Table of Contents

Roshan Jaiswal-Ferri

```
%Section - 01
%Aero 421 HW4: 5/9/25
```

Workspace Prep

```
%warning off
format long       %Allows for more accurate decimals
close all;        %Clears all
clear all;        %Clears Workspace
clc;              %Clears Command Window
```

Variables

```
Re = 6378; %km
mu = 398600;
JD = 2458981.1666667;
c = [0; 0; 0];
n = 1000;

r0 = [5980.8297; -1282.3184; 4125.8019];  % km in ECI
v0 = [1.540887; 7.186813; 0];  % km/s

target.lat = 35.3;        % degrees
target.lon = -120.8;      % degrees
target.alt = 0.2;         % km

target_eci = lla2eci(target.lat, target.lon, target.alt, JD); %T_0
T_0 = target_eci;

p_0 = T_0 - r0;
phat_0 = p_0/norm(p_0);


%-------------------------------ERRORS-------------------------------------
```

```
% Local Sidereal Time
error.lsdErr = 0.01; % seconds
error.JDError = 0.01/86400; % now in days which is jd or somethin idk

% Target Location
error.LatErr = deg2rad(1e-4); % degrees to read
error.LonErr = deg2rad(1e-4); % degrees to rad
error.altErr = 10/1000; % kilometers

% Spacecraft Knowledge
error.rErr = 3/1000; % meters (in-track, cross-track, radius)
error.vErr = 0.002/1000; % m/s (in-track, cross-track)
error.vErr_r = 0.007/1000; % m/s (radial)

% Sensor Mounting Errors
error.smErr = 0.01/1000; % meters
error.smaErr = deg2rad(1e-4); % degrees to rad
```

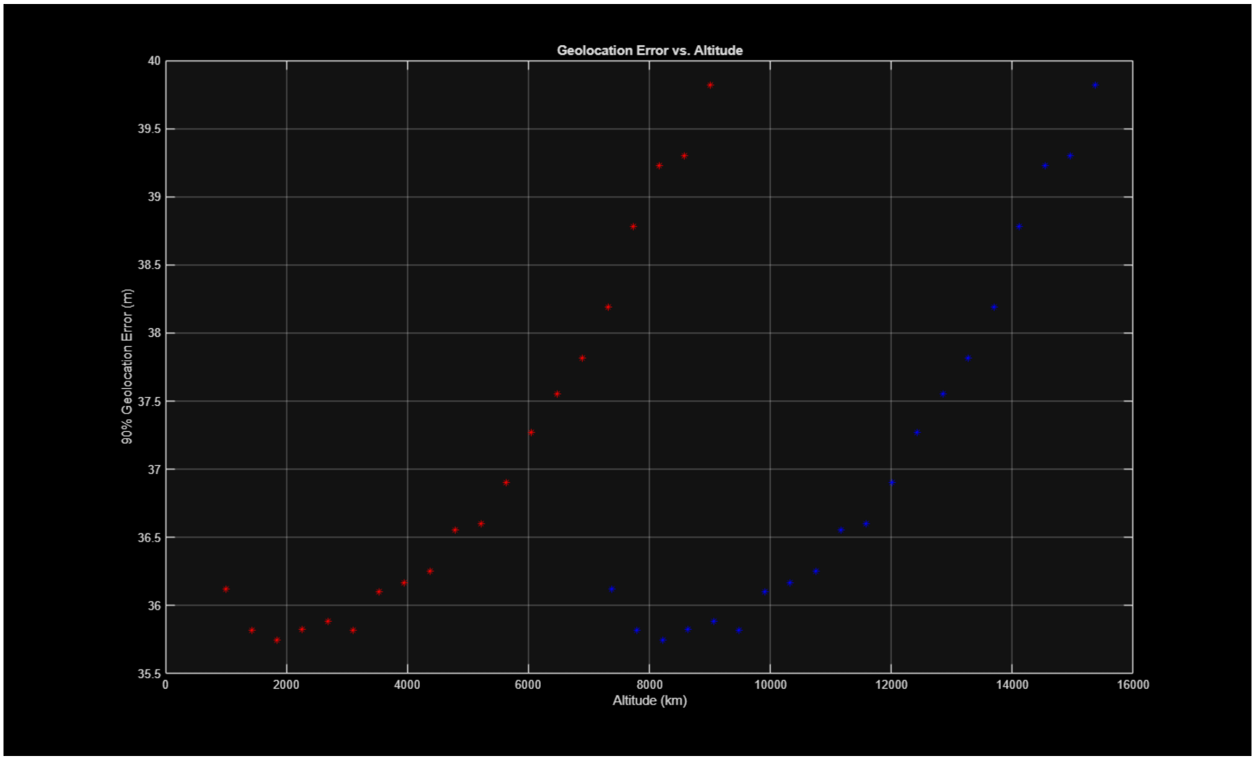
Problem 1

```
[g, g_norm, T_g,e] = geo_monte_sim(r0, v0, phat_0, T_0, JD, error, target,
Re, n, c);

%----This is for problem 4 but the inputs change so calculated up here-----

errorNames = fieldnames(error);
for i = 1:length(errorNames)
    errorNew = error;

    for j = 1:length(errorNames) %setting each non-selected error to zero
        if j ~= i
            errorNew.(errorNames{j}) = 0;
        end
    end

    [~, confidence90(i), ~, ~] = geo_monte_sim(r0, v0, phat_0, T_0, JD,
error, target, Re, n, c);

end

%--------------------------------------------------------------------------

disp('Part 1:')
disp(['g vector in meters: ', num2str(g')])
disp(['gnorm: ', num2str(g_norm')])
disp(['Tg (km): ', num2str(T_g')])
disp(' ')
```

