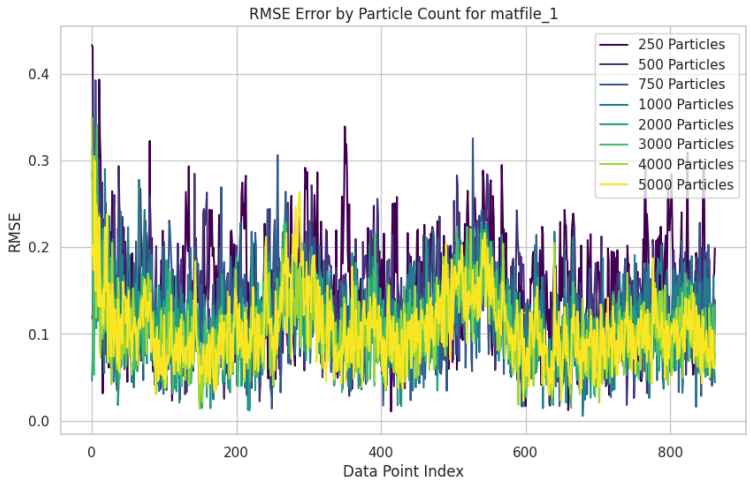

Figure 17: MatFile1 Particles# Performance Comparison

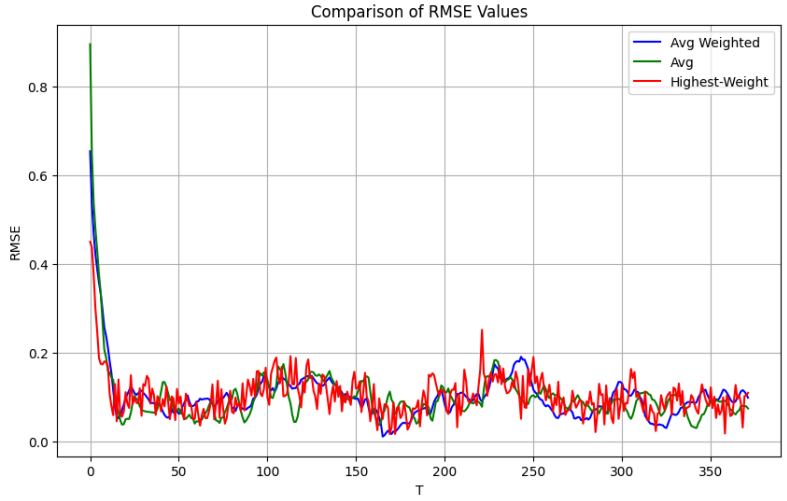

Figure 18: MatFile2 RMSE Comparison

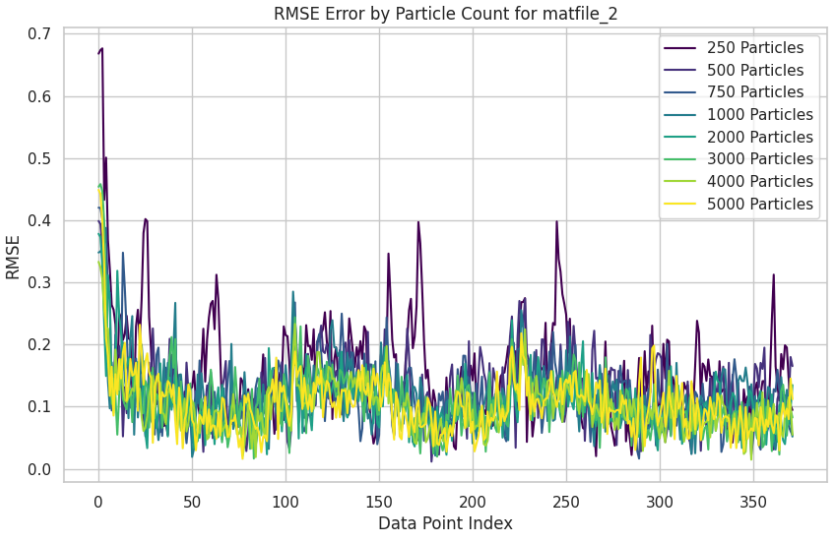

Figure 19: MatFile2 Particles# Performance Comparison

