# AOE X224 Homework #2 Solutions

**Problem 1.** Obtain SIDPAC from the following URL:

`https://software.nasa.gov/software/LAR-16100-1`

Once you've downloaded the software and added the subdirectory to your Matlab path, run the command `fly` and begin the simulated flight. Take a screenshot of the scene and include it as your response to this problem.

**Solution.** Hopefully everyone was able to obtain a copy of SIDPAC for use in the course.

**Problem 2.** Create a Matlab simulation of the nonlinear flight dynamic model developed in [1], which is appended to this assignment. (You may omit the actuator model that accounts for time delay and dynamics.) Append your script as your response to this problem.

**Solution.** A Matlab script for this problem and for Problem 3 is attached.

**Problem 3.** Using the Matlab simulation developed in Problem 2, simulate a steady flight condition corresponding constant-altitude, wings-level flight at 45 ft/s. Provide plots of all twelve state variables and all four inputs versus time over a 10 second window.

**Solution.** A Matlab script for this problem and for Problem 2 is attached.

**Problem 4.** Using the Matlab simulation developed in Problem 2, and starting from the steady flight condition determined in Problem 3, apply a square doublet with a total duration of 1 second. Provide plots of all twelve state variables and all four inputs versus time over a 10 second window.

**Solution.** A Matlab script for this problem is attached.

**Problem 5. (Graduate Students)** Incorporate the actuator dynamic model developed in [1] into your model and repeat the simulation of Problem 4.

**Solution.** I have not provided a Matlab script for this problem.

## References

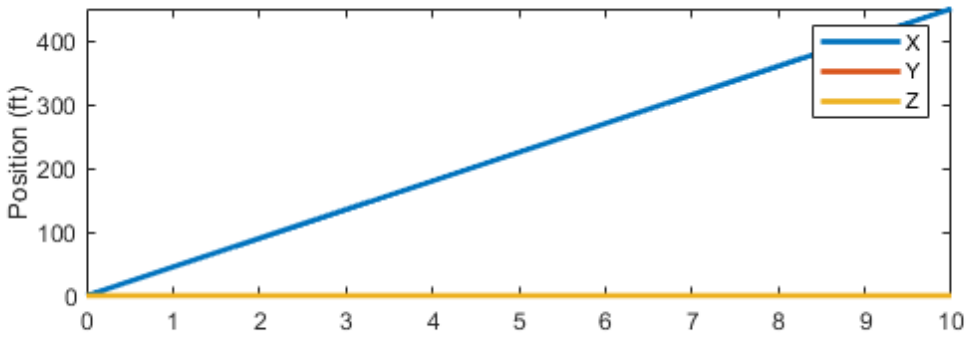

[1] J L Gresham, J-m W Fahmi, b M Simmons, J W Hopwood, W Foster, and C A Woolsey. Flight test approach for modeling and control law validation for unmanned aircraft. In *AIAA SciTech 2022 Forum*, 2022.

