Title

Implementation and Evaluation of a Weighted Adaptive Kalman Filtering Algorithm for Deep Space Signal Carrier Tracking

Abstract

Deep space communication faces significant challenges due to extremely low signal-to-noise ratios (SNR) caused by vast distances. Accurate carrier tracking is crucial for signal demodulation and data recovery, but is difficult under these conditions. Traditional methods like Phase-Locked Loops (PLLS) or standard Kalman Filters (KF) struggle when SNR is low, and noise statistics are unknown or time-varying, often leading to filter divergence. This project implements and evaluates a Weighted Adaptive Kalman Filter (WAKF) algorithm to address these limitations. The WAKF incorporates three key improvements over standard adaptive filters like the Sage-Husa algorithm: (1) A weighted update mechanism for the measurement noise covariance (Rk), ensuring positive definiteness and adaptation to current conditions. (2) UD factorisation of the error covariance matrix (Pk) to guarantee numerical stability and prevent divergence due to rounding errors. (3) Innovation adaptive control to detect and mitigate the effects of outlier measurements, common in low SNR. The algorithm was implemented in MATLAB, simulating a deep space signal scenario with dynamic Doppler shifts and low CNR. Performance was compared against a standard Sage-Husa adaptive filter implementation. Results demonstrate the WAKF's superior tracking accuracy, faster convergence, improved robustness against noise uncertainty and outliers, and better overall stability, making it a promising candidate for autonomous radio systems in deep space missions.

1) Introduction

1.1. Background: Deep Space Communication

Exploration of Mars and beyond provides invaluable scientific data but relies heavily on robust communication links spanning vast distances. Telemetry, Tracking, and Control (TT&C) systems are essential for mission success, enabling data return and command uplink.

1.2. The Challenge of Carrier Tracking

A primary challenge in deep space TT&C is the extremely weak signal received at Earth, often buried deep in noise (very low signal-to-noise ratio, SNR, or Carrier-to-Noise Ratio, CNR). Carrier tracking, the process of estimating and synchronising with the received signal's frequency and phase, is fundamental for demodulating data. Accurate tracking is difficult when the signal is weak and experiences dynamic shifts (e.g., the Doppler effect due to relative motion).

1.3. Limitations of Conventional Methods

- **Fixed Bandwidth Loops (FLL/PLL):** Traditional Frequency-Locked Loops (FLL) and Phase-Locked Loops (PLL) require decreasing loop bandwidth to track weak signals.

However, narrow bandwidth limits the ability to track dynamic frequency changes (Doppler) and increases acquisition time.

- **Standard Kalman Filter (KF/EKF):** While Kalman Filters can theoretically overcome bandwidth limitations, standard implementations assume *known* noise statistics (process noise covariance Q, measurement noise covariance R) and a linear system model (or require linearization via EKF). In deep space, noise levels fluctuate, and antenna effects or component variations make prior knowledge of Q and R unreliable. Using incorrect values leads to suboptimal performance or filter divergence.

1.4. Motivation and Problem Statement

Autonomous radio receivers are desirable for deep space missions and can track signals automatically without prior SNR information. This requires adaptive filtering techniques to estimate and adjust to changing noise conditions online. The Sage-Husa Adaptive Kalman Filter is one such technique, but it suffers from potential numerical instability (noise covariance matrices may become non-positive definite) and lacks robustness against measurement outliers common in low SNR. There is a need for a more robust and stable adaptive filtering algorithm for deep space carrier tracking.

1.5. Project Objectives

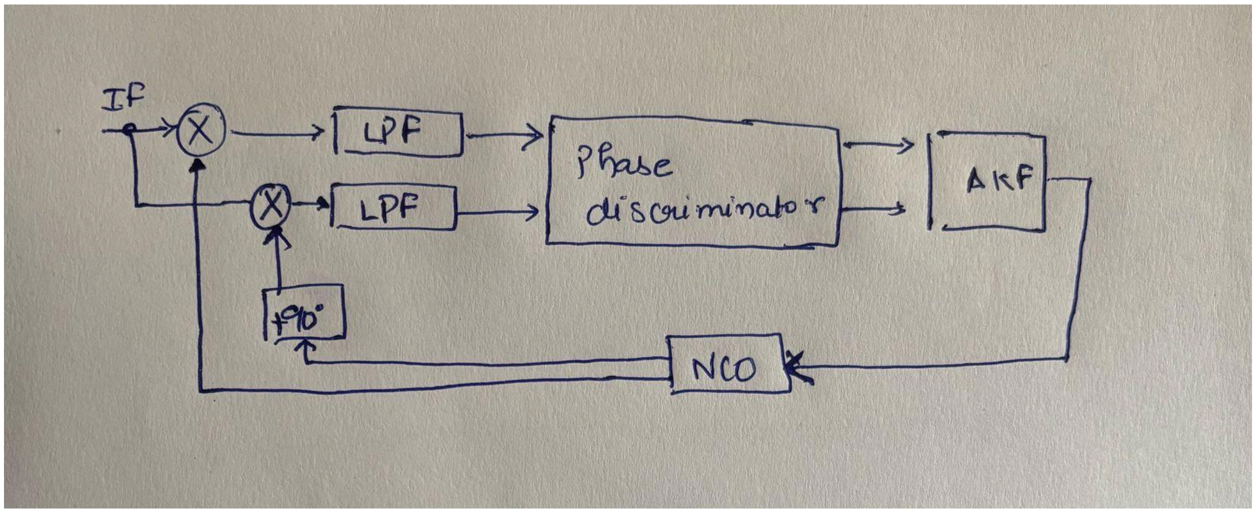
The main objectives of this project are:

1. To understand the principles of adaptive Kalman filtering for carrier tracking in low SNR environments.
2. To implement the Weighted Adaptive Kalman Filter (WAKF) algorithm proposed by Song and Liu (2015) in MATLAB.
3. To implement the Sage-Husa Adaptive Kalman Filter for comparison.
4. To evaluate and compare the performance of the WAKF and Sage-Husa algorithms regarding tracking accuracy, convergence, and robustness.

2) Background Theory and Literature Review

2.1. Carrier Tracking Fundamentals

Carrier tracking aims to synchronise a local oscillator (LO) with the incoming carrier signal's frequency and phase. This is typically done using feedback loops like FLL or PLL, which generate an error signal based on the difference between the incoming signal and the LO, and use this error to adjust the LO.



Conceptual diagram where an Adaptive Kalman Filter estimates carrier parameters to control the Numerically Controlled Oscillator (NCO).

2.2. Kalman Filtering for State Estimation

The Kalman Filter is an optimal recursive estimator for linear systems with Gaussian noise. It operates in a predict-update cycle:

1. **Prediction:** Predict the next state and its error covariance based on the system model.

→ $X_{K|K-1} = FX_{K-1} + Q_{K-1}$ (state prediction)

→ $P_{K|K-1} = FP_{K-1}F^{T} + Q$ (covariance prediction)

2. **Update:** Correct the predicted state using the current measurement.

