Table of Contents


```
clc
clear
close all
```

Aero force and moments simulink function tester

```
vB_BfromE_mps = [100;0;0]; % velocity of 100 m/s forward
vB_AfromN = [0;0;0]; % zero gust
C_BfromN = angle2dcm(0,0,0); % no rotation
rho = 1.225;
mass = 0.48; %kg

total_time = 10; %seconds
dt = 0.01; % timestep
time = 0:dt:total_time;
```

plotting variable allocation

```
drags            = time;
lifts            = time;
side_forces      = time;
alphas           = [time;time;time;time];
C_Ls             = [time;time;time;time];
lifts_surfs_b    = [time;time;time;time];
lifts_surfs_w    = [time;time;time;time];
airspeeds       = [time;time;time];
```

run simple simulation

```
for ii = 1:1:length(time)
```

calculate aerodynamic forces

```
   [total_force_b_N, total_moment_b_Nm, C_Ms_surfs, C_Ds_surfs] =...
aero_forces_and_moments(vB_BfromE_mps, vB_AfromN, C_BfromN, rho);
Get mass properties
```

```matlab
[cg_m, ~, ~] = get_mass_props;

% TODO think of a better way to do this
% Get aerodynamic design parameters
[surfs, aero] = const.aero_design_params;

% Incidence angles of each surface
i_surfs = aero(:, 3)';

% Get parameters of the planform
% These are lists with the order: elevator, rudder, left wing, right
% wing.
[surf_pos_m, c_m, ~, S_m2, AR] = get_planform_params;

% The airspeed of the plane, taking gust speed into account. Used to
% calculate angle of attack of each individual control surface.
vB_BfromA_mps = vB_BfromE_mps + C_BfromN*vB_AfromN;
v_inf = norm(vB_BfromA_mps); % QUESTION: is this true??

% Allocate array for angles of attack
alpha_surfs = zeros(1,4);

%TODO: try this without the for loop
% use the formula from lecture 6 to calculate each surface's AoA
for i = 1:1:length(aero(:,1))
    dotprod = dot(vB_BfromA_mps,surfs(:,i));
    alpha_surfs(i) = i_surfs(i) - asin(dotprod/norm(vB_BfromA_mps));
end

% Coefficient slopes
C_La_surfs = 2*pi.*AR./(2+AR); % 3D lift slope approx. for each surface
C_Da_surfs = aero(:,5)';
C_Ma_surfs = aero(:,8)';

% Coefficients at 0 AoA for each surface
C_L0_surfs = aero(:,1);
C_D0_surfs = aero(:,4);
C_M0_surfs = aero(:,7);

% AoA of 0 drag
a0_surfs = aero(:,6)';

% Calculate all coefficients for each aerodynamic surface
C_Ls_surfs = C_L0_surfs' + C_La_surfs.*alpha_surfs;
C_Ds_surfs = C_D0_surfs' + C_Da_surfs.*(alpha_surfs - a0_surfs).^2 +...
    C_Ls_surfs.^2./(pi.*aero(:,2)'.*AR);
C_Ms_surfs = C_M0_surfs' + C_Ma_surfs.*alpha_surfs;

% Calculate lift and drag of each surface. These are scalars.
lifts_surfs_N       = 0.5*rho*v_inf^2.*C_Ls_surfs.*S_m2;
drags_surfs_N       = 0.5*rho*v_inf^2.*C_Ds_surfs.*S_m2;

% Moments due to the surfaces themselves (different from moments_s_Nm)
moments_surfs_Nm    = 0.5*rho*v_inf^2.*C_Ms_surfs.*S_m2.*c_m;
```

```
% Initialize total force and moment
total_force_b_N       = [0;0;0];
total_moment_b_Nm     = [0;0;0];
utility_matrix  = [0,0,0; 0,0,-1; 0,1,0]; % for moment calculation
for i = 1:1:length(aero(:,1))
    % Calculate net force vector on the surface in wind frame
    force_surf_w = lifts_surfs_N(i)*surfs(:,i) - [drags_surfs_N(i);0;0];

    % Calculate net force vector on the surface in body frame
    force_surf_b = C_BfromN*force_surf_w;



    % Some extra post processing!!!
    lifts_surfs_w(i,ii) = force_surf_w(3);
    lifts_surfs_b(i,ii) = force_surf_b(3);
    % no more extra post processing!!!



    % Calculate net moment vector on the surface in the wind frame.
    % Moments on the surfaces (different from moments_surfs_Nm)
    moment_s_Nm = moments_surfs_Nm(i)*utility_matrix*surfs(:,i) + ...
        cross((surf_pos_m(i,:) - cg_m), force_surf_b)';

    % Calculate net moment vector on the surface in the body frame
    % NOTE: Theyre the same because of surface vector definition
    moment_b_Nm = moment_s_Nm;

    % Total the body forces and moments
    total_force_b_N       = total_force_b_N + force_surf_b;
    total_moment_b_Nm     = total_moment_b_Nm + moment_b_Nm;
end

C_D_fuse        = 0.5;
S_fuse          = 0.001;
total_force_b_N = total_force_b_N - [0.5*rho*v_inf^2*C_D_fuse*S_fuse;0;0];
```

