Table of Contents

Thomas Dunnington Rocket Equation Model

```
%Author: Thomas Dunnington
%Student ID Number: 109802853
%Date Created: 11/06/2021
%Date Modified: 4/11/2022
```

Main Code

```
%Clean up
close all; clear; clc;

%Reading in data
IspMat = readmatrix("Specific_Impulses.csv");
Isp = mean(IspMat);
Isp = Isp - 0.13; %Correction from static test stand data

%Provided constants and initial values
rhoWater = 1000;                                %kg/m^3
VolWaterInit = 0.001;                           %m^3

g = 9.81;
m0 = 0.125 + 1;
mf = 0.125;
Cd = 0.2;
DBottle = 0.105;
rhoAirAmb = 0.961;
startAngle = 45;
mu = 0.5;

constVec = [g;m0;mf;Cd;DBottle;rhoAirAmb;startAngle;mu];


wind = (10 / 2.237) * [cosd(270);sind(270);0]; %Zero wind baseline, 7
 mph SSW from 30 NE

%Calculating change in velocity
delV = Isp * g * log(m0 / mf);

%Getting initial parameters
```

```
x0 = 0;
y0 = 0;
z0 = 0.25;
vx0 = delV * cosd(startAngle);
vy0 = 0;
vz0 = delV * sind(startAngle);

%Initial State Vector
initStateODE = [x0;y0;z0;vx0;vy0;vz0];
tspan = [0 5];

%Creating the function handle, constVec is passed into the function, t
 and
%state are variable to the handle
ROCfunc = @(t,state) ROC(t,state,constVec,wind);

%Creating ode options to have a more accurate calculation
options = odeset('RelTol', 1e-8, 'AbsTol',1e-10);

%Calling ode45
[timeVec, finalMat] = ode45(ROCfunc, tspan, initStateODE, options);

% %Extracting integrated values from the ode45 output
xPosition = finalMat(:,1);
yPosition = finalMat(:,2);
zPosition = finalMat(:,3);

baseLineImpact = [xPosition(end), yPosition(end)];
```