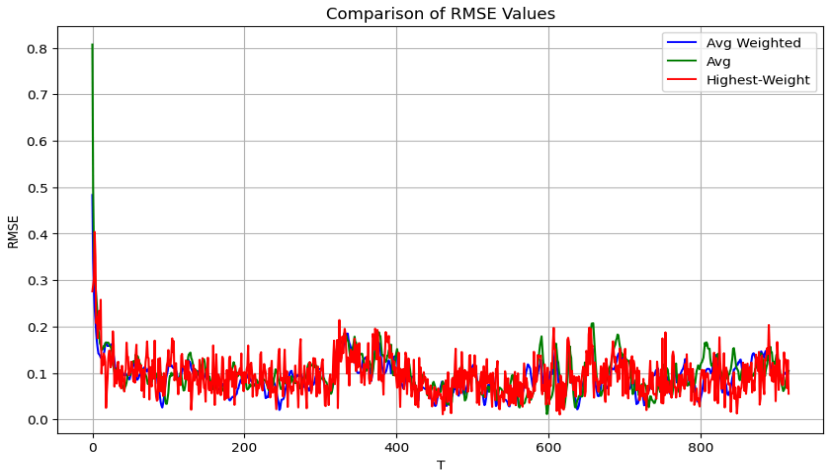

Figure 20: MatFile3 RMSE Comparison

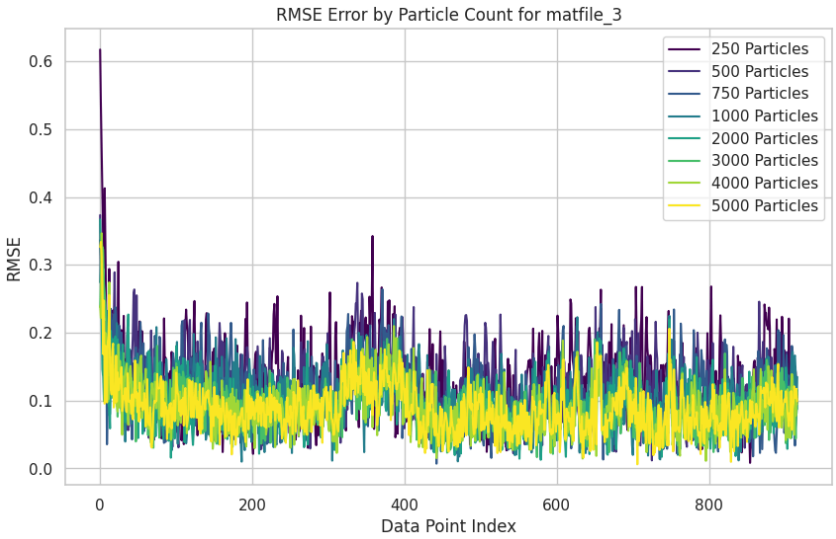

Figure 21: MatFile3 Particles# Performance Comparison

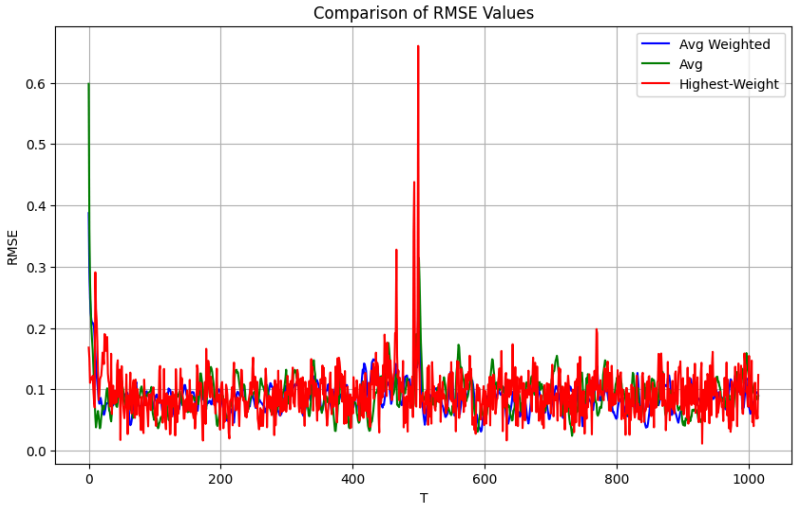

Figure 22: MatFile4 RMSE Comparison

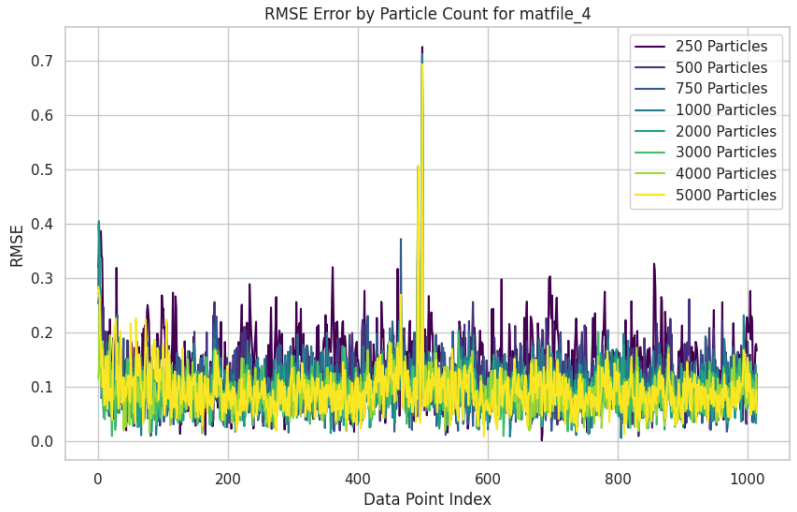

