Introduction to Kalman Filter

Aamir Mahmood
aamir.mahmood@miun.se

Mid Sweden University

August 15, 2024

Acknowledgments

- The lecture slides are mainly based on the book, *Kalman Filter from the Ground Up* by *Alex Becker*.
- The figures in these slides are primarily sourced from *Kalman Filter from the Ground Up*, except for those generated from my own implementation of the codes.
- The codes and the results generated are my own contributions and were developed specifically for these lecture slides.
- Additional information in the slides has been compiled from various sources.

Outline I

Outline II

Outline III

- Kalman Filter Extensions (Non-linear Kalman Filters)
- Extended Kalman Filter (EKF)
  - Analytic linearization
  - First-order Taylor series expansion
  - Uncertainty Projection
  - EKF Equations
  - Example 11 – vehicle location estimation using radar
  - Example 12 - estimating the pendulum angle
  - Limitations of EKF
- Unscented Kalman Filter (UKF)
  - The Unscented Transform (UT)
  - The UKF algorithm: Prediction and Update Stages
  - Example 13 – vehicle location estimation using radar
  - Sigma Point Algorithm Modification
  - Example 14 - Estimating the pendulum angle
  - Non-linear filters comparison: EKF vs UKF

5. Part 4: Kalman Filter in practice
   - Sensor Fusion
     - Combining measurements in one dimension
     - Combining n measurements
     - Combining measurements in k dimensions
     - Sensor data fusion using Kalman filter
     - Multirate Kalman Filter
   - Variable Measurement Error
   - Treating missing measurements
   - Treating outliers: Identification and Treatment
   - Kalman Filter Initialization

Outline IV

- KF Development Process
  - Kalman Filter Design
  - Simulation
  - Performance Examination

On Rudolf E. Kalman

- Born in Budapest, Hungary, May 19, 1930
- Passed away in Gainsville, Florida, July 2, 2016
- Emigrated to USA in 1943
- Bachelor and Master in EE at MIT, 1953 and 1954, respectively
- Ph.D. at Columbia University, 1957 (Advisor: J. R. Ragazzini)
- Institutions: Stanford University, University of Floria, ETC Zurich (and more)

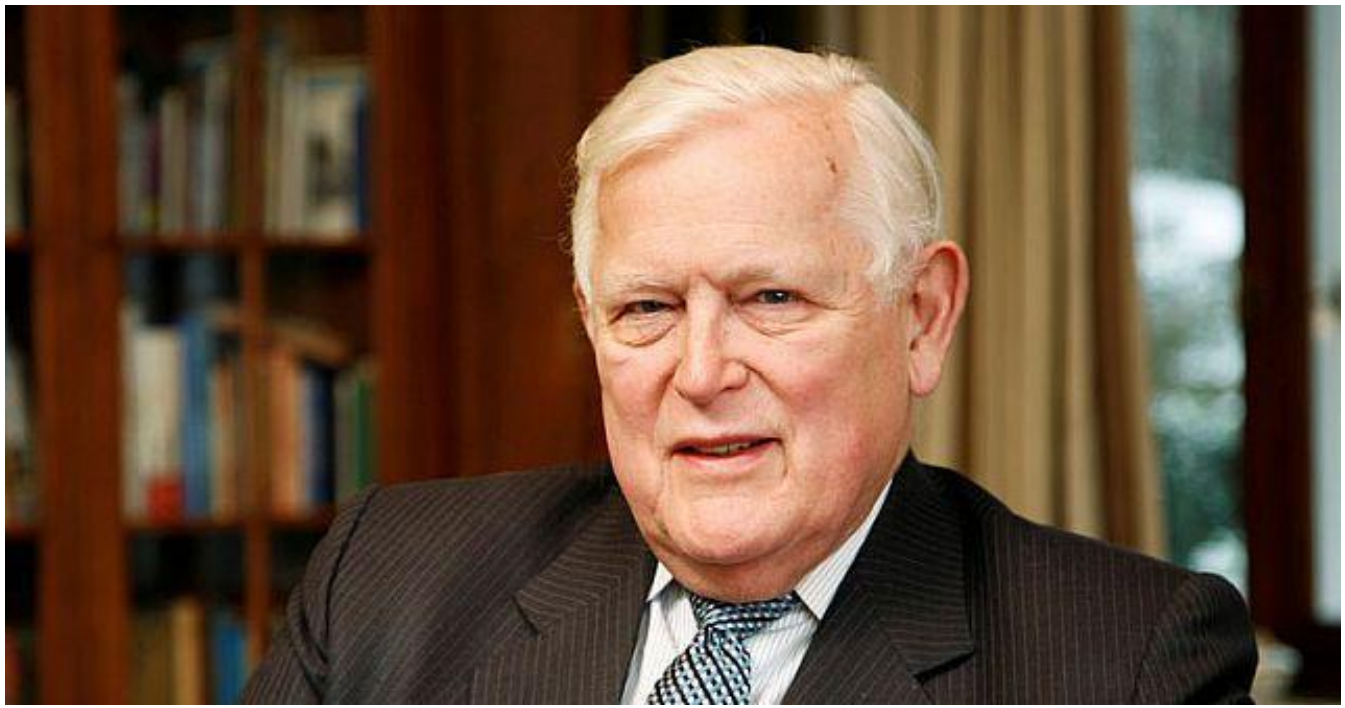


Figure: Rudolf E. Kalman

This is where it all started: [1]

The Filter

- The filter invented by R. E. Kalman in the 1960s, based on the seminal papers:
  1. R. E. Kalman: Contribution to the Theory of Optimal Control (1960)
  2. **R. E. Kalman: A New Approach to Linear Filtering and Prediction Problems (1960)**
  3. R. E. Kalman: Mathematical Description of Linear Dynamical Systems (1963)
- (2) describes a recursive solution to the discrete-data linear filtering problem—Kalman Filter
- Enormous impact on the fields of **linear systems theory, statistics, signal processing, identification, feedback control, and adaptive systems**
- Additionally, on motivating new formulations of feasible filtering problems outside the linear domain.
- Kalman Filtering is used extensively, mainly for *target tracking*. However, it can be applied to any field where *estimation and prediction* are required, e.g., location and navigation systems, control systems, computer graphics, and much more.

Estimation problem: Estimating hidden (unknown) states based on a series of measurements.

- For example, a GPS receiver provides location and velocity estimation, where location and velocity are the hidden states and differential time of the satellites' signals' arrival are the measurements.
- Challenge is providing an accurate and precise estimation of the hidden states in the **presence of uncertainty**.
  - In GPS receivers, the measurement uncertainty depends on many external factors such as thermal noise, atmospheric effects, slight changes in satellite positions, receiver clock precision, etc.
- The Kalman Filter produces estimates of hidden variables based on inaccurate and uncertain measurements.

Filtering

Estimate the current value of the stochastic signal, using the history and current value of another observed stochastic process; e.g., using the measurements $Y_k = \{y_i, i \leq k\}$, we want to estimate $\hat{x}_{k+m}$. It leads to three cases:

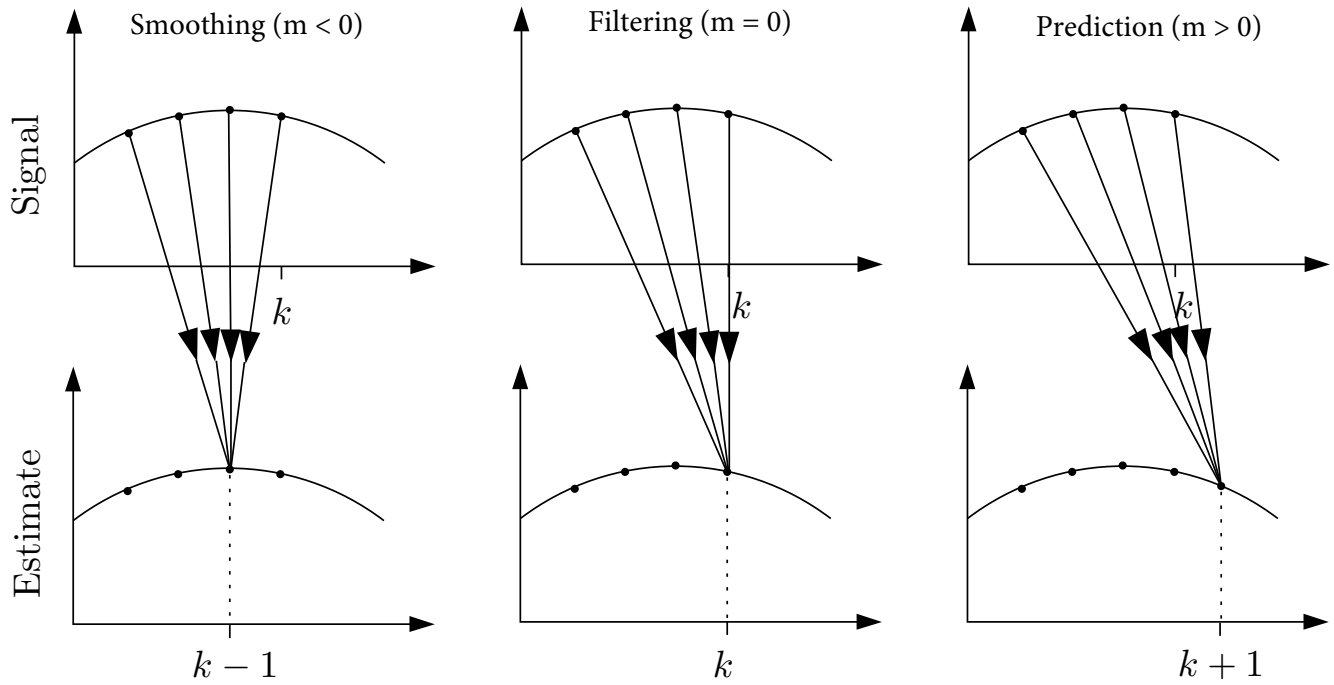


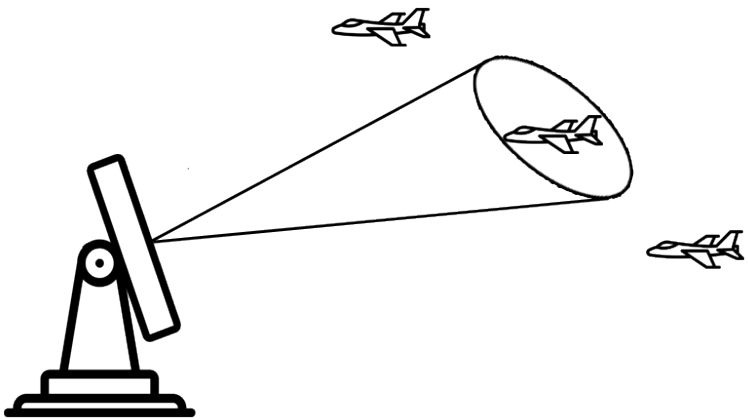
Figure: Smoothing, filtering, prediction

- **Smoothing**: Estimation of the past state-variable at the present time
- **Filtering**: Estimating the present state-variable at the present time
- **Prediction**: Estimating the future state-variable at the present time

The Necessity of Prediction

Radar tracking algorithm

- The tracking radar sends a pencil beam in the direction of the target.
- In every track cycle of Δt, the radar revisits the target by sending a dedicated track beam in the direction of the target.
- After sending the beam, the radar estimates the current target position and velocity. The radar also estimates (or predicts) the target position at the next track beam.



- The future target position can be easily calculated using Newton's motion equations:

$$x = x_0 + v_0 \Delta t + \frac{1}{2} a \Delta t^2$$

where
x is the target's position
x_0 is the target's initial position
v_0 is the target's initial velocity
a is the target's acceleration
Δt is the track cycle

In 3D, it can be written as a system of equations

$$\begin{aligned} x &= x_0 + v_{x0} \Delta t + \frac{1}{2} a_x \Delta t^2 \\ y &= y_0 + v_{y0} \Delta t + \frac{1}{2} a_y \Delta t^2 \\ z &= z_0 + v_{z0} \Delta t + \frac{1}{2} a_z \Delta t^2 \end{aligned}$$

The above set of equations is called a **Dynamic Model** (or a **State Space Model**). The Dynamic Model describes the relationship between input and output.

Constant Acceleration

$$y = \int v dt$$

$$= \int (v_0 + at) dt$$

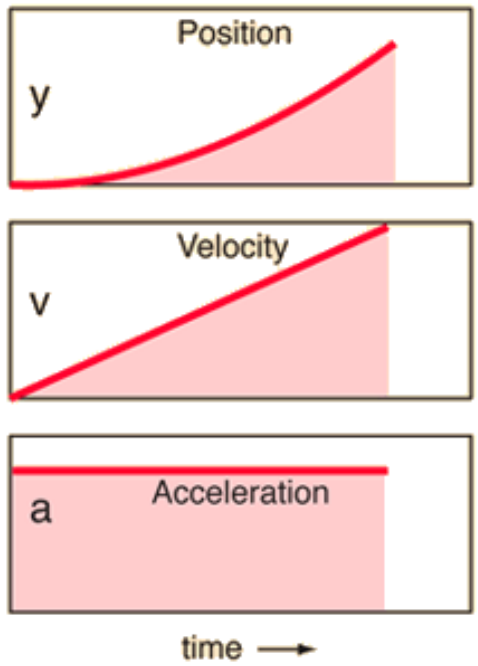
$$y = y_0 + v_0 t + \frac{1}{2} a t^2$$

Integrate velocity to get position

$$v = \int a dt = v_0 + at$$

Integrate acceleration to get velocity

$$a = \text{constant}$$



Motion relationships in one dimension.

$$y = y_0 + v_0 t + \frac{1}{2} a t^2$$

Derivative of position is velocity

$$v = \frac{dy}{dt}$$

$$v = v_0 + at$$

Derivitive of velocity is acceleration

$$a = \frac{dv}{dt} = a$$

Terminologies using tracking example

- **Dynamic or State Space Model:** Describes the relationship between input and output.
- **System State**
  - The target parameters $[x, y, z, v_x, v_y, v_z, a_x, a_y, a_z]$ are called a **System State**.
  - The current state is the input to the prediction algorithm and the next state (the target parameters at the next time interval) is the algorithm's output.
- So, if the current state and the dynamic model are known, the next target state can be easily predicted? **Well, not really!**
- **Measurement Noise:** The radar measurement is not absolute; it includes a random error (or uncertainty). The error magnitude depends on many parameters, such as radar calibration, the beam width, and the signal-to-noise ratio of the returned echo. *The error included in the measurement is called Measurement Noise.*
- **Process Noise:** The target motion is not strictly aligned to motion equations due to external factors such as wind, air turbulence, and pilot maneuvers. *The dynamic model error (or uncertainty) is called Process Noise.*

Due to Measurement Noise and Process Noise, the estimated target position can be far away from the actual target position. In this case, the radar might send the track beam in the wrong direction and miss the target.

Part 1: The Kalman Filter for Dummies

Objectives

- Understand the concept of the Kalman Filter and develop "Kalman Filter intuition"
- Being able to design a one-dimensional Kalman Filter
- *While*, developing necessary mathematical background using practical numerical examples with easy and intuitive explanations

"The road to learning by precept is long, by example short and effective."

— Lucius Seneca

Mean and Expected Value // Variance and Standard Deviation

- **Mean (μ):** Find mean of 5 coins, e.g.,
  $V_{\text{mean}} = \frac{1}{N}\sum_1^N V_n = \frac{1}{5}(5+5+10+10+10) = 8\,\text{cent}$ (no hidden system states)
- **Expected value (E)**: The expected value is the value you would expect your hidden variable to have over a long time or many trials. Example: Different weight measurements of the same person.
  - Person is the system, person's weight is the system state.
  - The measurements will be different due to random measurement errors of the scale.
  - True weight is unknown since it is a **Hidden State**. To estimate the weight: $W = \frac{1}{N}\sum_1^N W_n$
- **Variance (σ^2):** is a measure of the spreading of the data set from its mean.

$$\sigma^2 = \frac{1}{N}\sum_1^N (x_n - \mu)^2$$

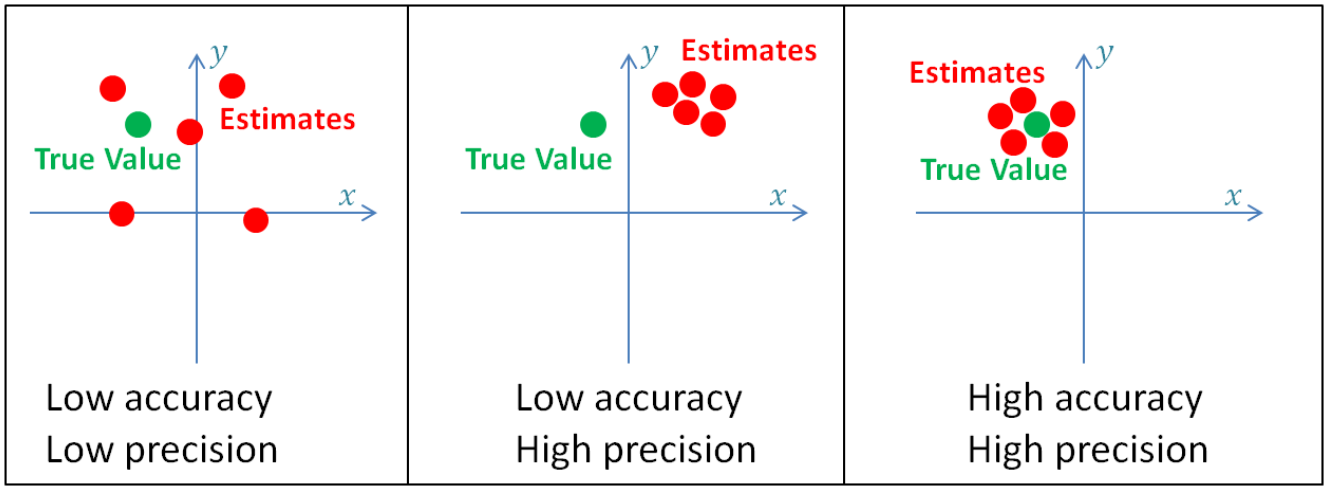
- **Standard deviation (σ)** is the square root of the variance.
- When estimating variance, with factor $N-1$ is called Bessel's correction:

$$\sigma^2 = \frac{1}{N-1}\sum_1^N (x_n - \mu)^2$$

See proof at visiondummy.

Estimation, Accuracy, Precision

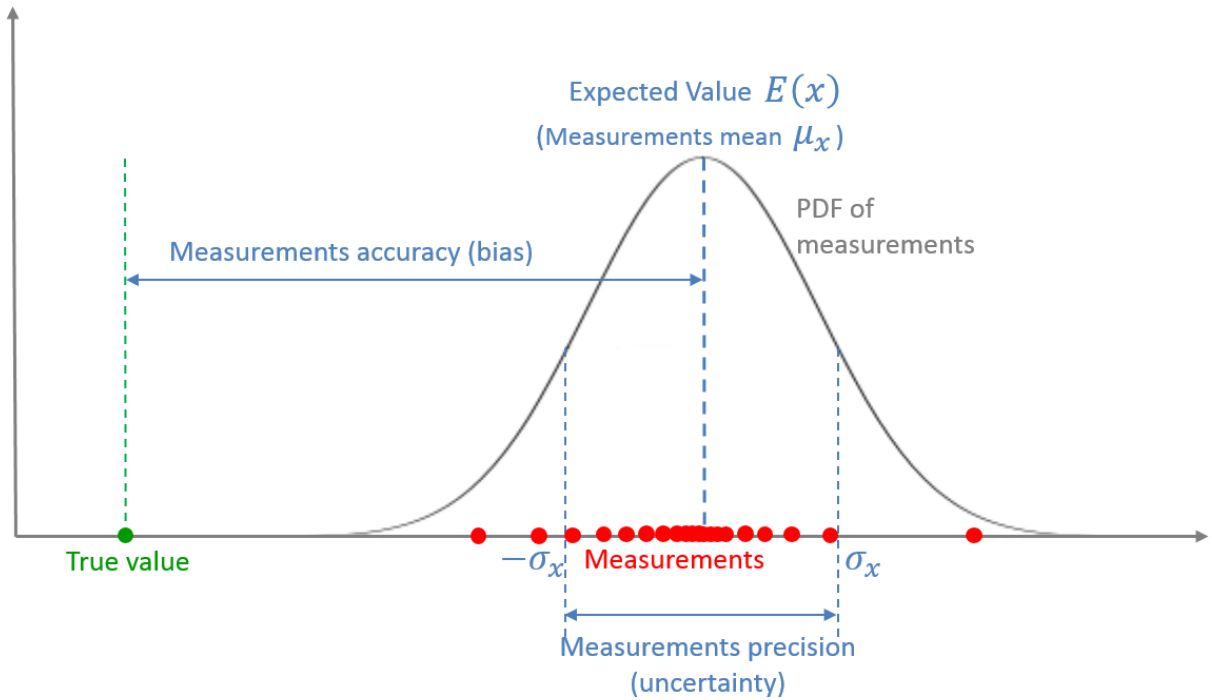
- **Estimate:** Evaluating the hidden state (e.g., aircraft's true position) using sensor(s) (i.e., radar) measurements
- **Accuracy:** indicates how close the measurement or estimate is to the true value
- **Precision** describes the variability of measurements or estimate of the parameter of interest



High-precision systems have low variance in their measurements (i.e., low uncertainty) and vice versa. *The random measurement error produces the variance.*
Q: How to reduce the influence of variance (A: Averaging or smoothing)

While, low-accuracy systems are called **biased** systems as their measurements have a built-in systematic error or bias. *A biased thermometer will produce a constant systematic error in the estimate.*

Estimation, Accuracy, Precision: Summary



The offset between the measurement's mean and the true value is the measurement's accuracy, also known as bias or systematic measurement error.

The dispersion of the distribution is the measurement precision, also known as the measurement noise, random measurement error, or measurement uncertainty.

$\alpha - \beta - \gamma$ Filter

Introduction to $\alpha - \beta$ and $\alpha - \beta - \gamma$ filters, which are frequently used for time series data smoothing. The principles of $\alpha - \beta(-\gamma)$ are closely related to Kalman Filter principles.

Example 1: Weighting the Gold (A Static System)

Objective: Estimate static system's state, e.g., estimate a gold bar's weight

- The scale is unbiased—the measurements don't have a systematic error but the random noise.
- The system's dynamic model is constant, i.e., $\boxed{\hat{x}_{n+1,n} = \hat{x}_{n,n}}$
- To estimate the system's state (i.e., the weight value), we can take multiple measurements and average them.
- At the time n, the estimate $\hat{x}_{n,n}$ is

$$\hat{x}_{n,n} = \frac{1}{n}\sum_{i=1}^{n} z_i$$

Example notations:

x is the true value of the weight

z_n is the measured value of the weight at time n

$\hat{x}_{n,n}$ is the estimate of x at time n (the estimate is made after taking the measurement z_n)

$\hat{x}_{n+1,n}$ is the estimate of the future state $(n+1)$ of x. The estimate is made at the time n.

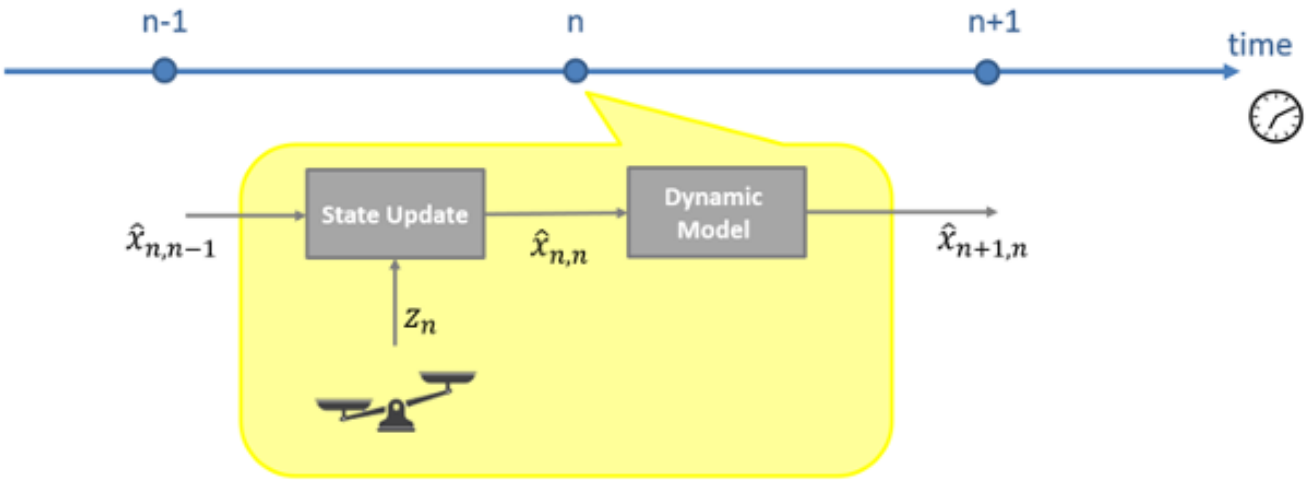
In other words, $\hat{x}_{n+1,n}$ is a predicted state or extrapolated state

$\hat{x}_{n-1,n-1}$ is the estimate of x at time $n-1$ (the estimate is made after taking measurement z_{n-1})

$\hat{x}_{n,n-1}$ is the previous prediction—estimate of the state at time n, made at time $n-1$

State Update Equation

It is more practical to keep the last estimate only ($\hat{x}_{n-1,n-1}$) and update it after every new measurement.



- Estimate the current state based on the measurement and prior prediction.
- Predict the next state based on the current state estimate using the Dynamic Model.

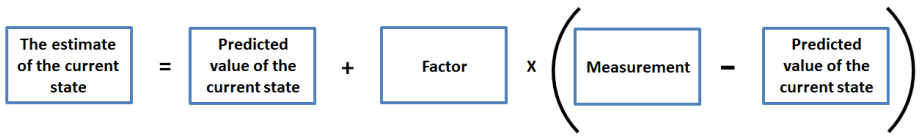
State Update Equation

$$\begin{aligned}
\hat{x}_{n,n} &= \frac{1}{n}\sum_{i=1}^{n} z_i \\
&= \frac{1}{n}\left(\sum_{i=1}^{n-1} z_i + z_n\right) \\
&= \frac{n-1}{n}\left(\frac{1}{n-1}\sum_{i=1}^{n-1} z_i\right) + \frac{1}{n} z_n \\
&= \hat{x}_{n-1,n-1} - \frac{1}{n}\hat{x}_{n-1,n-1} + \frac{1}{n} z_n \\
&= \hat{x}_{n-1,n-1} + \frac{1}{n}\left(z_n - \hat{x}_{n-1,n-1}\right)
\end{aligned}$$

Using the system's dynamic model, we can extrapolate $\hat{x}_{n-1,n-1}$ to predict the state of x at the time n, i.e., $\hat{x}_{n,n-1}$. For a static system, we have $\hat{x}_{n,n-1} = \hat{x}_{n-1,n-1}$. Thus,

$$\boxed{\hat{x}_{n,n} = \hat{x}_{n,n-1} + \frac{1}{n}\left(z_n - \hat{x}_{n,n-1}\right)} \qquad (1)$$

Eq. (1) is one of the five Kalman Filter equations, called as State Update Equation, and described as



- In Kalman Filter, the factor $\frac{1}{n}$ (specific to this example) is called as Kalman Gain, denoted by K_n.
- The term $(z_n - \hat{x}_{n,n-1})$ is the measurement residual, also called as the INNOVATION (containing new information).
- $1/n$ decreases as n increases $\rightarrow$ each successive measurement has less weight in the estimation process.
- Using $\alpha_n = 1/n$ in our example,

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha_n\left(z_n - \hat{x}_{n,n-1}\right)$$

Estimation Algorithm in Example 1

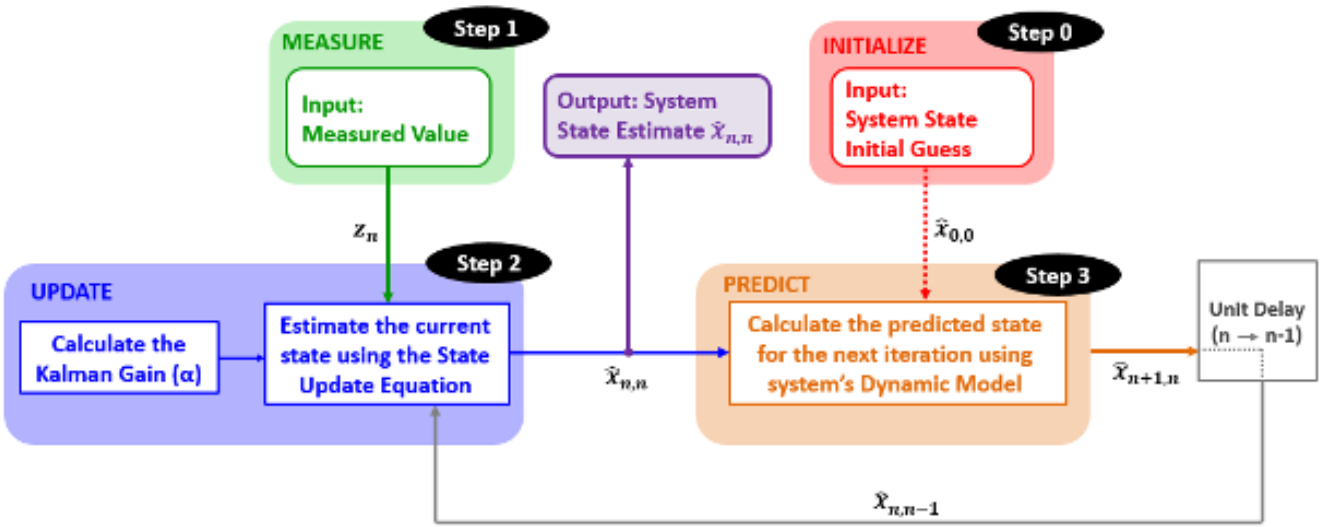


Figure: State Update Equation.

Iteration Zero

- **Initialization:** We start by making a guess or rough estimate of the gold bar weight. It is called the Initial Guess for the filter initialization.

$$\hat{x}_{0,0} = 1000\,g$$

- **Prediction:** As the dynamic model of the system is static, our next state estimate (prediction) equals initialization, i.e.,

$$\hat{x}_{1,0} = \hat{x}_{0,0} = 1000\,g$$

Estimation Algorithm in Example 1

Iteration 1

- **Step 1:** Making the weight measurement

$$z_1 = 1030g$$

- **Step 2:**
  Calculating the gain from $\alpha_n = 1/n$

$$\alpha_1 = 1/1 = 1$$

  Calculating the current estimate using the State Update Equation

$$\hat{x}_{1,1} = \hat{x}_{1,0} + \alpha_1 (z_1 - \hat{x}_{1,0}) = 1030g$$

- **Step 3:** The dynamic model of the system is static; thus, our next state estimate (prediction) equals to current state estimate

$$\hat{x}_{2,1} = \hat{x}_{1,1} = 1030g$$

Iteration 2

- After a unit time delay, the predicted estimate from the previous iteration becomes the prior estimate in the current iteration

$$\hat{x}_{2,1} = 1030g$$

- **Step 1:** Making the second measurement of the weight

$$z_2 = 989g$$

- **Step 2:**
  Calculating the gain

$$\alpha_2 = 1/2$$

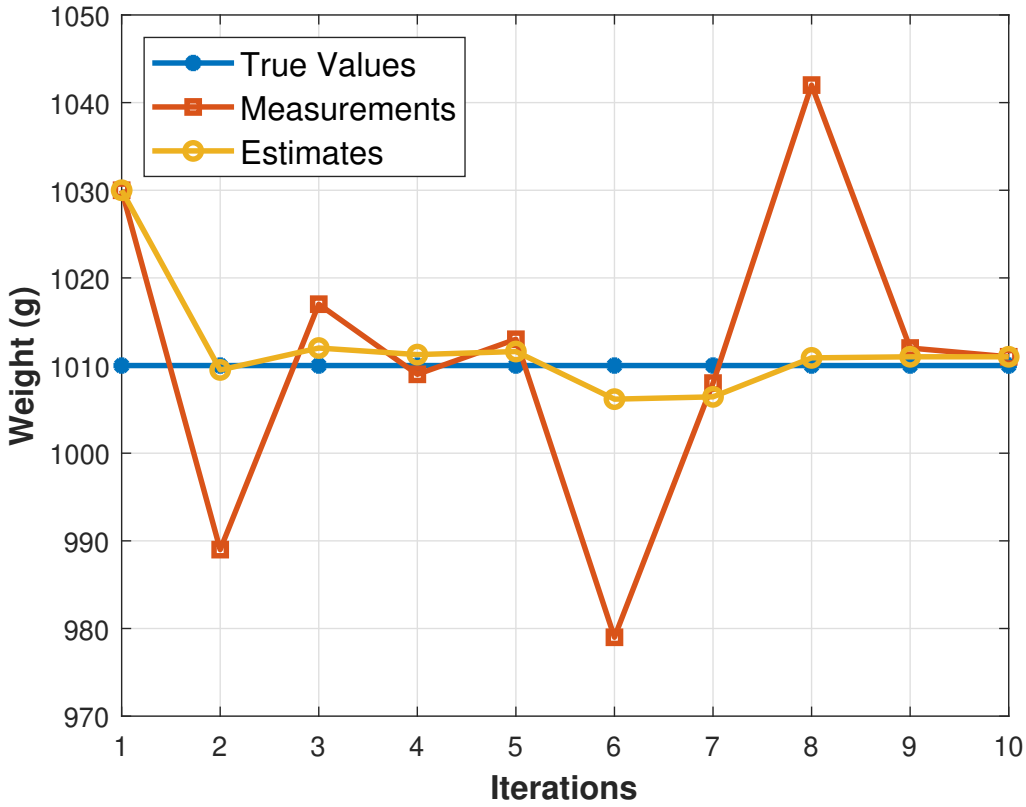
  Calculating the current estimate:

$$\hat{x}_{2,2} = \hat{x}_{2,1} + \alpha_2 (z_2 - \hat{x}_{2,1}) = 1009.5g$$

- **Step 3:**

$$\hat{x}_{3,2} = \hat{x}_{2,2} = 1009.5$$

Example 1: Results



- The gain decreases with each measurement. Therefore, the contribution of each successive measurement is lower than the contribution of the previous measurement.
- The estimation algorithm has a smoothing effect on the measurements and converges toward the true value.

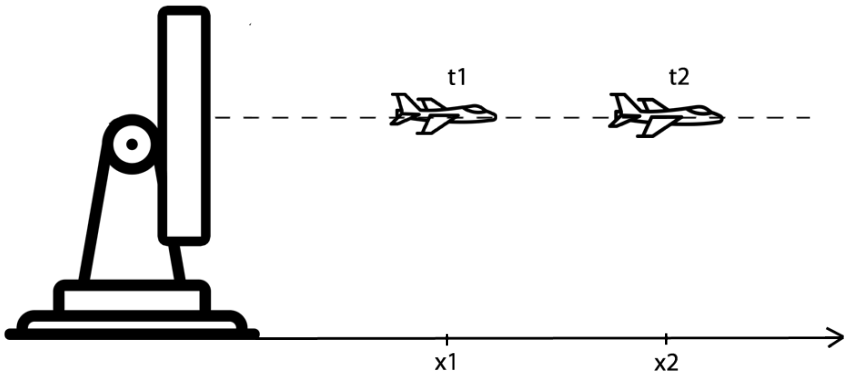
[Code: a-b-c Filter/Ex1_alpha_EstimationAlgorithm.m]

Example 2: Tracking the Constant Velocity Aircraft in 1D

Analyzing a dynamic system that changes over time. We track a constant velocity aircraft using the $\alpha - \beta$ filter.

Assumptions:

- Aircraft is moving radially away/toward the radar
- In 1D, the angle to the radar is constant, and the aircraft's altitude is constant



- x_n represents the range to the aircraft at time n.
- The velocity is a derivative of the range

$$\dot{x} = v = \frac{dx}{dt}$$

- The radar sends a track beam in the direction of the target at a constant rate, Δt.
- The system's dynamic model for constant velocity motion is

$$x_{n+1} = x_n + \Delta t \dot{x}_n, \qquad \text{[Aircraft range]}$$
$$\dot{x}_{n+1} = \dot{x}_n, \qquad \text{[Constant velocity]}$$

- This system of eqs. is called as State Extrapolation Equation or Transition Equation or Prediction Equation $\rightarrow$ Also, one of the Kalman Filter equations.

The $\alpha - \beta$ Filter

- Assume that at time $n-1$ the estimated range of the aircraft is 30,000m, and its estimated velocity is 40m/s.
- Using the system's dynamic model, the target position at time n is (with $\Delta t = 5\,s$)

$$\hat{x}_{n,n-1} = \hat{x}_{n-1,n-1} + \Delta t \dot{x}_{n-1,n-1} = 30200m$$

- The target velocity prediction for time n

$$\dot{x}_{n,n-1} = \dot{x}_{n-1,n-1} = 40m/s$$

- What if: the radar measures the range $z_n = 30,110$ instead. Two possibilities of the gap:
  - The radar measurements are not precise.
  - The aircraft velocity has changed.
- Let's update the velocity State Update Equation

$$\boxed{\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta \left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t} \right)}$$

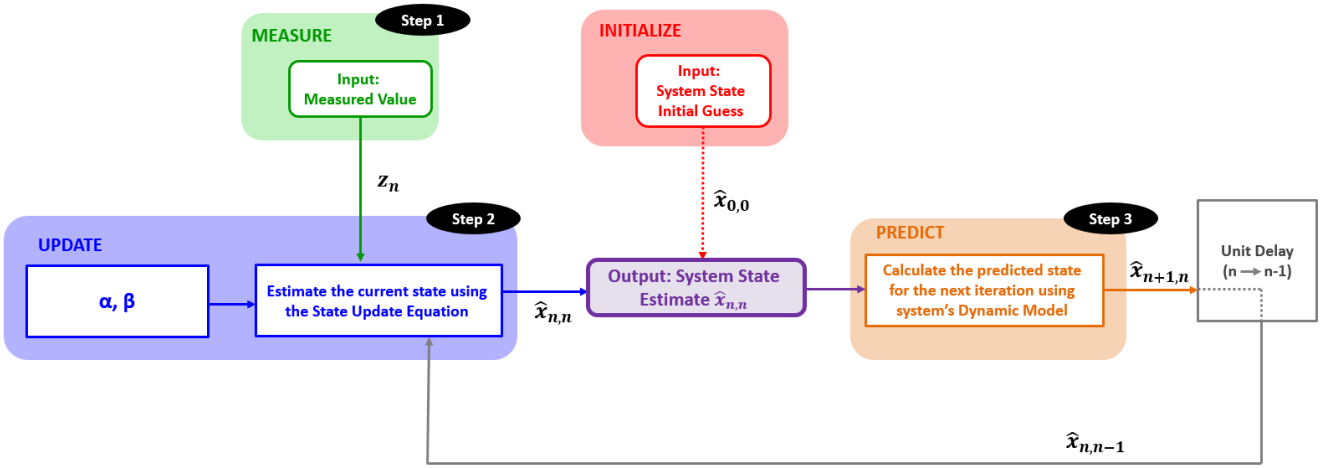
- β value depends on the precision level of the radar. If precision level is high, the range difference results from the change in velocity. Therefore, we should set high β. For $\beta = 0.9 \rightarrow \hat{\dot{x}}_{n,n} = 23.8m/s$
- On the other hand, if radar precision is low, the gap results from the radar measurement error. For $\beta = 0.1 \rightarrow \hat{\dot{x}}_{n,n} = 38.2m/s$
- The range State Update Equation for the aircraft position is (similar to Example 1)

$$\boxed{\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha \left(z_n - \hat{x}_{n,n-1} \right)}$$

- α-value depends on the radar measurement precision. For high precision radar, we should choose high α, giving high weight to the measurements, and lower otherwise.
- This system of equations, State Update Equations, are also called as $\alpha - \beta$ track update equations or track filtering equations.

Estimation Algorithm for Example 2

Parameters:
$\alpha = 0.2,$
$\beta = 0.1,$
$\Delta t = 5s$



Note: Unlike Example 1, the Gain values α and β are given. In the Kalman Filter, the α and β are replaced by **Kalman Gain**, which is calculated at each iteration.
Iteration Zero:

Initialization: Initial Guess:

$$\hat{x}_{0,0} = 30,000m$$

$$\hat{\dot{x}}_{0,0} = 40m/s$$

Prediction: Using the State Extrapolation Eqs.,
Range prediciton:

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t \hat{\dot{x}}_{n,n}$$

$$\hat{x}_{1,0} = \hat{x}_{0,0} + \Delta t \hat{\dot{x}}_{0,0} = 30200m$$

Velocity prediction:

$$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n}$$

$$\hat{\dot{x}}_{1,0} = \hat{\dot{x}}_{0,0} = 40m/s$$

Results for Example 2

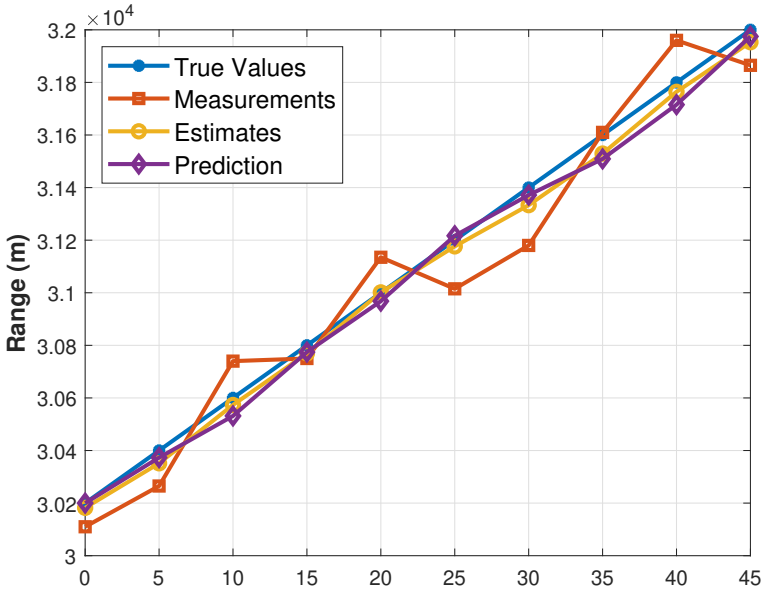


Figure: $\alpha = 0.2, \beta = 0.1$

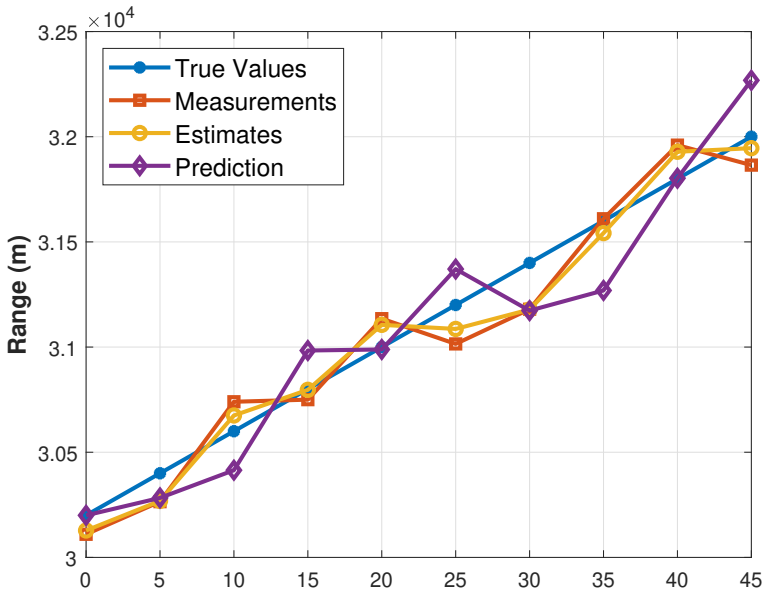


Figure: $\alpha = 0.8, \beta = 0.5$

The estimation algorithm has a smoothing effect on the measurements and converges toward the true value.

"Smoothing" degree is much lower. The "current estimate" is very close to the measured values, and predicted estimate errors are high.

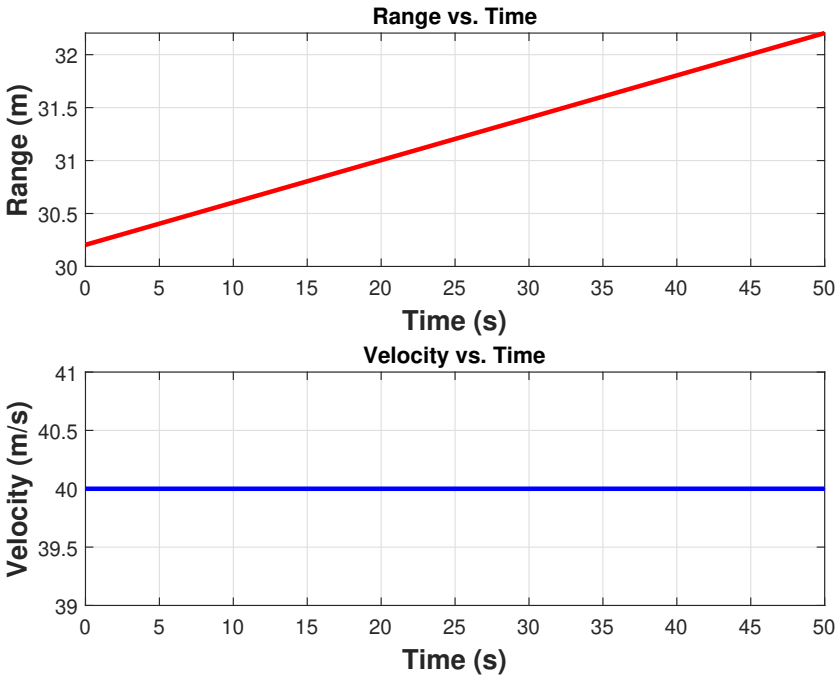
α and β values should depend on the measurement precision. If we use high precision equipment, we would prefer a high α and β that follow measurements. In this case, the filter would quickly respond to a velocity change of the target. OTOH, if measurement precision is low, we prefer low α and β. In this case, the filter smoothes the uncertainty (errors) in the measurements. However, the filter reaction to target velocity changes would be much slower.

