HW3

Tuesday, October 29, 2024 9:37 PM

Akshat Dubey, worked with Vishwa Mehta

1) **Problem 1.** Consider a map in 2D where points have coordinates (x, y), where x increases along East and y increases along North. Bearing measurements are relative to North, where positive angles represent clockwise rotations as seen from above. Let L_1 and L_2 be points on the map representing the locations of lighthouses, and let P denote your location. The coordinates of L_1 are (x_1, y_1), the coordinates of L_2 are (x_2, y_2), and your coordinates are (x_0, y_0). For $i = 1, 2$, denote the bearing of L_i relative to North by θ_i. For $i = 1, 2$, denote the ray determined by P and L_i by $y = m_i x + b_i$.

a) Derive the expressions for m_i and b_i given by

$$m_i = \frac{y_0 - y_i}{x_0 - x_i} = \tan\left(\frac{\pi}{2} - \theta_i\right), \quad (1)$$

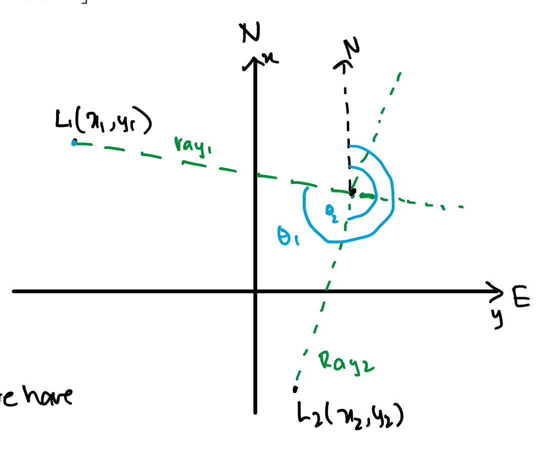
$$b_i = \frac{x_0 y_i - x_i y_0}{x_0 - x_i}. \quad (2)$$

As a check, derive

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} m_i & -1 \\ b_i - y_i & x_i \end{bmatrix}^{-1} \begin{bmatrix} m_i x_i - y_i \\ b_i x_i \end{bmatrix}. \quad (3)$$

1.a) m_i : slope of the line

$= \dfrac{y_0 - y_i}{x_0 - x_i}$ — ①

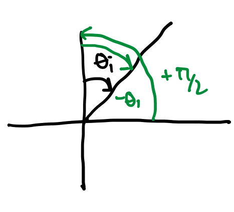


$\tan\theta = \dfrac{\text{opposite}}{\text{adjacent}}$ = slope of θ

· However, usually θ starts at 0° facing x axis and increases anticlockwise.

↳ Since here we increase the angle clockwise, we have to use $-\theta$

↳ Since we also start at an offset of 90° $\left(\frac{\pi}{2}\right)$, because 0° is facing the y axis, we need to add 90° $\left(\frac{\pi}{2}\right)$ to our values of θ



$\boxed{\therefore \text{slope} = m_i = \tan\left(\frac{\pi}{2} - \theta_i\right)}$ — ② //

Both rays pass through P at (x_0, y_0), so

$y_0 = m_i x_0 + b_i$

Substitute ①

$y_0 = \left(\dfrac{y_0 - y_i}{x_0 - x_i}\right) x_0 + b_i$

$b_i = y_0 - \left(\dfrac{y_0 - y_i}{x_0 - x_i}\right) x_0$

$= \dfrac{y_0(x_0 - x_i) - (y_0 - y_i) x_0}{x_0 - x_i}$

$= \dfrac{\cancel{x_0 y_0} - x_i y_0 - \cancel{x_0 y_0} + x_0 y_i}{x_0 - x_i}$

$= \boxed{\dfrac{x_0 y_i - x_i y_0}{x_0 - x_i}}$ — ③ //

From ①,

$m_i = \dfrac{y_0 - y_i}{x_0 - x_i}$

$\Rightarrow (x_0 - x_i) m_i = y_0 - y_i$

$\Rightarrow m_i x_0 - m_i x_i = y_0 - y_i$

$\Rightarrow m_i x_0 - y_0 = m_i x_i - y_i$ — ④

From ③

$b_i = \dfrac{x_0 y_i - x_i y_0}{x_0 - x_i}$

$\Rightarrow b_i(x_0 - x_i) = x_0 y_i - x_i y_0$

$\Rightarrow b_i x_0 - b_i x_i = x_0 y_i - x_i y_0$

$\Rightarrow b_i x_0 - y_i x_0 + x_i y_0 = b_i x_i$

$\Rightarrow (b_i - y_i) x_0 + x_i y_0 = b_i x_i$ — ⑤

Expressing ④ ⑤ in matrix form

$$\begin{bmatrix} m_i & -1 \\ b_i - y_i & x_i \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} m_i x_i - y_i \\ b_i x_i \end{bmatrix}$$

