Aadesh Surendra Varude
avarude@wpi.edu

ASSIGNMENT-5

1 Introduction

This project aims to implement an INS/GNSS integration using an Unscented Kalman Filter (UKF). The choice of filter will depend on the specific requirements of the application, focusing on achieving optimal performance in global-scale navigation contexts. The state model adopted from a previous Nonlinear Kalman Filter project will be utilized, with modifications to accommodate the larger scale of operation.
The navigation framework will be based on a Local Navigation Frame (LNF), which can be configured either as North-East-Down or East-North-Up. The position coordinates are expressed in:

$$\mathbf{p} = \begin{bmatrix} \text{Latitude (decimal degrees)} \\ \text{Longitude (decimal degrees)} \\ \text{Altitude (meters below mean sea level)} \end{bmatrix}$$

Attitude is represented in Euler angles relative to the LNF:

$$\mathbf{q} = \begin{bmatrix} \text{Roll} \\ \text{Pitch} \\ \text{Yaw} \end{bmatrix}$$

Velocity vectors are given in meters per second. This setup will utilize a dual-architecture approach involving both feed-forward and feedback mechanisms. The feedback architecture includes IMU biases within the state vector, reporting positions corrected for these biases. Conversely, the feedforward model employs an error-state vector:

$$\mathbf{x} = [\mathbf{p}\ \mathbf{q}\ \dot{\mathbf{p}}\ \mathbf{e}]$$

where $\mathbf{e}$ denotes the error correction to the base position states $\mathbf{p}$, ensuring the reported position is $\mathbf{p} - \mathbf{e}$. Key updates involve attitude, velocity, and position, following the specific sequence to maintain accuracy in the calculated state. The integration and propagation of these updates are critical for ensuring the system's accuracy and reliability in real-time operational environments. The use of the haversine formula for calculating position errors will enhance the precision of the integration process, supported by open-source Python tools or custom implementations as needed.

1.1 Attitude Update

The attitude update utilizes gyroscope measurements to compute changes in orientation. Assuming a feedback filter model and including IMU biases, the update is performed as follows:

$$\begin{aligned} \omega_{\text{corrected}} &= \omega_{\text{measured}} - \mathbf{b}_g \\ \mathbf{\Omega}_{\text{earth}} &= \begin{bmatrix} 0 & -\omega_E & 0 \\ \omega_E & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \mathbf{\Omega}_{\text{corrected}} &= [\omega_{\text{corrected}} \times] \\ \mathbf{R}_t &\approx \mathbf{R}_{t-1}(\mathbf{I} + \mathbf{\Omega}_{\text{corrected}}\Delta t - (\mathbf{\Omega}_{\text{earth}} + \mathbf{\Omega}_{\text{corrected}})\mathbf{R}_{t-1}\Delta t) \end{aligned}$$

Where ω_E is the Earth's rotation rate, and $\mathbf{b}_g$ is the bias in the gyroscope.

1.2 Velocity Update

The velocity update accounts for accelerometer readings, adjusting for biases and gravitational effects:

$$\begin{aligned} \mathbf{f}_{\text{corrected}} &= \mathbf{f}_{\text{measured}} - \mathbf{b}_a \\ \mathbf{v}_t &= \mathbf{v}_{t-1} + \left(\frac{\mathbf{R}_{t-1} + \mathbf{R}_t}{2} \mathbf{f}_{\text{corrected}} + \mathbf{g}(\mathbf{L}_{t-1}, h_{t-1}) \right) \Delta t \end{aligned}$$

Here, $\mathbf{b}_a$ represents the accelerometer biases, and $\mathbf{g}(\mathbf{L}, h)$ is the gravity vector calculated using the Somigliana model at latitude $\mathbf{L}$ and altitude h.

