Lenny Martinez
16.90 Project 3 Report
May 12, 2017

I. Estimation of the mission outcomes

Part 1.

Code is uploaded in a zip file, and attached at the end of report. Attached in print are two pages explaining how I got the triangular distribution to work.

Part 2.

Using N=1,000 the following figures are histogram of the inputs for the simulation.

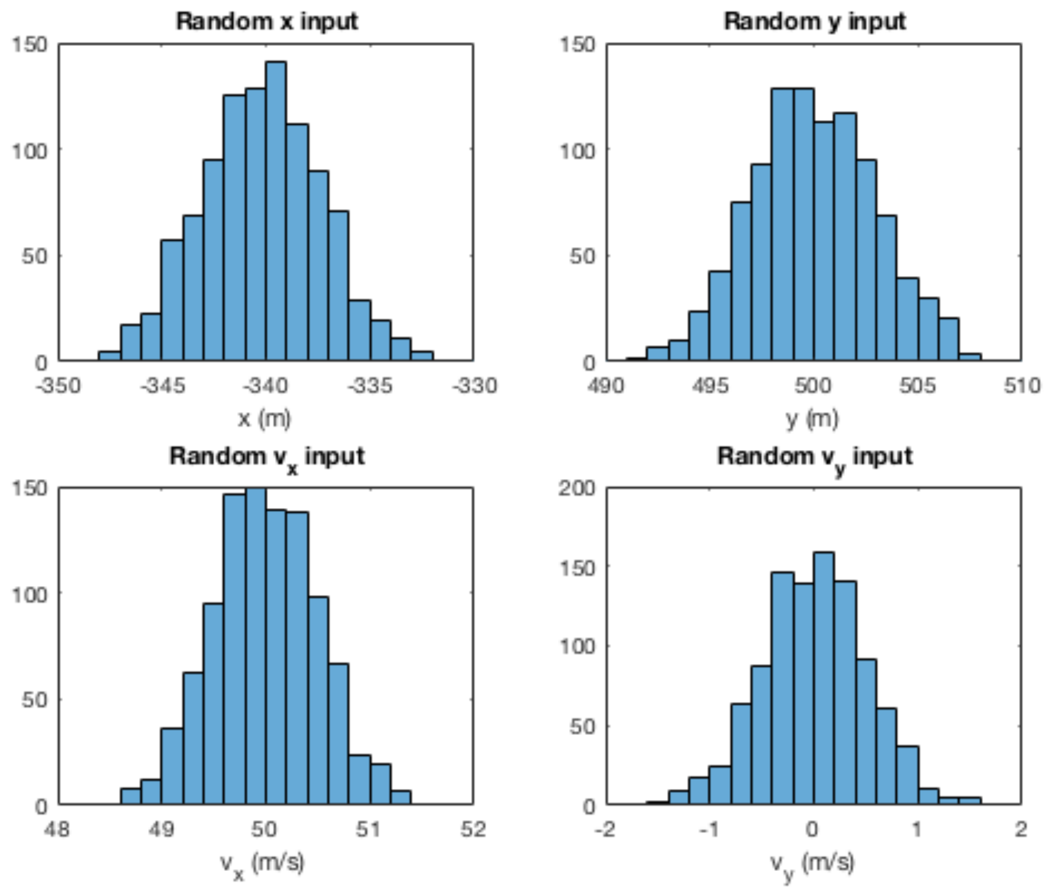

Figure 1. Histograms of simulation inputs for position and velocity inputs in the x and y directions.

