Simplified Mission Code Report

Gregory Golonka, Connor Hack, John Tuma, and Timothy Welch

September 2024

1 Introduction

A code was generated to simulate the trajectory of a missile launched from sea. This was a simplified form of a code that will later be generated to re-engineer the BGM-109A Tomahawk Cruise Missile to optimize the delivery of a payload from a launch point in the Persian Gulf to Jalal-Abad, Afghanistan, following very specific phases. The code discussed here, however, exists as a proof-of-concept and foundation for that later code, and as such has fewer and simpler mission requirements summarized as:

1. Simulate stand-off distance by cruising at an altitude of 100 ft at a speed of Mach 0.5 which will be held constant for the entirety of the simplified simulation.
2. Climb to an altitude of 2.0 km over a range of 300 km.
3. Cruise at an altitude of 2.0 km for 300 km.
4. Descend to an altitude of 1.5 km over a range of 500 km.
5. Climb to an altitude of 2.8 km over a range of 1000 km.

2 Theory

This simulation was conducted using parameters taken from the actual BGM-109A Tomahawk missile and given in the problem handout [1].

The general theory behind the code is to use the Breguet Range Equations, which for cruise is

$$R_2 - R_1 = 0.97\frac{VC_L}{SC_D}ln\left(\frac{W_1}{W_2}\right) \tag{1}$$

where R is range, V is airspeed, L and D are the lift and drag, S is the thrust-specific fuel consumption, and W is the weight. For climb and descent, these equations are

$$R_2 - R_1 = \frac{VL}{SF}ln\left(\frac{W_1}{W_2}\right) \tag{2}$$

$$h_2 - h_1 = \frac{V}{SF}(T - D)ln\left(\frac{W_1}{W_2}\right) \tag{3}$$

where h is the altitude, F is the uninstalled thrust, and T is the installed thrust.

Specifically, the procedure involves determining lift, drag, and their coefficients from a force analysis on the vehicle in climb, descent, or cruise. Then, the thrust to maintain the current Mach number is calculated from these values, and the thrust-specific fuel consumption is calculated from a theoretical thrust hook supplied in the problem handout, shown in Figure 1, and mathematically represented by the equation

$$S = 0.83 - 1.240 * 10^{-3} * T + 1.593 * 10^{-6} * T^2 + \frac{0.2}{20000} z + \frac{0.2}{0.75} M \tag{4}$$

where z is altitude and M is mach number [1]. The thrust-specific fuel consumption is then used to find the change in weight via the equation

$$\Delta W = S * T * \Delta t * g \tag{5}$$

where Δt is the time-step and g is the acceleration due to gravity. Then, Equations 1, 4, and 3 can be used, and the other relevant parameters can be advanced.

