.

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

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

KalmanFilter.m

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 Kalman Filter algorithm to estimate the state of a simple system. My system is a robot moving in one direction with position and velocity as its state variables. The measurements take are position, velocity and angular velocity of the wheels. The system can have user defined trajectories and control inputs. 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. The standard deviations of all the random variables are again user defined.
Formula & References:
As the base case, I have considered the trajectory to be SHM with a damping term proportional to velocity and a control input, i.e., the differential equation is

$$\ddot{x} = -\omega^2 x - b\dot{x} + u$$

where I took $\omega = 10$, b = 2 and u = 1. I defined an object of class KalmanFilter and implented Kalman Filter algorithm with the help of functions defined in that class. I initially referred this source to understand how to implement kalmn filter for a continuous system and the functions which were needed. Later, I understood how to convert continuous time to discrete time from 8.3.2 section of this document.
I used odeint function from scipy to generate the model track of the system from the continuous time state space differential equations.
Input parameters:
There are no inputs to this function.
Output:
It plots position and velocity of the system vs. time. It returns a 2xn matrix which contains all the estimated position values in first the column and velocity values in the second column at each time-step.

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 and converts continuous state space equations to discrete state space equations. It contains three functions - **kf_predict**, **kf_update** which apply kalman filter algorithm and **real_ode** which is used to solve differential equations of state.
Formula & References: The original time continuous state space equations are:

$$\dot{X} = AX + BU + \tau$$

$$Y = CX + v$$

where tau is state disturbance and v is measurement noise. The system is discretized as shown below:

$$\phi = e^{A \times dt}$$

$$\tau = [\phi - I]A^{-1}B$$

where the new discretized state space equations are

$$X[k+1] = \phi X[k] + \tau U[k]$$

$$P[k+1] = \phi P[k] \phi^T + Q$$

we predict the next state and error covariance.
Input parameters:
The input arguments to the class are:

1. **R** : (Float) - Radius of the wheels of the robot. *meters*
2. **DT** : (Float) - Sampling Rate of the discretized system and the measurements taken. *Seconds*
3. **SIGMA_POS_M** : (Float) - standard deviation of measurement errors of position. *meters*
4. **SIGMA_VEL_M** : (Float) - Standard deviation of measurement errors of velocity. *meters/sec*
5. **SIGMA_W_BIB_M** : (Float) - Standard deviation of measurement errors of angular velocity of wheels. *1/sec*
6. **SIGMA_SD_POS** : (Float) - standard deviation of state disturbances of position. *meters*
7. **SIGMA_SD_VEL** : (Float) - standard deviation of state disturbances of velocity *meters/sec*
8. **m_A** : (Float) - State transition matrix. 2×2
9. **m_B** : (Float) - Input matrix 2×2
10. **v_U** : (Float) - Time invariant control inputs. 2×1
11. **v_X_0** : (Float) - Initial state. 2×1

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:
Note that the system has been discretized in the initialization of the class.
Input parameters:
The input is *self*, i.e. all the variables defined in the class.
Output:
It updates the state and state covariarance 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 using equations derived from the kalman filter algorithms. The function is used recursively to estimate state.
Formula & References:
The understood the derivation and the equations of Kalman Filter algorithm from here.
Updating Kalman Gain:

$$K = PC^T[R + CPC^T]^{-1}$$

Updating state:

$$X_{new} = X + K[Y - CX]$$

Updating error covariance:

$$P_{new} = [I - KC]P$$

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

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