$$\Rightarrow \boxed{\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} m_i & -1 \\ b_i - y_i & x_i \end{bmatrix}^{-1} \begin{bmatrix} m_i x_i - y_i \\ b_i x_i \end{bmatrix}}$$ //

b) Derive the expression for (x_0, y_0) in terms of the lighthouse locations and bearing measurements given by

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} T_1 & -1 \\ T_2 & -1 \end{bmatrix}^{-1} \begin{bmatrix} T_1 x_1 - y_1 \\ T_2 x_2 - y_2 \end{bmatrix}, \quad (4)$$

where $T_i \triangleq \tan\left(\frac{\pi}{2} - \theta_i\right)$.

1.b) Using ④,

$m_i x_0 - y_0 = m_i x_i - y_i$

Using ②, $m_i = \tan\left(\frac{\pi}{2} - \theta_i\right) = T_i$

For $n = 1, 2$,

$m_1 x_0 - y_0 = m_1 x_1 - y_1$

and $m_2 x_0 - y_0 = m_2 x_2 - y_2$

$\Rightarrow T_1 x_0 - y_0 = T_1 x_1 - y_1$

$T_2 x_0 - y_0 = T_2 x_2 - y_2$

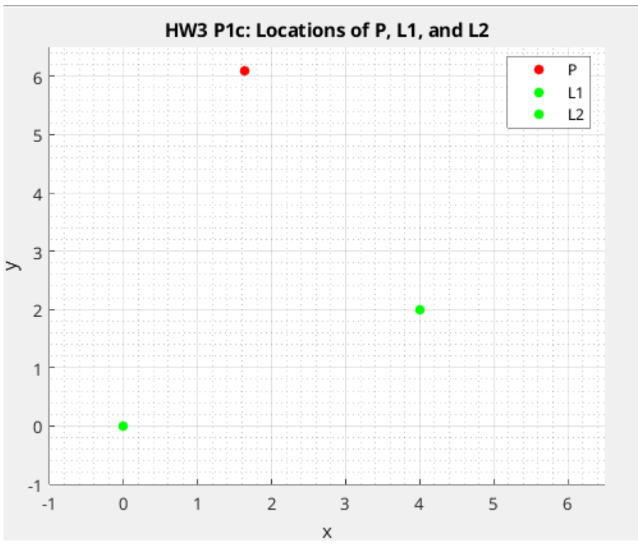
Expressing in matrix form

$$\begin{bmatrix} T_1 & -1 \\ T_2 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} T_1 x_1 - y_1 \\ T_2 x_2 - y_2 \end{bmatrix}$$

$$\Rightarrow \boxed{\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} T_1 & -1 \\ T_2 & -1 \end{bmatrix}^{-1} \begin{bmatrix} T_1 x_1 - y_1 \\ T_2 x_2 - y_2 \end{bmatrix}}$$ //

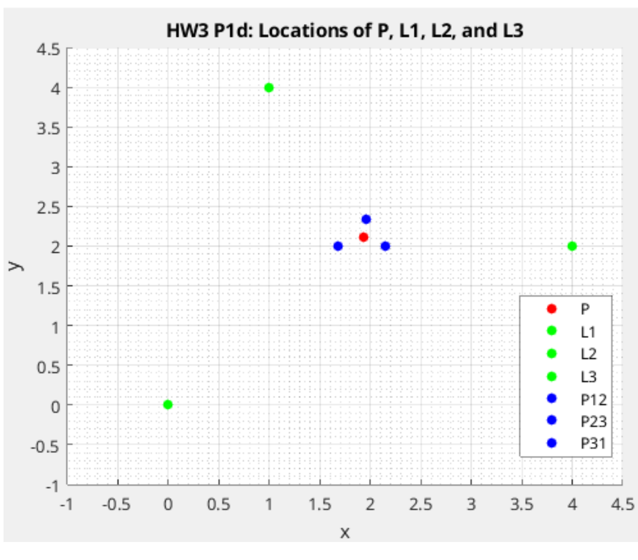
c) Let $(x_1, y_1) = (0, 0)$ m, $(x_2, y_2) = (4, 2)$ m, and let $\theta_1 = -165^o, \theta_2 = 150^o$. Write a Matlab program that obtains (x_0, y_0) using the derived equations. In a plot, place green dots on the locations of the lighthouses and place a red dot on the obtained location.