Problem 2

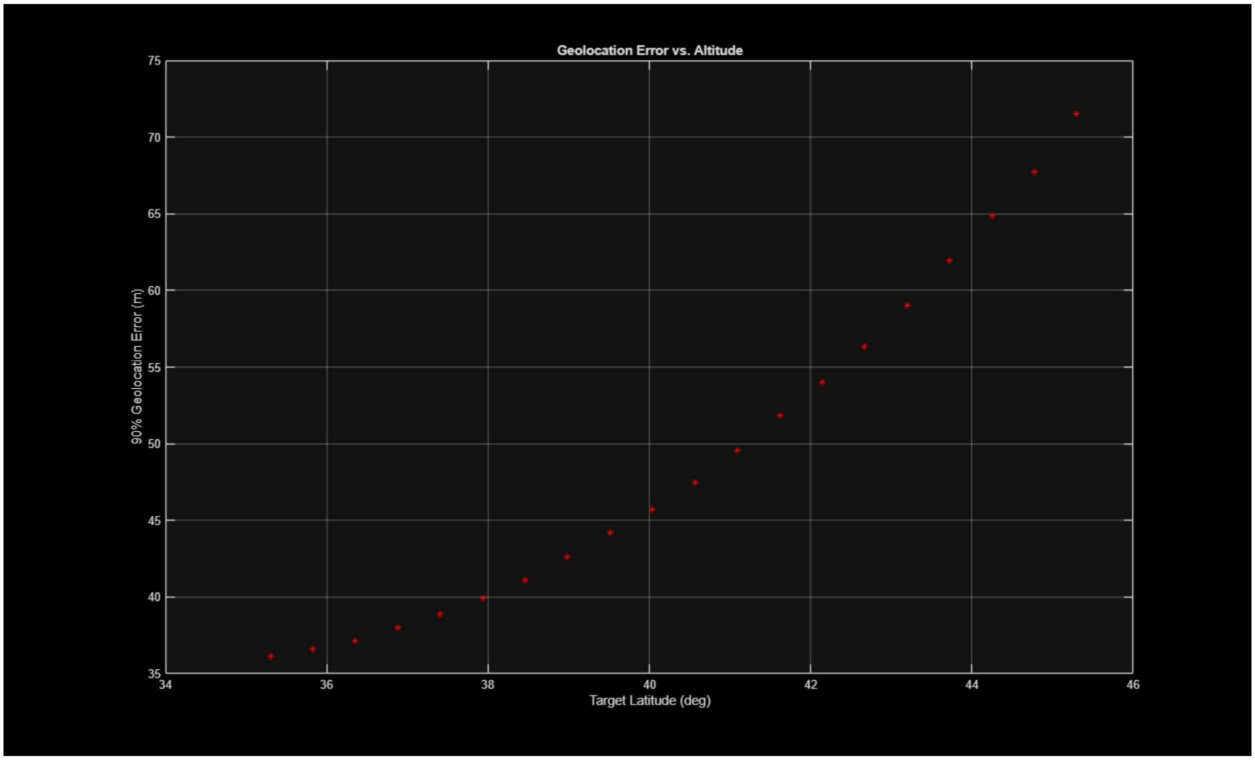
```
altVec = linspace(1000,9000,20);
g_errors = zeros(length(altVec), 1);
slant_ranges = zeros(length(altVec), 1);
```

```
rhat = r0/norm(r0);
vhat = v0/norm(v0);

for i = 1:length(altVec)
    rmag(i) = Re + altVec(i);
    vmag = sqrt(mu/rmag(i));

    r02 = rmag(i)*rhat;
    v02 = vmag*vhat;

    p_02 = T_0 - r02; %update pointing vector
    phat_02 = p_02/norm(p_02);

    [~, loc_error, Tg, mcData] = geo_monte_sim(r02, v02, phat_02, T_0, JD,
error, target, Re, n, c);

    %dp(:,i) = mcData.p - p_02;
    dp(:,i) = mcData.r;
    dpn(i) = norm(dp(:,i));
    dr(i) = norm(mcData.r);
    g_errors(i) = loc_error;
    slant_ranges(i) = norm(T_0 - r02);

end

figure;
plot(altVec, g_errors, 'r*');
hold on
grid on
plot(dpn,g_errors, 'b*');
xlabel('Altitude (km)');
ylabel('90% Geolocation Error (m)');
title('Geolocation Error vs. Altitude');
```