1.3 Position Update

Finally, the position update adjusts latitude, longitude, and altitude based on the new velocity estimates:

$$
\begin{aligned}
h_t &= h_{t-1} - \Delta t \cdot \frac{v_{D,t-1} + v_{D,t}}{2} \\
L_t &= L_{t-1} + \Delta t \cdot \frac{v_{N,t-1} \cdot R_N(L_{t-1}, h_{t-1}) + v_{N,t} \cdot R_N(L_t, h_t)}{2} \\
\lambda_t &= \lambda_{t-1} + \Delta t \cdot \frac{v_{E,t-1} \cdot (R_E(L_{t-1}, h_{t-1}) \cos L_{t-1}) + v_{E,t} \cdot (R_E(L_t, h_t) \cos L_t)}{2}
\end{aligned}
$$

Where R_N and R_E are the Earth's radii of curvature in the north-south and east-west directions, respectively.

2 Task 1: Observation Model

2.1 Observation Model Function Overview for Feedback

The function `measurement_model` is designed to construct and apply a measurement matrix **C** to a state vector **x**. This model is crucial for the integration of INS/GNSS systems, where it is used to map the state vector into the measurement space.
The function begins by defining a 6x15 zero matrix **C**, where N is the length of the state vector **x**. This matrix is structured to select specific elements from **x** for the measurement process:

- The first three rows of **C** are set to form an identity matrix in the top left corner. This configuration extracts the first three elements of **x**, which typically represent positional coordinates in a navigation system.
- The last three rows are similarly configured with an identity matrix but are placed starting from the seventh column. This part of the matrix is used to extract elements from **x** corresponding to velocities.

The C matrix is:

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

The function returns the product of **C** and **x**, effectively filtering and reformatting the state variables into a measurement vector suitable for further processing in the INS/GNSS integration.

2.2 Observation Model Function Overview for Feedforward

The function `measurement_model_ff` provided represents a linear measurement model used in a feedforward control mechanism.
The measurement model can be mathematically expressed as:

$$e = Cx - z \tag{2}$$

where:

- $x \in \mathbb{R}^n$ is the state vector of the system.
- $z \in \mathbb{R}^3$ represents the observed measurements.
- $C \in \mathbb{R}^{3\times n}$ is the measurement matrix.
- $e \in \mathbb{R}^3$ is the error in measurements.

Here we know that the state space contains the term e which is an error for the position only hence the C in this equation is (3,12).

$$
C = \begin{bmatrix}
1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0
\end{bmatrix} \tag{3}
$$

Once we calculate the error we just put that error in the state vector and then we propagate those states and update the covariance matrix and Kalman gain and propagate the error and further report x -e positions.

3 Task 2: Nonlinear Error State implementation

In this task, in the iterative process of a Kalman filter, the update step plays a crucial role in refining the state estimate based on incoming measurements. This process involves several key stages, each contributing to enhancing the accuracy of the estimation. I have defined the state space as suggested in the assignment

$$\mathbf{x} = [\mathbf{p}\ \mathbf{q}\ \dot{\mathbf{p}}\ \mathbf{e}]$$

Upon initializing the update step, sigma points are calculated from the predicted state estimate. These sigma points represent potential state estimates and are pivotal in the subsequent stages of the update process. Concurrently, two matrices are initialized: Z and Z_{error}. Z serves as a container for storing the predicted measurements.
Further, each sigma point is individually evaluated. The sigma point X is extracted and its predicted measurement is computed. Subsequently, the prediction error $d1$ is determined by subtracting the mean from each predicted measurement, contributing to the assessment of uncertainty in the measurement predictions. This error is utilized to update the measurement prediction error covariance matrix S. Moreover, the cross-covariance matrix $cross\_cov$, delineating the relationship between state prediction error and measurement prediction error, is computed. Finally, the Kalman gain is derived, facilitating the optimal integration of measurement information into the state estimate.
The Kalman gain, derived from the previously computed covariance matrices, governs the adjustment of the state estimate x. This adjustment is orchestrated by considering the difference between the actual measurement $z[i]$ and the predicted measurement z_{mean}, ensuring the state estimate aligns optimally with the incoming measurement information.
Lastly, the results of the update step are stored for subsequent iterations. The updated state estimate x finds its place in the historical record of state estimates, captured within $x_{\text{prop\_hist}}$. Moreover, for the feed-forward model, adjustments are made to x_{mean} to incorporate the error mean, enriching the understanding of state estimation uncertainties.
In essence, the update step of the Kalman filter embodies a meticulous process, orchestrating the fusion of measurement information with state estimates while meticulously managing uncertainties, thereby refining the accuracy of the estimation over iterative cycles.