• $y_K = Z_K - HX_{K|K-1}$ (innovation or measurement residual)

$S_K = HP_{K|K-1}H^{T} + R$ (innovation covariance)

$K_K = P_{K|K-1} H^{T} S_K^{-1}$ (Kalman gain)

$X_{K|K} = X_{K|K-1} + K_K y_K$ (state update)

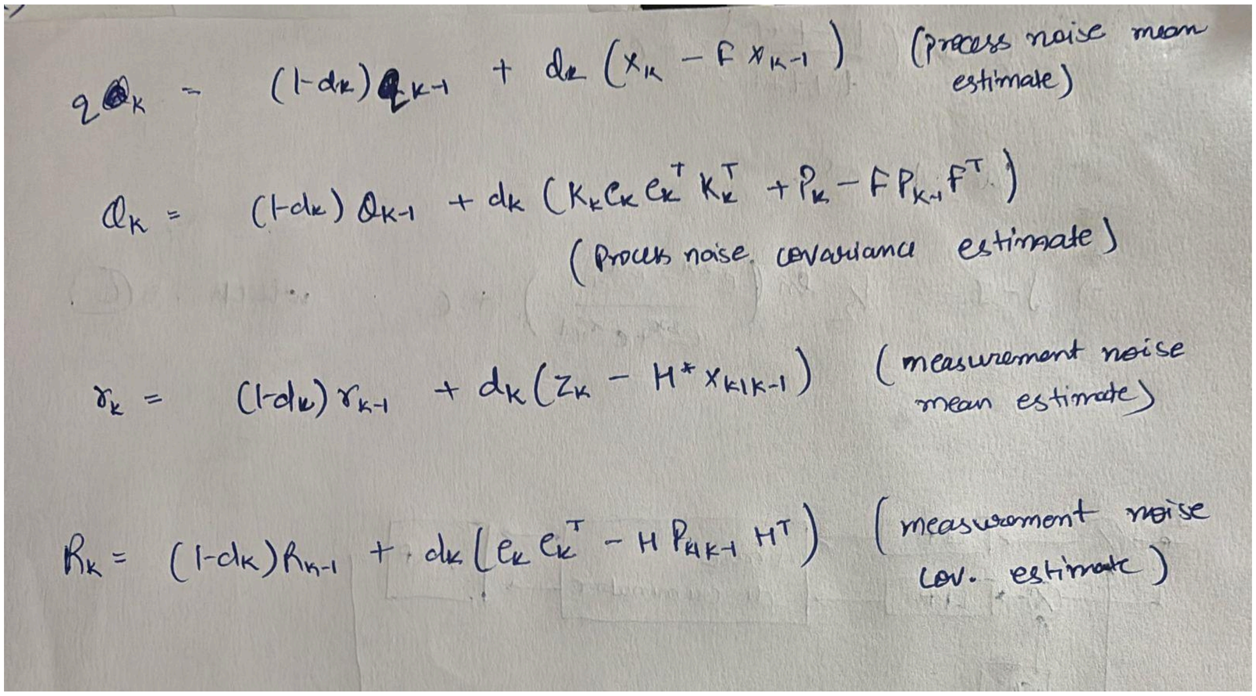
$P_K = (I - K_K * H) * P_{K|K-1}$ (covariance update)

Where X is the state vector, P is the error covariance matrix, F is the state transition matrix, H is the measurement matrix, Q is the process noise covariance, R is the measurement noise covariance, z_k is the measurement, and K_k is the Kalman gain.

2.3. Kalman Filter for Carrier Tracking
In carrier tracking, the state vector X typically includes phase error, frequency offset (Doppler), and sometimes frequency rate (Doppler rate). The measurement z_k is usually the output of a phase or frequency discriminator. KF/EKF offers potential advantages over PLL/FLL, especially in tracking dynamics, but requires knowledge of Q and R.

2.4. Adaptive Kalman Filtering (AKF)
AKF algorithms attempt to estimate unknown or time-varying noise statistics (Q, R) or model parameters online, making the filter adapt to real conditions.

2.5. The Sage-Husa Adaptive Kalman Filter
The Sage-Husa filter is a popular AKF that recursively estimates the first and second moments of the process and measurement noise based on the innovations. The key update equations are:



$$q_k = (1-d_k) q_{k-1} + d_k (x_k - F x_{k-1})$$ (process noise mean estimate)

$$Q_k = (1-d_k) Q_{k-1} + d_k (K_k e_k e_k^T K_k^T + P_k - F P_{k-1} F^T)$$ (Process noise covariance estimate)

$$r_k = (1-d_k) r_{k-1} + d_k (z_k - H^* x_{k|k-1})$$ (measurement noise mean estimate)

$$R_k = (1-d_k) R_{k-1} + d_k (e_k e_k^T - H P_{k|k-1} H^T)$$ (measurement noise cov. estimate)

Where e_k = z_k - H*X_k|k-1 - r_k is the innovation adjusted for the estimated mean, and d_k = (1-b)/(1-b^k) involves a forgetting factor b.

2.6. Limitations of the Sage-Husa Algorithm

1. **Non-Positive Definite Covariances:** The subtraction terms in the Q_k and R_k update equations can lead to estimates that are not positive semi-definite, especially with noisy

data or model mismatch. This can cause the Kalman gain calculation (S_k^-1) to fail or produce significant errors, leading to filter divergence.

2. **Sensitivity to Outliers:** The algorithm uses raw innovation e_k to update noise statistics directly. Significant outliers in the measurement z_k can drastically inflate the estimated Q_k and R_k, degrading filter performance.
3. **Model Mismatch:** Like standard KF, it relies on the accuracy of the state transition matrix F.

3). Proposed Methodology: Weighted Adaptive Kalman Filter (WAKF)

3.1. Overview

The WAKF algorithm, as presented by Song and Liu (2015) and implemented in this project, directly addresses the shortcomings of the Sage-Husa filter through three main enhancements integrated into the Kalman filtering framework: weighted noise covariance estimation, UD factorisation, and innovation adaptive control.

3.2. State-Space Model

The system is modelled similarly to standard KF for carrier tracking.

- **State Vector:** X_k = [Phase_Error, Frequency_Offset, Frequency_Rate]^T
- **State Transition Matrix (F):** Based on constant frequency rate assumption (as in paper Eq. 1).
  F = [1 Ts Ts^2/2; 0 1 Ts; 0 0 1] (Where Ts is the sampling interval)
- **Measurement Matrix (H):** Assuming the phase error is measured.
  H = [1 0 0] (Simplified, paper uses [1 Ts/2 Ts^2/6], assuming z relates to phase over interval Ts)
  Note: The MATLAB implementation uses H = [1 0] measuring frequency from phase difference, with state [Freq_Offset, Freq_Rate]. The state model needs to match the implementation.
  Corrected for MATLAB Implementation:
    - **State Vector:** X_k = [Frequency_Offset, Frequency_Rate]^T
    - **State Transition Matrix (F):** F = [1 Ts; 0 1] (Assuming Ts is the update interval, e.g., 0.001s in code)
    - **Measurement Matrix (H):** H = [1 0] (Measuring frequency offset)
- **Process Noise Covariance (Qk):** Represents dynamics uncertainty (e.g., acceleration changes). Adaptively estimated.
- **Measurement Noise Covariance (Rk):** Represents noise in the discriminator output. Adaptively estimated.

