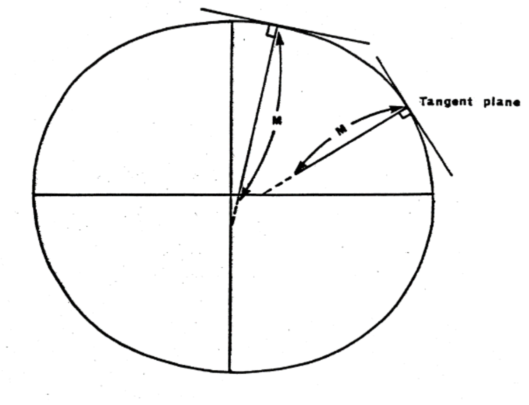
Complete report of Sensor fusion and data acquisition in python

GPS: The NMEA 0183 standard helps devices like GPS receivers talk to each other. An example of an NMEA sentence tells us where something is. It starts with a dollar sign and has letters like 'GGA' to show what kind of data it is. Then it tells the time in a special format and gives the latitude and longitude of the location. It also tells how good the location is, how many satellites it's using, and how high above sea level it is. There is even a number to check if the data is correct. So, this sentence holds all the important details about a place in a way that devices can understand and use.

```
$GPGGA,123519, 27.62195972, N,85.53777239E,14,08.9,545.4, M,46.9, M
```

From above NMEA data the latitude longitude and altitude are extracted, the message format provides 27.62195972-degree latitude(φ) ,85.53777239-degree longitude(and 1408.9M altitude from GPS.



As our vehicle move in local tangent plane, we need to calculate cartesian coordinate plane using WSG84 earth model.

$$e^2 = \frac{a^2 - b^2}{a^2}.$$

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2(\phi)}}$$

$$M = \frac{a(1 - e^2)}{\sqrt{1 - e^2 \sin^2(\phi)}}$$

- Where M is Meridian Radius of Curvature and N is prime vertical radius of curvature
- $a \approx 6378137$ m is the semi-major axis of the Earth ellipsoid.
- $b \approx 6356752.3142$ m is the semi-minor axis of Earth ellipsoid.
- e is the eccentricity of the Earth ellipsoid.
- ϕ is the latitude (in radians).

The conversion of Degrees i.e. latitude and longitude , and altitude in cartesian coordinate system using WSG84 earth model the x,y,z coordinates are given below.

$$X = (N + h)\cos(\phi)\cos(\lambda)$$

$$Y = (N + h)\cos(\phi)\sin(\lambda)$$

$$Z = \left((1 - e^2)N + h\right)\sin(\phi)$$

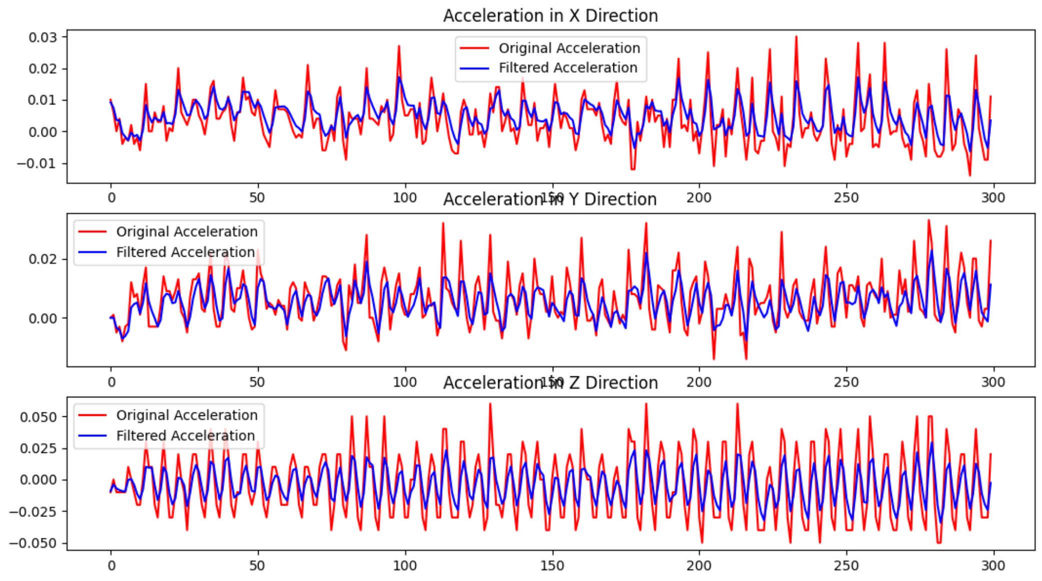
While data acquisition first location point is set as reference X_0, Y_0, Z_0 , the final point from GPS is X_n, Y_n, Z_n the difference in those data points gives the distance between two points at Local tangent plane. We need to fuse those GPS data with Body frame acceleration i.e Inertial data.