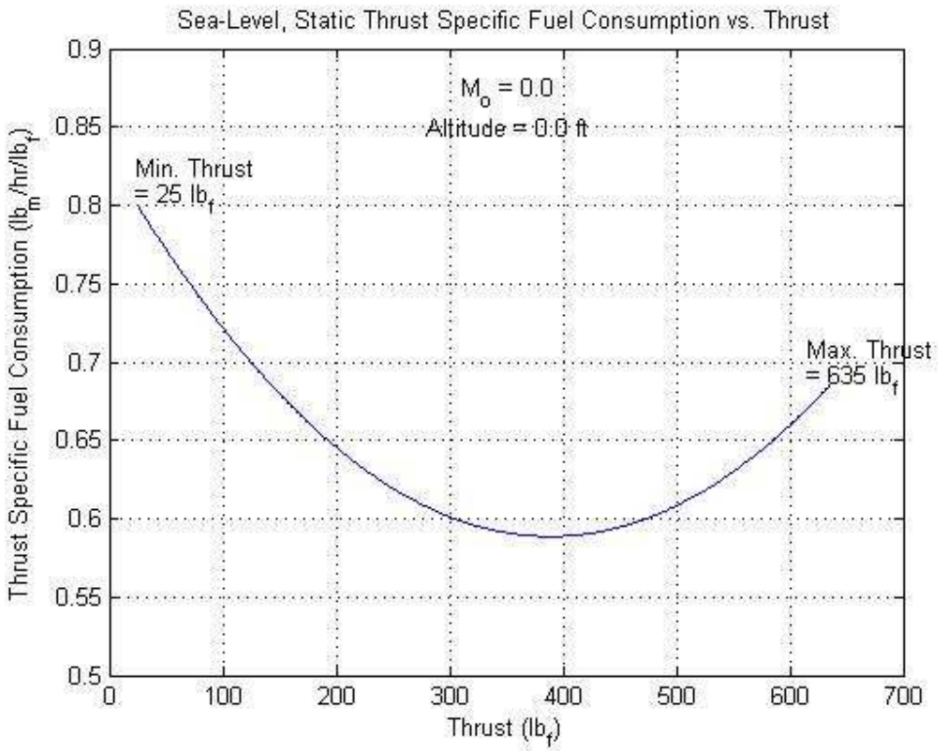

Figure 1: Theoretical thrust hook supplied for the problem.

3 Results

The mission code was used to analyze the flight plan of a BGM-109A Tomahawk Cruise Missile. Within the flight simulation, the overall fuel consumption, altitude, and range were considered across the different flight stages.

Table 1: Relevant data values at key mission checkpoints

Phase	Altitude, h (km)	Range, R (km)	Fuel Percent (%)	Weight, W (N)
Start	0.030	0	100	12904.1
Standoff	0.030	1691.5	52.76	10429.9
Climb 1	2.0	1992.2	44.40	9992.1
Cruise 1	2.0	2292.6	36.44	9575.3
Decent	1.5	2793.1	23.94	8920.8
Climb 2	2.8	3794.1	0.09	7671.3

Table 1 shows the flight data for the simplified mission at each stage. Each of the altitudes listed in the table represents important altitude markers dictated by mission parameters. The range, fuel percentage, and weight were each tracked at each point of the mission. The altitude and range of the missile were then plotted to show the flight path over the course of the mission.

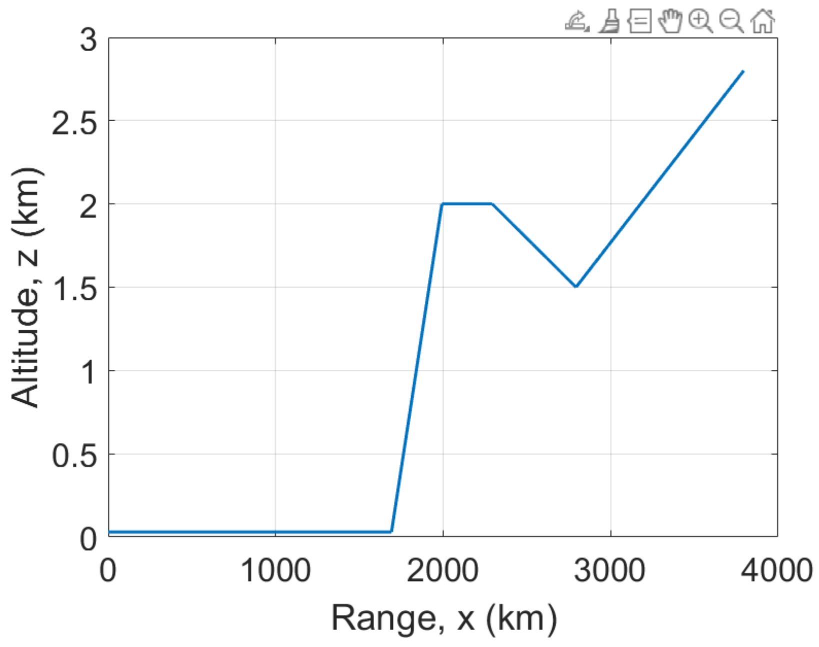

Figure 2: Flight path for the simplified mission.

Figure 2 shows the completed flight path for the mission. Each phase of the mission is apparent in 2, starting with the initial low altitude cruise, a climb to specified altitude, level flight, a descent, and curtailed by a climb terminating with the delivery of the payload.

Several other plots were generated to facilitate the analysis of the code. These are TSFC versus time (shown in Figure 3), altitude versus time (shown in Figure 4), and TSFC versus thrust (shown in Figure 5).