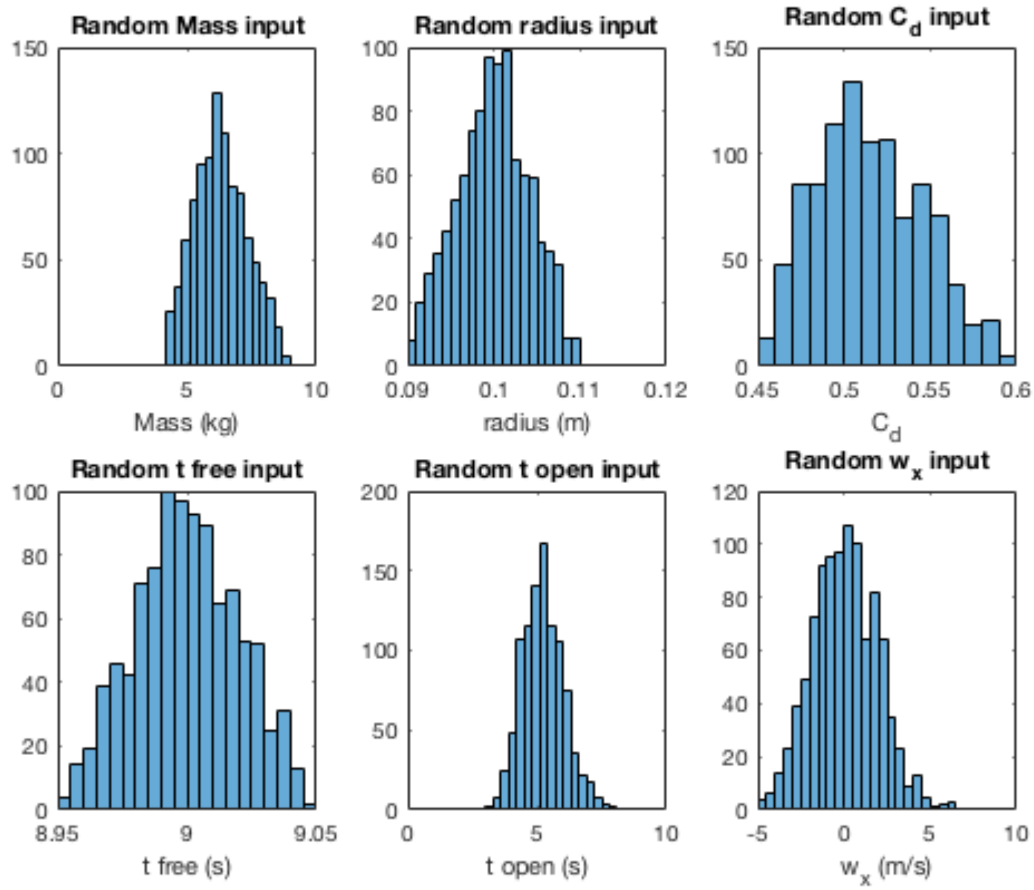

Figure 2. Histograms of other simulation inputs.

The randomly generated inputs match the expectations. Twenty simulated trajectories are show in Figure 3.