[Code: a-b-c Filter/Ex2_alpha-beta_EstimationAlgorithm.m]

Example 3: Tracking Accelerating Aircraft in 1D

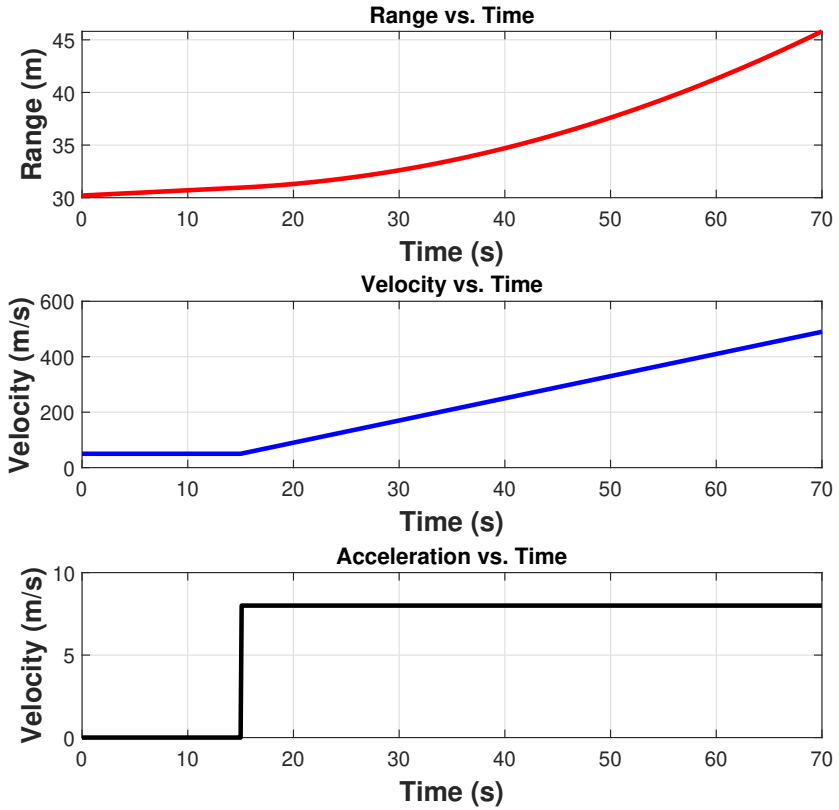
This example tracks an aircraft moving with constant acceleration with the $\alpha - \beta$ filter.

In Example 2, aircraft was moving at a constant velocity of 40m/s.



[Code: a-b-c Filter/Ex3_preliminary.m]

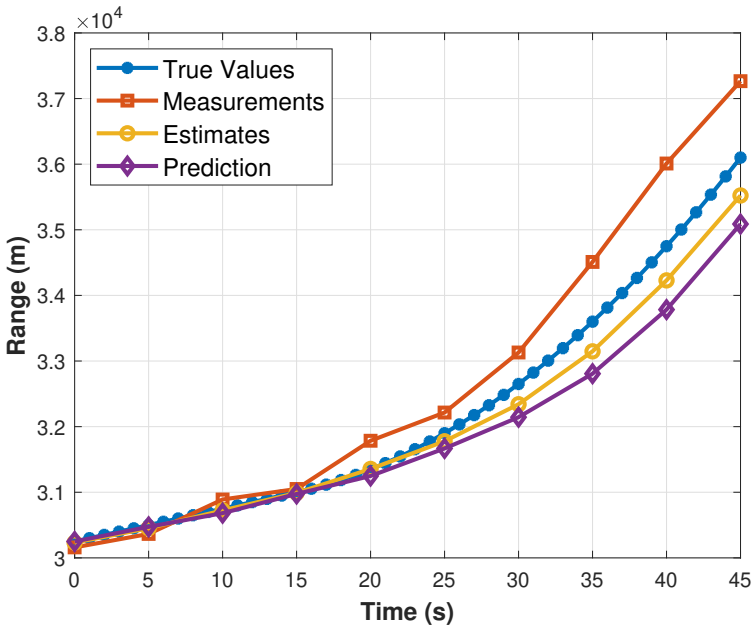
Assume, a fighter aircraft moving at a constant velocity of 50 m/s for 15 seconds. Then the aircraft accelerates with a constant acceleration of 8 m/s^2 for another 35 seconds.

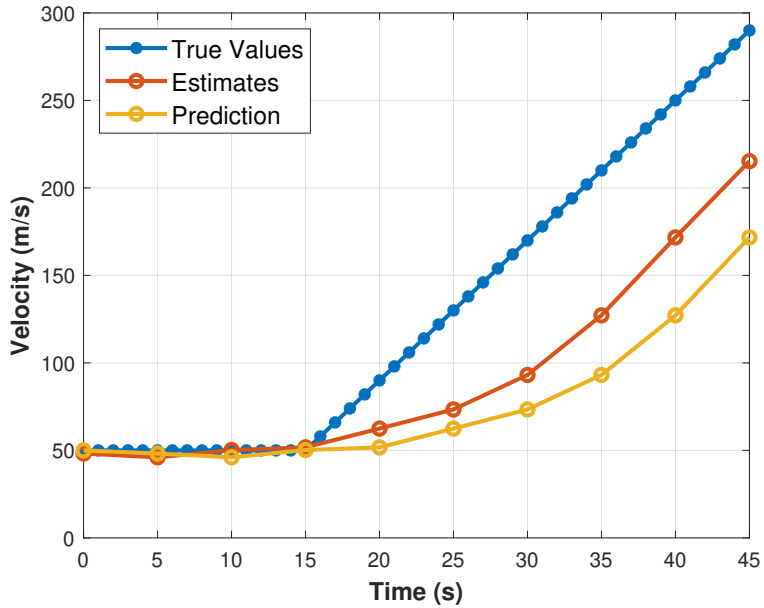


Let's track this aircraft with the filter used in Example 2.

Example 3: Tracking Accelerating Aircraft in 1D

Parameters: $\alpha = 0.2$, $\beta = 0.1$, $\Delta t = 5s$





There is a constant gap between true or measured values and estimates. The gap is called a lag error, or Dynamic error or Systematic error or Bias error or Truncation error.

The lag error appears during the acceleration period. After the acceleration period, the filter closes the gap and converges toward the true value. However, a significant lag error can result in the target loss, which is unacceptable in certain applications, such as missile guidance or air defense.

[Code: a-b-c Filter/Ex3_alpha-beta_EstimationAlgorithm.m]

Example 4: Tracking the Accelerating Aircraft with the $\alpha - \beta - \gamma$ Filter

State Extrapolation Equations:

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta_t + \hat{\ddot{x}}_{n,n}\frac{\Delta_t^2}{2}$$
$$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta_t$$
$$\hat{\ddot{x}}_{n+1,n} = \hat{\ddot{x}}_{n,n}$$

State Update Equations:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha(z_n - \hat{x}_{n,n-1})$$
$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta_t}\right)$$
$$\hat{\ddot{x}}_{n,n} = \hat{\ddot{x}}_{n,n-1} + \gamma\left(\frac{z_n - \hat{x}_{n,n-1}}{0.5\Delta_t^2}\right)$$

Scenario from Example 3: an aircraft that moves with a constant velocity of 50m/s for 15 seconds and then accelerates with a constant acceleration of $8m/s^2$ for another 35 seconds.

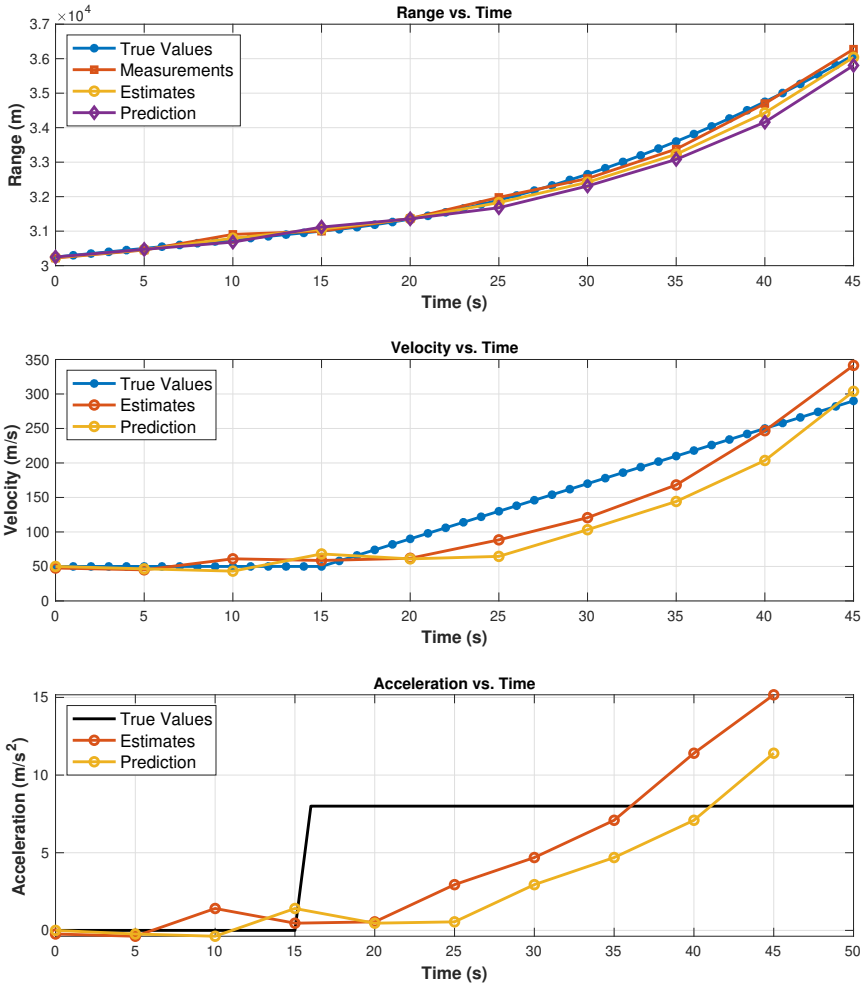
The $\alpha - \beta - \gamma$ filter parameters:

- $\alpha = 0.5$
- $\beta = 0.4$
- $\gamma = 0.1$
- $\Delta_t = 5s$

Initialization:

- $\hat{x}_{0,0} = 30,000\,\text{m}$
- $\hat{\dot{x}}_{0,0} = 50\,\text{m/s}$
- $\hat{\ddot{x}}_{0,0} = 0\,\text{m/s}^2$

Example 4: Results



- The $\alpha - \beta - \gamma$ filter with dynamic model equations that include acceleration can track the target with constant acceleration and eliminate the lag error.
- But what happens in the case of a maneuvering target? The target can suddenly change the flight direction by making a maneuver. The target's dynamic model can also include a jerk (changing acceleration).
- In such cases, the $\alpha - \beta - \gamma$ filter with constant $\alpha - \beta - \gamma$ coefficients produces estimation errors and, in some cases, loses the target track.
- The Kalman filter can handle uncertainty in the dynamic model.

[Code: a-b-c Filter/Ex4_alpha-beta-gamma_EstimationAlgorithm.m]

Summary of the $\alpha - \beta - \gamma$ Filter

- There are many types of $\alpha - \beta - \gamma$ filters, based on the same principle:
  - The current state estimation is based on the state update equations.
  - The following state estimation (prediction) is based on the dynamic model equations.
- The main difference between these filters is the selection of weighting coefficients. Some filter types use constant, others compute at every iteration
- Selection of parameters is discussed in detail in
  - Dirk Tenne, Tarunraj Singh. "Optimal Design of $\alpha - \beta - \gamma$ Filters". State University of New York at Buffalo.
- Another important issue is the initiation of the filter, i.e., providing the initial value for the first filter iteration.

Most Popular $\alpha - \beta - \gamma$ Filters

- Wiener Filter
- Bayes Filter
- Fading-memory polynomial Filter
- Expanding-memory (or growing-memory) polynomial Filter
- Least-squares Filter
- Benedict–Bordner Filter
- Lumped Filter
- Discounted least-squares $\alpha - \beta$ Filter
- Critically damped $\alpha - \beta$ Filter
- Growing-memory Filter
- Kalman Filter
- Extended Kalman Filter
- Unscented Kalman Filter
- Extended Complex Kalman Filter
- Gauss-Hermite Kalman Filter
- Cubature Kalman Filter
- Particle Filter

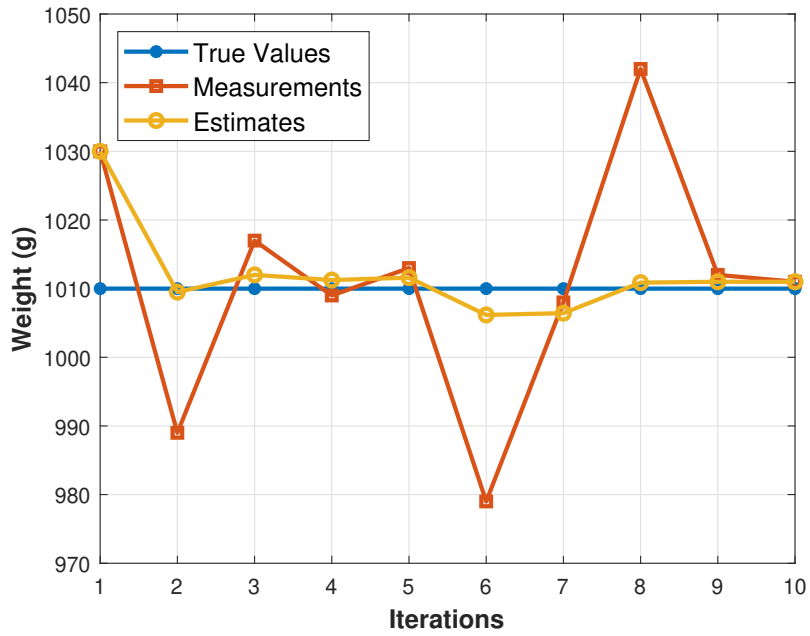
One Dimensional Kalman Filter without Process Noise

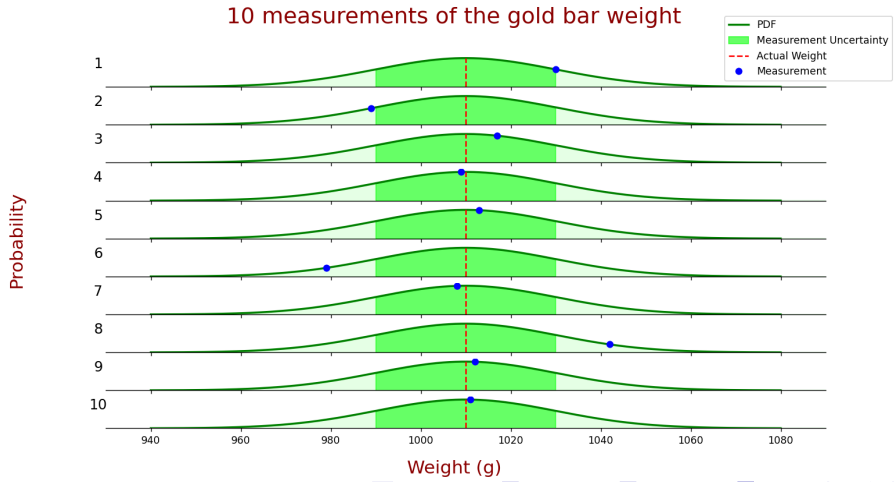
Example 1: Gold Bar Weight Measurements.
Measurement errors:

- The different between true values and measurements; often described by variance (σ^2) for their randomness.
- The variance of the measurement errors could be provided by the scale vendor or derived by a calibration procedure.
- The variance of the measurement errors is the measurement uncertainty, denoted by r.

Estimate error

- The difference between the estimates and the true values is the estimate error.
- the estimate error becomes smaller and smaller as we make additional measurements, and it converges towards zero, while the estimated value converges towards the true value.
- We don't know the estimate error, but we can estimate the uncertainty in estimate, denoted by p (estimate variance).





State Prediction: Estimate Uncertainty Extrapolation or Covariance Extrapolation Equation (Eq. 3)

Like state extrapolation, the estimate uncertainty extrapolation is done with the dynamic model equations.

- In Example 1 (gold bar weight measurement), the system's dynamics is constant. Thus, the estimate uncertainty extrapolation is:

$$p_{n+1,n} = p_{n,n}$$

where p is the gold bar's weight estimate uncertainty

- In Example 2, the predicted target position is:

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t \hat{\dot{x}}_{n,n}, \quad \text{[Aircraft range]}$$
$$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n}, \quad \text{[Constant velocity]}$$

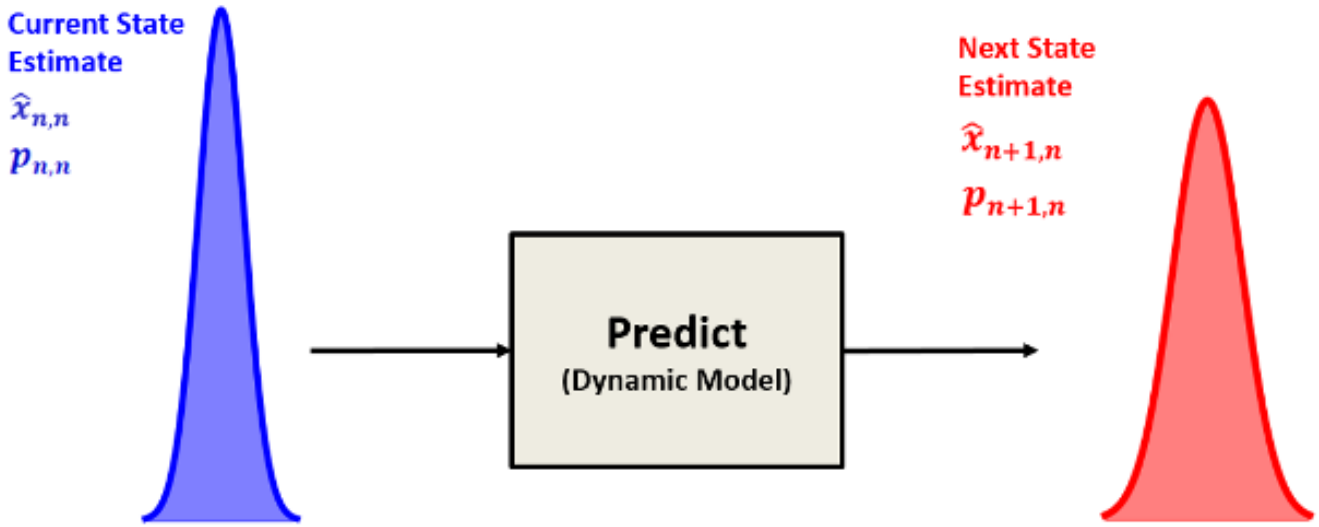
i.e., the predicted position equals the current estimated position + the current estimated velocity x time. The predicted velocity equals the current velocity estimate (assuming a constant velocity model).

- The estimate uncertainty extrapolation would be:

$$p^x_{n+1,n} = p^x_{n,n} + \Delta t^2 p^v_{n,n},$$
$$p^v_{n+1,n} = p^v_{n,n}$$

where
p^x is the position estimate uncertainty p^v is the velocity estimate uncertainty
i.e., the predicted position estimate uncertainty equals the current position estimate uncertainty + current velocity estimate uncertainty $\times$ by time squared. The predicted velocity estimate uncertainty equals the current velocity estimate uncertainty (assuming a constant velocity model).

State Prediction Illustration

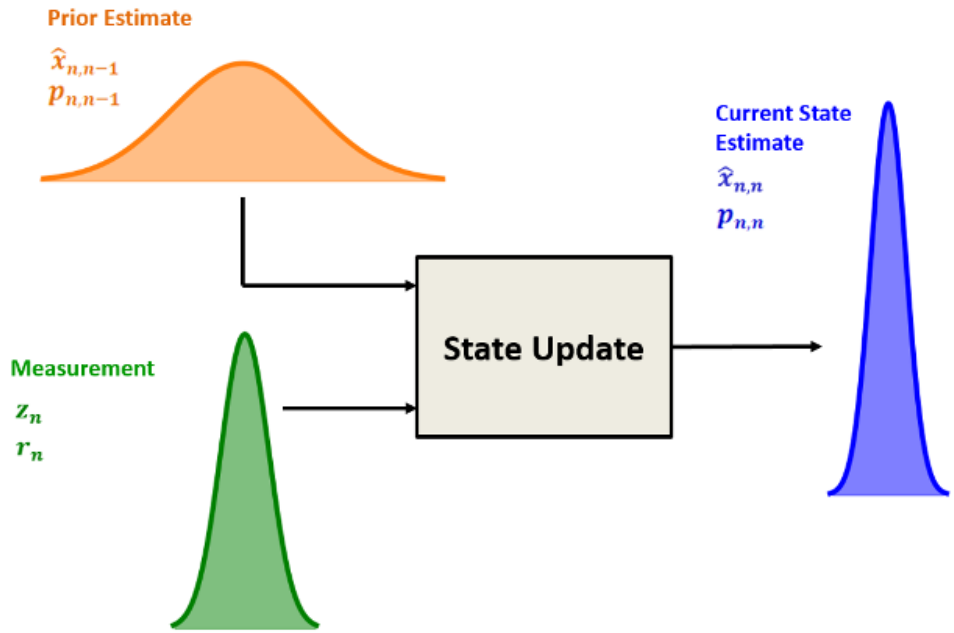


Note that for any normally distributed random variable x with variance σ^2, kx is distributed normally with variance $k^2\sigma^2$, therefore the time term in the uncertainty extrapolation equation is squared.

State Update: Kalman Gain Equation (Eq.4)

To estimate the current state of the system, we combine two RVs:

- The prior state estimate (the current state estimate predicted at the previous state).
- The measurement



Kalman Gain cont...

The Kalman Filter is the optimal filter; it combines the prior state estimate with the measurements such that it minimizes the current state estimate uncertainty. Derivation for 1D Kalman Filter:

- Given the measurement z_n and the prior estimate $\hat{x}_{n,n-1}$, we want to find an optimum combined estimate $\hat{x}_{n,n}$ based on the measurement and the prior estimate.
- Which is actually a weighted mean of the prior estimate and the measurement:

$$\hat{x}_{n,n} = k_1 z_n + k_2 \hat{x}_{n,n-1}$$

Where k_1 and k_2 are the weights of the measurement and the prior estimate, i.e.,

$$k_1 + k_2 = 1$$

$$\hat{x}_{n,n} = k_1 z_n + (1 - k_1)\hat{x}_{n,n-1}$$

- Since, for any normally distributed random variable x with variance σ^2, kx is distributed normally with variance $k\sigma^2$, thus:

$$p_{n,n} = k_1^2 r_n + (1 - k_1)^2 p_{n,n-1}$$

where:
$p_{n,n}$ is the variance of the optimum combined estimate $\hat{x}_{n,n}$
$p_{n,n-1}$ is the variance of the prior estimate $\hat{x}_{n,n-1}$
r_n is the variance of the measurement z_n

- Since we are looking for an **optimum** estimate, we are interested to minimize $p_{n,n}$.
- To find k_1 that minimizes $p_{n,n}$, we differentiate $p_{n,n}$ with respect to k_1 and set the result to zero, i.e.,

$$\frac{dp_{n,n}}{dk_1} = 2k_1 r_n - 2(1 - k_1)p_{n,n-1}$$

$$\boxed{\implies k_1 = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}}$$

- Since the Kalman Gain yields the minimum variance estimate, the Kalman Filter is also called an **optimal filter**.

State Update: Kalman Gain Equation (Eq.4)

The Kalman Gain (denoted by Kn) is the weight given to the current state estimate and the measurements:

$$K_n = \frac{\text{Uncertainty in Estimate}}{\text{Uncertainty in Estimate} + \text{Uncertainty in Measurement}} = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$$

where $p_{n,n-1}$ is the extrapolated estimate uncertainty.

- The Kalman Gain is a number between zero and one: $0 \leq K_n \leq 1$
- Rewrite state update equation:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n \left(z_n - \hat{x}_{n,n-1}\right) = \underbrace{(1 - K_n)}_{\text{Current State Estimate Weight}} \hat{x}_{n,n-1} + \underbrace{K_n}_{\text{Measurement Weight}} z_n$$

- When the measurement uncertainty is large and the estimate uncertainty is low, the Kalman Gain is close to zero. Hence we give significant weight to the estimate and a small weight to the measurement.
- OTOH, when the measurement uncertainty is low and the estimate uncertainty is large, the Kalman Gain is close to one. Hence we give a low weight to the estimate and a significant weight to the measurement.
- If the measurement uncertainty equals the estimate uncertainty, then the Kalman gain equals 0.5.

"The Kalman gain tells how much the measurement changes the estimate."

Estimate Uncertainty Update or Covariance Update Equation (Eq. 5)

Estimate uncertainty update, also called as Covariance Update Equation is defined as

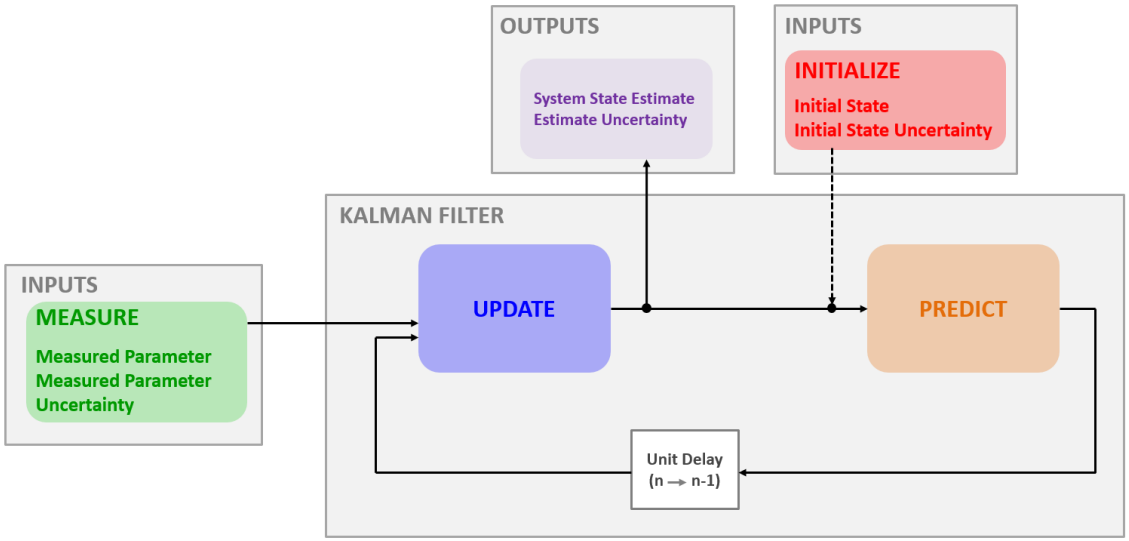
$$p_{n,n} = (1 - K_n)p_{n,n-1},$$

where
K_n is the Kalman Gain
$p_{n,n-1}$ is the estimate uncertainty, calculated during previous filter estimation
$p_{n,n}$ is estimate uncertainty of current state

- Since $(1 - K_n) \leq 1$, the estimate uncertainty is constantly getting smaller with each filter iteration.
- When the measurement uncertainty is large, the Kalman gain is low. Therefore, the convergence of the estimate uncertainty would be slow.
- However, the Kalman gain is high when the measurement uncertainty is small. Therefore, the estimate uncertainty would quickly converge towards zero.

Let's Put it All Together - Basic

Like the $\alpha - \beta(-\gamma)$ filter, the Kalman filter utilizes the "Measure, Update, Predict" algorithm.

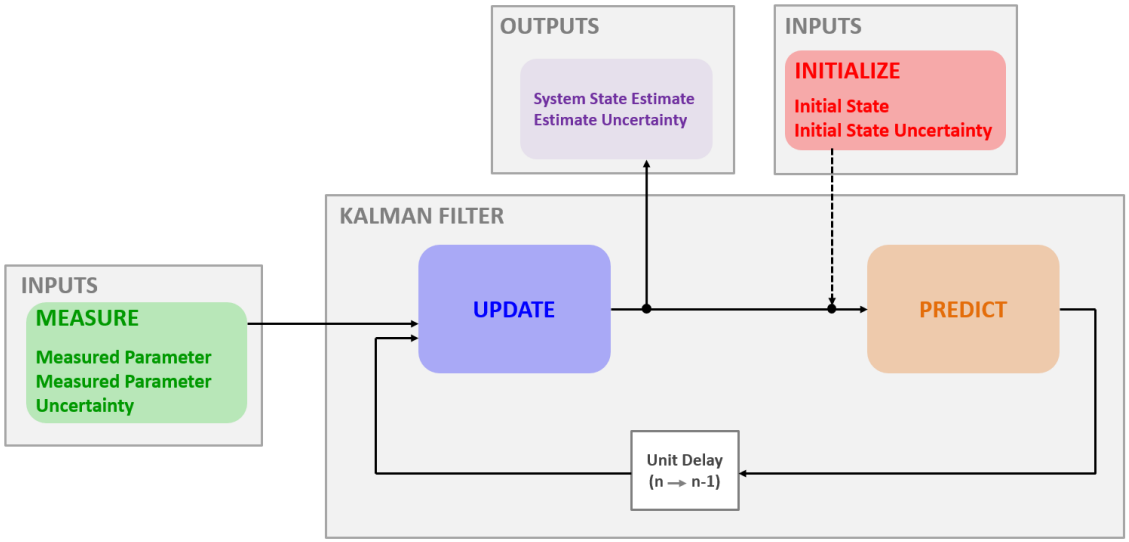


Inputs:

- Initialization: (performed only once). Init. parameters can be provided by another system or process or an educated guess based on experience/theoretical knowledge. If not precise, Kalman filter tries to converge toward true value.
  - Initial System State ($\hat{x}_{0,0}$)
  - Initial Uncertainty ($p_{0,0}$)
- Measurement:(performed at each filter cycle)
  - Measured System State (z_n)
  - Measurement Uncertainty (r_n). Usually, r_n is provided by the equipment vendor, or can be derived from equipment calibration. For example, the radar measurement uncertainty depends on several parameters such as SNR, beam width, bandwidth, time on target, clock stability, and more. While, every radar measurement has a different SNR, beam width, and time on target.

Let's Put it All Together - Basic (Cont.)

Contrary to the $\alpha - \beta(-\gamma)$ filter, the Kalman Filter treats measurements, current state estimation, and next state estimation (predictions) as normally distributed random variables.



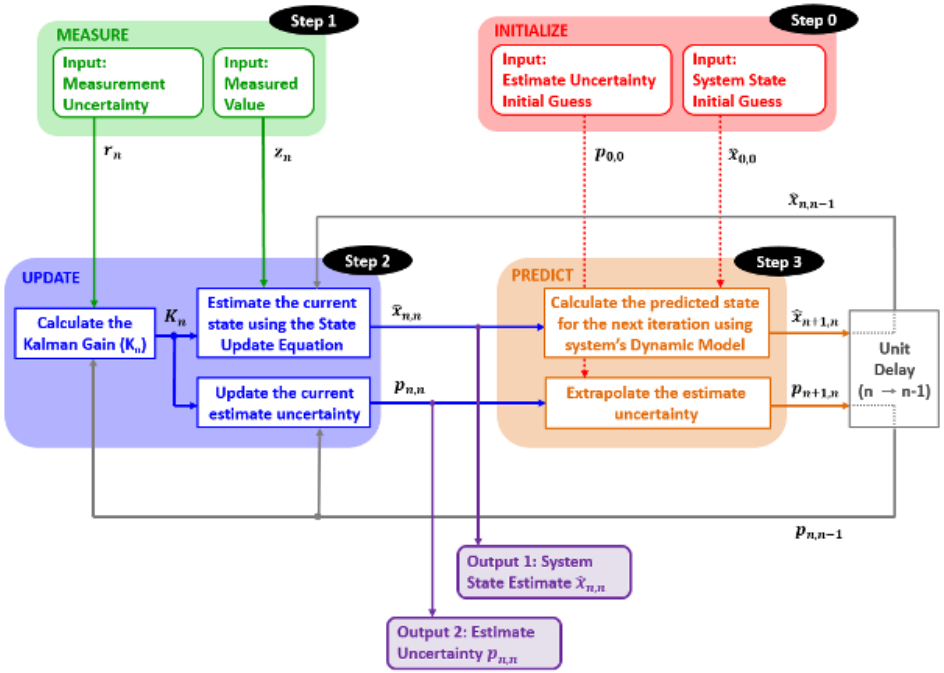
Outputs:

- System State Estimate ($\hat{x}_{n,n}$)
- Estimate Uncertainty ($p_{n,n}$).
  Since $p_{n,n} = (1 - K_n)p_{n,n-1}$, $p_{n,n}$ is getting smaller with each filter iteration, as $(1 - K_n) \leq 1$. So it's up to us to decide how many measurements to take.

Let's Put it All Together - Basic (Cont.)

	Equation	Equation Name	Alternative names used in the literature
State Update	$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - \hat{x}_{n,n})$	State Update	Filtering Equation
	$p_{n,n} = (1 - K_n)p_{n,n-1}$	Covariance Update	Corrector Equation
	$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$	Kalman Gain	Weight Equation
State Predict	$\hat{x}_{n+1,n} = \hat{x}_{n,n}$ (for constant dynamics)	State Extrapolation	Predictor Equation Transition Equation Prediction Equation Dynamic Model State Space Model
	$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t\hat{\dot{x}}_{n,n}$ $\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n}$ (for constant velocity dynamics)		
	$p^x_{n+1,n} = p^x_{n,n}$ (for constant dynamics)	Covariance Extrapolation	Predictor Covariance Equation
	$p^x_{n+1,n} = p^x_{n,n} + \Delta t^2 p^v_{n,n}$ $p^v_{n+1,n} = p^v_{n,n}$ (for constant velocity dynamics)		

Let's Put it All Together - Detailed



- **Step 0: Initialization**
  - Initial System State ($\hat{x}_{0,0}$)
  - Initial Uncertainty ($p_{0,0}$)
- **Step 1: Measurement**
  - Measured System State (z_n)
  - Measurement Uncertainty (r_n).
- **Step 2 - Inputs:** State Update (estimation of the system's current state).
  - Measured value (z_n)
  - Measurement Uncertainty (r_n).
  - Previous System State Estimate ($\hat{x}_{n,n-1}$)
  - Estimate Uncertainty ($p_{n,n-1}$)

Based on the inputs, the state update process calculates the Kalman Gain and provides two outputs:

- Current System State Estimate ($\hat{x}_{n,n}$)
- Current State Estimate Uncertainty ($p_{n,n}$)
- **Step 3: Prediction:**
  - The prediction process extrapolates the current system state and uncertainty of the current system state estimate to the next system state, based on the system's dynamic model.
  - At the first filter iteration the initialization outputs are treated as the Previous State Estimate and Uncertainty.
  - The prediction outputs become the Previous State Estimate and Uncertainty on the following filter iterations.

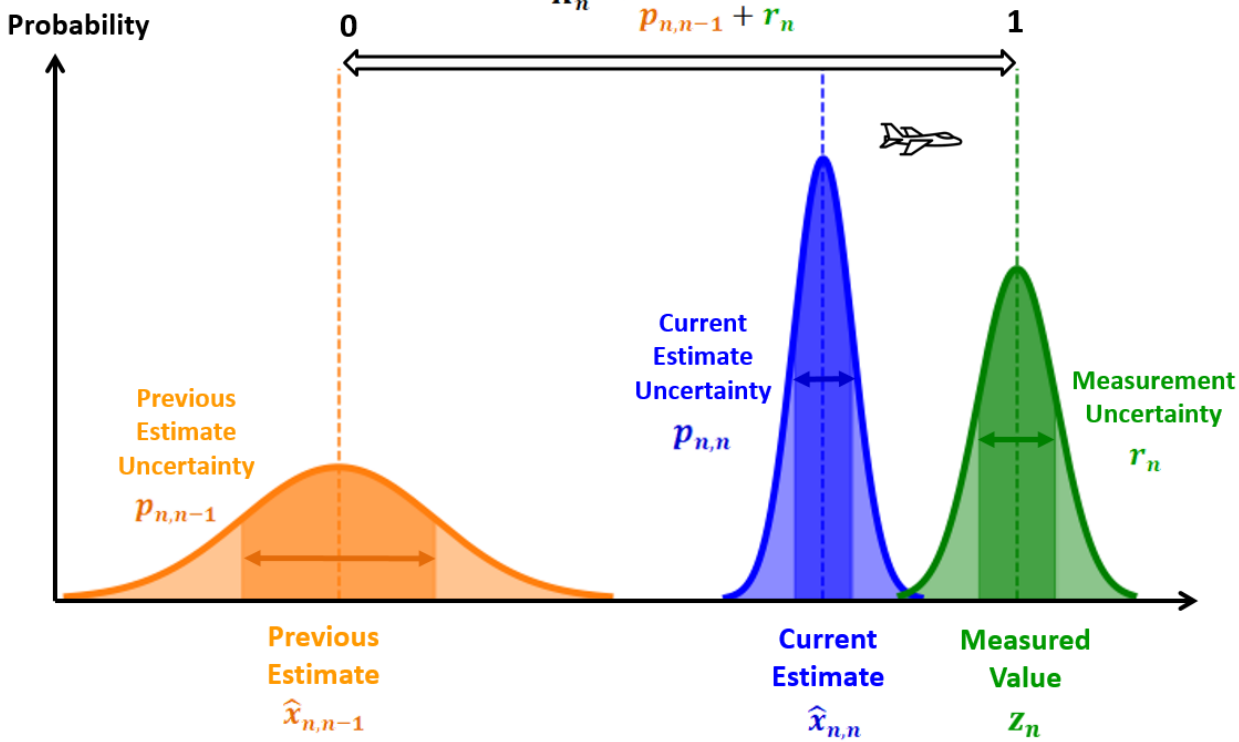
Kalman Gain Intuition - High Kalman Gain

The Kalman Gain defines the weight of the measurement (z_n) and the weight of the previous estimate ($\hat{x}_{n,n-1}$) when forming a new estimate ($\hat{x}_{n,n}$).

A low measurement uncertainty relative to the estimate uncertainty would result in a high Kalman Gain (close to 1). Therefore the new estimate would be close to the measurement.

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n\left(z_n - \hat{x}_{n,n-1}\right)$$
$$p_{n,n} = (1 - K_n)p_{n,n-1}$$

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$$

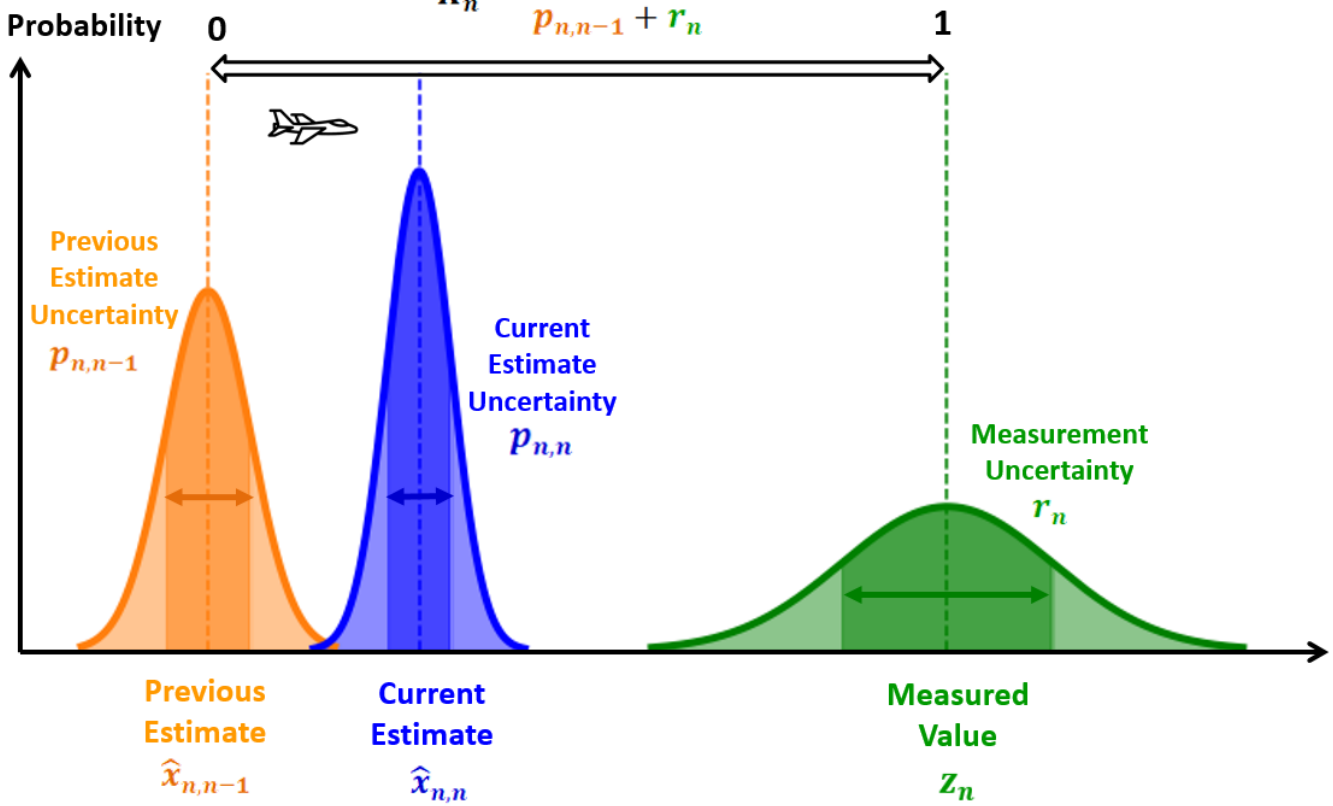


Kalman Gain Intuition - Low Kalman Gain

A high measurement uncertainty relative to the estimate uncertainty would result in a low Kalman Gain (close to 0). Therefore the new estimate would be close to the previous estimate.

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n\left(z_n - \hat{x}_{n,n-1}\right)$$
$$p_{n,n} = (1 - K_n)p_{n,n-1}$$

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$$



Example 5: Estimating the Height of the Building

We would like to estimate the height of a building using a very imprecise altimeter. The building height doesn't change over time, at least during the short measurement process.

- The true building height is 50 meters.
- The altimeter measurement error (standard deviation) is 5 meters.
- The ten measurements are: 48.54m, 47.11m, 55.01m, 55.15m, 49.89m, 40.85m, 46.72m, 50.05m, 51.27m, 49.95m.

ITERATION ZERO

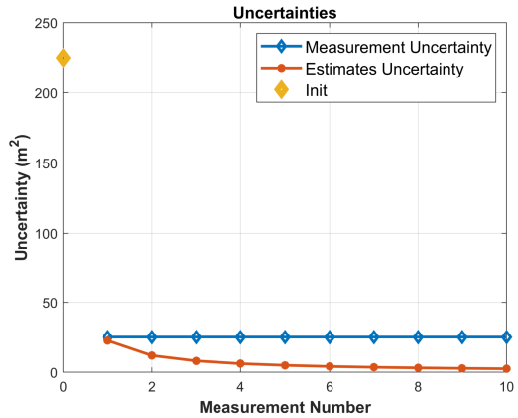
- **Initialization:**
  - Estimated build height
$$\hat{x}_{0,0} = 60m$$
  - Estimate uncertainty with std. $\sigma = 15m$,
$$p_{0,0} = 225$$
- **Prediction:**
  - Since Dynamic Model is constant
$$\hat{x}_{1,0} = \hat{x}_{0,0} = 60$$
  - The extrapolated estimate uncertainty (variance) also doesn't change:
$$\hat{p}_{1,0} = \hat{p}_{0,0} = 225$$

FIRST ITERATION

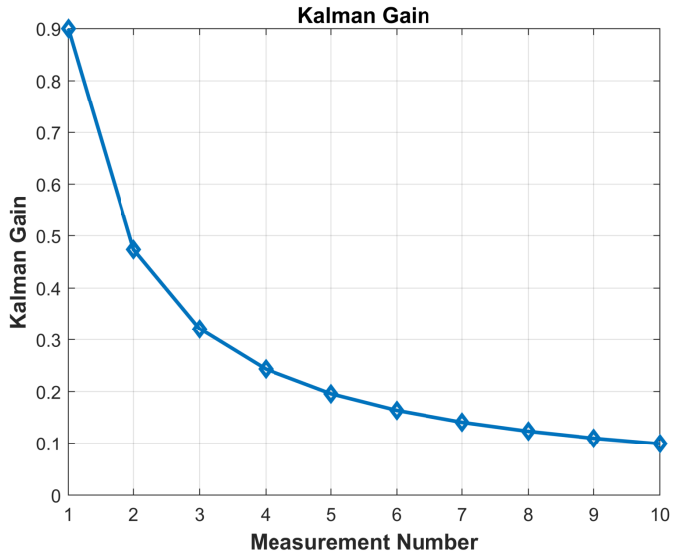
- **Step 1: Measure:**
  - The first measurement is
$$z_1 = 48.54m$$
  - The measurement uncertainty (since $\sigma = 5$)
$$r_1 = 25$$
- **Step 2: Update:**
  - Kalman gain
$$K_1 = \frac{p_{1,0}}{p_{1,0} + r_n} = 0.9$$
  - Estimating the current state:
$$\hat{x}_{1,1} = \hat{x}_{1,0} + K_1(z_1 - \hat{x}_{1,0}) = 49.69m$$
  - Current estimate uncertainty
$$p_{1,1} = (1 - K_1)p_{1,0} = 22.5$$
- **Step 3: Predict:**
  - For constant dynamic model,
$$\hat{x}_{2,1} = \hat{x}_{1,1} = 49.69m$$
$$p_{2,1} = p_{1,1} = 22.5$$

Example 5: Estimating the Height of the Building — Results

The estimated value converges to about 49.5 meters after 7 measurements.



At 1st filter iteration, the estimate uncertainty is close to the measurement uncertainty and quickly decreases. After 10 measurements, the estimate uncertainty (σ^2) is 2.47, i.e., the estimate error std is: $\sigma = \sqrt{(2.47)} = 1.57m$.



Summary:

- The Kalman Gain is dynamic and depends on the precision of the measurement device.
- The initial value used by the Kalman Filter is not precise. Therefore, the measurement weight in the State Update Equation is high, and the estimate uncertainty is high.
- With each iteration, the measurement weight is lower; therefore, the estimate uncertainty is lower.

Example 5: Estimating the Height of the Building — Results

One can define accuracy criteria based on the specific application requirements. The typical accuracy criteria are: Maximum error, Mean error, Root Mean Square Error (RMSE). Assume that for a building height measurement application, there is a requirement for 95% confidence.