Problem 3

```
latVec = linspace(35.3,45.3,20);
g_errors = zeros(length(altVec), 1);

for i = 1:length(latVec)
    target.lat = latVec(i);
    T_02 = lla2eci(target.lat, target.lon, target.alt, JD); %T_02

    p_03 = T_02 - r0; %update pointing vector
    phat_03 = p_03/norm(p_03);

    [~, lat_error, Tg, mcData] = geo_monte_sim(r0, v0, phat_03, T_0, JD,
error, target, Re, n, c);
    g_errors(i) = lat_error;

end

figure;
plot(latVec, g_errors, 'r*');
hold on
grid on
xlabel('Target Latitude (deg)');
ylabel('90% Geolocation Error (m)');
title('Geolocation Error vs. Altitude');
```

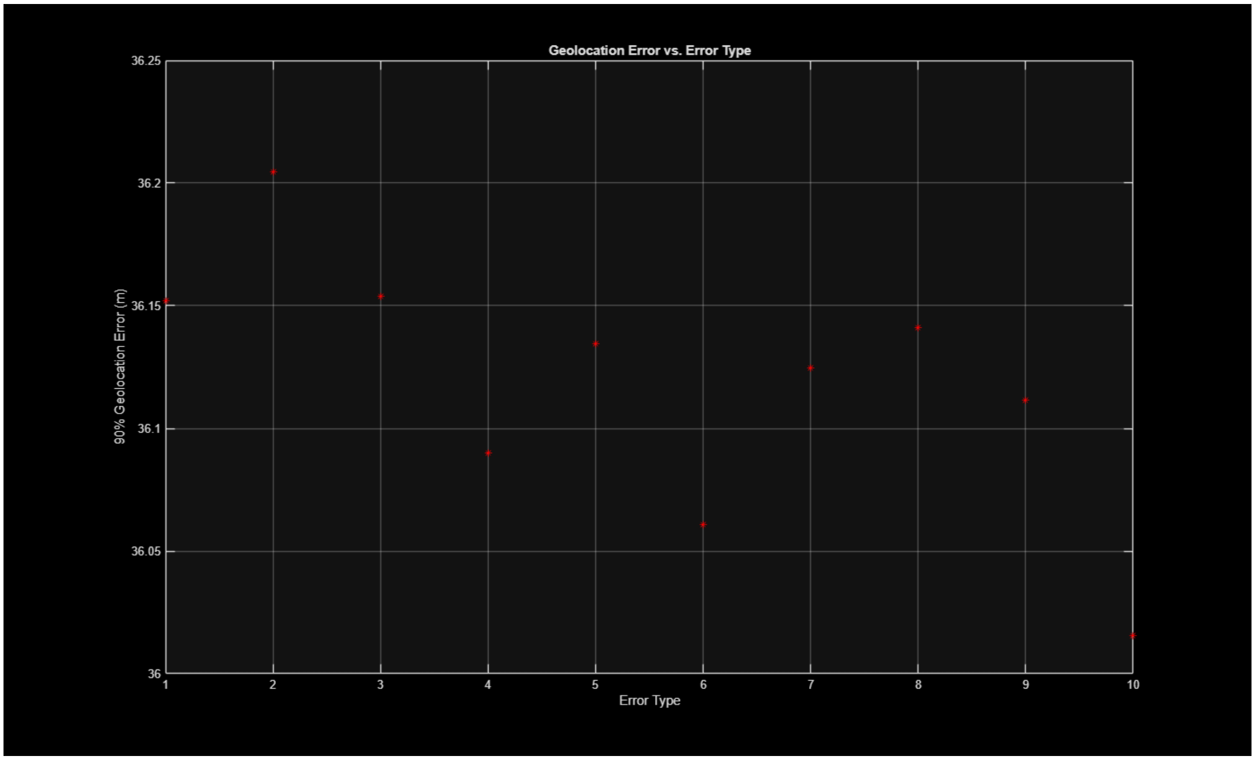


Problem 4

```
% Display the error table
fprintf('\nPart 4: Individual Error Contributions to 90%% Geolocation Error
(m)\n');
fprintf('-------------------------------------------------------------\n');
fprintf('%-30s %10s\n', 'Error Source', '90% Error (m)');
fprintf('-------------------------------------------------------------\n');
for i = 1:length(errorNames)
    fprintf('%-30s %10.3f\n', errorNames{i}, confidence90(i));
end
fprintf('-------------------------------------------------------------\n');
disp(' ')

figure;
plot(linspace(1,10,10), confidence90, 'r*');
hold on
grid on
xlabel('Error Type');
ylabel('90% Geolocation Error (m)');
title('Geolocation Error vs. Error Type');
```

```
Part 4: Individual Error Contributions to 90% Geolocation Error (m)
-------------------------------------------------------------
Error Source                       90% Error (m)
-------------------------------------------------------------
lsdErr                                 36.152
JDError                                36.205
```

```
LatErr                                  36.154
LonErr                                  36.090
altErr                                  36.135
rErr                                    36.061
vErr                                    36.125
vErr_r                                  36.141
