```
def eom(data):
    import numpy as np

    rho = data[4]
    m = data[1]*0.454
    theta = np.deg2rad(data[2])

    #import control
    """ Variables not sure needed """
    w = m*9.81 # N (standard aircraft mass)
    m_fs = 0.048 # kg/s (standard engine fuel flow per engine)
#    rho_0 = 0.90 # kg/m^3 (standard air density)

    """ Input variables """

    V = data[5]
    S = 30.0 # m^2 (Surface area wing)
    c = 2.0569 # m (MAC)
    b = 15.911 # m (Wing span)
    e = 0.8 # (Oswald efficiency factor)

    mu_c = m/(rho*S*c)
    mu_b = m/(rho*S*b)

    D_c = c/V
    D_b = b/V
    C_D0 = 0.04 # (Zero drag coefficient)
    C_La = 5.084 # (dC_L/da)
    Ks_xx = 0.019 # (Radius of gyration about x)
    Ks_yy = 1.3925 # (Radius of gyration about y)
    Ks_zz = 0.042 # (Radius of gyration about z)
    Ks_xz = 0.002

    """ Linear model stability derivatives - symmetric flight """
    # Longitudinal force derivatives
    C_Xu = -0.0279 # (dC_X / du)
    C_Xa = -0.4797 # (dC_X / dalpha)
    C_Xda = 0.0833 # (dC_X / dalpha_dot)
    C_Xq = -0.2817 # (dC_X / dq)
    C_Xdelta = -0.0373 # (dC_X / ddelta)

    # Normal force derivatives
    C_Zu = -0.3762 # (dC_Z / du)
    C_Za = -5.7434 # (dC_Z / dalpha)
    C_Zda = -0.0035 # (dC_Z / dalpha_dot)
    C_Zq = -5.6629 # (dC_Z / dq)
    C_Zdelta = -0.6961 # (dC_Z / ddelta)

    #Pitch moment derivatives
    C_m0 = 0.0297 # (C_m0)
    C_mu = 0.0699 # (dC_m / du)
    C_ma = -0.5626 # (dC_m / dalpha)
    C_mda = 0.1780 # (dC_m / dalpha_dot)
    C_mq = -8.7941 # (dC_m / dq)
#    C_mdelta = -1.1642 # (dC_m / ddelta)
    C_mdelta = -0.987672
    C_mTc = -0.0064 # (dC_m / dTc)

    """ Linear model stability derivatives - asymmetric flight """
    # Lateral force derivatives
    C_Yb = -0.7500 # (dC_Y / dbeta)
    C_Ydb = 0 # (dC_Y / dbeta_dot)
    C_Yp = -0.0304 # (dC_Y / dp)
    C_Yr = 0.8495 # (dC_Y / dr)
    C_Ydeltaa = -0.0400 # (dC_Y / delta_a)
    C_Ydeltar = 0.2300 # (dC_Y / delta_r)

    # Roll moment derivatives
    #C_D = (C_D0**2)+(C_L**2)/(np.pi*(b/c)*e)
    C_lb = -0.1026 # (dC_l / dbeta)
    C_lp = -0.7108 # (dC_l / dp)
    C_lr = 0.2376 # (dC_l / dr)
    C_ldeltaa = -0.2309 # (dC_l / delta_a)
    C_ldeltar = 0.0344 # (dC_l / delta_r)

    # Yaw moment derivatives
    C_nb = 0.1348 # (dC_n / dbeta)
    C_ndb = 0 # (dC_n / dbeta_dot)
    C_np = -0.0602 # (dC_n / dp)
    C_nr = -0.2061 # (dC_n / dr)
    C_ndeltaa = -0.0120 # (dC_n / delta_a)
```

```
C_ndeltar = -0.0939 # (dC_n / delta_r)

""" Coefficient Calculations """
C_L = -(w)/(0.5*rho*(V**2)*S)
C_X0 = (w*np.sin(theta))/(0.5*rho*(V**2)*S)
C_Z0 = -(w*np.cos(theta))/(0.5*rho*(V**2)*S)

print('C_Z0 = ' + str(C_Z0))
""" Equations of symmetric motion """
Psym = np.matrix([[-2*mu_c*D_c/V, 0, 0, 0],
                  [0, (C_Zda-2*mu_c)*D_c, 0, 0],
                  [0, 0, -D_c, 0],
                  [0, C_mda*D_c, 0, -2*mu_c*Ks_yy*D_c*D_c]])

Qsym = -1*np.matrix([[C_Xu/V, C_Xa, C_Z0, 0],
                  [C_Zu/V, C_Za, C_X0, (C_Zq+2*mu_c)*D_c],
                  [0, 0, 0, 1*D_c],
                  [C_mu/V, C_ma, 0, C_mq*D_c]])

Rsym = np.matrix([[-C_Xdelta],
                  [-C_Zdelta],
                  [0],
                  [-C_mdelta]])


Pasym = np.matrix([[(C_Ydb-2*mu_b)*D_b, 0, 0, 0],
                   [0, -0.5*D_b, 0, 0],
                   [0, 0, -4*mu_b*Ks_xx*D_b*D_b/2, 4*mu_b*Ks_xz*D_b*D_b/2],
                   [C_ndb*D_b, 0, 4*mu_b*Ks_xz*D_b*D_b/2, -4*mu_b*Ks_zz*D_b*D_b/2]])

Qasym = -1*np.matrix([[C_Yb, C_L, C_Yp*D_b/2, (C_Yr-4*mu_b)*D_b/2],
                   [0, 0, -1*D_b/2, 0],
                   [C_lb, 0, C_lp*D_b/2, C_lr*D_b/2],
                   [C_nb, 0, C_np*D_b/2, C_nr*D_b/2]])

Rasym = np.matrix([[-C_Ydeltaa, -C_Ydeltar],
                  [0,0],
                  [-C_ldeltaa, -C_ldeltar],
                  [-C_ndeltaa, -C_ndeltar]])

return Psym, Qsym, Rsym, Pasym, Qasym, Rasym
```