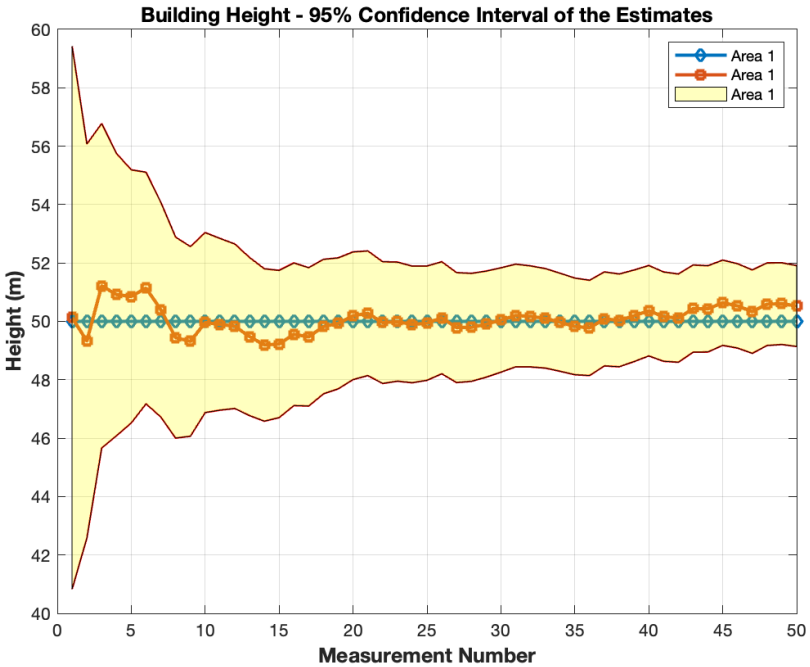


Figure: High Uncertainty: $r = 5\,\text{m}$

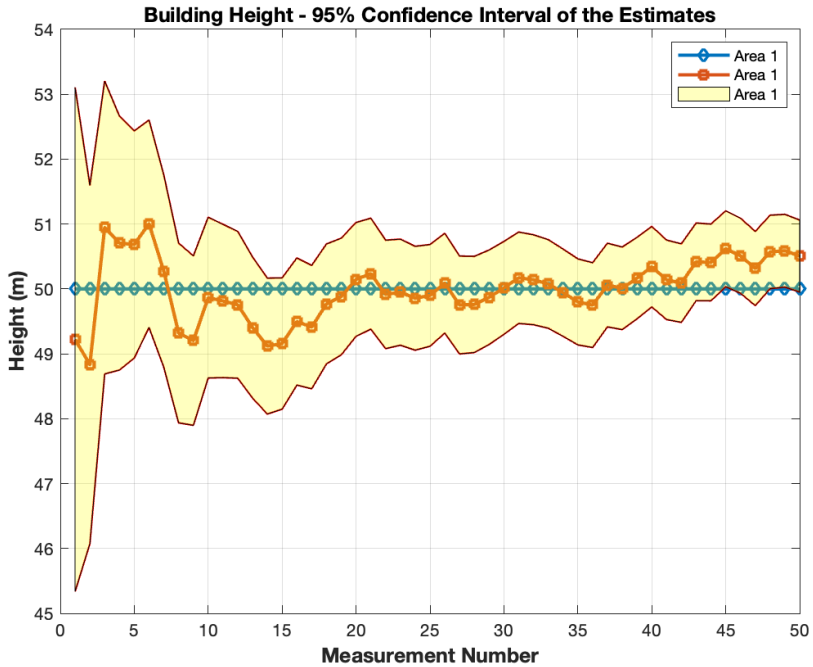


Figure: High Uncertainty: $r = 2\,\text{m}$

With measurement uncertainty $r = 2\,\text{m}$, 2 out of 50 samples slightly exceed the 95% confidence region; satisfying our requirements of desired estimate uncertainty.

`[Code: 1D KF/Ex5_FirstKalmanFilter.m]`

Complete Model of 1D Kalman Filter including Process Noise

To complete the one-dimensional Kalman Filter model, we need to add the process noise variable to the Covariance Extrapolation Equation.

The Process Noise:

- In the real world, there are uncertainties in the system dynamic model. For example,
  - When estimating the resistance value of the resistor, we assume a constant dynamic model, i.e., the resistance doesn't change between the measurements. However, the resistance can change slightly due to the fluctuation of the environment temperature.
  - When tracking ballistic missiles with radar, the uncertainty of the dynamic model includes random changes in the target acceleration.
  - The uncertainties are much more significant for an aircraft due to possible aircraft maneuvers.
  - On the other hand, when estimating the location of a static object using a GPS receiver, the uncertainty of the dynamic model is zero since the static object doesn't move.
- The uncertainty of the dynamic model is called the Process Noise. In the literature, it is also called *plant noise, driving noise, dynamics noise, model noise, and system noise*. The process noise produces estimation errors.
- The Process Noise Variance is denoted by the letter q.
- The Covariance Extrapolation Equation shall include the Process Noise Variance.
- For example, the Covariance Extrapolation Equation for constant dynamics is:

$$p_{n+1,n} = p_{n,n} + q_n$$

Note: The State Extrapolation Equation and the Covariance Extrapolation Equation depend on the system dynamics.

Example 6: Estimating the Temperature of the Liquid in a Tank

We assume that at a steady state, the liquid temperature is constant. However, some fluctuations in the true liquid temperature are possible.

We can describe the system dynamics by the following equation:

$$x_n = T + w_n$$

where
T is the constant temperature
w_n is a random process noise with variance q
Assumptions:

- Tanks true temperature of 50 degrees Celsius
- We assume that the model is accurate. Thus, we set the process noise variance, $q = 0.0001$.
- The measurement error (standard deviation) is 0.1 degrees Celsius.
- The measurements are taken every 5 seconds.
- The true liquid temperature values at the measurement points are (in degrees Celsius): 49.979, 50.025, 50, 50.003, 49.994, 50.002, 49.999, 50.006, 49.998, and 49.991.
- The measurements are (in degrees Celsius): 49.95, 49.967, 50.1, 50.106, 49.992, 49.819, 49.933, 50.007, 50.023, and 49.99.

Example 6: Estimating the Temperature of the Liquid in a Tank

ITERATION ZERO

- **Initialization:**
  - We don't know the true temperature of the liquid in a tank; our estimate (or guess) is

$$\hat{x}_{0,0} = 10^o C$$

  - Estimate uncertainty with std. $\sigma = 100$,

$$p_{0,0} = \sigma^2 = 10,000$$

- **Prediction:**
  - Since our model has constant dynamics, the predicted estimate is equal to the current estimate:

$$\hat{x}_{1,0} = \hat{x}_{0,0} = 10^o C$$

  - The extrapolated estimate uncertainty (variance):

$$\hat{p}_{1,0} = \hat{p}_{0,0} + q = 10000 + 0.0001$$

FIRST ITERATION

- **Step 1: Measure:**
  - The first measurement is

$$z_1 = 49.95^o C$$

  - The measurement uncertainty (since $\sigma = 0.1$)

$$r_1 = 0.01$$

- **Step 2: Update:**
  - Kalman gain

$$K_1 = \frac{p_{1,0}}{p_{1,0} + r_n} = 0.999999 \ [\approx 1]$$

  - Estimating the current state:

$$\hat{x}_{1,1} = \hat{x}_{1,0} + K_1(z_1 - \hat{x}_{1,0}) = 49.95^o C$$

  - Current estimate uncertainty

$$p_{1,1} = (1 - K_1)p_{1,0} = 0.01$$

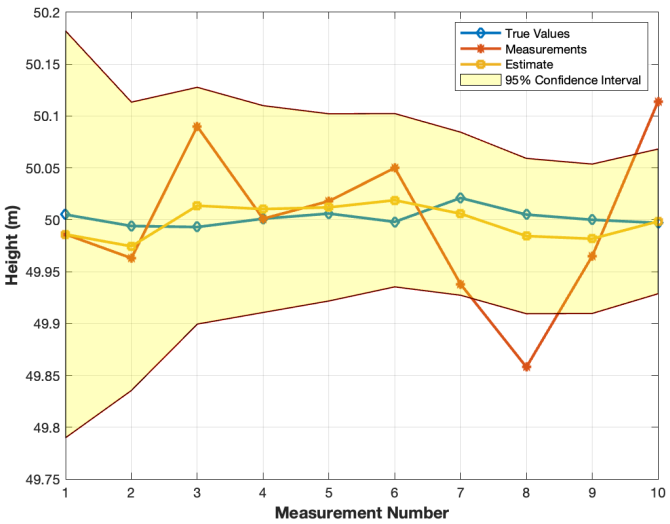
- **Step 3: Predict:**
  - For constant dynamic model,

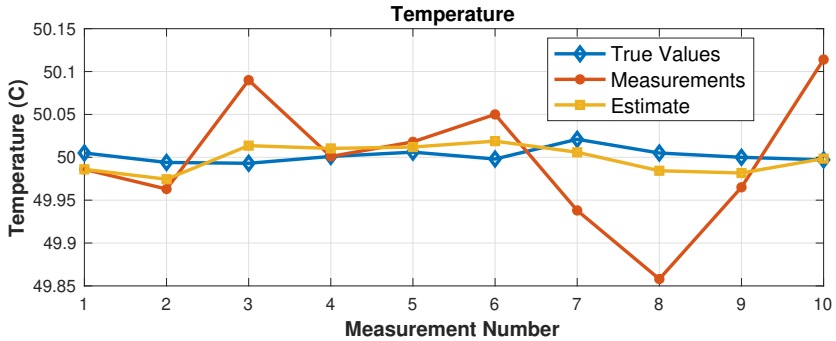
$$\hat{x}_{2,1} = \hat{x}_{1,1} = 49.95^o C$$

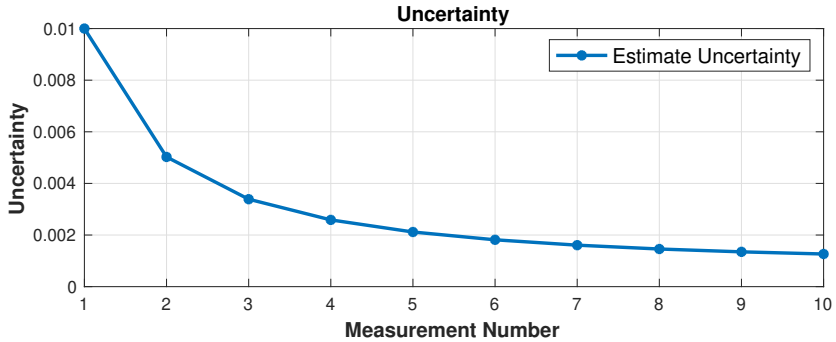
$$p_{2,1} = p_{1,1} + q = 0.0101$$

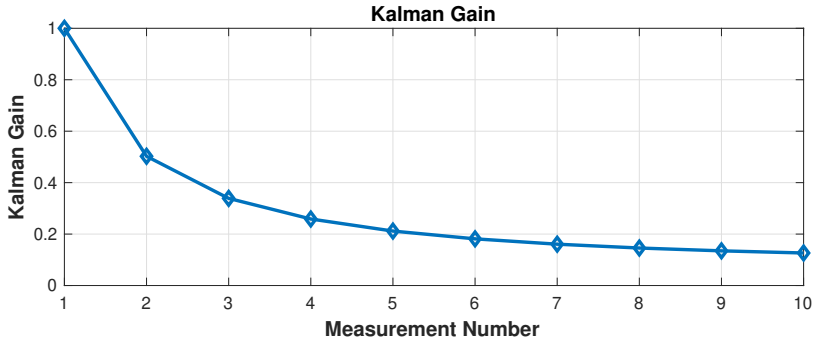
Example 6: Estimating the Temperature of the Liquid in a Tank — Results

- The estimated value converges toward the true value.
- The estimate uncertainty quickly goes down. After 10 measurements, the estimate uncertainty is $\sigma^2 = 0.0013$, i.e., the estimate error std is: $\sigma = 0.036^o C$.
- So we can say that the liquid temperature estimate is: $49.988 \pm 0.036^o C$.
- **Summary:** Although the system dynamics include a random process noise, the Kalman Filter can provide a good estimation.





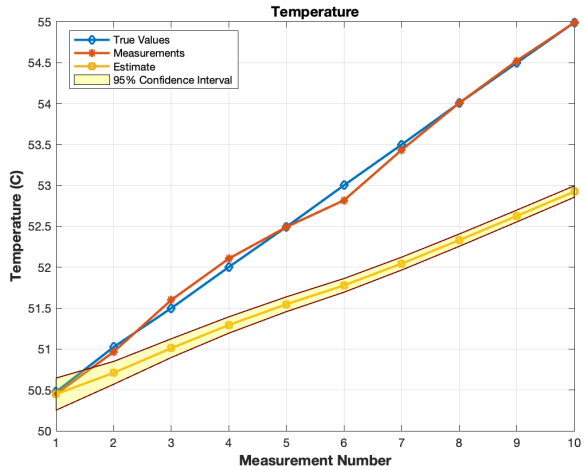




[Code: 1D KF/Ex6_KalmanFilter_ProcessNoise.m]

Example 7: Estimating the Temperature of a Heating Liquid in a Tank I

- Let's estimate the temperature of a liquid in a tank as in Example 6. However, in this case, the dynamic model of the system is not constant— the liquid is heating at a rate of $0.1^oC/sec$.
- Parameters are same as in Example 6.
- The dynamic model of the system is constant. Although the true dynamic model of the system is not constant (since the liquid is heating), we treat the system as a system with a constant dynamic model (the temperature doesn't change)
- The true liquid temperature values at the measurement points are: 50.479, 51.025, 51.5, 52.003, 52.494, 53.002, 53.499, 54.006, 54.498, and 54.991.
- The measurements are: 50.45, 50.967, 51.6, 52.106, 52.492, 52.819, 53.433, 54.007, 54.523, and 54.99.
- Kalman Filter has failed to provide a reliable estimation as there is a lag error in the estimation.



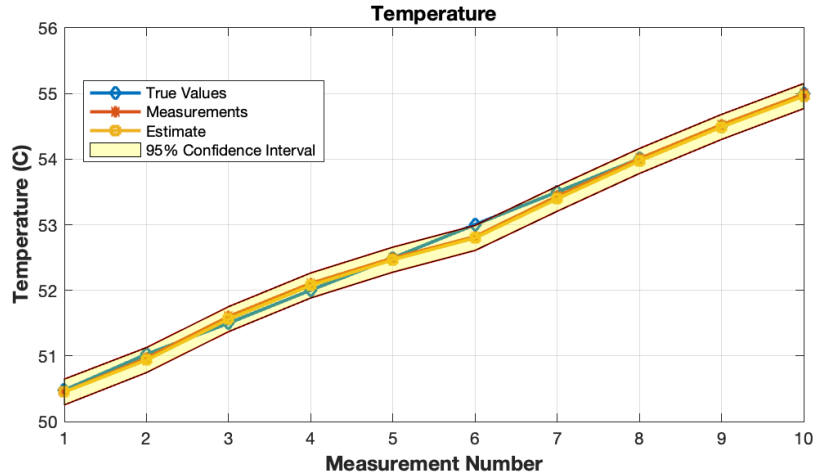
- **Two Reason for the constant lag error:**
  - The dynamic model doesn't fit the case.
  - Process noise is very low ($q = 0.0001$) while the true temperature fluctuations are much more significant.
- Two possible ways to fix the lag error:
  - If we know that the liquid temperature can change linearly, we can define a new model that considers that. *What of the temperature change cannot be modeled?*
  - If the model is not well defined, we can adjust the process model reliability by increasing the process noise (q).

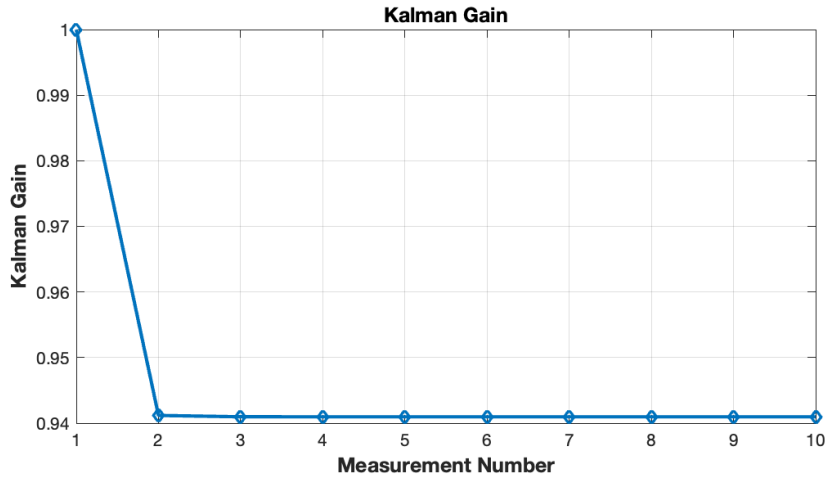
[Code: 1D KF/Ex7_KalmanFilter_ProcessNoise_I.m]

Example 8: Estimating the Temperature of a Heating Liquid in a Tank II

This example is similar to the previous example, with only one change. Since our process is not well-defined, we increase the process uncertainty (q) from 0.0001 to 0.15.

- Due to the high process uncertainty, the measurement weight is much higher than the weight of the estimate. Thus, the Kalman Gain is high, and it converges to 0.94.
- We can eliminate the lag error by setting a high process uncertainty. However, since our model is not well-defined, we get noisy estimates that are almost equal to the measurements, and we miss the goal of the Kalman Filter.
- The best Kalman Filter implementation would involve a model that is very close to reality, leaving little room for process noise. However, a precise model is not always available





[Code: 1D KF/Ex8_KalmanFilter_ProcessNoise_II.m]

Part 2: Multidimensional (Multivariate) Kalman Filter

Objectives

- Kalman Filter with matrix notation for designing a multivariate Kalman Filter
- Linear Kalman Filter (LKF): assumes that the system dynamics are linear
- Mathematical derivation of Kalman Filter equations, dynamic systems modeling, and two numerical examples

Introduction

- Until now, we have been looking at 1D processes, like estimating the liquid temperature.
- But many dynamic processes have 2, 3, or even more dimensions.
- **Ex1:** The state vector that describes the airplane's position in space is 3D:

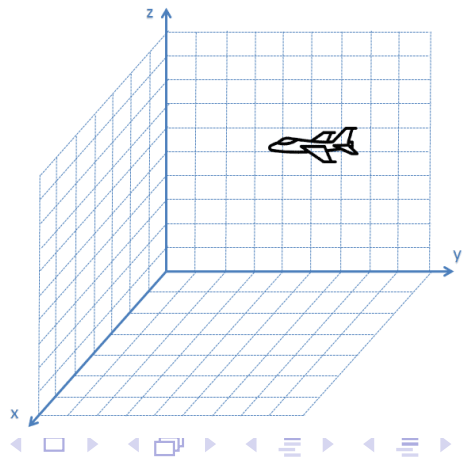
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- **Ex2:** The state vector that describes the airplane position and velocity is 6-D:

$$\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

- **Ex3:** The state vector that describes the airplane position, velocity, and acceleration is nine-dimensional:

$$\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$



Introduction

- Assuming a constant acceleration dynamic model, we can describe the extrapolated airplane state at time n by nine motion equations:

$$
\begin{aligned}
x_n &= x_{n-1} + \dot{x}_{n-1}\Delta t + \frac{1}{2}\ddot{x}_{n-1}\Delta t^2 \\
y_n &= y_{n-1} + \dot{y}_{n-1}\Delta t + \frac{1}{2}\ddot{y}_{n-1}\Delta t^2 \\
z_n &= z_{n-1} + \dot{z}_{n-1}\Delta t + \frac{1}{2}\ddot{z}_{n-1}\Delta t^2 \\
\dot{x}_n &= \dot{x}_{n-1} + \ddot{x}_{n-1}\Delta t \\
\dot{y}_n &= \dot{y}_{n-1} + \ddot{y}_{n-1}\Delta t \\
\dot{z}_n &= \dot{z}_{n-1} + \ddot{z}_{n-1}\Delta t \\
\ddot{x}_n &= \ddot{x}_{n-1} \\
\ddot{y}_n &= \ddot{y}_{n-1} \\
\ddot{z}_n &= \ddot{z}_{n-1}
\end{aligned}
$$

- Common practice to describe a multidimensional process with a single equation in matrix form.
- Second, computers are highly efficient at matrix calculations. Implementing the Kalman Filter in matrix form yields faster computation run time.

The Background Break - Expectation Algebra

The necessary mathematical background for Kalman Filter:

- Matrix Operations: Vector/matrix addition and multiplication, transpose, Matrix inverse, Symmetric matrices
- Expectation Algebra
- Multivariate Normal Distribution (Covariance and Covariance Matrices)

Expectation Algebra

- The expectation of a random variable X equal the mean of X: $E(X) = \mu_X$.

Rule	**Notes**
$E(X) = \mu_X = \sum xp(x)$	$p(x)$ is PDF of x
$E(a) = a$	a is constant
$E(aX) = aE(x)$	$--$
$E(a \pm X) = a \pm E(x)$	$--$
$E(a \pm bX) = a \pm bE(x)$	b is constant
$E(X \pm Y) = E(x) \pm E(Y)$	Y is another R.V.
$E(XY) = E(X)E(Y)$	X, Y are independent

Table: Expectation Rules

The Background Break - Expectation Algebra

The following table includes the variance and covariance expectation rules.

- Variance of a R.V. X: $V(X) = E\left((X - \mu_X)^2\right)$
- Covariance of R.V. X and Y: $COV(X, Y) = E\left((X - \mu_X)(Y - \mu_Y)\right)$

Rule	**Notes**
$V(a) = 0$	Variance of a constant a
$V(a \pm X) = V(X)$	Variance of X
$V(X) = E(X^2) - \mu_X^2$	$--$
$V(a \pm X) = a^2 V(X)$	a is constant
$COV(X, Y) = E(XY) + \mu_X \mu_Y$	Covariance of X and Y
$COV(X, Y) = 0$	If X and Y are Independent
$V(X \pm Y) = V(X) + V(Y) \pm 2COV(X, Y)$	
$V(XY) \neq V(X)V(Y)$	

Table: Variance and Covariance Expectation Rules

The Background Break - Multivariate Normal Distribution

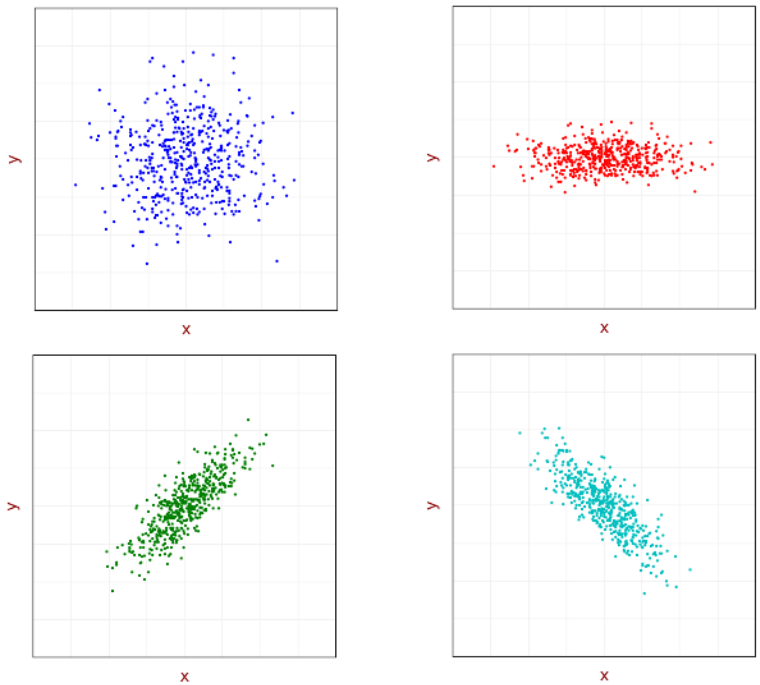
- Introduction to the Kalman Filter:
  - The Kalman Filter output is a random variable.
  - The mean of the random variable is the state estimate.
  - The variance represents the estimation uncertainty.
  - It provides us with the estimate and the level of confidence of its estimate.
- One-dimensional Kalman Filter equations include four uncertainty variables:
  1. $p_{n,n}$ is the variance of an estimate (the current state).
  2. $p_{n+1,n}$ is the variance of a prediction (the next state).
  3. r_n is the measurement variance.
  4. q is the process noise.
- Multivariate Kalman Filter
  - Describes the system state by a vector with more than one variable.
  - For example, object's position on the plane:

$$x = \begin{bmatrix} x \\ y \end{bmatrix}$$

  - The Kalman Filter output is a multivariate random variable.
  - A covariance matrix describes the squared uncertainty of the multivariate random variable.
- Uncertainty variables of multivariate Kalman Filter
  - The uncertainty variables are represented by covariance matrices:
    1. $P_{n,n}$ describes the squared uncertainty of an estimate.
    2. $P_{n+1,n}$ describes the squared uncertainty of a prediction.
    3. R_n describes the squared measurement uncertainty.
    4. Q describes the process noise.

The Background Break - Multivariate Normal Distribution

- Covariance is a measure of the strength of the correlation between two or more sets of random variates.
- On the x-y plane, variance in measurements exists due to random error.
- **Uncorrelated measurements:**
  - The x and y values don't depend on each other. The covariance of x and y equals zero.
  - For the blue data set, a circular distribution shape indicates equal variance.
  - For the red data set, an elliptic distribution shape indicates greater variance in x values.
- **Correlated measurements:**
  - Dependency exists between x and y values.
  - For the green data set, positive correlation and covariance due to concurrent increase.
  - For the cyan data set, negative correlation and covariance due to inverse changes.



The Background Break - Covariance

The covariance between population X and population Y with size N

$$\begin{aligned} \text{COV}(X, Y) &= \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \\ &= \frac{1}{N} \sum_{i=1}^{N} (x_i y_i) - \mu_x \mu_y \end{aligned}$$

The covariance of a sample with size N is normalized by $N - 1$

$$\begin{aligned} \text{COV}(X, Y) &= \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \\ &= \frac{1}{N-1} \left(\sum_{i=1}^{N} x_i y_i \right) - \frac{N}{N-1} \mu_x \mu_y \\ &= \frac{1}{N-1} \mathbf{x}^T y - \frac{N}{N-1} \mu_x \mu_y \quad \text{[In vector notation]} \\ &= \frac{1}{N-1} \mathbf{x}^T y \quad \text{[For a zero-mean random variable]} \end{aligned}$$

The Background Break - Covariance Matrix

A covariance matrix is a square matrix that represents the covariance between each pair of elements in a given multivariate random variable.
For a two-dimensional random variable, the covariance matrix is given by

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix} = \begin{bmatrix} \text{VAR}(x) & \text{COV}(x,y) \\ \text{COV}(y,x) & \text{VAR}(y) \end{bmatrix}$$

Note that the off-diagonal entries of the covariance matrix are equal since $\text{COV}(x,y) = \text{COV}(y,x)$. If x and y are uncorrelated, the off-diagonal entries of the covariance matrix are zero.

Properties of the covariance matrix:

- The diagonal entries of this covariance matrix are the variances of the components of the multivariate random variable:
$$\Sigma_{ii} = \sigma_i^2$$
- Since the diagonal entries are all non-negative, the trace (the sum of diagonal entries) of this covariance matrix is non-negative:
$$\text{tr}(\mathbf{\Sigma}) = \sum_{i=1}^{n} \Sigma_{ii} \geq 0$$
- Since $\Sigma_{ij} = \sigma_{ij} = \sigma_{ji} = \Sigma_{ji}$, the covariance matrix is symmetric:
$$\mathbf{\Sigma} = \mathbf{\Sigma}^{\mathbf{T}}$$
- The covariance matrix is **positive semidefinite**. The matrix **A** is called positive semidefinite if $\mathbf{v}^T\mathbf{A}\mathbf{v} \geq 0$, for any vector v. **The eigenvalues of A are non-negative.**

The Background Break - Expectation Algebra - Covariance Matrix and Expectation

- Assume a vector $\mathbf{x}$ with k elements:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$$

- The covariance matrix of the vector $\mathbf{x}$ is:

$$\begin{aligned} COV(\mathbf{x}) =& E\left((\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)^T\right) \\ =& E\left(\begin{bmatrix} (x_1 - \mu_{x_1}) \\ (x_2 - \mu_{x_2}) \\ \vdots \\ (x_k - \mu_{x_k}) \end{bmatrix} \begin{bmatrix} (x_1 - \mu_{x_1}) & (x_2 - \mu_{x_2}) & \cdots & (x_k - \mu_{x_k}) \end{bmatrix} \right) \end{aligned}$$

The Background Break - Multivariate Normal Distribution

Univariate Gaussian distribution:

$$p(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

The n - dimensional multivariate normal distribution:

$$p(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

- $\mathbf{x}$ is an n-dimensional random vector,
- $\boldsymbol{\mu}$ is an n-dimensional mean vector,
- $\boldsymbol{\Sigma}$ is a square $n \times n$ covariance matrix,
- $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix,
- $|\boldsymbol{\Sigma}| \equiv \det(\boldsymbol{\Sigma})$ is the determinant of $\boldsymbol{\Sigma}$.

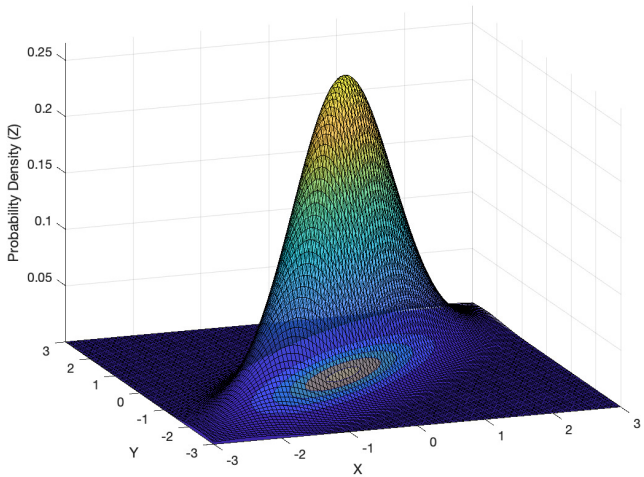


Figure: Bivariate Gaussian distribution ($\rho = 0.4$).

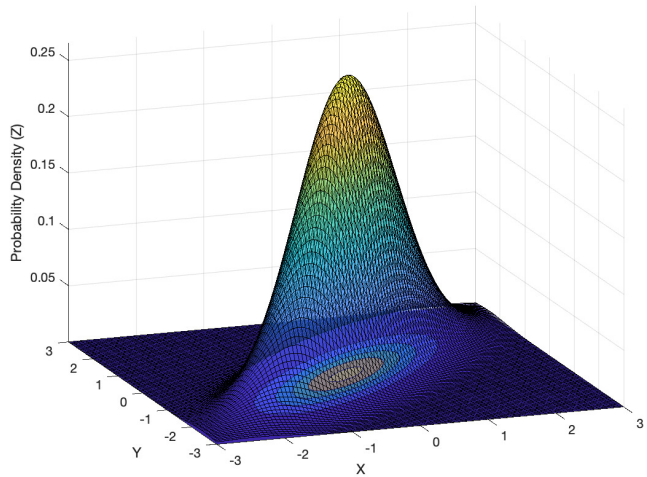


Figure: Bivariate Gaussian distribution ($\rho = 0.8$).

The Background Break - Bivariate Normal Distribution

For univariate distribution, the area between the 1σ boundaries is 68.26% of the total area under the Gaussian function.

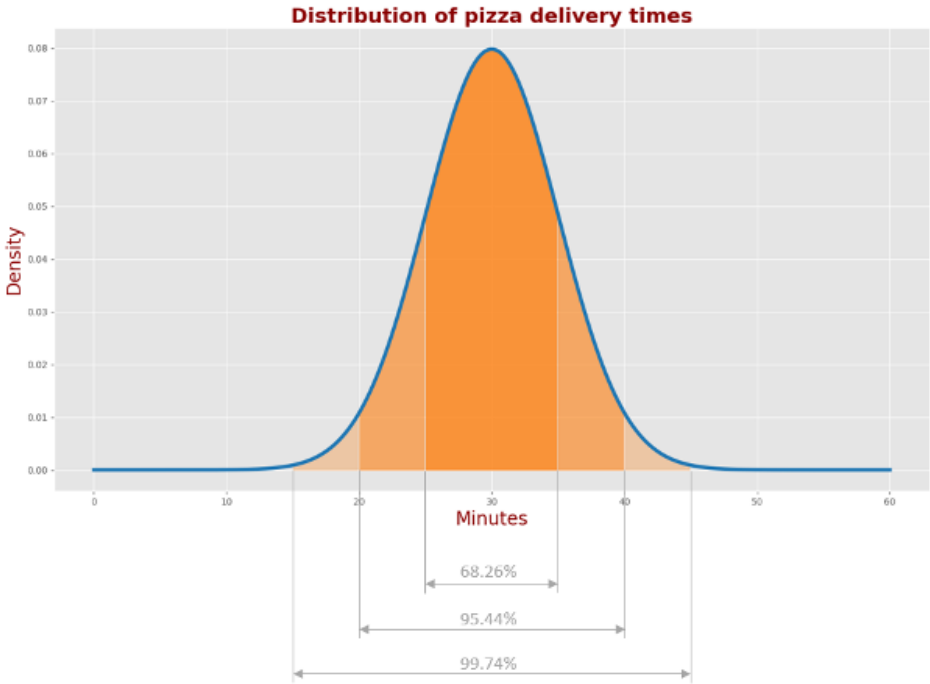


Figure: Univariate Gaussian.

The probability of the bivariate normal distribution is a volume of the 3D Gaussian function. For example, the volume of the 3D Gaussian function sliced at 1σ level is 39.35% of the total volume of the 3D Gaussian function.

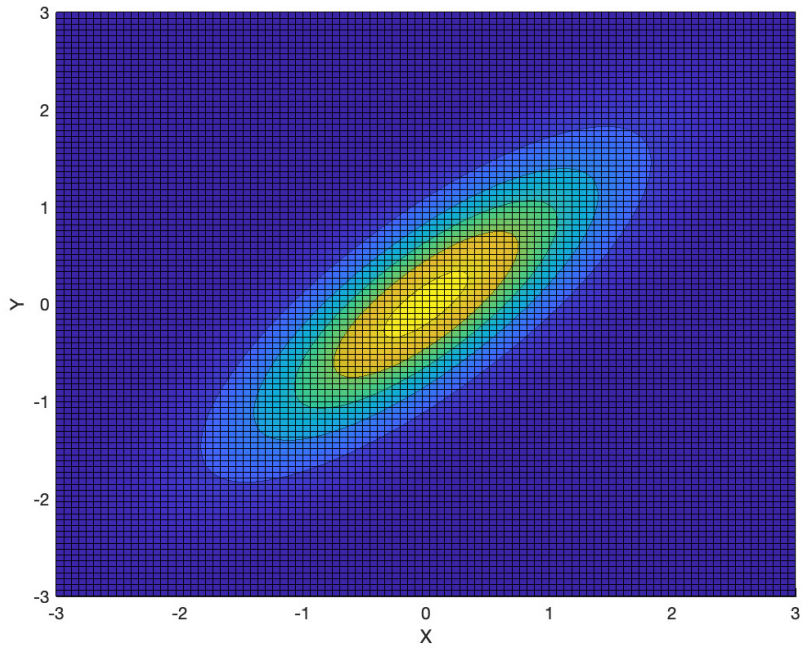


Figure: Bivariate Gaussian distribution - Contour ($\rho = 0.4$).

[Code: Multivariate KF/bivariateGaussian_withContours.m]

The Background Break - Bivariate Normal Distribution - Covariance Ellipse

- The covariance ellipse represents an *iso-contour* of the Gaussian distribution and allows visualization of a 1σ confidence interval in two dimensions.
- It provides a geometric interpretation of the covariance matrix.
- Any ellipse can be described by four parameters:
  - Ellipse center μ_x, μ_y
  - Half-major axis a
  - Half-minor axis b
  - Orientation angle θ

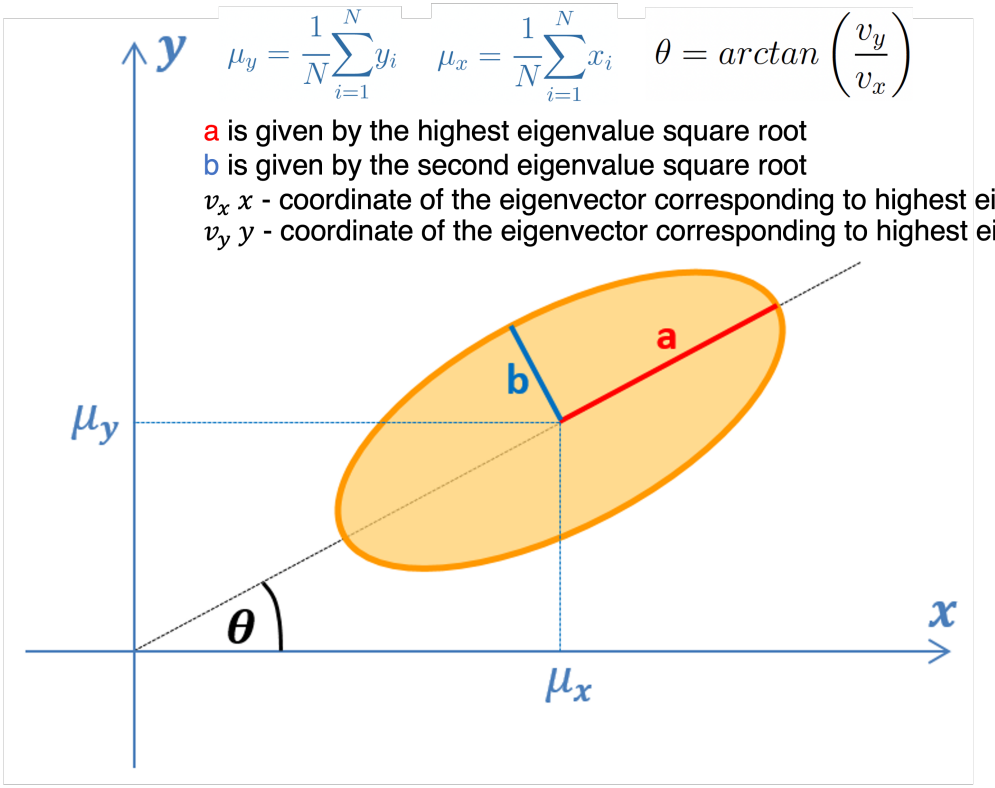


Figure: Covariance ellipse.

The Background Break - Confidence Ellipse and Finding Probability Boundaries

- Often there is an interest in finding the boundaries of specific probability. For example, for 95% probability, we should find the boundary that includes 95% of Gaussian function volume.
- The projection of this boundary onto the $x - y$ plane is the confidence ellipse.
- **Find an elliptical scale factor k, that extends the covariance ellipse to the confidence ellipse associated with 95% probability.**

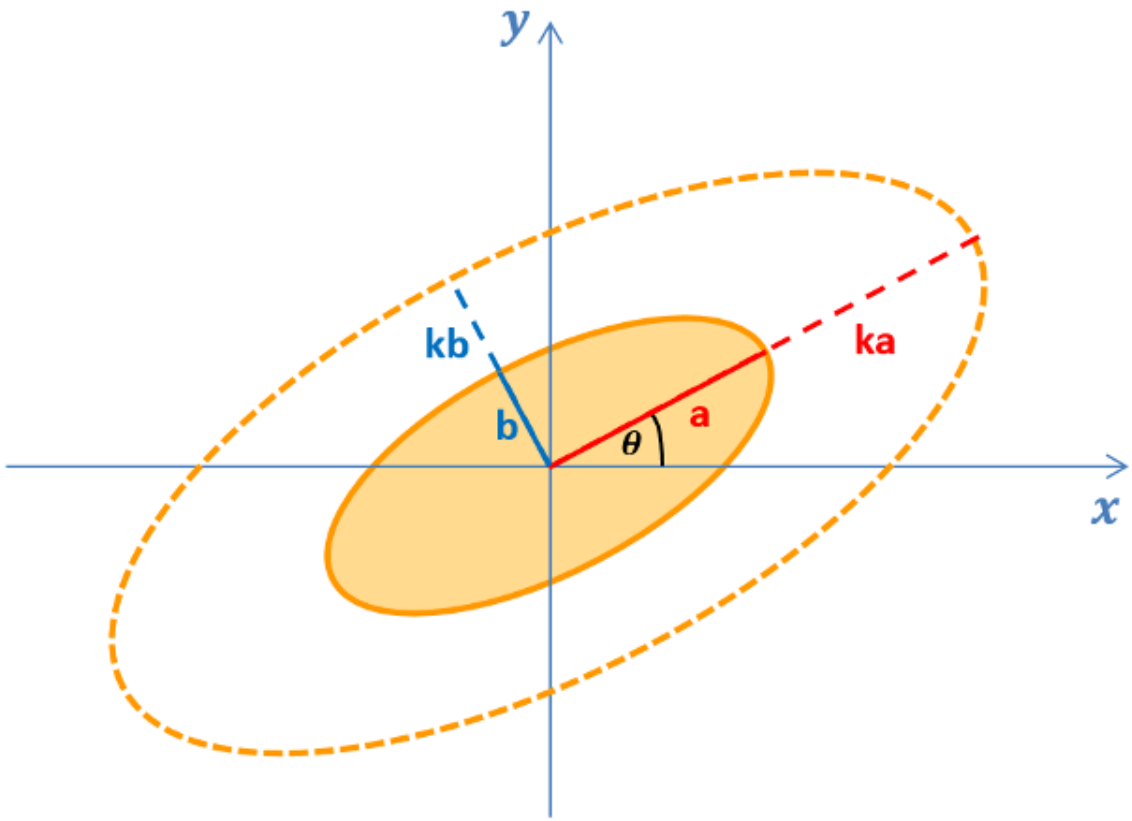


Figure: Covariance ellipse.

The Background Break - Confidence Ellipse and Finding Probability Boundaries

- As σ_x and σ_y represent the standard deviations of stochastically independent random variables, the addition theorem for the chi-square distribution may be used to show that the probability associated with a confidence ellipse is given by:

$$p = 1 - \exp\left(-\frac{1}{2}k^2\right)$$

- For a covariance ellipse $k = 1$, the probability associated with a covariance ellipse is:

$$p = 1 - \exp\left(-\frac{1}{2}\right) = 39.35\%$$

- For a given probability, we can find an elliptical scale factor:

$$k = \sqrt{-2\ln(1-p)}$$

- For the probability of 95%:

$$k = \sqrt{-2\ln(1-0.95)} = 2.45$$

The properties of the confidence ellipse associated with 95% probability are:

- Ellipse center (μ_x, μ_y) is similar to the covariance ellipse.
- Orientation angle θ is similar to the covariance ellipse.
- Half-major axis length is $2.45a$ – a scaled half-major axis of the covariance ellipse.
- Half-minor axis length is $2.45b$ – a scaled half-minor axis of the covariance ellipse.

State Extrapolation Equation or Predictor Equation or Transition Equation or Dynamic Model or State Space Model

Using the state extrapolation equation, we can predict the **next system state** based on the knowledge of the **current state**. It extrapolates the state vector from the present (time step n) to the future (time step $n+1$).
The State Extrapolation Equation can be expressed as:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\mathbf{u}_n + \mathbf{w}_n \tag{1}$$

where:

- $\hat{\mathbf{x}}_{n+1,n}$ is a predicted system state vector at time step $n+1$.
- $\hat{\mathbf{x}}_{n,n}$ is an estimated system state vector at time step n.
- $\mathbf{u}_n$ is a control variable or input variable - a measurable (deterministic) input to the system.
- $\mathbf{w}_n$ is a process noise or disturbance - an unmeasurable input that affects the state.
- $\mathbf{F}$ is a state transition matrix.
- $\mathbf{G}$ is a control matrix or input transition matrix (mapping control to state variables).

**The process noise w_n does not typically appear directly in the equations of interest. Instead, this term is used to model the uncertainty in the Covariance Extrapolation Equation.*

State Extrapolation Equation

- The state variables may represent attributes of the system that we wish to know. For example, a moving vehicle has three attributes: position, velocity, and acceleration.

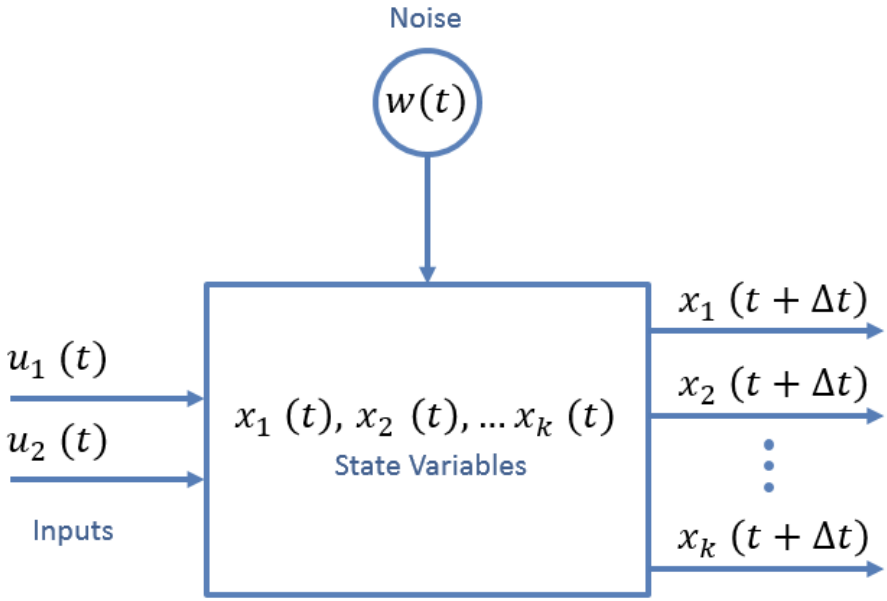


Figure: Kalman Filter Extrapolation

State Extrapolation Equation

Which attributes are the state variables, and which attributes are the input to the system?

- Moving mechanical systems have attributes such as position, velocity, acceleration, and drag.
- A force that acts on a system should be considered an external forcing function, i.e., an input to the system that controls the state vector (position and velocity in the constant acceleration case).
- Newton's second law tells us that $F = ma$. Thus we can consider acceleration as an external input to the system.
- The position and the velocity are the primary state variables of interest.

In a spring system, the force applied to the spring $F(t)$ is an input $u(t)$, while the spring displacement $x(t)$ is the system state.

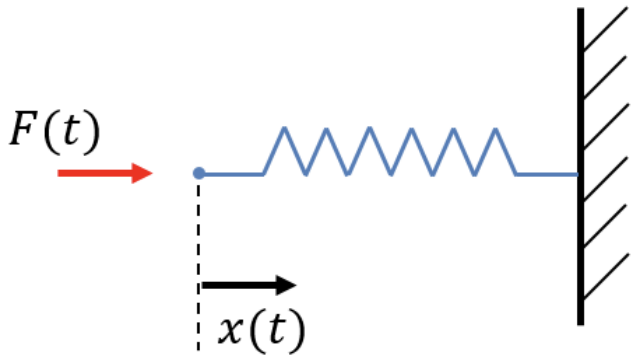


Figure: Spring System

For a falling object, the inputs are the gravitational force F_g and the drag force $F_{\text{drag}}(t)$, while the object height $h(t)$ and velocity $v(t)$ are the system states.

