# To do:

* Predictor should be able to call something from the Simulator to start running the whole true trajectory of the satellite. This is to help us compare the true trajectory with the rough trajectory used for Kalman Filter prediction step.
* Predictor team should work with Simulator team to design the process model F (a 6x6 matrix).
* Predictor team to design the measurement function H which converts state space into measurement space.
* Predictor team to design the measurement noise R which takes radar accuracy from different radars into account.

# Assumptions

Radar reading noises:

* Up & down: ~N(0, precision1)
* Left & right: ~N(0, precision2)
* (do we need to collar and cap these errors to avoid the radar reading to become out of range?)

# Jargons:

The system is what we are trying to model: satellite.

The state is the current configuration of it: satellite’s position. (we don’t know the actual state. But our filters produce the estimated state of the system.

One cycle of prediction and updating with a measurement: system propagation or time evolution. Refers to how the state of the system change over time.

# Steps:

1. Belief: posterior from time t-1 on the current state
2. Prior (or predicted belief): At time t, system moves, what do I expect the state (position) to be: prior at time t on the state after movement.
3. Posterior (or updated belief): get measurement, update on the prior for the state at time t. = belief from time t.

In the context of the project. At time t-1, I have a belief (distribution) on the position of the satellite. At t, the satellite moves. Using my knowledge of physical laws and the system, I have a prior distribution (wide) on the position of the satellite. I then receive a radar measurement at time t, I use this to compute the likelihood of this measurement given each position, and use this to update on my prior to give me a posterior distribution (narrower) on the satellite position at time t, which is my new belief (distribution) for the satellite position.

**Revised:**  
In the context of the project, at time t−1, I have a belief (a probability distribution) over the possible positions of the satellite. Between t−1 and t, the satellite moves. Using my knowledge of physical laws and system dynamics, I apply a motion model to my previous belief to produce a prior distribution at time t — typically a wider distribution reflecting uncertainty in motion. Then, at time t, I receive a radar measurement. I compute the likelihood of receiving this measurement given each possible satellite position, and use it to update the prior via Bayes' rule. This results in a posterior distribution at time t, which is my updated belief about the satellite's position — typically more concentrated than the prior.

adapt to satellite tracking using the dog-hallway example.

(kernel in the example is to be replaced by a physical model for the satellite’s motion)

A close-up of a text

AI-generated content may be incorrect.

# What is Kalman filter:

Kalman filter is - a Bayesian filter that uses Gaussians.

# What does Predictor need:

* Position of satellite (distance, elevation, azimuth) ~~(x,y,z)~~
* timestamp of the reading (2025-04-20 10:54:58.183207 year, month, day, hour, minute, second, and microsecond)
* which radar took the measurements (and radar precisions on the position and velocity) (range\_acc, angular\_acc, velocity\_acc)
* a blackbox model for the system evolution

predict satellite position in cart coord.

get measurement from radar in spherical coord.

Convert predicted satellite position into spherical coord.

Update satellite position & velocity (state) (???) in spherical coord.

Convert updated state into cart coord.

# Notes from the book:

Sum of 2 Gaussians is another Gaussian:

A math symbols and symbols

AI-generated content may be incorrect.

Product of 2 Gaussians is another Gaussian:

A math equations with numbers and symbols

AI-generated content may be incorrect.

Mean can be re-written as a weighted sum of the measurement mean and process model mean.

A group of symbols with numbers

AI-generated content may be incorrect.

The weights sum up to 1 (obvious). Re-write the mean as:

A math equations on a white background

AI-generated content may be incorrect.

K = Kalman gain. A weighting factor. Used for choosing a value between process model mean (prior mean) and measurement mean (likelihood (measurement given state) mean) (how much weight to place on the measurement).

For variance of posterior:

A math equations and formulas

AI-generated content may be incorrect.

Posterior variance = K\* measurement(likelihood) variance (1D only) = (1-K) \* prior variance.

A black background with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

* z = measurement
* P = prior variance after prediction (or posterior variance after update) (or covariance matrix)
* Q = process noise
* R = measurement noise (or likelihood variance)

A graph with a red line

AI-generated content may be incorrect.