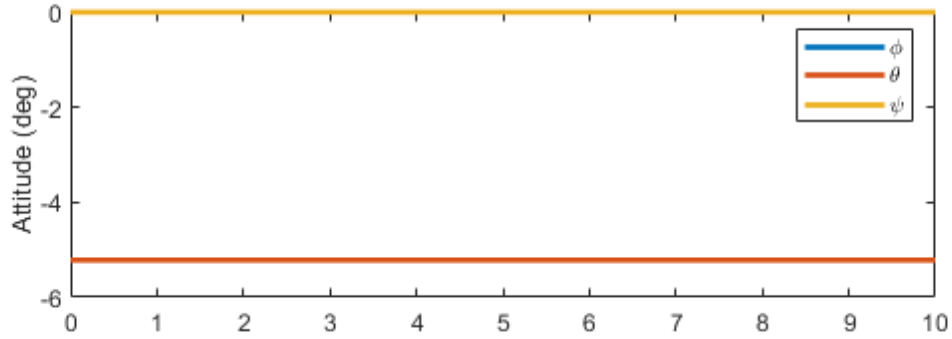
```
% MTD Simulation
% C. A. Woolsey

clear
close all

% GenericFixedWingScript.m solves the nonlinear equations of motion for a
% fixed-wing aircraft flying in ambient wind with turbulence.

global e1 e2 e3 rho m g S b c Inertia ...
Cx0 Cxalpha Cxalpha2 Cxde ...
Cz0 Czalpha Czalpha2 Czde ...
Cm0 Cmalpha Cmalpha2 Cmalpha3 Cmq Cmde ...
Cy0 Cybeta Cybeta3 Cyp Cyr Cyda Cydr ...
Cl0 Clbeta Clp Clr Clda Cldr ...
Cn0 Cnbeta Cnbeta3 Cnp Cnr Cnda Cndr ...
VEq alphaEq thetaEq deEq Power

% Basis vectors
e1 = [1;0;0];
e2 = [0;1;0];
e3 = [0;0;1];

% Atmospheric and gravity parameters (Constant altitude: Sea level)
rho = 2.3769e-3;    % Density (sl/ft^3)
g = 32.174;         % Gravitational acceleration (ft/s^2)

% Aircraft parameters (MTD)
m = 0.211;          % Mass (sl)
W = m*g;            % Weight (lb)
Ix = 0.2163;        % Roll inertia (sl-ft^2)
Iy = 0.1823;        % Pitch inertia (sl-ft^2)
Iz = 0.3396;        % Yaw inertia (sl-ft^2)
Ixz = 0;            % Roll/Yaw product of inertia (sl-ft^2)
Inertia = [Ix, 0, -Ixz; 0, Iy, 0; -Ixz, 0, Iz];

S = 4.92;           % Wing area (ft^2)
b = 5.91;           % Wing span (ft)
c = 0.833;          % Wing chord (ft)
AR = (b^2)/S;       % Aspect ratio

% Longitudinal nondimensional stability and control derivatives
Cx0 = -0.061;
Cxalpha = 0.440;
Cxde = -0.140;
Cxalpha2 = 2.59;

Cz0 = -0.206;
Czalpha = -3.947;
Czde = -21.19;
Czalpha2 = -0.8207;

Cm0 = -0.061;
Cmalpha = -0.8068;
Cmq = -4.937;
Cmde = -0.7286;
Cmalpha2 = -1.251;
Cmalpha3 = -29.92;
```

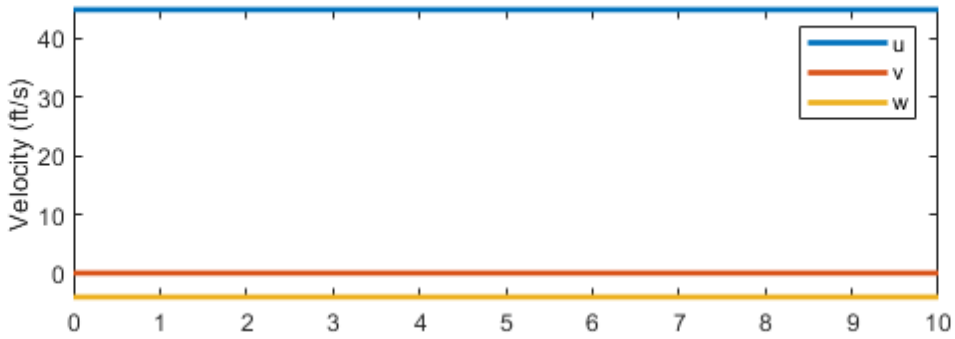
```
% Lateral-directional nondimensional stability and control derivatives
Cy0 = 0;
Cybeta = -0.3761;
Cyp = 0.6323;
Cyr = 0.1951;
Cyda = 0.2491;
Cydr = 0.1544;
Cybeta3 = -0.2294;

Cl0 = 0;
Clbeta = -0.0529;
Clp = -0.6586;
Clr = 0.1365;
Clda = -0.2729; % Note: Aileron sign conventions vary!
Cldr = 0;

Cn0 = 0;
Cnbeta = 0.1040;
Cnp = 0.0427;
Cnr = -0.1511;
Cnda = 0.0522;
Cndr = -0.0726;
Cnbeta3 = 0.1777;

% FIND EQUILIBRIUM FLIGHT CONDITIONS

% Longitudinal force and moment balance for wings-level equilibrium flight:
%
% 0 = X + T - m g sin(alpha)
% 0 = Z + m g cos(alpha)
% 0 = M
%
% Note that
%
% X = L sin(alpha) - D cos(alpha)
% Z = -L cos(alpha) - D sin(alpha)
%
% Set the airspeed and assume alpha << 1 to determine a seed value for the
% equilibrium pitch and elevator angles from the plunge and pitch balance:
%
% 0 = (Cz0 + Czalpha*alpha + Czde*de) + [(m g)/(P_dyn S)]
% 0 = (Cm0 + Cmalpha*alpha + Cmde*de)
%
% Solve the preceding two equations for equilibrium alpha and de
VEq = 45;               % Equilibrium speed (ft/s)
CW = W/((0.5*rho*(VEq^2))*S);   % Non-dimensional weight
temp = inv([Czalpha, Czde; Cmalpha, Cmde])*[-Cz0-CW;-Cm0];
alphaGuess = temp(1);
deGuess = temp(2);
clear temp

FSOLVEoptions = optimoptions('fsolve','OptimalityTolerance',1e-12);
Roots = fsolve('MTDWingsLevelEquilibrium',[alphaGuess;deGuess],FSOLVEoptions);
alphaEq = Roots(1);
thetaEq = alphaEq; % Constant-altitude flight: Zero climb angle
deEq = Roots(2);
clear FSOLVEoptions Roots

% Check plunge and pitch balance
%Cz0 + Czalpha*alphaEq + Czalpha2*(alphaEq^2) + Czde*deEq + CW*cos(alphaEq)
%Cm0 + Cmalpha*alphaEq + Cmalpha2*(alphaEq^2) + Cmalpha3*(alphaEq^3) + Cmde*deEq
```