Monte Carlo Simulation

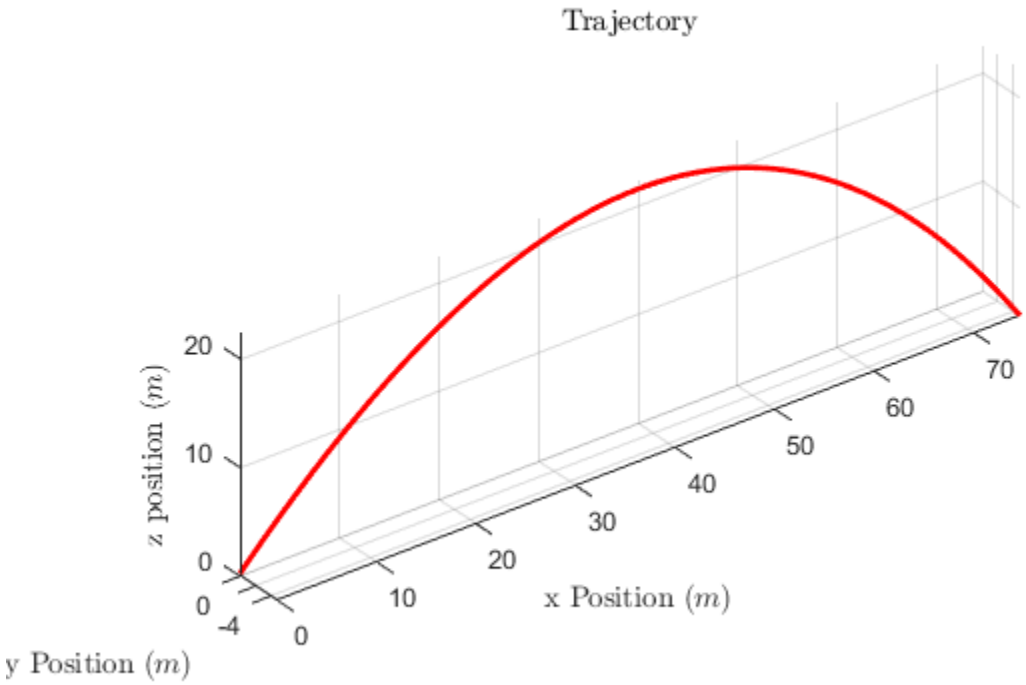
```
%Variables to vary
stdAngle = 1;
stdWater = 0.5 / 1000;
stdCd = 0.05;
stdRhoAirAmb = 0.05;
stdMu = 0.05;
stdIsp = 0.1;
stdWind = (5 / 2.237);

%Performing 500 simulations
monteCell = cell(1,500);
impactMat = zeros(500, 2);

for j = 1:500
    %Calculating random variation for the uncertain values
    randAngle = (2 * rand(1) - 1) * stdAngle;
    randWater = (2 * rand(1) - 1) * stdWater;
    randCd = (2 * rand(1) - 1) * stdCd;
    randRhoAirAmb = (2 * rand(1) - 1) * stdRhoAirAmb;
    randMu = (2 * rand(1) - 1) * stdMu;
    randIsp = (2 * rand(1) - 1) * stdIsp;
    randWindx = (2 * rand(1) - 1) * stdWind;
```

```
    randWindy = (2 * rand(1) - 1) * stdWind;

    %Constant Values
    m0 = 0.125 + 1 + randWater;
    mf = 0.125;
    Cd = 0.2 + randCd;
    rhoAirAmb = 0.961 + randRhoAirAmb;
    startAngle = 45 + randAngle;
    mu = 0.5 + randMu;

    constVec = [g;m0;mf;Cd;DBottle;rhoAirAmb;startAngle;mu];

    %Wind and Isp
    windMonte = wind + [randWindx; randWindy; 0];
    IspMonte = Isp + randIsp;

    %Performing integration
    %Calculating change in velocity
    delV = IspMonte * g * log(m0 / mf);

    %Getting initial parameters
    x0 = 0;
    y0 = 0;
    z0 = 0.25;
    vx0 = delV * cosd(startAngle);
    vy0 = 0;
    vz0 = delV * sind(startAngle);

    %Initial State Vector
    initStateODE = [x0;y0;z0;vx0;vy0;vz0];
    tspan = [0 5];

    %Creating the function handle, constVec is passed into the
 function, t and
    %state are variable to the handle
    ROCfunc = @(t,state) ROC(t,state,constVec,windMonte);

    %Creating ode options to have a more accurate calculation
    options = odeset('RelTol', 1e-8, 'AbsTol',1e-10);

    %Calling ode45
    [timeVec, finalMat] = ode45(ROCfunc, tspan, initStateODE,
 options);
    monteCell{j} = finalMat;

    %Getting the impact location
    impactMat(j,:) = finalMat(end, 1:2);
end
```