Bno055 Sensor.

Bno055 sensor is 9 axis sensor that provides the 3 axis acceleration(m/s^2) , 3 axis magnetometer and 3 axis gyroscope rotation in radian per second.

Due to sensor noise and movement noise in sensor the data might be noisy to make noise free data, it is needed to implement filter algorithms

1. firstly Kalman estimation filter was implemented the result plot of filters were given in each accelerometer axis.



From visualization of graph it is clearly seen that the Kalman estimation cause some lead-lag problem as Kalman filters works on prediction and update estimation mechanism. Due to this problem the true acceleration in certain axis is hidden and removed. To introduce good filter.

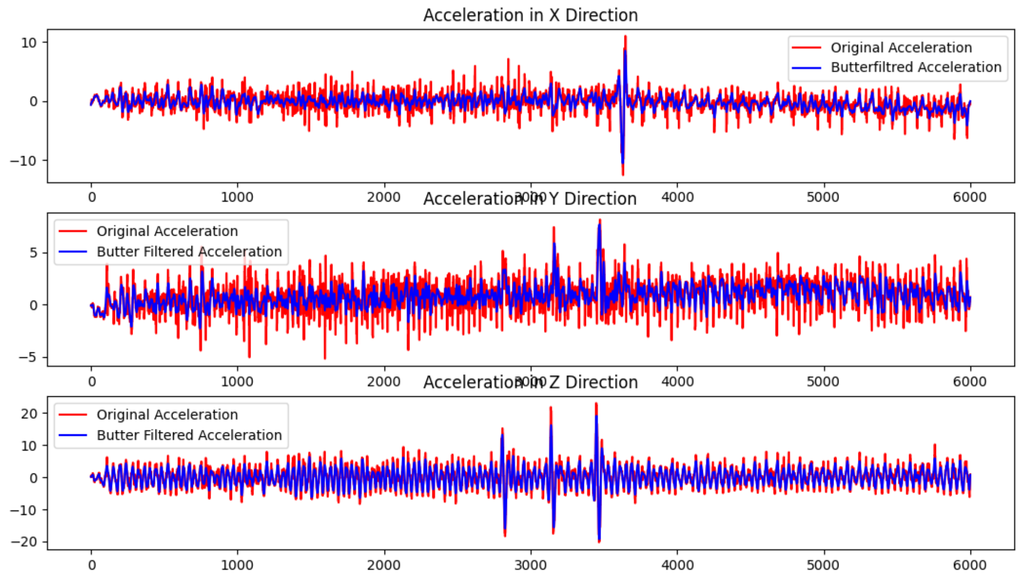
Additionally, from study,

An important class of theoretical and practical problems in communication and control is of a statistical nature. Such problems are:

1. *Prediction of random signals;*
2. *separation of random signals from random noise;*

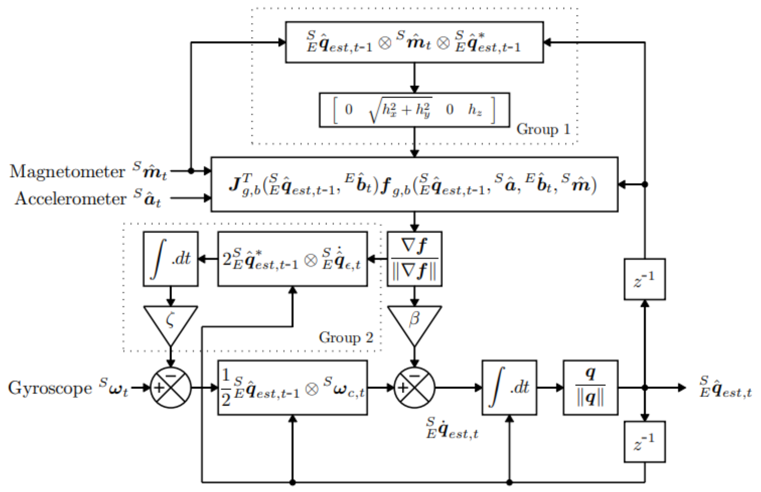
We should understand that Kalman filter is used to solve the problem of type 1 to predict the random variables that we cannot directly measure (this shows why the word *filter* should be misleading in Kalman Filter, technically it should have been called Kalman estimator). Now the problem of smoothing the accelerometer reading (which we are going to introduce) is of type 2. Actual filters like High-Pass Filter (HPF), Low-Pass Filter (LPF) etc. are good candidates to solve this problem. This is the reason why not to use Kalman Filter but LPFs.