Figure 23: MatFile4 Particles# Performance Comparison

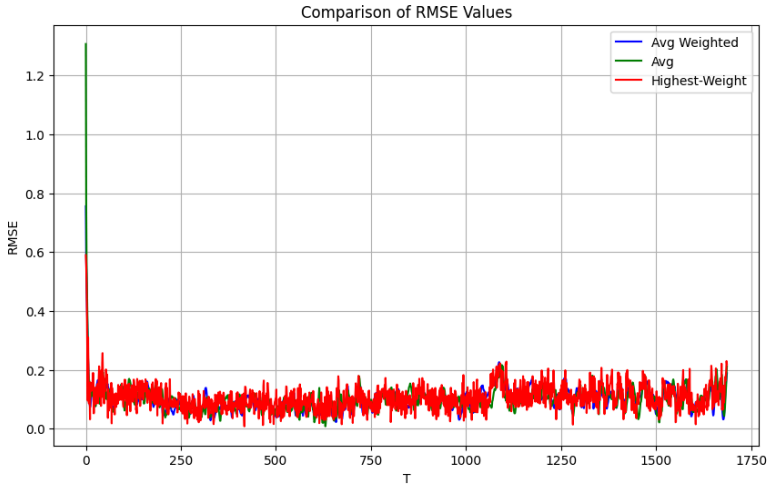

Figure 24: MatFile5 RMSE Comparison

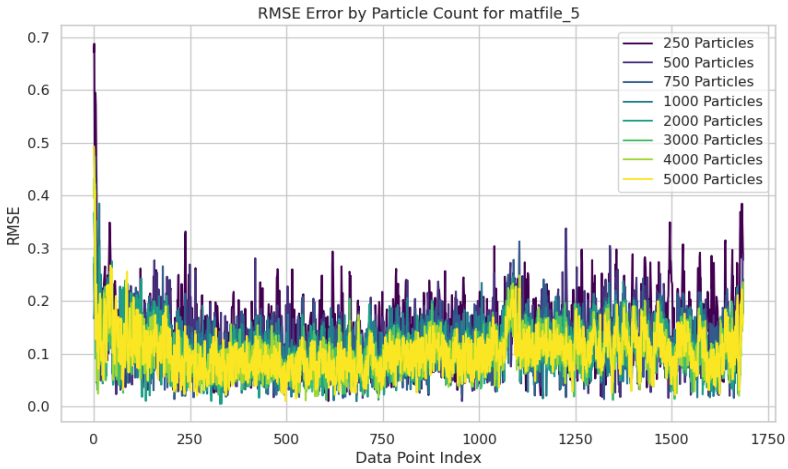

Figure 25: MatFile5 Particles# Performance Comparison

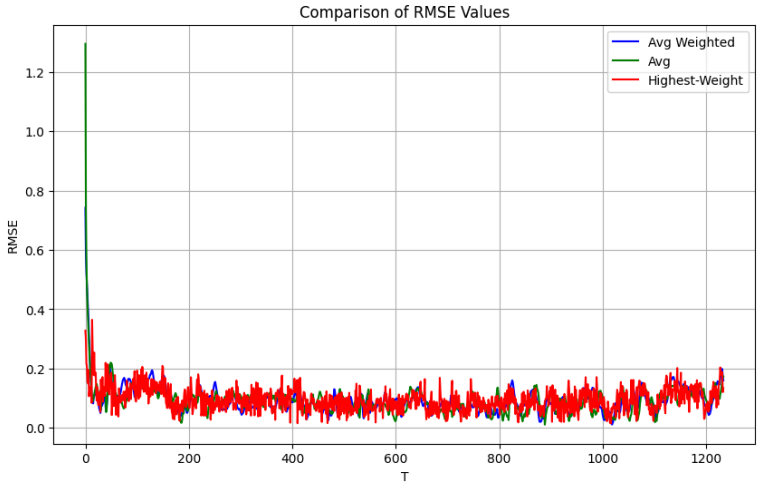

Figure 24: MatFile6 RMSE Comparison