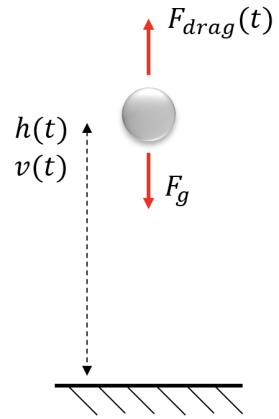


Figure: Falling Object

State Extrapolation Equation (Example - airplane - no control input)

Let's define the State Extrapolation Equation for an airplane moving in three-dimensional space with constant acceleration (i.e., there is no control input, $u_n = 0$).
The state vector $\hat{\mathbf{x}}_n$ (describing the estimated airplane position, velocity, and acceleration in a cartesian coordinate system (x, y, z)), the state transition matrix $\mathbf{F}$ is:

$$\hat{\mathbf{x}}_n = \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \\ \hat{z}_n \\ \dot{\hat{x}}_n \\ \dot{\hat{y}}_n \\ \dot{\hat{z}}_n \\ \ddot{\hat{x}}_n \\ \ddot{\hat{y}}_n \\ \ddot{\hat{z}}_n \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{z}_{n+1,n} \\ \dot{\hat{x}}_{n+1,n} \\ \dot{\hat{y}}_{n+1,n} \\ \dot{\hat{z}}_{n+1,n} \\ \ddot{\hat{x}}_{n+1,n} \\ \ddot{\hat{y}}_{n+1,n} \\ \ddot{\hat{z}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{y}_{n,n} \\ \hat{z}_{n,n} \\ \dot{\hat{x}}_{n,n} \\ \dot{\hat{y}}_{n,n} \\ \dot{\hat{z}}_{n,n} \\ \ddot{\hat{x}}_{n,n} \\ \ddot{\hat{y}}_{n,n} \\ \ddot{\hat{z}}_{n,n} \end{bmatrix}$$

State Extrapolation Equation (Example - airplane - no control input)

The state update equations after matrix multiplication are given by:

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{x}}_{n,n}\Delta t^2$$

$$\hat{y}_{n+1,n} = \hat{y}_{n,n} + \hat{\dot{y}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{y}}_{n,n}\Delta t^2$$

$$\hat{z}_{n+1,n} = \hat{z}_{n,n} + \hat{\dot{z}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{z}}_{n,n}\Delta t^2$$

$$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta t$$

$$\hat{\dot{y}}_{n+1,n} = \hat{\dot{y}}_{n,n} + \hat{\ddot{y}}_{n,n}\Delta t$$

$$\hat{\dot{z}}_{n+1,n} = \hat{\dot{z}}_{n,n} + \hat{\ddot{z}}_{n,n}\Delta t$$

$$\hat{\ddot{x}}_{n+1,n} = \hat{\ddot{x}}_{n,n}$$

$$\hat{\ddot{y}}_{n+1,n} = \hat{\ddot{y}}_{n,n}$$

$$\hat{\ddot{z}}_{n+1,n} = \hat{\ddot{z}}_{n,n}$$

State Extrapolation Equation (Example - airplane - with control input)

Considering additional information about the airplane acceleration (based on the pilot's controls) based on the pilot's commands.

The state vector $\hat{\mathbf{x}}_n$ (describing the estimated airplane position and velocity a cartesian coordinate system (x, y, z)) is:

$$\hat{\mathbf{x}}_n = \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \\ \hat{z}_n \\ \hat{\dot{x}}_n \\ \hat{\dot{y}}_n \\ \hat{\dot{z}}_n \end{bmatrix}$$

The control vector u_n that describes the measured airplane acceleration in a cartesian coordinate system (x, y, z) is:

$$\mathbf{u}_n = \begin{bmatrix} \hat{\ddot{x}}_n \\ \hat{\ddot{y}}_n \\ \hat{\ddot{z}}_n \end{bmatrix}$$

The state transition matrix $\mathbf{F}$ is:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The control matrix $\mathbf{G}$ is:

$$\mathbf{G} = \begin{bmatrix} 0.5\Delta t^2 & 0 & 0 \\ 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 0.5\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix}$$

State Extrapolation Equation (Example - airplane - with control input)

The state extrapolation equation is:

$$\hat{x}_{n+1,n} = \mathbf{F}\hat{x}_{n,n} + \mathbf{G}u_{n,n}$$

Given the system dynamics, the state update can be expressed as:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{z}_{n+1,n} \\ \dot{\hat{x}}_{n+1,n} \\ \dot{\hat{y}}_{n+1,n} \\ \dot{\hat{z}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{y}_{n,n} \\ \hat{z}_{n,n} \\ \dot{\hat{x}}_{n,n} \\ \dot{\hat{y}}_{n,n} \\ \dot{\hat{z}}_{n,n} \end{bmatrix} + \begin{bmatrix} 0.5\Delta t^2 & 0 & 0 \\ 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 0.5\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{x}_n \\ \ddot{y}_n \\ \ddot{z}_n \end{bmatrix}$$

State Extrapolation Equation (Example – Falling Object)

Consider a free-falling object. The state vector includes the altitude h and the object's velocity $\dot{h}$:

$$\hat{\mathbf{x}}_n = \begin{bmatrix} \hat{h}_n \\ \dot{\hat{h}}_n \end{bmatrix} \tag{2}$$

The state transition matrix $\mathbf{F}$ is:

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{3}$$

The control matrix $\mathbf{G}$ is:

$$\mathbf{G} = \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} \tag{4}$$

The input variable u_n is:

$$u_n = \begin{bmatrix} g \end{bmatrix} \text{ [the gravitational acceleration]} \tag{5}$$

We don't have a sensor that measures acceleration, but we know that for a falling object, acceleration equals g.
The state extrapolation equation is:

$$\begin{bmatrix} \hat{h}_{n+1,n} \\ \dot{\hat{h}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{h}_{n,n} \\ \dot{\hat{h}}_{n,n} \end{bmatrix} + \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} \begin{bmatrix} g \end{bmatrix} \tag{6}$$

The matrix multiplication results in the following:

$$\hat{h}_{n+1,n} = \hat{h}_{n,n} + \dot{\hat{h}}_{n,n}\Delta t + 0.5\Delta t^2 g$$

$$\dot{\hat{h}}_{n+1,n} = \dot{\hat{h}}_{n,n} + \Delta t g$$

Linear Time-Invariant Systems

The Linear Kalman Filter assumes the Linear Time-Invariant (LTI) system model.
So, what is "linear," and what is "time-invariant"?

Linear Systems

Linear systems are described by systems of equations in which the variables are never multiplied with each other but only with constants and then summed up. They are used to describe both static and dynamic relationships between variables.
A linear system is a system whose output function $y(t)$ satisfies the following equation:

$$y(t) = F(a \times g(t) + b \times h(t)) = a \times F(g(t)) + b \times F(h(t))$$

- a and b are constant real numbers.
- g and h are any arbitrary functions of an independent variable t.

A linear system follows two basic rules:

1. You can "factor out" constant multiplicative scale factors.
2. The system's response to a sum of inputs is the sum of the responses to each input separately.

Linear Time-Invariant Systems

Time-Invariant Systems

A time-invariant system has a system function that is not a direct function of time. For example, an amplifier with gain $G = 10$. This system is time-invariant. Although the system's output changes with time, the system function is not time-dependent.

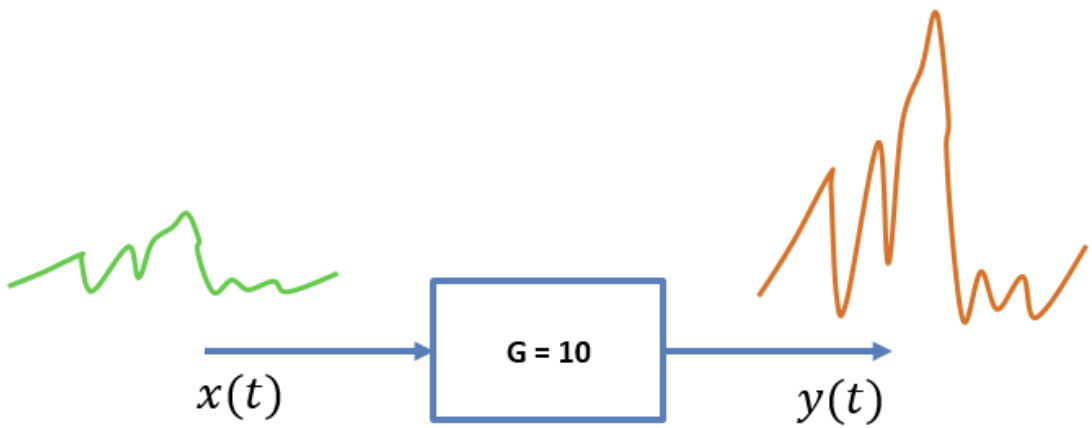


Figure: Amplifier

A time-invariant system is one where a time delay (or shift) in the input sequence causes an equivalent time delay in the system's output sequence.

See Appendix C of the book "Kalman Filter from the ground up" by Alex Becker for generalized dynamic model derivation for any linear dynamic model.

Covariance Extrapolation Equation

The general form of the Covariance Extrapolation Equation is given by:

$$\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T + \mathbf{Q} \tag{2}$$

where:

- $\mathbf{P}_{n,n}$ is the squared uncertainty of an estimate (covariance matrix) of the current state.
- $\mathbf{P}_{n+1,n}$ is the squared uncertainty of a prediction (covariance matrix) for the next state.
- $\mathbf{F}$ is the state transition matrix that we derived in Appendix C ("Modeling linear dynamic systems").
- $\mathbf{Q}$ is the process noise matrix.

The estimate covariance without process noise

Assuming $\mathbf{Q} = 0$: $\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T$
The covariance of a vector $\mathbf{x}$ is defined as:

$$COV(\mathbf{x}) = \mathbb{E}\left[(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^T\right] \text{ (vector } \mathbf{x} \text{ is a system state vector)}$$

Therefore, the covariance matrix $\mathbf{P}_{n,n}$ is:

$$\mathbf{P}_{n,n} = \mathbb{E}\left[(\hat{\mathbf{x}}_{n,n} - \boldsymbol{\mu}_{\mathbf{x}_{n,n}})(\hat{\mathbf{x}}_{n,n} - \boldsymbol{\mu}_{\mathbf{x}_{n,n}})^T\right]$$

For the next state, this becomes:

$$\mathbf{P}_{n+1,n} = \mathbb{E}\left[(\hat{\mathbf{x}}_{n+1,n} - \boldsymbol{\mu}_{\mathbf{x}_{n+1,n}})(\hat{\mathbf{x}}_{n+1,n} - \boldsymbol{\mu}_{\mathbf{x}_{n+1,n}})^T\right]$$

Following the state extrapolation equation:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\hat{\mathbf{u}}_{n,n}$$

The update for the covariance matrix can be represented as:

$$\begin{aligned}
\mathbf{P}_{n+1,n} &= \mathbb{E}\left[(\mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\hat{\mathbf{u}}_{n,n} - \mathbf{F}\mu_{x,n,n} - \mathbf{G}\hat{\mathbf{u}}_{n,n}) \times (\mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\hat{\mathbf{u}}_{n,n} - \mathbf{F}\mu_{x,n,n} - \mathbf{G}\hat{\mathbf{u}}_{n,n})^T\right] \\
&= \mathbb{E}\left[(\mathbf{F}(\hat{\mathbf{x}}_{n,n} - \mu_{x,n}))(\mathbf{F}(\hat{\mathbf{x}}_{n,n} - \mu_{x,n}))^T\right] \\
&= \mathbb{E}\left[(\mathbf{F}(\hat{\mathbf{x}}_{n,n} - \mu_{x,n}))(\hat{\mathbf{x}}_{n,n} - \mu_{x,n})^T\mathbf{F}^T\right] \quad ((\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T) \\
&= \mathbf{F}\mathbb{E}\left[((\hat{\mathbf{x}}_{n,n} - \mu_{x,n}))(\hat{\mathbf{x}}_{n,n} - \mu_{x,n})^T\right]\mathbf{F}^T \\
&= \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T
\end{aligned}$$

Constructing the process noise matrix $\mathbf{Q}$

The system dynamics is described by:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\hat{\mathbf{u}}_{n,n} + \underbrace{\mathbf{w}_n}_{\text{Process noice}}$$

In 1D Kalman Filter, the process noise variance is denoted by $\mathbf{q}$ while for 3D case by $\mathbf{Q}$.
The process noise variance has a critical influence on the Kalman Filter performance.

- Too small q causes a lag error (Example 7).
- If the q value is too high, the Kalman Filter follows the measurements (Example 8) and produces noisy estimations.

The process noise can be independent between different state variables. In this case, the process noise covariance matrix $\mathbf{Q}$ is a diagonal matrix:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 & \cdots & 0 \\ 0 & q_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_{kk} \end{bmatrix} \tag{8.32}$$

The process noise can also be dependent. For example, the constant velocity model assumes zero acceleration ($a = 0$). However, a random variance in acceleration (σ_a^2) causes a variance in velocity and position. In this case, the process noise is correlated with the state variables.

Models for the environmental process noise - Discrete Noise

There are two models for the environmental process noise; Discrete and Continuous noise model.

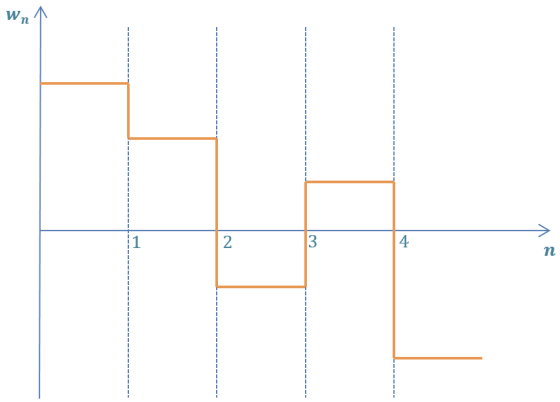


Figure: Discrete Noise

The process noise covariance matrix $\mathbf{Q}$ for the constant velocity model can be expressed as:

$$\mathbf{Q} = \begin{bmatrix} V(x) & \text{COV}(x,v) \\ \text{COV}(v,x) & V(v) \end{bmatrix}$$

where $V(x)$ and $V(v)$ represent the position and velocity variance, respectively, and COV(x, v) is the covariance between position and velocity. These terms are expressed in terms of the random acceleration variance σ_a^2 of the model.

$$\begin{aligned} V(v) &= \sigma_v^2 = E(v^2) - \mu_v^2 \\ &= E\left((a\Delta t)^2\right) - (\mu_a \Delta t)^2 \\ &= \Delta t^2 \left(E(a^2) - \mu_a^2\right) \\ &= \Delta t^2 \sigma_a^2 \\ V(x) &= \sigma_x^2 = E(x^2) - \mu_x^2 \\ &= E\left(\left(\frac{1}{2} a\Delta t^2\right)^2\right) - \left(\frac{1}{2}\mu_a \Delta t^2\right)^2 \\ &= E\left((a\Delta t)^2\right) - (\mu_a \Delta t)^2 \\ &= \Delta t^4 \left(\frac{1}{4}\right)\left(E(a^2) - \mu_a^2\right) \\ &= \Delta t^4 \left(\frac{1}{4}\right)\sigma_a^2 \\ \text{COV}(x,v) &= \text{COV}(v,x) = E(xv) - \mu_x \mu_v \\ &= \left(E\left(\frac{1}{2} a\Delta t^2 a\Delta t\right) - \left(\frac{1}{2}\mu_a \Delta t^2 \mu_a \Delta t\right)\right) \\ &= \frac{\Delta t^3}{2}\left(E(a^2) - \mu_a^2\right) = \frac{\Delta t^3}{2}\sigma_a^2 \end{aligned}$$

Discrete Noise-How to construct **Q** matrix

By substituting the results into **Q** matrix:

$$\mathbf{Q} = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix}$$

There are two methods for a faster construction of the Q matrix

- Projection using the state transition matrix (without control input)
- Projection using the control matrix (with control input)

Discrete Noise: How to construct Q matrix

Without Control Input: If the dynamic model doesn't include a control input, project the random variance in acceleration σ_a^2 on the dynamic model using state transition matrix $\mathbf{F}$.

- Let us define a matrix $\mathbf{Q}_a$:

$$\mathbf{Q}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \sigma_a^2$$

- The process noise matrix is $\mathbf{Q} = \mathbf{F}\mathbf{Q}_a\mathbf{F}^T$.
- For the motion model, $\mathbf{F}$ matrix is:

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

$$Q = \sigma_a^2 \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \Delta t & 1 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix}$$

With Control Input: When the dynamic model includes a control input, we can compute $\mathbf{Q}$ matrix even faster by projecting the random variance in acceleration σ_a^2 using the control matrix $\mathbf{G}$.

- $\mathbf{Q} = \mathbf{G}\sigma_a^2\mathbf{G}^T$ where $\mathbf{G}$ is the control (or input transition) matrix.
- For the motion model, the $\mathbf{G}$ matrix is:

$$\mathbf{G} = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}$$

Resulting Q matrix:

$$\mathbf{Q} = \sigma_a^2 \mathbf{G}\mathbf{G}^T$$

$$= \sigma_a^2 \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \begin{bmatrix} \frac{\Delta t^2}{2} & \Delta t \end{bmatrix}$$

$$= \sigma_a^2 \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix}$$

Continuous Noise: How to construct $\mathbf{Q}$ matrix

The continuous model assumes that the noise changes continuously over time.

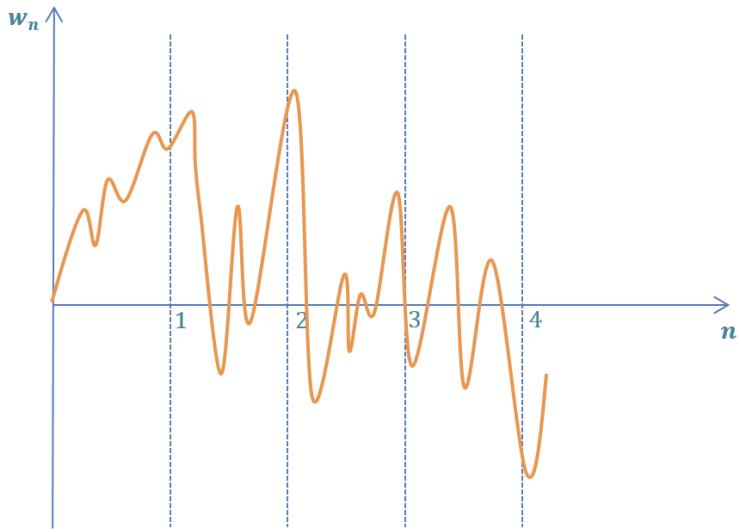


Figure: Continuous Noise

To derive the process noise covariance matrix for the continuous model $\mathbf{Q}_c$, we need to integrate the discrete process noise covariance matrix $\mathbf{Q}$ over time.

$$\mathbf{Q}_c = \int_0^{\Delta t} \mathbf{Q} dt = \int_0^{\Delta t} \sigma_a^2 \begin{bmatrix} \frac{t^4}{4} & \frac{t^3}{2} \\ \frac{t^3}{2} & t^2 \end{bmatrix} dt = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^5}{20} & \frac{\Delta t^4}{8} \\ \frac{\Delta t^4}{8} & \frac{\Delta t^3}{3} \end{bmatrix}$$

Selecting Process Noise Variance and Noise Model

Process Noise Variance (σ_a^2)

- Calculated using stochastic statistics or chosen based on engineering practice.
- Depends on target characteristics and model completeness.
- For **maneuvering targets** (e.g., airplanes), σ_a^2 should be relatively high.
- For **non-maneuvering targets** (e.g., rockets), a smaller σ_a^2 is appropriate.
- Environmental influences like air drag reduce process noise randomness if included in the model.

Choosing the Noise Model

- No definitive answer: recommend trying both discrete and continuous models.
- Use **discrete noise model** when Δt is small.
- Prefer **continuous noise model** when Δt is high.
- Experiment to see which model performs better with your Kalman Filter.

Measurement Equation

In 1D KF, measurement denoted by z_n

- Represented the true system state plus random measurement noise v_n.

Measurement Noise Variance (r_n):

- Can be **constant** for all measurements (e.g., scales with a precision of 0.5 kg).
- May **vary** for each measurement (e.g., thermometer with 0.5% precision).
- In variable cases, the noise variance depends on the measured quantity (e.g., temperature).

The Measurement Equation is given by:

$$\mathbf{z}_n = \mathbf{H}\mathbf{x}_n + \mathbf{v}_n$$

where:

- $\mathbf{z}_n$ is a measurement vector.
- $\mathbf{x}_n$ is a true system state (hidden state).
- $\mathbf{v}_n$ is a random noise vector.
- $\mathbf{H}$ is an observation matrix.

The Observation Matrix

In many scenarios, the measured value is not the direct system state we are interested in. For instance, a digital electric thermometer measures an electric current, which needs to be translated to temperature – the actual system state of interest.

The observation matrix **H** serves the crucial function of transforming the system state (input) into the measurement (output) through linear transformations. This process ensures that the Kalman Filter can work with measurements that represent the system states accurately.

Scaling: A range meter sends a signal toward a destination and receives a reflected echo. The measurement is the time delay between the transmission and reception of the signal. The system state is the range.
In this case, we need to perform a scaling:

$$z_n = \left[\frac{2}{c}\right] x_n + v_n$$

$$H = \left[\frac{2}{c}\right]$$

where:

- c is the speed of light.
- x_n is the range.
- z_n is the measured time delay.

The Observation Matrix

State selection: Sometimes certain states are measured while others are not. For example, the first, third, and fifth states of a 5D state vector are measurable, while the second and fourth states are not measurable:

$$z_n = \mathbf{H}\mathbf{x}_n + \mathbf{v}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \mathbf{v}_n = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} + \mathbf{v}_n$$

Combination of states: Sometimes some combination of states can be measured instead of each separate state. For example, maybe the lengths of the sides of a triangle are the states, and only the total perimeter can be measured:

$$z_n = \mathbf{H}\mathbf{x}_n + \mathbf{v}_n = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \mathbf{v}_n = (x_1 + x_2 + x_3) + \mathbf{v}_n$$

Covariance Equations

The terms **w** and **v**, which correspond to the process and measurement noise, do not typically appear directly in the calculations since they are unknown. Instead, these terms are used to model the uncertainty (or noise) in the equations themselves.

All covariance equations are covariance matrices in the form of:

$$E(\mathbf{e}\mathbf{e}^T)$$

i.e., an expectation of a squared error.

- **Measurement uncertainty:**

$$\mathbf{R}_n = E(\mathbf{v}_n\mathbf{v}_n^T)$$

- **Process noise uncertainty:**

$$\mathbf{Q}_n = E(\mathbf{w}\mathbf{w}^T)$$

- **Estimation uncertainty:**

$$\mathbf{P}_n = E(\mathbf{e}_n\mathbf{e}_n^T) = E\left((x_n - \hat{x}_{n,n})(x_n - \hat{x}_{n,n})^T\right)$$

State Update Equation

The State Update Equation is given by:

$$\hat{\mathbf{x}}_{n,n} = \hat{\mathbf{x}}_{n,n-1} + \mathbf{K}_n(\mathbf{z}_n - \mathbf{H}\hat{\mathbf{x}}_{n,n-1}) \tag{3}$$

where:

- $\hat{\mathbf{x}}_{n,n}$ is an estimated system state vector at time step n.
- $\hat{\mathbf{x}}_{n,n-1}$ is a predicted system state vector at time step $n-1$.
- $\mathbf{K}_n$ is the Kalman Gain.
- $\mathbf{z}_n$ is a measurement.
- $\mathbf{H}$ is an observation matrix.

Covariance Update Equation

The Covariance Update Equation is given by:

$$\mathbf{P}_{n,n} = (\mathbf{I} - \mathbf{K}_n\mathbf{H})\mathbf{P}_{n,n-1}(\mathbf{I} - \mathbf{K}_n\mathbf{H})^T + \mathbf{K}_n\mathbf{R}_n\mathbf{K}_n^T \tag{4}$$

where:

- $\mathbf{P}_{n,n}$ is the covariance matrix of the current state estimation.
- $\mathbf{P}_{n,n-1}$ is the prior estimate covariance matrix of the current state (predicted at the previous state).
- $\mathbf{K}_n$ is the Kalman Gain.
- $\mathbf{H}$ is the observation matrix.
- $\mathbf{R}_n$ is the measurement noise covariance matrix.
- $\mathbf{I}$ is an Identity Matrix (the $n \times n$ square matrix with ones on the main diagonal and zeros elsewhere).

See derivation in the book.

Simplified Covariance Update Equation

A simplified form of the Covariance Update Equation is:

$$\mathbf{P}_{n,n} = (\mathbf{I} - \mathbf{K}_n\mathbf{H})\mathbf{P}_{n,n-1}$$

This equation is much more elegant and easier to remember and it performs well in many cases. However, a minor error in computing the Kalman Gain (due to round-off) can lead to huge computation errors. The subtraction $(\mathbf{I} - \mathbf{K}_n\mathbf{H})$ can lead to nonsymmetric matrices due to floating-point errors. This equation is numerically unstable!

The Kalman Gain

The Kalman Gain Equation is given by:

$$\mathbf{K}_n = \mathbf{P}_{n,n-1}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{n,n-1}\mathbf{H}^T + \mathbf{R}_n\right)^{-1} \tag{5}$$

Where:

- $\mathbf{K}_n$ is the Kalman Gain.
- $\mathbf{P}_{n,n-1}$ is the prior estimate covariance matrix of the current state (predicted at the previous step).
- $\mathbf{H}$ is the observation matrix.
- $\mathbf{R}_n$ is the measurement noise covariance matrix.

See derivation in the book.

Summary

The Kalman Filter operates in a "predict-correct" loop:

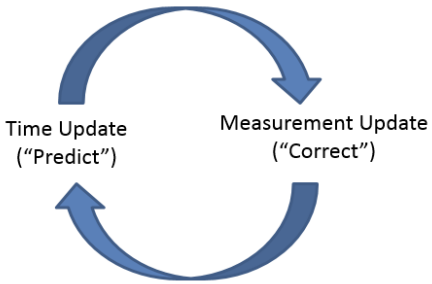


Figure: Predict Update Diagram

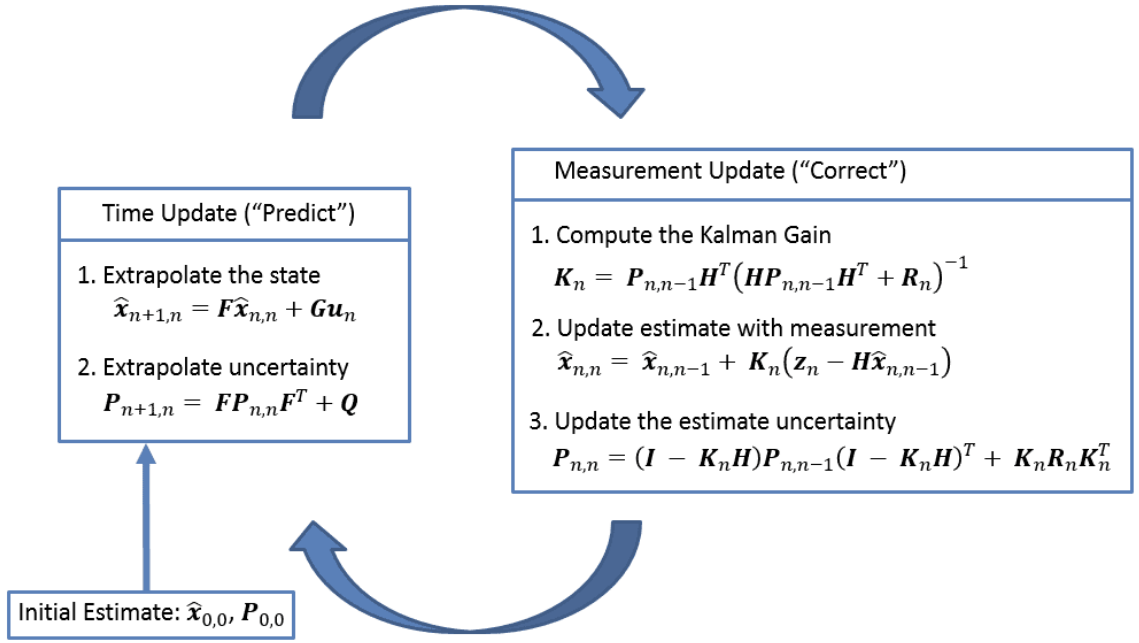


Figure: Kalman Filter diagram

- Once initialized, the Kalman Filter predicts the system state at the next step. It also provides the uncertainty of the prediction.
- Once the measurement is received, the Kalman Filter updates (or corrects) the prediction and the uncertainty of the current state. As well the Kalman Filter predicts the following states, and so on. The following diagram provides a complete picture of the Kalman Filter operation.

Summary

	Equation	Name	Alternative names in the literature
Predict	$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$	State Extrapolation	Predictor Equation Transition Equation Prediction Equation Dynamic Model State Space Model
	$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$	Covariance Extrapolation	Predictor Covariance Equation
Update	$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n \times (\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$	State Update	Filtering Equation
	$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1} \times (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$	Covariance Update	Corrector Equation
	$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T \times \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$	Kalman Gain	Weight Equation
Auxiliary	$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n$	Measurement Equation	
	$\boldsymbol{R}_n = E\left(\boldsymbol{v}_n\boldsymbol{v}_n^T\right)$	Measurement Covariance	Measurement Error
	$\boldsymbol{Q}_n = E\left(\boldsymbol{w}_n\boldsymbol{w}_n^T\right)$	Process Noise Covariance	Process Noise Error
	$\boldsymbol{P}_{n,n} = E\left(\boldsymbol{e}_n\boldsymbol{e}_n^T\right) = E\left(\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)^T\right)$	Estimation Covariance	Estimation Error

Figure: KF Equations

Multidimensional (Multivariate) Kalman Filter Examples

This section includes two examples:

- Vehicle location estimation using six-dimensional Kalman Filter without control input
- Rocket height estimation with a two-dimensional Kalman Filter with a control input

Example 9: Vehicle Location Estimation

Estimating vehicle's location on the XY plane using onboard location sensor reporting its X and Y coordinates. *Assumption: Constant acceleration dynamics.*

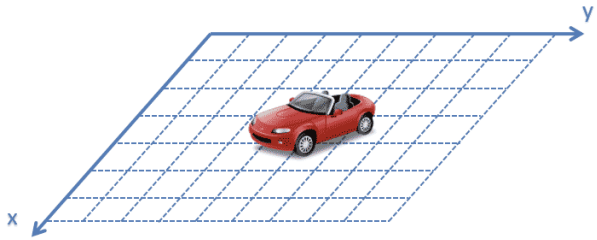


Figure: Vehicle location estimation.

KF Equations

1. The state extrapolation equation

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\hat{\mathbf{u}}_{n,n} + \mathbf{w}_n$$

In this example, without control input,

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n}$$

The system state $\mathbf{x}_n$ is defined by:

$$\mathbf{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix}$$

The extrapolated vehicle state for $n+1$ can be described as:

$$\begin{aligned}
\hat{x}_{n+1,n} &= \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{x}}_{n,n}\Delta t^2 \\
\hat{\dot{x}}_{n+1,n} &= \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta t \\
\hat{\ddot{x}}_{n+1,n} &= \hat{\ddot{x}}_{n,n} \\
\hat{y}_{n+1,n} &= \hat{y}_{n,n} + \hat{\dot{y}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{y}}_{n,n}\Delta t^2 \\
\hat{\dot{y}}_{n+1,n} &= \hat{\dot{y}}_{n,n} + \hat{\ddot{y}}_{n,n}\Delta t \\
\hat{\ddot{y}}_{n+1,n} &= \hat{\ddot{y}}_{n,n}
\end{aligned}$$

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\ddot{x}}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{\dot{y}}_{n+1,n} \\ \hat{\ddot{y}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\ddot{x}}_{n,n} \\ \hat{y}_{n,n} \\ \hat{\dot{y}}_{n,n} \\ \hat{\ddot{y}}_{n,n} \end{bmatrix}$$

Example 9: Vehicle Location Estimation

2. The covariance extrapolation equation

$$\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T + \mathbf{Q}$$

The estimate covariance matrix is:

$$\mathbf{P} = \begin{bmatrix} p_x & p_{x\dot{x}} & p_{x\ddot{x}} & p_{xy} & p_{x\dot{y}} & p_{x\ddot{y}} \\ p_{\dot{x}x} & p_{\dot{x}} & p_{\dot{x}\ddot{x}} & p_{\dot{x}y} & p_{\dot{x}\dot{y}} & p_{\dot{x}\ddot{y}} \\ p_{\ddot{x}x} & p_{\ddot{x}\dot{x}} & p_{\ddot{x}} & p_{\ddot{x}y} & p_{\ddot{x}\dot{y}} & p_{\ddot{x}\ddot{y}} \\ p_{yx} & p_{y\dot{x}} & p_{y\ddot{x}} & p_y & p_{y\dot{y}} & p_{y\ddot{y}} \\ p_{\dot{y}x} & p_{\dot{y}\dot{x}} & p_{\dot{y}\ddot{x}} & p_{\dot{y}y} & p_{\dot{y}} & p_{\dot{y}\ddot{y}} \\ p_{\ddot{y}x} & p_{\ddot{y}\dot{x}} & p_{\ddot{y}\ddot{x}} & p_{\ddot{y}y} & p_{\ddot{y}\dot{y}} & p_{\ddot{y}} \end{bmatrix}$$

Assuming the estimation errors in X and Y axes are uncorrelated

$$\mathbf{P} = \begin{bmatrix} p_x & p_{x\dot{x}} & p_{x\ddot{x}} & 0 & 0 & 0 \\ p_{\dot{x}x} & p_{\dot{x}} & p_{\dot{x}\ddot{x}} & 0 & 0 & 0 \\ p_{\ddot{x}x} & p_{\ddot{x}\dot{x}} & p_{\ddot{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_y & p_{y\dot{y}} & p_{y\ddot{y}} \\ 0 & 0 & 0 & p_{\dot{y}y} & p_{\dot{y}} & p_{\dot{y}\ddot{y}} \\ 0 & 0 & 0 & p_{\ddot{y}y} & p_{\ddot{y}\dot{y}} & p_{\ddot{y}} \end{bmatrix}$$

Assuming a discrete noise model—the noise is different at each time sample but is constant between time samples.
The process noise matrix for the two-dimensional constant acceleration model looks as

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 & \sigma_{x\ddot{x}}^2 & \sigma_{xy}^2 & \sigma_{x\dot{y}}^2 & \sigma_{x\ddot{y}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\ddot{x}}^2 & \sigma_{\dot{x}y}^2 & \sigma_{\dot{x}\dot{y}}^2 & \sigma_{\dot{x}\ddot{y}}^2 \\ \sigma_{\ddot{x}x}^2 & \sigma_{\ddot{x}\dot{x}}^2 & \sigma_{\ddot{x}}^2 & \sigma_{\ddot{x}y}^2 & \sigma_{\ddot{x}\dot{y}}^2 & \sigma_{\ddot{x}\ddot{y}}^2 \\ \sigma_{yx}^2 & \sigma_{yx\dot{x}}^2 & \sigma_{y\ddot{x}}^2 & \sigma_y^2 & \sigma_{y\dot{y}}^2 & \sigma_{y\ddot{y}}^2 \\ \sigma_{\dot{y}x}^2 & \sigma_{\dot{y}x\dot{x}}^2 & \sigma_{\dot{y}\ddot{x}}^2 & \sigma_{\dot{y}y}^2 & \sigma_{\dot{y}}^2 & \sigma_{\dot{y}\ddot{y}}^2 \\ \sigma_{\ddot{y}x}^2 & \sigma_{\ddot{y}x\dot{x}}^2 & \sigma_{\ddot{y}\ddot{x}}^2 & \sigma_{\ddot{y}y}^2 & \sigma_{\ddot{y}\dot{y}}^2 & \sigma_{\ddot{y}}^2 \end{bmatrix}$$

Assuming that the process noise in X and Y axes is not correlated:

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 & \sigma_{x\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\ddot{x}x}^2 & \sigma_{\ddot{x}\dot{x}}^2 & \sigma_{\ddot{x}}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_y^2 & \sigma_{y\dot{y}}^2 & \sigma_{y\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\dot{y}y}^2 & \sigma_{\dot{y}}^2 & \sigma_{\dot{y}\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\ddot{y}y}^2 & \sigma_{\ddot{y}\dot{y}}^2 & \sigma_{\ddot{y}}^2 \end{bmatrix}$$

Example 9: Vehicle Location Estimation

Based on "How to construct **Q** matrix" Sec. 86, we have:

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2$$

where:

- Δt is the time between successive measurements
- σ_a^2 is a random variance in acceleration

Now Let's look at Auxiliary equations:

- Measurement equation
- Measurement uncertainty equation

Measurement equation

$$\mathbf{z}_n = \mathbf{H}\mathbf{x}_n + \mathbf{v}_n$$

The only measurements provided to us are X and Y coordinates of the vehicle.

$$\mathbf{z}_n = \begin{bmatrix} x_{n,\text{measured}} \\ y_{n,\text{measured}} \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix}$$

The dimension of $\mathbf{z}_n$ is 2×1 and the dimension of $\mathbf{x}_n$ is 6×1. Therefore, the dimension of the observation matrix $\mathbf{H}$ shall be 2×6.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Example 9: Vehicle Location Estimation

Measurement uncertainty is represented by the measurement covariance matrix:

$$\mathbf{R}_n = \begin{bmatrix} \sigma_{x_m}^2 & \sigma_{yx_m}^2 \\ \sigma_{xy_m}^2 & \sigma_{y_m}^2 \end{bmatrix}$$

Assuming that the x and y measurements are uncorrelated:

$$\mathbf{R}_n = \begin{bmatrix} \sigma_{x_m}^2 & 0 \\ 0 & \sigma_{y_m}^2 \end{bmatrix}$$

In real-life applications, the measurement uncertainty can differ between measurements. In many systems, the measurement uncertainty depends on the measurement SNR, the angle between the sensor (or sensors) and target, signal frequency, and many other parameters. If we assume a constant measurement uncertainty:

$$\mathbf{R}_1 = \mathbf{R}_2 = \ldots = \mathbf{R}_{n-1} = \mathbf{R}_n = \mathbf{R}$$

3. **The Kalman Gain**

$$\mathbf{K}_n = \mathbf{P}_{n,n-1}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{n,n-1}\mathbf{H}^T + \mathbf{R}_n\right)^{-1}$$

4. **The state update equation**

$$\mathbf{x}_{n,n} = \hat{\mathbf{x}}_{n,n-1} + \mathbf{K}_n(\mathbf{z}_n - \mathbf{H}\hat{\mathbf{x}}_{n,n-1})$$

5. **The covariance update equation**

$$\mathbf{P}_{n,n} = (\mathbf{I} - \mathbf{K}_n\mathbf{H})\mathbf{P}_{n,n-1}(\mathbf{I} - \mathbf{K}_n\mathbf{H})^T + \mathbf{K}_n\mathbf{R}_n\mathbf{K}_n^T$$

Example 9: Vehicle Location Estimation

Numerical example: Assume a vehicle moving in a straight line in the X direction with a constant velocity. After traveling 400 meters, the vehicle turns left, with a turning radius of $R = 300$ meters. During the turning maneuver, the vehicle experiences acceleration due to the circular motion (angular acceleration). Recall from introductory physics that angular acceleration is given by:

$$\alpha = \frac{\Delta V}{R \Delta t}$$

The parameters are given as follows:

- The measurement period: $\Delta t = 1\,\text{s}$
- The random acceleration standard deviation: $\sigma_a = 0.2\,\text{m/s}^2$
- The measurement error standard deviation: $\sigma_{x_m} = \sigma_{y_m} = 3\,\text{m}$

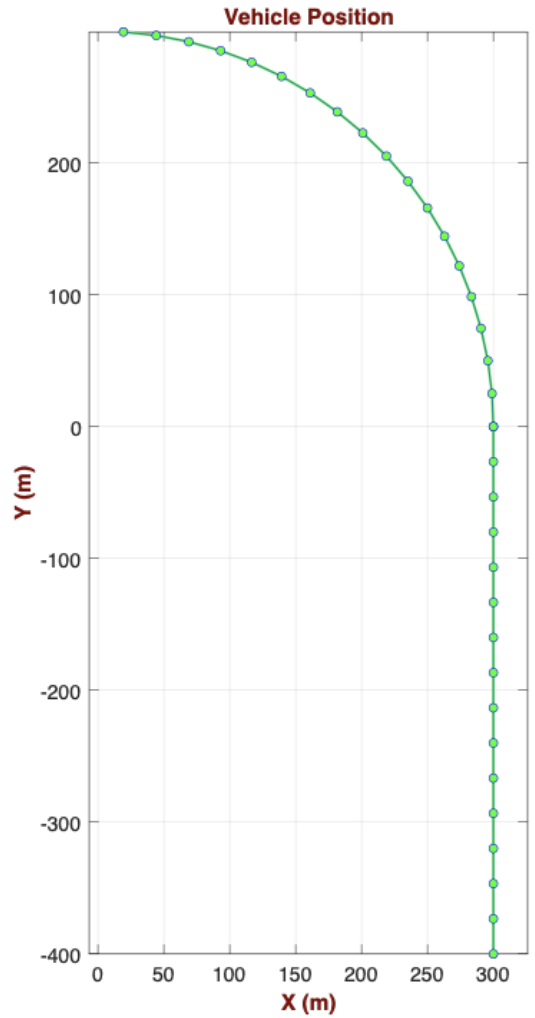


Figure: Vehicle trajectory

Example 9: Vehicle Location Estimation

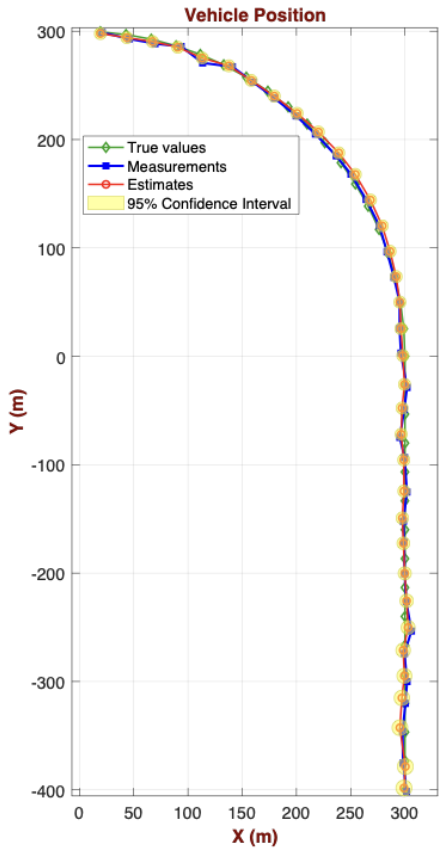
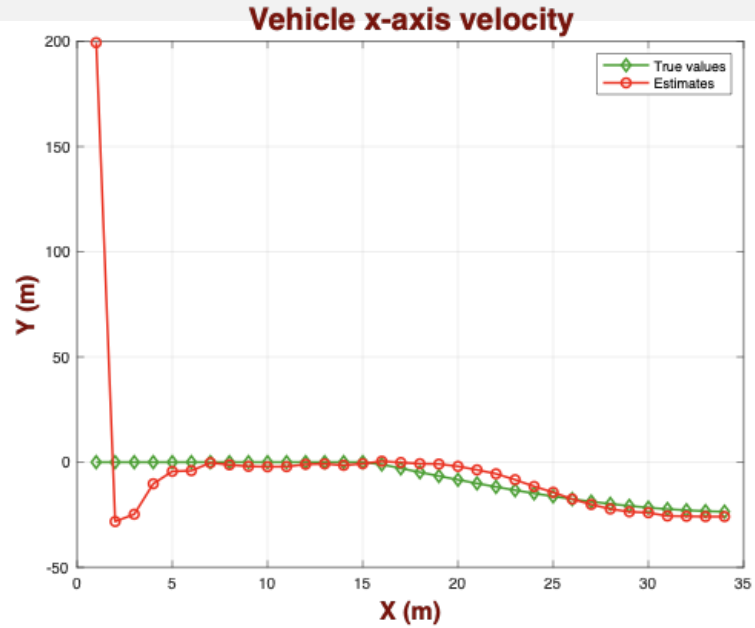
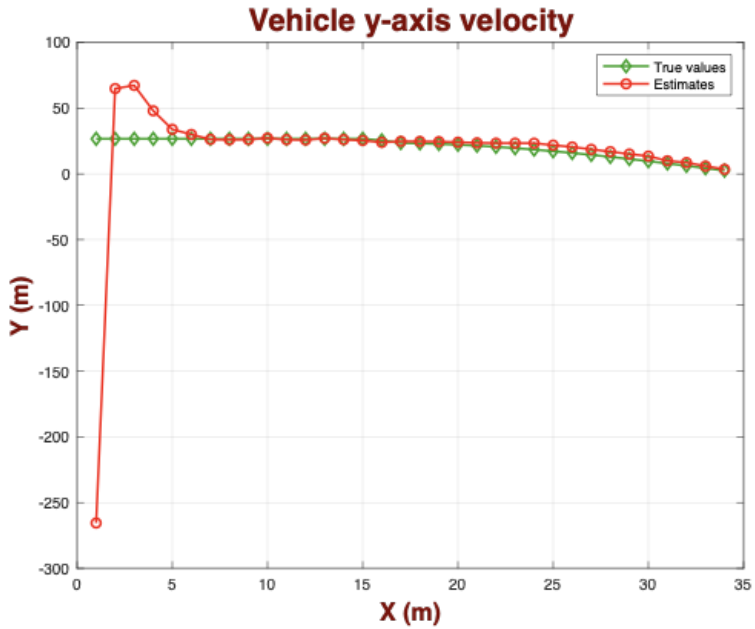


Figure: True value, measured values and estimates





The circles on the plot represent the 95% confidence ellipse. Since the x and y axes' measurement errors are equal, the confidence ellipse is a circle.