Plotting

```
figure(1)
set(0, 'defaultTextInterpreter', 'latex');
```

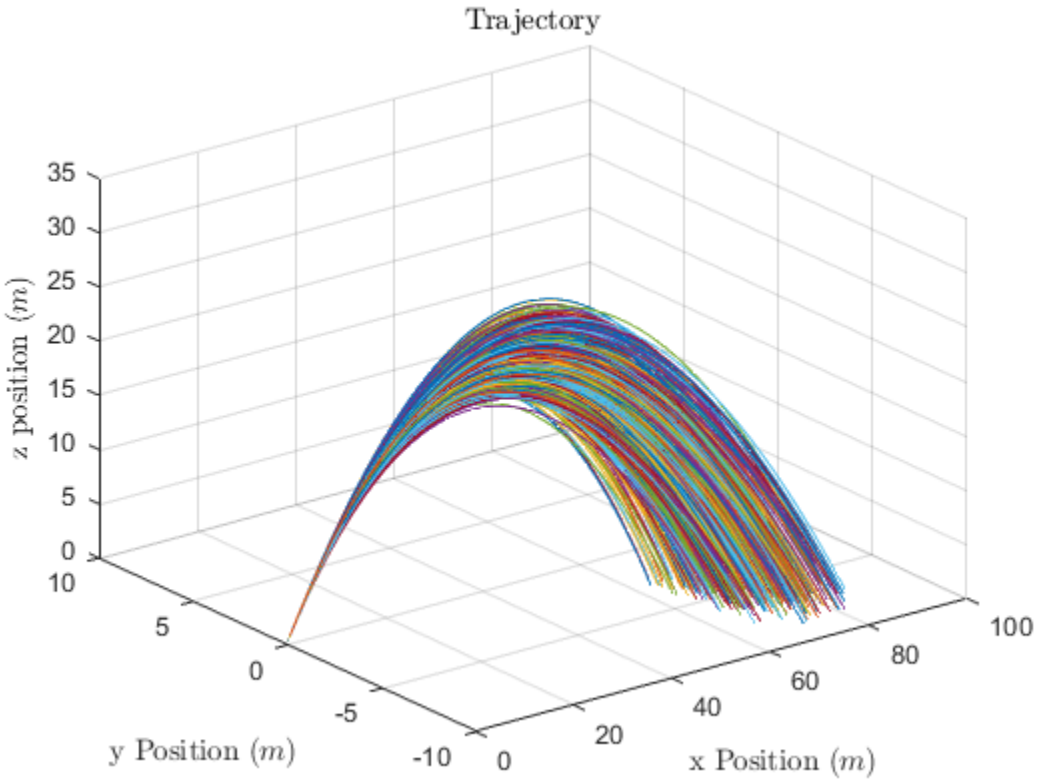
```
set(gca, 'FontSize', 12);
h = plot3(xPosition, yPosition, zPosition, 'r');
set(h,'defaultaxesfontname','cambria math');
h.LineWidth = 2;
grid on
hold on
xlim([0 90]);
ylim([-7 1]);
zlim([0 35]);
axis equal;
title('Trajectory');
xlabel('x Position ($m$)');
ylabel('y Position ($m$)');
zlabel('z position ($m$)');
```



Monte Plot

```
figure(2)
mat = monteCell{2};
xPos = mat(:,1);
yPos = mat(:,2);
zPos = mat(:,3);
plot3(xPos, yPos, zPos);
grid on
hold on
```

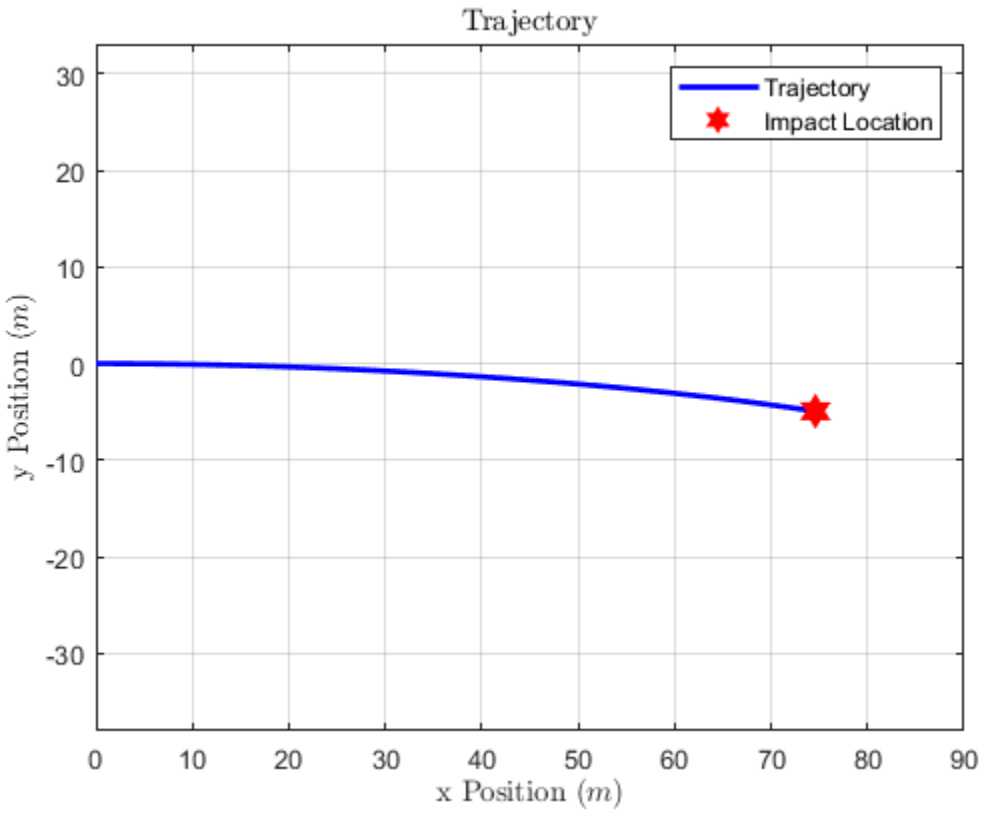
```
for j = 2:500
   mat = monteCell{j};
   xPos = mat(:,1);
   yPos = mat(:,2);
   zPos = mat(:,3);
   plot3(xPos, yPos, zPos);
end
xlim([0 100]);
ylim([-10 10]);
zlim([0 35]);
title('Trajectory');
xlabel('x Position ($m$)');
ylabel('y Position ($m$)');
zlabel('z position ($m$)');
```



