```
%% Numerical Simulation of a Sounding Rocket Flight
% Author: Karl Parks
% Class: AE 530
% Date: 2.13.19
% GitHub Repo: https://github.com/Kheirlb/rockets
clc; clear; clear all; close all;

fprintf('\n --- Numerical Simulation of a Sounding Rocket Flight ---\n\n');
fprintf('Author: Karl Parks\n');
fprintf('Class: AE 530\n');
fprintf('Date: 2.13.19\n\n');

% Used this for help:
% https://mintoc.de/index.php/Gravity_Turn_Maneuver
% https://carlospereyradotus.wordpress.com/independent-projects/drag-simulation/

%% Dependent Functions
% rocketSimODE_Real
% rocketSimODE_Ideal
% valueAt
% findrho
% findTemp
% valueOfMach
% findCd
% simpleInterp
% yzero

% All the above functions can be found on the github repository.
% https://github.com/Kheirlb/rockets

%% Initial Variables
global Mo g0 drag0 beta0 thrust0 burntime m_dot r0 atmosphereData CdvsMach frontArea;

%% Time Adjustments
tStep = 0.1;
tFinal = 3000;
tSpan = 1:tStep:tFinal;

%% Choose Initial Conditions
rocketType = 1;
%rocketName = '';
% 1 = HW2
% 2 = FAR/MARS
% 3 = GAH
% 4 = HW1
% 5 = BASE11
% otherwise = LR101 Rocket

switch rocketType
```

```
    case 1
        %% Initial Values - HW2
        fprintf('rocketType: HW2\n');
        rocketName = 'HW2';
        beta0 = 1; %deg launch angle
        thrust0 = 20000; %Newtons
        burntime = 60; %seconds
        frontArea = 0.196; %m^2
        Mo = 750; %kg total weight
        Ms = 240; %strucure mass
        Ml = 10; %payload mass
    case 2
        %% Initial Values - FAR/MARS
        fprintf('rocketType: FAR/MARS\n');
        rocketName = 'FAR/MARS';
        beta0 = 0.0; %deg launch angle
        thrust0 = 2224; %Newtons
        burntime = 20; %seconds
        frontArea = 0.031; %m^2
        Mo = 75; %kg total weight
        Ms = 45; %strucure mass
        Ml = 0; %payload mass
    case 3
        %% Initial Values - GAH
        fprintf('rocketType: GAH\n');
        rocketName = 'GAH';
        beta0 = 1; %deg launch angle
        thrust0 = 4500; %Newtons
        burntime = 60; %seconds
        frontArea = 0.071; %m^2
        Mo = 150; %kg total weight
        Ms = 90; %strucure mass
        Ml = 0; %payload mass
    case 4
        %% Initial Values - HW1
        fprintf('rocketType: HW1\n');
        rocketName = 'HW1';
        beta0 = 1; %deg launch angle
        thrust0 = 25162; %Newtons
        burntime = 75; %seconds
        frontArea = 0.196; %m^2
        Mo = 1000; %kg total weight
        Ms = 240; %strucure mass
        Ml = 85; %payload mass
    case 5
        %% Initial Values - BASE11
        fprintf('rocketType: BASE11\n');
        rocketName = 'BASE11';
        beta0 = 0; %deg launch angle
```

```
        thrust0 = 31400; %Newtons
        burntime = 31; %seconds %31.4
        frontArea = 0.2; %m^2
        Mo = 640; %kg total weight %640
        Ms = 418; %strucure mass
        Ml = 0; %payload mass
    otherwise
        %% Initial Values - LR101
        fprintf('rocketType: LR101?\n');
        rocketName = 'LR101?';
        beta0 = 1; %deg launch angle
        thrust0 = 4500; %Newtons
        burntime = 15; %seconds
        frontArea = 0.071; %m^2
        Mo = 113; %kg total weight
        Ms = 90; %strucure mass
        Ml = 0; %payload mass
end

%% Propellant/Burnout
Mb = Ml + Ms; %mass at burnout (structure and payload)
Mp = Mo - Mb; %mass of propellant

%% Earth Values
g0 = 9.81;
r0 = 6.3781*10^6; %raduis of earth

%% Excel Data
fprintf("Importing Data: This may take a moment...\n");
fileName = 'rocketSimExcel.xlsx';
%col1 = Mach 0.01 increments
%col2 = Cd Power-off
%col3 = Cd Power-on