4 Task 3: Nonlinear Full State implementation

The state vector for this implementation includes various parameters necessary for comprehensive navigation and error correction. The model encompasses positional, attitudinal, and velocity components, as well as bias corrections for both gyroscope and accelerometer outputs. Specifically, the state vector is defined as:

$$\mathbf{x} = \begin{bmatrix} L & \lambda & h & \phi & \theta & \psi & v_N & v_E & v_D & b_{ax} \\ b_{ay} & b_{az} & b_{gx} & b_{gy} & b_{gz} \end{bmatrix}^T$$

where:

- L, λ, h represent the latitude, longitude, and altitude.
- ϕ, θ, ψ denote the roll, pitch, and yaw angles.
- v_N, v_E, v_D are the velocities in the north, east, and downward directions.
- b_{ax}, b_{ay}, b_{az} are the accelerometer biases in the respective axes.
- b_{gx}, b_{gy}, b_{gz} are the gyroscope biases in the respective axes.

For this task, I have implemented the Unscented Kalman Filter (UKF) this is implemented as per the previous assignment and lecture notes provided for the UKF. The UKF class is initialized with the dimension n of the state space, which reflects the number of state variables in the system. The UKF parameters—alpha, kappa, and beta—are set to tune the filter's performance, affecting the spread of the sigma points and the confidence in the prior knowledge of the state distribution. Weights for the covariance (wc) and the mean (wm) are computed to adjust the influence of each sigma point during the update phase. The first element of these weight arrays is computed differently to incorporate prior knowledge (captured by beta), and the remaining weights are identical for the rest of the sigma points. Sigma points are sampled around the mean state to capture the posterior state distribution. These points are spread according to the Cholesky decomposition of the scaled covariance matrix, ensuring that they represent the state uncertainty effectively.

5 Task 4: Discussion and performance analysis

5.1 Accuracy and Precision Evaluation

Accuracy is assessed by how close the estimated positions are to the GNSS measurements, quantified by mean error. The nonlinear method provides decent accuracy and the Harvessian error is around 3000 m, whereas the error state performs better and the Harvessian error is around 100m.
Below are the plots for the position by the non-linear full-state implementation with the feedback method :

5.1.1 Feedback full-state bias Plots:

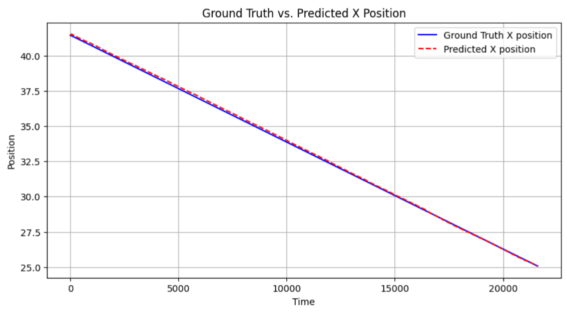

Figure 1: Latitude plot using Nonlinear Full State implementation.

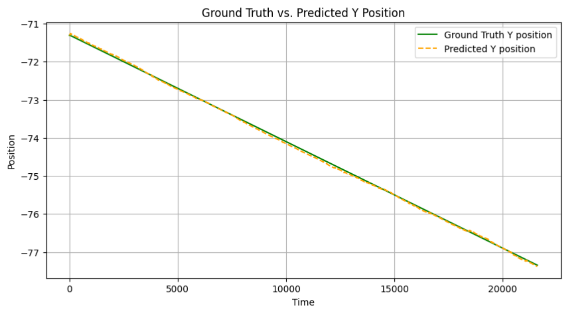

Figure 2: Longitude plot using Nonlinear Full State implementation.

