Boeing 737 MAX 8 Preliminary Sizing

Clear previous data

```
format long
clc
clear
```

Unit exchange

```
ft_to_m = 0.3048;       % ft to m
m_s_to_mph = 2.236936;  % m/s to mph
m_s_to_kt = 1.943844;   % m/s to kt
m_s_to_ft_s = 3.280840; % m/s to ft/s
ft_s_to_kt = 0.592484;  % ft/s to kt
```

Mission profile parameters

```
% Payload weight
% 180 Passengers at 175 lbs each and 30 lbs of baggage each
W_PL = (175+30)*180; % unit: lbs

% Crew weight
% 2 Pilots and 6 flight attendents at 175 lbs each and 30 lbs of baggage each
W_crew = (175+30)*8; % unit: lbs
D = W_crew + W_PL;

% CruiseAltitude (unit: ft)
CruiseAltitudeMin = 35000;
CruiseAltitudeMax = 41000;
CruiseAltitudeInterval = 1000;
CruiseAltitudeMatrix = [CruiseAltitudeMin:CruiseAltitudeInterval:CruiseAltitudeMax];

% Range (unit: nm)
RangeMin = 2000;
RangeMax = 3500;
RangeInterval = 100;
RangeMatrix = [RangeMin:RangeInterval:RangeMax];

% LoverD_Cruise
LoverD_CruiseMin = 13;
LoverD_CruiseMax = 14;
LoverD_CruiseInterval = 0.1;
LoverD_CruiseMatrix = [LoverD_CruiseMin:LoverD_CruiseInterval:LoverD_CruiseMax];

% LoverD_Loiter
LoverD_LoiterMin = 17;
LoverD_LoiterMax = 18;
LoverD_LoiterInterval = 0.1;
LoverD_LoiterMatrix = [LoverD_LoiterMin:LoverD_LoiterInterval:LoverD_LoiterMax];

% c_j_cruise
c_j_cruiseMin = 0.5;
c_j_cruiseMax = 0.55;
c_j_cruiseInterval = 0.01;
c_j_cruiseMatrix = [c_j_cruiseMin:c_j_cruiseInterval:c_j_cruiseMax];

% c_j_loiter
c_j_loiterMin = 0.55;
c_j_loiterMax = 0.6;
c_j_loiterInterval =0.01;
c_j_loiterMatrix = [c_j_loiterMin:c_j_loiterInterval:c_j_loiterMax];
```

```
% Loiter
Endurance = 0.5; % unit: hr

%
AverageClimbRate = 2500; % unit: fpm
CruiseSpeed_Mach = 0.79; % unit: Mach
AlternateCruiseSpeed = 250; % unit: kts
AlternateRange = 100;% unit: nm

% Regression Line Constants A and B of Equation
% Airplane type: Transport jet
A = 0.0833;
B = 1.0383;

% W_TO & W_OE from wikipedia (unit: lbs)
W_OE_wiki = 99360;
W_TO_wiki = 182200;
W_E_wiki = W_OE_wiki - W_TO_wiki*0.005 - W_crew;
W_E_wiki2W_TO_guess = 10^(A+B*log10(W_E_wiki));
W_E_real_wiki = 10^((log10(W_TO_wiki)-A)/B);
error_wiki = abs(W_E_real_wiki-W_E_wiki)/W_E_real_wiki;
```

Fuel Fraction parameters

```
% Engine start and warm up, from Table 2.1
W1_W_TO_guess_ratio = 0.990;
% Taxi, from Table 2.1
W2_W1_ratio = 0.990;
% Take-off, from Table 2.1
W3_W2_ratio = 0.995;
% Climb, from Table 2.1
W4_W3_ratio = 0.980;
% Decent,from Table 2.1
W7_W6_ratio = 0.990;
% Fly to alternate and descend, from Brequet's range equation, L/D = 10, c_j = 0.9
W8_W7_ratio = 1/(exp(AlternateRange/(AlternateCruiseSpeed/0.9*10)));
% Landing, Taxi and Shutdown, From Table 2.1
W9_W8_ratio = 0.992;
```