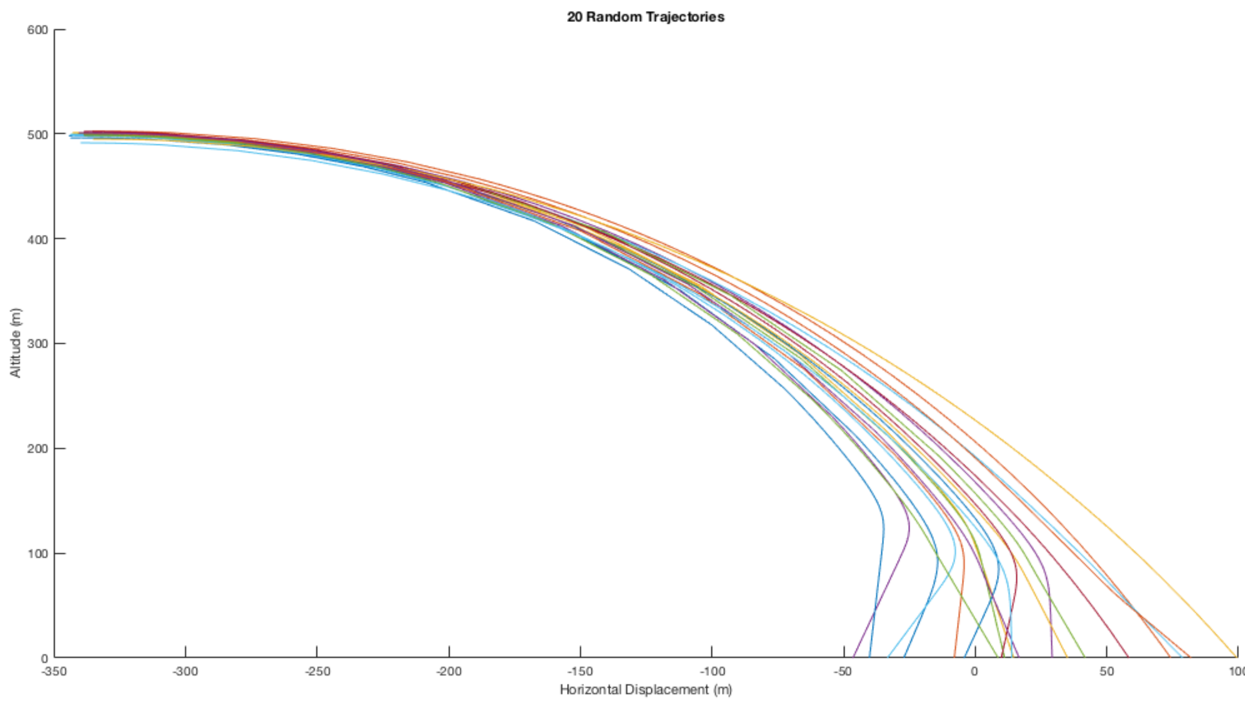

Figure 3. Twenty simulated trajectories generated and plotted using samples from input distributions.

Part 3.

Running the simulation with N = 1000, we got the probabilities:

	Inside	Outside
Intact	A: 0.5210	B: 0.1390
Destroyed	C: 0.1600	D: 0.1800

For this run, the standard deviation of p_A is given by:

$$\sigma_{p_A} = sqrt\left[\frac{p_A \cdot (1 - p_A)}{N}\right] = \; 0.0157974$$

We can use this to check if N is within ±0.01 at 99% confidence using the equation:

$$\frac{2.575 \cdot \sigma_{p_A}}{\sqrt{N}} \leq 0.01$$

Putting in variance and N, the left hand side is 0. 0.00128636 which is less than 0.01. Put another way, our sample size N works.

Part 4.

Figures 4 and 5 are histograms of the payload landing sites and impact velocities respectively.

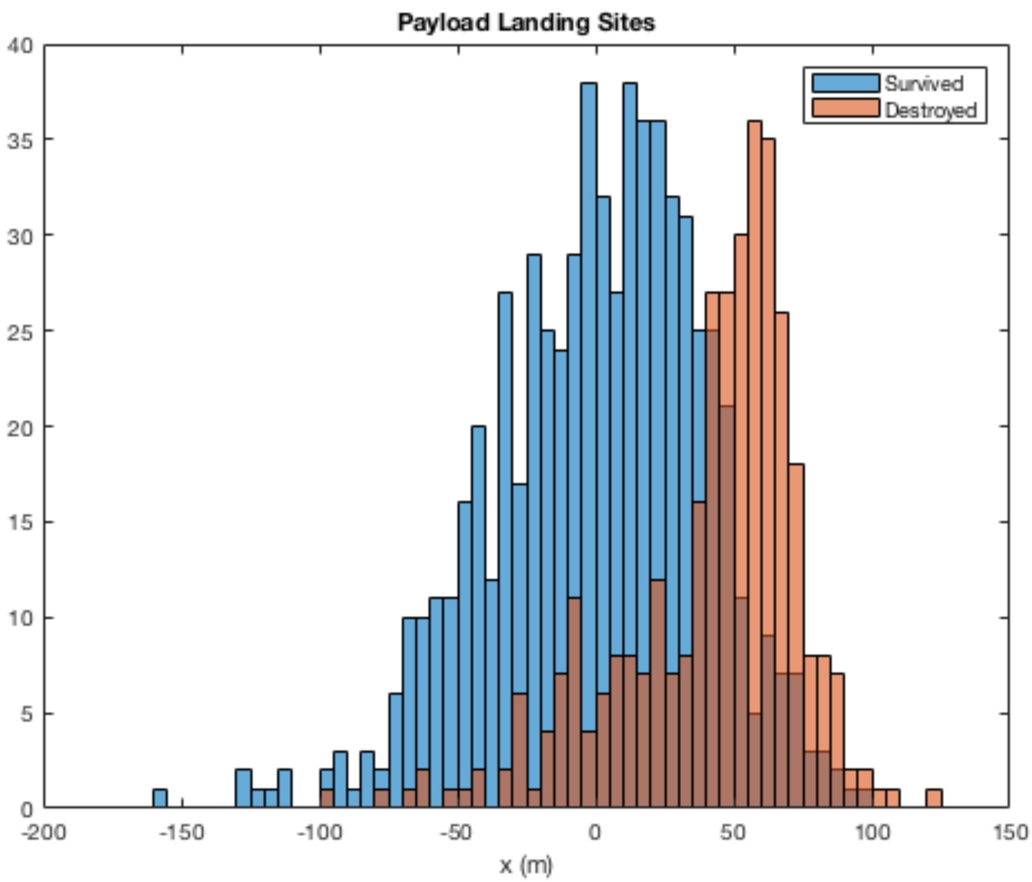

Figure 4. Histogram of payload landing sites

The histogram of payload landing sites makes sense in that the simulations are designed to safely drop the payload at a target location so most payloads that fall at or before the target have a higher chance of surviving the landing. On the same thought, payloads that land further ahead than the target location are more likely to be destroyed because they may be falling too fast.