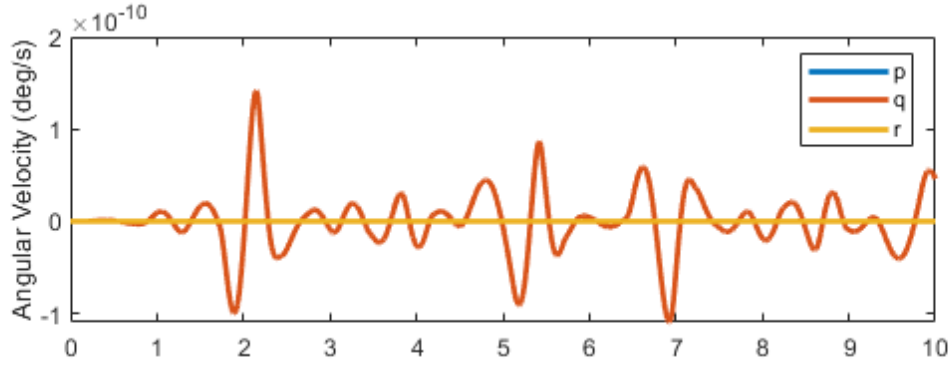
```
% Compute (constant) power for equilribium flight
CT0 = - (Cx0 + Cxalpha*alphaEq + Cxalpha2*(alphaEq^2) + Cxde*deEq) + CW*sin(alphaEq);
ThrustEq = CT0*(0.5*rho*(VEq^2))*S;
Power = ThrustEq*VEq;

% VERIFY EQUILIBRIUM FLIGHT CONDITIONS IN SIMULATION

% Define initial state
X0 = zeros(3,1); % (m)
Theta0 = [0;thetaEq;0]; % (rad)
V0 = [VEq*cos(thetaEq);0;VEq*sin(thetaEq)]; % (m/s)
omega0 = zeros(3,1); % (rad/s)

y0 = [X0; Theta0; V0; omega0];
t_final = 10;
ODE45options = odeset('RelTol',1e-10,'AbsTol',1e-12);
[t,y] = ode45('MTDEOM',[0:0.01:t_final]',y0,ODE45options);

% PLOT COMPUTED STATE HISTORIES

figure(1)
subplot(2,1,1)
plot(t,y(:,1:3),'LineWidth',2.0)
legend('X','Y','Z')
ylabel('Position (ft)')
subplot(2,1,2)
plot(t,y(:,4:6)*(180/pi),'LineWidth',2.0)
legend('\phi','\theta','\psi')
ylabel('Attitude (deg)')

figure(2)
subplot(2,1,1)
plot(t,y(:,7:9),'LineWidth',2.0)
legend('u','v','w')
ylabel('Velocity (ft/s)')
subplot(2,1,2)
plot(t,y(:,10:12)*(180/pi),'LineWidth',2.0)
ylabel('Angular Velocity (deg/s)')
legend('p','q','r')
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.
```