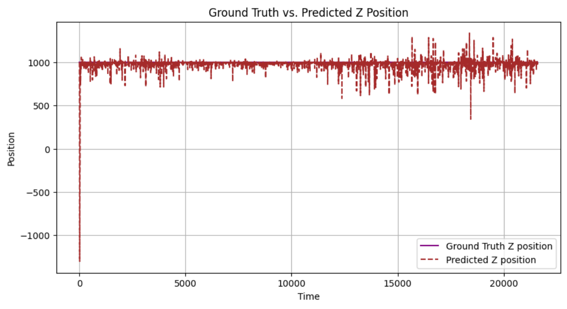

Figure 3: Altitude plot using Nonlinear Full State implementation.

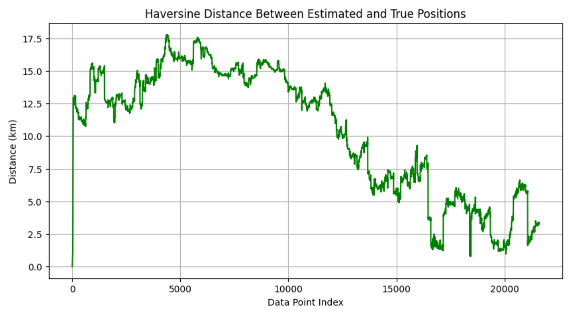

Figure 4: Haverine distance graph Nonlinear Full State implementation.

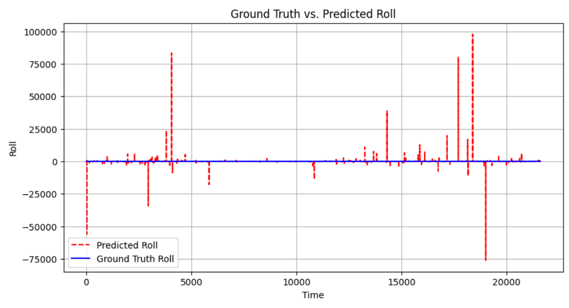

Figure 5: Roll plot using Nonlinear Full State implementation.

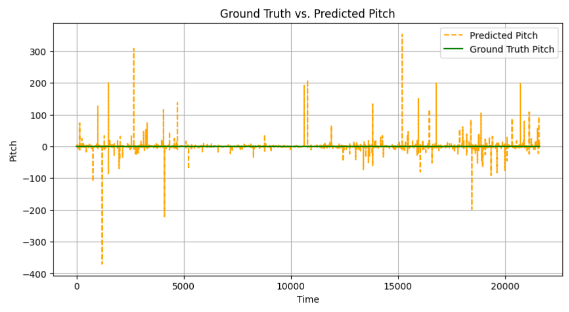

Figure 6: Pitch plot using Nonlinear Full State implementation.

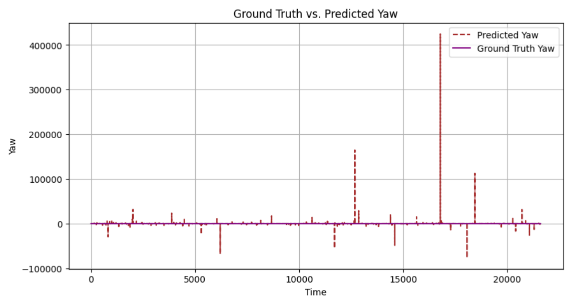

Figure 7: Yaw plot using Nonlinear Full State implementation.

5.1.2 Feedforward error state Plots:

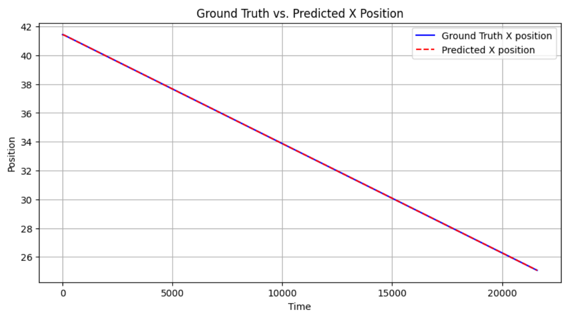

Figure 8: Latitude plot using Nonlinear Full State implementation.