InputParametersMatrix setup section

```
% InputParametersMatrix sizing
InputParametersMatrix_row = width(CruiseAltitudeMatrix)*width(RangeMatrix)...
                    *width(LoverD_CruiseMatrix)*width(LoverD_LoiterMatrix)...
                    *width(c_j_cruiseMatrix)*width(c_j_loiterMatrix);
InputParametersMatrix_column = 15;
InputParametersMatrixTemp = zeros(InputParametersMatrix_row,InputParametersMatrix_column);
InputParametersMatrix = zeros(InputParametersMatrix_row,InputParametersMatrix_column);

% Create InputParametersMatrix
n=1;
for CruiseAltitude = CruiseAltitudeMin:CruiseAltitudeInterval:CruiseAltitudeMax
    for Range = RangeMin:RangeInterval:RangeMax
        for LoverD_Cruise = LoverD_CruiseMin:LoverD_CruiseInterval:LoverD_CruiseMax
            for LoverD_Loiter = LoverD_LoiterMin:LoverD_LoiterInterval:LoverD_LoiterMax
                for c_j_cruise = c_j_cruiseMin:c_j_cruiseInterval:c_j_cruiseMax
                    for c_j_loiter = c_j_loiterMin:c_j_loiterInterval:c_j_loiterMax
                        InputParametersMatrixTemp(n,1) = CruiseAltitude;
                        InputParametersMatrixTemp(n,2) = Range;
                        InputParametersMatrixTemp(n,3) = LoverD_Cruise;
                        InputParametersMatrixTemp(n,4) = LoverD_Loiter;
                        InputParametersMatrixTemp(n,5) = c_j_cruise;
                        InputParametersMatrixTemp(n,6) = c_j_loiter;
                        n=n+1;
```

```
                    end
                end
            end
        end
    end
end
```

Calculate CruiseSpeed/AverageClimbSpeed/ClimbTime/CruiseRange/W5_W4_ratio/W6_W5_ratio/M_ff/C

```
parfor row = 1:InputParametersMatrix_row

    % Temporary matrix for parallel computing
    temp = zeros(1,InputParametersMatrix_column);

    % Read data form InputParametersMatrixtemp
    CruiseAltitude = InputParametersMatrixTemp(row,1);
    Range = InputParametersMatrixTemp(row,2);
    LoverD_Cruise = InputParametersMatrixTemp(row,3);
    LoverD_Loiter = InputParametersMatrixTemp(row,4);
    c_j_cruise  = InputParametersMatrixTemp(row,5);
    c_j_loiter = InputParametersMatrixTemp(row,6);

    % Calculate parameters
    [a]=Standard_Atmosphere(CruiseAltitude);% unit:Imperial system
    CruiseSpeed = CruiseSpeed_Mach*a*ft_s_to_kt;% unit: kts
    AverageClimbSpeed = CruiseSpeed*0.6;% unit: kts
    ClimbTime = CruiseAltitude/AverageClimbRate;% unit: minute
    ClimbRange = AverageClimbSpeed*(ClimbTime/60); % unit: nm
    CruiseRange = Range - ClimbRange; % unit: nm
    % Cruise, from Breguet's range equation
    W5_W4_ratio = 1/(exp(CruiseRange/(CruiseSpeed/c_j_cruise*LoverD_Cruise)));
    % Loiter, from Breguet's endurance equation
    W6_W5_ratio = 1/(exp(0.5/(1/c_j_loiter*LoverD_Loiter)));
    M_ff = W1_W_TO_guess_ratio*W2_W1_ratio*W3_W2_ratio*W4_W3_ratio*...
        W5_W4_ratio*W6_W5_ratio*W7_W6_ratio*W8_W7_ratio*W9_W8_ratio
    C = 1-(1-M_ff)-0.005;

    % Output data
    temp(1) = CruiseAltitude;
    temp(2) = Range;
    temp(3) = LoverD_Cruise;
    temp(4) = LoverD_Loiter;
    temp(5) = c_j_cruise;
    temp(6) = c_j_loiter;
    temp(7) = CruiseSpeed;
    temp(8) = AverageClimbSpeed;
    temp(9) = ClimbTime;
    temp(10) = ClimbRange;
    temp(11) = CruiseRange;
    temp(12) = W5_W4_ratio;
    temp(13) = W6_W5_ratio;
    temp(14) = M_ff;
    temp(15) = C;

    % Output calculate result into InputParametersMatrix
    InputParametersMatrix(row, :) = temp;
end
```