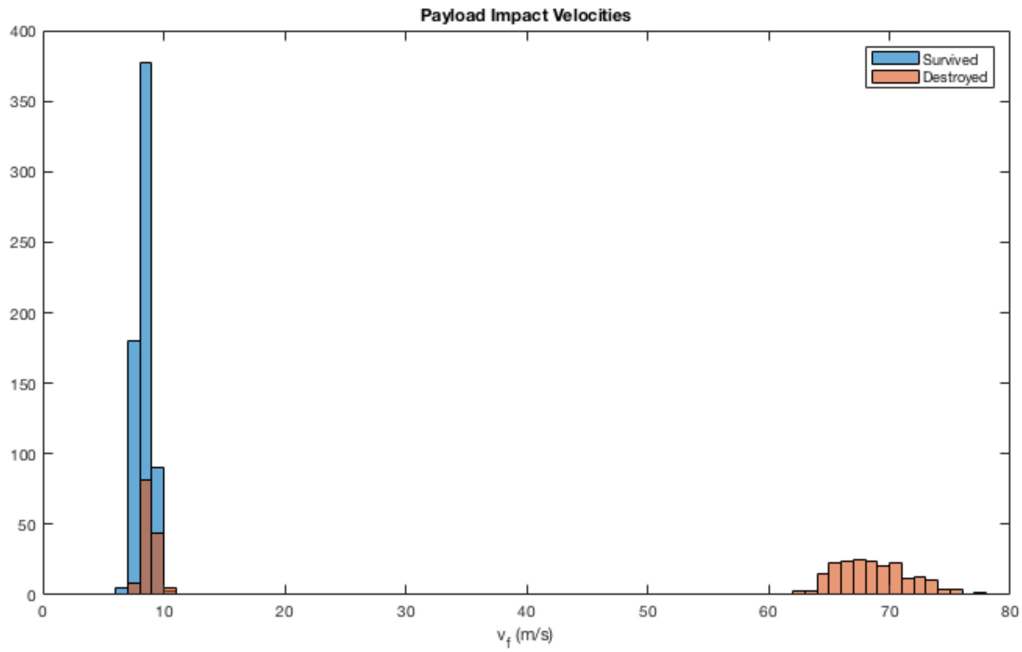

Figure 5. Histogram of payload impact velocities

The impact velocity histogram makes sense, because higher impact velocities are more likely to result in destroyed payload since they're a result of having the parachute opened for less time. Destroyed payloads at low impact velocities are likely a result of the simulations starting too close to the ground where the parachute being opened didn't help decrease the velocity enough.

I. Design under Uncertainty

Using N = 1000, I ran the simulations with each of the following implementations and am presenting $p_{success}$ which is $p_A \equiv P(inside, intact)$

1. Improved payload shock absorption – Increases the survival probability of payloads impacting at higher velocities so that μ = 2.7 in the survival function S(vf).

$$p_{success} = 0.6210 \; or \; 62.10\%$$

2. Stricter regulations on payload weight – Restricts extremely underweight and over- weight payloads so that the input distribution becomes m ~ Triangular(5,6,7).

$$p_{success} = 0.6480 \; or \; 64.80\%$$

3. Improved parachute deployment – Reduces the variance of parachute opening time so that the input distribution becomes topen ~ ln N (1.75, (0.1)2).

$$p_{success} = 0.2860 \; or \; 28.60\%$$

If all three changes are applied at the same time, $p_{success} = 0.8340 \; or \; 83.40\%$

In terms of design recommendations from just these four trials, I'd recommend implementing all 3 ideas as the probability of success is the highest in this scenario. But I'm also curious to know if trying combinations in pair (1&2, 1&3, 1&2) would increase the success probability.

