.

Student Satellite Project
Indian Institute of Technology, Bombay
Powai, Mumbai - 400076, INDIA

Website: www.aero.iitb.ac.in/satlab

ekf_quadrotor

Guidance, Navigation and Controls Subsystem

---

main ()

Code author: Shreya JVS
Created on: DD/MM/YYYY
Last modified: DD/MM/YYYY
Revived by: Name of the person who has reviewed the code
Description:
This is the implementation of Extended Kalman Filter algorithm to estimate the attitude and angular rates of a quadrotor. Magnetometer, Accelerometer and Gyroscopes are the sensors considered. In the code, the measurements are generated from the model parameters with standard normal measurement noise. The state disturbances are also taken to be standard normal random variables. I have considered a control law which reduces angular rates constantly and brings Euler angles to a constant value. Time step is 0.05secs
Formula & References:
State space equations

$$\dot{\phi} = \omega_1 + tan\theta(\omega_2 sin\phi + \omega_3 cos\phi)$$

$$\dot{\theta} = \omega_2 cos\phi - \omega_3 sin\phi$$

$$\dot{\psi} = sec\theta(\omega_2 sin\phi + \omega_3 cos\phi)$$

$$\dot{\omega_1} = \frac{I_{yy} - I_{zz}}{I_{xx}}\omega_2\omega_3 + M_1/I_{xx}$$

$$\dot{\omega_2} = \frac{I_{zz} - I_{xx}}{I_{yy}}\omega_1\omega_3 + M_2/I_{yy}$$

$$\dot{\omega_3} = \frac{I_{xx} - I_{yy}}{I_{zz}}\omega_1\omega_2 + M_3/I_{zz}$$

Measurement model

$$y = \begin{bmatrix} R^T m_I \\ R^T a_I \\ \omega \end{bmatrix} + \nu$$

where m_I and a_I are system dependent constants.
The main reference for the above equations is the EKF course project done by Piyush. I defined an object of class KalmanFilter and implented Extended Kalman Filter algorithm with the help of functions defined in that class.

I used odeint function from scipy to generate the model track of the system from the continuous time state space equations.
Input parameters:
There are no inputs to this function.
Output:
It plots the estimates of all the states of the system with model track values. It also plots actual measurements vs. predicted measurements

kf_control

Code author: Shreya JVS
Created on: DD/MM/YYYY
Last modified: DD/MM/YYYY
Reviwed by: Name of the person who has reviewed the code
Description:
It defines the control law. This function is defined in the main function
Formula & References:
Control Law

$$M = \omega \times I\omega I k_p \omega, \; k_p = 0.5I_{33}$$

Input parameters:
The input is *state* which is the estimated state of quadrotor.
Output:
The output is the control input vector v_U

Class - KalmanFilter

Code author: Shreya JVS
Created on: DD/MM/YYYY
Last modified: DD/MM/YYYY
Reviwed by: Name of the person who has reviewed the code
Description:
This class initializes several variables. It contains three functions - **kf_predict**, **kf_update** and **real_ode** which is used to solve differential equations of state.
Formula & References:
Extended Kalman Filter equations
Model

$$\dot{x} = f(x, u) + G\omega, \omega\ N(0, Q)$$

$$y_k = h(x_k) + \nu_k, \nu_k\ N(0, R_k)$$

I took these equations and learned EKF from here
Input parameters:
The input arguments to the class are:

1. **DT** : (Float) - Sampling Rate of the discretized system and the measurements taken. *Seconds*
2. **SIGMA_MAG_M** : (Float) - standard deviation of measurement errors of magnetometer. *meters*

3. **SIGMA_ACC_M** : (Float) - Standard deviation of measurement errors of accelerometer. *meters/sec*

4. **SIGMA_W_M** : (Float) - Standard deviation of measurement errors of angular velocities.

5. **SIGMA_SD** : (Float) - standard deviation of state disturbances.

6. **m_A** : (Float) - State transition matrix. 6×6

7. **m_B** : (Float) - Input matrix 6×6

8. **v_X_0** : (Float) - Initial state. 6×1

9. **mx** : (Float) - magnetic moment in x direction

10. **my** : (Float) - magnetic moment in y direction

11. **mz** : (Float) - magnetic moment in z direction

12. **ax** : (Float) - acceleration in x direction

13. **ay** : (Float) - acceleration in y direction

14. **az** : (Float) - acceleration in z direction

kf_predict

Code author: Shreya JVS
Created on: DD/MM/YYYY
Last modified: DD/MM/YYYY
Reviwed by: Name of the person who has reviewed the code
Description:
This function calculates the next predicted state of the system and its error covariance matrix, from previous state using state space equations.
Formula & References:
Propagate

$$F_k = e^{A\delta t}, with\ A = \frac{\partial f}{\partial x}$$

$$B_k = [\int_0^{\delta t} d\tau e^{A\tau} B], with\ B = \frac{\partial f}{\partial u}$$

$$x_k^- = F_{k-1} x_{k-1}^+ + B_{k-1} u_{k-1}$$

$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + \delta t G Q G^T$$

The discretization is done in this function is uses a slightly more modified formula when the matrix A is singular as the original formula includes inverse of A. I have taken this from here
Input parameters:
The input is *self*, i.e. all the variables defined in the class and **v_U** which is the control input vector.
Output:
It updates the state and state covariance with new control inputs and without considering new measurements and returns this state. It returns the predicted state(v_X)

kf_update

Code author: Shreya JVS
Created on: DD/MM/YYYY
Last modified: DD/MM/YYYY
Reviwed by: Name of the person who has reviewed the code
Description:
This function updates the state, its error covariance matrix and the kalman gain using equations derived from the extended kalman filter algorithms. The function is used recursively to estimate state.
Formula & References:
Kalman Gain

$$H_k = \frac{\partial h}{\partial x}$$

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R^T]^{-1}$$

Update

$$x_k^+ = x_k^- + K_k[y_k - h(x_k^-)]$$

$$P_K^+ = [I - K_k H_k] P_k^-$$

Input parameters:
The input is *self*, i.e. all the variables defined in the class and the following:

1. **v_y_m** : (Float) - This is the new measurement(vector) taken. 2×1
2. **mx** : (Float) - magnetic moment in x direction
3. **my** : (Float) - magnetic moment in y direction
4. **mz** : (Float) - magnetic moment in z direction
5. **ax** : (Float) - acceleration in x direction
6. **ay** : (Float) - acceleration in y direction
7. **az** : (Float) - acceleration in z direction

Output:
This function updates kalman gain, state and covariance matrices and does not return any value.