ASE 367K Term Project (2024.12.05)
Bonsuck Koo(bk22569), Kyeongchan Choi(kc48336)

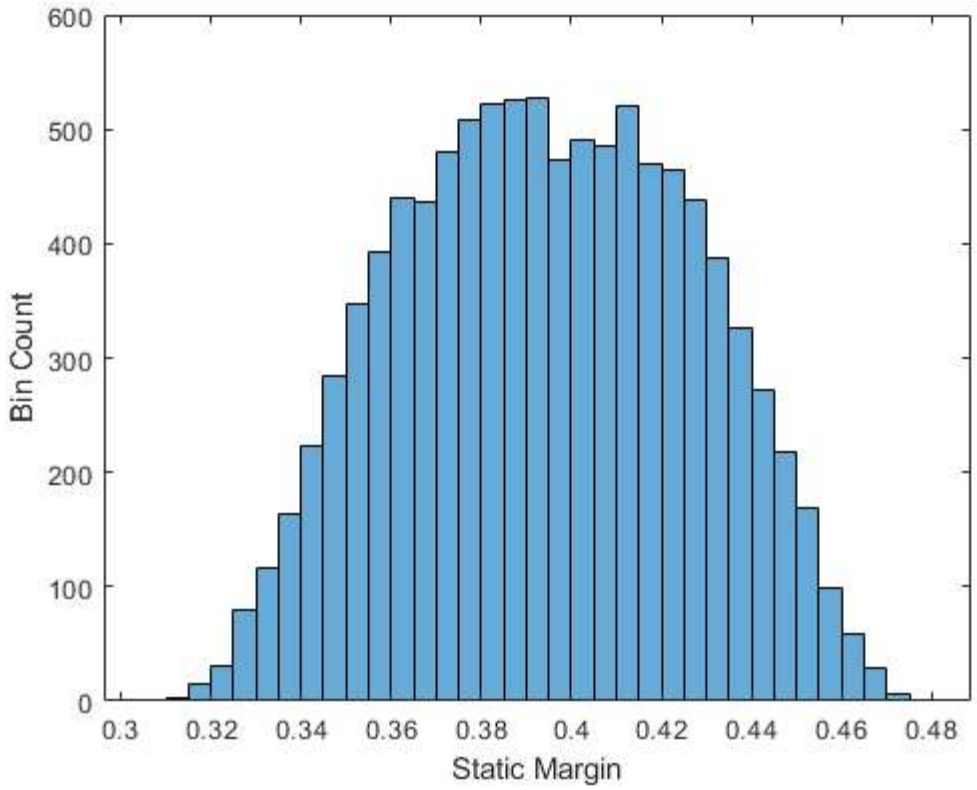
Contents


```
close all; clear all; clc
```

Determine CG and CP

```
N_sample = 10000;
static_margins = zeros([N_sample,1]);
for i = 1:N_sample
    [cg, mass_total, moment] = cg_sample();
    [cp, C_N_alpha, S_ref, cp_moment] = cp_sample();
    static_margins(i) = (cg-cp)/0.4;
    %derived properties
    C_N_q = C_N_alpha*(cg-cp);
    C_M_alpha = -C_N_alpha*(cg-cp);
    C_M_q = -(C_N_alpha*cp_moment +2*C_N_alpha*cp*cg-C_N_alpha*cg.^2);
end

figure(1)
clf
mean(static_margins)
std(static_margins)
histogram(static_margins)
xlabel('Static Margin')
ylabel('Bin Count')
```

```
ans =

    0.3945


ans =

    0.0323
```