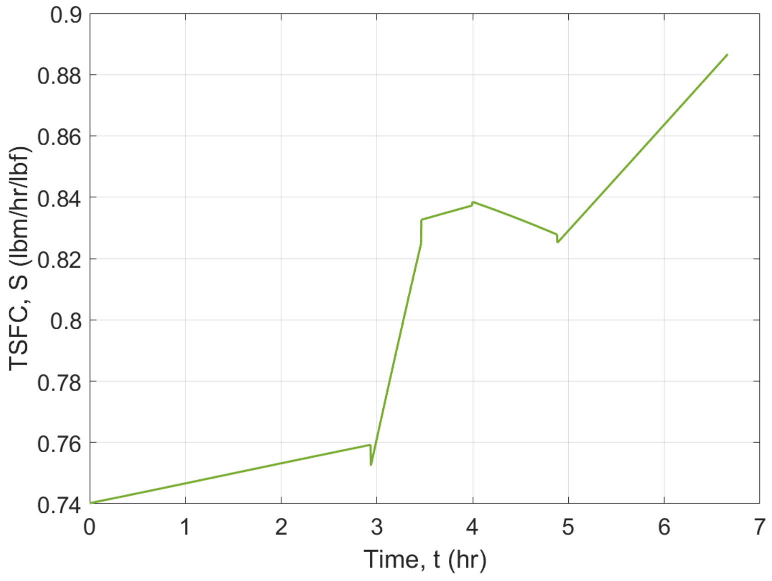

Figure 3: Thrust-specific fuel consumption versus time.

The value TSFC can clearly be seen to change over time and change with regard to the mission phase. This result is expected as the thrust required at different phases changes.

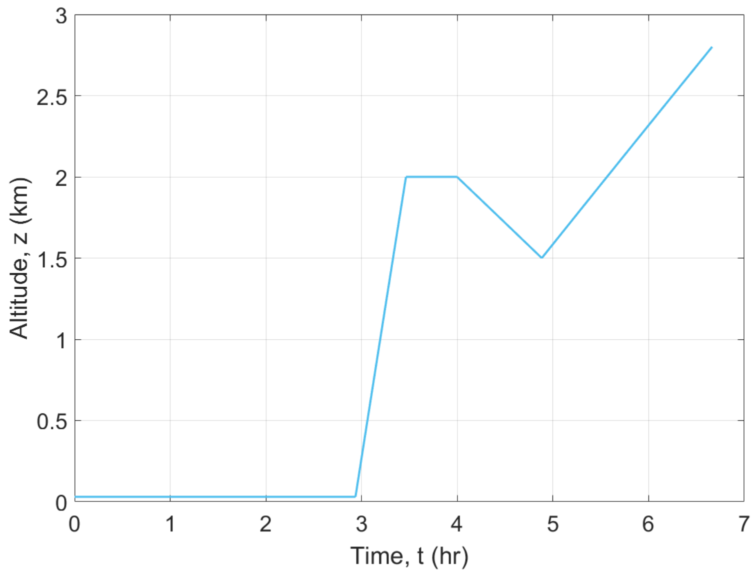

Figure 4: Altitude versus time.

Figure 4 appears remarkably similar to Figure 2. This result is expected because, though not a constant rate, range consistently increases with time, leading to a similar shape between Figures 2 and 4.

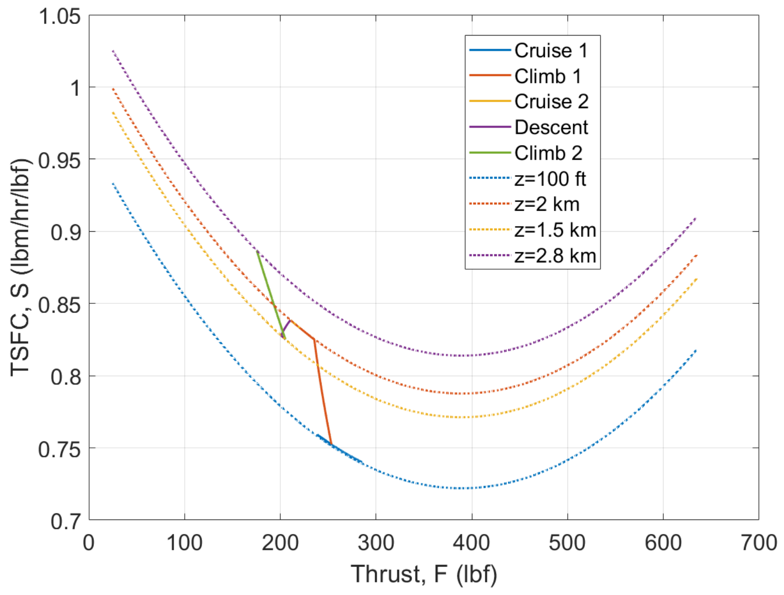

Figure 5: Thrust-specific fuel consumption versus thrust, given at multiple altitudes with theoretical curve.