1.c)



//

1.d)



//

2) **Problem 2.** Reconsider Problem 1, but now we will use the equations 2 and 3 derived in the notes for determining the position fix (2D Position Fixing with 1 Subtended Angle and 1 Bearing section in the class 9 notes). We'll use fminunc for this problem.

a) Let $(x_1, y_1) = (0, 0)$ m, $(x_2, y_2) = (4, 2)$ m, and let $\theta_1 = -165^o, \theta_2 = 150^o$. Write a Matlab program that obtains (x_0, y_0) by solving the two equations in the lecture notes using numerical optimization (fminunc). To do this, use a grid of initial points $p_{\text{init}} \in \{-2, 0, \ldots, 8, 10\} \times \{-2, 0, \ldots, 8, 10\}$ and choose the result that yields the lowest cost function. In a plot, place green dots on the locations of the lighthouses and place a red dot on the obtained location.

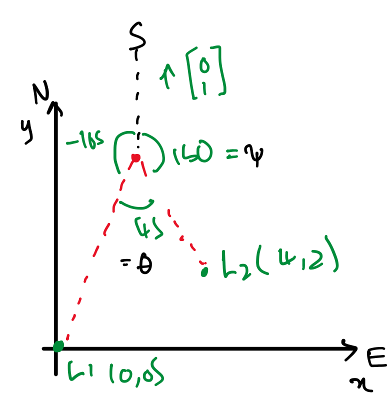
NOTE 1: The equations in the lecture notes are derived such that the bearing used corresponds to lighthouse 2. The subtended angle can be obtained from the pair of bearings. For example, in this case, the subtended angle can be chosen as 45°. Finally, remember that the star used for the bearings is in the direction of the North.

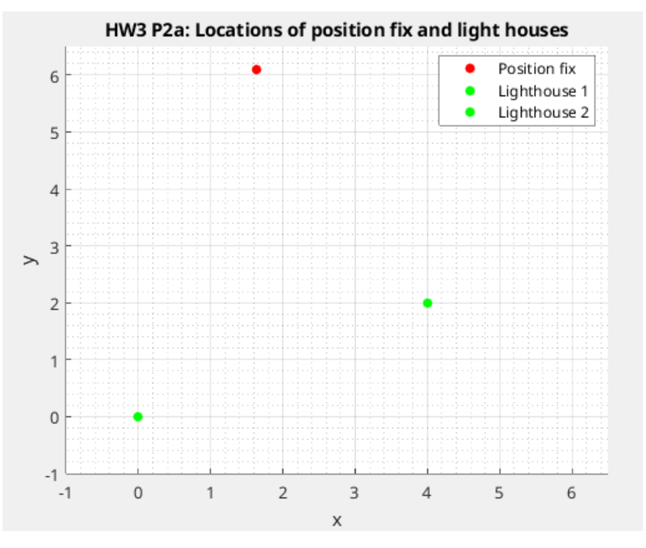
NOTE 2: Due to the structure of your optimization cost function, the positions of the lighthouses may yield a cost function value close to 0. You can assume that the position fix cannot be within $\varepsilon = 0.01$ m of both lighthouses.

2.a) Star is in the north, so unit vector to it is just $S = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