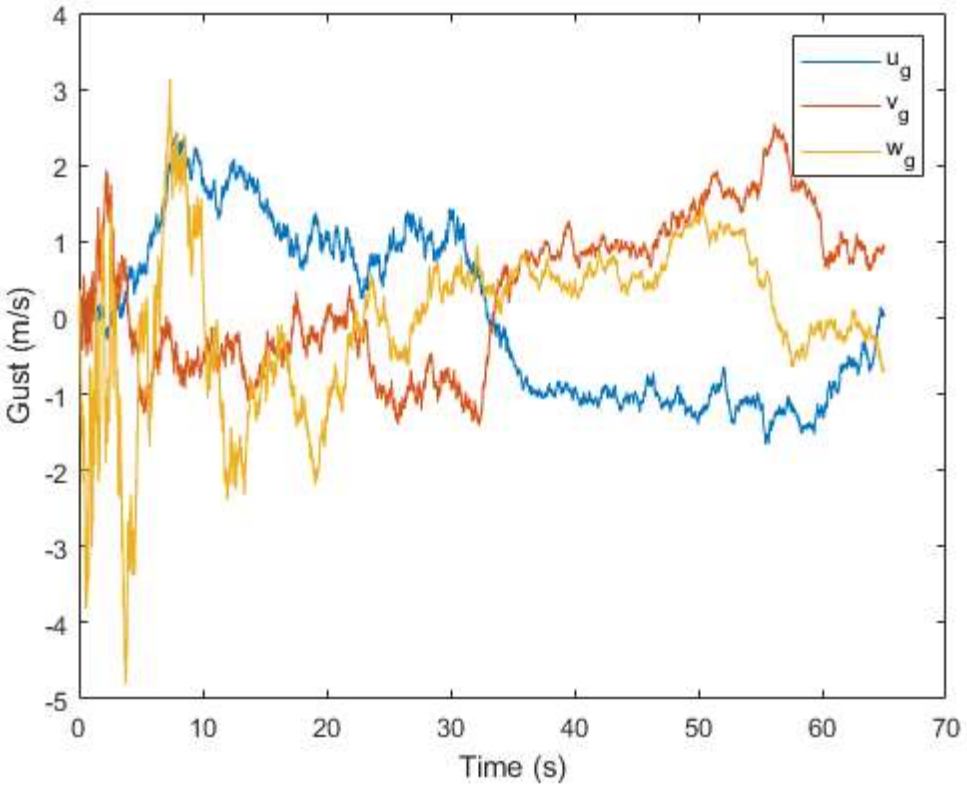


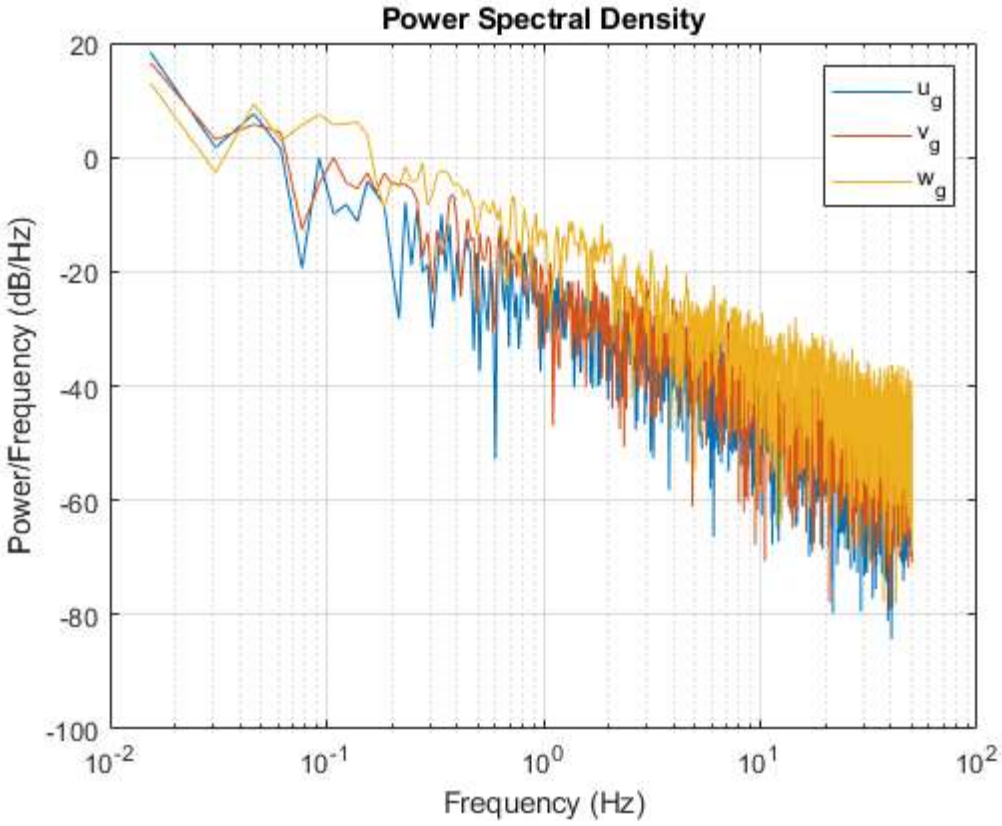
Wind Model

```
%simple gust sampling example with a vertically accelerating vehicle
N_timestep = 6500;
dt = 0.01;
accel = 1;
gust_state = [0,0,0]';
gustintensity = 1;
gustdata = zeros(3,N_timestep);
V = 0;
h = 0;
for t = 1:N_timestep
    gust_state = dryden_gust_sample(gust_state,V,h,gustintensity,dt);
    gustdata(:,t) = gust_state;
    V = V + accel*dt;
    h = h + V*dt + 0.5*accel*dt.^2;
end

%plot time series
figure(2)
clf
plot((1:N_timestep)*dt, gustdata(1:3,:)' )
legend('u_g','v_g','w_g')
xlabel('Time (s)')
ylabel('Gust (m/s)')

%plot Power Spectral Densities
fs = 1/(dt);
xdft = fft(gustdata(1,:)');
xdft = xdft(1:N_timestep/2+1);
psdx = (1/(fs*N_timestep)) * abs(xdft).^2;
```

```
psdx(2:end-1) = 2*psdx(2:end-1);
freq = 0:fs/N_timestep:fs/2;
figure(3)
clf
semilogx(freq,10*log10(psdx) )
grid on
hold on
title("Power Spectral Density")
xlabel("Frequency (Hz)")
ylabel("Power/Frequency (dB/Hz)")
legend('u_g','v_g','w_g')
xdft = fft(gustdata(2,:)');
xdft = xdft(1:N_timestep/2+1);
psdx = (1/(fs*N_timestep)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);
freq = 0:fs/N_timestep:fs/2;
semilogx(freq,10*log10(psdx) )
xdft = fft(gustdata(3,:)');
xdft = xdft(1:N_timestep/2+1);
psdx = (1/(fs*N_timestep)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);
freq = 0:fs/N_timestep:fs/2;
semilogx(freq,10*log10(psdx) )
legend('u_g','v_g','w_g')
```

```
Warning: Ignoring extra legend entries.