II. Importance Sampling

Part 1.
After running the simulation 20 times, and using the original mission parameters, I got the following success probabilities, p_A:

0.0150, 0.0160, 0.0190, 0.0240, 0.0190, 0.0250, 0.0130, 0.0200, 0.0210, 0.0200,
0.0210, 0.0170, 0.0170, 0.0250, 0.0170, 0.0170, 0.0220, 0.0140, 0.0200, 0.0150

The average p_A from this set is 0.189 and the variance is 1.2345e-05.

When ran with the new mission parameters (implementing all three design ideas), the series of successful probabilities is:

0.0240, 0.0180, 0.0290, 0.0290, 0.0260, 0.0200, 0.0180, 0.0120, 0.0220, 0.0200,
0.0220, 0.0240, 0.0210, 0.0310, 0.0120, 0.0230, 0.0160, 0.0290, 0.0140, 0.0170

The average p_A from this set is 0.0214 and the variance is 3.2134e-05.

Part 2.
After implementing an appropriate biasing distribution, the average importance sampling estimate was 0.2651 with a variance of 0.0988.

Part 3.
Yes, there was The Force.

Appendix A: Matlab code

logrand.m

```
function x = logrand(mu, sigma)
% draw sample from a log normal distribution ln N(mu, sigma^2)
z = rand;
x = exp(mu + sigma*z);
end
```

trirand.m

```
function x = trirand(a, b, c)
% draw a sample from a triangular distribution with min=a, mode=b, max=c
a1 = a;
b1 = c;
c1 = b;

z = rand;

area = (c1-a1)/(b1-a1);

if z < area
    x = a1 + sqrt(z*(b1-a1)*(c1-a1));
elseif z > area
    x = b1 - sqrt((1-z)*(b1-a1)*(b1-c1));
else
    x = c1;
end

end
```

