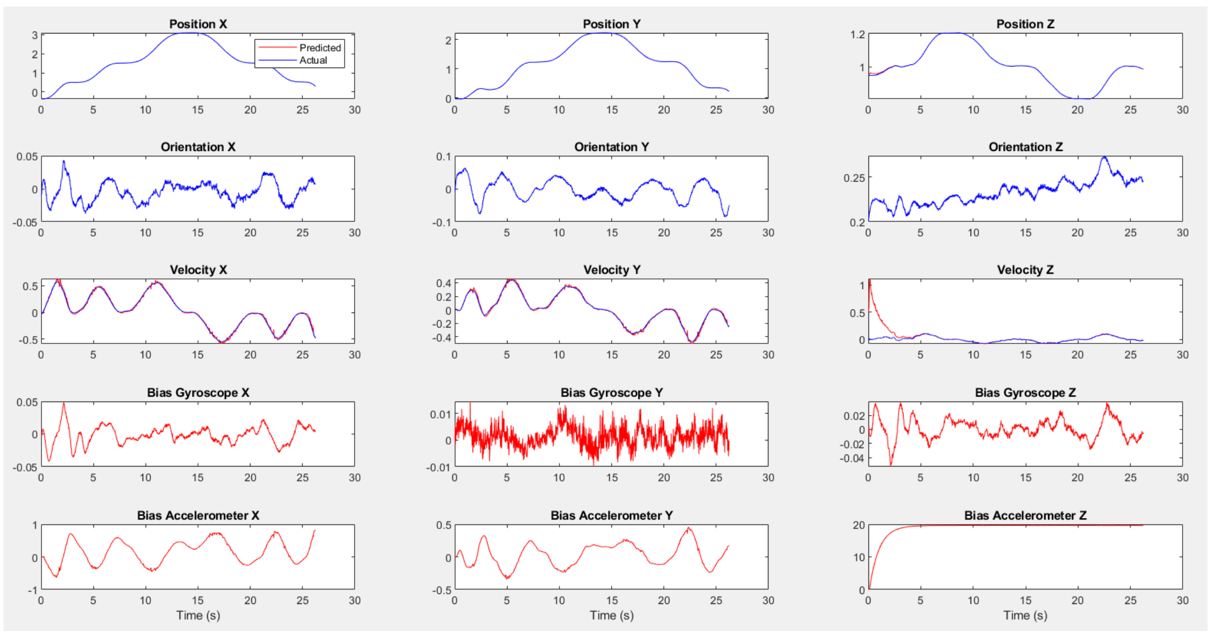
Kota Naveen Chinta
N14724517

ROBOT LOCALIZATION AND NAVIGATION PROJECT 1 REPORT

Approach to Part 1 of the Project



(i) *Results on Implementation of Extended Kalman Filter using the body frame acceleration and angular velocity from the onboard of IMU as your control inputs and position, the orientation of the Vicon.*

Approach for the results obtained in Fig(i):

- **Process Model**

(a) The given control inputs are body frame acceleration, and angular velocity from the onboard of IMU is taken into consideration for obtaining states.

Can accurately estimate the commanded linear acceleration and angular velocity using the IMU

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_a \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{linear velocity} \\ \text{gyroscope bias} \\ \text{accelerometer bias} \end{bmatrix} \in \mathbf{R}^{15}$$

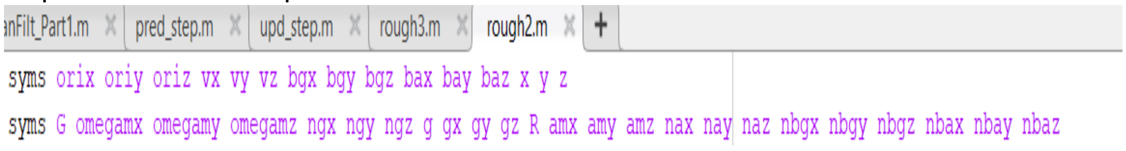
(b) In this project, the states were given and are taken by indexing uPrev(i), where is the index position.

(c) Derivative states give the process model, and necessary equations are substituted (noisy estimate and drift bias of gyroscope and accelerometer, angular velocity, rotation matrix).