Generic algorithm

**Initialization**

1. Initialize the state of the filter

2. Initialize our belief in the state

**Predict**

1. Use system behavior to predict state at the next time step

2. Adjust belief to account for the uncertainty in prediction

**Update**

1. Get a measurement and associated belief about its accuracy

2. Compute residual between estimated state and measurement

3. Compute scaling factor based on whether the measurement

or prediction is more accurate

4. set state between the prediction and measurement based

on scaling factor

5. update belief in the state based on how certain we are

in the measurement

Likelihood = gaussian(measurement, radar\_accuracy)???

“In practice we would likely assign the first measurement from the sensor as the initial value, but you can see it doesn't matter much if we wildly guess at the initial conditions - the Kalman filter still converges so long as the filter variances are chosen to match the actual process and measurement variances.” --- Chapter 04

Velocity covariance with position???

“multivariate Gaussians express the correlation between multiple random variables, such as the position and velocity of an aircraft.

Correlation between variables drastically improves the posterior.” --- Chapter 05

## Chapter 06 – multivariate Kalman filter

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a math problem

AI-generated content may be incorrect.

Your job as a designer will be to design the state (**x,P**), the process (**F,Q**), the measurement (**z,R**), and the measurement function **H**.

If the system has control inputs, such as a robot, you will also design **B** and **u**.

“Kalman filter computes estimates for hidden variables” (e.g. velocity if radar only measures position but doesn’t measure velocity) --- chapter 06

“In the last chapter we showed that the position and velocities are correlated. But how correlated are they for a dog? I have no idea. As we will see the filter computes this for us, so I initialize the covariances to zero. Of course, if you know the covariances you should use them.”

“Our job as Kalman filters designers is to specify F such that  performs the innovation (prediction) for our system. To do this we need one equation for each state variable.”

A white background with black text

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

**F** is called the *state transition function* or the *state transition matrix. A square matrix*.

**Q** equals the expected value of the white noise **w**.

Update:

Residual = measurement – predicted state

We need to convert the satellite position (x,y,z) into a radar measurement (distance, elevation, azimuth) so we can perform the subtraction.

The Kalman filter generalizes this problem by having you supply a *measurement function* that converts a state into a measurement.

Why not work in state space by converting the spherical coord into cartesian coord, allowing the residual to be a difference in cartesian coord?

We cannot do that because most measurements are not *invertible*. The state for the tracking problem contains the hidden variable velocity. There is no way to convert a measurement of position into a state containing velocity. On the other hand, it is trivial to convert a state containing position and velocity into a equivalent "measurement" containing only position. We have to work in measurement space to make the computation of the residual possible.

A black text with a white background

AI-generated content may be incorrect. = initial prior uncertainty before prediction

A black text with a plus and a black text

AI-generated content may be incorrect.= prior uncertainty after prediction

A black text on a white background

AI-generated content may be incorrect. = *system uncertainty* or *innovation covariance =*  total uncertainty used to calculate Kalman gain (remember in the 1D case, Kalman gain = prior variance / (prior variance + likelihood (or measurement) variance). This S is the denominator in that.

A black text with black letters

AI-generated content may be incorrect. = weighting factor that determines how far to shift towards the measurement from the prediction. **It’s a vector (if measurement is 1D) or matrix (if measurement is n-D) in the multivariate case, each entry is a weighting factor for each state**, but a scalar real number in the univariate case between 0 and 1.

A black text and black letters

AI-generated content may be incorrect. = difference between the measurement and current state in measurement space.

A math equation with black text

AI-generated content may be incorrect.= converts the residual back to state space (embedded in K using ), scales residual, and update the current state to the next state.

A close-up of a black text

AI-generated content may be incorrect.= variance of the posterior

All steps:

A math equations and formulas

AI-generated content may be incorrect.

“A rule of thumb for Q is to set it between ½ ∆a to ∆a, where ∆a is the maximum amount that the acceleration will change between sample periods.

This only applies for the assumption we are making in this chapter - that acceleration is constant and uncorrelated between each time period.” ???

“To some extent you can get similar looking output by varying either R or Q, but I urge you to not ‘magically’ alter these until you get output that you like. **Always think about the physical implications of these assignments, and vary R and/or Q based on your knowledge of the system you are filtering. Back that up with extensive simulations and/or trial runs of real data**.”