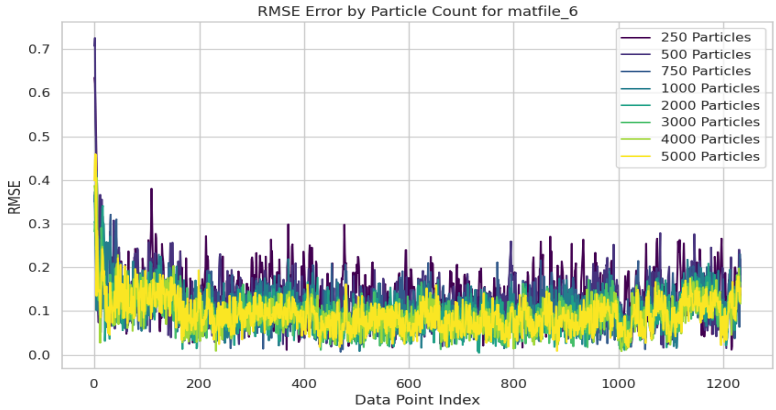

Figure 25: MatFile6 Particles# Performance Comparison

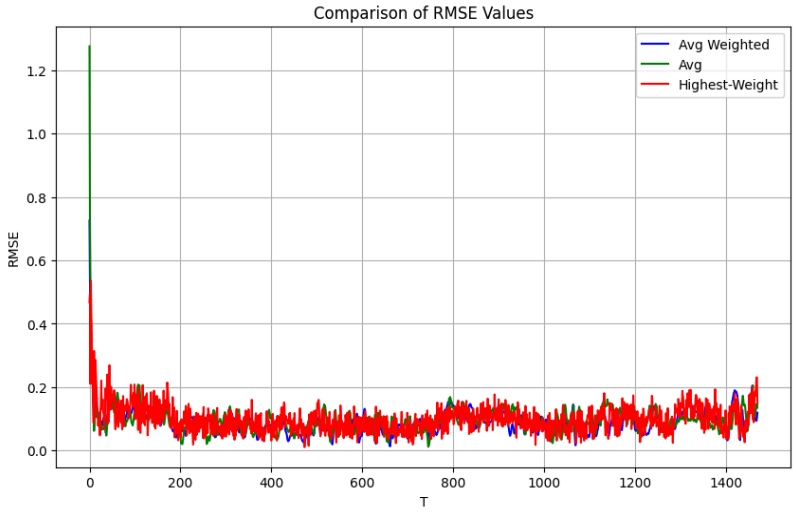

Figure 26: MatFile7 RMSE Comparison

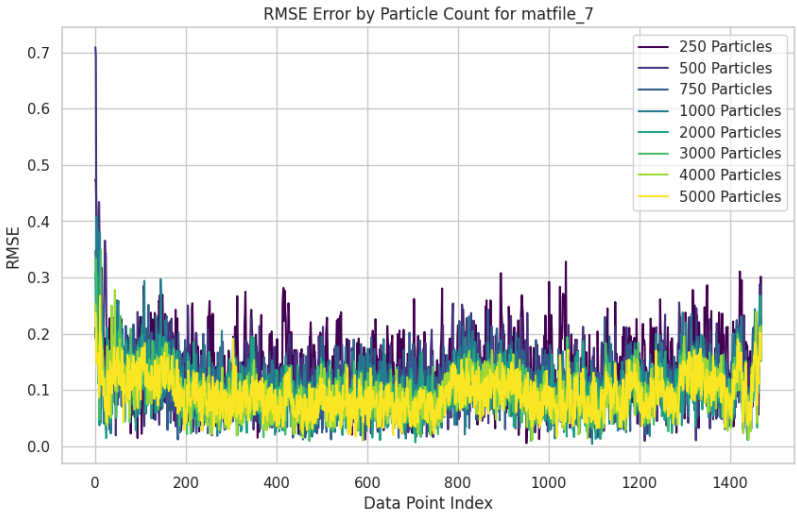

Figure 27: MatFile7 Particles# Performance Comparison

Task 3:

