```
% quadsim_forces_moments.m
%
% Developed for JHU EP 525.461, UAV Systems & Control
% Adapted from design project in "Small Unmanned Aircraft: Theory and
% Practice", RWBeard & TWMcClain, Princeton Univ. Press, 2012
%
function out = quadsim_forces_moments(uu, P)

    % Extract variables from input vector uu
    %   uu = [wind_ned(1:3); deltas(1:4); x(1:12); time(1)];
    k=(1:3);          wind_ned=uu(k);   % Total wind vector, ned, m/s
    k=k(end)+(1:4);   deltas=uu(k);     % Control surface commands: [delta_e
 delta_a delta_r delta_t]
    k=k(end)+(1:12);  x=uu(k);          % states
    k=k(end)+(1);     time=uu(k);       % Simulation time, s

    % Extract state variables from x
    pn    = x(1);   % North position, m
    pe    = x(2);   % East position, m
    pd    = x(3);   % Down position, m
    u     = x(4);   % body-x groundspeed component, m/s
    v     = x(5);   % body-y groundspeed component, m/s
    w     = x(6);   % body-z groundspeed component, m/s
    phi   = x(7);   % EulerAngle: roll, rad
    theta = x(8);   % EulerAngle: pitch, rad
    psi   = x(9);   % EulerAngle: yaw, rad
    p     = x(10);  % body rate about x, rad/s
    q     = x(11);  % body rate about y, rad/s
    r     = x(12);  % body rate about z, rad/s

    % Combine states to vector form for convenience
    P_ned = [pn; pe; pd];   % NED position, m
    vg_b  = [u; v; w];      % Groundspeed vector, body frame, m/s
    w_b   = [p; q; r];      % body rates about x,y,z, rad/s

    % Extract control commands from deltas
    delta_e = deltas(1); % Elevator, +/-
    delta_a = deltas(2); % Aileron, +/-
    delta_r = deltas(3); % Rudder, +/-
    delta_t = deltas(4); % Throttle, 0 - 1
    [delta_1, delta_2, delta_3, delta_4] =
 mapChannelsToMotors(delta_e,delta_a,delta_r,delta_t);

    % Your code goes below...

    % Prop rotation rates
    omega_1 = P.k_omega*delta_1 + P.prop_1_omega_bias;
    omega_2 = P.k_omega*delta_2 + P.prop_2_omega_bias;
    omega_3 = P.k_omega*delta_3 + P.prop_3_omega_bias;
    omega_4 = P.k_omega*delta_4 + P.prop_4_omega_bias;

    % Torques
```

```
    Tp_1 = P.k_Tp*(omega_1^2);
    Tp_2 = P.k_Tp*(omega_2^2);
    Tp_3 = P.k_Tp*(omega_3^2);
    Tp_4 = P.k_Tp*(omega_4^2);

    % Air in calculatins
    R_ned2b = eulerToRotationMatrix(phi,theta,psi);
    wind_b = R_ned2b*wind_ned;
    va_b = vg_b-wind_b;
    Vair_in = -(va_b(3));

    % Forces
    Fp_1 = P.rho*P.C_prop*P.S_prop ...
        *(Vair_in + (omega_1/P.k_omega)*(P.k_motor - Vair_in)) ...
        *((omega_1/P.k_omega)*(P.k_motor - Vair_in));

    Fp_2 = P.rho*P.C_prop*P.S_prop ...
        *(Vair_in + (omega_2/P.k_omega)*(P.k_motor - Vair_in)) ...
        *((omega_2/P.k_omega)*(P.k_motor - Vair_in));

    Fp_3 = P.rho*P.C_prop*P.S_prop ...
        *(Vair_in + (omega_3/P.k_omega)*(P.k_motor - Vair_in)) ...
        *((omega_3/P.k_omega)*(P.k_motor - Vair_in));

    Fp_4 = P.rho*P.C_prop*P.S_prop ...
        *(Vair_in + (omega_4/P.k_omega)*(P.k_motor - Vair_in)) ...
        *((omega_4/P.k_omega)*(P.k_motor - Vair_in));

    % Set moments
    m_x = (P.delta_y*(Fp_3 + Fp_2) - dy*(Fp_1 + Fp_4));
    m_y = (P.delta_x*(Fp_1 + Fp_3) - dx*(Fp_2 + Fp_4));
    m_z = ((Tp_1 + Tp_2) - (Tp_3 + Tp_4));

    % Add matrices
    f_grav_b = P.mass*R_ned2b*[0; 0; P.gravity];
    f_props_b = [0;0;-(Fp_1+Fp_2+Fp_3+Fp_4)];
    f_b_rotorDrag = -P.mu_rotorDrag*[vg_b(1)-wind_b(1);vg_b(2)-wind_b(2);0];

    % Combine
    f_b=f_grav_b+f_props_b+f_b_rotorDrag;
    m_b=[m_x;m_y;m_z];

    % Compile function output
    out = [f_b; m_b]; % Length 3+3=6

end
```

```
Not enough input arguments.

Error in quadsim_forces_moments (line 11)
    k=(1:3);          wind_ned=uu(k);   % Total wind vector, ned, m/s
```

Published with MATLAB® R2023a