```

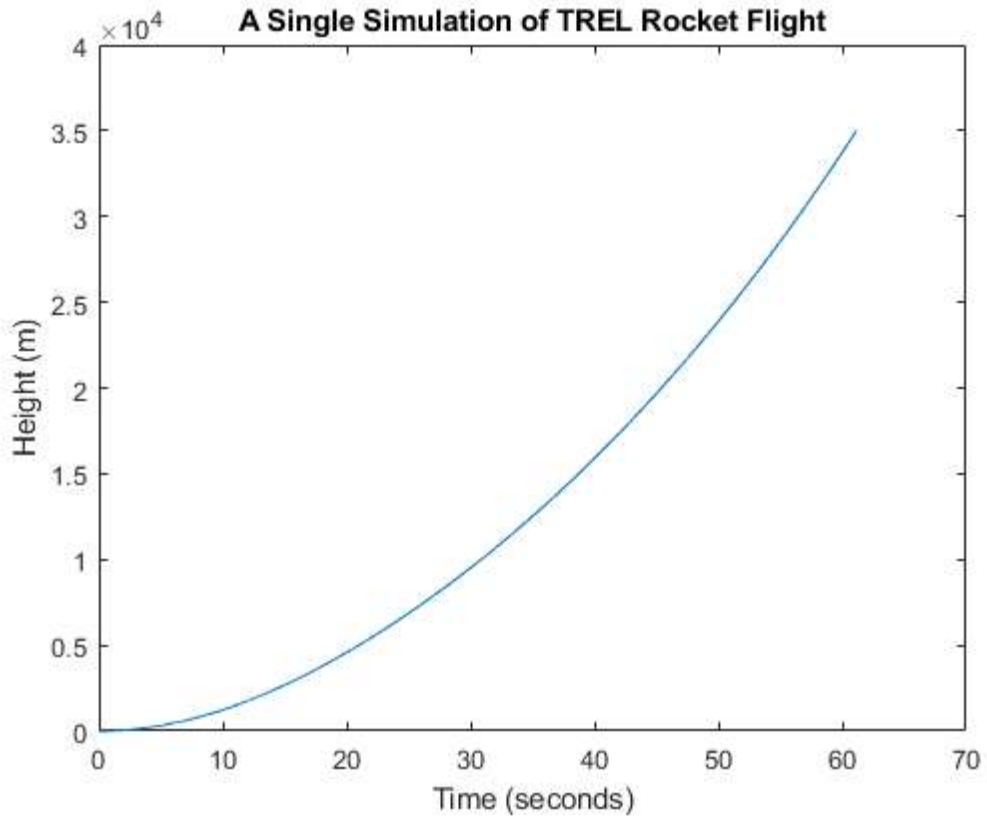




Simulate Flight

```
%---- Set Up
state  = zeros(N_timestep,9);
params = zeros(1,10);
```

```
% RocketParameters
params(1) = mass_total;    % Mass
params(2) = moment;   % Ixx
params(3) = S_ref;    % Area
params(4) = C_N_alpha;    % CNa
params(5) = C_N_q;    % CN_q
params(6) = C_M_alpha;    % CM_a
params(7) = C_M_q;    % CM_q
params(8) = 0.5;    % CD_
params(9) = 4.5;    % Gust Intensity
params(10) = cg;   % Gimbal CG
% Initial Conditions
stateInit  = zeros(1,size(state,2));
statek     = stateInit;


for k = 1:N_sample
    [statek,~]=vehicle_dynamics(statek,params,dt);
    state(k,:) = statek;
    if statek(2) >= 35000
        break
    end
end
figure,
plot((1:k)*dt,state((1:k),2))
title('A Single Simulation of TREL Rocket Flight')
xlabel('Time (seconds)')
ylabel('Height (m)')
```

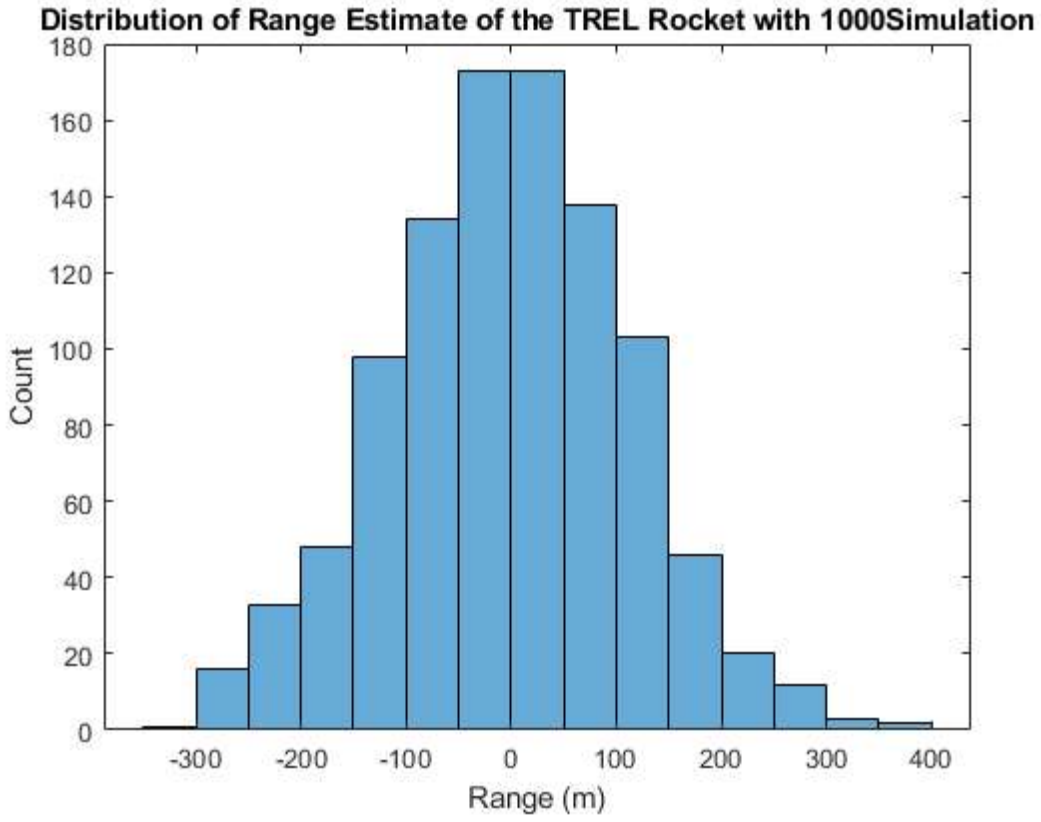


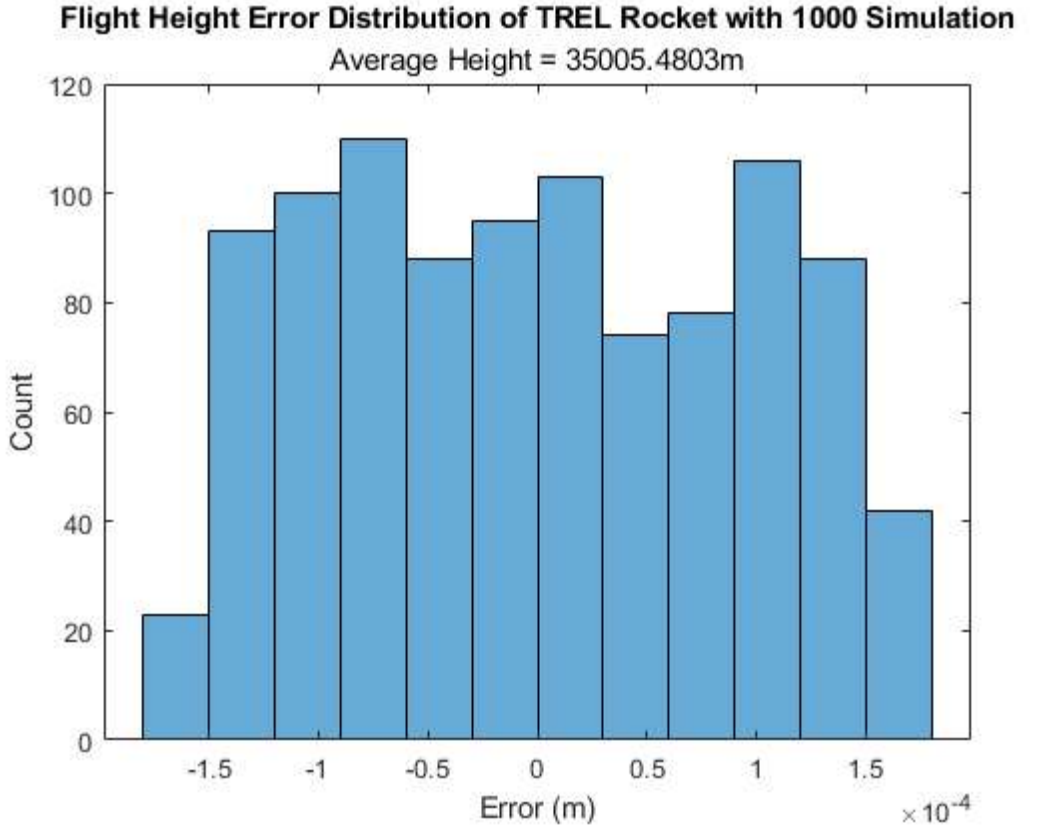
Monte Carlo Sim

```
ensemble= 1000;
X = zeros(ensemble,1);
Z = zeros(ensemble,1);
for m = 1:ensemble
    %---- GET CG AND CP
    [cg, mass_total, moment] = cg_sample();
    [cp, C_N_alpha, S_ref, cp_moment] = cp_sample();
    static_margins(i) = (cg-cp)/0.4;
    %derived properties
    C_N_q = C_N_alpha*(cg-cp);
    C_M_alpha = -C_N_alpha*(cg-cp);
    C_M_q = -(C_N_alpha*cp_moment +2*C_N_alpha*cp*cg-C_N_alpha*cg.^2);
    %---- Get Gust Parameter
    gust_state = dryden_gust_sample(gust_state,V,h,gustintensity,dt);
    gustdata(:,t) = gust_state;
    V = V + accel*dt;
    h = h + V*dt + 0.5*accel*dt.^2;
    %---- Set Up
    state  = zeros(N_timestep,9);
    params = zeros(1,10);
    % RocketParameters
    params(1) = mass_total;    % Mass
    params(2) = moment;   % Ixx