project3.m

```
% % nominal values of parameters
clear all
close all
% m      = 6;    % mass of payload and parachute assembly kilograms
% r      = 0.1;  % radius of payload in meters
```

```
% Cd      = 0.5;  % payload coefficient of drag
% wx      = 0;    % horizontal wind speed in m/s
% tfree   = 9;    % time before parachute opens
% topen   = 5;    % time it takes for parachute to open completely
%
% % initial conditions x(0), y(0), vx(0), vy(0)
% u0 = [-340, 500, 50, 0];
%
% % compute a single trajectory
% [t, u] = payload_sim(u0, m, r, Cd, wx, tfree, topen);
%
% % plot the trajectory
% close all;
% figure(1);
% plot(u(:,1), u(:,2), '-')
% xlabel('horizontal displacement (m)');
% ylabel('altitude (m)');
%
% % plot the velocity over time
% figure(2);
% plot(t, sqrt(u(:,3).^2 + u(:,4).^2),'-')
% xlabel('time after release (s)');
% ylabel('velocity (m/s)');

%% Monte Carlo Simulation

% Monte Carlo loop
N = 1000;

% initialize input samples
x     = zeros(N,1);
y     = zeros(N,1);
vx    = zeros(N,1);
vy    = zeros(N,1);
u0    = zeros(N,4);
m     = zeros(N,1);
r     = zeros(N,1);
Cd    = zeros(N,1);
tfree = zeros(N,1);
topen = zeros(N,1);
```

```
wx    = zeros(N,1);

% initialize output samples
xf     = zeros(N,1); % landing sites (m)
vf     = zeros(N,1); % impact velocities (m/s)
intact = zeros(N,1); % payload intact after impact (0, 1)


  for i=1:N % consider using parfor, especially for large N


    % initial conditions
    x(i)     = normrnd(-340, 3);
    y(i)     = normrnd(500, 3);
    vx(i)    = normrnd(50, 0.5);
    vy(i)    = normrnd(0, 0.5);

    % random inputs
    m(i)     = trirand(5,6,7);
    r(i)     = trirand(0.09,0.1,0.11);
    Cd(i)    = trirand(0.45,0.5,0.6);
    tfree(i) = trirand(8.95,9,9.05);
    topen(i) = lograndd(1.75,0.1);
    wx(i)    = normrnd(0,2);


    u0(i,:)  = [x(i), y(i), vx(i), vy(i)];
    % simulate trajectory
    [t, u] = payload_sim(u0(i,:), m(i), r(i), Cd(i), wx(i), tfree(i), topen(i));

    % get the impact location and impact velocity
    xf(i) = u(end,1);
    vf(i) = sqrt(u(end,3)^2 + u(end,4)^2);

    % check if payload survived impact
    if rand() < survival(vf(i), 2.7, 0.15),
      intact(i) = 1;
    end

    if i<=20,
```

```
        hold on
        plot(u(:,1), u(:,2), '-')
        title('20 Random Trajectories')
        xlabel('Horizontal Displacement (m)');
        ylabel('Altitude (m)');
    end
end

% hold off
% % Plot histograms of input parameters
% figure()
% subplot(2,2,1); histogram(x)
% title('Random x input')
% xlabel('x (m)')
%
% subplot(2,2,2); histogram(y)
% title('Random y input')
% xlabel('y (m)')
%
% subplot(2,2,3); histogram(vx)
% title('Random v_x input')
% xlabel('v_x (m/s)')
%
% subplot(2,2,4); histogram(vy)
% title('Random v_y input')
% xlabel('v_y (m/s)')
%
% figure()
% subplot(2,3,1);histogram(m)
% title('Random Mass input')
% xlabel('Mass (kg)')
%
% subplot(2,3,2); histogram(r)
% title('Random radius input')
% xlabel('radius (m)')
%
% subplot(2,3,3);histogram(Cd)
% title('Random C_d input')
% xlabel('C_d')
%
```

```
% subplot(2,3,4);histogram(tfree)
% title('Random t free input')
% xlabel('t free (s)')
%
% subplot(2,3,5);histogram(topen)
% title('Random t open input')
% xlabel('t open (s)')
%
% subplot(2,3,6);histogram(wx)
% title('Random w_x input')
% xlabel('w_x (m/s)')
%% Monte Carlo Analysis

% Below is starting point for conducting your Monte Carlo analysis

inside = find(abs(xf) < 1);
outside = find(abs(xf) > 1);

survived = find(intact == 1);
destroyed = find(intact == 0);

survived_inside = intersect(survived, inside);
survived_outside = intersect(survived, outside);
destroyed_inside = intersect(destroyed, inside);
destroyed_outside = intersect(destroyed, outside);

% probability of success (outcome A)
p_A = length(survived_inside) / N;
p_B = length(survived_outside) / N;
p_C = length(destroyed_inside) / N;
p_D = length(destroyed_outside) / N;

p = [ p_A, p_B; p_C, p_D];

% % plot histogram of landing sites of intact payloads
% figure()
% histogram(xf(survived),'BinWidth',5);
% hold on
% histogram(xf(destroyed),'BinWidth',5);
% hold off
```

```
% xlabel('x (m)');
% title('Payload Landing Sites')
% legend('Survived', 'Destroyed')
%
% figure()
% histogram(vf(survived),'BinWidth',1);
% hold on
% histogram(vf(destroyed),'BinWidth',1);
% hold off
% xlabel('v_f (m/s)');
% title('Payload Impact Velocities')
% legend('Survived', 'Destroyed')
```