*Published with MATLAB® R2022a*

```
function F = MTDWingsLevelEquilibrium(x)

% MTDWingsLevelEquilibrium.m
%
% Static force and moment balance for pitch angle and elevator deflection
% corresponding to steady, wings level flight at constant speed and altitude.

global e1 e2 e3 rho m g S b c Inertia ...
Cx0 Cxalpha Cxalpha2 Cxde ...
Cz0 Czalpha Czalpha2 Czde ...
Cm0 Cmalpha Cmalpha2 Cmalpha3 Cmq Cmde ...
Cy0 Cybeta Cybeta3 Cyp Cyr Cyda Cydr ...
Cl0 Clbeta Clp Clr Clda Cldr ...
Cn0 Cnbeta Cnbeta3 Cnp Cnr Cnda Cndr ...
VEq alphaEq thetaEq deEq Power


% Solving for pitch angle and elevator deflection
alpha = x(1); % rad
theta = alpha; % Constant altitude: Zero climb angle
de = x(2); % rad

% Nondimensional force and moment coefficients in equilibrium flight
Cz = Cz0 + Czalpha*alpha + Czalpha2*(alpha^2) + Czde*de;
Cm = Cm0 + Cmalpha*alpha + Cmalpha2*(alpha^2) + Cmalpha3*(alpha^3) + Cmde*de;

% Nondimensional forces and moments balance to zero. (Find the roots.)
F(1) = Cz + (m*g*cos(alpha))/((0.5*rho*(VEq^2))*S);
F(2) = Cm;
```

```
function xdot = MTDEOM(t,x)

% MTDEOM.m
%
% Nonlinear equations of motion for the MTD small, fixed-wing UAV.

global e1 e2 e3 rho m g S b c Inertia ...
Cx0 Cxalpha Cxalpha2 Cxde ...
Cz0 Czalpha Czalpha2 Czde ...
Cm0 Cmalpha Cmalpha2 Cmalpha3 Cmq Cmde ...
Cy0 Cybeta Cybeta3 Cyp Cyr Cyda Cydr ...
Cl0 Clbeta Clp Clr Clda Cldr ...
Cn0 Cnbeta Cnbeta3 Cnp Cnr Cnda Cndr ...
VEq alphaEq thetaEq deEq Power


% Parse out state vector components
X = x(1:3);
Theta = x(4:6);
    phi = Theta(1);
    theta = Theta(2);
    psi = Theta(3);
V = x(7:9);
    u = V(1);
    v = V(2);
    w = V(3);
omega = x(10:12);
    p = omega(1);
    q = omega(2);
    r = omega(3);

    phat = (p*b)/(2*VEq);
    qhat = (q*c)/(2*VEq);
    rhat = (r*b)/(2*VEq);

RIB = expm(psi*hat(e3))*expm(theta*hat(e2))*expm(phi*hat(e1));
LIB = [1, sin(phi)*tan(theta), cos(phi)*tan(theta);
    0, cos(phi), -sin(phi);
    0, sin(phi)/cos(theta), cos(phi)/cos(theta)];

% Dynamic pressure
PDyn = 0.5*rho*norm(V)^2;

%%% AERODYNAMIC FORCES AND MOMENTS %%%

% Aerodynamic angles
beta = asin(v/norm(V));
alpha = atan2(w,u);

% Control deflections
de = deEq;
da = 0;
dr = 0;

% Aerodynamic force coefficients
Cx = Cx0 + Cxalpha*alpha + Cxalpha2*(alpha^2) + Cxde*de;
```

```
Cy = Cy0 + Cybeta*beta + Cybeta3*(beta^3) + Cyp*phat + Cyr*rhat + Cyda*da + Cydr*dr;
Cz = Cz0 + Czalpha*alpha + Czalpha2*(alpha^2) + Czde*de;

Thrust = Power/norm(V);  % Constant power
X = PDyn*S*Cx + Thrust;
Y = PDyn*S*Cy;
Z = PDyn*S*Cz;
Force_Aero = [X; Y; Z];

% Components of aerodynamic moment (modulo "unsteady" terms)
Cl = Cl0 + Clbeta*beta + Clp*phat + Clr*rhat + Clda*da + Cldr*dr;
Cm = Cm0 + Cmalpha*alpha + Cmalpha2*(alpha^2) + Cmalpha3*(alpha^3) + Cmq*qhat + Cmde*de;
Cn = Cn0 + Cnbeta*beta + Cnbeta3*(beta^3) + Cnp*phat + Cnr*rhat + Cnda*da + Cndr*dr;

L = PDyn*S*b*Cl;
M = PDyn*S*c*Cm;
N = PDyn*S*b*Cn;
Moment_Aero = [L; M; N];

% Sum of forces and moments
Force = RIB'*(m*g*e3) + Force_Aero;
Moment = Moment_Aero;

%%% EQUATIONS OF MOTION %%%

%%% KINEMATIC EQUATIONS %%%
XDot = RIB*V;
ThetaDot = LIB*omega;

%%% DYNAMIC EQUATIONS %%%
VDot = (1/m)*(cross(m*V,omega) + Force);
omegaDot = inv(Inertia)*(cross(Inertia*omega,omega) + Moment);

xdot = [XDot; ThetaDot; VDot; omegaDot];
```