Single Trajectory X/Y Plane

```
figure(3)
plot(xPosition, yPosition, 'LineWidth', 2, 'color', 'b');
xlim([0 90])
ylim([-20 20])
axis equal
grid on
hold on
plot(baseLineImpact(1), baseLineImpact(2), '-
h', 'LineStyle', 'none', 'color', 'r', 'MarkerSize',12, 'MarkerFaceColor', 'r');
```

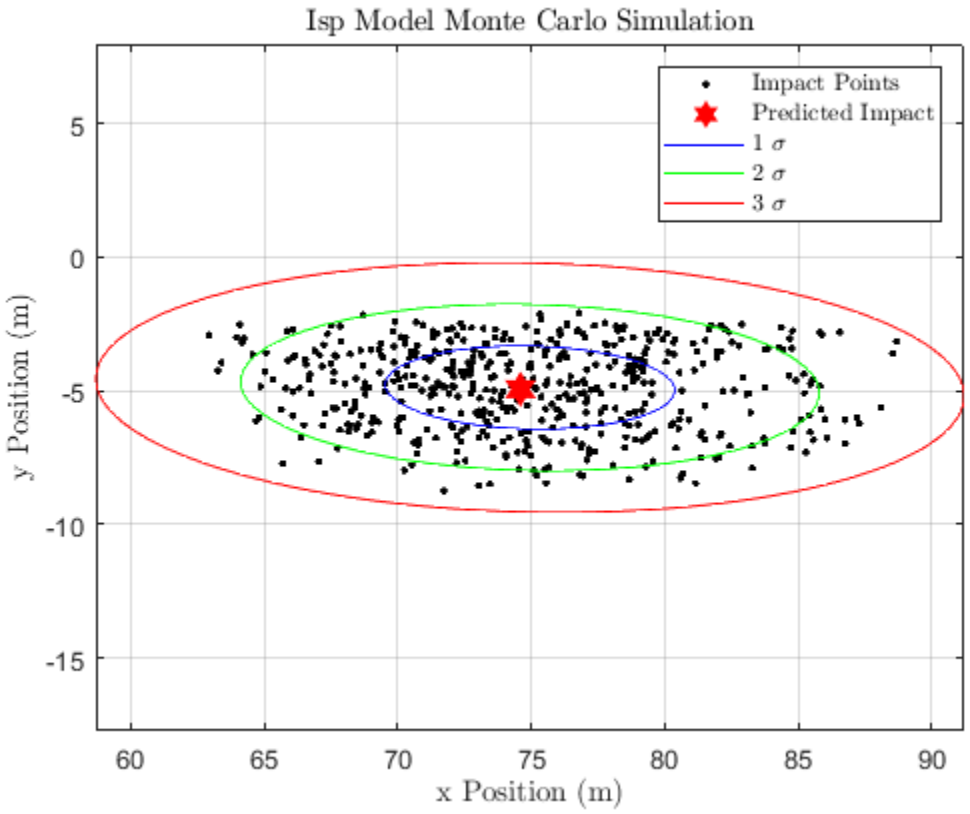
```
title('Trajectory');
xlabel('x Position ($m$)');
ylabel('y Position ($m$)');
legend('Trajectory', 'Impact Location');
```