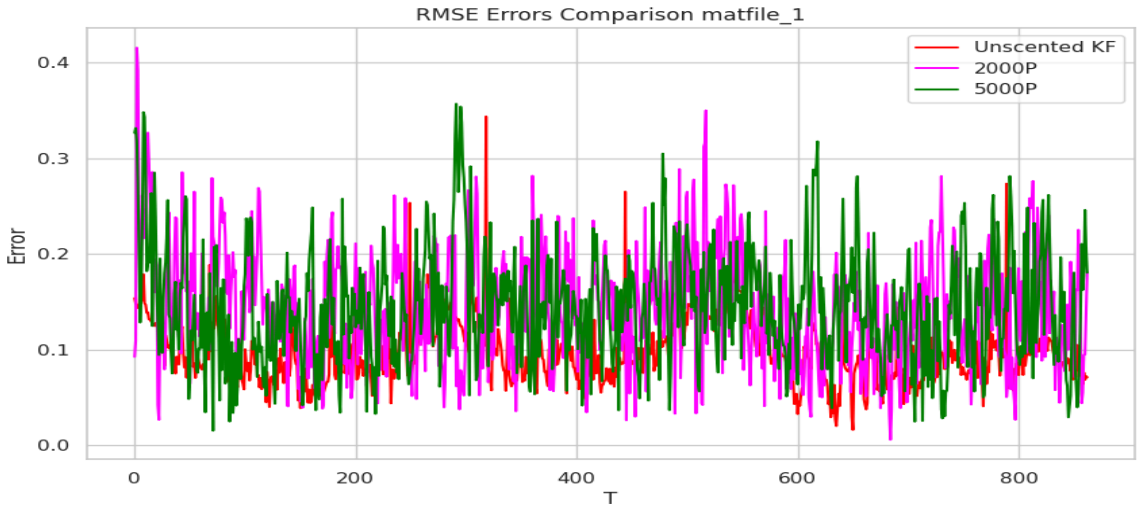

Figure 28: MatFile1 UKF vs PF

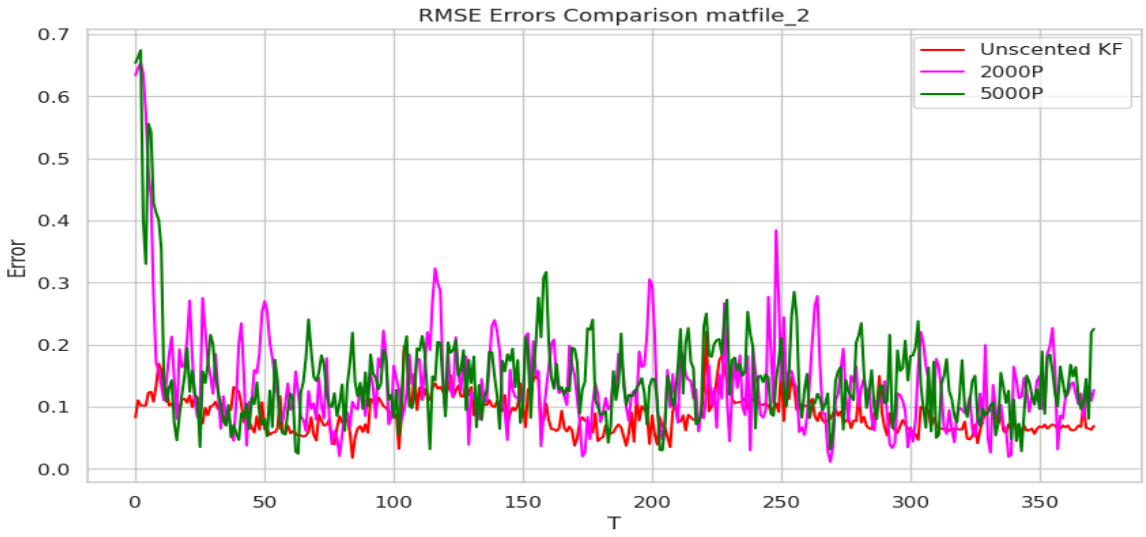

Figure 29: MatFile2 UKF vs PF

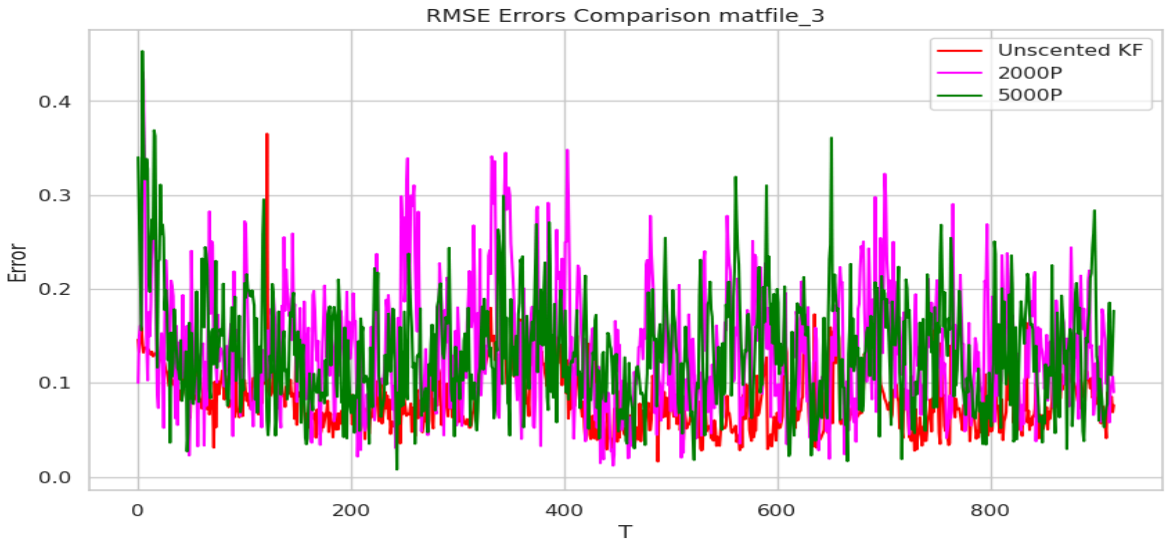

Figure 30: MatFile3 UKF vs PF

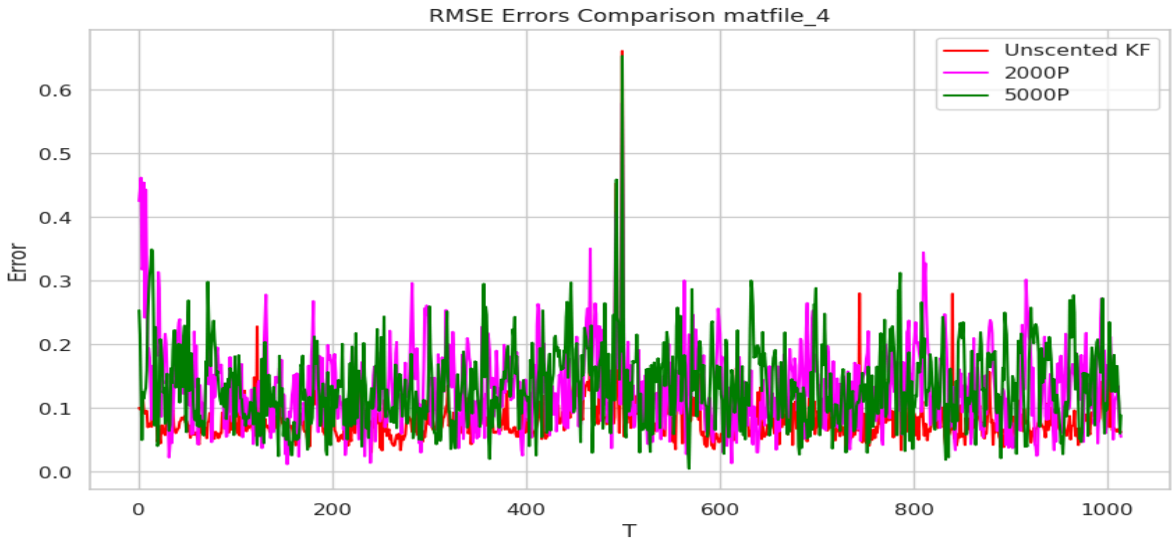

Figure 31: MatFile3 UKF vs PF

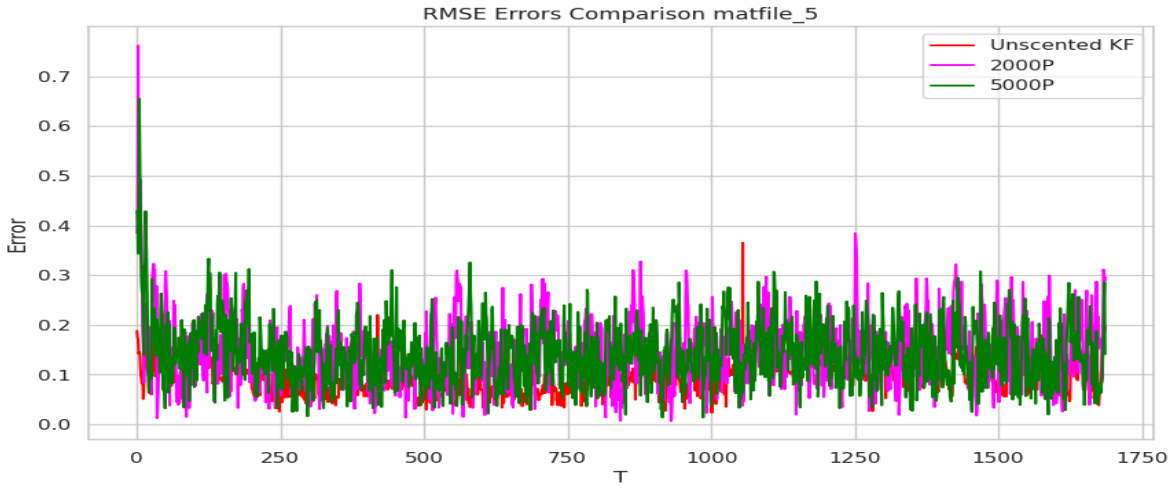

Figure 32: MatFile2 UKF vs PF

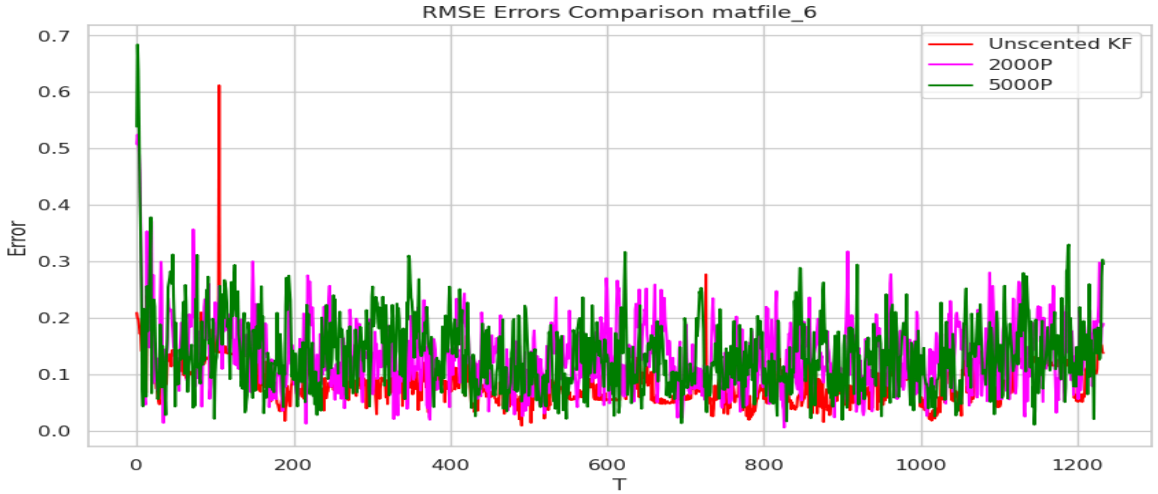

Figure 33: MatFile2 UKF vs PF

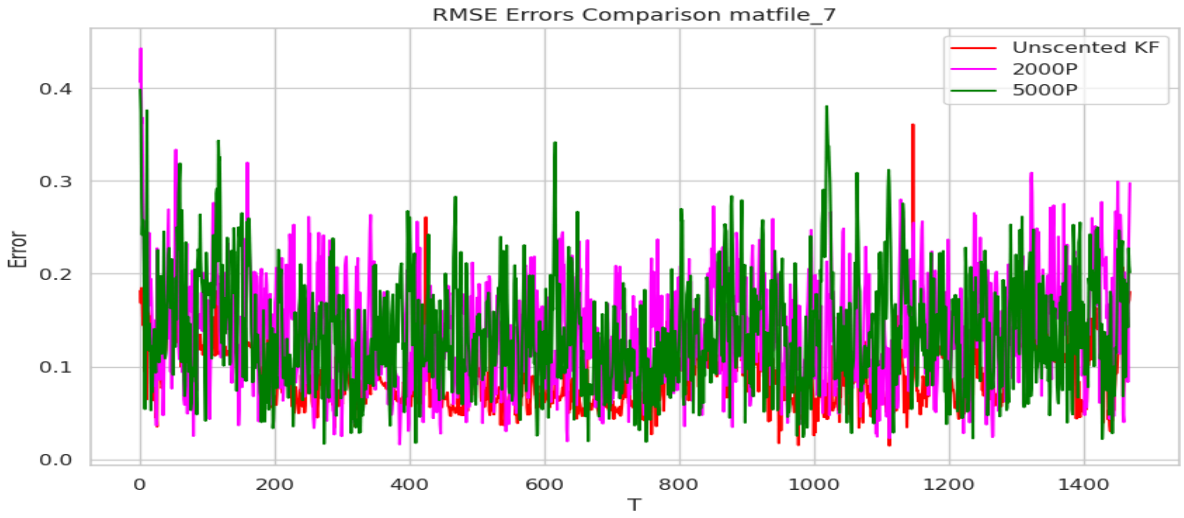

Figure 34: MatFile2 UKF vs PF

The UKF has a well mathematical framework and is slightly straightforward to implement compared to the PF. The UKF follows a predictable sequence of prediction and update steps, which, once understood, can be directly translated into code. In contrast, the PF required a careful consideration of particle initialization, prediction, and resampling, which added layers of complexity to the coding task.

Tuning parameters for the UKF was also found to be less cumbersome. Due to the PF's reliance on a large number of particles, it became computationally intensive to iterate over different parameter values to find the optimal settings. The UKF, with fewer parameters to tune, such as process and measurement noises, allowed for faster and more efficient fine-tuning.

The UKF typically runs faster than the PF because it deals with a fixed number of states and a deterministic approach to updating these states. The PF, especially with a high number of particles, requires more computations per iteration, which can significantly slow down the performance.

Regarding accuracy, the UKF tends to produce lower error rates as it provides a more precise estimate due to its mathematical rigor in tracking the mean and covariance of the state distribution. The PF showed improved performance with an increase in particle count, suggesting better approximation of the probability density function.

However, it's important to note that the performance gain diminishes beyond a certain point, as seen from the modest improvements when particle counts increased from 2000 to 5000.