integrate velocity

```
acceleration    = total_force_b_N./mass;
vB_BfromE_mps   = vB_BfromE_mps + acceleration*dt;
```

save variables for plotting

```
drags(ii)           = total_force_b_N(1); % seems correct
lifts(ii)           = total_force_b_N(3); % seems wrong
side_forces(ii)     = total_force_b_N(2); % seems correct
alphas(:, ii)       = alpha_surfs'; % seems wrong
C_Ls(:, ii)         = C_Ls_surfs'; % seems wrong
airspeeds(:,ii)     = vB_BfromA_mps;
```

```
end
```

post processing

figure() plot(time, drags); title("drag force time history (negative is backwards)");

figure() plot(time, lifts); title("lift force time history (negative is upwards)");

figure() plot(time, side_forces); title("side force time history (should be zero when flying straight)");

figure() plot(time, alphas(1,:)); hold on plot(time, alphas(2,:)); hold on plot(time, alphas(3,:)); hold on plot(time, alphas(4,:)); title("angles of attack (wings should be positive, elevator could be negative, rudder should be 0"); legend(["elevator", "rudder", "left wing", "right wing"]);

figure() plot(time, C_Ls(1,:)); hold on plot(time, C_Ls(2,:)); hold on plot(time, C_Ls(3,:)); hold on plot(time, C_Ls(4,:)); title("lift coefficients (wings should be positive, elevator could be negative, rudder should be 0"); legend(["elevator", "rudder", "left wing", "right wing"]);

figure() plot(time, lifts_surfs_w(1,:)); hold on plot(time, lifts_surfs_w(2,:)); hold on plot(time, lifts_surfs_w(3,:)); hold on plot(time, lifts_surfs_w(4,:)); title("lifts for each surface in the wind frame"); legend(["elevator", "rudder", "left wing", "right wing"]);

figure() plot(time, lifts_surfs_b(1,:)); hold on plot(time, lifts_surfs_b(2,:)); hold on plot(time, lifts_surfs_b(3,:)); hold on plot(time, lifts_surfs_b(4,:)); title("lifts of each surface in the body frame"); legend(["elevator", "rudder", "left wing", "right wing"]);

figure() plot(time, airspeeds(1,:)); hold on plot(time, airspeeds(2,:)); hold on plot(time, airspeeds(3,:)); title("airspeed xyz (FRD) components"); legend(["forward", "right", "down"]);