[Code: Multivariate KF/ex9_MultivariateKF_LocationEstimation.m]

Example 10: Rocket Altitude Estimation

- The rocket is equipped with an onboard altimeter that provides altitude measurements.
- The rocket is also equipped with an accelerometer that measures the rocket's acceleration. The accelerometer serves as a **control input to the Kalman Filter**.
- We assume constant acceleration dynamics.

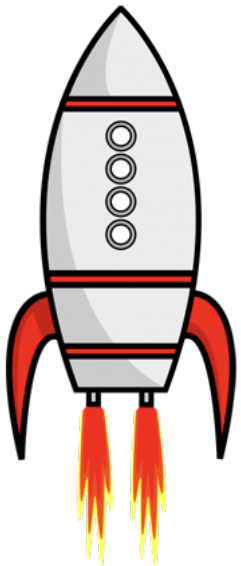


Figure: Rocket altitude estimation

- Accelerometers don't sense gravity. An accelerometer at rest on a table measures 1g upwards, while an accelerometer in free fall measures zero acceleration.
- Thus, we need to subtract the gravitational acceleration constant g from each accelerometer measurement.

The accelerometer measurement at time step n is:

$$a_n = \ddot{x} - g + \epsilon$$

where:

- $\ddot{x}$ is the actual acceleration of the object (the second derivative of the object position),
- g is the gravitational acceleration constant; $g = -9.8\,\mathrm{m/s^2}$,
- ϵ is the accelerometer measurement error.

Example 10: Rocket Altitude Estimation

1. The state extrapolation equation

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\hat{\mathbf{u}}_{n,n} + \mathbf{w}_n$$

Control variable u is based on the accelerometer measurement.
The system state $\mathbf{x}_n$ is defined by:

$$\mathbf{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix}$$

where:

- x_n is the rocket altitude at time n,
- $\dot{x}_n$ is the rocket velocity at time n.

We can express the state extrapolation equation as follows:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \end{bmatrix} + \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} (a_n + g)$$

2. The covariance extrapolation equation

$$\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T + \mathbf{Q}$$

The estimate covariance matrix is:

$$\mathbf{P} = \begin{bmatrix} p_x & p_{x\dot{x}} \\ p_{\dot{x}x} & p_{\dot{x}} \end{bmatrix}$$

The process noise matrix: Assuming a discrete noise model the process noise matrix for a constant acceleration model is:

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 \end{bmatrix}$$

The $\mathbf{Q}$ matrix for our example is:

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \epsilon^2$$

where:

- Δt is the time between successive measurements,
- ϵ^2 is the random variance in accelerometer measurement.

Example 10: Rocket Altitude Estimation

In Example 9, we used the system's random variance in acceleration σ_a^2 in $\mathbf{Q}$. But here, we have an accelerometer that measures the system's random acceleration. The accelerometer error v is much lower than the system's random acceleration; therefore, we use ϵ^2 as a multiplier in $\mathbf{Q}$, which makes our estimation uncertainty much lower!

The measurement equation: The measurement provides only the altitude of the rocket:

$$z_n = [x_{n,\text{measured}}]$$

The measurement model is given by:

$$z_n = \mathbf{H}\mathbf{x}_n$$

$$[x_{n,\text{measured}}] = \mathbf{H} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix}$$

The dimension of z_n is 1×1 and the dimension of $\mathbf{x}_n$ is 2×1, so the dimension of the observation matrix $\mathbf{H}$ is 1×2.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The measurement uncertainty

$$\mathbf{R}_n = \left[\sigma_{x_m}^2\right]$$

If we assume a constant measurement uncertainty:

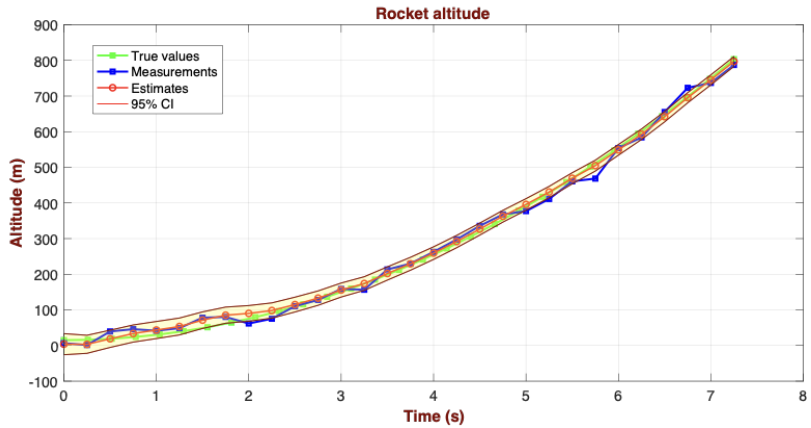
$$\mathbf{R}_1 = \mathbf{R}_2 = \ldots = \mathbf{R}_{n-1} = \mathbf{R}_n = \mathbf{R}$$

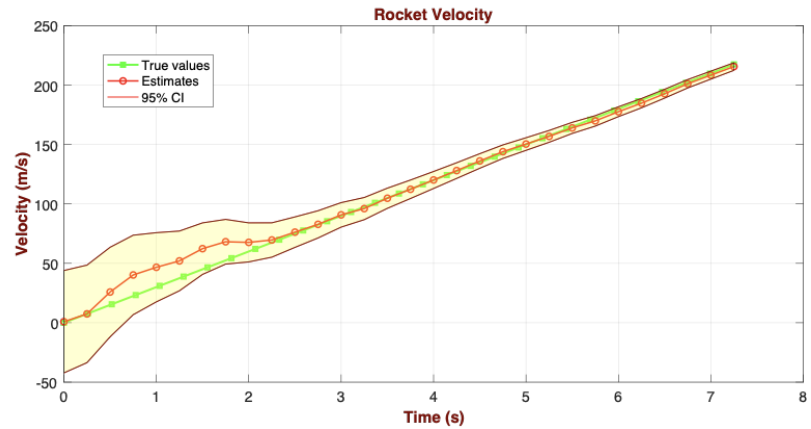
....The rest of the equations as before...

Example 10: Rocket Altitude Estimation

Let us assume a vertically boosting rocket with constant acceleration. The rocket is equipped with an altimeter that provides altitude measurements and an accelerometer that serves as a control input.

- The measurement period: $\Delta t = 0.25\,\text{s}$
- The rocket acceleration: $\ddot{x} = 30\,\text{m/s}^2$
- The altimeter measurement error standard deviation: $\sigma_{x_m} = 20\,\text{m}$
- The accelerometer measurement error standard deviation: $\epsilon = 0.1\,\text{m/s}^2$





[Code: Multivariate KF/ex10_MultivariateKF_RocketAltitudeEstimation.m]

Essential Background III—The square root of a matrix

The non-linear Kalman Filter requires prior knowledge of the Derivatives (EKF) and Matrix Square Root (UKF).

A matrix **B** is a square root of a matrix **A** if:

$$\mathbf{A} = \mathbf{BB}$$

The equation above can have several possible solutions, and there are different computation methods for finding the square root of a matrix.

The Unscented Kalman Filter employs an Unscented Transform that requires a square root computation of a covariance matrix.

As covariance matrices are positive and semi-definite, we can use **Cholesky decomposition (or Cholesky factorization)** for the computation of a covariance matrix square root.

Cholesky decomposition

The Cholesky decomposition is a decomposition of a positive definite matrix into a product of a lower triangular matrix and its transpose.
If a matrix **A** is positive definite:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

The lower triangular matrix is a square matrix where all the values above the diagonal are zero:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ 0 & l_{22} & l_{23} & l_{24} \\ 0 & 0 & l_{33} & l_{34} \\ 0 & 0 & 0 & l_{44} \end{bmatrix}$$

If matrix **A** is positive semi-definite, then the diagonal entries of **L** are allowed to be zero.
Cholesky decomposition algorithm:
- Diagonal elements of **L**:

$$l_{vv} = \sqrt{a_{vv} - \sum_{u<v} l_{vu}^2}$$

- Off-diagonal elements of **L**:

$$l_{tv} = \frac{1}{l_{vv}} \left(a_{tv} - \sum_{u<v} l_{tu} l_{vu} \right)$$

First, find the elements of the first row of **L**, then find the elements of the second row of **L**, and then find the elements of the third row of **L**. Continue the process until you reach the final row.
Implementation: **Python**: L = np.linalg.cholesky(A), **Matlab**: L = chol(A)

Non-linearity Problem

Necessary background on what are the non-linear systems, and why does the standard Linear Kalman Filter fail with non-linear systems?
We need to distinguish between two types of non-linearities:

- State-to-measurement non-linear relation
- Non-linear system dynamics

We must treat each type of non-linearity separately and then combine them.

Example – linear system

- Assume an air balloon that can move only upwards or downwards with constant acceleration dynamics. We are interested in estimating the balloon altitude.
- Since the balloon system dynamic model is linear, it can be described as:

$$\hat{\mathbf{x}}_{n+1,n} = F\hat{\mathbf{x}}_{n,n}$$

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\ddot{x}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\ddot{x}}_{n,n} \end{bmatrix}$$

- The balloon altitude is measured by the radar located beneath the balloon.
- The radar measurement error distribution is Gaussian.
- The radar sends an EM pulse; the pulse is reflected from the balloon and received by the radar. The radar measures the time elapsed between pulse transmission and reception. Since the pulse travels with the speed of light, we can calculate the target range (i.e., balloon altitude) as $x = t \times c/2$

Figure: Balloon altitude measurement using radar

- Let us construct the observation matrix $\mathbf{H}$.

$$z_n = \mathbf{H}x_n \quad \text{with} \quad \mathbf{H} = \begin{bmatrix} \frac{c}{2} \end{bmatrix}$$

- The dependency between the measured value (z_n–elapsed time) and the estimated value (x_n–balloon altitude) is linear.
- However, what happens to the estimation uncertainty? Is it still Gaussian?

Example – linear system

The following chart depicts the dependency between the elapsed time and the balloon altitude.

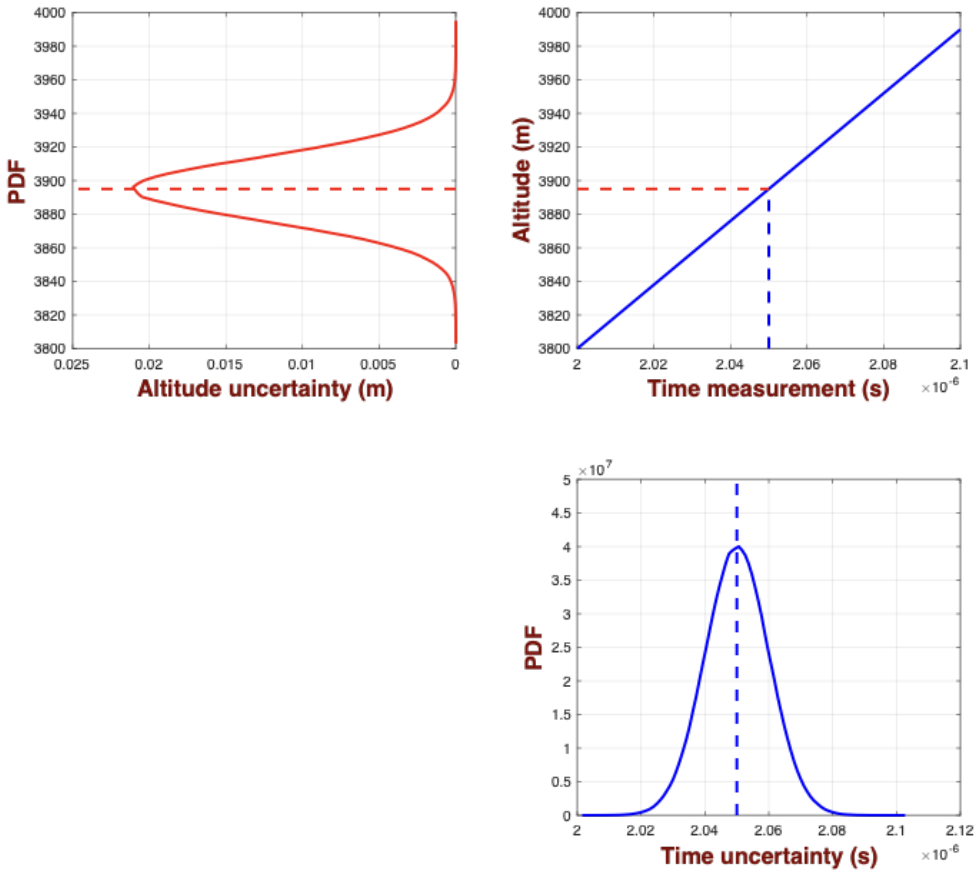


Figure: The dependency between the elapsed time and the balloon altitude: the distribution of measurement error (uncertainty) on the bottom plot and the state estimation error (uncertainty) on the left plot.

Since the dependency between the measured value and the estimated value is linear, the estimated value uncertainty is also Gaussian!

[Code: Non-linear KF/Ballon_LinearDynamics.m]

Example – State-to-measurement non-linear relation

This example presents the first type of non-linearity: state-to-measurement relation.

- Like the earlier example, we are interested in measuring the balloon altitude, while the balloon system dynamic model is linear.
- The balloon altitude is measured by the optical sensor that is located aside, using the target angle, with Gaussian measurement error distribution.
- The distance d between the sensor and the balloon nadir is known.

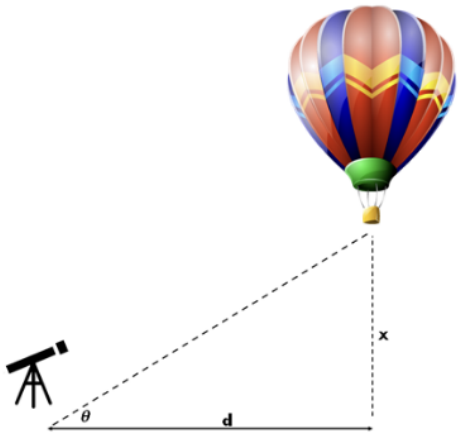


Figure: Balloon altitude measurement using an optical sensor.

- The balloon altitude can be calculated by using a trigonometric function:

$$x = d \cdot \tan(\theta)$$

  - x is the balloon altitude
  - d is the distance between the sensor and the balloon nadir
  - θ is the balloon's elevation angle
- As tangent function is non-linear, we can't construct the observation matrix $\mathbf{H}$!
- recall: for a linear system, the measurement equation is

$$\mathbf{z}_n = \mathbf{H}\mathbf{x}_n + \mathbf{v}_n$$

- For non-linear state-to-measurement relation system, the observation matrix $\mathbf{H}$ is a function of x, with measurement eq.:

$$\mathbf{z}_n = \mathbf{h}(\mathbf{x}_n, \mathbf{v}_n)$$

In this example, $\theta = \tan^{-1} \frac{x}{d}$

Example – State-to-measurement non-linear relation

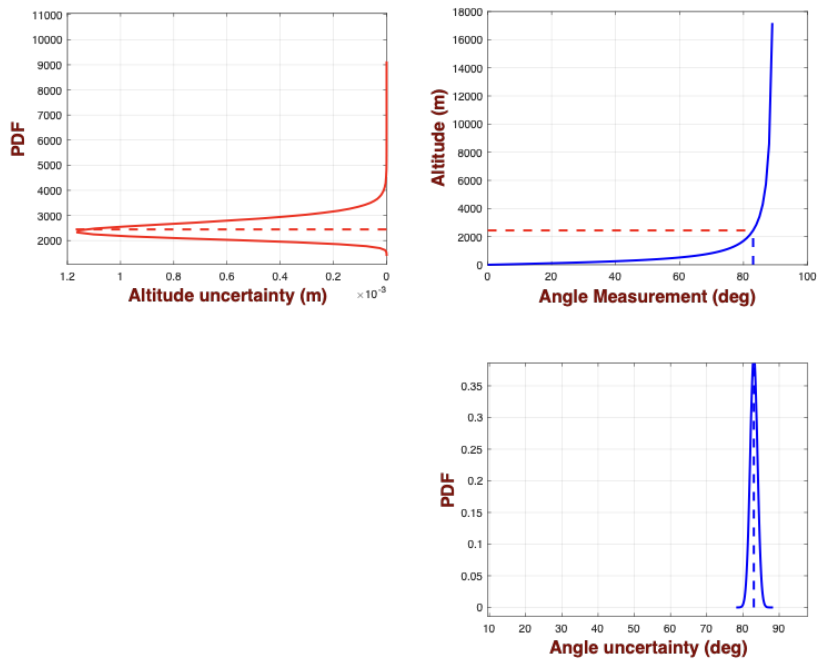


Figure: Non-linear System: The dependency between the measured angle and the balloon altitude, with the distribution of measurement error (uncertainty) on the bottom plot and the state estimation error (uncertainty) on the left plot.

The altitude uncertainty distribution is not Gaussian! It also changes its shape at different measurement points.
The Kalman Filter algorithm assumes that the distribution of all random variables is Gaussian. For non-linear systems, this assumption does not hold anymore. The algorithm is not stable and yields significant estimation errors.

[Code: Non-linear KF/Ballon_State2MeasurementUncertainty.m]

Example – Non-linear system dynamics

This example presents the second type of non-linearity: non-linear system dynamics.

- Assume an ideal gravity pendulum that consists of a body with mass m hung by a string with length L from fixed support - the pendulum swings back and forth at a constant amplitude. We want to estimate the angle θ (radians) from the vertical to the pendulum.

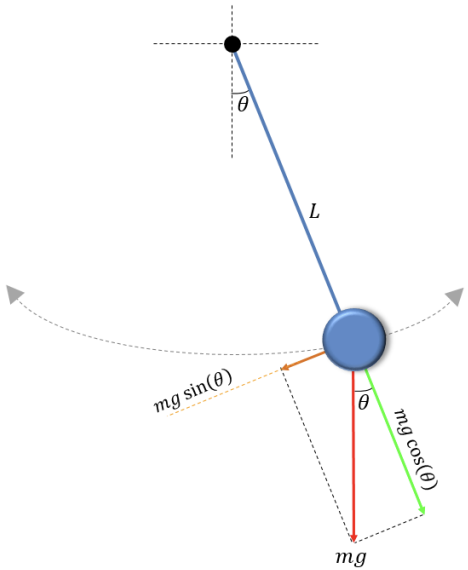


Figure: Pendulum

- According to Newton's second law, the sum of forces on the object equals $F = ma$.
- The force applied to the pendulum equals

$$F = -mg\sin(\theta) = ma \Rightarrow a = -g\sin(\theta)$$

- The arc length s that corresponds to angle θ is

$$s = L\theta$$

- The pendulum velocity equals

$$v = \frac{ds}{dt} = L\frac{d\theta}{dt}$$

- The pendulum acceleration equals

$$a = \frac{d^2s}{dt^2} = L\frac{d^2\theta}{dt^2} = -g\sin\theta$$

- Thus, the differential equation that describes the pendulum movement is a second-order homogeneous differential equation:

$$L\frac{d^2\theta}{dt^2} = -g\sin\theta$$

Example – Non-linear system dynamics

Dynamic model of the system: The state vector of the pendulum is in the form of the following:

$$\mathbf{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix}$$

- θ_n is the pendulum angle at time n
- $\dot{\theta}_n$ is the pendulum angular velocity at time n

$$\hat{\theta}_{n+1,n} = \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t$$

$$\hat{\dot{\theta}}_{n+1,n} = \hat{\dot{\theta}}_{n,n} + \hat{\ddot{\theta}}_{n,n}\Delta t = \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t$$

The dynamic model is not linear. For a linear system, the general form of the state extrapolation equation in a matrix notation is:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}u_n + \mathbf{w}_n$$

For the non-linear system dynamics, the state transition matrix $\mathbf{F}$ is a function of $\mathbf{x}$ and $\mathbf{u}$. Therefore, the state extrapolation equation looks as:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{f}(\hat{\mathbf{x}}_{n,n}, \mathbf{u}_n, \mathbf{w}_n)$$

In this example:

$$f(\hat{\mathbf{x}}_{n+1,n}) = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix}$$

The measurement equation depends on the measured parameter. Let's review two cases:
Case 1 - Measured parameter is pendulum angle θ: In this case, the measurement equation looks like:

$$\theta_{n,\text{measured}} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix}$$

The above equation is in the form of: $\mathbf{z}_n = \mathbf{H}\mathbf{x}_n$. Therefore, the state-to-measurement relation (the first type of non-linearity) is linear.
Case 2 - Measured parameter is the pendulum x-position: In this case, the measurement equation is:

$$x_{n,\text{measured}} = L\sin(\theta_n)$$

which is in the form of: $\mathbf{z}_n = \mathbf{h}(\mathbf{x}_n)$. Therefore, the state-to-measurement relation is non-linear.

Kalman Filter Extensions (Non-linear Kalman Filters)

Objectives: Since most real-life systems are non-linear, non-linear Kalman Filter design is essential
Approach:

- For non-linear systems handling, the *Linear Approximation* techniques are applied
- Mathematical derivation and examples of the most common non-linear filters, designed based on *Linear Approximation*:
  - Extended Kalman Filter (EKF): The EKF performs **analytic linearization** of the model at each point in time. EKF is the most common non-linear Kalman Filter.
  - Unscented Kalman Filter (UKF) or Sigma-point Kalman Filter (SPKF): The UKF performs **statistical linearization** of the model at each point in time.

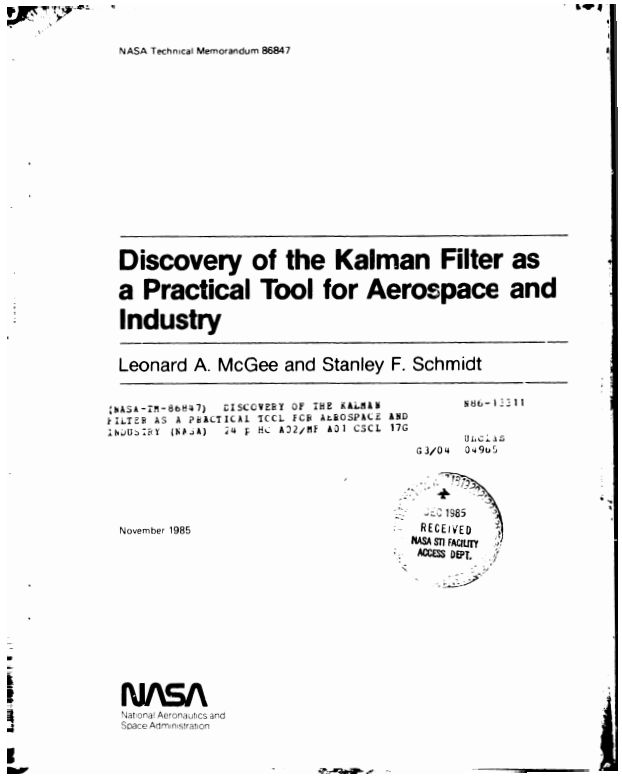
Facts:

- While the standard Linear Kalman Filter (LKF) is an *optimal filter* since we minimize the estimate uncertainty, all Kalman Filter modifications for non-linear systems are *sub-optimal* since we use approximated models.

An Early Implementation

*In fact, Kalman's paper had not attracted much attention until Stanley Schmidt at the NASA Ames Research Center in Mountain View, California, recognized that it could be applied to the satellite and rocket-tracking problems just gaining prominence because of President Kennedy's challenge to land a man on the moon. Schmidt made two important contributions to the success of Kalman's ideas:

1. He broke up the equations into the two steps of time updates and measurement updates, important if the data did not come in a steady stream, and
2. Because the dynamics were nonlinear, he applied Kalman's equations to linearized models, leading to the first of a family of what are known now as extended Kalman filters (EKF).


NASA Technical Memorandum 86847

Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry

Leonard A. McGee and Stanley F. Schmidt

November 1985

NASA
National Aeronautics and Space Administration

* [Encounters and Interactions with Rudolf E. Kalman, Thomas Kailith]

Kalman Filter Extensions

- Kalman–Bucy Filter (Continuous time)
- **Extended Kalman Filter (EKF)**
- Distributed Kalman Filter
- **Unscented Kalman Filter**
- Discriminative Kalman Filter
- Adaptive Kalman Filter
- Hybrid Kalman Filter
- ...

Extended Kalman Filter (EKF)—Analytic linearization

The main idea behind the EKF is the linearization of the dynamic model at the working point.

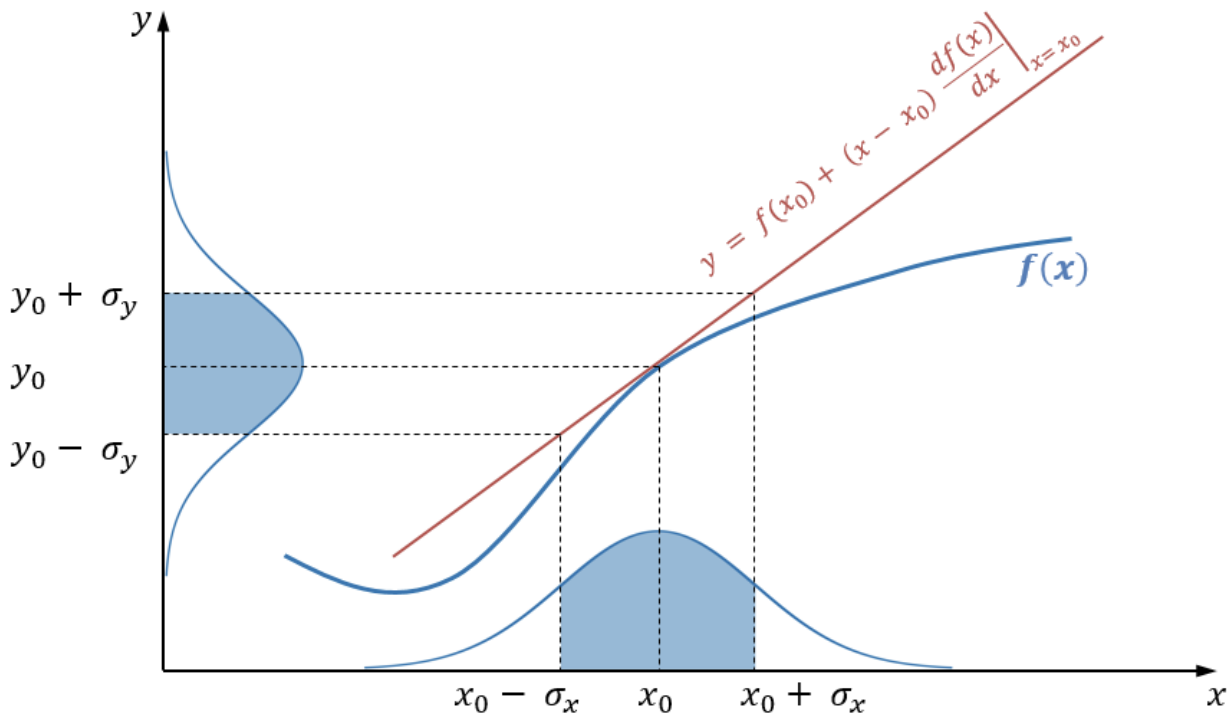


Figure: One-dimensional case for analytic linearization of the model at each point in time

Using the tangent line at a point $x = x_0$, we can project the uncertainty to the y-axis and keep its' shape Gaussian.

Analytic linearization

- The slope (m) of the tangent line to the function $f(x)$ at the point x_0 equals to the derivative of the function $f(x)$ at the point x_0:

$$m = \left.\frac{df(x)}{dx}\right|_{x=x_0}$$

- The general straight-line equation is:

$$y = mx + b$$

- We can find the line equation given the slope m and any point (x_0, y_0):

$$y - y_0 = m(x - x_0)$$

- Expand and rearrange:

$$y = mx \underbrace{-mx_0 + y_0}_{b} \quad (1)$$

- We can find y_0 using the original function $f(x)$:

$$y_0 = f(x_0)$$

- The tangent line equation, using (1), is

$$\begin{aligned} y &= mx - mx_0 + y_0 \\ &= y_0 + (x - x_0)m \\ &= f(x_0) + (x - x_0) \left.\frac{df(x)}{dx}\right|_{x=x_0} \\ &= f(x_0) + f(x)'(x - x_0) \end{aligned}$$

First-order Taylor series expansion

The process of analytic linearization is also called: "The approximation of $f(x)$ by a first-order Taylor series expansion about the point $x = x_0$."

- According to Taylor's theorem, the function $f(x)$ equals an infinite sum of terms that are expressed in terms of the function derivatives at a single point $x = x_0$:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \cdots + \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \cdots$$

- We can approximate the function $f(x)$ by calculating the first k terms of the Taylor Series. For high approximation precision, we shall select a high k value.
- For the linear approximation, we should keep only the first two terms of the Taylor Series:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

Uncertainty Projection in One Dimension

- The EKF projects the uncertainty using the linearization technique.
- When projecting the uncertainty, there is no need to evaluate the observation function $h(x)$ and the state transition function $f(x)$. Intead, we only need to evaluate derivatives $\frac{dh(x)}{dx}$ and $\frac{df(x)}{dx}$.
- After finding the tangent line equation at the point x_0, we can project the uncertainty using the tangent line.
- The line equation is:

$$y = mx + b$$

$$m = \left.\frac{df(x)}{dx}\right|_{x=x_0}$$

$$b = -mx_0 + y_0$$

- The standard deviation (sigma) points are: $(x_0 + \sigma_x, x_0 - \sigma_x)$.

$$y - \sigma_y = m(x_0 - \sigma_x) + b$$

$$y + \sigma_y = m(x_0 + \sigma_x) + b$$

- The measurement uncertainty is the difference between the sigma points:

$$(y + \sigma_y) - (y - \sigma_y) =$$
$$m(x_0 + \sigma_x) + b - (m(x_0 - \sigma_x) + b)$$

$$\Rightarrow \sigma_y = m\sigma_x = \left|\frac{df(x)}{dx}\right| \sigma_x$$

The estimation variance: $p_x = \sigma_x^2$

$$p_y = \left|\frac{df(x)}{dx}\right|^2 p_x$$

Example – linearization in a single dimension

- Recall balloon altitude measurement example with a state-to-measurement non-linear relation.
- The balloon altitude is measured by the optical sensor that is located aside. The distance d between the sensor and the nadir of the balloon is known. The optical sensor can measure the target angle θ.
- The measurement equation has the following form $z_n = h(x_n)$, i.e.,

$$z_n = \theta = \tan^{-1} \frac{x_n}{d}$$

$$h(x_n) = \theta = \tan^{-1} \frac{x_n}{d}$$

- To propagate the measurement uncertainty from the angle domain to the altitude domain using analytic linearization, i.e., differentiate $h(x_n)$.

$$\frac{dh(x)}{dx} = \frac{d}{dx}\left(\tan^{-1} \frac{x}{d}\right)$$

- The derivative of the arctan is given by:

$$\frac{d}{dx} \tan^{-1}(x) = \frac{1}{1+x^2}$$

$$\Rightarrow \frac{dh(x)}{dx} = \left[\frac{d}{d^2+x^2}\right]$$

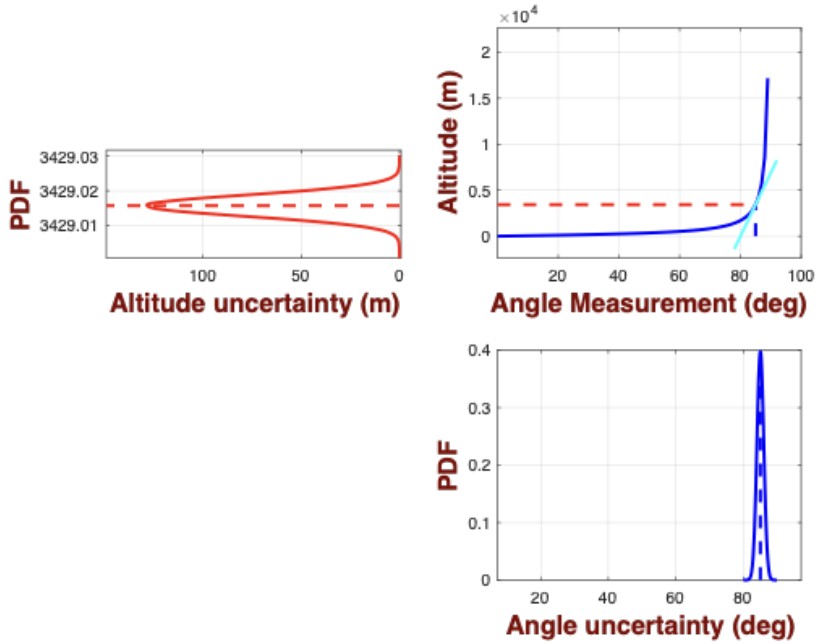


Figure: Linearization example with tangent line to the function $x = d \cdot tan(\theta)$ at the point $\theta = 85^o$.

[Code: Non-linear KF/Balloon_State2MeasurementUncertainty_Linearization.m]

Uncertainty projection in two dimensions

- In two dimensions, the uncertainty is projected through a tangent plane.

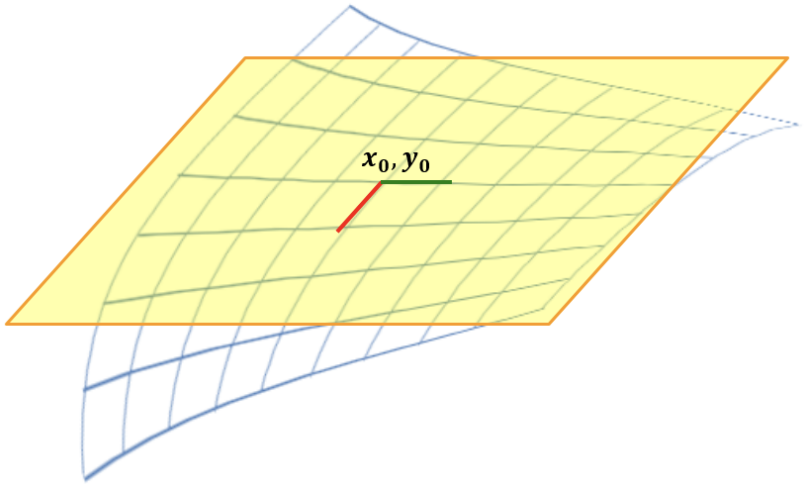


Figure: Tangent plane: An example of a tangent plane of the non-linear function $f(x, y)$ at a point $x0, y0$.

- The tangent plane is characterized by two orthogonal slopes – the x-axis slope and the y-axis slope. The partial derivatives of $f(x, y)$ at a point (x_0, y_0) are the slopes of the tangent plane:

$$\frac{\partial f(x,y)}{\partial x}, \quad \frac{\partial f(x,y)}{\partial y}$$

- In two dimensions also, we need to find two partial derivatives for projecting the uncertainty.
- For the pendulum example, the state vector of the pendulum is

$$\mathbf{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix}$$

  - θ_n is the pendulum angle at time n
  - $\dot{\theta}_n$ is the pendulum angular velocity at time n
- We measure the pendulum position: $L \cdot \sin(\theta_n)$.
- Since the **state-to-measurement relation is non-linear**, the measurement equation is a type of:

$$\mathbf{z}_n = \mathbf{h}(\mathbf{x}_n)$$

$$\mathbf{h}(\mathbf{x}_n) = L \cdot \sin(\theta_n)$$

- The multivariate analytical linearization is:

$$\mathbf{P}_{\text{out}} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \mathbf{P}_{\text{in}} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right)^T$$

Multivariate uncertainty projection

- We must find the partial derivatives of $h(\mathbf{x})$:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial (L\sin(\theta_n))}{\partial \theta} & \frac{\partial (L\sin(\theta_n))}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} L\cos(\theta_n) & 0 \end{bmatrix}$$

$$\mathbf{P}_{\text{out}} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \mathbf{P}_{\text{in}} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right)^T = (L\cos(\theta_n))^2 \mathbf{P}_{\text{in}}$$

The dynamic model of the pendulum is also non-linear(the second type of non-linearity). It has the form of:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{f}(\hat{\mathbf{x}}_{n,n})$$

$$\hat{\mathbf{x}}_{n+1,n} = \begin{bmatrix} \hat{\theta}_{n+1,n} \\ \hat{\dot{\theta}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix}$$

$$\mathbf{f}(\hat{\mathbf{x}}_{n,n}) = \begin{bmatrix} f_1(\hat{\mathbf{x}}_{n,n}) \\ f_2(\hat{\mathbf{x}}_{n,n}) \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix}$$

- The dynamic model function $f(\hat{\mathbf{x}}_{n,n})$ is a matrix that contains two different sub-functions. We should find partial derivatives for each sub-function.

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial \hat{\theta}} & \frac{\partial f_1}{\partial \hat{\dot{\theta}}} \\ \frac{\partial f_2}{\partial \hat{\theta}} & \frac{\partial f_2}{\partial \hat{\dot{\theta}}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial (\hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t)}{\partial \hat{\theta}} & \frac{\partial (\hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t)}{\partial \hat{\dot{\theta}}} \\ \frac{\partial (\hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t)}{\partial \hat{\theta}} & \frac{\partial (\hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t)}{\partial \hat{\dot{\theta}}} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L}\cos(\hat{\theta}_{n,n})\Delta t & 1 \end{bmatrix}$$

The multivariate analytical linearization is given by:

$$\mathbf{P}_{\text{out}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{P}_{\text{in}} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^T$$

Multivariate uncertainty projection

- We can generalize the two-dimensional case by extending it to an N - dimensional case.
- For multi-dimensional problems, we propagate the multivariate Gaussian random variable (represented by covariance matrix) using a linear approximation of the multi-dimensional function.
- For the non-linear system dynamics, the multivariate analytical linearization is given by:

$$\mathbf{P}_{\text{out}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{P}_{\text{in}} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^T$$

  where $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is state transition matrix Jacobian.
- Similarly, for the state-to-measurement non-linear relation, the multivariate analytical linearization is given by:

$$\mathbf{P}_{\text{out}} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \mathbf{P}_{\text{in}} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right)^T$$

  where $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ an observation matrix Jacobian.

For additional material on error propagation refer to [2].

Jacobian derivation example

Let's refer back to "Vehicle location estimation example" Slide 1, where we estimated the vehicle location in the XY plane using onboard location sensor, reported x and y coordinates.

Now, we want to track the vehicle using radar, located at the plane origin, measuring the vehicle range (r) and bearing angle (ϕ). The radar measurement error distribution is Gaussian.

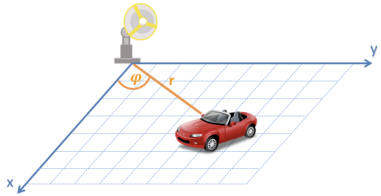


Figure: Vehicle location estimation using radar.

- The measurement vector $\mathbf{z}_n$ and the state vector $\mathbf{x}_n$ is:

$$\mathbf{z}_n = \begin{bmatrix} r_n \\ \phi_n \end{bmatrix}, \quad \mathbf{x}_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

- Let us find the relation between the measurement vector and the state vector. The vehicle range (r) can be expressed by x and y as:

$$r = \sqrt{x^2 + y^2}$$

- The vehicle bearing angle (ϕ) can be expressed by x and y as:

$$\phi = \tan^{-1}\left(\frac{y}{x}\right)$$

- Since the state-to-measurement relation is non-linear, the measurement equation is a type of $\mathbf{z}_n = \mathbf{h}(x_n)$:

$$\begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}\left(\frac{y}{x}\right) \end{bmatrix}$$

Jacobian derivation:

$$\frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial \sqrt{x^2+y^2}}{\partial x} & \frac{\partial \sqrt{x^2+y^2}}{\partial y} \\ \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial x} & \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial y} \end{bmatrix}$$

$$\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ -\frac{y}{x^2+y^2} & \frac{x}{x^2+y^2} \end{bmatrix}$$

EKF Equations—The EKF observation matrix - **H**

EK requires modifications related to the observation matrix **H** and the state transition matrix **F**. If the state-to-measurement relation (the first type of non-linearity) of the system is non-linear, the observation matrix is of the type

$$\mathbf{H} = \mathbf{h}(\mathbf{x}_n)$$

The State Update Equation looks like the following:

$$\hat{\mathbf{x}}_{n,n} = \hat{\mathbf{x}}_{n,n-1} + \mathbf{K}_n(\mathbf{z}_n - \mathbf{h}(\hat{\mathbf{x}}_{n,n-1}))$$

For the uncertainty propagation, the observation matrix **H** should be linearized to keep the uncertainty PDF Gaussian, Therefore, the **Covariance Update Equation** looks like the following:

$$\mathbf{P}_{n,n} = \left(\mathbf{I} - \mathbf{K}_n \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right) \mathbf{P}_{n,n-1} \left(\mathbf{I} - \mathbf{K}_n \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right)^T + \mathbf{K}_n \mathbf{R}_n \mathbf{K}_n^T$$

The Kalman Gain Equation looks like the following:

$$\mathbf{K}_n = \mathbf{P}_{n,n-1} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right)^T \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \mathbf{P}_{n,n-1} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right)^T + \mathbf{R}_n\right)^{-1}$$

EKF Equations—The EKF state transition matrix **F**

If the dynamic model (the second type of non-linearity) of the system is non-linear, the observation matrix is of the type:

$$\mathbf{F} = \mathbf{f}(\mathbf{x}_n)$$

For the uncertainty propagation, the state transition matrix **F** should be linearized to keep the uncertainty PDF Gaussian. The State Extrapolation Equation looks like the following:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{f}(\hat{\mathbf{x}}_{n,n}) + \mathbf{G}\mathbf{u}_n$$

The Covariance Extrapolation Equation looks like the following:

$$\mathbf{P}_{n+1,n} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{P}_{n,n} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^T + \mathbf{Q}$$

Summary of EKF Equations

	Equation	LKF Equation	EKF Equation
Predict	State Extrapolation	$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$	$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) + \boldsymbol{G}\boldsymbol{u}_n$
	Covariance Extrapolation	$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$	$\boldsymbol{P}_{n+1,n} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{P}_{n,n}\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{Q}$
Update	State Update	$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}\right)$	$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1})\right)$
	Covariance Update	$\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)\boldsymbol{P}_{n,n-1} \times \left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$	$\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)\boldsymbol{P}_{n,n-1} \times \left(\boldsymbol{I} - \boldsymbol{K}_n\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$
	Kalman Gain	$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T \times \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$	$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T \times \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\boldsymbol{P}_{n,n-1}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{R}_n\right)^{-1}$

Example 11 – vehicle location estimation using radar

- We want to track the vehicle using radar, which is located at the plane origin, measuring the vehicle range (r) and the bearing angle (φ).
- The radar measurement error distribution is Gaussian.

Let's look at KF equations.

The state extrapolation equation

- The state extrapolation equation:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n}$$

The system state $\mathbf{x}_n$ is defined by:

$$\mathbf{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix}$$

- Assuming constant acceleration dynamics, the extrapolated vehicle state for time $n+1$ can be described as:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\ddot{x}}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{\dot{y}}_{n+1,n} \\ \hat{\ddot{y}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\ddot{x}}_{n,n} \\ \hat{y}_{n,n} \\ \hat{\dot{y}}_{n,n} \\ \hat{\ddot{y}}_{n,n} \end{bmatrix}$$

- The dynamic model of the system (the second type of non-linearity) in this example is linear! There is no need to calculate the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$.

The covariance extrapolation equation is similar to example 9 1:

$$\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T + \mathbf{Q}$$

The estimate covariance is:

$$\mathbf{P} = \begin{bmatrix} p_x & p_{x\dot{x}} & p_{x\ddot{x}} & 0 & 0 & 0 \\ p_{\dot{x}x} & p_{\dot{x}} & p_{\dot{x}\ddot{x}} & 0 & 0 & 0 \\ p_{\ddot{x}x} & p_{\ddot{x}\dot{x}} & p_{\ddot{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_y & p_{y\dot{y}} & p_{y\ddot{y}} \\ 0 & 0 & 0 & p_{\dot{y}y} & p_{\dot{y}} & p_{\dot{y}\ddot{y}} \\ 0 & 0 & 0 & p_{\ddot{y}y} & p_{\ddot{y}\dot{y}} & p_{\ddot{y}} \end{bmatrix}$$

Example 11 – vehicle location estimation using radar

The process noise matrix is also similar to example 9:

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 & \sigma_{x\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\ddot{x}x}^2 & \sigma_{\ddot{x}\dot{x}}^2 & \sigma_{\ddot{x}}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_y^2 & \sigma_{y\dot{y}}^2 & \sigma_{y\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\dot{y}y}^2 & \sigma_{\dot{y}}^2 & \sigma_{\dot{y}\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\ddot{y}y}^2 & \sigma_{\ddot{y}\dot{y}}^2 & \sigma_{\ddot{y}}^2 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix}$$

The **measurement equation** is different from example 9. The measurement vector $\mathbf{z}_n$ is:

$$\mathbf{z}_n = \begin{bmatrix} r_n \\ \phi_n \end{bmatrix}$$

The system state vector $\mathbf{x}_n$ is defined by:

$$\mathbf{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix}$$

Let us find the relation between the measurement vector and the state vector.

$$r = \sqrt{x^2 + y^2}$$

$$\phi = \tan^{-1}\left(\frac{y}{x}\right)$$

Since the state-to-measurement relation (the first type of non-linearity) is non-linear, the measurement equation is a type of $\mathbf{z}_n = \mathbf{h}(\mathbf{x}_n)$:

$$\begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}\left(\frac{y}{x}\right) \end{bmatrix}$$

Example 11 – vehicle location estimation using radar

Jacobian derivation:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \sqrt{x^2+y^2}}{\partial x} & \frac{\partial \sqrt{x^2+y^2}}{\partial \dot{x}} & \frac{\partial \sqrt{x^2+y^2}}{\partial \ddot{x}} & \frac{\partial \sqrt{x^2+y^2}}{\partial y} & \frac{\partial \sqrt{x^2+y^2}}{\partial \dot{y}} & \frac{\partial \sqrt{x^2+y^2}}{\partial \ddot{y}} \\ \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial x} & \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial \dot{x}} & \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial \ddot{x}} & \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial y} & \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial \dot{y}} & \frac{\partial \tan^{-1}\left(\frac{y}{x}\right)}{\partial \ddot{y}} \end{bmatrix}$$

$$= \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & 0 & 0 & \frac{y}{\sqrt{x^2+y^2}} & 0 & 0 \\ -\frac{y}{x^2+y^2} & 0 & 0 & \frac{x}{x^2+y^2} & 0 & 0 \end{bmatrix}$$

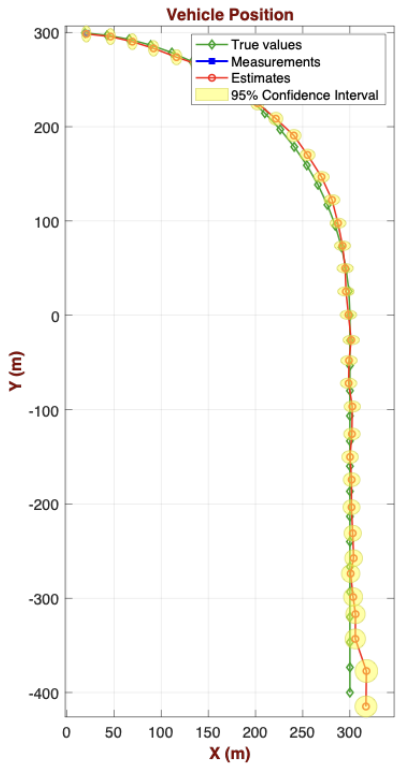
The **measurement uncertainty** is represented by the measurement covariance matrix:

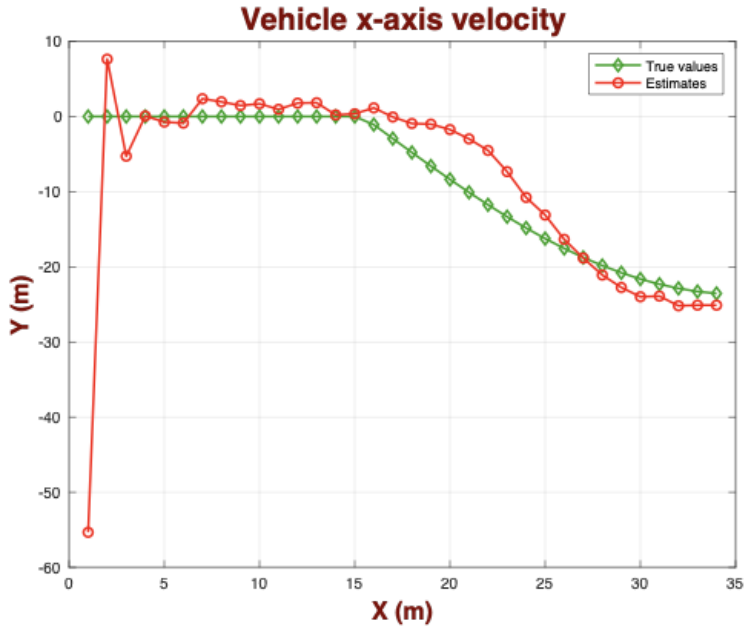
$$\mathbf{R}_n = \begin{bmatrix} \sigma^2_{r_m} & 0 \\ 0 & \sigma^2_{\phi_m} \end{bmatrix}$$

The **Kalman Gain**, **The state update equation**, and **The covariance update equation** relations are the same as expressed before.