Find W_TO_guess_min

```
% Read data
C_max = max(InputParametersMatrix(:,14));

% Find the lower bound of W_To_guess
syms x
```

```
W_TO_guess_min = vpasolve( A + B*log10(C_max*x - D) - log10(x) == 0 );
W_TO_guess_LowerBound = floor(W_TO_guess_min);
```

Numerical approximation of W_TO_guess

```
% ResultMatrixApporx sizing
W_TO_Apporx_row = InputParametersMatrix_row;
W_TO_Approx_column = 13;
W_TO_Approx = zeros(W_TO_Apporx_row,W_TO_Approx_column);

% Calculate W_TO_guess
parfor row = 1:W_TO_Apporx_row

    % Temporary matrix for parallel computing
    temp = zeros(1,W_TO_Approx_column);

    % Read data
    CruiseAltitude = InputParametersMatrix(row,1);
    Range = InputParametersMatrix(row,2);
    LoverD_Cruise = InputParametersMatrix(row,3);
    LoverD_Loiter = InputParametersMatrix(row,4);
    c_j_cruise  = InputParametersMatrix(row,5);
    c_j_loiter = InputParametersMatrix(row,6);
    CruiseSpeed = InputParametersMatrix(row,7);
    M_ff = InputParametersMatrix(row,13);
    C = InputParametersMatrix(row,14);

    % Iteration section
    for W_TO_guess = 165991:182200 % W_TO_guess_LowerBound:W_TO_wiki
        W_E_real = 10^((log10(W_TO_guess)-A)/B);
        W_E_tent = C*W_TO_guess - D;
        error = abs(W_E_tent - W_E_real)/ W_E_real;
        if error  < 0.005
            W_E_error =  abs(W_E_real - W_E_wiki)/W_E_real;

            % Output data
            temp(1) = CruiseAltitude;
            temp(2) = Range;
            temp(3) = LoverD_Cruise;
            temp(4) = LoverD_Loiter;
            temp(5) = c_j_cruise;
            temp(6) = c_j_loiter;
            temp(7) = CruiseSpeed;
            temp(8) = M_ff;
            temp(9) = C;
            temp(10) = W_TO_guess;
            temp(11) = W_E_tent;
            temp(12) = W_E_real;
            temp(13) = W_E_error;

            % Output calculate result into Result matrix
            W_TO_Approx(row, :) = temp;
            break
        end
    end
end
```

Check the amount of numerical approximation solutions

```
% ResultMatrixApporxSolutions sizing
W_TO_ApproxSolutions = zeros(W_TO_Approx_column);
n=1;
for row = 1:W_TO_Apporx_row
    if W_TO_Approx(row,10) > 0 && W_TO_Approx(row,10) < W_TO_wiki
        if W_TO_Approx(row,13) < 0.005 | W_TO_Approx(row,12) < W_E_wiki
```

```
            W_TO_ApproxSolutions(n,:) = W_TO_Approx(row,:);
            n=n+1;
        end
    end
end
```