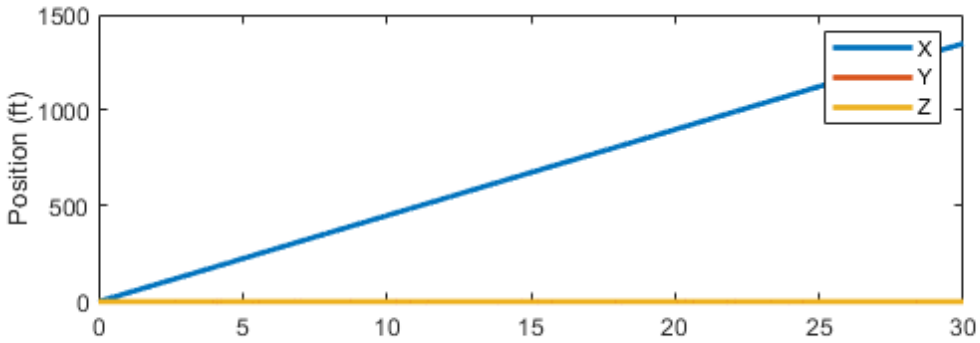
```
function xhat = hat(x)

xhat = [0,-x(3),x(2);x(3),0,-x(1);-x(2),x(1),0];
```