Error Ellipse Plot

```
impacts = impactMat;
baseImpact = baseLineImpact;
xBase = baseImpact(1);
yBase = baseImpact(2);
x = impacts(:,1); % Randomly create some x data, meters
y = impacts(:,2); % Randomly create some y data, meters

figure(4); plot(x,y,'k.','markersize',6)
axis equal; grid on; xlabel('x Position (m)'); ylabel('y Position
 (m)'); hold on;
plot(xBase,yBase,'-
h', 'LineStyle', 'none', 'color', 'r', 'MarkerSize',12, 'MarkerFaceColor', 'r');

% Calculate covariance matrix
P = cov(x,y);
mean_x = mean(x);
mean_y = mean(y);

% Calculate the define the error ellipses
```

```
n=100; % Number of points around ellipse
p=0:pi/n:2*pi; % angles around a circle

[eigvec,eigval] = eig(P); % Compute eigen-stuff
xy_vect = [cos(p'),sin(p')] * sqrt(eigval) * eigvec'; % Transformation
x_vect = xy_vect(:,1);
y_vect = xy_vect(:,2);

% Plot the error ellipses overlaid on the same figure
plot(1*x_vect+mean_x, 1*y_vect+mean_y, 'b')
plot(2*x_vect+mean_x, 2*y_vect+mean_y, 'g')
plot(3*x_vect+mean_x, 3*y_vect+mean_y, 'r')

title('Isp Model Monte Carlo Simulation');
legend('Impact Points', 'Predicted Impact', '1 $\sigma$', '2 $\sigma
$', '3 $\sigma$', 'Interpreter','latex');
```



Rate of Change Function

```
function [dX] = ROC(t,state,const,wind)
% Purpose: To calculate the rate of change of various parameters along
% the flight trajector of a water rocket
%
% Inputs:   t = anonymous time variable
%         state = anonymous state vector containing
%            = [x, y, z, vx, vy, vz]
```

```
%           const = vector to hold constant values
%           wind = wind vector that holds the corresponding change in
 vel
% Outputs: dX = derivative state vector containing
%             =  [vx,vy,vz,d2x,d2y,d2z]
%

%Constants from constant vector
g = const(1);
m0 = const(2);
mf = const(3);
Cd = const(4);
DBottle = const(5);
rhoAirAmb = const(6);
startAngle = const(7);
mu = const(8);

%Calculating area
areaBottle = pi*(DBottle / 2)^2;

%Getting initial states
x = state(1);
y = state(2);
z = state(3);
vx = state(4);
vy = state(5);
vz = state(6);

%Velocity vector and heading vector
vVec = [vx; vy; vz];
vHeading = vVec / norm(vVec);


%When the rocket is on the stand this is the heading
%It is fixed at an angle of 45 degrees, the stand is 0.5 meters long
if z < (0.25 + 0.5*cosd(45)) && t < 1
    vHeading = [cosd(startAngle); 0; sind(startAngle)];
    fGrav = [0; 0; -mf * g];
    fDrag = -((1/2)*rhoAirAmb*(norm(vVec))^2*Cd*areaBottle) *
 vHeading;

    %Adding friction on the rails
    FricMag = mu * mf * g * cosd(startAngle);
    fFric = -1 * abs(FricMag * vHeading);

elseif z <= 0 %Condition for rocket hitting the ground
    %All rates of change go to zero
    fGrav = [0; 0; 0];
    fDrag = [0; 0; 0];
    fFric = [0; 0; 0];
    vVec(1) = 0;
    vVec(2) = 0;
    vVec(3) = 0;
```

```
else
    vHeading = (vVec - wind) ./ norm((vVec - wind));
    fGrav = [0; 0; -mf * g];
    fDrag = -((1/2)*rhoAirAmb*(norm(vVec))^2*Cd*areaBottle) *
 vHeading;
    fFric = [0;0;0];
end

%Calculating the net force by summing all forces
fnet = fGrav + fDrag + fFric;
a = fnet ./ mf; %acceleration from F=ma

%Outputting the thrust and the rate of change vector
dX = [vVec(1); vVec(2); vVec(3); a(1); a(2); a(3)];
end
```

Published with MATLAB® R2020b