Due to this reason the second order butterworth filter was used, this implementation is good candidate to make noise free accelerometer data.

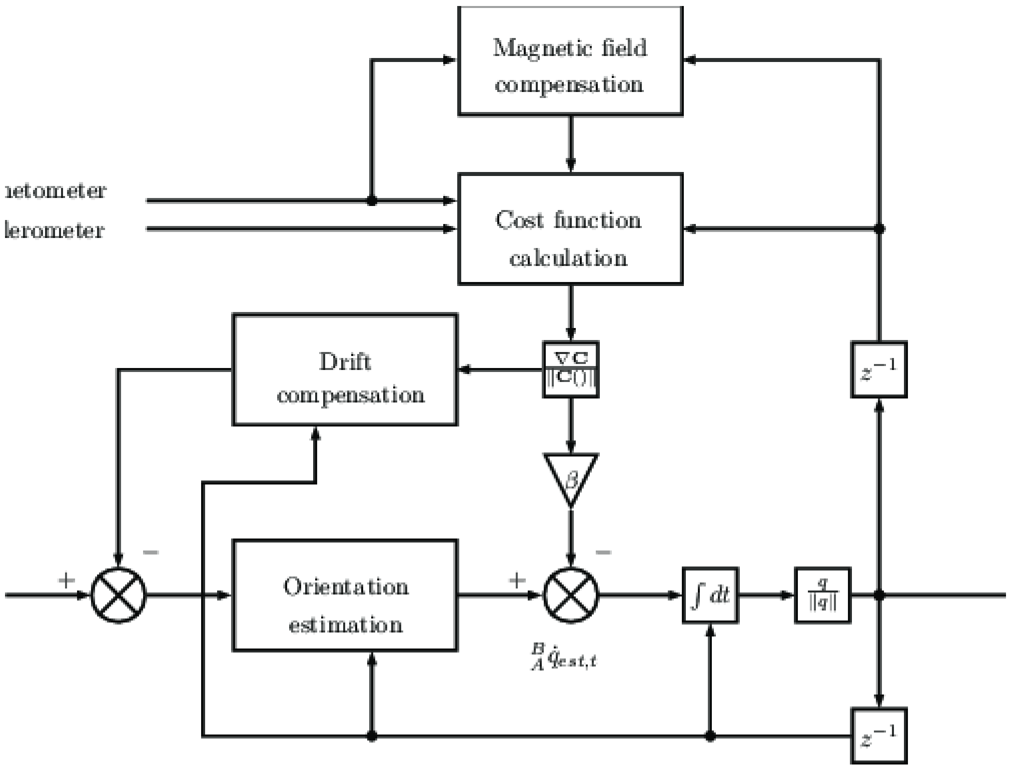


From above acceleration plot shows the smooth acceleration data in each axis.

Implementation of Madgwick algorithm(AHRS) to estimate orientation of data with the fusion of accelerometer, gyroscope and magnetometer. In each axis which gives the roll pitch and yaw also provides the quaternions.



A pictorial representation of Madgwick algorithm is given below.



The orientation is given from Madgwick algorithm and also rotate the sensor's accelerometer frame of reference to magnetic North , east and down acclerations in each axis. Magnetic declination at dhulikhel is 0 degree and 16 minutes so equivalent to 0.26 degree, So in this model magnetic declination is not considered.

Again it is necessary to remove gravitational component of acceleration due to gravity to make gravity free the matrix multiplication is done.