    params(3) = S_ref;    % Area
    params(4) = C_N_alpha;    % CNa
    params(5) = C_N_q;    % CN_q
    params(6) = C_M_alpha;    % CM_a
    params(7) = C_M_q;    % CM_q
    params(8) = 0.5;    % CD_
    params(9) = 4.5;    % Gust Intensity
    params(10) = cg;   % Gimbal CG
    % Initial Conditions
    stateInit  = zeros(1,size(state,2));
    statek     = stateInit;

    for k = 1:N_timestep
        [statek,~]=vehicle_dynamics(statek,params,dt);
        if statek(2) >= 35000
            X(m) = statek(1);
            Z(m) = statek(2);
            break
        end
    end
end

histogram(X)
title(['Distribution of Range Estimate of the TREL Rocket with ',num2str(ensemble), 'Simulation'])
xlabel('Range (m)');
ylabel('Count')
avgZ=mean(Z);

figure
histogram((Z-avgZ)/avgZ)
title(['Flight Height Error Distribution of TREL Rocket with ',num2str(ensemble), ' Simulation' ])
subtitle(['Average Height = ', num2str(avgZ), 'm'])
xlabel('Error (m)')
ylabel('Count')
```

```
fprintf(['The rocket is stable\n'])
```

```
The rocket is stable
```





Published with MATLAB® R2023a

```
function [cg, mass_total, moment] = cg_sample()
%point mass mean locations
x0 =[0,0.3,0.7,1.1,1.2,2.2,2.3,2.9,3.1,3.3,3.4,4,4.2,4.3,4.9,6,6.5,6.7,6.9,7.1,7.6];
%uniform sample location errors
x_err = 0.002*(1-2*rand(size(x0)));
%point mass means
mass0 = [0,20,25,25,20,25,70,1,10,15,1,20,120,1,50,5,1,5,5,10,10];
%uniform sample mass errors
mass_err = 0.1*(1-2*rand(size(x0)));
%add errors to means
x = x0 + x_err;
mass = mass0 + mass_err;
%compute properties
mass_total = sum(mass);
cg = sum(mass.*x)/mass_total;
moment = sum( mass.*(x-cg).^2 );
end
```

```
ans =

    3.3329
```

Published with MATLAB® R2023a

```