When **R** **is very large** we are telling the filter that there is a lot of noise in the measurements. In that case the Kalman gain K is set to favour the prediction over the measurement, and the prediction comes from the velocity state variable. Thus there is a **large correlation between x and .** Conversely, if **R is small**, we are telling the filter that the measurement is very trustworthy, and K is set to favour the measurement over the prediction. Why would the filter want to use the prediction if the measurement is nearly perfect? If the filter is not using much from the prediction **there will be very little correlation reported**.

Large correlation means the eclipse is slanted and narrow. Low correlation means the eclipse is either not slanted, or not narrow.

A white background with black text

AI-generated content may be incorrect.

**Batch processing**

The Kalman filter is designed as a recursive algorithm - as new measurements come in we immediately create a new estimate. But **it is very common to have a set of data that have been already collected which we want to filter**. **Kalman filters can be run in a batch mode, where all of the measurements are filtered at once**. We have implemented this in **KalmanFilter.batch\_filter().**

**A screen shot of a computer code

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

A computer code with numbers and letters

AI-generated content may be incorrect.

**Smoothing**

“the RTS smoother's output is much smoother than the KF output. Second, it is almost always more accurate than the KF output.” Try this!!!

“We made a key insight: hidden variables have the ability to significantly increase the accuracy of the filter. This is possible because the hidden variables are correlated with the observed variables.”

“If you do not have a velocity sensor and yet are estimating velocity, you will need to test that the velocity estimates are correct.; do not trust that they are.”

“Initialization poses a particularly difficult problem for hidden variables. If you start with a bad initialization the filter can usually recover the observed variables, but may struggle and fail with the hidden one.”

## Chapter 07 – Kalman filter maths

**Control inputs**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

( = velocity)

A screenshot of a computer

AI-generated content may be incorrect.

Model a system with a set of nth-order differential equations. Convert them into an equivalent set of first-order differential equations. Put them into the vector-matrix form used in the previous section: . Once in this form we use of of several techniques to convert these linear differential equations into the recursive equation:

It’s a bit difficult to obtain F. It’s derived from the A.

How? 🡺 (F is called the matrix exponential)

But we don’t need to compute this by hand! Python does it (solves the matrix exponential):

A screenshot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

### Design the process noise matrix Q (most difficult part):

A text on a page

AI-generated content may be incorrect.

Process model:

w = process noise

I’m lost from here…

**Model 1: Continuous White Noise Model:**

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a math problem

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

… (check the lecture)

**Model 2: Piecewise White Noise Model**

… (check the lecture)

**How to set the noise in the 2nd model:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**But do I need to understand this? --- probably not because:**

**there’s a simplification of Q that most entries would have noise close to 0. We can just give positive values to the most rapidly changing term as an approximation of the process noise matrix. If our state is (x,y,z,vx,vy,vz), then Q is 6x6. The elements for vx, vy, vz will be set to non-zero in Q.**

**Stable Computation of the Posterior Covariance**

**A white background with black text

AI-generated content may be incorrect.**

**A close up of a text

AI-generated content may be incorrect.**

After the measurement update, covariance P is as that in the Joseph form. The trace (sum of diagonal entries) of the covariance matrix P (of posterior in Joseph form) tells the total uncertainty in all my states. For the filter to be as accurate as possible, we want to minimise the trace of P after the update. So we want to find K that minimises tr(P). we do this by taking the derivative of tr(P) wrt K, set this to 0, then solve for K. the optimal K must satisfy which is the usual formula for Kalman gain. So the Kalman gain is the mathematically optimal choice to minimise total expected error under assumptions of Gaussian noise, linear dynamics etc. It explains why Kalman filter is the best possible linear unbiased estimator.

**Bayesian filtering**

**A screenshot of a text

AI-generated content may be incorrect.**

## Chapter 08 – designing KF

Plot the position residual (trajectory from KF vs true trajectory) for a long period of time to check if the KF is diverging or converging.

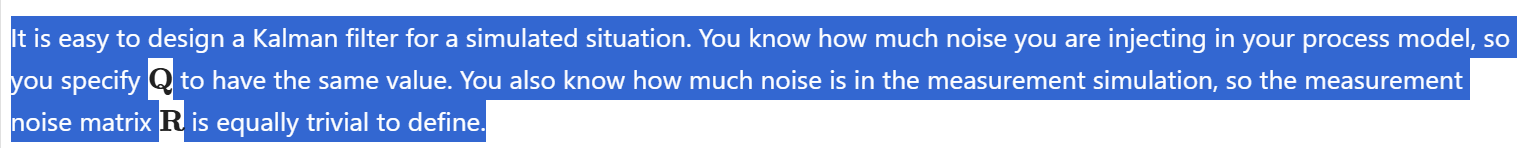
A text on a page

AI-generated content may be incorrect.