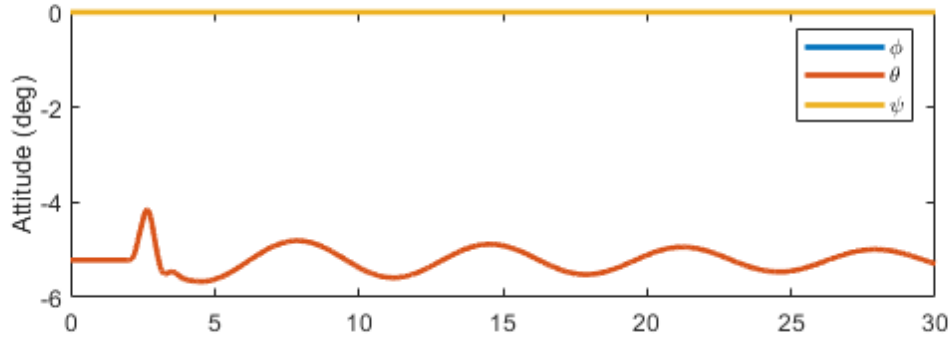
```
% MTD Pitch Doublet Simulation
% C. A. Woolsey

clear
close all

% GenericFixedWingScript.m solves the nonlinear equations of motion for a
% fixed-wing aircraft flying in ambient wind with turbulence.

global e1 e2 e3 rho m g S b c Inertia ...
Cx0 Cxalpha Cxalpha2 Cxde ...
Cz0 Czalpha Czalpha2 Czde ...
Cm0 Cmalpha Cmalpha2 Cmalpha3 Cmq Cmde ...
Cy0 Cybeta Cybeta3 Cyp Cyr Cyda Cydr ...
Cl0 Clbeta Clp Clr Clda Cldr ...
Cn0 Cnbeta Cnbeta3 Cnp Cnr Cnda Cndr ...
VEq alphaEq thetaEq deEq Power

% Basis vectors
e1 = [1;0;0];
e2 = [0;1;0];
e3 = [0;0;1];

% Atmospheric and gravity parameters (Constant altitude: Sea level)
rho = 2.3769e-3;    % Density (sl/ft^3)
g = 32.174;         % Gravitational acceleration (ft/s^2)

% Aircraft parameters (MTD)
m = 0.211;          % Mass (sl)
W = m*g;            % Weight (lb)
Ix = 0.2163;        % Roll inertia (sl-ft^2)
Iy = 0.1823;        % Pitch inertia (sl-ft^2)
Iz = 0.3396;        % Yaw inertia (sl-ft^2)
Ixz = 0.0364;       % Roll/Yaw product of inertia (sl-ft^2)
Inertia = [Ix, 0, -Ixz; 0, Iy, 0; -Ixz, 0, Iz];

S = 4.92;           % Wing area (ft^2)
b = 5.91;           % Wing span (ft)
c = 0.833;          % Wing chord (ft)
AR = (b^2)/S;       % Aspect ratio

% Longitudinal nondimensional stability and control derivatives
Cx0 = -0.061;
Cxalpha = 0.440;
Cxde = -0.140;
Cxalpha2 = 2.59;

Cz0 = -0.206;
Czalpha = -3.947;
Czde = -21.19;
Czalpha2 = -0.8207;

Cm0 = -0.061;
Cmalpha = -0.8068;
Cmq = -4.937;
Cmde = -0.7286;
Cmalpha2 = -1.251;
Cmalpha3 = -29.92;
```

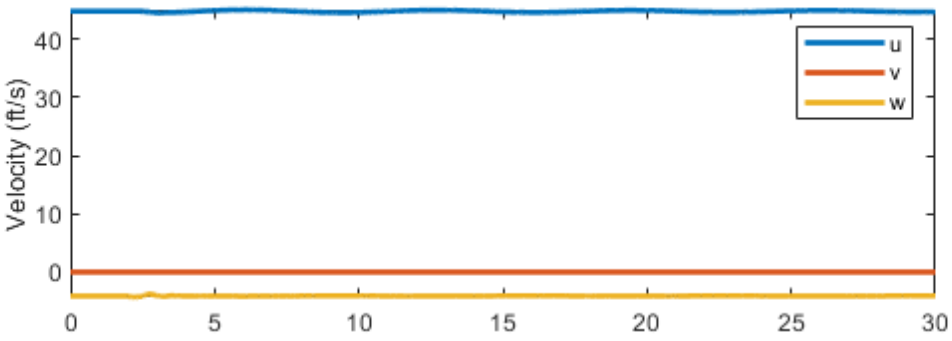
```
% Lateral-directional nondimensional stability and control derivatives
Cy0 = 0;
Cybeta = -0.3761;
Cyp = 0.6323;
Cyr = 0.1951;
Cyda = 0.2491;
Cydr = 0.1544;
Cybeta3 = -0.2294;

Cl0 = 0;
Clbeta = -0.0529;
Clp = -0.6586;
Clr = 0.1365;
Clda = -0.2729; % Note: Aileron sign conventions vary!
Cldr = 0;

Cn0 = 0;
Cnbeta = 0.1040;
Cnp = 0.0427;
Cnr = -0.1511;
Cnda = 0.0522;
Cndr = -0.0726;
Cnbeta3 = 0.1777;

% FIND EQUILIBRIUM FLIGHT CONDITIONS

% Longitudinal force and moment balance for wings-level equilibrium flight:
%
% 0 = X + T - m g sin(alpha)
% 0 = Z + m g cos(alpha)
% 0 = M
%
% Note that
%
% X = L sin(alpha) - D cos(alpha)
% Z = -L cos(alpha) - D sin(alpha)
%
% Set the airspeed and assume alpha << 1 to determine a seed value for the
% equilibrium pitch and elevator angles from the plunge and pitch balance:
%
% 0 = (Cz0 + Czalpha*alpha + Czde*de) + [(m g)/(P_dyn S)]
% 0 = (Cm0 + Cmalpha*alpha + Cmde*de)
%
% Solve the preceding two equations for equilibrium alpha and de
VEq = 45;               % Equilibrium speed (ft/s)
CW = W/((0.5*rho*(VEq^2))*S);   % Non-dimensional weight
temp = inv([Czalpha, Czde; Cmalpha, Cmde])*[-Cz0-CW;-Cm0];
alphaGuess = temp(1);
deGuess = temp(2);
clear temp

FSOLVEoptions = optimoptions('fsolve','OptimalityTolerance',1e-12);
Roots = fsolve('MTDWingsLevelEquilibrium',[alphaGuess;deGuess],FSOLVEoptions);
alphaEq = Roots(1);
thetaEq = alphaEq; % Constant-altitude flight: Zero climb angle
deEq = Roots(2);
clear FSOLVEoptions Roots

% Check plunge and pitch balance
%Cz0 + Czalpha*alphaEq + Czalpha2*(alphaEq^2) + Czde*deEq + CW*cos(alphaEq)
%Cm0 + Cmalpha*alphaEq + Cmalpha2*(alphaEq^2) + Cmalpha3*(alphaEq^3) + Cmde*deEq
```