Example 11 – vehicle location estimation using radar: Numerical example

- The measurements period: $\Delta t = 1\text{s}$
- The random acceleration standard deviation: $\sigma_a = 0.2\text{m/s}^2$
- The range measurement error standard deviation: $\sigma_{r_m} = 5\text{m}$
- The bearing angle measurement error standard deviation: $\sigma_{\phi_m} = 0.0087\text{rad}$





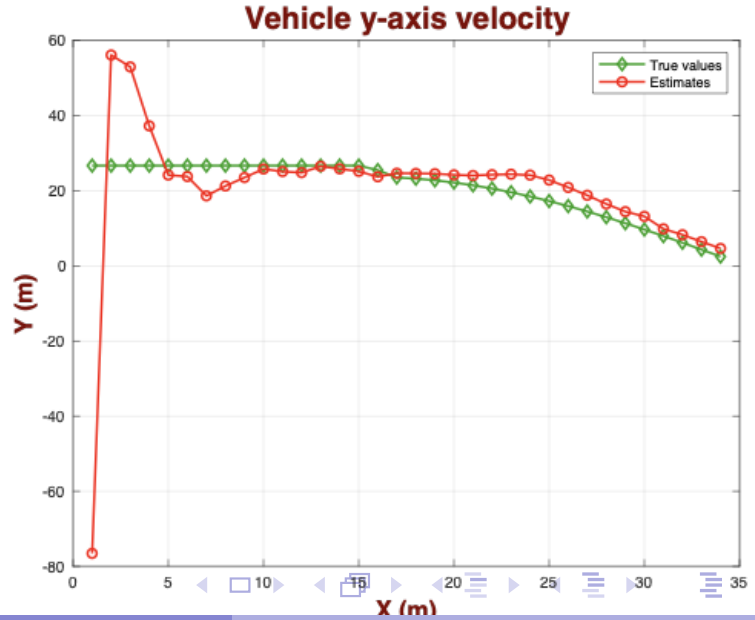
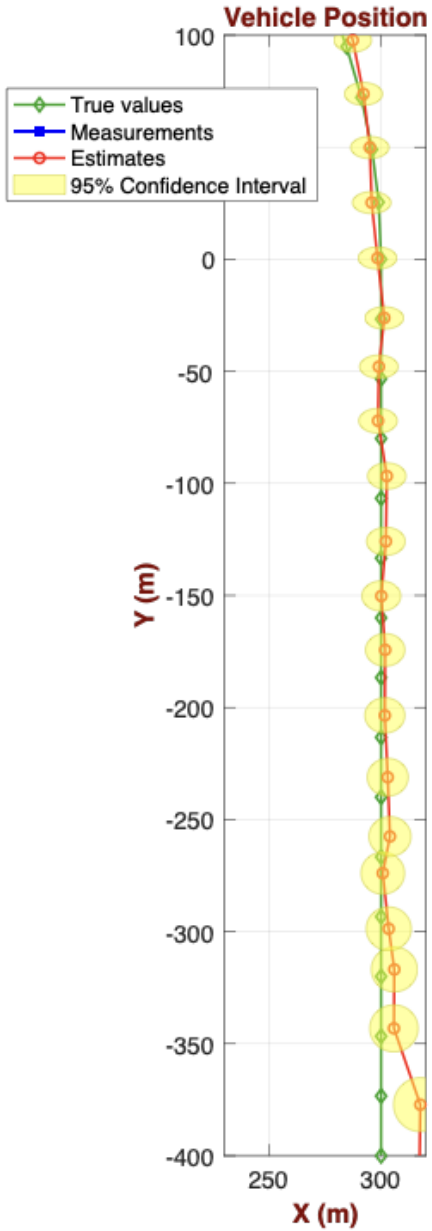
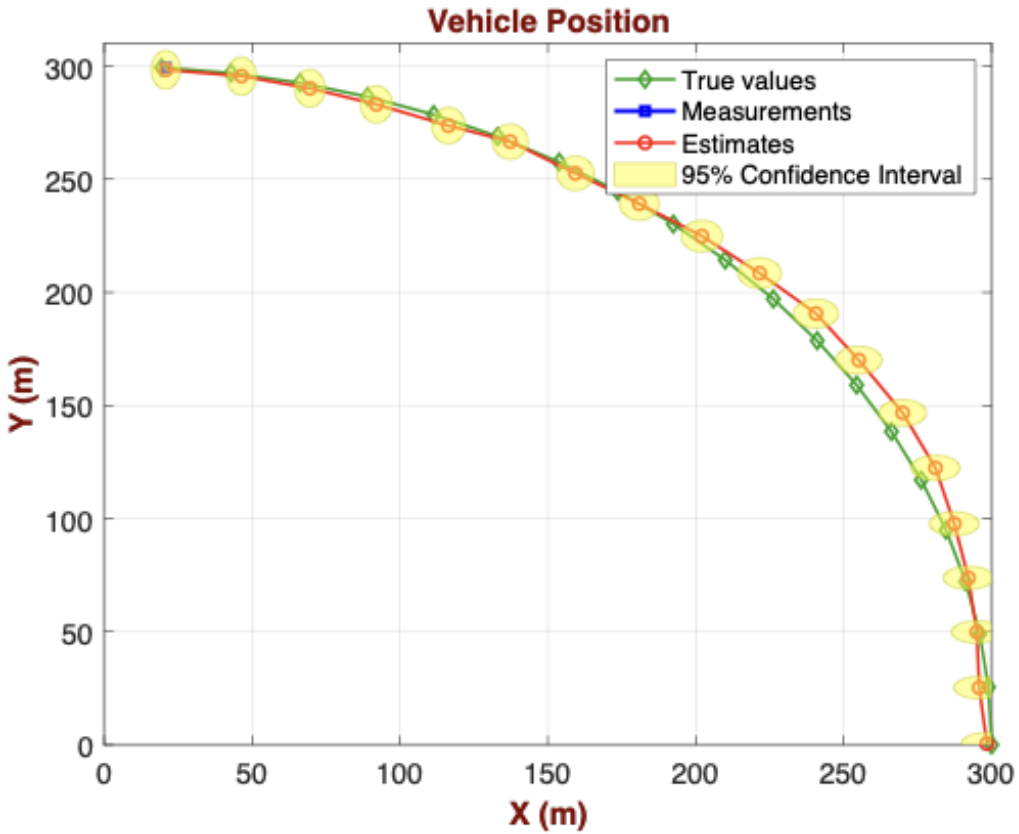


Figure: True, measured, and estimated vehicle position compared to the 95% confidence ellipses.

Example 11 – vehicle location estimation using radar: Numerical example

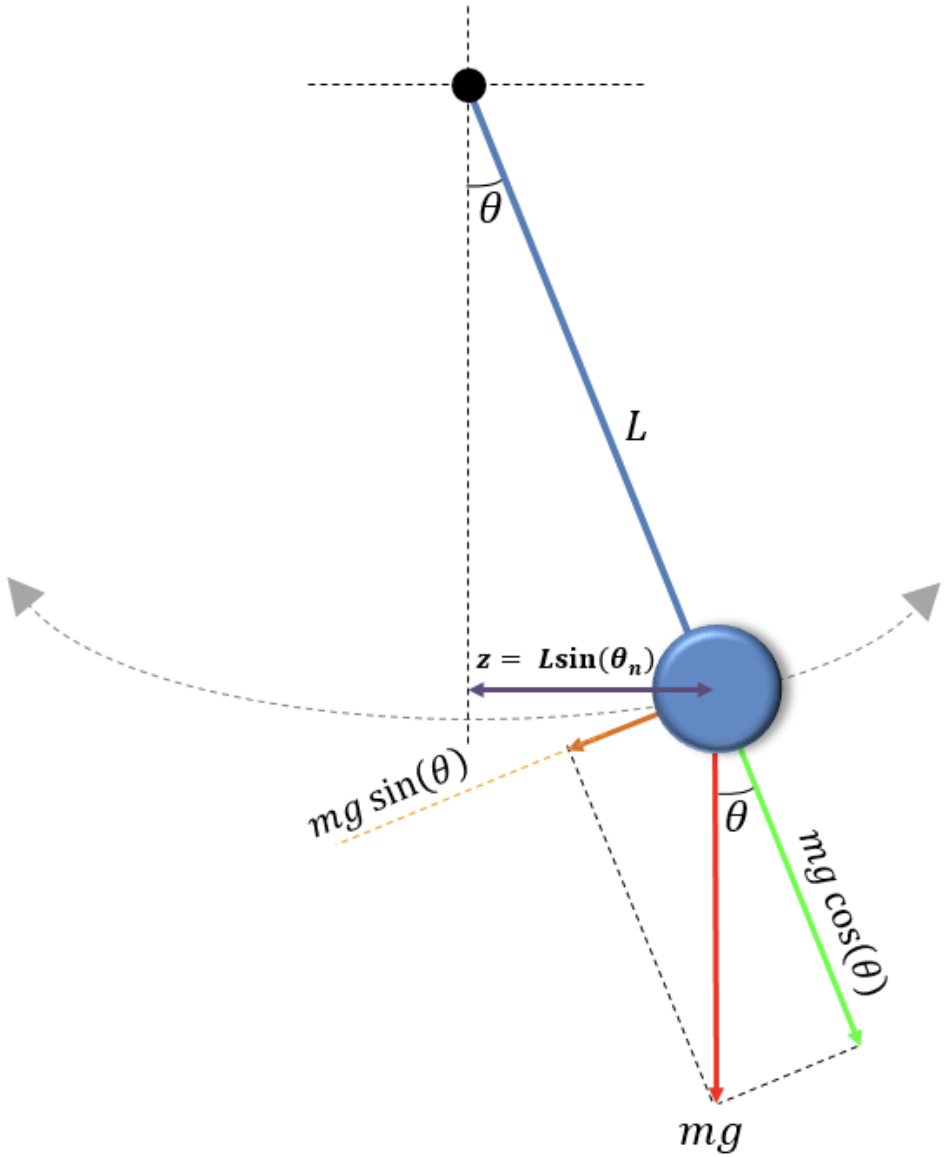




[Code: Non-linear KF/Ex11_EKF_State2MeasurementUncertainty.m]

Example 12 - estimating the pendulum angle

- In this example, we want to estimate the pendulum angle θ from the measured pendulum position.
- The dynamic model of the the pendulum was derived earlier (slide XYZ).
- Assume an ideal gravity pendulum that consists of a body with mass m hung by a string with length L from fixed support - the pendulum swings back and forth at a constant amplitude.
- We want to estimate the angle θ from the vertical to the pendulum. The angle units are radians.
- We measure the pendulum position $z = Lsin(\theta_n)$.



Example 12 - estimating the pendulum angle: Kalman Filter Equations

The State Extrapolation Equation: The state vector of the pendulum is:

$$\mathbf{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix}$$

where θ_n is the pendulum's angle $\dot{\theta}_n$ the angular velocity at time n
The dynamic model of the pendulum is non-linear:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{f}(\hat{\mathbf{x}}_{n,n})$$

$$\hat{\mathbf{x}}_{n+1,n} = \begin{bmatrix} \hat{\theta}_{n+1,n} \\ \hat{\dot{\theta}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix}$$

$$\mathbf{f}(\hat{\mathbf{x}}_{n,n}) = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix}$$

Jacobian Derivation: The Jacobian for the pendulum dynamic model:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial \hat{\theta}} & \frac{\partial f_1}{\partial \hat{\dot{\theta}}} \\ \frac{\partial f_2}{\partial \hat{\theta}} & \frac{\partial f_2}{\partial \hat{\dot{\theta}}} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L}\cos(\hat{\theta}_{n,n})\Delta t & 1 \end{bmatrix}$$

The estimate covariance is:

$$\mathbf{P} = \begin{bmatrix} p_x & p_{x\dot{x}} \\ p_{\dot{x}x} & p_{\dot{x}} \end{bmatrix}$$

The Process Noise Matrix

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2$$

The Measurement Equation We measure the pendulum position: $L\sin(\theta_n)$, i.e., the state-to-measurement relation is non-linear:

$$\mathbf{z}_n = \mathbf{h}(\mathbf{x}_n), \quad \mathbf{h}(\mathbf{x}_n) = L\sin(\theta_n)$$

Jacobian Derivation of position measurement:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial(L\sin(\theta_n))}{\partial \theta} & \frac{\partial(L\sin(\theta_n))}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} L\cos(\theta_n) & 0 \end{bmatrix}$$

The Measurement Uncertainty is:

$$\mathbf{R}_n = \begin{bmatrix} \sigma_{x_m}^2 \end{bmatrix}$$

Example 12 - Estimating the pendulum angle: Pendulum Motion Simulation

Example Parameters

- The Pendulum string length: $L = 0.5\,\text{m}$
- Gravitational acceleration constant: $g = 9.8\,\text{m/s}^2$
- Measurement Uncertainty (standard deviation): $\sigma_{x_m} = 0.01\,\text{m}$
- Process Noise Uncertainty (angular acceleration standard deviation): $\sigma_a = 1\,\text{rad/s}^2$

Let's first do the maths for pendulum motion simulation for establish ground truth.

- The differential equation that describes the pendulum movement:

$$L\frac{d^2\theta}{dt^2} = -g\sin\theta, \quad \text{or} \quad \ddot{\theta} + \frac{g}{L}\sin\theta = 0$$

- **Approximation:** if θ is small, then $\sin\theta \approx \theta$,

$$\ddot{\theta} + \frac{g}{L}\theta = 0 \qquad \text{(E1)}$$

- Using "the method of inspired guessing"– the sine and cosine functions are periodic–we can have a solution for the angle θ as a function of time t:

$$\theta(t) = A\cos(\omega t) + B\sin(\omega t) \qquad \text{(E2)}$$

- At $t = 0$, from E2:

$$\theta(t = 0) = A$$

So, A is an initial angle, denoted by θ_0.

- Re-write E2:

$$\theta(t) = \theta_0\cos(\omega t) + B\sin(\omega t)$$

- The derivative of $\theta(t)$ is the angular velocity (ω) of the pendulum:

$$\omega(t) = \frac{d\theta}{dt} = \dot{\theta} = -\omega\theta_0\sin(\omega t) + \omega B\cos(\omega t)$$

- To find B, evaluate $\omega(t)$ at $t = 0$:

$$\omega(t = 0) = \omega B, \Rightarrow B = \frac{\omega_0}{\omega}$$

Example 12 - Estimating the Pendulum Angle: Pendulum Motion Simulation

- Thus, our solution now has the following form:

$$\theta(t) = \theta_0 \cos(\omega t) + \frac{\omega_0}{\omega} \sin(\omega t) \qquad \text{(E3)}$$

- From E3:

$$\dot{\theta} = -\omega\theta_0 \sin(\omega t) + \omega_0 \cos(\omega t)$$
$$\ddot{\theta} = -\omega^2\theta_0 \cos(\omega t) - \omega\omega_0 \sin(\omega t)$$
$$\ddot{\theta} = -\omega^2 \left(\theta_0 \cos(\omega t) + \frac{\omega_0}{\omega} \sin(\omega t)\right)$$
$$\ddot{\theta} = -\omega^2\theta \qquad \text{(E4)}$$

- Comparing E4 with E1:

$$\omega^2 = \frac{g}{L}$$

$$\omega = \sqrt{\frac{g}{L}}$$

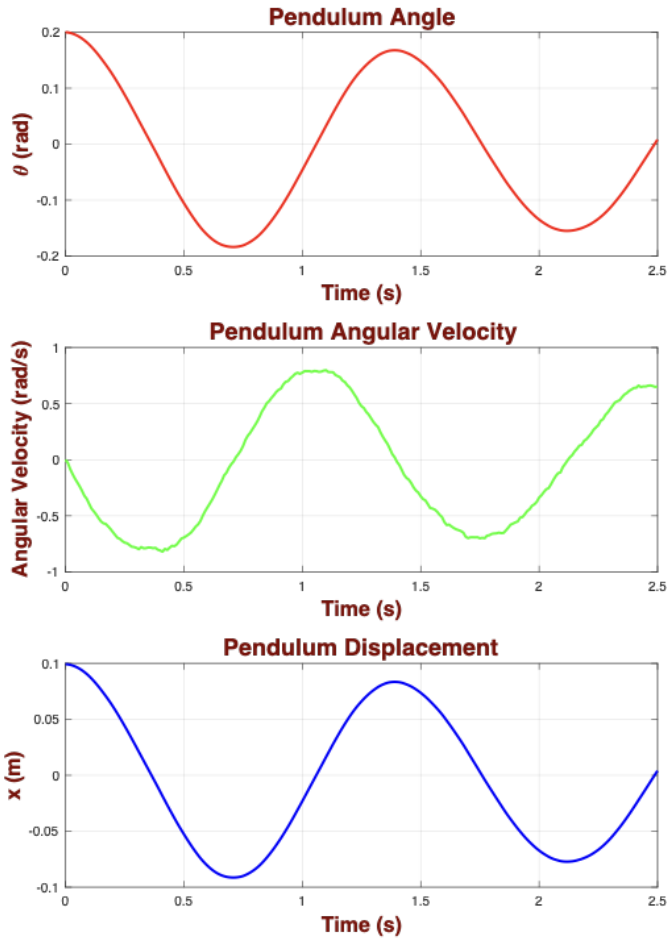


Figure: Pendulum true position, angular velocity, and position, including the process noise.

[Code: Non-linear KF/ex12_Pendulum_background.m]

Example 12 - Estimating the Pendulum Angle: Example Summary

The following figures compare the true, measured, and estimated pendulum angle and angular velocity for 51 measurements.

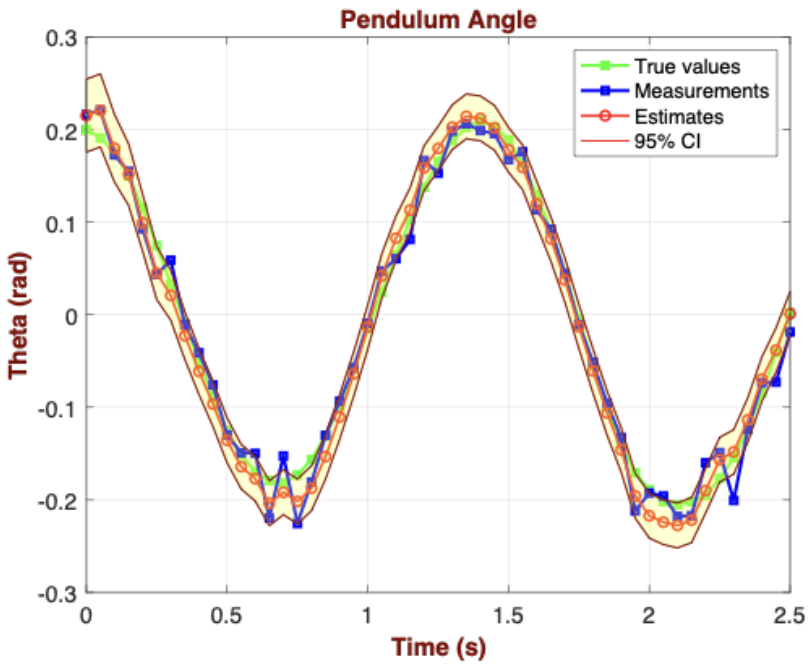


Figure: pendulum angle - true value, measured values and estimates

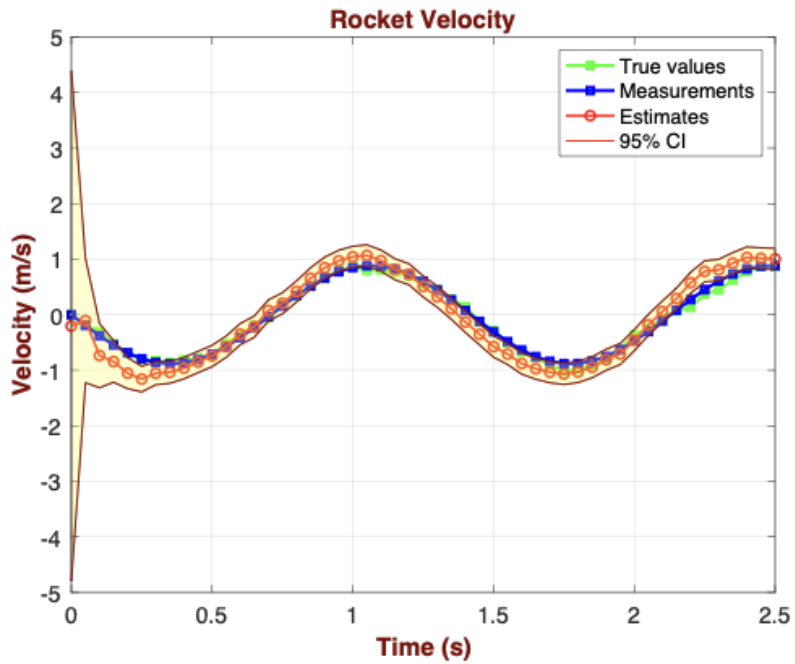


Figure: pendulum velocity - true value, measured values and estimates.

[Code: Non-linear KF/Ex12_EKF_BothUncertainties.m]

Limitations of EKF–Linearization Error

EKF performs well for many practical problems when $f(x)$ or $h(x)$ are close to linear. However, it fails in highly non-linear regions.

The EKF concept is based on the linearization of the model. The EKF estimation includes the **linearization error**. The linearization error depends on the nonlinearity degree of the function compared to the propagated uncertainty, as shown in the following figure.

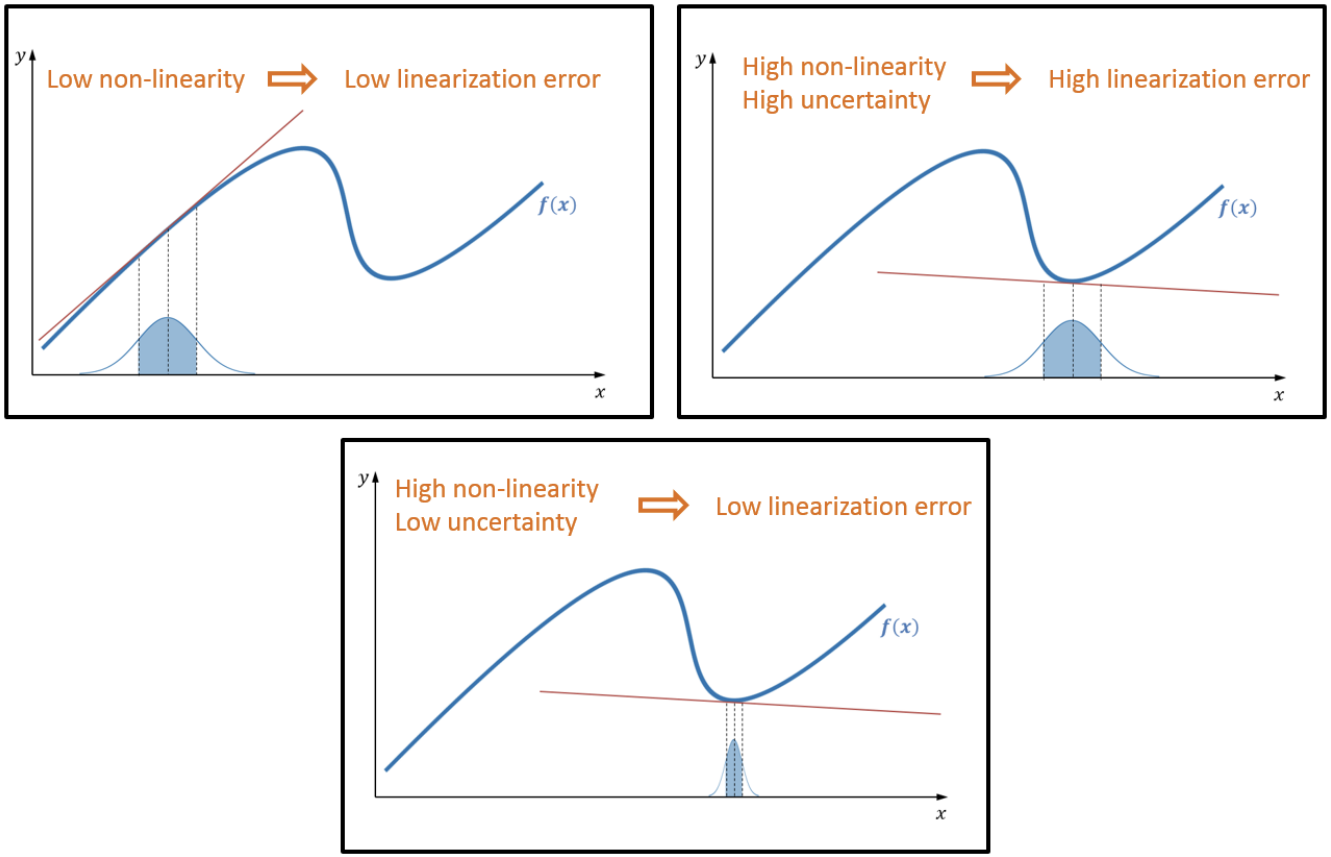


Figure: LinearizarionError

Limitations of EKF–Linearization Error: 2D Example

- Let us see the effect of the linearization error on polar to cartesian transformation.
- Assume a normally distributed random variable in polar coordinates. We want to estimate the random variable parameters in cartesian coordinates.
- A distance vector r and an angle θ describe any value in the polar coordinates. In cartesian coordinates, the values are described by x and y coordinates. The dependency between r, θ, and x, y is non-linear:

$$x = r \cdot \cos(\theta)$$

$$y = r \cdot \sin(\theta)$$

- The random variable parameters (r, θ) in polar coordinates are:

$$\mu = \begin{pmatrix} 1 & \frac{\pi}{2} \end{pmatrix}, \quad \sigma = \begin{pmatrix} 0.05 & 0.5 \end{pmatrix}$$

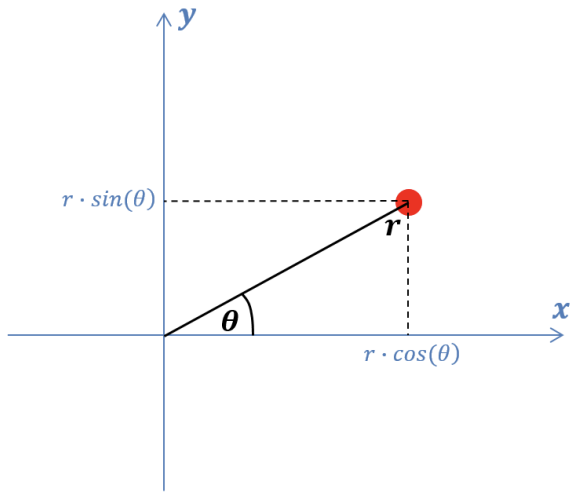


Figure: Linearization Error - 2D Example

Limitations of EKF–Linearization Error: 2D Example

The plot on the left describes the random samples of the random variable in polar coordinates. The right plot describes the random samples of the random variable in cartesian coordinates after the transformation.
The ellipses on the plots represent the covariance of the random variable with 65% confidence interval.

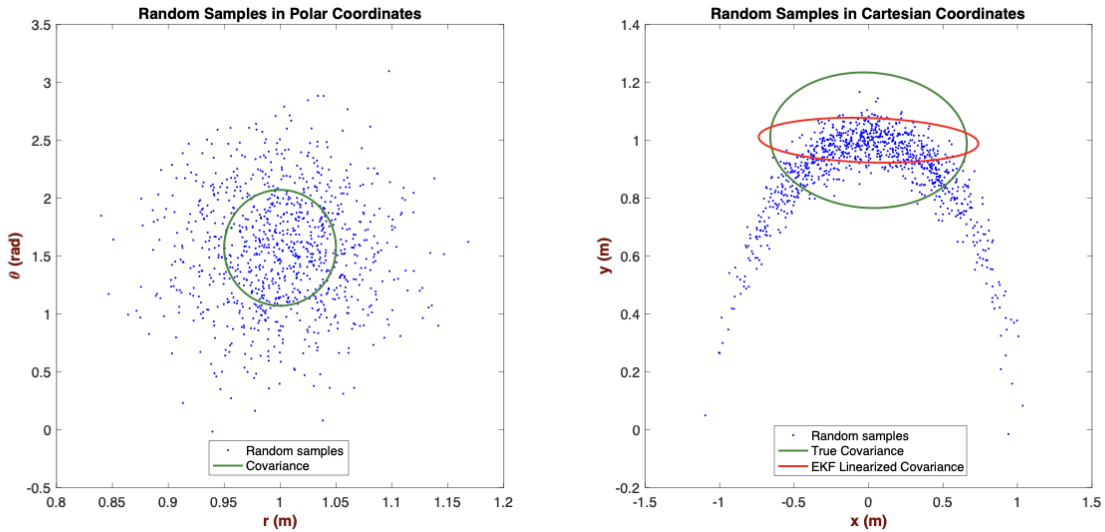


Figure: EKF linearized covariance

We can see a significant difference between actual and EKF linearized covariance. The EKF linearized covariance includes a high linearization error.

The EKF yields a wrong estimation. The EKF estimation uncertainty is also relatively low (the error ellipse is relatively narrow). The EKF is overconfident while making a wrong estimate!

[Code: Non-linear KF/LinearizationError_Polar2Cartisian.m]

Limitations of EKF–Linearization Error: 2D Example - Making a case for UKF

TODO-Update this slide with own figure.

A common alternative to the Extended Kalman Filter is the Unscented Kalman Filter. The following plot compares EKF and UKF linearized covariance for the same example.

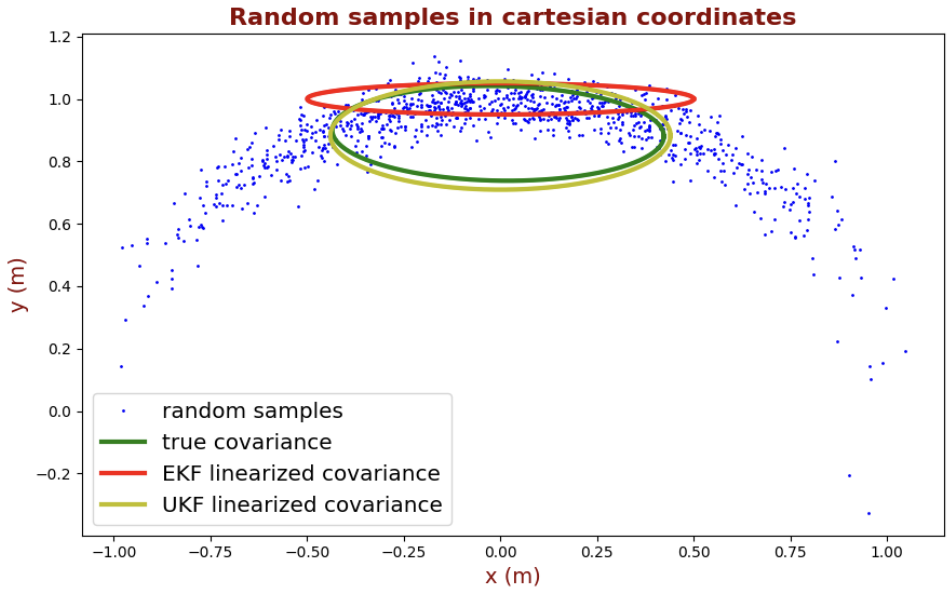


Figure: EKF vs. UKF linearized covariance.

Observe that the UKF linearized covariance is much closer to the actual covariance than the EKF linearized covariance.

Unscented Kalman Filter (UKF)—Motivation

- When the State Transition model $f(x)$ and Observation Model $h(x)$ are close to linear, the EKF performance is satisfying.
- However, when $f(x)$ or $h(x)$ models are highly non-linear, the linearization error can cause estimations that are significantly different from the true value of the state and estimation uncertainties that don't capture the true uncertainties in the state.
- The Unscented Kalman Filter is an alternative approach to linearization. While Extended Kalman Filter treats the non-linearity using analytical linearization, the Unscented Kalman Filter performs Unscented Transform (UT)—statistical linearization.
- Jeffrey Uhlmann initially proposed the unscented transform (UT) as a component of his PhD thesis; however, it is predominantly known from [3].

The Unscented Transform (UT)

Unscented Transform: The Unscented Transform is a method for calculating the statistics of a random variable that undergoes a non-linear transformation. The Unscented Transform includes three steps:

- **Step 1** - Select a set of points from the input distribution. The points are selected according to a specific, deterministic algorithm.
- **Step 2** - Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.
- **Step 3** - Compute sigma points weights.
- **Step 4** - Approximate the sample mean and covariance of the output distribution using the propagated set of points and carefully chosen weights.

The Unscented Transform (UT): Step 1 – Sigma Points Selection

- The set of sigma points includes the mean and a certain number of points located at a certain distance away from the mean.

Number of selected points:

- The number of selected points depends on the input distribution.
- The N–dimensional random variable is approximated by $2N+1$ points. For a one-dimensional distribution ($N=1$), the number of points is 3. For a two-dimensional distribution ($N=2$), the number of points is 5.

Selected points location:

- The first point is the mean of the input distribution:

$$\mathcal{X}_{n,n}^{(0)} = \hat{\mathbf{x}}_{n,n}$$

The superscript in parentheses of $\mathcal{X}_{n,n}^{(0)}$ is a sigma point number.

- The other points are located at a certain **statistical distance** from the mean, expressed in terms of standard deviation or sigma (σ). For this reason, the selected points are called the Sigma Points, and the Unscentenced Transform is sometimes called the Sigma point transform.
- The other sigma points location is:

$$\mathcal{X}_{n,n}^{(i)} = \hat{\mathbf{x}}_{n,n} + \left(\sqrt{(N+\kappa)\mathbf{P}_{n,n}}\right)_i, \quad i = 1, \cdots, N$$

$$\mathcal{X}_{n,n}^{(i)} = \hat{\mathbf{x}}_{n,n} - \left(\sqrt{(N+\kappa)\mathbf{P}_{n,n}}\right)_{i-N}, i = N+1, \cdots, 2$$

- N is the number of dimensions
- κ is a tuning parameter
- $\sqrt{(N+\kappa)\mathbf{P}_{n,n}}_i$ is the i-th row or column of the matrix square root of $\sqrt{(N+\kappa)\mathbf{P}_{n,n}}$

For Gaussian distribution, the rule of thumb is to set: $N+\kappa = 3$. The sigma points should be chosen so that they capture the most important statistical properties of the prior random variable $\hat{\mathbf{x}}$

The Unscented Transform (UT): Step 1 – Sigma Points Selection

Note: In "Kalman Filter language," the mean of the input distribution is the current estimate $\hat{\mathbf{x}}_{n,n}$, and the uncertainty of the input distribution is represented by the covariance matrix of the current estimate $\mathbf{P}_{n,n}$. The main diagonal of the covariance matrix includes variances for each dimension $(\sigma_{xx}, \sigma_{yy})$.

Example: One-dimensional random variable Assume a zero-mean normally distributed one-dimensional random variable with a standard deviation of 2: $\hat{x}_{n,n} = 0$, $p_{n,n} = 2$

- The number of dimensions: $N = 1$
- The number of sigma points: $2N + 1 = 3$
- Set: $N + \kappa = 3$
- The first point is the mean of the input distribution: $\mathbf{X}_{n,n}^{(0)} = 0$
- The second point: $\mathbf{X}_{n,n}^{(1)} = \hat{x}_{n,n} + \sqrt{(N+\kappa)P_{n,n}}_1 = 0 + \sqrt{3 \cdot 2^2} = 3.46$
- The third point: $\mathbf{X}_{n,n}^{(2)} = \hat{x}_{n,n} - \sqrt{(N+\kappa)P_{n,n}}_2 = 0 - \sqrt{3 \cdot 2^2} = -3.46$

The Unscented Transform sigma points are not necessarily on the standard deviation boundaries.

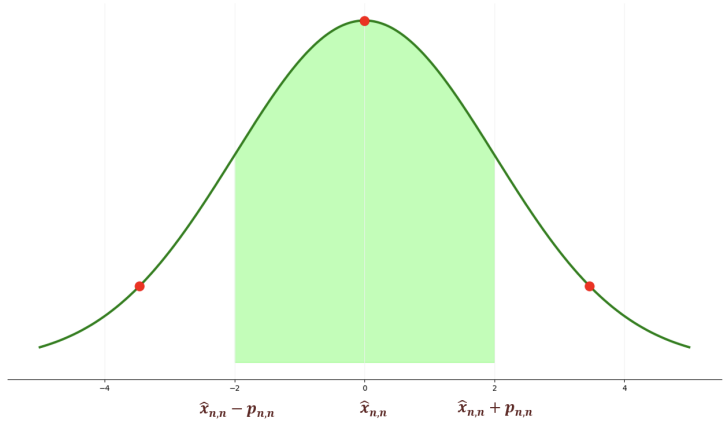


Figure: 1D Sigma Points

The Unscented Transform (UT): Step 1 – Sigma Points Selection

Example: Two-dimensional random variable:
Consider the earlier example with polar (r, θ) to cartesian transformation.

$$\mathbf{x}_{n,n} = \begin{bmatrix} 1 \\ \pi/2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix}$$

$$\mathbf{P}_{n,n} = \begin{bmatrix} 0.05^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} = \begin{bmatrix} 0.0025 & 0 \\ 0 & 0.25 \end{bmatrix}$$

Finding the sigma points:

- The number of dimensions: $N = 2$
- The number of sigma points: $2N + 1 = 5$
- Set: $N + \kappa = 3$
- The first point is the mean of the input distribution: $\mathcal{X}_{n,n}^{(0)} = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix}$
- To find the other points, we should compute: $\sqrt{(N+\kappa)\mathbf{P}_{n,n}}$

$$\sqrt{(N+\kappa)\mathbf{P}_{n,n}} = \sqrt{3\begin{bmatrix} 0.0025 & 0 \\ 0 & 0.25 \end{bmatrix}} = \sqrt{\begin{bmatrix} 0.0075 & 0 \\ 0 & 0.75 \end{bmatrix}}$$

- We need to find the square root of the matrix. Luckily, the covariance matrix is positive and semi-definite; therefore, we can use Cholesky decomposition to find the square root.

$$(N+\kappa)\mathbf{P}_{n,n}$$

- L is a lower triangular matrix, which can be computed in Matlab as L = chol (P)'

$$\mathbf{L} = \begin{bmatrix} 0.0866 & 0 \\ 0 & 0.866 \end{bmatrix}$$

The Unscented Transform (UT): Step 1 – Sigma Points Selection

The second point: $\mathcal{X}_{n,n}^{(1)} = \hat{\mathbf{x}}_{n,n} + \left(\sqrt{(N+\kappa)\mathbf{P}_{n,n}}\right)_1 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} + \begin{bmatrix} 0.0866 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.0866 \\ 1.57 \end{bmatrix}$

The third point: $\mathcal{X}_{n,n}^{(2)} = \hat{\mathbf{x}}_{n,n} + \left(\sqrt{(N+\kappa)\mathbf{P}_{n,n}}\right)_2 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.866 \end{bmatrix} = \begin{bmatrix} 1 \\ 2.436 \end{bmatrix}$

The fourth point: $\mathcal{X}_{n,n}^{(3)} = \hat{\mathbf{x}}_{n,n} - \left(\sqrt{(N+\kappa)\mathbf{P}_{n,n}}\right)_1 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} - \begin{bmatrix} 0.0866 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.9134 \\ 1.57 \end{bmatrix}$

The fifth point: $\mathcal{X}_{n,n}^{(4)} = \hat{\mathbf{x}}_{n,n} - \left(\sqrt{(N+\kappa)\mathbf{P}_{n,n}}\right)_2 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.866 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.704 \end{bmatrix}$

The following figure describes the covariance ellipse of the random variable PDF with sigma points (red circles).

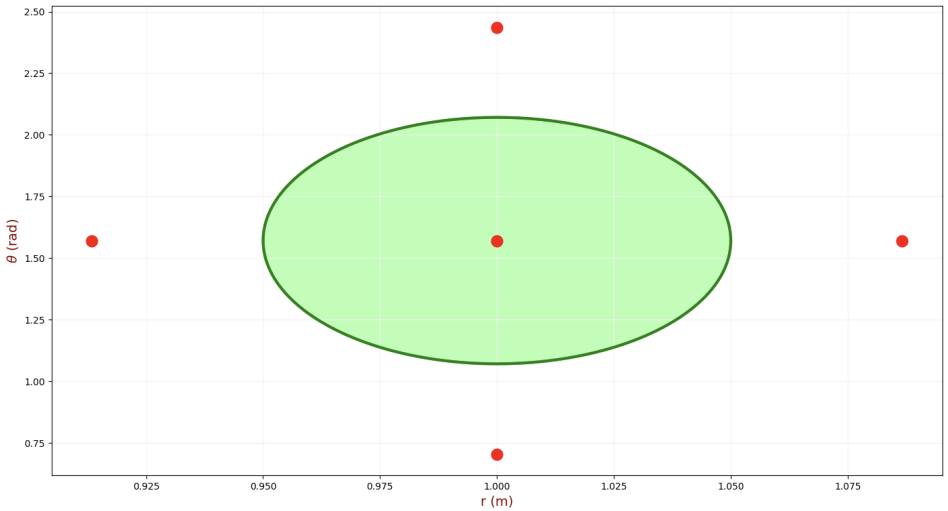


Figure: 2D RV Sigma Points

The Unscented Transform sigma points are not necessarily on the covariance ellipse boundaries.

The Unscented Transform (UT): Step 2 – Points Propagation

Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.

Example: one-dimensional random variable (continued) The non-linear function is:

$$f(x) = \sin(2x)\sin(0.3x) + 2x$$

$$\mathcal{X}_{n+1,n} = f(\mathcal{X}_{n,n})$$

The propagated (or transformed) sigma points:

$$\mathcal{X}^{(0)}_{n+1,n} = \sin(2 \cdot 0)\sin(0.3 \cdot 0) + 2 \cdot 0 = 0$$

$$\mathcal{X}^{(1)}_{n+1,n} = \sin(2{\cdot}3.46)\sin(0.3{\cdot}3.46){+}2{\cdot}3.46 = 7.45$$

$$\mathcal{X}^{(2)}_{n+1,n} = \sin(2 \cdot (-3.46))\sin(0.3 \cdot (-3.46)) + 2 \cdot (-3.46) = -6.41$$

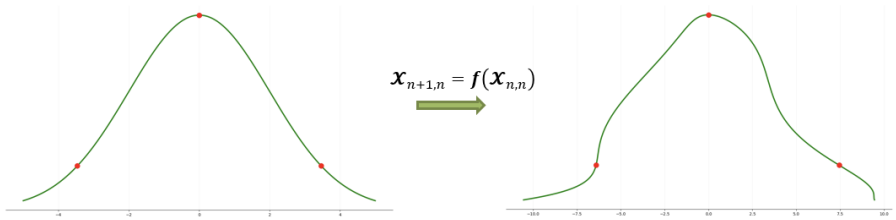


Figure: 1D RV Sigma Points propagation.

- The green line on the left plot is the PDF of the input random variable. The red circles on the left plot are the sigma points ($\mathcal{X}_{n,n}$) of the input random variable.
- The green line on the right plot is the PDF of the input random variable after the non-linear transformation. The red circles on the right plot are the sigma points after the non-linear transformation ($\mathcal{X}_{n+1,n}$).