If the system has changing acceleration, having 2nd order filter ( is more accurate a more accurate filter. But it’s okay to use a 1st order filter if we increase the process noise to an appropriate level (need to experiment).

A blue and white text

AI-generated content may be incorrect.



### Evaluating KF performance

A blue and white text box

AI-generated content may be incorrect.

Note!!! The “estimated state” means the updated state instead of predicted state.

Calculate NEES (normalised error) whenever possible to measure filter’s performance.

A close-up of a document

AI-generated content may be incorrect.

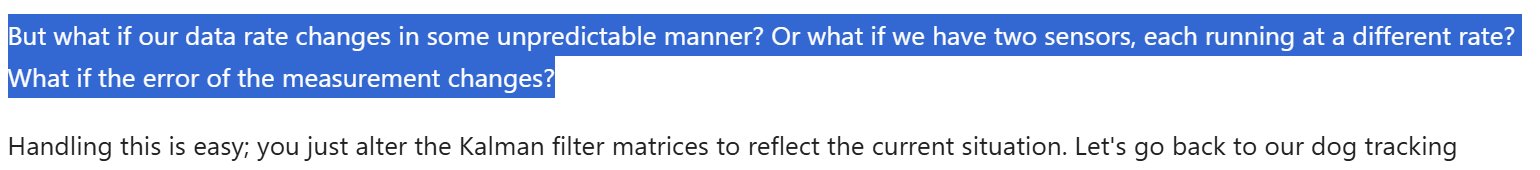
A math equations on a white background

AI-generated content may be incorrect.

typical control inputs are changes to steering angle and changes in acceleration. This introduces nonlinearities which we will learn to deal with in a later chapter.

**Sensor fusion: combine multiple radar measurements.**

### Nonstationary Processes (e.g. different dt)

****

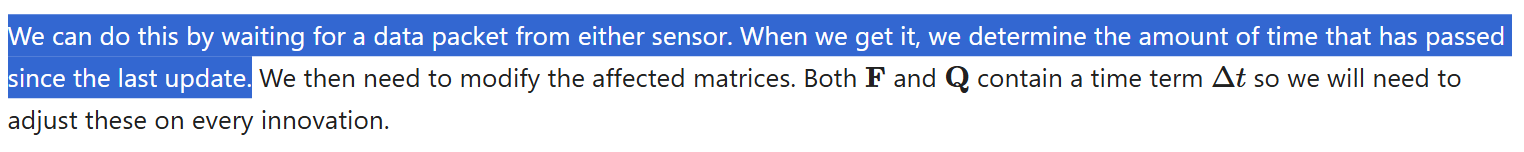
**A screenshot of a math problem

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.** **A number and numbers in a box

AI-generated content may be incorrect.**



How to wait for a data packet from any radar?

## Chapter 09 – nonlinear filtering

Unscented KF is much more accurate than extended KF, and easier to implement. Do UKF first!. But it’s slower.

## Chapter 10 – Unscented KF

the **true output distribution is non-Gaussian**.

**Why they still use a Gaussian approximation**

Despite this non-Gaussian result:

* Filters like the **UKF** or **EKF** **approximate the result as a Gaussian** anyway.
* They compute:
  + The mean of the transformed points.
  + The covariance of the transformed points.
* Then they **pretend** that this mean and covariance describe a Gaussian — even though the true shape might not be.

This is necessary because:

* The Kalman filter framework assumes Gaussians (for linearity and tractability).
* It needs to **update the state estimate and uncertainty using a Gaussian representation**.

So what the author means:

"Even though the output distribution isn't really Gaussian, we **approximate** it with a Gaussian by matching the mean and standard deviation computed from the transformed samples."

what would be fewest number of sampled points that we can use?

* The fewest number of points that we can use is one per dimension (for the linear case).
* Non-linear case: minimum 3 points per dimension

A text on a white background

AI-generated content may be incorrect.