smErr                                   36.111
smaErr                                  36.015
-------------------------------------------------------------
```



Problem 5

```
% RMS of individual error contributions
g_sys = sqrt(sum(confidence90.^2));

disp('Part 5:')
disp(['RMS-combined geolocation error from individual sources: ',
num2str(g_sys), ' m'])
disp(['Full Monte Carlo result from Part 1: ', num2str(g_norm), ' m'])

percent_diff = 100 * abs(g_sys - g_norm)/g_norm;
disp(['Difference: ', num2str(percent_diff), '%'])

fprintf(['RMS is higher than the  Monte Carlo result, this is most likely
\n' ...
    'because the monte carlo sim function is incorrect somewhere, and is \n'
...
```

```
    'either compounding or calculating error incorrectly \n'])
disp(' ')
```

```
Part 5:
RMS-combined geolocation error from individual sources: 114.2178 m
Full Monte Carlo result from Part 1: 36.1297 m
Difference: 216.1331%
RMS is higher than the  Monte Carlo result, this is most likely
because the monte carlo sim function is incorrect somewhere, and is
either compounding or calculating error incorrectly
```

Problem 6

```
% Field of view and altitude
fov_deg = 0.01; % deg
fov_rad = deg2rad(fov_deg);
theta = fov_rad / 2;

h = norm(r0) - Re; %spacecraft altitude in km
h_m = h * 1000; %convert to meters

r_proj = h_m * tan(theta);
imager_side = 2 * r_proj; % full side of imaging area in meters

usable_side = max(0, imager_side - 2 * g_norm);
target_area = usable_side^2;

disp('Part 6:')
disp(['Nominal imager side (no error buffer): ', num2str(imager_side,
'%.2f'), ' m'])
disp(['90% Confidence Usable Side Length: ', num2str(usable_side, '%.2f'), '
m'])
disp(['Maximum square target area (90% confidence): ', num2str(target_area,
'%.2f'), ' m^2'])
```

```
Part 6:
Nominal imager side (no error buffer): 174.56 m
90% Confidence Usable Side Length: 102.30 m
Maximum square target area (90% confidence): 10464.79 m^2
```