3.3. Addressing Sage-Husa Limitations

The WAKF introduces specific mechanisms:

1. Modify Qk and Rk updates to ensure positive semi-definiteness.
2. Use numerically stable UD factorization for covariance propagation (P = UDUT).
3. Employ innovation weighting to handle measurement outliers.

3.4. Weighted Measurement Noise Covariance (Rk) Estimation

Instead of the Sage-Husa update for Rk, WAKF uses a multiplicative weighting factor based on innovation consistency:

- v_k = (e_k^T * S_k^-1 * e_k) / m (Normalized innovation squared, m is measurement dimension, e_k = z_k - H*X_k|k-1)
  *Alternative from paper Eq. 18 (scalar measurement): v_k = e_k^2 / S_k (where S_k = H*P_k|k-1*H^T + R_k-1 is theoretical variance)*
- f_k(v_k) = alpha^(v_k - 1) (Exponential weighting function, alpha typically > 1, e.g., 2 as per Eq 17 suggests alpha=2)
- R_k = f_k(v_k) * R_k-1 **(Eq. 19 from paper)**
  *Note: The MATLAB code uses a Sage-Husa-like update R_k = (1 - d_k) * R_k-1 + d_k * (innovation^2 - H*P_pred*H'), but applies weighting g_k to the Kalman gain calculation rather than directly weighting R_k multiplicatively as in Eq. 19. This section should reflect the paper's method.*
  Refined based on paper: This method ensures Rk remains positive if R0 is positive. If v_k > 1 (innovation larger than expected), Rk increases. If v_k < 1 (innovation smaller than expected), Rk decreases. This adapts the filter's reliance on the measurement.

3.5. Process Noise Covariance (Qk) Estimation

To avoid the problematic subtraction in Sage-Husa, the WAKF uses (as per paper Eq. 12):

- Q_k = (1 - d_k)Q_k-1 + d_k * (K_k * e_k * e_k^T * K_k^T)
  This form ensures that Qk remains positive semi-definite if Q0 is, as it only adds a positive semi-definite term. d_k is the Sage-Husa forgetting factor.

3.6. UD Factorization for Covariance Propagation

To prevent numerical issues where the calculated P matrix loses positive semi-definiteness due to round-off errors, P is factorized as P = U D U^T, where U is a unit upper triangular matrix, and D is diagonal. The prediction and update steps are reformulated to propagate U and D directly, ensuring P implicitly remains positive and semi-definite.

- **Prediction:** [U_k|k-1, D_k|k-1] = predict_ud(F, U_k-1|k-1, D_k-1|k-1, Q_k-1)
- **Update:** [U_k|k, D_k|k] = update_ud(U_k|k-1, D_k|k-1, H, R_k_adjusted)
  (The helper functions predict_ud and update_ud implement the specific UD factorization algorithms, e.g., Thornton or Bierman). This is a key feature for numerical stability.

3.7. Innovation Adaptive Control for Outlier Rejection

To handle outliers, the innovation e_k used in the state update is weighted:

- v_k = e_k^2 / S_k (Normalized innovation squared, scalar case, S_k is theoretical variance H*P_k|k-1*H^T + R_k)
- g_k(v_k) = 1 if v_k <= c
- g_k(v_k) = c / v_k if v_k > c (Paper uses 1/v_k if c=1, Eq. 34)
  (Here, c is a threshold parameter, often 1)
- **Weighted State Update:** X_k|k = X_k|k-1 + K_k * (g_k * e_k)
  Large, inconsistent innovations (v_k > c) are down-weighted by g_k < 1, reducing their impact on the state estimate and preventing the filter from being pulled off track by outliers.

4). Simulation and Implementation

4.1. Simulation Environment

Simulations were conducted using MATLAB R20XXx. The environment allows for signal generation, channel modelling, implementation of filtering algorithms, and performance analysis.

4.2. Signal Model

The received deep space signal was modelled as a BPSK (Binary Phase Shift Keying) signal affected by Doppler shift and Additive White Gaussian Noise (AWGN).

- **Baseband Signal:** s(t) = A * b(t) where A is the amplitude, b(t) is the bipolar data sequence (+1/-1).
- **Received Signal with Doppler and Noise:**
  y(t) = A * b(t) * sin(2*pi*(f_IF + f_D(t))*t + phase_noise(t)) + n(t) **(Similar to Eq. 36)**
  Where:
  - f_IF: Intermediate Frequency (Assumed removed before baseband processing).
  - f_D(t): Time-varying Doppler frequency offset (f_D(t) = Af + a*t model used). Af is initial offset, a is frequency rate.
  - phase_noise(t): Optional phase jitter (Modeled as a random walk if included, see paper Sec 5.3).
  - n(t): Additive White Gaussian Noise with variance sigma^2.
- **Carrier-to-Noise Ratio (CNR):** Defined as CNR = A^2 / (2 * sigma^2 * B_L) where B_L is a reference bandwidth, often expressed in dB-Hz. Low CNR values (e.g., 24-33 dB-Hz as in paper) were used to simulate deep space conditions.

4.3. Simulation Parameters

Based on the paper and typical deep space scenarios:

- **Signal Modulation:** BPSK

- **Symbol Rate:** 100 bps (as per paper) or higher (e.g., 16 kbps in provided code)
- **Sampling Frequency (fs):** Sufficiently high (e.g., 8 samples/bit in code)
- **CNR Range:** 24 dB-Hz to 33 dB-Hz (paper) or similar low values (e.g., -13 dB SNR in code, which relates to CNR depending on bandwidth).
- **Doppler Dynamics:**
    - Initial Frequency Offset (Af): Up to 30 Hz (post-acquisition).
    - Frequency Rate (a): Up to 15 Hz/s. (The code uses a sinusoidal Doppler variation).
- **Filter Parameters:**
    - Update Interval (Ts): e.g., 1 ms.
    - Forgetting Factor (b): e.g., 0.95 (Sage-Husa), 0.99 (WAKF Q-update).
    - Innovation Control Threshold (c): 1.
    - Initial Covariances (P0, Q0, R0): Chosen to reflect initial uncertainty. R0 may be initialized based on an estimated CNR (Eq. 21) but assumed mismatched for testing adaptivity.