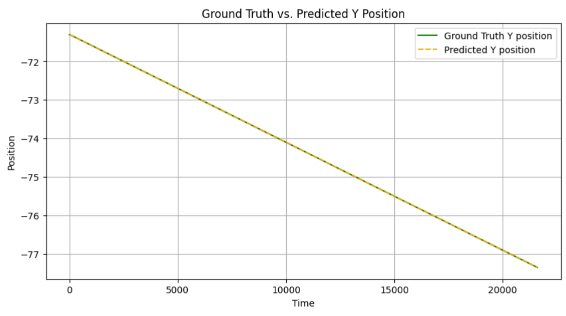

Figure 9: Longitude plot using Nonlinear Full State implementation.

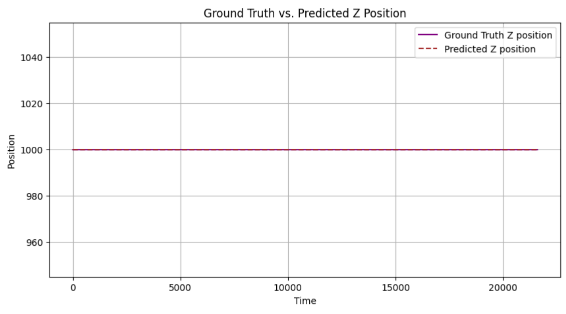

Figure 10: Altitude plot using Nonlinear Full State implementation.

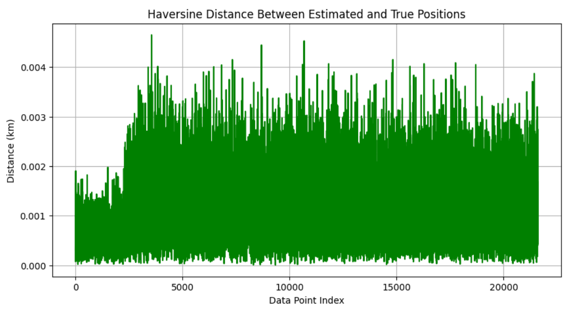

Figure 11: Haversine distance graph Error State implementation.

5.2 Comparative Analysis and Tuning

Process noise matrix Q tuning is critical as it reflects expected noise in system dynamics. So basically while tuning we know that our measurement model is more reliable so we set Q a bit high and R low.

Feedback Case:

$$\begin{aligned}R &= \text{diag}([2e-3, 2e-3, 2e-3, 2e-3, 2e-3, 2e-3])\\Q &= \text{np.eye}(15) \times 1e2\end{aligned}$$

Feedforward Case:

$$\begin{aligned}R &= \text{diag}([5e-3, 5e-3, 5e-3])\\Q &= \text{np.eye}(12) \times 1e2\end{aligned}$$

In my implementation, I have observed the error-based UKF system to perform better than the normal biased-based feedback state propagation. The latitude and longitude are tracked nicely whereas the actual distance in meters is better for the error state. As one can observe in the graphs the error for the feedforward is much less whereas the feedback rises and then decreases.

5.3 Conclusion

The use of Haversine distance to measure the error between the INS estimates and GNSS measurements has its limitations:

- **Dimension Limitation:** Haversine distance calculates the shortest path between two points on a sphere, considering only latitude and longitude, and not altitude, which may be significant.
- **Noise Sensitivity:** Using GNSS measurements, which include inherent noise, as a baseline for error calculation might not always accurately reflect the performance of the INS system.
- **Geographical Bias:** The Haversine formula assumes a spherical Earth, which can introduce small errors in the calculation, especially over long distances or at specific latitudes.

Possible ways for Improvements: Integration of additional sensors such as LIDAR, cameras, and more advanced GNSS systems to provide richer data inputs. Utilization of advanced filtering techniques and adaptive filters to dynamically adjust to observed noise characteristics and improve estimation accuracy.