function [cp, C_N_alpha, S_ref, cp_moment] = cp_sample()
% This functiion determines the center of pressure of TREL rocket, which
% dimensions are provided in the ASE 367K Flight Dynamics Term Project
%
%
% ---- Assumptions:
%       1. Dimensions start from the nose
%       2. At sea level

%sample errors
errors = 0.002*(1-2*rand([1,9]));
%aerodynamic geometry
body_diameter = 0.4 + errors(1);
nosecone_length = 1.2 + errors(2);
fin_diameter = (0.8 + errors(3))*sqrt(2);
fin_tip_chord = 0.7 + errors(4);
fin_root_chord = 0.8 + errors(5);
fin_station = 1.2 + errors(6);
body_length = 6 + errors(7);
nozzle_length = 0.8 + errors(8);
nozzle_diameter = 0.25 + errors(9);

%nosecone properties
C_N_nosecone_alpha = 2;
x_nosecone = nozzle_length+body_length+(1-0.466)*nosecone_length;
C_N_body_alpha = 0;%low angle of attack

%fin properties
N_fin = 4;
f = 1;%Interference coefficient for 3,4 fins
radius = body_diameter / 2;
span = (fin_diameter-body_diameter)/2;
C_N_fin_alpha = (1+f*(radius)./(span+radius))*((4*N_fin*(span/body_diameter).^2)...
/(1+sqrt(1+(2*span/(fin_root_chord+fin_tip_chord)).^2) ) );
x_R = (fin_root_chord-fin_tip_chord)/2;
x_fin = x_R/3*(fin_root_chord+2*fin_tip_chord)/(fin_root_chord+fin_tip_chord) +...
1/6*((fin_root_chord+fin_tip_chord)-(fin_root_chord*fin_tip_chord)/...
((fin_root_chord+fin_tip_chord)));
x_fin = fin_station+nozzle_length-x_fin;

% Bonical Boattail
% CP of Conical transition XT
Xp = 7.2;               % m; Nose to d1
LT = 0.8;               % m; d1 to d2
XT = Xp + LT/3*(1+(1-body_diameter/nozzle_diameter)/(1-(body_diameter/nozzle_diameter)^2));
rocket_length = 1.2+6+0.8;
XT = rocket_length - XT;
S1 = pi*body_diameter^2/4;
S2 = pi*nozzle_diameter^2/4;
CNaCB = 8/(pi*body_diameter^2)*(S2-S1);