CdvsMach = xlsread(fileName, 1, 'A2:C2502');
%col1 = Altitude [m]
%col2 = Temp [K]
%col3 = Density [kg/m^3]
atmosphereData = xlsread(fileName, 2, 'A3:C1204');

%% Some Initial Calcs
m_dot = Mp/burntime; %mass flow rate
isp = thrust0/(m_dot*g0); %isp
ueq = isp*g0; %ueq
R = Mo/Mb; %mass ratio
deltaU = ueq*log(R); %total deltaU

%% Iterate 1 Second and Grab Initial Values
accel_y = (thrust0*cosd(beta0)-Mo*g0)/Mo;
accel_x = (thrust0*sind(beta0))/Mo;
```

```
uy_1 = accel_y;
ux_1 = accel_x;

z1o=0;     % x-(initial x position)
z2o=ux_1;  % x-(initial x velocity)
z3o=0;     % y-(initial y position)
z4o=uy_1;  % y-(initial y velocity)
y=[z1o;z2o;z3o;z4o];

%% Run Numerical Simulation
options = odeset('Events',@yzero);
%fprintf("\n------- Real --------\n");
[t, x, te, ye, ie]=ode45(@rocketSimODE_Real,tSpan,y, options);
%fprintf("\n------- Ideal --------\n");
[t2, x2, te2, ye2, ie2]=ode45(@rocketSimODE_Ideal,tSpan,y, options);

x_pos = x(:,1);  % x-(x position)
x_vel = x(:,2);  % x-(x velocity)
y_pos = x(:,3);  % y-(y position)
y_vel = x(:,4);  % y-(y velocity)

%% Post Processing Using Solved Position and Velocity Values
% This section will find Acceleration, Dynamic Pressure, and Mach # vs Time
return_t = length(t);
Mach = zeros(return_t,1);
Q = zeros(return_t,1);
AccelFlight = zeros(return_t,1);
velVecFlight = zeros(return_t,1);
fprintf('\nNOTICE: Holding some values constant on plots after t: %4.1f\n' , te(2) -
te(2)/10);

for i = 1:return_t
    area_ref = frontArea; %m^2
    rho = findrho(y_pos(i)); %rho = f(atl)
    temp = findTemp(y_pos(i));
    velVecFlight(i) = sqrt(x_vel(i)^2 + y_vel(i)^2);
    %fprintf('i: %2.1f   rho: %2.1f  temp: %2.1f  velVecFlight(i): %2.1f\n',i,rho,
temp,velVecFlight(i));
    Mach(i) = valueOfMach(velVecFlight(i), temp);
    Cd = findCd(Mach(i));
    drag = 0.5*rho*Cd*(velVecFlight(i)^2)*area_ref;
    beta = asind(x_vel(i)/(sqrt(x_vel(i)^2 + y_vel(i)^2)));
    g = g0*((r0/(r0+y_pos(i)))^2);
    m = valueAt(t(i), 'mass'); %kg total weight
    thrust = valueAt(t(i), 'thrust');
    x_accel = (thrust*sind(beta))/m - (drag*sind(beta))/m;     %x-accel
    y_accel = (thrust*cosd(beta))/m - (drag*cosd(beta))/m - g; %y-accel
    % This last few values have issues so we omit them.
    if t(i) < te(2) - te(2)/10
```

```
        Q(i) = 0.5*rho*(velVecFlight(i)^2);
        AccelFlight(i) = sqrt(x_accel^2 + y_accel^2);
        tempIndex = i;
    else % Simply set a flat line near flight completion
        velVecFlight(i) = velVecFlight(tempIndex);
        y_vel(i) = y_vel(tempIndex);
        Q(i) = Q(tempIndex);
        AccelFlight(i) = AccelFlight(tempIndex);
        Mach(i) = Mach(tempIndex);
    end
end

%% Plots

labelIdeal = 'Ideal - No Drag';
labelComplex = 'Real - Drag';
plotR = 2;
plotC = 3;

maxAltReal = max(x(:,3));
maxHorzReal = max(x(:,1));
maxVertVelReal = max(x(:,4));
maxHorzVelReal = max(x(:,2));

burnoutIndex = find(t == burntime);
maxAltIndex = find(y_pos == maxAltReal);
tIb = t(burnoutIndex);

firstPlot = subplot(plotR,plotC,1);
plot(t,x(:,3),'r',t2,x2(:,3),'k')
title('Altitude vs Time')
xlabel('Time [sec]');
ylabel('Altitude [m]');
legend(labelComplex,labelIdeal)
firstPlot.YAxis.TickLabelFormat='%.f';
firstPlot.YAxis.Exponent = 0;
y1 = get(firstPlot,'ylim');
hold on
plot([tIb tIb],y1, '--m')
legend(labelComplex,labelIdeal,'Burnout')
grid on

firstPlot(2) = subplot(plotR,plotC,4);
plot(t,x(:,1),'r',t2,x2(:,1),'k')
title('Downrange Distance vs Time')
xlabel('Time [sec]');
ylabel('Horizontal Distance [m]');
firstPlot(2).YAxis.TickLabelFormat='%.f';
firstPlot(2).YAxis.Exponent = 0;
```

```
y1 = get(firstPlot(2),'ylim');
hold on