Simulation function

```
function [total_force_b_N, total_moment_b_Nm, C_Ms_surfs, C_Ds_surfs] =...
    aero_forces_and_moments(vB_BfromE_mps, vB_AfromN, C_BfromN, rho)


    % Get mass properties
    [cg_m, ~, ~] = get_mass_props;

    % TODO think of a better way to do this
    % Get aerodynamic design parameters
    [surfs, aero] = const.aero_design_params;

    % Incidence angles of each surface
    i_surfs = aero(:, 3)';

    % Get parameters of the planform
    % These are lists with the order: elevator, rudder, left wing, right
    % wing.
    [surf_pos_m, c_m, ~, S_m2, AR] = get_planform_params;

    % The airspeed of the plane, taking gust speed into account. Used to
    % calculate angle of attack of each individual control surface.
    vB_BfromA_mps = vB_BfromE_mps + C_BfromN*vB_AfromN;
    v_inf = norm(vB_BfromA_mps); % QUESTION: is this true??
```

```
% Allocate array for angles of attack
alpha_surfs = zeros(1,4);

%TODO: try this without the for loop
% use the formula from lecture 6 to calculate each surface's AoA
for i = 1:1:length(aero(:,1))
    dotprod = dot(vB_BfromA_mps,surfs(:,i));
    alpha_surfs(i) = i_surfs(i) - asin(dotprod/norm(vB_BfromA_mps));
end

% Coefficient slopes
C_La_surfs = 2*pi.*AR./(2+AR); % 3D lift slope approx. for each surface
C_Da_surfs = aero(:,5)';
C_Ma_surfs = aero(:,8)';

% Coefficients at 0 AoA for each surface
C_L0_surfs = aero(:,1);
C_D0_surfs = aero(:,4);
C_M0_surfs = aero(:,7);

% AoA of 0 drag
a0_surfs = aero(:,6)';

% Calculate all coefficients for each aerodynamic surface
C_Ls_surfs = C_L0_surfs' + C_La_surfs.*alpha_surfs;
C_Ds_surfs = C_D0_surfs' + C_Da_surfs.*(alpha_surfs - a0_surfs).^2 +...
    C_Ls_surfs.^2./(pi.*aero(:,2)'.*AR);
C_Ms_surfs = C_M0_surfs' + C_Ma_surfs.*alpha_surfs;

% Calculate lift and drag of each surface. These are scalars.
lifts_surfs_N       = 0.5*rho*v_inf^2.*C_Ls_surfs.*S_m2;
drags_surfs_N       = 0.5*rho*v_inf^2.*C_Ds_surfs.*S_m2;

% Moments due to the surfaces themselves (different from moments_s_Nm)
moments_surfs_Nm    = 0.5*rho*v_inf^2.*C_Ms_surfs.*S_m2.*c_m;

% Initialize total force and moment
total_force_b_N       = [0;0;0];
total_moment_b_Nm     = [0;0;0];
utility_matrix  = [0,0,0; 0,0,-1; 0,1,0]; % for moment calculation
for i = 1:1:length(aero(:,1))
    % Calculate net force vector on the surface in wind frame
    force_surf_w = lifts_surfs_N(i)*surfs(:,i) - [drags_surfs_N(i);0;0];

    % Calculate net force vector on the surface in body frame
    force_surf_b = C_BfromN*force_surf_w;

    % Calculate net moment vector on the surface in the wind frame.
    % Moments on the surfaces (different from moments_surfs_Nm)
    moment_s_Nm = moments_surfs_Nm(i)*utility_matrix*surfs(:,i) + ...
        cross((surf_pos_m(i,:) - cg_m), force_surf_b)';

    % Calculate net moment vector on the surface in the body frame
```

```
        % NOTE: Theyre the same because of surface vector definition
        moment_b_Nm = moment_s_Nm;

        % Total the body forces and moments
        total_force_b_N       = total_force_b_N + force_surf_b;
        total_moment_b_Nm     = total_moment_b_Nm + moment_b_Nm;
    end

    C_D_fuse        = 0.5;
    S_fuse          = 0.001;
    total_force_b_N = total_force_b_N - [0.5*rho*v_inf^2*C_D_fuse*S_fuse;0;0];

end
```

Published with MATLAB® R2021b