θ: subtended angle: 45°

ψ: bearing of L_2 wrt star: 150°

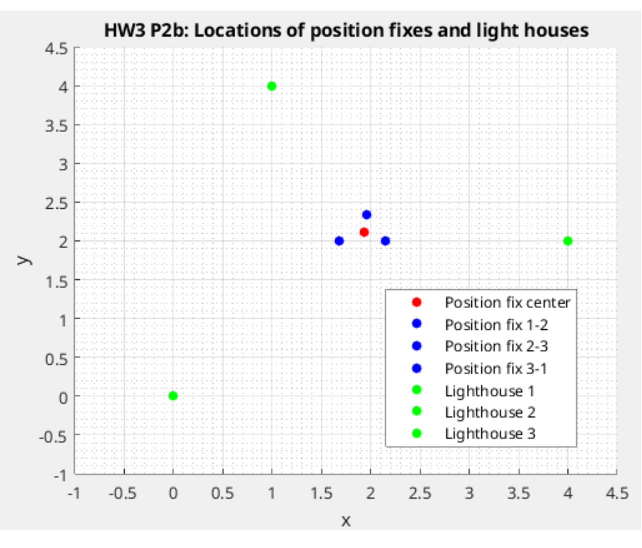




//

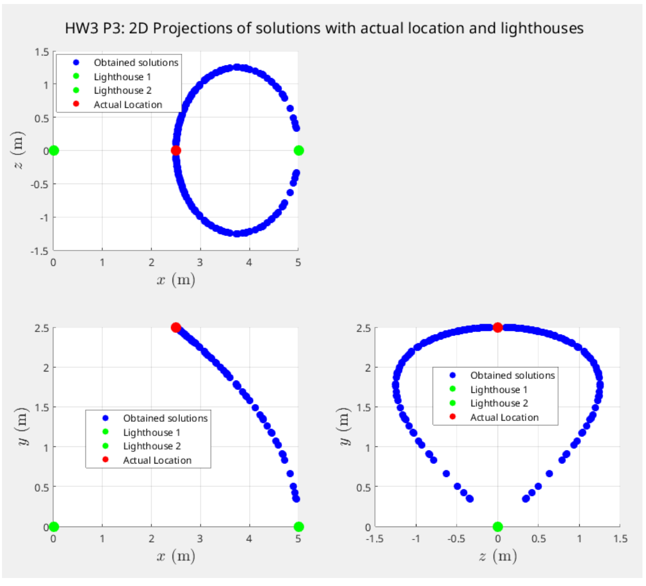
2.b) b) Suppose there is a third lighthouse L_3 located at $(x_3, y_3)m$ that yields a bearing θ_3. Let $(x_1, y_1) = (0, 0)$ m, $(x_2, y_2) = (4, 2)$ m, $(x_3, y_3) = (1, 4)$ m, and let $\theta_1 = -140^o, \theta_2 = 90^o$, $\theta_3 = -30^o$. Obtain the 3 position fixes that each pair of lighthouses yield via numerical optimization. To do this, for each position fix, use a grid of initial points $p_{\text{init}} \in \{-2, 0, \ldots, 8, 10\} \times \{-2, 0, \ldots, 8, 10\}$ and choose the result that yields the lowest cost function. In a plot, place green dots on the locations of the lighthouses, place blue dots on the locations of the obtained position fixes and place a red dot at the center of the three-cornered hat (the mean of the three vertices you obtained).

NOTE: As in a), for every pair of lighthouses, you'll have to determine the subtended angle and use the bearing of a single lighthouse. Take into account the notes of a) as well.



//

3):

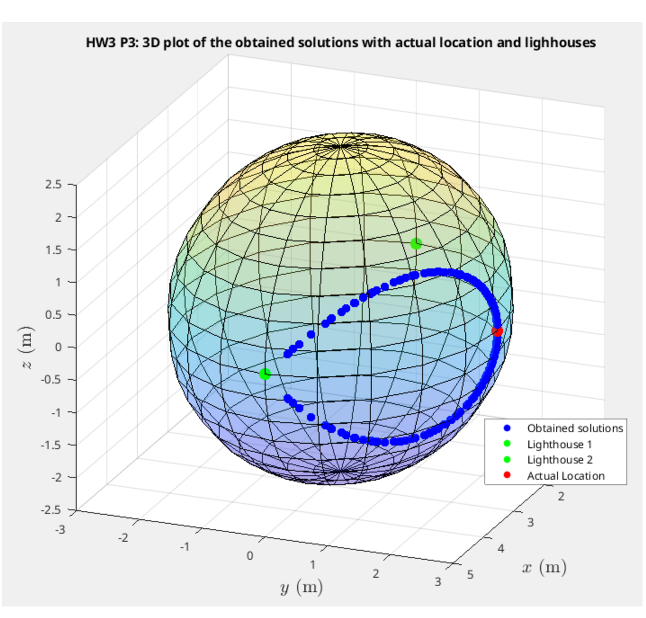


· The distribution of solutions is more tight near the actual location.