A math equations and formulas

AI-generated content may be incorrect.

What does the *unscented transform do?*

* Pass the sigma points through a nonlinear function (process model) which gives the transformed set of points
* Computes the mean and covariance of the transformed points, which becomes the new estimates
* In short, the unscented transform takes points sampled from some arbitrary probability distribution, passes them through an arbitrary, nonlinear function and produces a Gaussian for each transformed points.

### Prediction step

* Generate sigma points & weights for mean and covariance of the distribution of the initial state
* Transform the sigma points using the non-linear function (process model) f(x, dt)
* Use unscented transform to compute the mean and covariance of the transformed sigma points. This forms our prior (prediction step completes)

A math equation with black text

AI-generated content may be incorrect.

### Update step

* Convert the sigma points from the prior into measurement space using h(x)
* Compute the mean and covariance of these converted sigma points using unscented transform

A black text on a white background

AI-generated content may be incorrect.

(note: is the mean of the prior, i.e. the prediction, after the raw prediction is converted to measurement space)

* Take measurement z. compute the residual (y = z - )
* Calculate Kalman gain
* Update state
* Compute covariance of the updated state

A black and white text

AI-generated content may be incorrect.

### Comparison: KF vs UKF

A table with black and white text

AI-generated content may be incorrect.

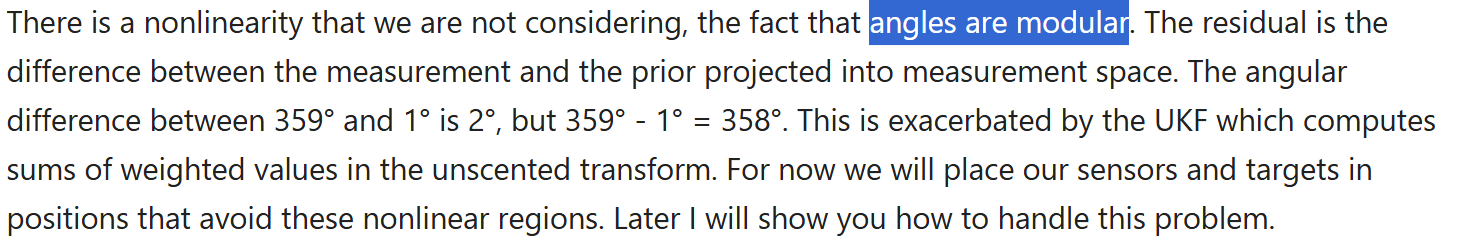
### Method to select sigma points

Van der Merwe's

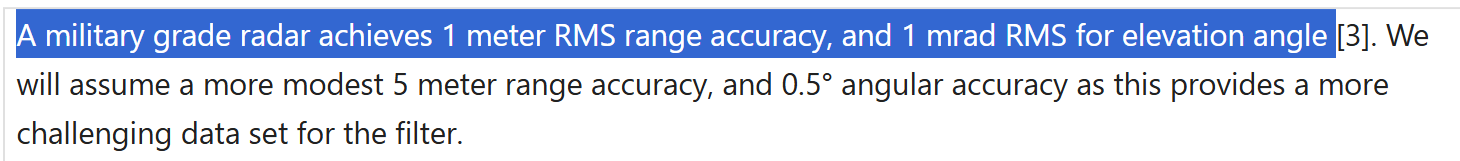
* α: controls the spread out and weights of the points. Higher α => more spread & more weight on the centre point (mean). Between 0 and 1.
* β: 2 is good. Dunno what it does…
* κ: 3-#dim = 3-n

### Tracking aircraft

Problem with angles



What radar precision to use



# Extensions in KF

* Smoothing the filtered trajectory (chapter 13)
* Batch process on many measurements (chapter 6?)
* Disregard bad measurement (gating. Chapter 8)
* Pruning? (chapter 8) particle filter (chapter 12)
* Adaptive filtering (chapter 14)
* Sensor fusion (use multiple radar measurements for the same position in the update)
* Add drag coefficient & mass as 2 extra dimensions in the KF state.

Ask Peter:

* In the simulation of the satellite, should the true trajectory be non-noise or add some noise to it so every time we simulate the result is slightly different.
* Should we use UKF or EKF? Is UKF too slow?