The Unscented Transform (UT): Step 2 – Points Propagation

Example: two-dimensional random variable (continued) The non-linear function is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r \cdot \cos(\theta) \\ r \cdot \sin(\theta) \end{bmatrix}$$

$$\mathcal{X}_{n+1,n} = \mathbf{f}(\mathcal{X}_{n,n})$$

The propagated (or transformed) sigma points:

$$\mathcal{X}^{(0)}_{n+1,n} = \mathbf{f}(\mathcal{X}^{(0)}_{n,n}) = \begin{bmatrix} 1 \cdot \cos\left(\frac{\pi}{2}\right) \\ 1 \cdot \sin\left(\frac{\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathcal{X}^{(1)}_{n+1,n} = \mathbf{f}(\mathcal{X}^{(1)}_{n,n}) = \begin{bmatrix} 1.0866 \cdot \cos\left(\frac{\pi}{2}\right) \\ 1.0866 \cdot \sin\left(\frac{\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 1.0866 \end{bmatrix}$$

$$\mathcal{X}^{(2)}_{n+1,n} = \mathbf{f}(\mathcal{X}^{(2)}_{n,n}) = \begin{bmatrix} 1 \cdot \cos(2.436) \\ 1 \cdot \sin(2.436) \end{bmatrix} = \begin{bmatrix} -0.762 \\ 0.648 \end{bmatrix}$$

$$\mathcal{X}^{(3)}_{n+1,n} = \mathbf{f}(\mathcal{X}^{(3)}_{n,n}) = \begin{bmatrix} 0.9134 \cdot \cos\left(\frac{\pi}{2}\right) \\ 0.9134 \cdot \sin\left(\frac{\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 0.9134 \end{bmatrix}$$

$$\mathcal{X}^{(4)}_{n+1,n} = \mathbf{f}(\mathcal{X}^{(4)}_{n,n}) = \begin{bmatrix} 1 \cdot \cos(0.704) \\ 1 \cdot \sin(0.704) \end{bmatrix} = \begin{bmatrix} 0.762 \\ 0.648 \end{bmatrix}$$

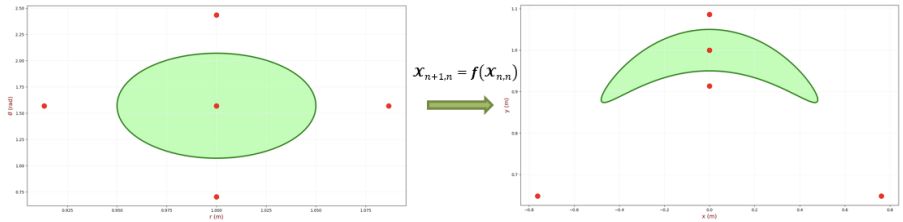


Figure: 2D RV Sigma Points propagation.

- The green shape on the left plot is the covariance ellipse of the input random variable. The red circles on the left plot are the sigma points ($\mathcal{X}_{n,n}$) of the input random variable.
- The green shape on the right plot is the covariance ellipse of the input random variable after the non-linear transformation. The red circles on the right plot are the sigma points after the non-linear transformation ($\mathcal{X}_{n+1,n}$).

The Unscented Transform (UT): Step 3 & 4

Step 3 – compute sigma points weights

We should compute two weights:

- w_0—weight of the first sigma point $\mathcal{X}_{n,n}^{(0)}$

$$w_0 = \frac{\kappa}{N + \kappa}$$

- w_i—weight of the other sigma points $\mathcal{X}_{n,n}^{(i)}$, $i > 0$

$$w_i = \frac{1}{2(N + \kappa)}, \quad i > 0$$

The Unscented Transform (UT): Step 4 - approximate the mean and covariance of the output distribution: In this step, we approximate the sample mean and covariance of the output distribution using the propagated set of points and carefully chosen weights.

- The mean of the output distribution:

$$\hat{x}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)}$$

- The covariance of the output is also computed with weights:

$$\mathbf{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{x}_{n+1,n} \right) \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{x}_{n+1,n} \right)^T$$

The Unscented Transform (UT): Step 4 - approximate the mean and covariance of the output distribution (Examples)

Example: one-dimensional random variable
Weights computation:

- The number of dimensions: $N = 1$
- $N + \kappa = 3,\ \kappa = 2$

$$w_0 = \frac{\kappa}{N + \kappa} = \frac{2}{3}$$

$$w_i = \frac{1}{2(N + \kappa)} = \frac{1}{2 \cdot 3} = \frac{1}{6}$$

Mean computation:

$$\hat{x}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)}$$

$$= \frac{2}{3} \cdot 0 + \frac{1}{6} \cdot 7.45 + \frac{1}{6} \cdot (-6.42) = 0.17$$

Covariance computation (variance in 1D):

$$\mathbf{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{x}_{n+1,n} \right) \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{x}_{n+1,n} \right)^T$$

$$= \frac{2}{3} \cdot (0 - 0.17)^2 + \frac{1}{6} \cdot (7.45 - 0.17)^2 +$$

$$\frac{1}{6} \cdot (-6.42 - 0.17)^2 = 16.06$$

The following plot depicts the output random variable after the Unscented Transform.

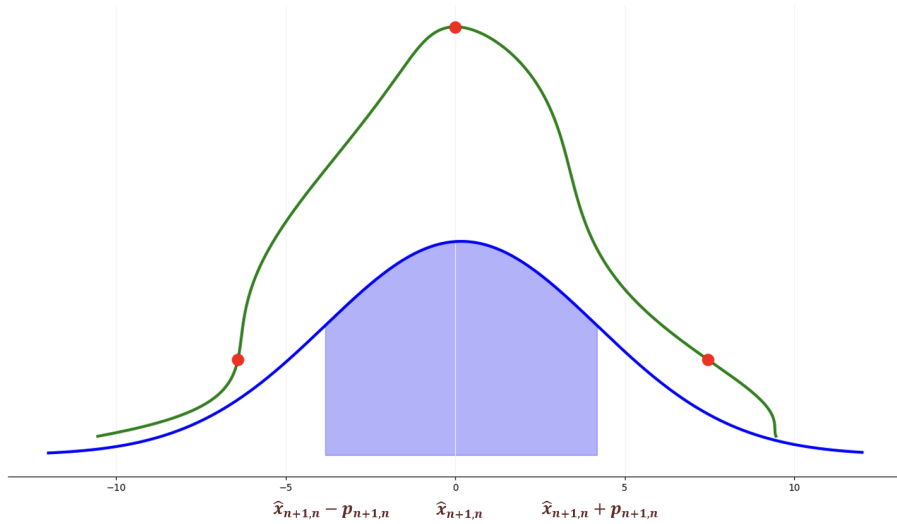


Figure: 1D RV Unscented Transform

- The green line is the PDF of the input random variable after non-linear transformation. The red circles are the sigma points after the non-linear transformation ($\mathcal{X}_{n+1,n}$).
- The blue line is the PDF of the output random variable after the Unscented

The Unscented Transform (UT): Step 4 - approximate the mean and covariance of the output distribution

Example: two-dimensional random variable (continued) Weights computation:

- The number of dimensions: $N = 2$
- $N + \kappa = 3$
- $\kappa = 1$

$$w_0 = \frac{\kappa}{N+\kappa} = \frac{1}{3}$$

$$w_i = \frac{1}{2(N+\kappa)} = \frac{1}{2 \cdot 3} = \frac{1}{6}$$

Mean computation:

$$\hat{x}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)} = \frac{1}{3}\begin{bmatrix}0\\1\end{bmatrix} + \frac{1}{6}\begin{bmatrix}0\\1.0866\end{bmatrix} + \frac{1}{6}\begin{bmatrix}-0.762\\0.648\end{bmatrix} + \frac{1}{6}\begin{bmatrix}0\\0.913\end{bmatrix} + \frac{1}{6}\begin{bmatrix}0.762\\0.648\end{bmatrix} = \begin{bmatrix}0\\0.8826\end{bmatrix}$$

$\mathbf{P}_{n+1,n}$

$$= \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{x}_{n+1,n}\right)\left(\mathcal{X}_{n+1,n}^{(i)} - \hat{x}_{n+1,n}\right)^T$$

$$= \frac{1}{3}\left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)\left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)^T + \frac{1}{6}\left(\begin{bmatrix}0\\1.0866\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)\left(\begin{bmatrix}0\\1.0866\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)^T + \frac{1}{6}\left(\begin{bmatrix}-0.762\\0.648\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)\left(\begin{bmatrix}-0.762\\0.648\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)^T + \frac{1}{6}\left(\begin{bmatrix}0\\0.913\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)\left(\begin{bmatrix}0\\0.913\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)^T + \frac{1}{6}\left(\begin{bmatrix}0.762\\0.648\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)\left(\begin{bmatrix}0.762\\0.648\end{bmatrix} - \begin{bmatrix}0\\0.8826\end{bmatrix}\right)^T$$

$$= \begin{bmatrix}0.193 & 0\\0 & 0.03\end{bmatrix}$$

The Unscented Transform (UT): Step 4 - approximate the mean and covariance of the output distribution

We can compute the mean and covariance more elegantly.
Define a matrix (container) of transformed sigma points:

$$\mathbf{X}_{n+1,n} = \begin{bmatrix} \mathcal{X}^{(0)}_{n+1,n} & \mathcal{X}^{(1)}_{n+1,n} & \cdots & \mathcal{X}^{(2N)}_{n+1,n} \end{bmatrix}$$

Define a vector of weights:

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_{2N} \end{bmatrix}$$

Mean computation:

$$\hat{x}_{n+1,n} = \mathcal{X}_{n+1,n}\mathbf{w}$$

$$\hat{x}_{n+1,n} = \begin{bmatrix} 0 & 0 & -0.762 & 0 & 0.762 \\ 1 & 1.0866 & 0.648 & 0.913 & 0.648 \end{bmatrix} \times \begin{bmatrix} \frac{1}{3} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \end{bmatrix} = \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}$$

Covariance computation:

$$\mathbf{P}_{n+1,n} = (\mathcal{X}_{n+1,n} - \hat{x}_{n+1,n}) \cdot \text{diag}(\mathbf{w}) \cdot (\mathcal{X}_{n+1,n} - \hat{x}_{n+1,n})^T$$

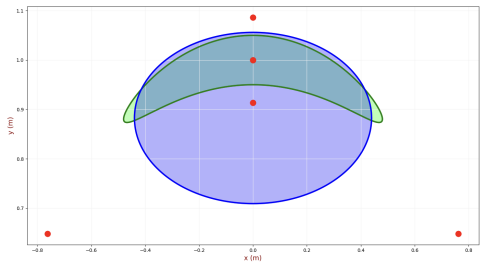


Figure: 2D RV Unscented Transform.

- The green shape is the covariance ellipse of the input random variable after the non-linear transformation. The red circles on the right plot are the sigma points after the non-linear transformation ($\mathcal{X}_{n+1,n}$).
- The blue ellipse is the covariance ellipse of the output random variable after the Unscented Transform.

The UKF algorithm - Prediction Stage

UKF operates in a "predict–correct" loop, business as usual. In the prediction stage, UKF extrapolates the current estimate to the next state using the Unscented Transform:

$$\hat{\mathbf{x}}_{n,n} \rightarrow \hat{\mathbf{x}}_{n+1,n}$$

$$\mathbf{P}_{n,n} \rightarrow \mathbf{P}_{n+1,n}$$

LKF	**UKF**
	$\mathcal{X}_{n,n}^{(0)} = \hat{\boldsymbol{x}}_{n,n}$
	$\mathcal{X}_{n,n}^{(i)} = \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_i, \qquad i = 1, \cdots, N$
	$\mathcal{X}_{n,n}^{(i)} = \hat{\boldsymbol{x}}_{n,n} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_{i-N}, \qquad i = N+1, \cdots, 2N$
$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n}$	$\mathcal{X}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}_{n,n}\right)$
$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$	$\hat{\boldsymbol{x}}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)}$
	$\boldsymbol{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)\left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)^T$

Figure: LKF and UKF predict stage equations.

The UKF algorithm - Update Stage

Before proceeding to the update stage, the **statistical linear regression** concept must be understood.

Consider a non-linear function $\mathbf{y} = g(\mathbf{x})$ evaluated at r points $(\mathcal{X}(i), \mathcal{Y}(i))$, where $\mathcal{Y}(i) = g(\mathcal{X}(i))$. Let's define:

$\overline{\boldsymbol{x}} = \frac{1}{r}\sum_{i=1}^{r}\mathcal{X}^{(i)}$	Mean of $\mathcal{X}^{(i)}$
$\overline{\boldsymbol{y}} = \frac{1}{r}\sum_{i=1}^{r}\mathcal{Y}^{(i)}$	Mean of $\mathcal{Y}^{(i)}$
$\boldsymbol{P}_{xx} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)^T$	Variance of $\mathcal{X}^{(i)}$
$\boldsymbol{P}_{yy} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)^T$	Variance of $\mathcal{Y}^{(i)}$
$\boldsymbol{P}_{xy} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)^T$	Cross-variance of $\mathcal{X}^{(i)}$ and $\mathcal{Y}^{(i)}$
$\boldsymbol{P}_{yx} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)^T$	Cross-variance of $\mathcal{Y}^{(i)}$ and $\mathcal{X}^{(i)}$

Figure: Definitions

We want to approximate the non-linear function $\mathbf{y} = g(\mathbf{x})$ by a linear function $\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{b}$. The linear approximation produces linearization error. For each point $\mathcal{X}^{(i)}$, the linearization error is given by

$$\mathbf{e}^{(i)} = \mathcal{Y}^{(i)} - (\mathbf{M}\mathcal{X}^{(i)} + \mathbf{b})$$

To minimize the linearization error, we should find $\mathbf{M}$ and $\mathbf{b}$ that minimize the sum of squared errors for all $\mathcal{X}(i)$ points:

$$\min_{\mathbf{M},\mathbf{b}} \sum_{i=1}^{r} (\mathbf{e}(i))^T \mathbf{e}(i)$$

The solution of this minimization problem is:

$$\boxed{\begin{aligned} \mathbf{M} &= \mathbf{P}_{xy}^T \mathbf{P}_{xx}^{-1} = \mathbf{P}_{yx}\mathbf{P}_{xx}^{-1} \\ \mathbf{b} &= \bar{\mathbf{z}} - \mathbf{M}\bar{\mathbf{x}} \end{aligned}}$$

See proof in Appendix F of the book.

The UKF algorithm - Update Stage

State update: After a unit delay, the $\mathcal{X}_{n+1,n}$ becomes $\mathcal{X}_{n,n-1}$, and $\hat{\mathbf{x}}_{n+1,n}$ becomes $\hat{\mathbf{x}}_{n,n-1}$.

Using the Unscented Transform, transfer the state sigma points ($\mathcal{X}_{n,n-1}$) to the measurement space ($\mathcal{Z}$) using the measurement function $\mathbf{h}(\mathbf{x})$:

$$\mathcal{Z}_n = \mathbf{h}(\mathcal{X}_{n,n-1})$$

$$\bar{\mathbf{z}}_n = \sum_{i=0}^{2N} w_i \mathbf{Z}_n^{(i)}$$

Update estimate with measurement:

$$\boxed{\hat{\mathbf{x}}_{n,n} = \hat{\mathbf{x}}_{n,n-1} + \mathbf{K}_n(\mathbf{z}_n - \bar{\mathbf{z}}_n)}$$

Kalman gain derivation: The Kalman Gain ($\mathbf{K}_n$) transforms the innovation ($\mathbf{z}_n - \bar{\mathbf{z}}_n$) and the measurement covariance ($\mathbf{P}_{\mathbf{z}_n}$) from the measurement space to the system space using linearization:

$$\text{innovation} = (\mathbf{M}\mathbf{z}_n + \mathbf{b}) - (\mathbf{M}\bar{\mathbf{z}}_n + \mathbf{b}) = \mathbf{M}(\mathbf{z}_n - \bar{\mathbf{z}}_n)$$

Since the Kalman Gain performs a linear transformation, it produces linearization errors. We want to minimize the linearization errors using the statistical linear regression method. Therefore, the Kalman gain is given by:

$$\boxed{\mathbf{K}_n = \mathbf{M} = \mathbf{P}_{\mathbf{x}\mathbf{z}_n}\mathbf{P}_{\mathbf{z}_n}^{-1}}$$

Compute the weighted variance of the measurement ($\mathbf{P}_{\mathbf{z}_n}$) and cross-covariance of the state and the measurement ($\mathbf{P}_{\mathbf{x}\mathbf{z}_n}$):

$$\mathbf{P}_{\mathbf{z}_n} = \sum_{i=0}^{2N} w_i \left(\mathcal{Z}_n^{(i)} - \bar{\mathbf{z}}_n\right)\left(\mathcal{Z}_n^{(i)} - \bar{\mathbf{z}}_n\right)^T + \mathbf{R}_n$$

$$\mathbf{P}_{\mathbf{x}\mathbf{z}_n} = \sum_{i=0}^{2N} w_i \left(\mathbf{X}_n^{(i)} - \hat{\mathbf{x}}_{n,n-1}\right)\left(\mathbf{Z}_n^{(i)} - \bar{\mathbf{z}}_n\right)^T$$

Covariance update equation:

$$\boxed{\mathbf{P}_{n,n} = \mathbf{P}_{n-1,n} - \mathbf{K}_n\mathbf{P}_{z_n}\mathbf{K}_n^T}$$

The UKF algorithm - Update Stage Summary

LKF	UKF
	$\mathcal{Z}_n = \boldsymbol{h}\left(\mathcal{X}_{n,n-1}\right)$
$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T \times \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$	$\boldsymbol{\mu}_{z_n} = \sum_{i=0}^{2N} w_i \mathcal{Z}_n^{(i)}$
	$\boldsymbol{P}_{z_n} = \sum_{i=0}^{2N} w_i \left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)\left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)^T + \boldsymbol{R}_n$
$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}\right)$	
	$\boldsymbol{P}_{xz_n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n,n-1}^{(i)} - \hat{\boldsymbol{x}}_{n,n-1}\right)\left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)^T$
$\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)\boldsymbol{P}_{n,n-1}\left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)^T +$	
$+\boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$	$\boldsymbol{K}_n = \boldsymbol{P}_{xz_n}\left(\boldsymbol{P}_{z_n}\right)^{-1}$
	$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)$
	$\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n-1,n} - \boldsymbol{K}_n\boldsymbol{P}_{z_n}\boldsymbol{K}_n^T$

Figure: LKF and UKF update stage equations.

The UKF algorithm Summary

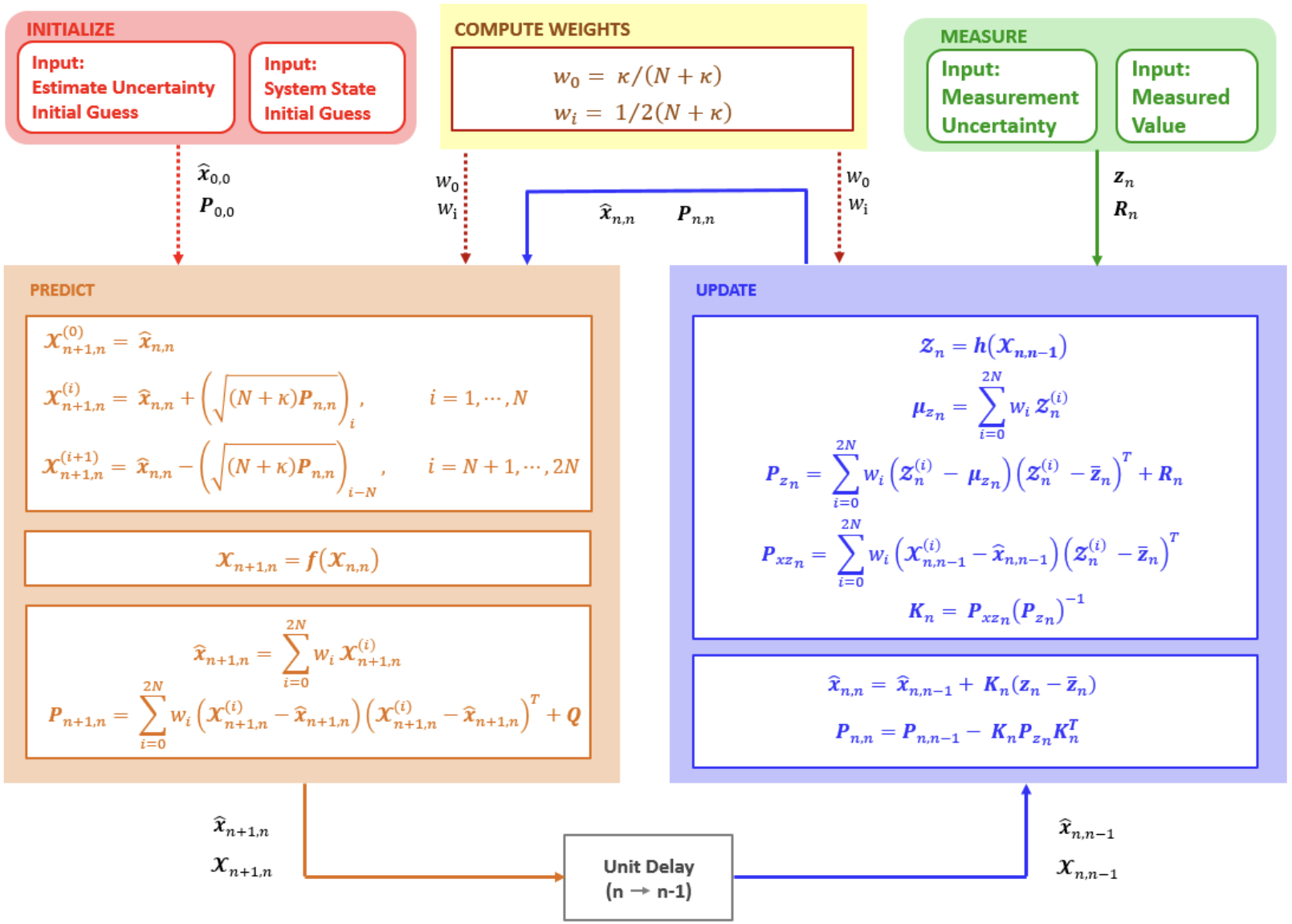


Figure: UKF algorithm summary diagram

Example 13 – vehicle location estimation using radar

Now we apply UKF to Example 11 (see Slide 134).

- We want to track the vehicle using radar, which is located at the plane origin, measuring the vehicle range (r) and the bearing angle (φ).
- The radar measurement error distribution is Gaussian.
- The state transition matrix $\mathbf{F}$, the process noise matrix $\mathbf{Q}$, the measurement covariance $\mathbf{R}$, and the measurement model $\mathbf{h}(\mathbf{x})$ are similar to example 11.

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

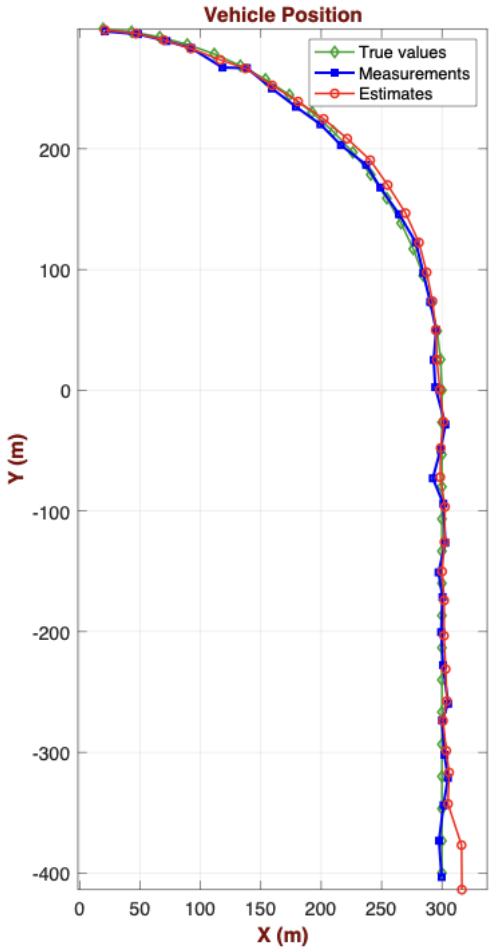
$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2$$

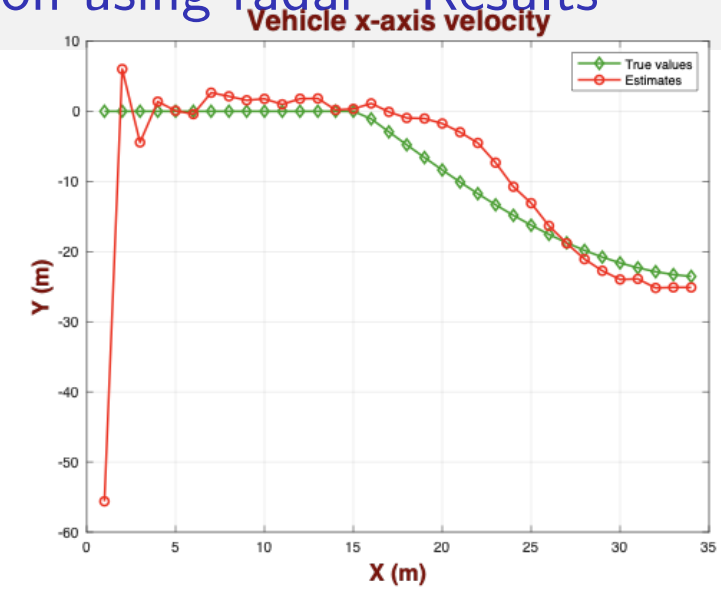
$$\mathbf{z}_n = \mathbf{h}(\mathbf{x}_n)$$

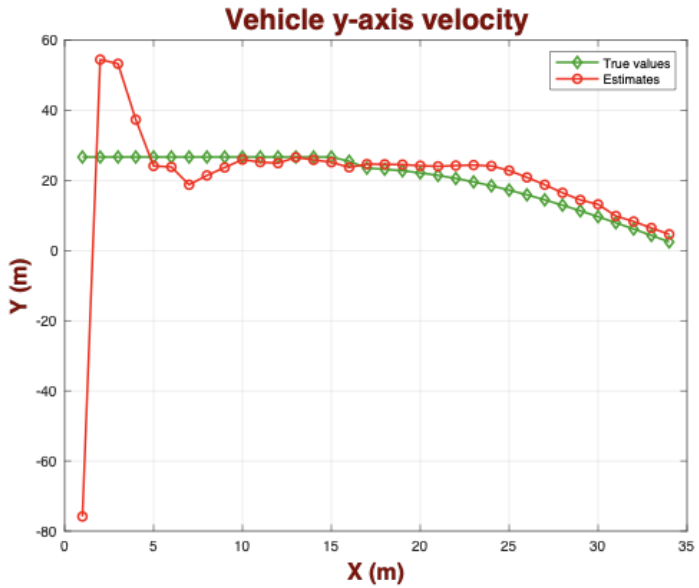
$$\mathbf{z}_n = \begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}\left(\frac{y}{x}\right) \end{bmatrix}$$

$$\mathbf{R}_n = \begin{bmatrix} \sigma_{rm}^2 & 0 \\ 0 & \sigma_{\phi m}^2 \end{bmatrix} = \begin{bmatrix} 5^2 & 0 \\ 0 & 0.0087^2 \end{bmatrix}$$

Example 13 – vehicle location estimation using radar - Results



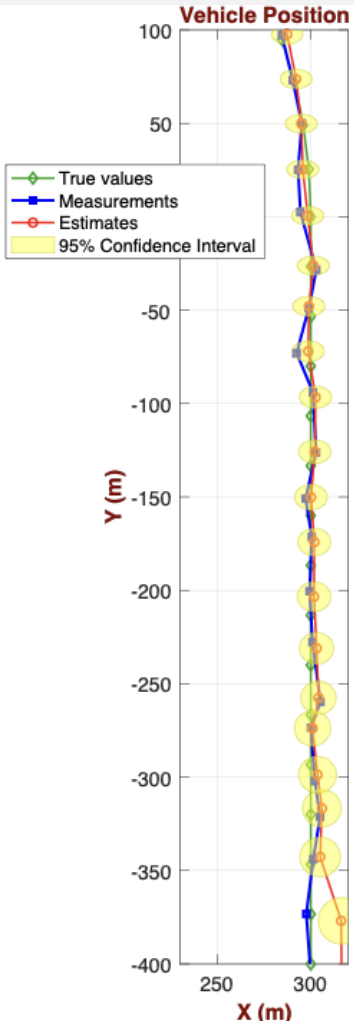




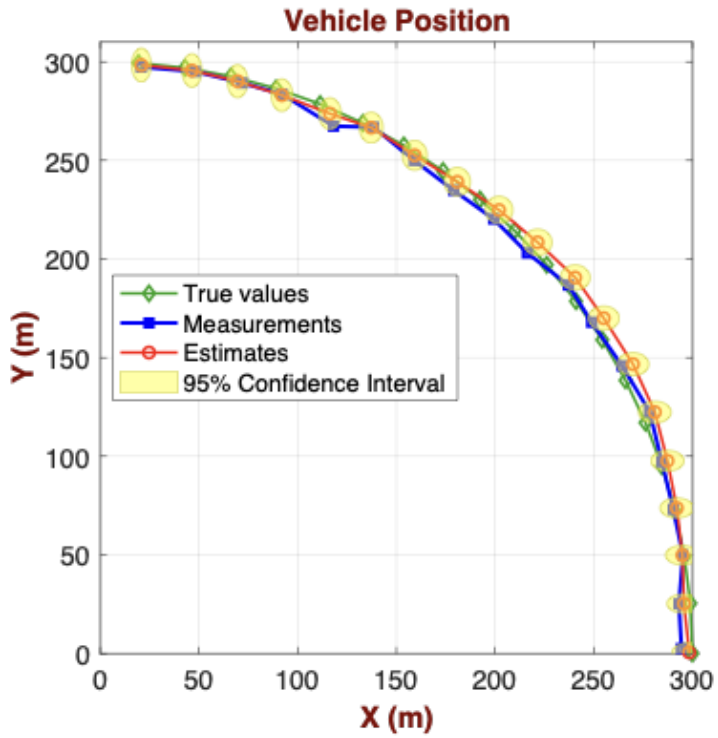
Although the filter is roughly initiated at about 100 meters from the true position with zero initial velocity, it provides a good position estimation after taking two measurements and a velocity estimation after taking four measurements.

[Code: Non-linear KF/Ex13_UKF_StatetoMeasurementUncertainty.m]

Example 13 – vehicle location estimation using radar - Results



- We can see that at the linear part of the vehicle motion, the UKF copes with the noisy measurements and follows the true vehicle position.



- During the vehicle turning maneuver, the UKF estimates are quite away from the true vehicle position, although they are within the 90% confidence ellipse bounds.
- We can see that the ellipses' size constantly decreases. That means that the UKF converges with time.

Sigma Point Algorithm Modification

- The most common/accepted modification to the *sigma point computation algorithm* in Unscented Transform is proposed in [4].
- The algorithm parameterizes sigma point computation by α, β, λ, and κ tuning parameters.
- The **first** key difference is the initial parameter λ (i.e., κ in UT) for the sigma points:

$$\boxed{\lambda = \alpha^2(N + \kappa) - N}$$

  - N is the number of dimensions.
  - $0 < \alpha \leq 1$ determines the spread of the sigma points around the mean; usually set to a small positive value (e.g., $\alpha = 0.001$). Higher $\alpha \rightarrow$ higher spread.
  - κ is a secondary scaling parameter that is usually set to 0.
  - β is used to incorporate prior knowledge of the distribution of the input random variable (for Gaussian distributions, $\beta = 2$ is optimal).

- Sigma points are calculated as before
- The **Second** difference is how the sigma oint weights are set:

$$w_0^{(m)} = \frac{\lambda}{N + \kappa}$$

$$w_0^{(c)} = \frac{\lambda}{N + \lambda} + (1 - \alpha^2 + \beta)$$

$$w_i = \frac{1}{2(N + \lambda)}, \quad i > 0$$

  - $w_0^{(m)}$ is the weight for the first sigma point $\mathcal{X}_{n,n}^{(0)}$ when computing the weighted mean.
  - $w_0^{(c)}$ is the weight for the first sigma point $\mathcal{X}_{n,n}^{(0)}$ when computing the weighted covariance.
  - w_i is the weight for the other sigma points $\mathcal{X}_{n,n}^{(i)}$, $i > 0$ when computing the weighted mean or covariance.

Sigma Point Algorithm Modification

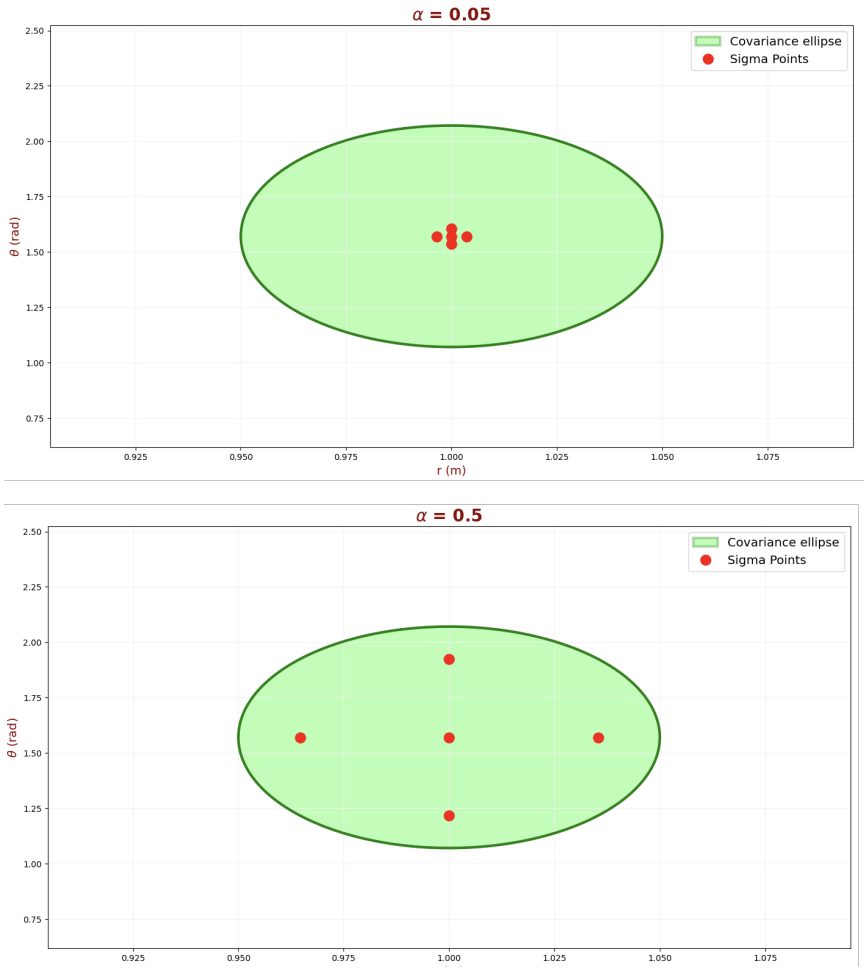


Figure: How α influences the Sigma Points.

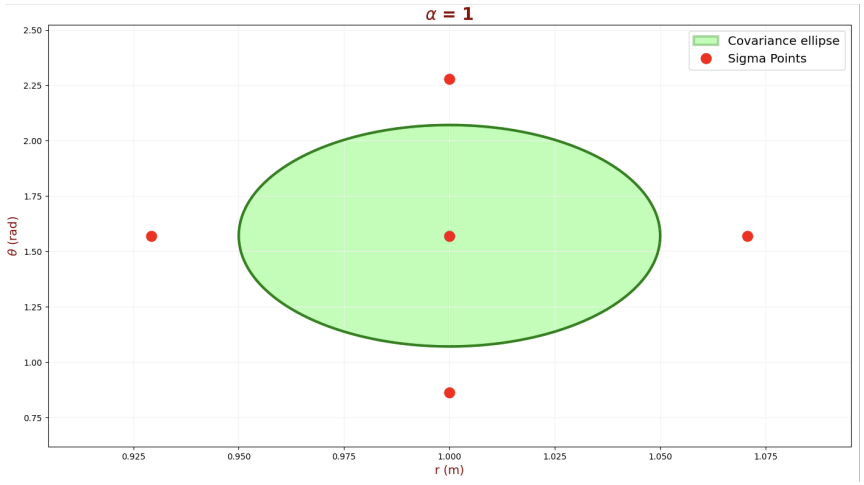
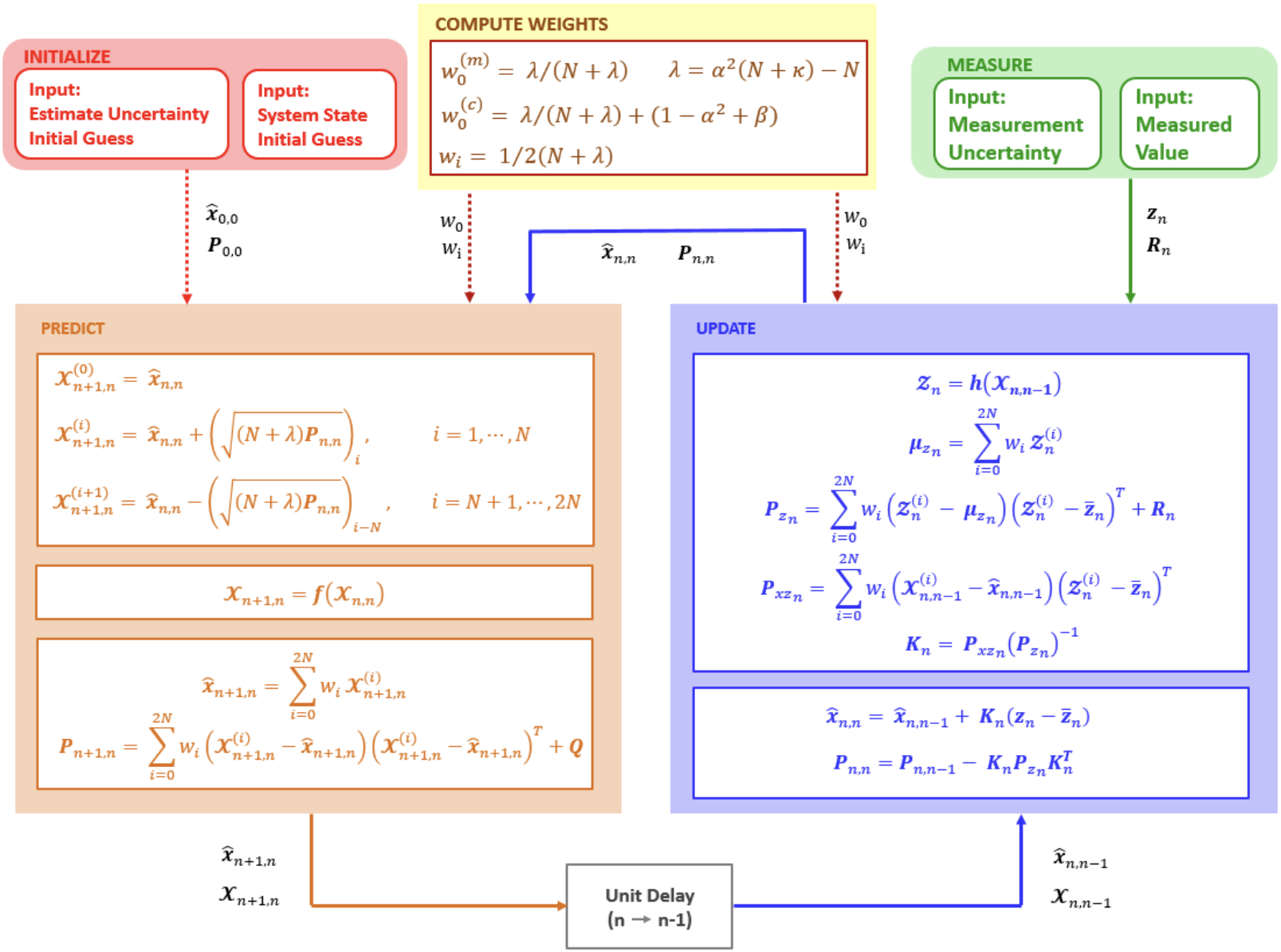


Figure: How α influences the Sigma Points.

One should experiment for the right choice of α.

Modified UKF algorithm summary



Example 14 - Estimating the pendulum angle

We measure the pendulum position $z = Lsin(\theta_n)$.
The state vector of the pendulum is:

$$\mathbf{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix}$$

where θ_n is the pendulum's angle $\dot{\theta}_n$ the angular velocity at time n
The dynamic model of the pendulum is non-linear:

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{f}(\hat{\mathbf{x}}_{n,n})$$

$$\hat{\mathbf{x}}_{n+1,n} = \begin{bmatrix} \hat{\theta}_{n+1,n} \\ \hat{\dot{\theta}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix}$$

$$\mathbf{f}(\hat{\mathbf{x}}_{n,n}) = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L}\sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix}$$

The estimate covariance is:

$$\mathbf{P} = \begin{bmatrix} p_x & p_{x\dot{x}} \\ p_{\dot{x}x} & p_{\dot{x}} \end{bmatrix}$$

The Process Noise Matrix:

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2$$

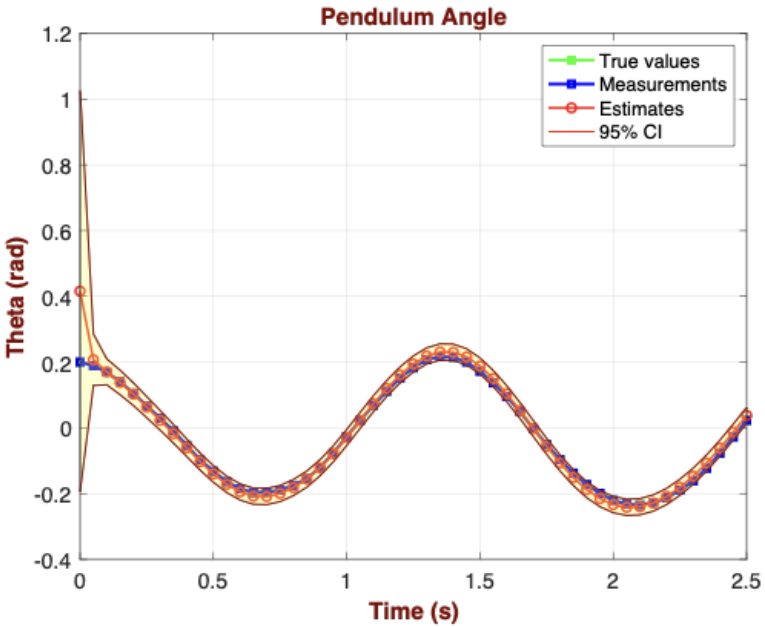
The Measurement Equation We measure the pendulum position: $z = L\sin(\theta_n)$, i.e., the state-to-measurement relation is non-linear:

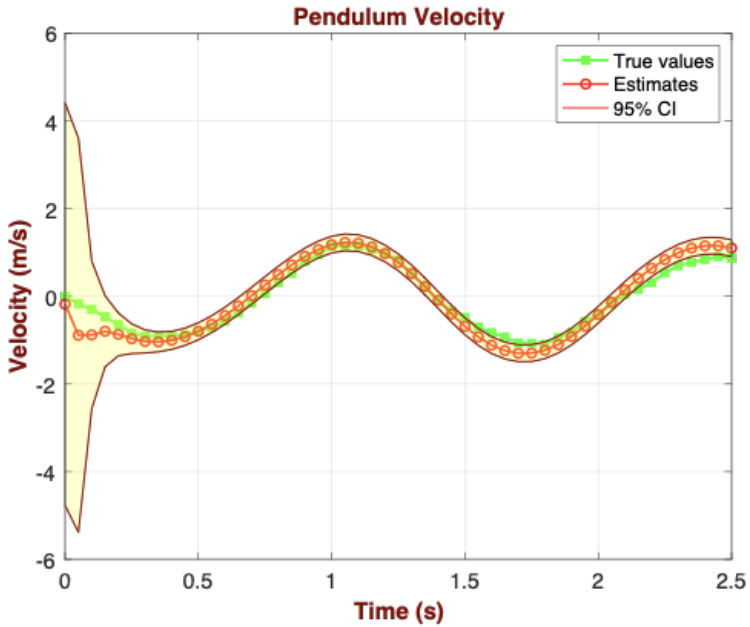
$$\mathbf{z}_n = \mathbf{h}(\mathbf{x}_n), \quad \mathbf{h}(\mathbf{x}_n) = L\sin(\theta_n)$$

The Measurement Uncertainty is:

$$\mathbf{R}_n = \begin{bmatrix} \sigma_{x_m}^2 \end{bmatrix}$$

Example 14 - Estimating the pendulum angle

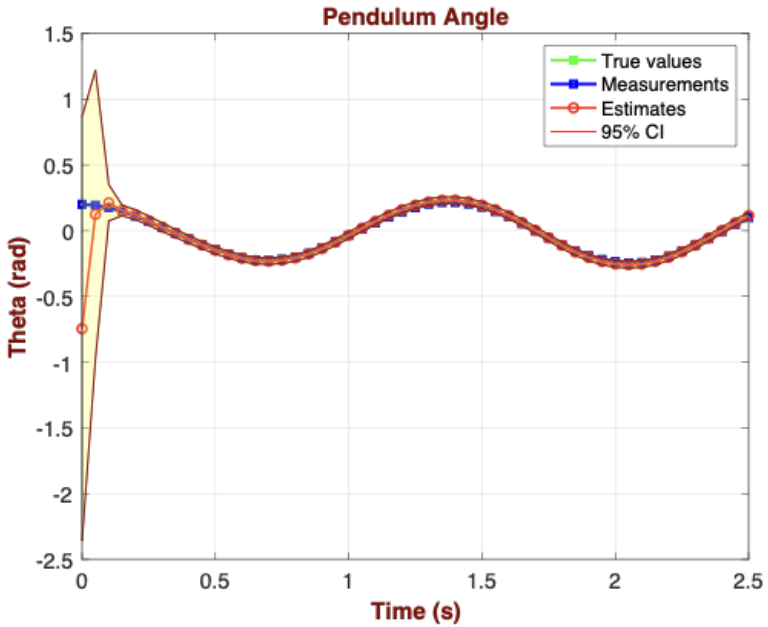


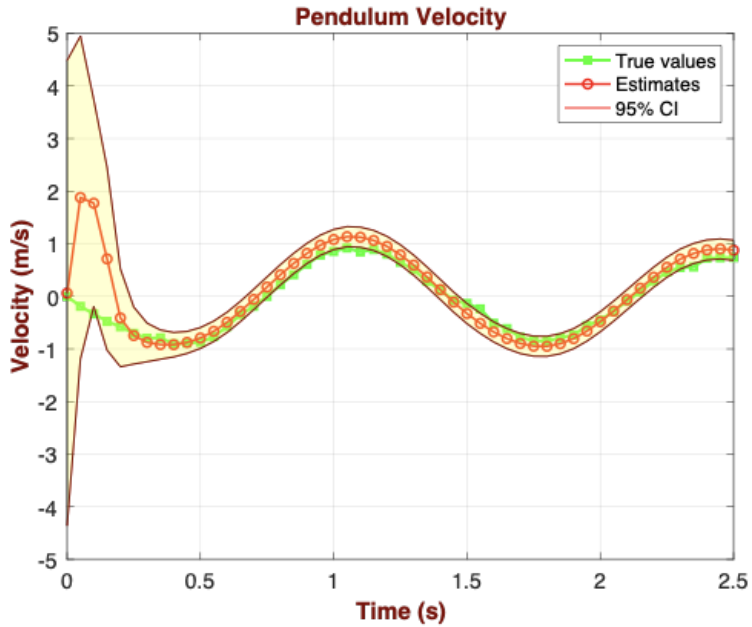


[Code: Non-linear KF/Ex14_UKF_for_Estimating_PendulumAngle.m]

Example 14 - Estimating the pendulum angle

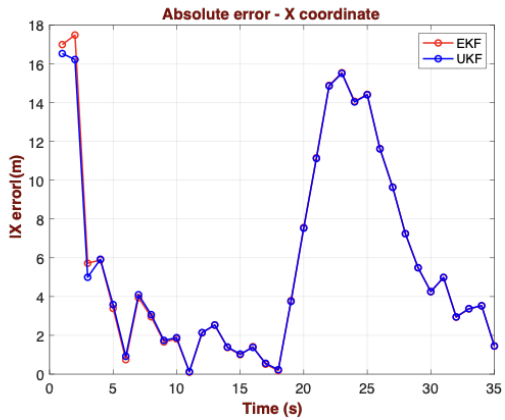
The results without the original UKF are as follows. There seems to be less uncertainty in pendulum angles estimates compared to the modified-UKF results.