· The distribution of the solutions look like different cross section views of the intersection between a sphere and a cone

· We can see that the cone intersects the sphere at an angle, which causes the intersection to not be a circle when projected in the y-z plane

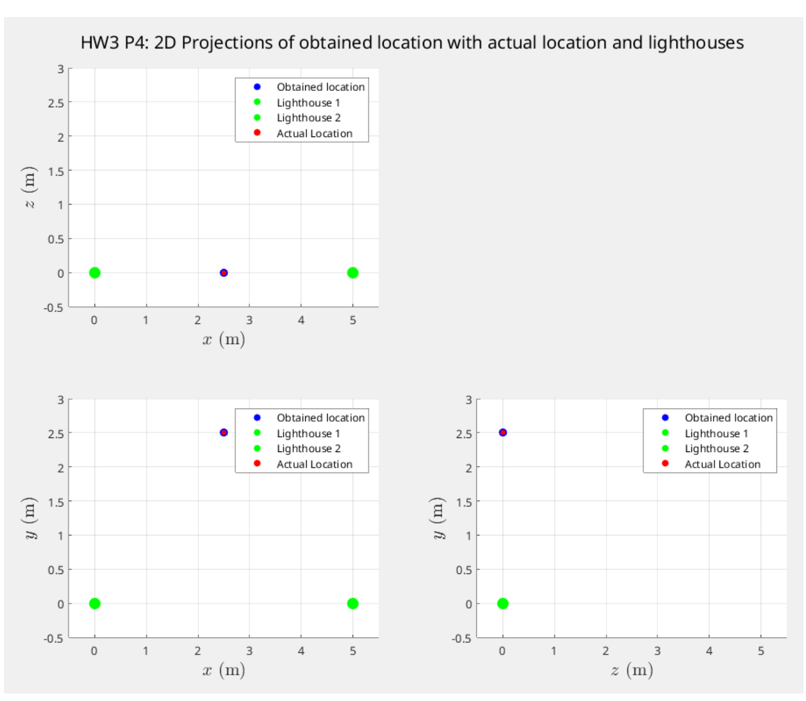
· In the x-y plane, we can also see that the obtained solutions are identical in x, y for a change in z, showing that there is no change in angle of the cone about the x and y axes



- The observation is that all points lie on the surface of the sphere that is determined by the subtended angle θ.
- All the points also lie along the surface of a cone that is determined by the bearing ψ to lighthouse L_2.
- The ψ or the bearing determines the conical fix, while the θ (subtended angle) determines a rotated circular arc fix, which is what looks like a sphere here because $\theta = 90°$.

//

4)

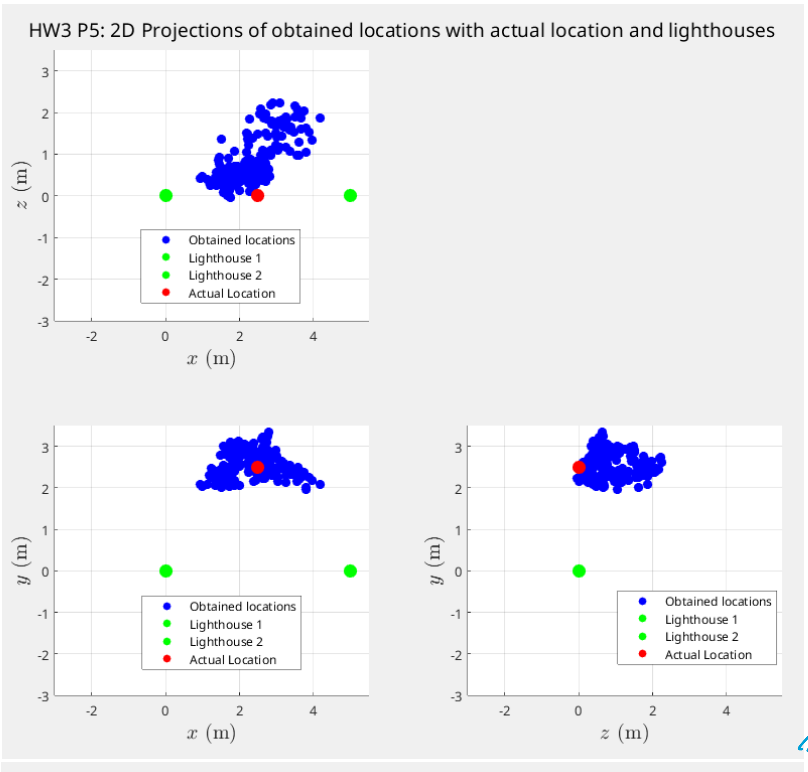


· Yes, the obtained location is the same as the actual location mentioned in P3.