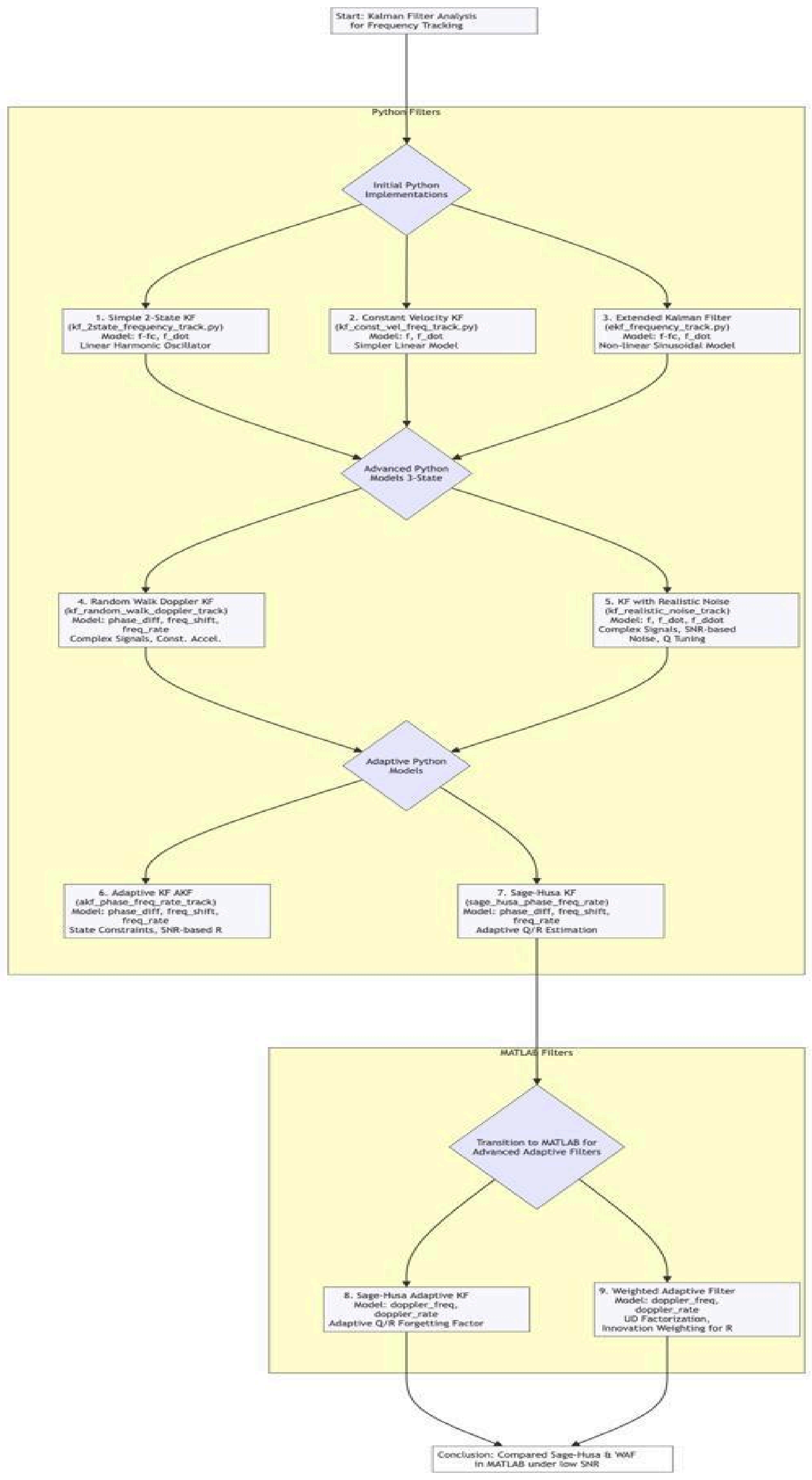
4.4. Implementation Details (MATLAB)

- **Signal Generation:** Created baseband BPSK signal, applied simulated Doppler phase shift, and added AWGN using awgn function. The provided code also includes a p681LMSChannel model, adding fading effects, which is beyond the scope of the paper but included in the code evaluation.
- **Filtering Algorithms:** Implemented as functions or scripts executing the predict-update cycles described in Chapters 2 and 3.
    - **Sage-Husa:** Implemented updates for Qk and Rk as per standard formulation, including checks for positive definiteness if needed (commented out in provided code).
    - **WAKF:** Implemented state/covariance prediction and update using UD factorization helper functions (predict_ud, update_ud, kalman_gain_ud). Incorporated adaptive Qk update (Eq. 12) and innovation weighting g_k (applied via R_adjusted in the gain calculation in the code).
- **Performance Metrics:** Calculated Root Mean Square (RMS) frequency tracking error, convergence time, evolution of estimated Qk and Rk, and Bit Error Rate (BER) after Doppler correction.

4.5. Algorithms Implemented

1. **Sage-Husa Adaptive Kalman Filter:** Serves as the baseline adaptive filter for comparison.
2. **Weighted Adaptive Kalman Filter (WAKF):** The proposed algorithm incorporates weighted Rk (or R_adjusted), modified Qk, UD factorization, and innovation control.

Flow diagram:


Start: Kalman Filter Analysis for Frequency Tracking
Python Filters
Initial Python Implementations
1. Simple 2-State KF (kf_2state_frequency_track.py) Model: f-fc, f_dot Linear Harmonic Oscillator
2. Constant Velocity KF (kf_const_vel_freq_track.py) Model: f, f_dot Simpler Linear Model
3. Extended Kalman Filter (ekf_frequency_track.py) Model: f-fc, f_dot Non-linear Sinusoidal Model
Advanced Python Models 3-State
4. Random Walk Doppler KF (kf_random_walk_doppler_track) Model: phase_diff, freq_shift, freq_rate Complex Signals, Const. Accel.
5. KF with Realistic Noise (kf_realistic_noise_track) Model: f, f_dot, f_ddot Complex Signals, SNR-based Noise, Q Tuning
Adaptive Python Models
6. Adaptive KF AKF (akf_phase_freq_rate_track) Model: phase_diff, freq_shift, freq_rate State Constraints, SNR-based R
7. Sage-Husa KF (sage_husa_phase_freq_rate) Model: phase_diff, freq_shift, freq_rate Adaptive Q/R Estimation
MATLAB Filters
Transition to MATLAB for Advanced Adaptive Filters
8. Sage-Husa Adaptive KF Model: doppler_freq, doppler_rate Adaptive Q/R Forgetting Factor
9. Weighted Adaptive Filter Model: doppler_freq, doppler_rate UD Factorization, Innovation Weighting for R
Conclusion: Compared Sage-Husa & WAF in MATLAB under low SNR