Figure 5 is especially interesting because, for the cruise phases, the simulation results match perfectly with the predicted values from the thrust hook in Figure 1. Furthermore, the altitude-varying phases (climb and descent) show the value of TSFC linearly transitioning between theoretical altitude values. Such a perfect match with the theory provides confirmation, indicating the success of the simulation.

4 Discussion

Successfully completing the provided simulated mission parameters was necessary before the initial standoff range could be determined. This is because the fuel used to propel the missile forward was constantly decreasing to provide the thrust necessary for level flight and each change of altitude. Thus before the initial cruise range could be decided, a base fuel level was necessary to obtain. The ever-changing weight and thrust of the missile presented the greatest challenge when determining the angle of ascent for both the climb and descent, as simple geometry only provided a jumping-off point when considering the necessary angle to meet the mission checkpoints. Once the missile was proven to be able to meet each checkpoint of the mission with fuel remaining and at the desired 0.5 Mach number, the initial standoff range was simply incremented until the remaining fuel of the missile upon completion of the mission became zero.

It is important to note the effect of each phase of the mission on the overall fuel remaining on the missile. Looking to Table 1 for values, it can be seen that the fuel used for each part of the mission in descending order goes as follows: initial cruise from launch, final climb, descent, initial climb, and midway cruise. As expected, the range required for each portion of the mission was directly correlated to the fuel consumed – the phases that occurred over a longer range also used more fuel than those of shorter ranges. Likewise, for phases of the mission that occurred over the same range, the more intense phases required more fuel, i.e. the climb over 300km required more fuel than the cruise over that same range.