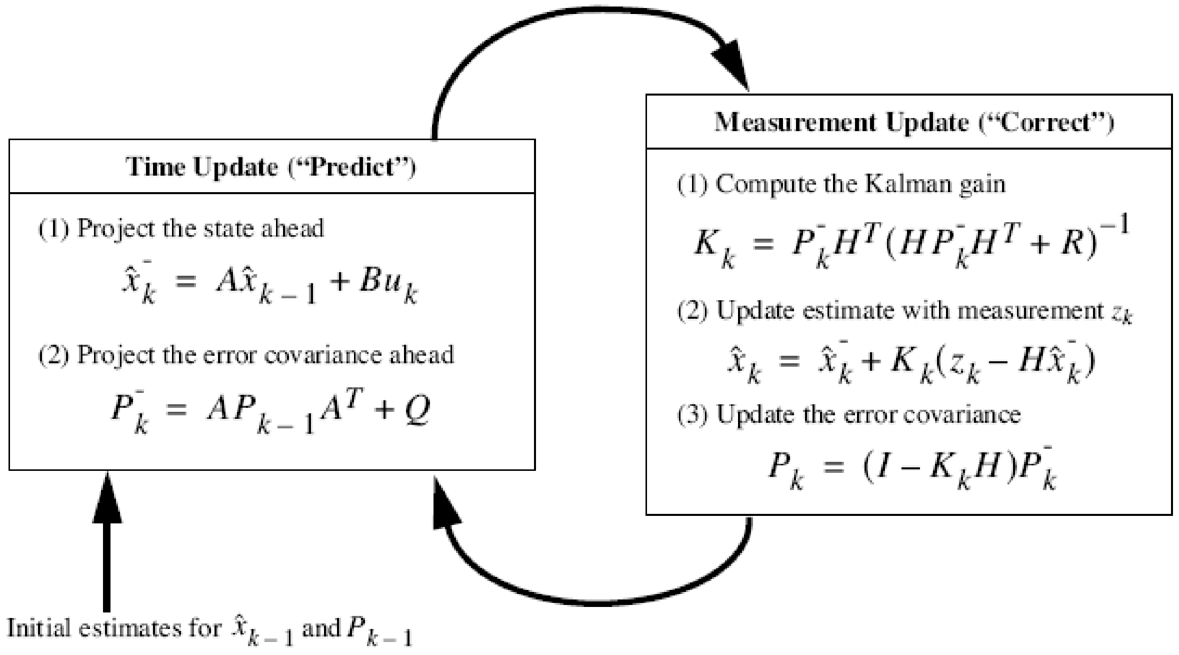
Now from IMU the Accleration component Ae,An,Ad are estimated at east north and down respectively.

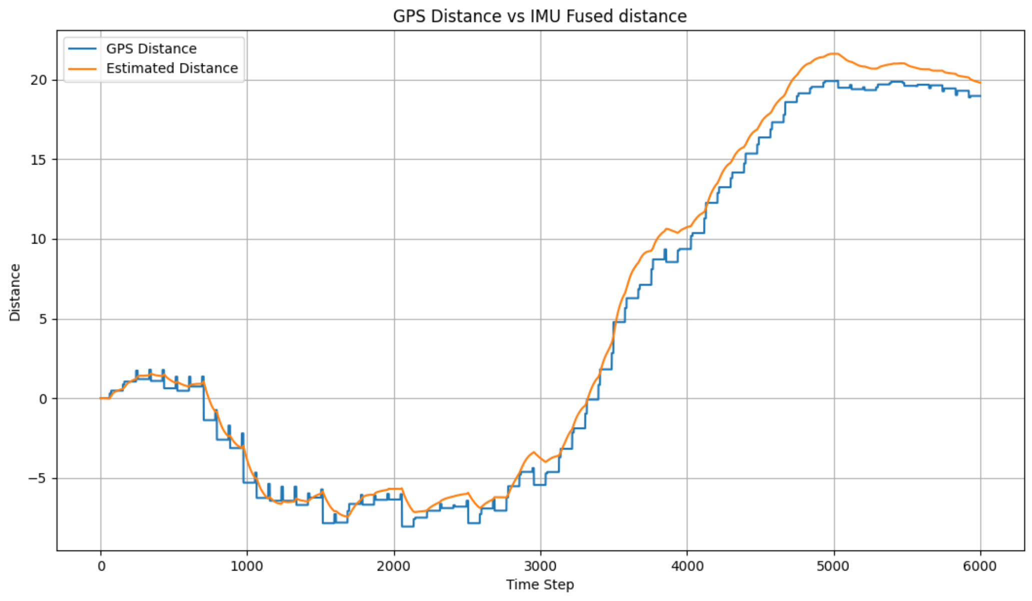
Kalman fusion between accelerometer and GPS. Data.