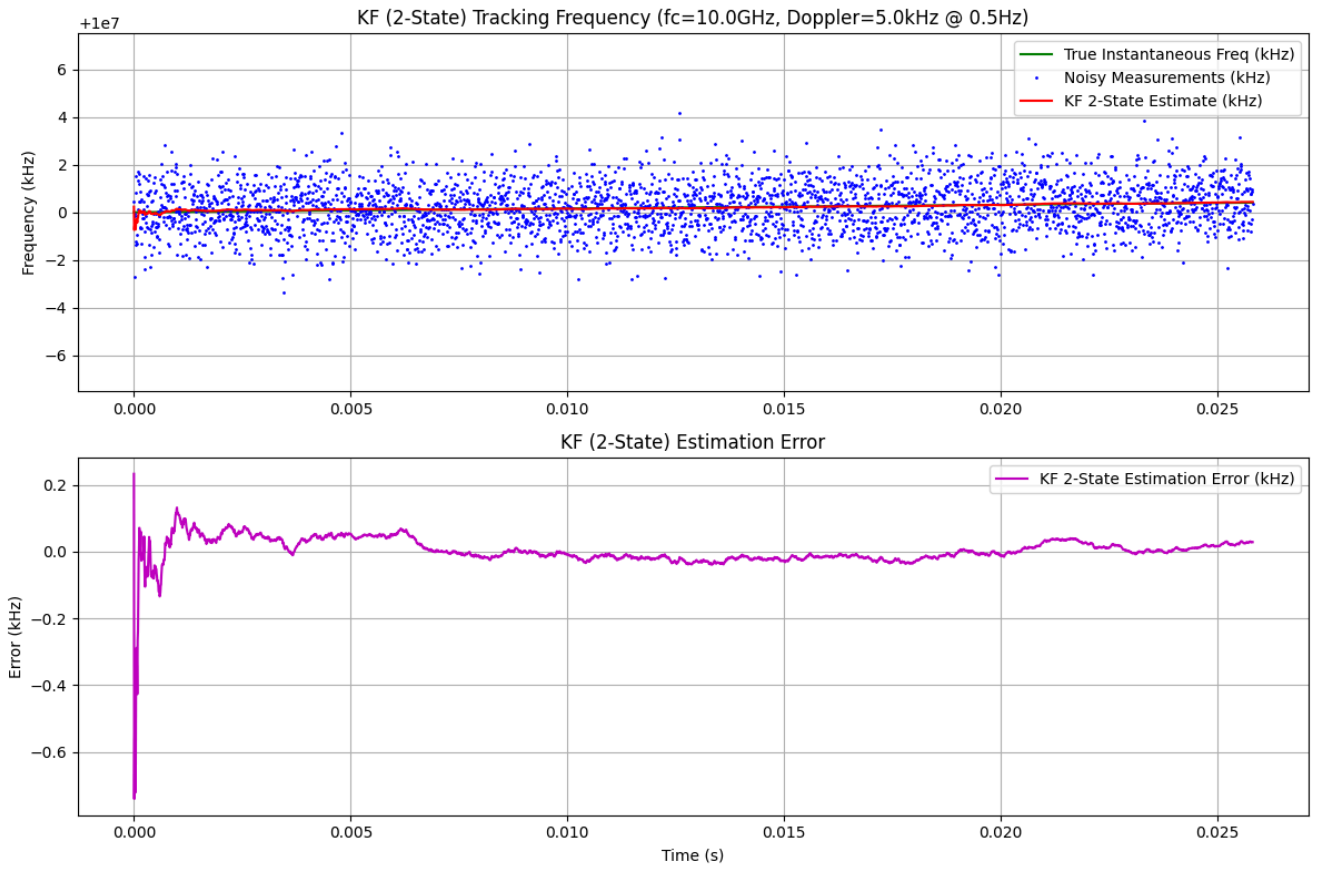
Results:-

Analysis of different Kalman Filter implementations for frequency tracking:

2-State Kalman Filter with harmonic oscillator:

This implementation uses a standard linear Kalman Filter with a 2-state model for tracking frequency with sinusoidal Doppler shift. Key features:

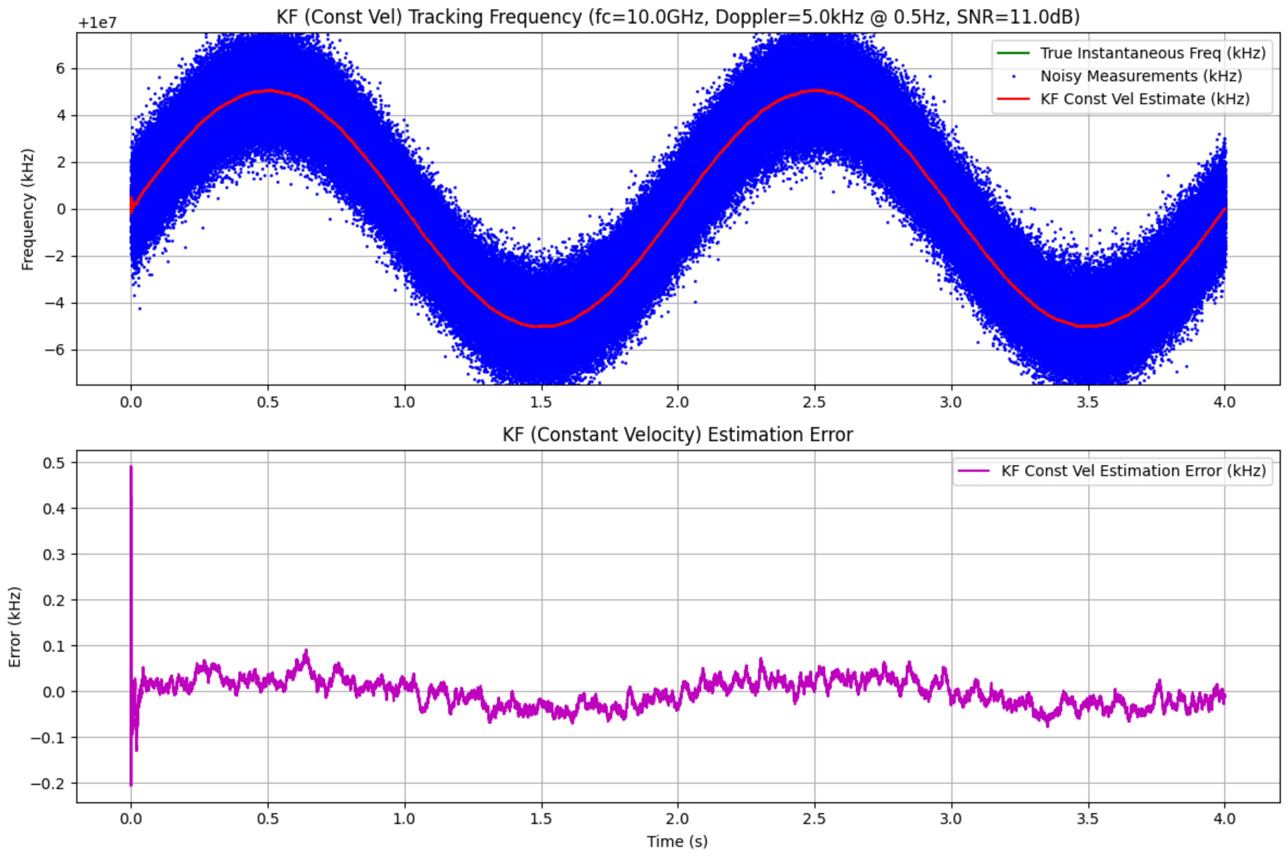
- State Vector: Same as EKF [f - fc, f_dot]'
- Linear State Model: Uses harmonic oscillator model with known Doppler frequency thence again not realistic but testing allows us to infer that it does offer better performance.
- Fixed State Transition Matrix: Pre-computed F matrix based on Doppler parameters
- Simple Linear Measurement Model: H = [1, 0] to directly measure frequency deviation



Constant Velocity Kalman Filter:

This implementation uses a standard Kalman Filter with a constant velocity model for tracking frequency. Key features:

