# The Party's Estimator

# MAE 5010 Autopilot

import numpy as np

import math

y = np.array([0.538385085, -0.035892339, 9.662217659, -0.002443459, 0.133168509, 0.014660753, -5.046, -20.184, 57.246])

def heading\_calc(mag\_x, mag\_y, mag\_z, phi, theta):

dt = 3 # declination angle in degrees

M\_B = np.array([mag\_x, mag\_y, mag\_z]) # magnetometer output

M\_i = Rot\_i\_B(M\_B, phi, theta) # put magnetometer in inertial frame

psi = dt + math.atan(M\_i[1]/M\_i[2]) # dt is declination angle

return psi

def Rot\_i\_B(Matrix, phi, theta):

Rot\_Mat = np.array([[math.cos(theta), math.sin(theta)\*math.sin(phi), math.sin(theta)\*math.cos(phi)],

[0, math.cos(phi), -math.sin(phi)],

[-math.sin(theta), math.cos(theta)\*math.sin(phi), math.cos(theta)\*math.cos(phi)]])

R\_Mat = Matrix \* Rot\_Mat

return R\_Mat

#define initial states from y

ax = y[0]

ay = y[1]

az = y[2]

gyro\_p = y[3]

gyro\_q = y[4]

gyro\_r = y[5]

mag\_x = y[6]

mag\_y = y[7]

mag\_z = y[8]

# definition of noise and biases

AccelNoise = .002 # noise of accelerometer

GyroNoise = 1e-5 # noise of gyro

AttitudeNoise = .3 # attitude noise

AccelBias = np.array([0,0,0]) # Bias of Accelerometer... from christian

GyroBias = np.array([0,0,0]) # gyro bias

MagBias = np.array([0,0,0]) # magnetometer bias

# initial orientation estimate

phi = 0

theta = 0

psi = heading\_calc(mag\_x, mag\_y, mag\_z, phi, theta) # need to create function to do this

euler = np.array([phi, theta, psi])

# define x\_hat

x\_hat = np.array([euler, GyroBias, AccelBias]) # 1-3 Euler angles ,,,, #4-6 xyz gyro bias estimates, #7-9 xyz gyro bias

# define R uncertainty in measurement

# Define Q uncertainty in model

Q = np.identity(9)