Kota Naveen Chinta
N14724517

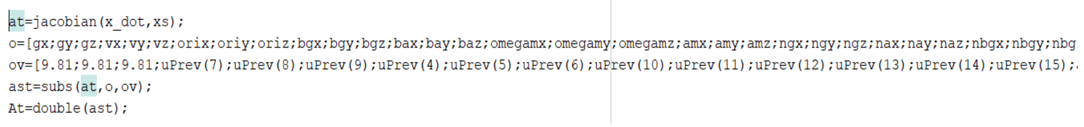
ROBOT LOCALIZATION AND NAVIGATION PROJECT 1 REPORT

- Matlab Approach for obtaining Process Model.
  (a) I am using the symbolic function in MatLab to create the symbolic required to solve the process Model.



```
syms orix oriy oriz vx vy vz bgx bgy bgz bax bay baz x y z
syms G omegamx omegamy omegamz ngx ngy ngz g gx gy gz R amx amy amz nax nay naz nbgx nbgy nbgz nbax nbay nbaz
```

  (b) After generating the process model, it has to be linearized because of the nonlinear functions, and it is done by calculating jacobian with respective states (At) and noises(Ut) as a function of previous mean, control inputs, zero-mean noise.



```
at=jacobian(x_dot,xs);
o=[gx;gy;gz;vx;vy;vz;orix;oriy;oriz;bgx;bgy;bgz;bax;bay;baz;omegamx;omegamy;omegamz;amx;amy;amz;ngx;ngy;ngz;nax;nay;naz;nbgx;nbgy;nbg
ov=[9.81;9.81;9.81;uPrev(7);uPrev(8);uPrev(9);uPrev(4);uPrev(5);uPrev(6);uPrev(10);uPrev(11);uPrev(12);uPrev(13);uPrev(14);uPrev(15);
ast=subs(at,o,ov);
At=double(ast);
```

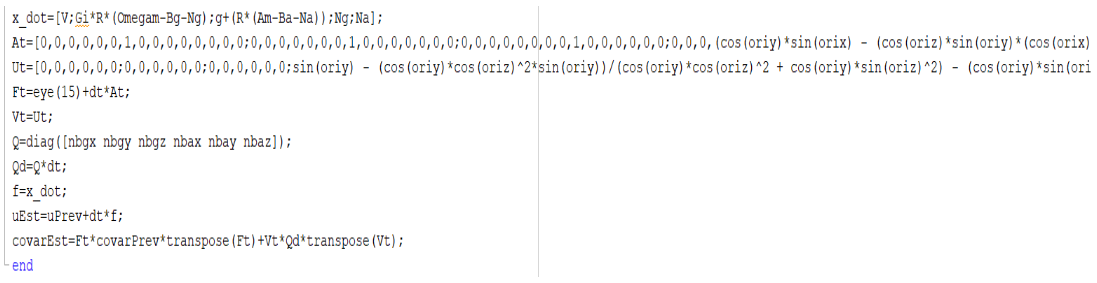
  (c) The values are substituted in the jacobian using the subs function. Since everything is in symbolics, it can be converted into numbers by double function.
  (d) After calculating At and Ut, we need to convert the continuous dynamics to a discrete-time system where Ut and At are the jacobians with respective noise and states, Q is the bias noise of the gyroscope and accelerometer.

$$\left.\begin{array}{l} \circ\ F_t = I + \delta t\, A_t \\ \circ\ V_t = U_t \\ \circ\ Q_d = Q\, \delta t \end{array}\right\} \text{Discretization}$$

  (e) After Discretization, the prediction step is computed.

```
Ft=eye(15)+dt*At;
Vt=Ut;
Q=diag([nbgx nbgy nbgz nbax nbay nbaz]);
Qd=Q*dt;
f=x_dot;
uEst=uPrev+dt*f;
covarEst=Ft*covarPrev*transpose(Ft)+Vt*Qd*transpose(Vt);
end
```