Appendix A - MATLAB Code

```
9/8/24 7:41 PM  C:\Users\tjwel...\SimplifiedMissionCode.m    1 of 8

% Simplified Mission Code

% by Timothy Welch

clear; clc; close all

%% Initialize Equations
Temp = @(z) 288.15 - (6.5/1000)*z; % Temperature [K]
rho = @(z) 1.225 - (0.113/1000)*z; % Density [kg/m^3]
TSFC = @(Thrust,z,M) 0.83-1.240*(10^-3)*Thrust+1.593*(10^-6)*Thrust^2 ...
    +(0.2/20000)*z + (0.2/0.75)*M; %TSFC [lbm/hr/lbf]

%% Initialize Parameters
% Missile Specifications
b = 2.67;        % [m]
TWA = 0.84;      % tail wetted area four fins [m^2]
AR = 6.0;
LwoB = 5.563;    % Length without boosters [m]
LwB = 6.248;     % Length with boosters [m]
d = 0.518;       % Diameter [m]

% Weights
We = 327.9;      % empty weight [kg]
Wp = 453.6;      % Payload weight [kg]
Wb = 272.2;      % booster weight [kg]
Wfm = 533.9;     % Max fuel weight [kg]
Wft = 1587.6;    % Max takeoff weight [kg]
Fc = 567.8;      % Fuel Capacity [l]

% Performance
Mmax = 0.75;     % Never exceed speed
C = 2987;        % Ceiling [m]
Vc9 = 246.5;     % Max Cruise @ 9800ft [m/s]
Vc5 = 250.8;     % Max Cruise @ 5000ft [m/s]
VcSL = 246.5;    % Max Cruise @ SL [m/s]
% RoC dictated by booster

% Range / Endurance
Rmax = 2500;     % Max Range [km]
Emax = 2.77;     % Max Endurance [hr]

% Air
gamma = 1.4;
R = 287;         % [J/kg/K]
g = 9.81;        % [m/s/s]

% Aerodynamics
Sref = 1.11;     % Reference Planform Area [m^2]
e = 0.7472;      % Oswald's Efficiency Factor
CLmax = 0.81;    % Max Coefficient of Lift
```

```
CD0 = 0.034;    % Parasitic Drag Coeff (Drag Coeff at alf=0)

% Simulation Parameters
dt = 10;         % Time Step [s]
bInd = 0;

%% Initialize Vectors
M = 0.5;        % Mach number
z = 100/3.281;  % Altitude [m]
T = Temp(z(1)); % Temperature [K]
W = g*(We+Wp+Wfm);
x = 0;
t = 0;
vInf = M*sqrt(gamma*R*T);
Thrust = 0;
S = 0;

%% Procedure
% Cruise 1
totRange = 1690; % [km]
[x,z,t,W,vInf,M,T,Thrust,S] = cruise(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,gamma,↙
R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000);

pfuel = 100*(W/g-We-Wp)/Wfm;
Ttemp = [z(1)/1000,x(1),pfuel(1),W(1)];

Ttemp = [Ttemp;z(end)/1000,x(end)/1000,pfuel(length(x)),W(length(x))];
bInd(1) = length(x);

% Climb 1
theta = 0.3; % [deg]
totRange = 300; % [km]
iterating = 1;
while iterating
    [xT,zT,tT,WT,vInfT,MT,TT,~,~] = climb(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,↙
gamma,R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000,theta);
    if zT(end) >= 2000
        break
    else
        theta=theta+0.00005;
    end
end
[x,z,t,W,vInf,M,T,Thrust,S] = climb(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,gamma,↙
R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000,theta);
disp(z(end))
disp(theta)

pfuel = 100*(W/g-We-Wp)/Wfm;
Ttemp = [Ttemp;z(end)/1000,x(end)/1000,pfuel(length(x)),W(length(x))];
```

```
bInd(end+1) = length(x);

% Cruise 2
totRange = 300; % [km]
[x,z,t,W,vInf,M,T,Thrust,S] = cruise(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,gamma,↙
R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000);

pfuel = 100*(W/g-We-Wp)/Wfm;
Ttemp = [Ttemp;z(end)/1000,x(end)/1000,pfuel(length(x)),W(length(x))];

bInd(end+1) = length(x);

% Descend
theta = -0.01; % [deg]
totRange = 500; % [km]
iterating = 1;
i=1;
while iterating
    [xT,zT,tT,WT,vInfT,MT,TT,~,~] = climb(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,↙
gamma,R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000,theta);
    if zT(end) <= 1500
        break
    else
        theta=theta-0.000005;
    end
end
[x,z,t,W,vInf,M,T,Thrust,S] = climb(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,gamma,↙
R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000,theta);
disp(z(end))
disp(theta)

pfuel = 100*(W/g-We-Wp)/Wfm;
Ttemp = [Ttemp;z(end)/1000,x(end)/1000,pfuel(length(x)),W(length(x))];

bInd(end+1) = length(x);