W_TO_solutions sizing

```
W_TO_solutions_row = height(W_TO_ApproxSolutions);
W_TO_sloutions_column = width(W_TO_ApproxSolutions) ;
W_TO_solutions = zeros(W_TO_solutions_row,W_TO_sloutions_column);

x = sym('x',[1 W_TO_solutions_row]);
parfor row = 1: W_TO_solutions_row

    % Temporary matrix for parallel computing
    temp = zeros(1,W_TO_sloutions_column);

    % Read data
    CruiseAltitude = W_TO_ApproxSolutions(row,1);
    Range = W_TO_ApproxSolutions(row,2);
    LoverD_Cruise = W_TO_ApproxSolutions(row,3);
    LoverD_Loiter = W_TO_ApproxSolutions(row,4);
    c_j_cruise  = W_TO_ApproxSolutions(row,5);
    c_j_loiter = W_TO_ApproxSolutions(row,6);
    CruiseSpeed = W_TO_ApproxSolutions(row,7);
    M_ff = W_TO_ApproxSolutions(row,8);
    C = W_TO_ApproxSolutions(row,9);

    % vpasolve
    W_TO_guess = vpasolve( A + B*log10(C*x(row) - D) - log10(x(row)) == 0 );

    % Calculate W_E_real/W_E_tent/W_E_error
    W_E_real = 10.^((log10(W_TO_guess)-A)/B);
    W_E_tent = C*W_TO_guess-D;
    W_E_error = abs(W_E_real - W_E_wiki)/W_E_real;

    % Output data
    temp(1) = CruiseAltitude;
    temp(2) = Range;
    temp(3) = LoverD_Cruise;
    temp(4) = LoverD_Loiter;
    temp(5) = c_j_cruise;
    temp(6) = c_j_loiter;
    temp(7) = CruiseSpeed;
    temp(8) = M_ff;
    temp(9) = C;
    temp(10) = W_TO_guess;
    temp(11) = W_E_tent;
    temp(12) = W_E_real;
    temp(13) = W_E_error;

    % Output calculate result into Result matrix
    W_TO_solutions(row, :) = temp;
end
```