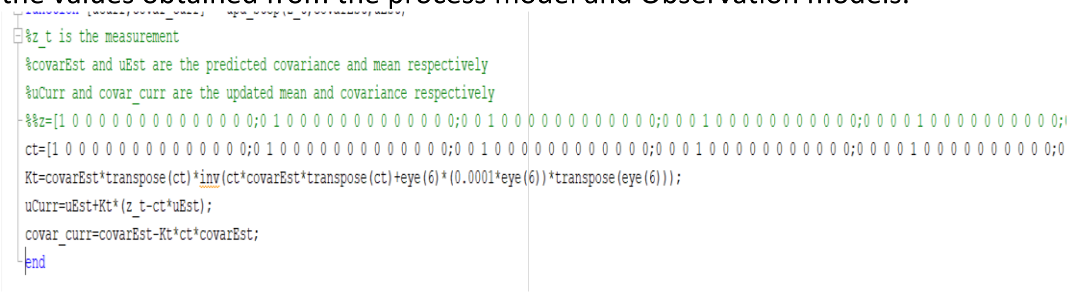
- Optimized Matlab code
  (a) Using symbolic functions, jacobian function, subs function can cause the code to execute in 5-10 minutes.
  (b) First, obtain the required equations using the syms function of Matlab in a new tab.
  (c) After obtaining the equations, it can be further optimized by initializing the values of the states at the beginning of the code and directly substituting them in the equations.



```
x_dot=[V;Gi*R*(Omegam-Bg-Ng);g+(R*(Am-Ba-Na));Ng;Na];
At=[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0;0,0,0,0,0,0,0,1,0,0,0,0,0,0,0;0,0,0,0,0,0,0,0,1,0,0,0,0,0,0;0,0,0,(cos(oriy)*sin(orix) - (cos(oriz)*sin(oriy)*(cos(orix)
Ut=[0,0,0,0,0,0;0,0,0,0,0,0;0,0,0,0,0,0;sin(oriy) - (cos(oriy)*cos(oriz)^2*sin(oriy))/(cos(oriy)*cos(oriz)^2 + cos(oriy)*sin(oriz)^2) - (cos(oriy)*sin(ori
Ft=eye(15)+dt*At;
Vt=Ut;
Q=diag([nbgx nbgy nbgz nbax nbay nbaz]);
Qd=Q*dt;
f=x_dot;
uEst=uPrev+dt*f;
covarEst=Ft*covarPrev*transpose(Ft)+Vt*Qd*transpose(Vt);
end
```

(d) The optimized MatLab code takes less than 1 min to execute.

- **Observation Model**
  (a) In part 1 of the project, position and orientation are given from vicon. Where is Ct is computed based on the measurements from the sensor.
  (b) In this project, the elements in the matrix are not nonlinear, so it does not have to be linearized.
  (c) Finally, the updated mean and covariance are obtained by substituting the values obtained from the process model and Observation models.