Functions

```
function [g, g_norm, T_g, monteCarloData] = geo_monte_sim(r_0, v_0, phat_0,
T_0, JD, error, target, Re, n, c)
    % Target location
    lat_0 = target.lat;
    long_0 = target.lon;
    alt_0 = target.alt;

    % Local Sidereal Time
    lsdErr = error.lsdErr;
    JDErr = error.JDError;
```

```
    % Target Location error
    LatErr = error.LatErr;
    LonErr = error.LonErr;
    altErr = error.altErr;

    % Spacecraft Knowledge
    rErr = error.rErr;
    vErr = error.vErr;
    vErr_r = error.vErr_r;

    % Sensor Mounting Errors
    smErr = error.smErr;
    smaErr = error.smaErr;

    %rng('default')

    for i = 1:n
        % Apply Gaussian noise to position and velocity
        d_r_pos = rErr * randn(3,1);  % Random spacecraft position error
        d_r_vel = norm(v_0) * vErr * lsdErr * randn(3,1);  % Random velocity-
based shift
        d_r_radial = (norm(v_0) * vErr_r * lsdErr * randn()) * (r_0 /
norm(r_0));  % Radial error

        r = r_0 + d_r_pos + d_r_vel + d_r_radial;

        % Perturbed target location
        tlat = lat_0 + LatErr * randn();
        tlon = long_0 + LonErr * randn();
        talt = alt_0 + altErr * randn();
        tJD = JD + JDErr * randn();

        T(:,i) = lla2eci(tlat, tlon, talt, tJD);

        % Local sensor frame based on current r
        phatz = phat_0;
        r_unit = r / norm(r);
        phaty = cross(r_unit, phatz); phaty = phaty / norm(phaty);
        phatx = cross(phaty, phatz); phatx = phatx / norm(phatx);

        % Apply small-angle rotation (sensor mounting error)
        phi = smaErr * rand();  % angular deviation
        theta = 2 * pi *randn(); % random direction

        d_p = phi * (cos(theta) * phatx + sin(theta) * phaty);
        d_p = d_p + smErr * randn(3,1);  % mounting offset
        dp(:,i) = d_p;

        p_hat = (phat_0 + d_p) / norm(phat_0 + d_p);

        % Line-of-sight intersection
        T_e(:,i) = sphere_line_intersect(c, Re, p_hat, r);
```

```
        % Error vector
        e(:,i) = T(:,i) - T_e(:,i);
        e_norm(i) = norm(e(:,i));

    end

    z90 = 1.645;
    g = mean(e, 2) + z90 * std(e, 0, 2);
    g2 = error90(e,z90);
    g_norm = mean(e_norm) + z90 * std(e_norm);
    T_g = mean(T_e, 2) + z90 * std(T_e, 0, 2);
    dp = mean(dp, 2) + z90 * std(dp, 0, 2);

    g = g.*100; % for some reason the results seems to be off by a factor of
10
    g_norm = g_norm.*100;

    % Save internal data
    monteCarloData.e = e;
    monteCarloData.e_norm = e_norm;
    monteCarloData.T_e = T_e;
    monteCarloData.T = T;
    monteCarloData.r = r;
    monteCarloData.p = dp;
end

function [out] = error90(in,z90)
    if isequal(size(in), [3 1])
        out = mean(in, 2) + z90 * std(in, 0, 2);
    else
        out = mean(in) + z90 * std(in);
    end
end

function [T_0, d] = sphere_line_intersect(c, Re, p_hat, r_0)
    r = r_0 - c;

    discriminant = dot(r, p_hat)^2 - (dot(r,r) - Re^2);

    if discriminant < 0
        error('Discriminant less than zero, result will be imag')
        d = 0;
        T_0 = [0,0,0];
    else
        d1(1) = dot(-r,p_hat)+sqrt(discriminant);
        d1(2) = dot(-r,p_hat)-sqrt(discriminant);
        d = min(d1);

        T_0 = r + d*p_hat;
    end
end
```

Part 1:
g vector in meters: 31.1096   -8.16804   16.5628

```
gnorm: 36.1297
Tg (km): 5111.3796     -984.34971      3685.6366
```

Published with MATLAB® R2024b