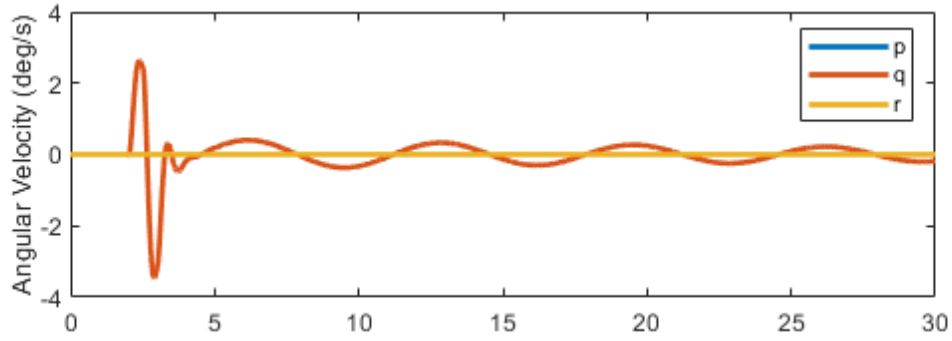
```
% Compute (constant) power for equilribium flight
CT0 = - (Cx0 + Cxalpha*alphaEq + Cxalpha2*(alphaEq^2) + Cxde*deEq) + CW*sin(alphaEq);
ThrustEq = CT0*(0.5*rho*(VEq^2))*S;
Power = ThrustEq*VEq;

% VERIFY EQUILIBRIUM FLIGHT CONDITIONS IN SIMULATION

% Define initial state
X0 = zeros(3,1); % (m)
Theta0 = [0;thetaEq;0]; % (rad)
V0 = [VEq*cos(thetaEq);0;VEq*sin(thetaEq)]; % (m/s)
omega0 = zeros(3,1); % (rad/s)

y0 = [X0; Theta0; V0; omega0];
t_final = 30;
ODE45options = odeset('RelTol',1e-10,'AbsTol',1e-12);
[t,y] = ode45('MTDDoubletEOM',[0:0.01:t_final]',y0,ODE45options);

% PLOT COMPUTED STATE HISTORIES

figure(1)
subplot(2,1,1)
plot(t,y(:,1:3),'LineWidth',2.0)
legend('X','Y','Z')
ylabel('Position (ft)')
subplot(2,1,2)
plot(t,y(:,4:6)*(180/pi),'LineWidth',2.0)
legend('\phi','\theta','\psi')
ylabel('Attitude (deg)')

figure(2)
subplot(2,1,1)
plot(t,y(:,7:9),'LineWidth',2.0)
legend('u','v','w')
ylabel('Velocity (ft/s)')
subplot(2,1,2)
plot(t,y(:,10:12)*(180/pi),'LineWidth',2.0)
ylabel('Angular Velocity (deg/s)')
legend('p','q','r')
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.
```