Sensitivity section

```
% W_TO_Senitivity sizing
W_TO_Senitivity = zeros();
for row = 1:W_TO_solutions_row
    if  W_TO_solutions(row,13) < error_wiki/250
        n = n+1;

        % Read data
```

```
        CruiseAltitude = W_TO_solutions(row,1);
        Range = W_TO_solutions(row,2);
        LoverD_Cruise = W_TO_solutions(row,3);
        LoverD_Loiter = W_TO_solutions(row,4);
        c_j_cruise  = W_TO_solutions(row,5);
        c_j_loiter = W_TO_solutions(row,6);
        CruiseSpeed = W_TO_solutions(row,7);
        M_ff = W_TO_solutions(row,8);
        C = W_TO_solutions(row,9);
        W_TO = W_TO_solutions(row,10);
        W_E_tent = W_TO_solutions(row,11);
        W_E_real = W_TO_solutions(row,12);
        W_E_error = W_TO_solutions(row,13);

        % Sensitivity calculate
        F=-B*(W_TO^2)*((C*W_TO*(1-B)-D)^-1)*(1+0)*M_ff;
        % W_TO over W_PL
        W_TO_over_W_PL = B*W_TO*(D-C*(1-B)*W_TO)^-1;
        % W_TO over W_E
        W_TO_over_W_E = B*W_TO*(10^((log10(W_TO)-A)/B))^-1;
        % W_TO over Range
        W_TO_over_Range = F*c_j_cruise*(CruiseSpeed*LoverD_Cruise)^-1;
        % W_TO over Endurance
        W_TO_over_Endurance = F*c_j_loiter*LoverD_Loiter^-1;
        % W_TO over Cruise speed
        W_TO_over_CriuseSpeed = -F*Range*c_j_cruise*(CruiseSpeed^2*LoverD_Cruise)^-1;
        % W_TO over c_j_Range
        W_TO_over_c_j_Range = F*Range*(CruiseSpeed*LoverD_Cruise)^-1;
        % W_TO over L/D_Range
        W_TO_over_LoverD_Range = -F*Range*c_j_cruise*(CruiseSpeed*LoverD_Cruise^2)^-1;
        % W_TO over c_j_Loiter
        W_TO_over_c_j_Loiter = F*Endurance*LoverD_Loiter^-1;
        % W_TO over L/D_Loiter
        W_TO_over_LoverD_Loiter = -F*Endurance*c_j_loiter*LoverD_Loiter^-2;

        % Output result
        W_TO_Senitivity(n,1) = CruiseAltitude;
        W_TO_Senitivity(n,2) = Range;
        W_TO_Senitivity(n,3) = LoverD_Cruise;
        W_TO_Senitivity(n,4) = LoverD_Loiter;
        W_TO_Senitivity(n,5) = c_j_cruise;
        W_TO_Senitivity(n,6) = c_j_loiter;
        W_TO_Senitivity(n,7) = CruiseSpeed;
        W_TO_Senitivity(n,8) = M_ff;
        W_TO_Senitivity(n,9) = W_TO;
        W_TO_Senitivity(n,10) = W_E_tent;
        W_TO_Senitivity(n,11) = W_E_real;
        W_TO_Senitivity(n,12) = W_E_error;
        W_TO_Senitivity(n,13) = W_TO_over_W_PL;
        W_TO_Senitivity(n,14) = W_TO_over_W_E ;
        W_TO_Senitivity(n,15) = W_TO_over_Range;