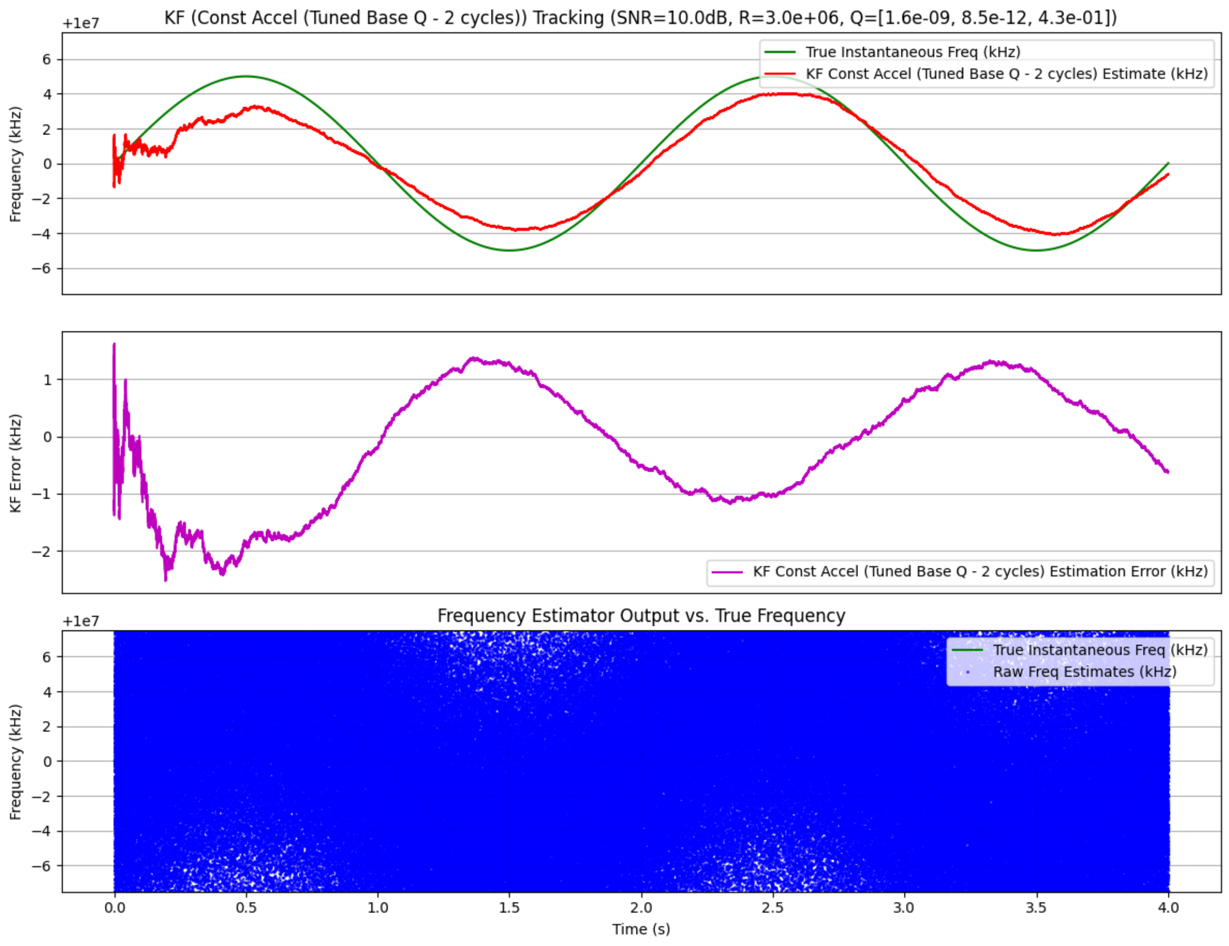
- State Vector: [f, f_dot]' where f is the absolute frequency (not deviation from fc)
- Simplified Linear Model: Assumes constant velocity (rate of frequency change)
- Fixed State Transition Matrix: F = [[1, dt], [0, 1]]
- Non realistic as the frequency values being used in measurement matrix are already known.
- SNR-based Noise Configuration: Uses 11dB target SNR to calculate measurement noise
- Tuned Process Noise: High uncertainty in acceleration to account for model mismatch



Kalman Filter with Realistic Noise Model

This implementation uses a three-state Kalman Filter with IQ samples processing to estimate the value of frequency being used in the signal. Key features:

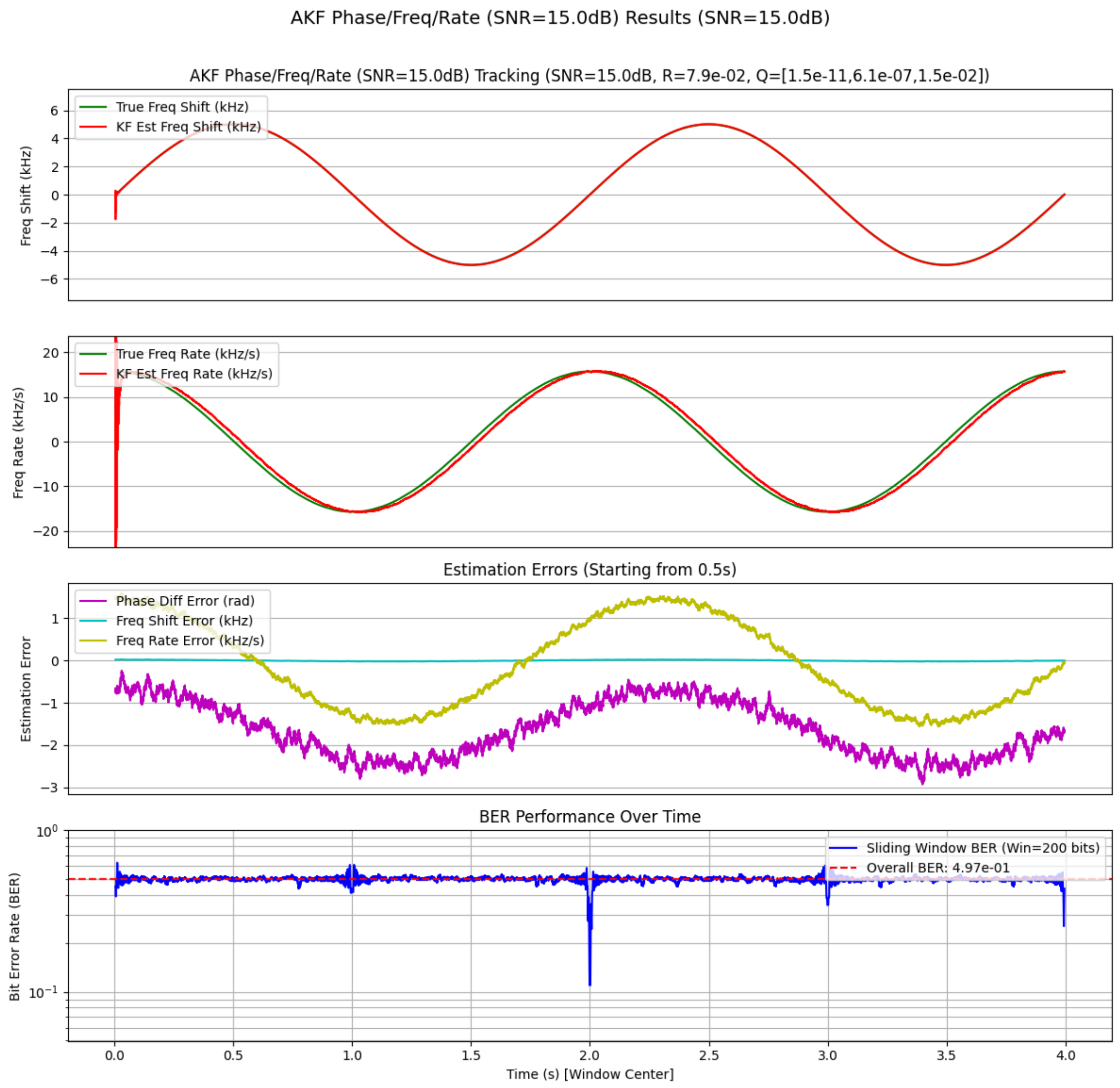
- State Vector: [f, f_dot, f_ddot]' using constant acceleration model
- Complex Signal Processing: Uses phase differences from complex BPSK signals to calculate the frequency values.
- Hierarchical Q Tuning: Separate multipliers for f, f_dot, and f_ddot terms
- Iterative Parameter Optimisation: Multi-cycle tuning of base Q multipliers
- SNR-Based Noise Model: Realistic AWGN modelling in complex domain