project3_biasing.m

```
%% Monte Carlo Simulation

close all; clear all;

% Monte Carlo loop
N = 1000;
load 'biasing_dist'
% draw N samples from the biasing distribution as a Nx10 matrix
samples = mvnrnd(biasing_dist.mean, biasing_dist.cov, N);

% compute pdf of the biasing distribution q(...) so that q is a Nx1 vector
% where q(i) is the density evaluated at samples(i,:)
q = mvnpdf(samples, biasing_dist.mean, biasing_dist.cov);

% initialize input samples
x     = zeros(N,1);
y     = zeros(N,1);
vx    = zeros(N,1);
vy    = zeros(N,1);
u0    = zeros(N,4);
m     = zeros(N,1);
r     = zeros(N,1);
Cd    = zeros(N,1);
tfree = zeros(N,1);
topen = zeros(N,1);
```

```
wx    = zeros(N,1);
p     = zeros(N,10);
u0    = zeros(N,4);
% initialize output samples
xf     = zeros(N,1); % landing sites (m)
vf     = zeros(N,1); % impact velocities (m/s)
intact = zeros(N,1); % payload intact after impact (0, 1)


    for i=1:N % consider using parfor, especially for large N

        % Draw samples from the input distributions
        % unpack row i of the samples

        m(i)       = samples(i,1);
        r(i)       = samples(i,2);
        Cd(i)      = samples(i,3);
        wx(i)      = samples(i,4);
        tfree(i)   = samples(i,5);
        topen(i)   = samples(i,6);
        x(i)       = samples(i,7);
        y(i)       = samples(i,8);
        vx(i)      = samples(i,9);
        vy(i)      = samples(i,10);


        % TODO: evaluate pdf of "nominal" distribution p(...) at samples(i,:)
        % Evaluate all log-normal random variables: topen
        p(i,5) = (1/(topen(i)*0.15*sqrt(2*pi())))*exp((log(topen(i))-1.65)^2/(-
2*(0.15^2)));

        % Evaluate all the normal random variables: x,y,vx,vy,wx
        p(i,7) = (1/sqrt(2*pi()*(3.0^2)))*exp((x(i)+340)^2/(-2*(3.0^2)));
        p(i,8) = (1/sqrt(2*pi()*(3.0^2)))*exp((y(i)-500)^2/(-2*(3.0^2)));
        p(i,9) = (1/sqrt(2*pi()*(0.5^2)))*exp((vx(i)-50)^2/(-2*(0.5^2)));
        p(i,10) = (1/sqrt(2*pi()*(0.5^2)))*exp((vy(i)-0)^2/(-2*(0.5^2)));
        p(i,4) = (1/sqrt(2*pi()*(2.0^2)))*exp((wx(i)-0)^2/(-2*(2.0^2)));

        %Evaluate all the triangular random variables: m,r,Cd,tfree
        j = m(i); a= 4.0; b = 6.0; c= 9.0;
```

```
if x < a
    p(i,1) = 0;
elseif x > c
    p(i,1) = 0;
elseif x >= a && x < b
    p(i,1) = (2*(x-a))/((c-a)*(b-a));
elseif x == b
    p(i,1) = 2/(c-a);
else
    p(i,1) = 2*(c-x)/((c-a)*(c-b));
end

j = r(i); a= 0.09; b = 0.10; c= 0.11;
if x < a
    p(i,2) = 0;
elseif x > c
    p(i,2) = 0;
elseif x >= a && x < b
    p(i,2) = (2*(x-a))/((c-a)*(b-a));
elseif x == b
    p(i,2) = 2/(c-a);
else
    p(i,2) = 2*(c-x)/((c-a)*(c-b));
end

j = Cd(i); a= 0.45; b = 0.5; c= 0.60;
if x < a
    p(i,3) = 0;
elseif x > c
    p(i,3) = 0;
elseif x >= a && x < b
    p(i,3) = (2*(x-a))/((c-a)*(b-a));
elseif x == b
    p(i,3) = 2/(c-a);
else
    p(i,3) = 2*(c-x)/((c-a)*(c-b));
end

j = tfree(i); a= 8.95; b = 9; c= 9.05;
if x < a
```

```
        p(i,5) = 0;
    elseif x > c
        p(i,5) = 0;
    elseif x >= a && x < b
        p(i,5) = (2*(x-a))/((c-a)*(b-a));
    elseif x == b
        p(i,5) = 2/(c-a);
    else
        p(i,5) = 2*(c-x)/((c-a)*(c-b));
    end

    u0(i,:)  = [x(i), y(i), vx(i), vy(i)];
    % simulate trajectory
    [t, u] = payload_sim(u0(i,:), m(i), r(i), Cd(i), wx(i), tfree(i), topen(i));

    % get the impact location and impact velocity
    xf(i) = u(end,1);
    vf(i) = sqrt(u(end,3)^2 + u(end,4)^2);

    % check if payload survived impact
    if rand() < survival(vf(i), 2.7, 0.15),
        intact(i) = 1;
    end

end

%% Monte Carlo Analysis

% Below is starting point for conducting your Monte Carlo analysis

inside = find(abs(xf) < 1);
outside = find(abs(xf) > 1);

survived = find(intact == 1);
destroyed = find(intact == 0);

survived_inside = intersect(survived, inside);
survived_outside = intersect(survived, outside);
destroyed_inside = intersect(destroyed, inside);
destroyed_outside = intersect(destroyed, outside);
```

```
    % probability of success (outcome A)
    p_A = length(survived_inside) / N;
    p_B = length(survived_outside) / N;
    p_C = length(destroyed_inside) / N;
    p_D = length(destroyed_outside) / N;

    prob = [ p_A, p_B; p_C, p_D];

%% Importance Sampling Analysis
success = survived_inside;

% f is the indicator function of a successful payload
f = zeros(N,1);
f(success) = 1;

% compute the importance sampling estimate as sum(f.*p./q) / N
E = sum(f.*p./q)/N;
```