plot([tIb tIb],y1, '--m')
legend(labelComplex,labelIdeal,'Burnout')
grid on

firstPlot(3) = subplot(plotR,plotC,2);
plot(t,velVecFlight,'r',t,x_vel,'b',t,y_vel,'g')
title('Velocity vs Time')
xlabel('Time [sec]');
ylabel('Velocity [m/s]');
legend('Flight Path','Horizontal','Vertical')
firstPlot(3).YAxis.TickLabelFormat='%.f';
firstPlot(3).YAxis.Exponent = 0;
y1 = get(firstPlot(3),'ylim');
hold on
plot([tIb tIb],y1, '--m')
legend('Flight Path','Horizontal','Vertical','Burnout')
grid on

firstPlot(4) = subplot(plotR,plotC,5);
plot(t,AccelFlight,'r')
title('Acceleration vs Time')
xlabel('Time [sec]');
ylabel('Acceleration [m/s^2]');
%legend('Flight Path')
firstPlot(4).YAxis.TickLabelFormat='%.f';
firstPlot(4).YAxis.Exponent = 0;
y1 = get(firstPlot(4),'ylim');
hold on
plot([tIb tIb],y1, '--m')
legend('Flight Path','Burnout')
grid on

firstPlot(5) = subplot(plotR,plotC,3);
plot(t,Q,'r')
title('Dynamic Pressure vs Time')
xlabel('Time [sec]');
ylabel('Q [Pa]');
firstPlot(5).YAxis.TickLabelFormat='%.f';
firstPlot(5).YAxis.Exponent = 0;
y1 = get(firstPlot(5),'ylim');
hold on
plot([tIb tIb],y1, '--m')
legend('Q','Burnout')
grid on

firstPlot(6) = subplot(plotR,plotC,6);
plot(t,Mach,'r')
```

```
title('Mach Number vs Time')
xlabel('Time [sec]');
ylabel('Mach');
firstPlot(6).YAxis.TickLabelFormat='%.f';
firstPlot(6).YAxis.Exponent = 0;
y1 = get(firstPlot(6),'ylim');
hold on
plot([tIb tIb],y1, '--m')
legend('Mach','Burnout')
grid on

%sgtitle(strcat('Plots for ',rocketName)); %only works on R2018b
fprintf('\nPlots Displayed\n');

fprintf('\nf. ');
fprintf('\nBurnout Time  (Real): %4.1f [sec]\n',t(burnoutIndex));
fprintf('Burnout Alt   (Real): %4.1f [m] %4.1f [ft]\n',y_pos(burnoutIndex),convlength
(y_pos(burnoutIndex),'m','ft'));
fprintf('Burnout Vel   (Real): %4.1f [m/s]\n',velVecFlight(burnoutIndex));

fprintf('\ng. ');
fprintf('\nTime @ Apogee (Real): %4.1f [sec]\n',t(maxAltIndex));
fprintf('Apogee Alt    (Real): %4.1f [m] %4.1f [ft]\n',maxAltReal,convlength
(maxAltReal,'m','ft'));

fprintf('\nh. ');
fprintf('\nTime @ Impact (Real): %4.1f [sec]\n',te(2));
fprintf('Max Horiz     (Real): %4.1f [m] %4.1f [ft]\n',maxHorzReal,convlength
(maxHorzReal,'m','ft'));

fprintf('\nl. ');
fprintf('\nMATLAB''s ode45 numerical integrator chooses its own time step.\n');
fprintf('https://www.mathworks.com/help/matlab/ref/ode45.html#bu00_4l_sep_shared-
tspan\n');

maxAltIdeal = max(x2(:,3));
maxVertVelIdeal = max(x2(:,4));
maxHorzVelIdeal = max(x2(:,2));

fprintf('\n--- Other Interesting Information ---');
fprintf("\nTotal Weight: %2.0f pounds\n", convmass(Mo,'kg','lbm'));
fprintf("Thrust:       %2.0f pounds\n", convforce(thrust0,'N','lbf'));
fprintf("T/W:          %2.3f \n", thrust0/(Mo*g0));
fprintf('Isp:          %4.1f [sec]\n',isp);
fprintf('u_eqivalent:  %4.1f [m/s]\n',ueq);
fprintf('Mass Ratio:   %1.0f \n',R);
fprintf('Delta V:      %4.1f [m/s]\n',deltaU);

h_max = (ueq^2*(log(R))^2)/(2*g0) - ueq*burntime*((R/(R-1))*log(R) - 1);
```

```
fprintf('\nMax Alt  (Real): %4.1f [m] %4.1f [ft]\n',maxAltReal,convlength
(maxAltReal,'m','ft'));
fprintf('Max Alt    (Eq): %4.1f [m] %4.1f [ft]\n',h_max,convlength(h_max,'m','ft'));
fprintf('Max Alt (Ideal): %4.1f [m] %4.1f [ft]\n',maxAltIdeal,convlength
(maxAltIdeal,'m','ft'));

fprintf('\nMax Vert Vel  (Real): %4.1f [m/s]\n',maxVertVelReal);
fprintf('Max Horz Vel  (Real): %4.1f [m/s]\n',maxHorzVelReal);
fprintf('Max Velocity  (Real): %4.1f [m/s]\n',max(velVecFlight(1:maxAltIndex)));
%fprintf('Max Vert Vel  (Ideal): %4.1f [m/s]\n',maxVertVelIdeal);
%fprintf('Max Horz Vel  (Ideal): %4.1f [m/s]\n',maxHorzVelIdeal);
```