%combined properties
C_N_alpha = C_N_fin_alpha + C_N_nosecone_alpha +CNaCB;
cp = (x_fin*C_N_fin_alpha + x_nosecone*C_N_nosecone_alpha+CNaCB*(XT))/C_N_alpha;
S_ref = 0.25*pi*body_diameter.^2;
cp_moment = (x_fin.^2*C_N_fin_alpha + x_nosecone.^2*C_N_nosecone_alpha+(XT).^2*CNaCB)/C_N_alpha;
```

```
ans =

    3.1841
```

Published with MATLAB® R2023a

```
function gststate_new = dryden_gust_sample(gust_state,V,h,gustintensity,dt)
%gust state: [u_g,v_G,w_g] in m/s
%V: total airspeed in m/s
%h: altitude above ground in m
%gustintensity: gust_rms in m/s (1.5 m/s light, 3 m/s moderate, 4.5 m/s severe)
%dt: time step in s
whitenoise = randn([3,1]);
gustu = gust_state(1);
gustv = gust_state(2);
gustw = gust_state(3);
%rotated such that x vector is pointed directly upwards
hft = h*3.28084;          % Unit conversion from m to ft
hft = max(hft,1);
if hft<1000
sigmaw = gustintensity/((0.177+0.000823*hft).^(0.4));
sigmav = sigmaw;
sigmau = gustintensity;
Lu = (hft/(0.177+0.000823*hft).^1.2)/3.28084;
Lv = Lu;
Lw = hft/3.28084;
elseif hft>2000
sigmaw = gustintensity;
sigmau = gustintensity;
sigmav = gustintensity;
Lu = 1750;
Lv = Lu;
Lw = Lu;
else
sigmaw = gustintensity;
sigmau = gustintensity;
sigmav = gustintensity;
Lu = (1000 + 0.75*(hft-1000))/3.28084;
Lv = Lu;
Lw = Lu;
end
au = V/Lu;
Cu = exp(-au*dt);
av = V/Lv;
Cv = exp(-av*dt);
aw = V/Lw;
Cw = exp(-aw*dt);
%simplified form from MIL-F-8785C
gustu = Cu*gustu + sqrt((1-Cu.^2))*sigmau*whitenoise(1);
gustv = Cv*gustv + sqrt((1-Cv.^2))*sigmav*whitenoise(2);
gustw = Cw*gustw + sqrt((1-Cw.^2))*sigmaw*whitenoise(3);
gststate_new = [gustu;gustv;gustw];
end
```

```
Not enough input arguments.

Error in dryden_gust_sample (line 8)
gustu = gust_state(1);