        W_TO_Senitivity(n,16) = W_TO_over_Endurance;
        W_TO_Senitivity(n,17) = W_TO_over_CriuseSpeed;
        W_TO_Senitivity(n,18) = W_TO_over_c_j_Range;
        W_TO_Senitivity(n,19) = W_TO_over_LoverD_Range;
        W_TO_Senitivity(n,20) = W_TO_over_c_j_Loiter;
        W_TO_Senitivity(n,21) = W_TO_over_LoverD_Loiter;
    end
end
```

Select Parameters from W_TO_Senitivity matrix

```
% Take-off weight
W_TO = 182043.622463998; % unit: lbs

% Parameters at CruiseAltitude
CruiseAltitude = 40000; % unit: ft
```

```
[a,rho]=Standard_Atmosphere(CruiseAltitude);
a_CruiseAltitude = a;
rho_CruiseAltitude = rho;
CruiseSpeed_Mach = 0.79;
CruiseSpeed = CruiseSpeed_Mach*a_CruiseAltitude;
q_overline = 0.5*rho_CruiseAltitude*CruiseSpeed^2;

% Parameters at FieldAltitude
FieldLength_TO = 11000; % unit: ft
FieldLength_L = 8000; % unit: ft
FieldAltitude = 2000; % unit: ft
[a,rho,P]=Standard_Atmosphere(FieldAltitude);
rho_FieldAltitude = rho;
P_FieldAltitude = P;

% Parameters at sea level
[a,rho,P,Rankine]=Standard_Atmosphere(0);
rho_SeaLevel = rho;
P_SeaLevel = P;
T_SeaLevel = Rankine;

% Ratio
P_FieldAltitude_over_P_SeaLevel = P_FieldAltitude/P_SeaLevel;
T_95F_over_T_SeaLevel = (95+459.7)/T_SeaLevel;
Density_ratio_TO = P_FieldAltitude_over_P_SeaLevel/T_95F_over_T_SeaLevel;
WoverS = 0:10:200;
c = 0.0199;
d = 0.7531;
S_wet = 10^(c+d*log10(W_TO));

% From table 3.4 correlation coefficients for parasite area versus wetted area
cf_2 = 0.003; a_2 = -2.5229; b_2 = 1;
cf_3 = 0.004; a_3 = -2.3979; b_3 = 1;
f_2 = 10^(a_2+b_2*log10(S_wet));
f_3 = 10^(a_3+b_3*log10(S_wet));

% Coefficients
delta_CD0_TOflaps = 0.015;  % From p.127, Table 3.6
delta_CDO_Lflaps = 0.065;   % From p.127, Table 3.6
delta_CD0_LG = 0.02;        % From p.127, Table 3.6
e_TOflaps = 0.8;            % From p.127, Table 3.6
e_Lflaps = 0.75;            % From p.127, Table 3.6
e_clean = 0.8;
W_L = W_TO*0.84;            % 0.84 is from p.107, Table 3.3
S = W_TO/100;               % W/S = 100
CD_0_clean = f_2/S;         % Take cf = 0.003
delta_C_D0 = 0.0001*2.5;    % p.166 figure 3.32
C_D0_modification = CD_0_clean + delta_C_D0;

W_TO_wiki = 182200; % unit: lb
b = 117.833; % unit: ft
S_wiki = 1370; % unit: ft^2
AR = b^2/S_wiki; % unit: ft
StaticThrust_TO = 26786; % unit: lbs
WoverS_TO_wiki = W_TO_wiki/S_wiki; % unit: lb/ft^2
ToverW_TO_wiki = StaticThrust_TO*2/W_TO_wiki;
```