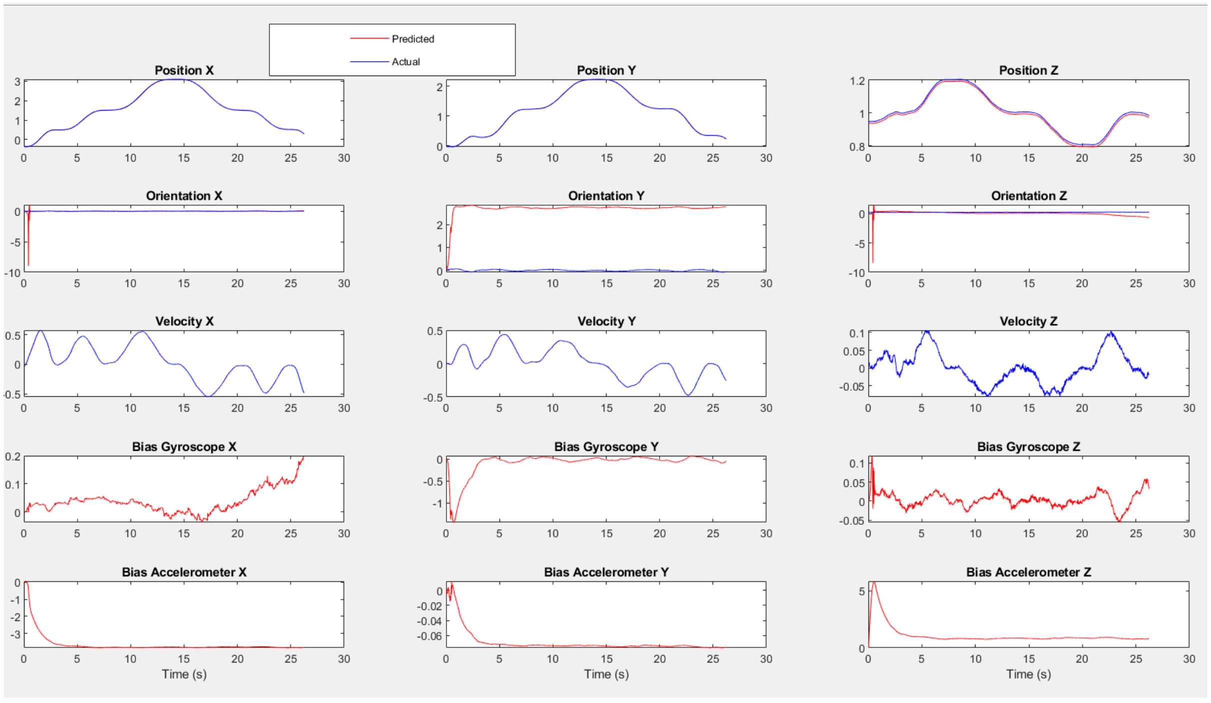
```
%z_t is the measurement
%covarEst and uEst are the predicted covariance and mean respectively
%uCurr and covar_curr are the updated mean and covariance respectively
%%z=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
ct=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;0
Kt=covarEst*transpose(ct)*inv(ct*covarEst*transpose(ct)+eye(6)*(0.0001*eye(6))*transpose(eye(6)));
uCurr=uEst+Kt*(z_t-ct*uEst);
covar_curr=covarEst-Kt*ct*covarEst;
end
```

- Noise values are tuned in order to match predicted values with the Actual values

Kota Naveen Chinta
N14724517

ROBOT LOCALIZATION AND NAVIGATION PROJECT 1 REPORT

Approach to Part 2 of the project



(ii) *Results on Implementation of Extended Kalman Filter using the body frame acceleration and angular velocity from the onboard of IMU as your control inputs and velocity of the Vicon.*

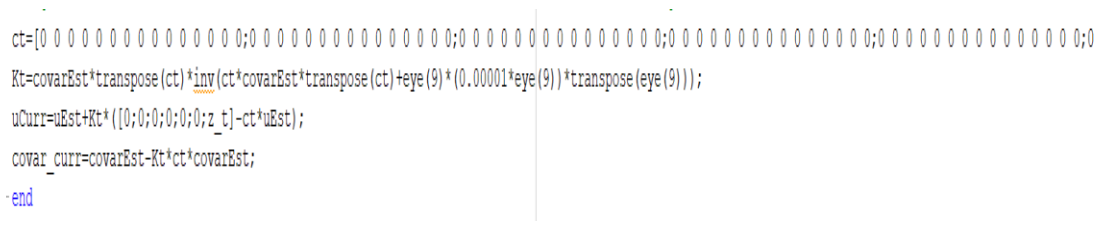
Approach for the results obtained in Fig(ii):

- **Process Model**
  (a) In this project, the control inputs are acceleration and angular velocity. So the above-discussed approach for Fig (i) of the process model equation can be used to solve the update step.

- **Observation Model**
  (a) In part 1 of the project, velocity is given from vicon. Where is Ct is computed based on the measurements from the sensor.
  (b) In this project, the elements in the matrix are not nonlinear, so it does not have to be linearized.
  (c) Finally, the updated mean and covariance are obtained by substituting the values obtained from the process model and Observation models

Kota Naveen Chinta
N14724517

ROBOT LOCALIZATION AND NAVIGATION PROJECT 1 REPORT



```
ct=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0
Kt=covarEst*transpose(ct)*inv(ct*covarEst*transpose(ct)+eye(9)*(0.00001*eye(9))*transpose(eye(9)));
uCurr=uEst+Kt*([0;0;0;0;0;0;z_t]-ct*uEst);
covar_curr=covarEst-Kt*ct*covarEst;
end
```

- Noise values are tuned to match predicted values with the Actual values.