```

Published with MATLAB® R2023a

```
function [state,aux] = vehicle_dynamics(state,params,dt)
    x = state(1);
    z = state(2);
    vx = state(3);
    vz = state(4);
    theta = state(5);
    q = state(6);

    gust_state = state(7:9);

    mass = params(1);
    Ixx = params(2);
    A = params(3);
    cna = params(4);
    cnq = params(5);
    cma = params(6);
    cmq = params(7);
    cd = params(8);
    gustintensity = params(9);
    gimbalcg = params(10);

    R = 6371000;
    mu = 3.986e14;
    T_max = 15500;

    fpa = atan2(vx,vz);

    r = sqrt((z+R).^2+(x).^2);
    h = r-R;
    psi = atan2(x,z+R);

    rho = 1.225*exp(-h/10400);

    g = mu/r.^2;

    gust_state = dryden_gust_sample(gust_state,sqrt(vx.^2+vz.^2),h,gustintensity,dt);
    gust_x = gust_state(1);
    gust_w = gust_state(2);

    u = vz*cos(theta) + vx*sin(theta)-gust_x;
    w = vx*cos(theta) - vz*sin(theta)-gust_w;

    alpha = atan2(-w,u);

    Q = 0.5*rho*(u.^2+w.^2);

    N = Q*A*(cna*alpha+cnq*q);
    D = Q*A*cd;

    T = 15500;

    delta_gimbal = 0;

    Fx = T*cos(delta_gimbal) - D;
    Fz = T*sin(delta_gimbal) + N;
```

```
    ax = (Fz*cos(theta)+Fx*sin(theta))/mass - g*sin(psi);
    az = (Fx*cos(theta)-Fz*sin(theta))/mass - g*cos(psi);

    M = Q*A*(cma*alpha)/Ixx + (exp(Q*A*cmq/Ixx*dt)-1)*q/dt - gimbalcg*sin(delta_gimbal)*T/Ixx;

    M1 = Q*A*(cma*alpha)/Ixx;
    M2 = (exp(Q*A*cmq/Ixx*dt)-1)*q/dt;

    dx = [vx+0.5*ax*dt;vz+0.5*az*dt; ax; az; q; M; 0; 0; 0];

    state = state+dx'*dt;
    state(7:9) = gust_state;
    aux = [alpha,rho,Q,ax,az,psi,fpa,delta_gimbal,M1,M2];

end
```

```
Not enough input arguments.

Error in vehicle_dynamics (line 2)
    x = state(1);
```

Published with MATLAB® R2023a