plot section

```
hold on

% FAR25 TAKEOFF DISTANCE SIZING
for CL_max_TO = 1.6:0.2:2.2
    ToverW = 37.5/(Density_ratio_TO*CL_max_TO*FieldLength_TO).*WoverS;
    plot(WoverS,ToverW,'color',[0 0.4470 0.7410]); % blue
end
```

```matlab
% FAR25 LANDING DISTANCE SIZING
for CL_max_L = 1.8:0.2:2.4
    V_stall_sqrt = FieldLength_L/(0.3*1.3^2)/ft_s_to_kt^2;
    WoverS_landing = V_stall_sqrt/2*rho_FieldAltitude*CL_max_L;
    WoverS_takeoff = WoverS_landing/0.84;
    ToverW_landing = [0 1.6];
    WoverS_takeoff = [WoverS_takeoff WoverS_takeoff];
    plot(WoverS_takeoff,ToverW_landing,'color',[0.4660 0.6740 0.1880]); % green
end

% CRUISE SPEED SIZING
ToverW_cruise_reqd = C_D0_modification*q_overline./WoverS+WoverS./(q_overline*pi*AR*e_clean);
ToverW_TO = ToverW_cruise_reqd./0.191;
plot(WoverS,ToverW_TO,'color',[0.9290 0.6940 0.1250]); % orange

% FAR25 CLIMB RATE SIZING
% FAR25.111 OEI (P.145)
CL_TO_max = 2;                    % From Table 3.1
CL = CL_TO_max/1.2^2;             % at 1.2 V_stall_TO
LoverD = CL/(CD_0_clean+delta_CD0_TOflaps+delta_CD0_LG+CL^2/(pi*AR*e_TOflaps)); % CL/CD_TO_GearDown
ToverW_TO = 2*(1/LoverD+0.012);    % CGR>0.012
ToverW_TO1 = ToverW_TO/0.8; % 50°F效應(除以0.8)

% FAR25.121 OEI
CL = CL_TO_max/1.1^2; % V_LOF = 1.1 V_stall_TO
LoverD = CL/(CD_0_clean+delta_CD0_TOflaps+delta_CD0_LG+CL^2/(pi*AR*e_TOflaps));
ToverW_TO = 2*(1/LoverD); % CGR>0
ToverW_TO2 = ToverW_TO/0.8; % 50°F效應(除以0.8)

% FAR25.121 OEI
CL = CL_TO_max/1.2^2; % at 1.2 V_stall_TO
LoverD = CL/(CD_0_clean+delta_CD0_TOflaps+CL^2/(pi*AR*e_TOflaps));
ToverW_TO = 2*(1/LoverD+0.024); % CGR>0.024
ToverW_TO3 = ToverW_TO/0.8;

% FAR25.121 OEI
CL_max = 1.4; % From Table 3.1
CL = CL_max/1.25^2; % at 1.25 V_stall
LoverD = CL/(CD_0_clean + CL^2/(pi*AR*e_clean));
ToverW_TO = 2*(1/LoverD+0.012); % CGR>0.012
ToverW_TO4 = ToverW_TO/0.94/0.8; % 最大推力校正(除以0.94), 50°F效應(除以0.8)

% FAR25.119 AEO
CL_max_L = 2.8; % From Table 3.1
CL = CL_max_L/1.3^2; % at 1.3 V_stall_L
LoverD = CL/(CD_0_clean+delta_CDO_Lflaps+delta_CD0_LG+CL^2/(pi*AR*e_Lflaps));
ToverW_L = 1/LoverD+0.032; % CGR>0.032
ToverW_TO5 = ToverW_L*(W_L/W_TO)/0.8;

% FAR25.121 OEI
CL_max_A = 2.4; % From Table 3.1
CL = CL_max_A/1.5^2; % at 1.5 V_stall_A
LoverD = CL/((CD_0_clean+delta_CD0_TOflaps+CD_0_clean+delta_CDO_Lflaps)/2+delta_CD0_LG+CL^2/(pi*AR*e_Lflaps));
ToverW_L = 2*(1/LoverD+0.021); % CGR>0.021
ToverW_TO6 = ToverW_L*(W_L/W_TO)/0.8;
WoverS_TO = [0 200];
ToverW_TO1 = [ToverW_TO1 ToverW_TO1];
ToverW_TO2 = [ToverW_TO2 ToverW_TO2];
ToverW_TO3 = [ToverW_TO3 ToverW_TO3];
ToverW_TO4 = [ToverW_TO4 ToverW_TO4];
ToverW_TO5 = [ToverW_TO5 ToverW_TO5];
ToverW_TO6 = [ToverW_TO6 ToverW_TO6];
plot(WoverS_TO,ToverW_TO1,'color',[0.4940 0.1840 0.5560]) % purple
plot(WoverS_TO,ToverW_TO2,'color',[0.4940 0.1840 0.5560]) % purple
plot(WoverS_TO,ToverW_TO3,'color',[0.4940 0.1840 0.5560]) % purple
plot(WoverS_TO,ToverW_TO4,'color',[0.4940 0.1840 0.5560]) % purple
plot(WoverS_TO,ToverW_TO5,'color',[0.4940 0.1840 0.5560]) % purple
plot(WoverS_TO,ToverW_TO6,'color',[0.4940 0.1840 0.5560]) % purple
plot(WoverS_TO_wiki,ToverW_TO_wiki,'rx')
```

```
title('MATCHING RESULT FOR SIZING OF BOEING 737MAX8')
xlabel('(W/S)_{TO}');
ylabel('(T/W)_{TO}');
hold off
```

```
function [a,rho,P,Rankine]=Standard_Atmosphere(h)

% Unit exchange
ft_to_m = 0.3048;        % ft to m
m_s_to_mph = 2.236936;  % m/s to mph
m_s_to_kt = 1.943844;   % m/s to kt
m_s_to_ft_s = 3.280840; % m/s to ft/s
ft_s_to_kt = 0.592484;  % ft/s to kt
kg_to_slug = 0.068522;  % kg to slug

h = h*ft_to_m;% unit:m
[T,a,P,rho] = atmosisa(h);
Rankine = T*1.8;% unit:Rankine
rho = rho*kg_to_slug*ft_to_m^3;% unit:slug^3/ft^3
a = a*m_s_to_ft_s;% unit:ft/s
end
```

Published with MATLAB® R2022a