1. Due to double integration, the error deviation is more in accelerometer data but it has high sampling and high accuracy for shorter distance estimation
2. Whereas GPS has high accuracy with low sample time I,e 1hz so to estimate the position in between 1 second the accelerometer data is useful

In order to make Kalman filter we have to define followings variables

1. State transition matrix(A) (change in state in my case it is position)
2. Control matrix(B) (how affect the control variable to state)
3. Control vector(meu)(acceleration vector from accelerometer)
4. Process noise covariance (Q)
5. Measurement covariances (R)
6. Measurement vector(GPS state)

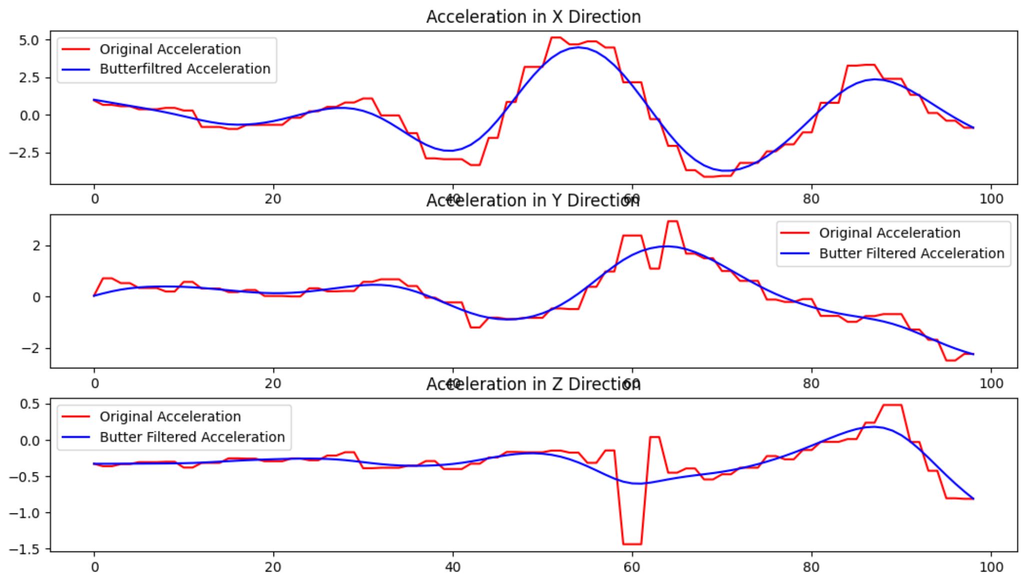

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

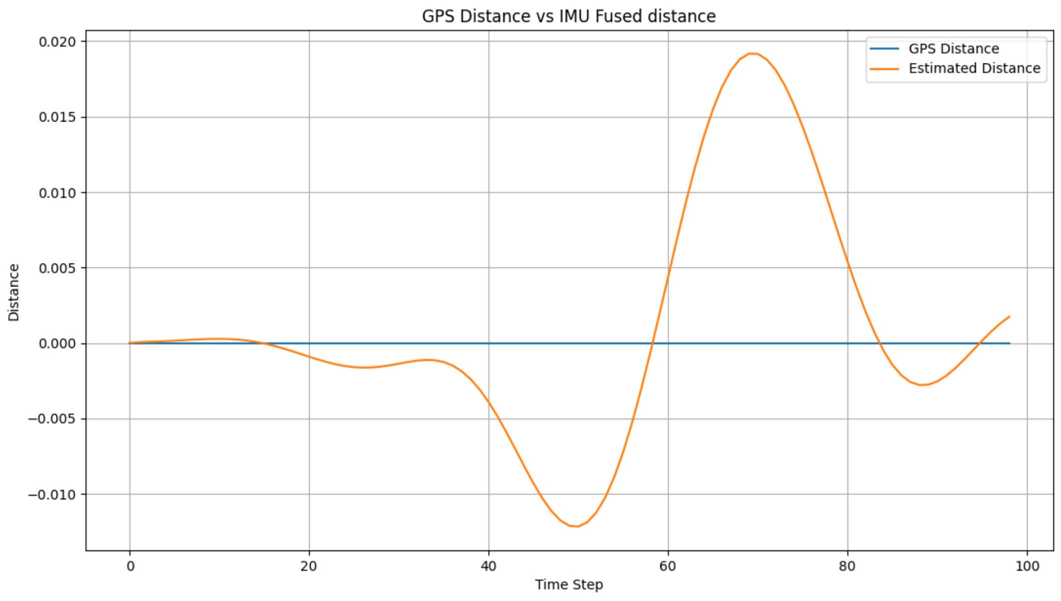


This figure shows the change in latitude and accelerometer prediction fusion

In order to look our model is right or wrong. When gps are at 0,0,0 the estimation is plotted below.

The acceleration estimation Is quite good.





We can clearly see the change In distance in estimation curve due to acceleration, again the curve is converges to zero as our gps reading is zero. The Kalman model is updated by gps data