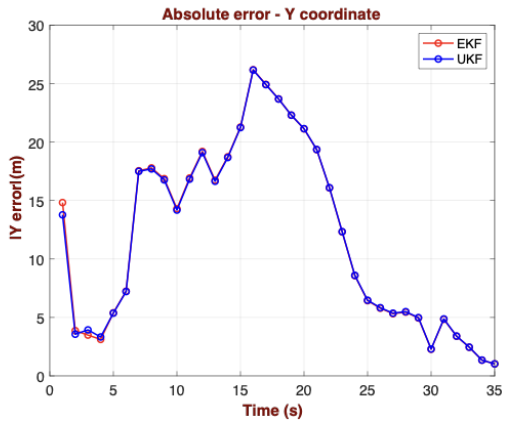


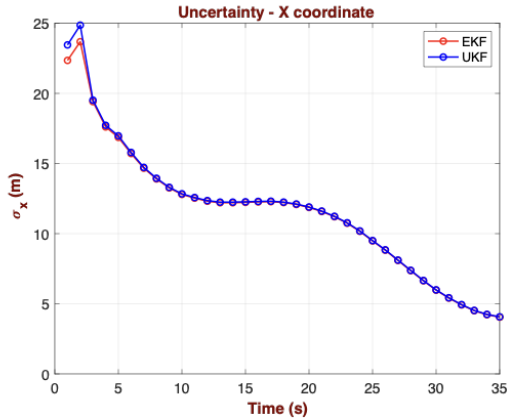


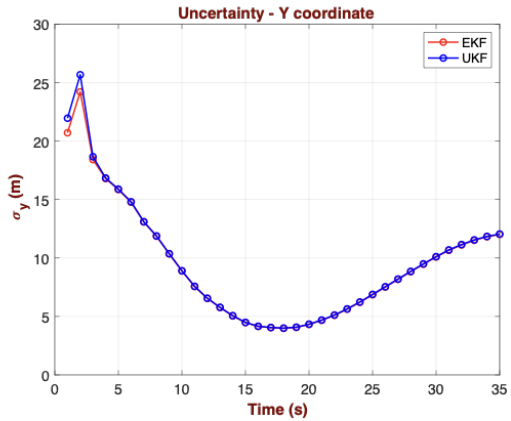
Non-linear filters comparison: EKF vs UKF

Example 11 (EKF) and Example 13 (UKF) dealt with vehicle location estimation using radar. Both examples have identical parameters: vehicle dynamics, radar, initialization, and measurements. `[Code:Non-linear KF/EKFvsUKF_VehileTrackingExamples.m]`







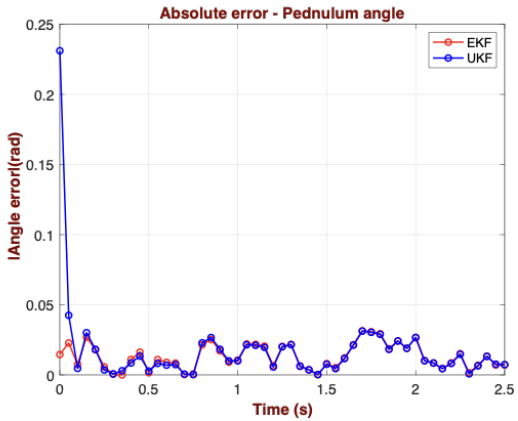


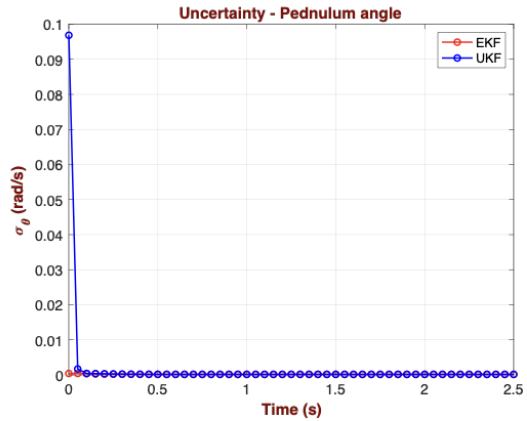
UKF performs slightly better for the first two iterations, but then the filters converge, and the performance of becomes identical.

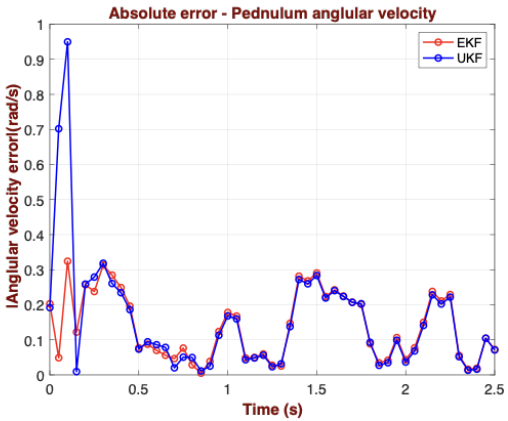
The estimation uncertainties are also identical for both filters.

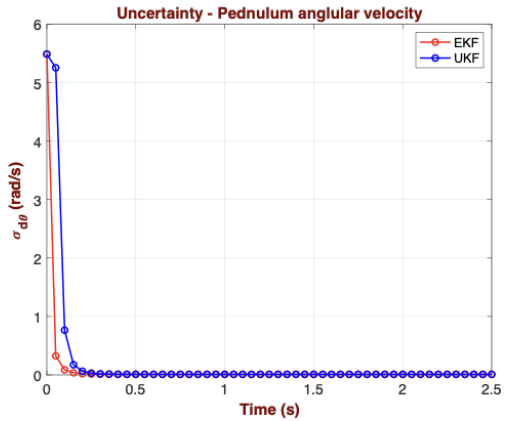
Example 14 - estimating the pendulum angle

Example 12 (EKF) and Example 14 (UKF) dealt the estimation of the pendulum angle and angular velocity. Both examples have identical parameters: pendulum dynamics, initialization, and measurements. `[Code: Non-linear KF/EKFvsUKF_PendulumExamples.m]`









During the first 0.5 seconds, the UKF error is significantly higher.

During the filter convergence period, the UKF uncertainty is higher than the EKF uncertainty.

Conclusions

- Extended and Unscented Kalman Filters perform a **linear approximation of the dynamic and state-to-measurement models**.
- The Extended Kalman Filter is a standard and most common technique used in non-linear estimation problems. Although the EKF became a standard for non-linear estimation, almost 60 years of EKF usage experience has led to a consensus that it is **unreliable for highly non-linear models**.
- The UKF is considered to be a better choice compared to EKF as demonstrated by examples in the original work [3] and [5].
- In the vehicle location examples, we have seen a similar performance of UKF and EKF. In the pendulum measurement example, the UKF was even better during the filter convergence period.
- Since both filters are sub-optimal, it is recommended testing both filters, and then choosing a better approach.
- **Particle Filter:** The particle filter uses using Monte-Carlo method technique for approximation. It is considered a precise technique; however, it often requires thousands of times more computations. One can find a good tutorial in [6], along with the code.

Part 4: Practical Guidelines for Kalman Filter Implementation

Including:

- Sensor fusion
- Variable measurement uncertainty
- Treatment of missing measurements
- Treatment of outliers
- Kalman Filter design process
  - Kalman Filter Initialization
  - KF Development Process

Sensor Fusion

Many practical systems are equipped with several complementary and sometimes interchangeable sensors measuring the same parameters, e.g.,

- A self-driving car has **Light Detection and Ranging (LiDAR)** and **radar** onboard. The LiDAR is much more precise than the radar. On the other hand, the radar measures velocity using the Doppler Effect, and its effective operational range is higher, especially in rain or fog conditions.
- The aircraft is equipped with Global Navigation Satellite System (GNSS) and Inertial Navigation System (INS) systems for navigation.
- Many surveillance systems include several radars for target tracking.

Using multiple sensors can significantly improve the state estimation precision in a process known as sensor fusion.
Sensor fusion refers to combining the measurements from multiple sensors resulting in joint information having less uncertainty than any of the sensors individually.

Combining measurements in one dimension

Consider two range measurements of the same target performed by two independent radars simultaneously. The SNR of the first radar measurement is higher (lower uncertainty) than the SNR of the second radar measurement.

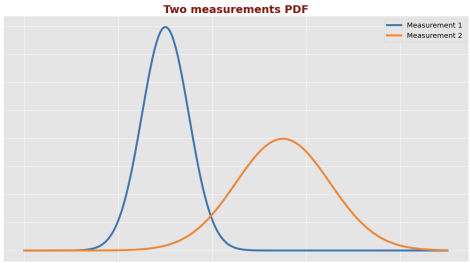


Figure: PDF of measurements of both radars as normally distributed random variables.

Since the measurements are independent, the joint PDF is a product of two PDFs:
The variance σ_{12}^2 is given by:

$$\sigma_{12}^2 = \left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right)^{-1}$$

The mean μ_{12} is given by:

$$\mu_{12} = \left(\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2} \right) \sigma_{12}^2$$

- Each measurement is weighted by the inverse of its variance. If the variance of the first measurement is lower than the variance of the second measurement ($\sigma_1^2 < \sigma_2^2$), then the weight of the first measurement is higher.
- That means the Gaussian PDFs product is closer to the measurement with lower variance.
- The variance of the Gaussian PDFs product is always lower than the variance of each measurement separately. Even a measurement with a very high variance contributes to the overall precision. If $\sigma_2^2 = \infty$ then $\sigma_{12}^2 = \sigma_1^2$.

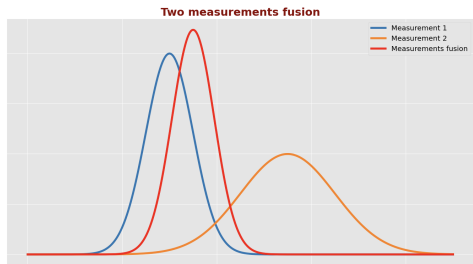


Figure: Joint PDF is closer to the measurement with a lower variance.

Combining n measurements

The product of univariate Gaussian PDFs results in a k-dimensional Gaussian PDF with the following properties:
The variance $\sigma^2_{1\cdots n}$ is given by:

$$\sigma^2_{1\cdots n} = \left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} + \cdots + \frac{1}{\sigma_n^2}\right)^{-1} = \left(\sum_{i=1}^{n} \frac{1}{\sigma_i^2}\right)^{-1}$$

Showing that every additional measurement minimizes the overall uncertainty.

The mean $\mu_{1\cdots n}$ is given by:

$$\mu_{1\cdots n} = \left(\sum_{i=1}^{n} \frac{\mu_i}{\sigma_i^2}\right) \sigma^2_{1\cdots n}$$

Combining measurements in k dimensions

The multivariate k-dimensional normal distribution is given by:

$$p(x) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

where:

- μ is the mean vector
- Σ is the covariance matrix

The product of n k-dimensional Gaussian PDFs is also a k-dimensional Gaussian PDF with the following properties:

$$\Sigma_{1\cdots n}^{-1} = \sum_{i=1}^{n} \Sigma_i^{-1}$$

$$\mu_{1\cdots n} = \Sigma\left(\sum_{i=1}^{n} \Sigma_i^{-1}\mu_i\right)$$

Like in the univariate Gaussian case, the mean of the product of the PDFs is the weighted mean of all PDFs, while the PDF with a lower variance has a higher weight.

Each PDF contributes to the joint PDF and reduces the joint PDF covariance.

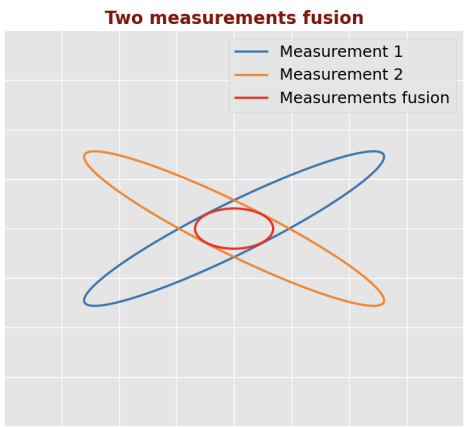


Figure: Two 2D measurements fusion: The blue and orange ellipses represent the covariance of two measurements, and the red ellipse represents the covariance of the joint PDF.

Sensor data fusion using Kalman filter: Measurements fusion

Kalman filter is the most widespread algorithm for multisensor data fusion. We discuss two methods for multisensor data fusion using the Kalman filter, assuming identical sensors' sample rates (for simplicity).

Method 1 – measurements fusion:

Measurement fusion is the most common sensor data fusion method. Typically it is used by a system that is equipped with several sensors.

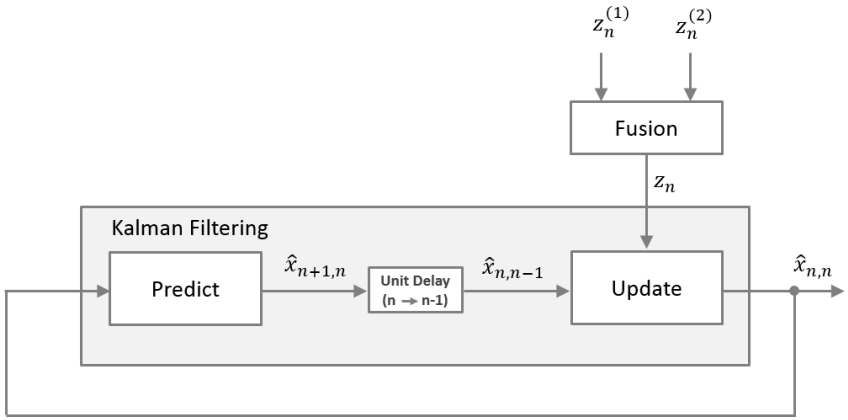


Figure: Sensors measurements fusion

Two or more sensors provide measurements $\boldsymbol{z}_n^{(1)}, \boldsymbol{z}_n^{(2)}, \ldots, \boldsymbol{z}_n^{(k)}$ with measurement covariance $\boldsymbol{R}_n^{(1)}, \boldsymbol{R}_n^{(2)}, \ldots, \boldsymbol{R}_n^{(k)}$. Assuming normally distributed measurement PDFs, we can calculate a joint PDF, with parameters:

$$\boldsymbol{R}_n^{-1} = \sum_{i=1}^{k} \boldsymbol{R}_i^{-1}$$

The unified measurement $\boldsymbol{z}_n$ is given by:

$$\boldsymbol{z}_n = \boldsymbol{R}_n \left(\sum_{i=1}^{k} \boldsymbol{R}_n^{(i)^{-1}} \boldsymbol{z}_n^{(i)} \right)$$

The conventional Kalman Filter receives the unified measurement $(\boldsymbol{z}_n, \boldsymbol{R}_n)$.

In the literature, you can also see that $\boldsymbol{z}_n$ for two sensors is calculated as follows:

$$\boldsymbol{z}_n = \boldsymbol{z}_n^{(1)} + \boldsymbol{R}_n^{(1)} \left(\boldsymbol{R}_n^{(1)} + \boldsymbol{R}_n^{(2)} \right)^{-1} \left(\boldsymbol{z}_n^{(2)} - \boldsymbol{z}_n^{(1)} \right)$$

It is identical to the Kalman Filter state update equation, which also calculates the fusion of two normally distributed PDFs – the measurement and the prior estimate uncertainty.

Second, it is computationally effective.

Sensor data fusion using Kalman filter: State fusion

Some applications involve different sensor systems that perform the same task. For example, several geographically distributed radars can track the same target. Each radar creates a separate target track using the Kalman filter.

An external system can receive radars' tracks and compute a unified track with higher location precision. **The state fusion method is also called the track-to-track fusion algorithm.**

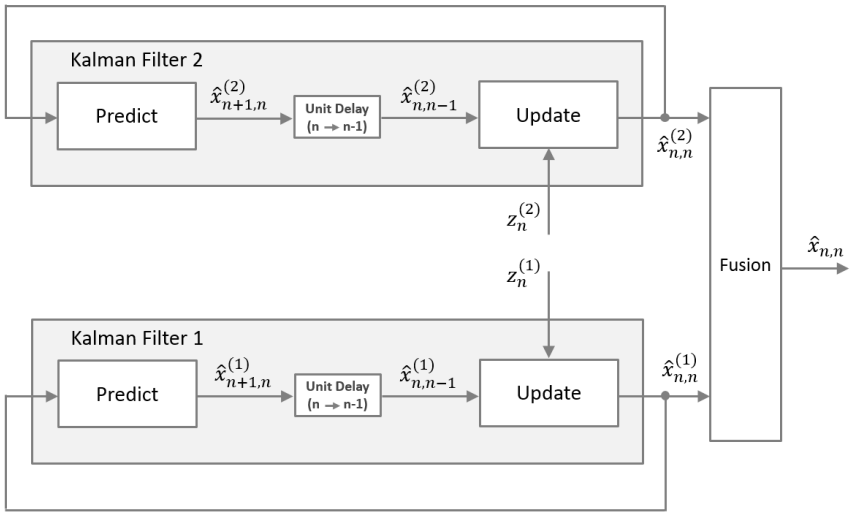


Figure: State fusion or track-to-track fusion.

State Fusion with Ignored Cross-Covariance: When the cross-covariance between the two track estimates can be ignored, the state fusion involves a simple fusion of two random variables $\hat{x}^{(1)}_{n,n}$ and $\hat{x}^{(2)}_{n,n}$ with estimate uncertainties $\boldsymbol{P}^{(1)}_{n,n}$ and $\boldsymbol{P}^{(2)}_{n,n}$:

$$\hat{x}_{n,n} = \hat{x}^{(1)}_{n,n} + \boldsymbol{P}^{(1)}_{n,n}\left(\boldsymbol{P}^{(1)}_{n,n} + \boldsymbol{P}^{(2)}_{n,n}\right)^{-1}\left(\hat{x}^{(2)}_{n,n} - \hat{x}^{(1)}_{n,n}\right)$$

The estimate covariance of the joint state $\hat{x}_{n,n}$ is:

$$\boldsymbol{P}^{-1}_{n,n} = \boldsymbol{P}^{(1)^{-1}}_{n,n} + \boldsymbol{P}^{(2)^{-1}}_{n,n}$$

textbfState Fusion with Cross-Covariance: The measurements from the two sensor tracks are not necessarily independent, as they can be correlated due to common process noise from the target dynamics. To avoid degradation in the target state estimates, these correlated estimation errors must be considered when combining the state estimates.

Sensor data fusion using Kalman filter: State fusion

The method for correlated state fusion was shown by Bar-Shalom [7]. The fused state is given by:

$$\hat{x}_{n,n} = \hat{x}_{n,n}^{(1)} + \left(\boldsymbol{P}_{n,n}^{(1)} - \boldsymbol{P}_{n,n}^{(12)}\right)\left(\boldsymbol{P}_{n,n}^{(1)} + \boldsymbol{P}_{n,n}^{(2)} - \boldsymbol{P}_{n,n}^{(12)} - \boldsymbol{P}_{n,n}^{(21)}\right)^{-1}\left(\hat{x}_{n,n}^{(2)} - \hat{x}_{n,n}^{(1)}\right)$$

$\boldsymbol{P}_{n,n}^{(12)} = \left(\boldsymbol{P}_{n,n}^{(21)}\right)^{T}$ is the cross-covariance matrix between $\hat{x}_{n,n}^{(1)}$ and $\hat{x}_{n,n}^{(2)}$. The cross-covariance matrix is given by:

$$\boldsymbol{P}_{n,n}^{(12)} = \left(\boldsymbol{I} - \boldsymbol{K}_n^{(1)}\boldsymbol{H}^{(1)}\right)\boldsymbol{F}\boldsymbol{P}_{n,n-1}^{(12)}\boldsymbol{F}^{T}\left(\boldsymbol{I} - \boldsymbol{K}_n^{(2)}\boldsymbol{H}^{(2)}\right)^{T} + \left(\boldsymbol{I} - \boldsymbol{K}_n^{(1)}\boldsymbol{H}^{(1)}\right)\boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^{T}\left(\boldsymbol{I} - \boldsymbol{K}_n^{(2)}\boldsymbol{H}^{(2)}\right)^{T}$$

where:

- $\boldsymbol{K}$ is the Kalman Gain
- $\boldsymbol{H}$ is the Observation Matrix
- $\boldsymbol{F}$ is the State Transition Matrix
- $\boldsymbol{Q}$ is the Process Noise Covariance Matrix
- $\boldsymbol{G}$ is the Control Matrix

Bar-Shalom and Campo showed [8] that when this dependence is taken into account, the area of the uncertainty ellipse is reduced by 70% instead of being cut by 50% as would be the case if the independent noise assumption were correct.

Multirate Kalman Filter

In many applications, the sensors' sample rates are not identical. For example, assume two sensors with a sampling rate of Δt and $3\Delta t$. The Kalman Filter update scheme is elaborated below.

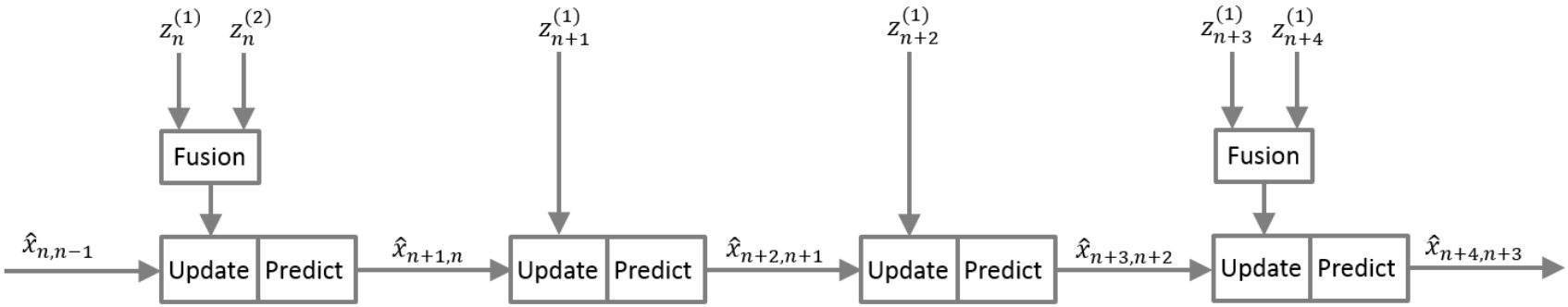


Figure: Multirate Kalman Filter

The first sensor measurement updates the Kalman Filter state every Δt period. Every $3\Delta t$ period, the measurements of the first and the second sensors are combined.

Variable Measurement Error

- In all examples, the measurement uncertainty is constant. For many measurement devices, the measurement uncertainty is a parameter given by the vendor. For example, the weight measurement accuracy of the scales or the power measurement accuracy of the power meter is a constant value.
- However, in many systems, the measurement uncertainty varies between successive measurements. For example, in a radar system, the range accuracy (standard deviation) is given by:

$$\sigma_R = \frac{c}{2B\sqrt{\text{SNR}}}$$

where:

- σ_R is the range measurement error (standard deviation)
- c is the speed of light
- B is the signal bandwidth
- SNR is the signal-to-noise ratio

The radar range measurement variance is:

$$r_n = \sigma_{R_n}^2 = \frac{c^2}{4B^2\text{SNR}_n}$$

We can see that the range measurement uncertainty depends on SNR. The SNR depends on many factors, such as interferences, target range, and perspective. For a radar system, the measurement uncertainty is not constant.

Treating missing measurements

Sometimes the sensor measurements are lost due to noisy communication channels, interferences, equipment malfunctions, or other anomalies. In such cases, you can set the measurement to an arbitrary value while setting the measurement variance to infinity.
The Kalman Gain would be zero:

$$\boldsymbol{K}_n = \frac{\boldsymbol{p}_{n,n-1}}{\boldsymbol{p}_{n,n-1} + \boldsymbol{r}_n} = \frac{\boldsymbol{p}_{n,n-1}}{\boldsymbol{p}_{n,n-1} + \infty} = 0$$

The current state estimation $\hat{x}_{n,n}$ would be equal to the prior estimate $\hat{x}_{n,n-1}$:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + 0(z_n - \hat{x}_{n,n-1}) = \hat{x}_{n,n-1}$$

The current state estimation variance $\boldsymbol{p}_{n,n}$ would be equal to the prior extrapolated state variance $\boldsymbol{p}_{n,n-1}$:

$$\boldsymbol{p}_{n,n} = (1 - 0)\boldsymbol{p}_{n,n-1} = \boldsymbol{p}_{n,n-1}$$

In case of missing measurement, the extrapolated state variance $\boldsymbol{p}_{n+1,n}$ would increase due to the process noise:

$$\boldsymbol{p}_{n+1,n} = \boldsymbol{p}_{n,n} + \boldsymbol{q}_n = \boldsymbol{p}_{n,n-1} + \boldsymbol{q}_n$$

Therefore, the missing measurement causes a temporary Kalman Filter divergence.

Treating outliers

Outliers are measurements outside an expected range or don't follow an expected pattern. Outliers can result from noisy communication channels, interferences, equipment malfunctions, recording errors, or other anomalies, such as radar signal reflection from a bypassing aircraft.

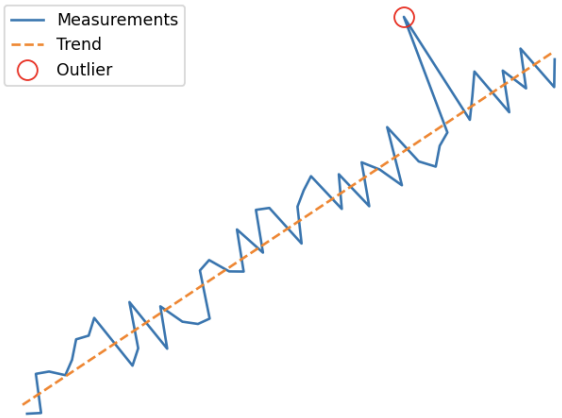


Figure: Outliers example

Identifying Outliers: We can divide outliers into two main categories:

1. Unlikely or unusual measurements: Sometimes you may decide that the measurement is unlikely based on your technical knowledge of the domain.

- If the vehicle speed measurement is 400km/h, you can decide that the measurement is an outlier since the car can't travel at that speed.
- If the vehicle position measurement is far from the road, it is likely an outlier.
- The water temperature above 100^oC is an outlier.

2. High statistical distance: Assume a system that estimates the vehicle speed using the Kalman Filter. The prior estimate of the speed (prediction) $\hat{x}n, n-1$ is 100km/h, and the measured speed z_n is 120km/h. The difference between the predicted and measured values is 20km/h. **Is it an outlier?**

- *Mahalanobis distance* is needed to answer this: The Mahalanobis distance is a measure of the distance between a point P and a distribution D or between two random variables, introduced by P. C. Mahalanobis in 1936.

Treating outliers

- In a single dimension, the Mahalanobis distance is given by:

$$d_M = \sqrt{\frac{(\hat{x}_{n,n-1} - z_n)^2}{p_{n,n-1}}}$$

where:

  - $\hat{x}_{n,n-1}$ is the prior estimate
  - $p_{n,n-1}$ is the prior estimate variance
  - z_n is the measurement

- For vehicle speed estimation example; if $p_{n,n-1} = 150\text{km}^2/\text{h}^2$:

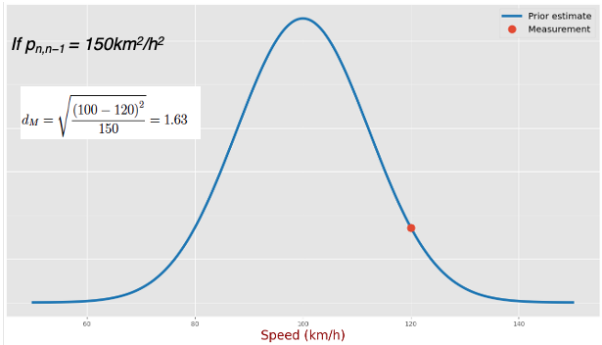


Figure: Low Mahalanobis Distance.

- if $p_{n,n-1} = 10\text{km}^2/\text{h}^2$:

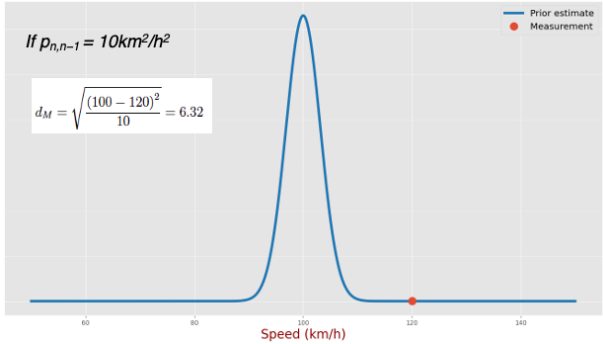


Figure: High Mahalanobis Distance.

- In the first case, the measurement is not an outlier since the distance of 1.63 standard deviations is reasonable. In the second case, the measurement is definitely an outlier.
- A Kalman Filter designer should define the Mahalanobis distance threshold based on the domain technical knowledge. Usually, it varies between 3 and 5.
- For a multivariate Kalman Filter, the d_M is:

$$d_M = \sqrt{(\hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{z}_n)^T \boldsymbol{P}_{n,n-1}^{-1} (\hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{z}_n)}$$

  - $\hat{\boldsymbol{x}}_{n,n-1}$ is the prior estimate
  - $\boldsymbol{P}_{n,n-1}$ is the prior estimate covariance matrix
  - $\boldsymbol{z}_n$ is the measurement vector

Treating outliers

Impact of Outliers:

- The impact of an outlier depends on the variance of the outlier. Let us continue with the example of vehicle speed estimation. The prior estimate of the speed (prediction) $\hat{x}n, n-1$ is 100km/h, and the measured speed z_n is 120km/h. The calculated Mahalanobis distance for $p_{n,n-1} = 10\text{km}^2/\text{h}^2$ is $d_M = 6.32$.
- The distance between the prior estimate and the measurement is 6.32 standard deviations; therefore, the measurement is an outlier.
- Let us take a look at the measurement uncertainty. The radar velocity measurement accuracy is given by:

$$\sigma_V = \frac{\lambda B}{2\sqrt{2\text{SNR}}}$$

  - σ_V is the velocity measurement error (standard deviation)
  - λ is the signal wavelength
  - B is the signal bandwidth

- The radar velocity measurement variance is:

$$r_n = \sigma_{V_n}^2 = \frac{\lambda^2 B^2}{8\text{SNR}_n}$$

- Assume a noisy measurement with a low SNR that yields $\sigma_{V_n} = 30$ km/h.

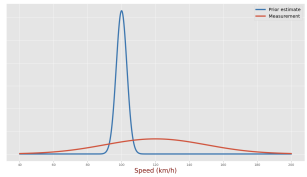


Figure: Abnormal measurement with high uncertainty

- The Kalman Gain is given by:

$$\boldsymbol{K}_n = \frac{\boldsymbol{p}_{n,n-1}}{\boldsymbol{p}_{n,n-1} + \boldsymbol{r}_n} = \frac{10}{10 + 30^2} = 0.01$$

- The state update equation is:

$$\begin{aligned}\hat{x}_{n,n} &= \hat{x}_{n,n-1} + 0.01(z_n - \hat{x}_{n,n-1}) \\ &= 0.99\hat{x}_{n,n-1} + 0.01 z_n\end{aligned}$$

- The Kalman Gain is very low; therefore, the impact of the noisy measurement with high uncertainty is very low.

Treating outliers

- Now assume an outlier measurement with high SNR that yields $\sigma_{V_n} = 2$ km/h. Such a measurement can be caused by an anomaly.

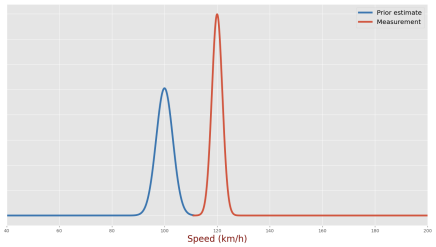


Figure: Abnormal measurement with low uncertainty

- The Kalman Gain is given by:

$$\boldsymbol{K}_n = \frac{\boldsymbol{p}_{n,n-1}}{\boldsymbol{p}_{n,n-1} + \boldsymbol{r}_n} = \frac{10}{10 + 2^2} = 0.71$$

- The state update equation is:

$$\begin{aligned} \hat{x}_{n,n} &= \hat{x}_{n,n-1} + 0.71(z_n - \hat{x}_{n,n-1}) \\ &= 0.29\hat{x}_{n,n-1} + 0.71 z_n \end{aligned}$$

- The Kalman Gain is very low; therefore, Kalman Gain is high; therefore, the impact of the abnormal measurement with low uncertainty is very high.

Treating Outliers:

- A high-impact outlier influences long-term filter stability. It influences the current estimation and the following estimations as following estimations are based on the measurements and past estimations.
- For an outlier with a high Mahalanobis distance, eliminate the outlier and treat it as a missing measurement.
- If the outlier is unlikely or unusual, but the Mahalanobis distance is low, consider changing the outlier value to a reasonable value. E.g., when estimating the water temperature, set a temperature bound to 0^oC and to 100^oC. If the measurement temperature is 101^oC, change it to 100^oC.
- The outlier treatment algorithm includes the following steps:
  - Identify an outlier by calculating the Mahalanobis Distance or based on the domain knowledge.
  - Estimate the outlier impact.
  - Eliminate the outlier or change its value to a lower or upper bound.

Kalman Filter Initialization

The Kalman Filter must be initiated with a prior estimation $\hat{x}_{0,0}$ and initialization covariance $P_{0,0}$.

Linear KF: In Example 9–vehicle location estimation, with true vehicle location at the time $t = 0$ as $x = 300, y = -425$, what if we change initialization as

$$\hat{x}_{0,0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow = \begin{bmatrix} 303 \\ 0 \\ 0 \\ -428 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

i.e., we initiate the Kalman Filter close to the actual vehicle position.

Since our initial state vector was a guess, we set a very high estimate uncertainty. Now to see the effect of initialization, we must reduce the estimate uncertainty. The high estimate uncertainty results in a high Kalman Gain by giving a high weight to the measurement.

[Code: Non-linear KF/Ex11_InitializationAspects.m]

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}$$

i.e., reduce to 50 from 500.

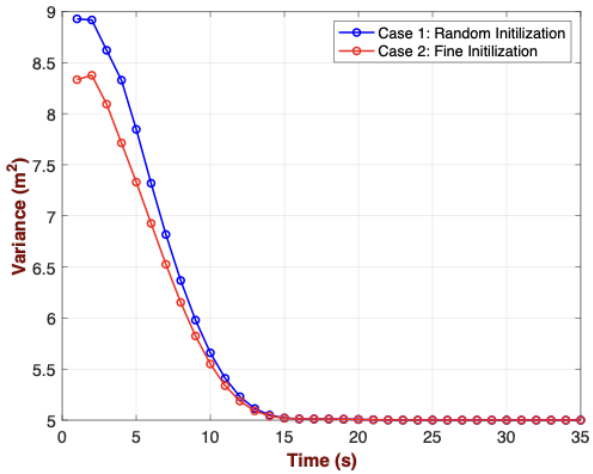


Figure: LKF uncertainty: rough vs. fine initiation.

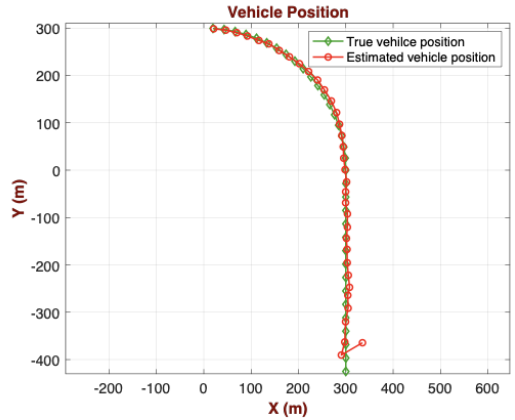
We can see a lower uncertainty of the fine initialization. The KF converges faster if we initiate it with meaningful values.

Kalman Filter Initialization

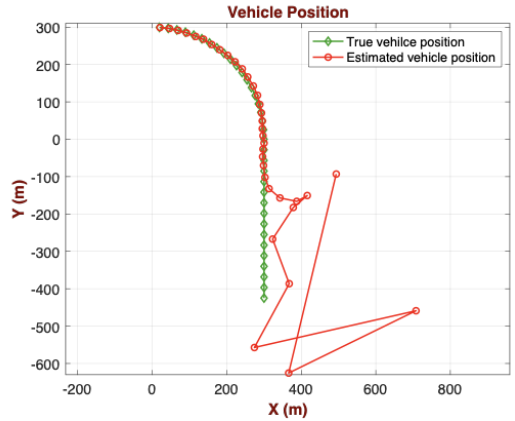
Non-linear KF initialization:
Example 11–vehicle location estimation using radar. If the EKF is initiated close to the true position, the EKF converges quickly and accurately tracks the target.

$$\boldsymbol{x}_{0,0} = \begin{bmatrix} 100 \\ 0 \\ 0 \\ -100 \\ 0 \\ 0 \end{bmatrix} \rightarrow = \boldsymbol{x}_{0,0} = \begin{bmatrix} 100 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$



If the initialization point is moved further, the EKF provides wrong estimations. After some time, it converges and tracks the target.



Unlike the LKF, the non-linear Kalman Filter requires fine initiation; otherwise, it wouldn't be able to provide satisfactory results.

KF initialization techniques

- There is no generic initialization technique or method. It depends on the system.
- For example, the radar search process provides coarse target measurement used as initialization for the tracker.
- For other systems, an educated guess is sufficient for KF initialization.
- You can also use the **first measurement as initialization data** and start the estimation process with the second measurement.

KF Development Process

The Kalman Filter development process includes four phases:

- Kalman Filter Design
- Simulation
- Performance Examination
- Parameters Tuning

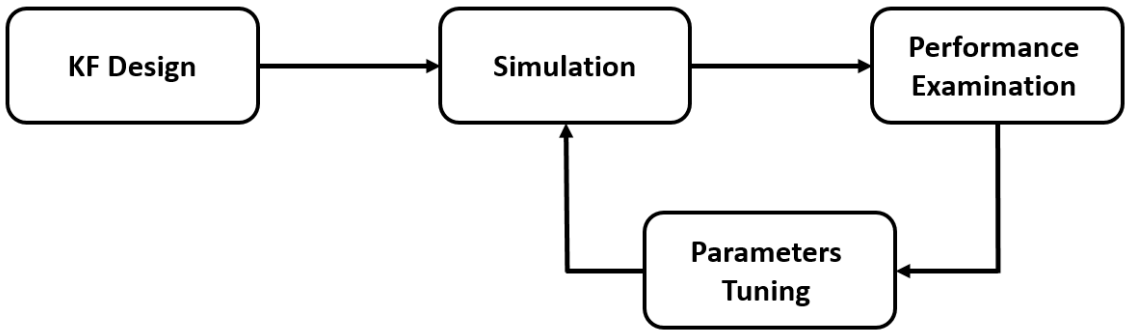


Figure: KF Design Process

KF Development Process: Follow 6-Steps of Kalman Filter Design

Step 1: Define the System Dynamic Model.

- Lucky if you know your system's Dynamic Model (state extrapolation equation).
- Otherwise, write down the differential equations that govern your system (the state space representation), as:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)$$

- Solve the equations to determine the state extrapolation equation.

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n}$$

- For a non-linear system, the differential equation is in the form of:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t))$$

- Solve the differential equation to determine the state extrapolation equation:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n}$$

- If the differential equation cannot be solved, perform the numerical integration of:

$$\boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) = \hat{\boldsymbol{x}}_{n,n} + \int_{t_n}^{t_{n+1}} \boldsymbol{g}(\boldsymbol{x}(t))\, dt$$

Step 2: Define the measurement equation.

- For LKF: $\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n$
- For EKF: $\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n)$

Step 3: Process Noise error sources:

- Write down a list of all the error sources.
- Define the Process Noise matrix Q based on uncorrelated error sources (white noise).
- The correlated noise sources must be a part of the Dynamic Model.

Step 4: Decide the KF initialization method:

- Initialization method is system-dependent; it can be an educated guess, a coarse or first measurement.
- Decide initialization covariance $\boldsymbol{P}_{0,0}$.

Step 5: Add treatment for missing measurements.

Step 6: Decide outliers treatment method:

- Which measurement should be considered an outlier and how to treat it?

KF Development Process: Simulation

Simulation is a critical phase of the Kalman Filter development process. It allows to observe the filter performance in a controlled environment. The simulation is a computer program that contains the following modules:

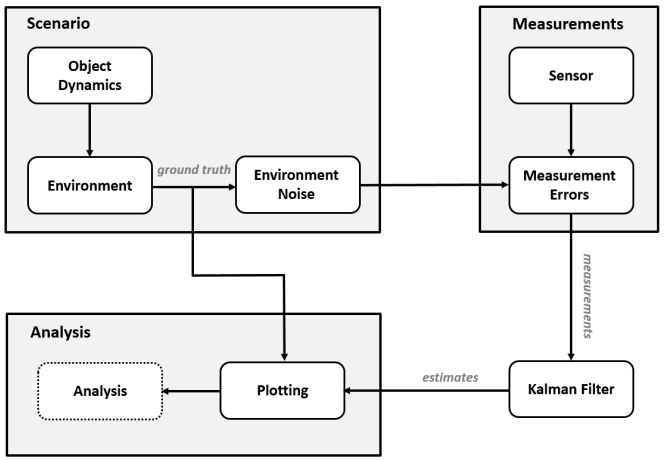


Figure: KF Simulation Flow

Scenario Module: Simulating the real-world

- **Object dynamics** module generates different dynamics scenarios of the object of interest.
  - For example, the vehicle trajectory, the pendulum position, or the heating liquid temperature.
  - It shall include possible extreme conditions, like sharp target maneuvers.
- **Environment** module simulates the environmental influence on the target dynamics, like air turbulence influence on the aircraft, the icy road influence on the vehicle, and the air-conditioner influence on the liquid temperature.
- **Generated scenario (or output):**
  - The generated scenario is a true state vector or the **ground truth**.
  - The environment noise module adds noise to the ground truth - for example, an aircraft radar cross-section fluctuations, radio interferences, etc.

Measurements Module: sensor measurements as input to Kalman Filter.

- For simple sensors like scales with a measurement uncertainty σ, generate a normally distributed random noise with std. σ, and add it to the generated scenario.

KF Development Process: Simulation

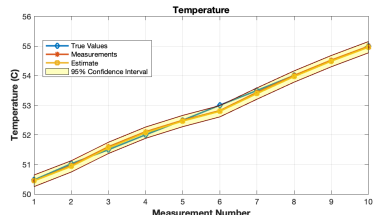
- For EKF, the ground truth should be transformed into the sensor coordinates. For example, the vehicle position coordinates are X and Y while the sensor coordinates are range (r) and the bearing angle (θ). Then the range noise and the angle noise should be added to the transformed coordinates.
- More sophisticated sensors require simulation of the sensor system to simulate phenomena like clock drift, imperfections due to sampling, and other system-specific aspects like beam broadening in radar.

KF Module: Same as KF design process.
Analysis: Visualization of results as

- Estimates vs. measurements vs. ground truth.
- If possible, add estimation confidence intervals or confidence ellipses to the plot.
- Estimation uncertainty vs. time
- Kalman Gain vs. time

KF Development Process: Performance Examination

- Before examining the KF performance, you should define the acceptable (reasonable) performance criteria. Don't expect low errors if your sensor is not accurate or environmental noise is too high.
  - The Root Mean Square (RMS) error between the ground truth and estimates.
  - The RMS error between the ground truth and predictions.
  - The maximum error between the ground truth and estimates.
  - The maximum error between the ground truth and predictions.
  - The filter convergence time.
  - Prediction and estimation uncertainties.
  - The confidence levels.
- If the KF errors are too high, examine your dynamic model. You can increase the process noise ($\boldsymbol{Q}$) value. However, improving the dynamic model is a better approach.
- Examine the filter convergence. The Kalman Gain should constantly decrease until it reaches a certain floor level. As a result, the uncertainty of the estimates should also decrease. If the convergence time is too high, examine the influence of the KF initialization on the convergence time. Find methods for finer initialization values. You can also decrease the process noise ($\boldsymbol{Q}$) value, but make sure it doesn't increase the errors.
- Consider the results of Example 8.



- Assume that 100% of the ground truth is within the confidence intervals. Is it good? Not necessarily. Your KF estimates uncertainty is too high. You can achieve lower uncertainty estimates without compromising the confidence level.

References I

[1] H. Sandberg, *Kalman and the filter*, Available: https://youtu.be/ZwADI3PX0cA, 1984.

[2] K. O. Arras, "An introduction to error propagation: Derivation, meaning and examples of equation $Cy = FxCxFx^T$," *EPFL-ASL-TR-98-01 R3*, 1998, Available: https://doi.org/10.3929/ethz-a-010113668.

[3] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, Spie, vol. 3068, 1997, pp. 182–193.

[4] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

[5] R. Van Der Merwe, *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. Oregon Health & Science University, 2004.

[6] J. Elfring, E. Torta, and R. Van De Molengraft, "Particle filters: A hands-on tutorial," *Sensors*, vol. 21, no. 2, p. 438, 2021, Code: https://github.com/jelfring/particle-filter-tutorial.

[7] K. Chang, R. Saha, and Y. Bar-Shalom, "On optimal track-to-track fusion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 4, pp. 1271–1276, 1997.

References II

[8] Y. Bar-Shalom and L. Campo, "The effect of the common process noise on the two-sensor fused-track covariance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-22, pp. 803–805, 1986.