% Climb 2
theta = 0.01; % [deg]
totRange = 1000; % [km]
iterating = 1;
while iterating
    [xT,zT,tT,WT,vInfT,MT,TT,~,~] = climb(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,↙
gamma,R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000,theta);
    if zT(end) >= 2800
        break
    else
        theta=theta+0.00005;
    end
end
[x,z,t,W,vInf,M,T,Thrust,S] = climb(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,g,gamma,↙
R,rho,Temp,TSFC,CLmax,CD0,dt,totRange*1000,theta);
disp(z(end))
disp(theta)

bInd(end+1) = length(x);

%% Final Calcs and Concatenations
pfuel = 100*(W/g-We-Wp)/Wfm;
Ttemp = [Ttemp;z(end)/1000,x(end)/1000,pfuel(length(x)),W(length(x))];

% Theoretical TSFC Calc
FTheo = linspace(25,635,1000);      % [lbf]
zTheo = [0,6561.68,4921.26,9186.35];% [ft]
MTheo = 0.5;
STheo = zeros(length(z),length(FTheo));
for i=1:length(zTheo)
    for j=1:length(FTheo)
        STheo(i,j) = TSFC(FTheo(j),zTheo(i),MTheo);
    end
end

%% Generate Table
T = array2table(Ttemp,'VariableNames',{'h [km]','x [km]','% fuel','Weight↙
[N]'},'RowName',{'Start','Standoff','Climb 1','Cruise 1','Decent','Climb 2'});
format long
round(T,1)
table2latex(T,'simplifiedMissionCodeResults')

%% Plot
plotColors{1} = '#0072BD'; % blue
plotColors{2} = '#D95319'; % orange
plotColors{3} = '#EDB120'; % yellow
plotColors{4} = '#7E2F8E'; % purple
plotColors{5} = '#77AC30'; % green
plotColors{6} = '#4DBEEE'; % teal
plotColors{7} = '#A2142F'; % red

figure(1)
plot(x/1000,z/1000,'-','Linewidth',1.5,'Color',plotColors{1})
set(gca,'fontSize',16)
ylabel('Altitude, z (km)')
xlabel('Range, x (km)')
grid on

figure(2)
plot(t/3600,W,'-','Linewidth',1.5,'Color',plotColors{2})
set(gca,'fontSize',16)
xlabel('Time, t (hr)')
ylabel('Weight, W (N)')
grid on
```

```
figure(3)
plot(x/1000,W,'-','Linewidth',1.5,'Color',plotColors{3})
set(gca,'fontSize',16)
xlabel('Range, x (km)')
ylabel('Weight, W (N)')
grid on

figure(4)
plot(x/1000,pfuel,'-','Linewidth',1.5,'Color',plotColors{4})
set(gca,'fontSize',16)
xlabel('Range, x (km)')
ylabel('Percent of Fuel Left, (%)')
grid on

figure(5)
plot(t/3600,S/0.10197,'-','Linewidth',1.5,'Color',plotColors{5})
set(gca,'fontSize',16)
xlabel('Time, t (hr)')
ylabel('TSFC, S (lbm/hr/lbf)')
grid on

figure(6)
plot(t/3600,z/1000,'-','Linewidth',1.5,'Color',plotColors{6})
set(gca,'fontSize',16)
xlabel('Time, t (hr)')
ylabel('Altitude, z (km)')
grid on

figure(8)
plot(Thrust(1:bInd(1))*0.224809,S(1:bInd(1))/0.10197,'-',...
    Thrust(bInd(1):bInd(2))*0.224809,S(bInd(1):bInd(2))/0.10197,'-',...
    Thrust(bInd(2):bInd(3))*0.224809,S(bInd(2):bInd(3))/0.10197,'-',...
    Thrust(bInd(3):bInd(4))*0.224809,S(bInd(3):bInd(4))/0.10197,'-',...
    Thrust(bInd(4):bInd(5))*0.224809,S(bInd(4):bInd(5))/0.10197,'-',...
    'Linewidth',1.5)
hold on
for i=1:length(zTheo)
    plot(FTheo,STheo(i,:),':','Linewidth',1.5,'Color',plotColors{i})
end
hold off
legend('Cruise 1','Climb 1','Cruise 2','Descent','Climb 2','z=100 ft','z=2 km','z=1.5↙
km','z=2.8 km')
set(gca,'fontSize',16)
xlabel('Thrust, F (lbf)')
ylabel('TSFC, S (lbm/hr/lbf)')
grid on

%% Cruise Function
function [x,z,t,W,vInf,M,T,Thrust,S] = cruise(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,↙
e,g,gamma,R,rho,Temp,TSFC,CLmax,CD0,dt,xEnd)
    i = length(x);
    xInit = x(i);
    while x(i) <= xInit+xEnd
        % Aeronautical Calcs
        CL = W(i)/(0.5*rho(z(i))*vInf(i)^2*Sref);
        if CL>CLmax % check if CL exceeds max
            CL=CLmax;
            disp("MAX CL EXCEEDED")
        end
        CD = CD0+CL^2/(pi*AR*e);
        Thrust(i) = W(i)*(CD/CL);
        S(i) = TSFC(Thrust(i)*0.224809,z(i)*3.28084,M(i)); % [lbm/hr/lbf]
        S(i) = S(i)*0.10197; % [kg/N/hr]
        dW = 0.97*g*Thrust(i)*S(i); % [N/hr]

        % Time Step Integration
        W(i+1) = W(i)-dW*dt/3600;
        x(i+1) = x(i)+(vInf(i)*(CL/CD)/(g*S(i)/3600))*log(W(i)/W(i+1))*0.97;