*Published with MATLAB® R2022a*

```
function xdot = MTDDoubletEOM(t,x)

% MTDEOM.m
%
% Nonlinear equations of motion for the MTD small, fixed-wing UAV.

global e1 e2 e3 rho m g S b c Inertia ...
Cx0 Cxalpha Cxalpha2 Cxde ...
Cz0 Czalpha Czalpha2 Czde ...
Cm0 Cmalpha Cmalpha2 Cmalpha3 Cmq Cmde ...
Cy0 Cybeta Cybeta3 Cyp Cyr Cyda Cydr ...
Cl0 Clbeta Clp Clr Clda Cldr ...
Cn0 Cnbeta Cnbeta3 Cnp Cnr Cnda Cndr ...
VEq alphaEq thetaEq deEq Power


% Parse out state vector components
X = x(1:3);
Theta = x(4:6);
    phi = Theta(1);
    theta = Theta(2);
    psi = Theta(3);
V = x(7:9);
    u = V(1);
    v = V(2);
    w = V(3);
omega = x(10:12);
    p = omega(1);
    q = omega(2);
    r = omega(3);

    phat = (p*b)/(2*VEq);
    qhat = (q*c)/(2*VEq);
    rhat = (r*b)/(2*VEq);

RIB = expm(psi*hat(e3))*expm(theta*hat(e2))*expm(phi*hat(e1));
LIB = [1, sin(phi)*tan(theta), cos(phi)*tan(theta);
    0, cos(phi), -sin(phi);
    0, sin(phi)/cos(theta), cos(phi)/cos(theta)];

% Dynamic pressure
PDyn = 0.5*rho*norm(V)^2;

%%% AERODYNAMIC FORCES AND MOMENTS %%%

% Aerodynamic angles
beta = asin(v/norm(V));
alpha = atan2(w,u);

% Control deflections
if or(t<2,t>=3)
    de = deEq;
elseif and(t>=2,t<2.5)
    de = deEq + 0.1*(pi/180);
elseif and(t>=2.5,t<3)
    de = deEq - 0.1*(pi/180);
```

```
end
da = 0;
dr = 0;

% Aerodynamic force coefficients
Cx = Cx0 + Cxalpha*alpha + Cxalpha2*(alpha^2) + Cxde*de;
Cy = Cy0 + Cybeta*beta + Cybeta3*(beta^3) + Cyp*phat + Cyr*rhat + Cyda*da + Cydr*dr;
Cz = Cz0 + Czalpha*alpha + Czalpha2*(alpha^2) + Czde*de;

Thrust = Power/norm(V);  % Constant power
X = PDyn*S*Cx + Thrust;
Y = PDyn*S*Cy;
Z = PDyn*S*Cz;
Force_Aero = [X; Y; Z];

% Components of aerodynamic moment (modulo "unsteady" terms)
Cl = Cl0 + Clbeta*beta + Clp*phat + Clr*rhat + Clda*da + Cldr*dr;
Cm = Cm0 + Cmalpha*alpha + Cmalpha2*(alpha^2) + Cmalpha3*(alpha^3) + Cmq*qhat + Cmde*de;
Cn = Cn0 + Cnbeta*beta + Cnbeta3*(beta^3) + Cnp*phat + Cnr*rhat + Cnda*da + Cndr*dr;

L = PDyn*S*b*Cl;
M = PDyn*S*c*Cm;
N = PDyn*S*b*Cn;
Moment_Aero = [L; M; N];

% Sum of forces and moments
Force = RIB'*(m*g*e3) + Force_Aero;
Moment = Moment_Aero;

%%% EQUATIONS OF MOTION %%%

%%% KINEMATIC EQUATIONS %%%
XDot = RIB*V;
ThetaDot = LIB*omega;

%%% DYNAMIC EQUATIONS %%%
VDot = (1/m)*(cross(m*V,omega) + Force);
omegaDot = inv(Inertia)*(cross(Inertia*omega,omega) + Moment);

xdot = [XDot; ThetaDot; VDot; omegaDot];
```