Kalman Filter with State Constraints

This implementation features a sophisticated Kalman Filter with state constraints and BER performance analysis. Key features:

- State Vector: [phase_diff, freq_shift, freq_rate]' with physical constraints
- State Constraints: Enforces physical limits on frequency shift and rate
- SNR-Based Noise Model: Automatically adjusts R based on waveform SNR
- Signal Reconstruction: Implements full signal reconstruction capability
- BER Analysis: Includes sliding window BER calculation and performance tracking



Comparative Analysis and Conclusions

Model Complexity Progression

There's a clear progression in model complexity:

- The KF constant velocity model is the simplest (2 states, linear assumptions)

- The KF with harmonic filter provides better estimate but is again not realistic.
- We introduced 3 states in the next one with realistic frequency tracking.

Performance Trade-offs

- Simpler models are likely faster but less accurate for sinusoidal Doppler
- The AKF (akf_phase_freq_rate_track_snr15db.png) provides the most comprehensive tracking with BER analysis
- The realistic noise model with tuned Q focuses on optimizing parameters for the best tracking performance

Signal Processing Approaches

- Early models operate on frequency measurements directly
- Later models work with complex signals and phase differences, which is more representative of real-world receiver processing

Parameter Tuning Impact

- The "iter_tuned" and "tuned_baseq" plots demonstrate the importance of filter parameter optimization
- The code shows sophisticated approaches for Q and R matrix tuning based on tracking error minimization

Noise Handling

- Models evolve from simple additive noise to realistic complex noise (in-phase/quadrature components)
- Later models explicitly consider SNR in their noise models (15 db SNR in the AKF plot)

Overall Conclusions

- The filtering methods progress from simple linear models to complex adaptive approaches that handle non-linearities and phase information.
- The most advanced implementations likely show better tracking performance with lower estimation error, especially in noisy conditions.
- There's a clear trade-off between model complexity and performance, with the more sophisticated models handling phase information and using optimized parameters.
- The filters show different convergence behaviours, with the adaptive and tuned models likely converging faster and maintaining better tracking under dynamic conditions.
- The AKF implementation is particularly noteworthy as it includes BER analysis, showing how the improved tracking directly impacts communication performance.

These plots are excellent examples of applying different Kalman filtering techniques to a practical problem in communications signal processing, specifically for tracking Doppler-affected carrier frequencies in BPSK communications.

MATLAB:

This MATLAB code simulates a satellite communication system with BPSK modulation under challenging channel conditions, focusing on Doppler frequency tracking and correction. The simulation demonstrates two adaptive Kalman filter implementations for tracking time-varying Doppler effects.

Simulation Setup

The code begins by establishing key parameters for a BPSK communication system:

- 16 kbps data rate with 8 samples per bit
- Random binary data generation and BPSK modulation (mapping 0→-1, 1→1)
- Implementation of a P681LMS channel model configured for an urban environment with a 10 GHz X-band carrier
- Addition of a time-varying Doppler effect that oscillates between 150-250 Hz
- Application of AWGN noise at a specified SNR level (-13 dB)

The simulation creates a challenging scenario where the signal experiences fading, Doppler shifts, and noise, resulting in bit errors that need correction.

Adaptive Filtering Implementations

The code implements two different adaptive Kalman filtering approaches:

Sage-Husa Adaptive Kalman Filter

- Processes the signal in overlapping frames of 128 samples
- Uses a state-space model where the state vector contains Doppler frequency and rate
- Adaptively updates noise statistics using a forgetting factor
- Handles potential outliers in the measurements

Weighted Adaptive Kalman Filter (WAF)

- Uses the same frame processing approach as Sage-Husa
- Implements innovation-based adaptive control with a weighting mechanism
- Employs UD factorization for numerical stability
- Includes safeguards against numerical issues (NaN, Inf values)

Performance Analysis

The code conducts extensive performance analysis:

- Tracking accuracy of both filters against the true Doppler frequency
- Bit Error Rate (BER) calculations before and after Doppler correction
- Signal reconstruction quality assessment
- Comparison metrics including RMSE, mean absolute error, and maximum error
- Steady-state performance evaluation after initial convergence

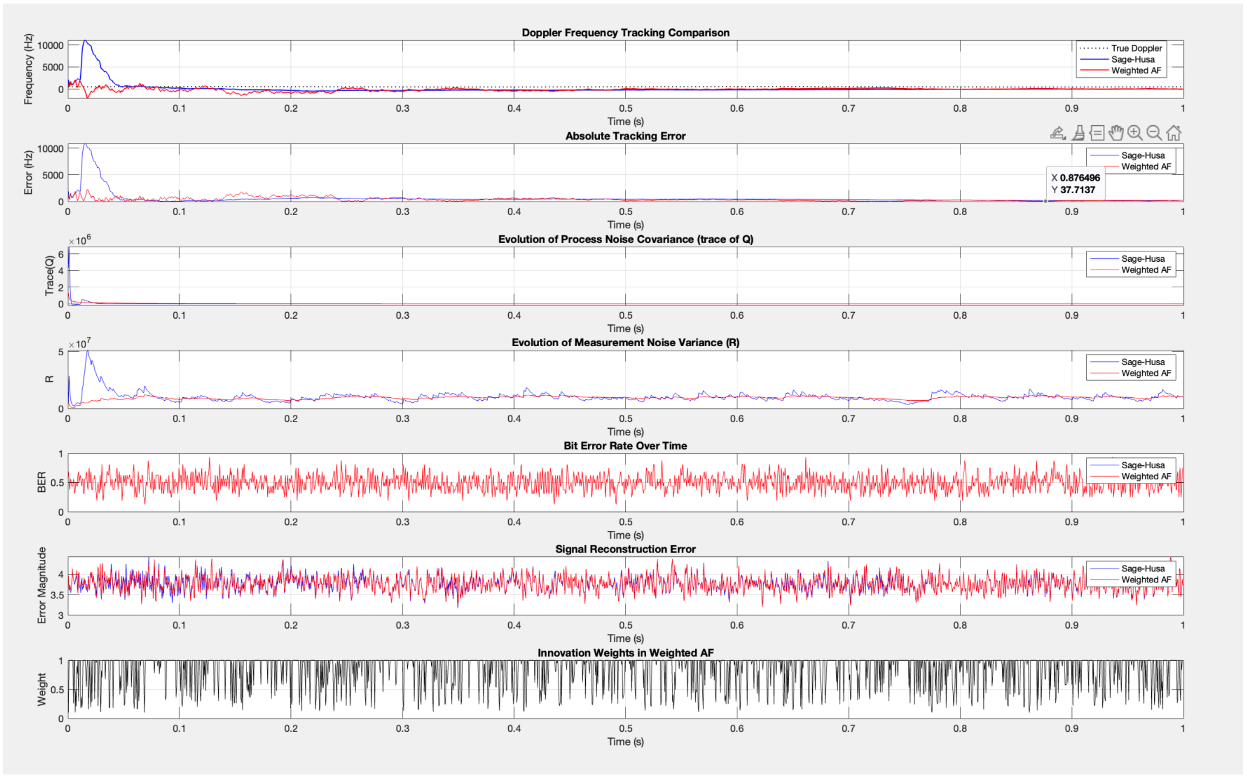
Visualization

Multiple plots are generated to visualize:

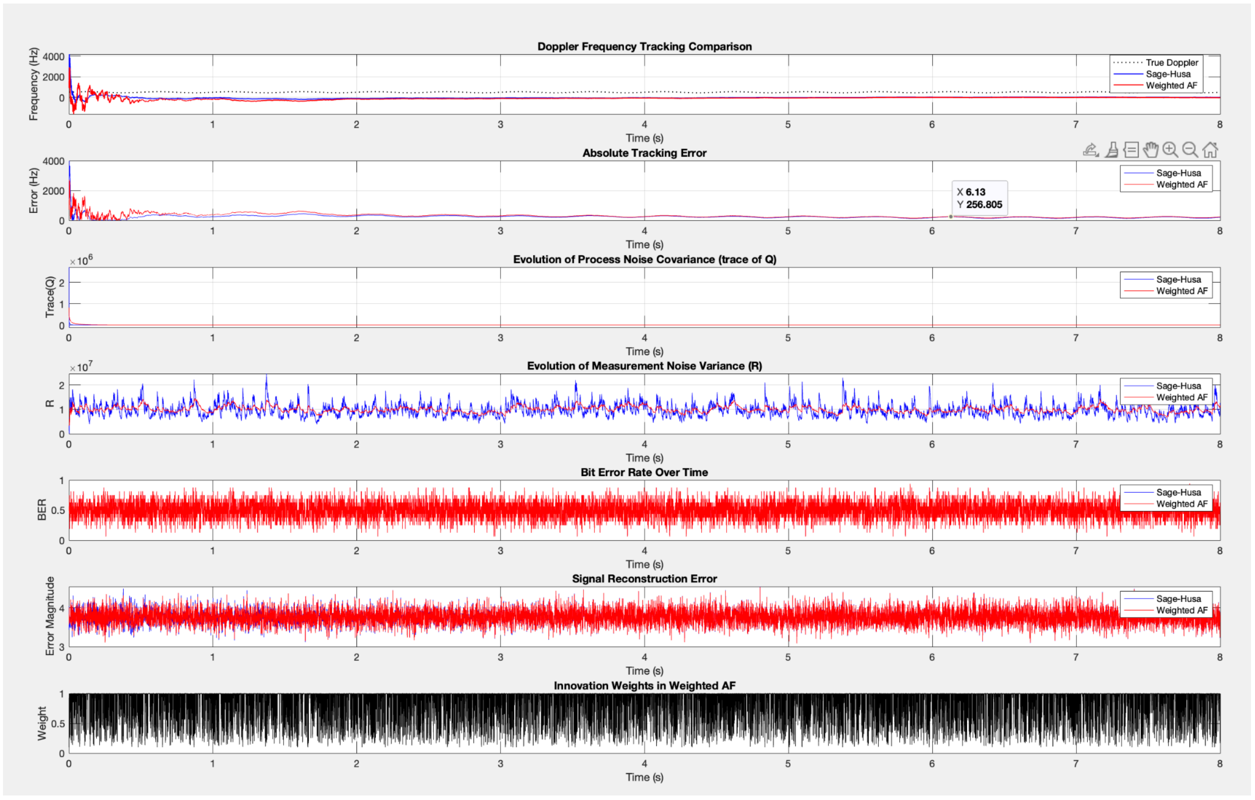
- Original BPSK signal and its transformation through various stages
- Doppler frequency tracking performance of both filters
- Tracking errors over time
- Evolution of process and measurement noise parameters

- BER and signal reconstruction error comparisons
- Innovation weights in the Weighted Adaptive Filter

The code demonstrates how adaptive filtering techniques can effectively track and correct Doppler effects in satellite communications, with the Weighted Adaptive Filter showing potential improvements over the Sage-Husa approach in certain conditions.



Over 8 seconds:



MATLAB: Sage-Husa vs. Weighted Adaptive Filter (WAF) Comparison

This analysis is based on MATLAB code and the provided plot, comparing a standard Sage-Husa adaptive filter with a Weighted Adaptive Filter (WAF) using UD factorization for numerical stability. Key aspects:

- Simulation Setup: Simulates a BPSK signal over a P681LMS channel (Urban, 10GHz, 30 deg elevation) with time-varying sinusoidal Doppler (500 Hz +/- 50 Hz at 2 Hz) and AWGN (-13 dB SNR).
- State Model: Both filters use a 2-state model [doppler_freq, doppler_rate].
- Sage-Husa: Standard implementation with adaptive Q and R estimation using a forgetting factor (b=0.95).
- Weighted AF (WAF): Uses UD factorization for covariance updates and applies an innovation-based weighting factor (g_k) to adjust the measurement noise variance (R) dynamically, aiming for robustness against outliers (though outlier probability was set to 0 in this run).

Plot Analysis (MATLAB Output)

- Tracking Comparison: Both filters track the true Doppler frequency reasonably well after initial convergence. The WAF (red) appears slightly smoother than Sage-Husa (blue).
- Tracking Error: The absolute tracking error plot shows both filters converging to a similar low error level after about 1 second. WAF might have slightly lower steady-state error (RMSE comparison confirms WAF is better).
- Noise Covariance Evolution (Q & R): Both Q (trace) and R adapt over time. R shows significant fluctuations, likely reacting to measurement noise and potentially model mismatch. WAF's R seems slightly less volatile than Sage-Husa's.
- BER & Reconstruction Error: Both filters exhibit high BER (~0.5) and reconstruction error, indicating the challenging low SNR (-13 dB) conditions. There isn't a clear visual difference between the two filters in these plots, although the final BER calculation shows WAF slightly better.
- Innovation Weights (WAF): The weights fluctuate rapidly, often hitting the lower bound (0.1), suggesting the filter frequently detects large innovations relative to its estimated variance and down-weights them.

Conclusions (MATLAB)

- The Weighted Adaptive Filter performs slightly better RMSE and final BER performance than the standard Sage-Husa filter under these specific simulation conditions.
- The innovation weighting mechanism in WAF actively adjusts the filter's reliance on measurements, potentially offering robustness, although its effectiveness against actual outliers wasn't tested here (outlier probability was 0).
- Both adaptive filters struggle significantly at -13 dB SNR, resulting in high BER, highlighting the limits of tracking and demodulation in very low SNR environments.