        z(i+1) = z(i);
        t(i+1) = t(i)+dt;
        T(i+1) = Temp(z(i+1));
        M(i+1) = M(i);  %vInf(i+1)/sqrt(gamma*R*T(i+1));
        vInf(i+1) = M(i+1)*sqrt(gamma*R*T(i+1));

        % Check if out of Fuel
        if W(i+1) < g*(We+Wp)
            disp("Out of Fuel")
            break
        end

        % Iterate
        i = i+1;
    end
    S(end+1) = S(end);
    Thrust(end+1) = Thrust(end);
end

%% Climb Function
function [x,z,t,W,vInf,M,T,Thrust,S] = climb(x,z,t,W,vInf,M,T,Thrust,S,We,Wp,Sref,AR,e,↙
g,gamma,R,rho,Temp,TSFC,CLmax,CD0,dt,xEnd,theta)
    i = length(x);
    xInit = x(i);
    while x(i) <= xInit+xEnd
        L = W(i)*cosd(theta);
        % Aeronautical Calcs
        CL = L/(0.5*rho(z(i))*vInf(i)^2*Sref);
        if CL>CLmax % check if CL exceeds max
            CL=CLmax;
            disp("MAX CL EXCEEDED")
        end
        CD = CD0+CL^2/(pi*AR*e);
        D = CD*0.5*rho(z(i))*vInf(i)^2*Sref;
        Thrust(i) = W(i)*sind(theta)+D;
        S(i) = TSFC(Thrust(i)*0.224809,z(i)*3.28084,M(i)); % [lbm/hr/lbf]
        S(i) = S(i)*0.10197; % [kg/N/hr]
        dW = 0.97*g*Thrust(i)*S(i); % [N/hr]

        % Time Step Integration
        W(i+1) = W(i)-dW*dt/3600;
        x(i+1) = x(i)+(vInf(i)*L/(Thrust(i)/0.97*g*S(i)/3600))*log(W(i)/W(i+1));
        z(i+1) = z(i)+(vInf(i).*(Thrust(i)-D)./(Thrust(i)*g.*S(i)/3600))*log(W(i)/W↙
(i+1));

        t(i+1) = t(i)+dt;
        T(i+1) = Temp(z(i+1));
        M(i+1) = M(i);  %vInf(i+1)/sqrt(gamma*R*T(i+1));
        vInf(i+1) = M(i+1)*sqrt(gamma*R*T(i+1));

        % Check if out of Fuel
        if W(i+1) < g*(We+Wp)
            disp("Out of Fuel")
            break
        end

        % Iterate
        i = i+1;
    end
    S(end+1) = S(end);
    Thrust(end+1) = Thrust(end);
end

function table2latex(T, filename)

    % Error detection and default parameters
    if nargin < 2
        filename = 'table.tex';
        fprintf('Output path is not defined. The table will be written in %s.\n',↙
filename);
    elseif ~ischar(filename)
        error('The output file name must be a string.');
    else
        if ~strcmp(filename(end-3:end), '.tex')
            filename = [filename '.tex'];
        end
    end
    if nargin < 1, error('Not enough parameters.'); end
    if ~istable(T), error('Input must be a table.'); end

    % Parameters
    n_col = size(T,2);
    col_spec = [];
    for c = 1:n_col, col_spec = [col_spec 'l']; end
    col_names = strjoin(T.Properties.VariableNames, ' & ');
    row_names = T.Properties.RowNames;
    if ~isempty(row_names)
        col_spec = ['l' col_spec];
        col_names = ['& ' col_names];
    end

    % Writing header
    fileID = fopen(filename, 'w');
    fprintf(fileID, '\\begin{tabular}{%s}\n', col_spec);
    fprintf(fileID, '%s \\\\ \n', col_names);
    fprintf(fileID, '\\hline \n');

    % Writing the data
    try
        for row = 1:size(T,1)
            temp{1,n_col} = [];
            for col = 1:n_col
                value = T{row,col};
                if isstruct(value), error('Table must not contain structs.'); end
                while iscell(value), value = value{1,1}; end
                if isinf(value), value = '$\infty$'; end
                temp{1,col} = num2str(value);
            end
            if ~isempty(row_names)
                temp = [row_names{row}, temp];
            end
            fprintf(fileID, '%s \\\\ \n', strjoin(temp, ' & '));
            clear temp;
        end
    catch
        error('Unknown error. Make sure that table only contains chars, strings or↙
numeric values.');
    end

    % Closing the file
    fprintf(fileID, '\\hline \n');
    fprintf(fileID, '\\end{tabular}');
    fclose(fileID);
end
```

References

[1] AME 40431, *Packet 2: GAS TURBINES AND PROPULSION DESIGN PROJECT FALL 2024*, University of Notre Dame, Notre Dame, IN, 2024.