· The actual location is shown as a smaller dot over the obtained location

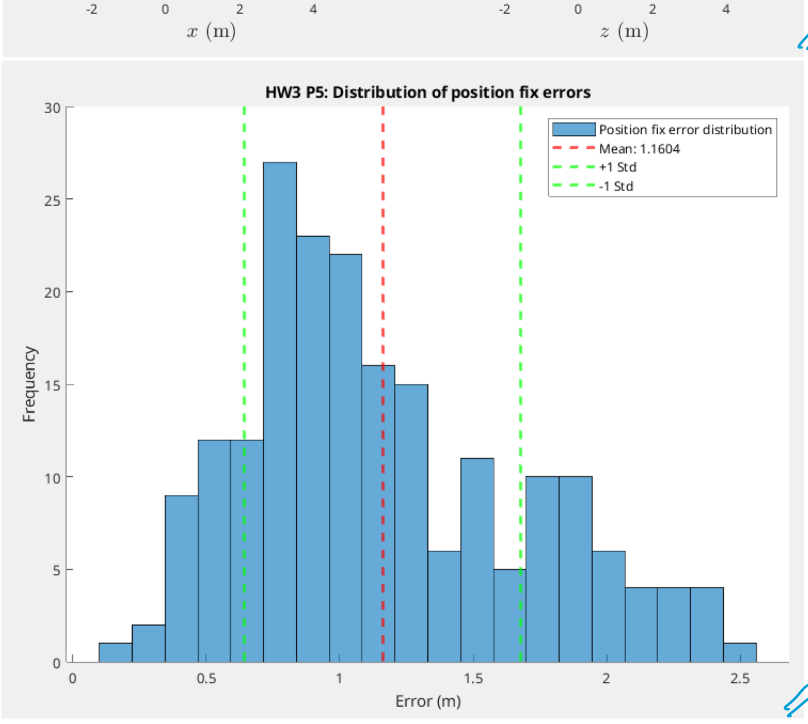
//

5)



· After adding sensor noise, the position fixes are spread across a very large region

//



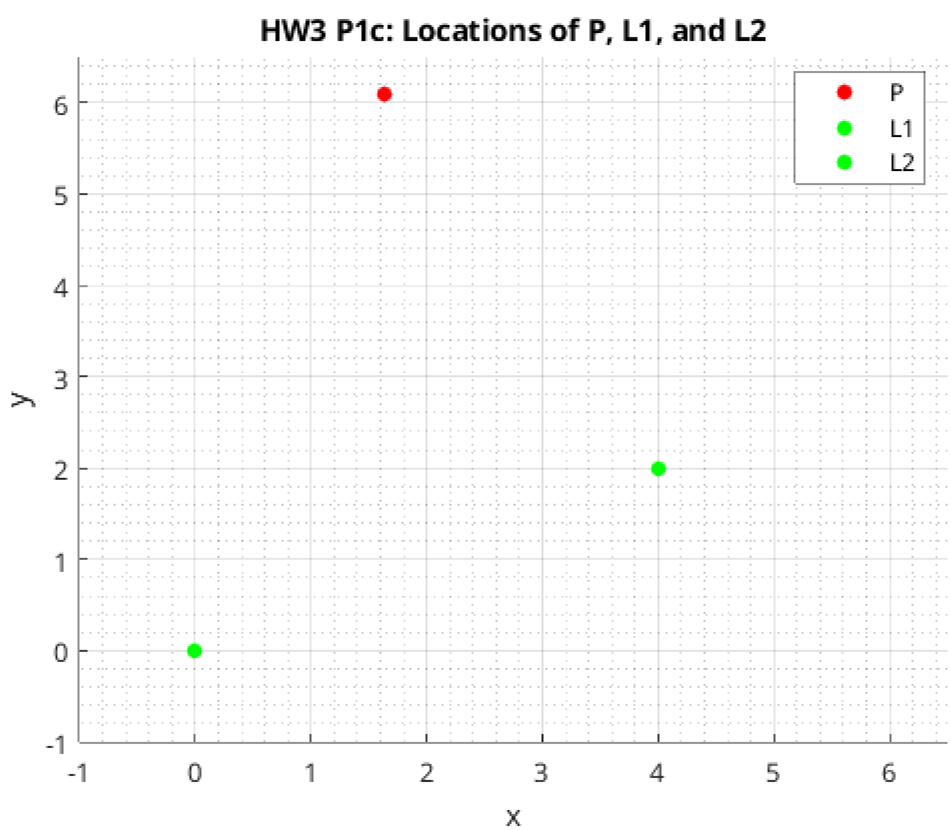
· The error mean = 1.1604

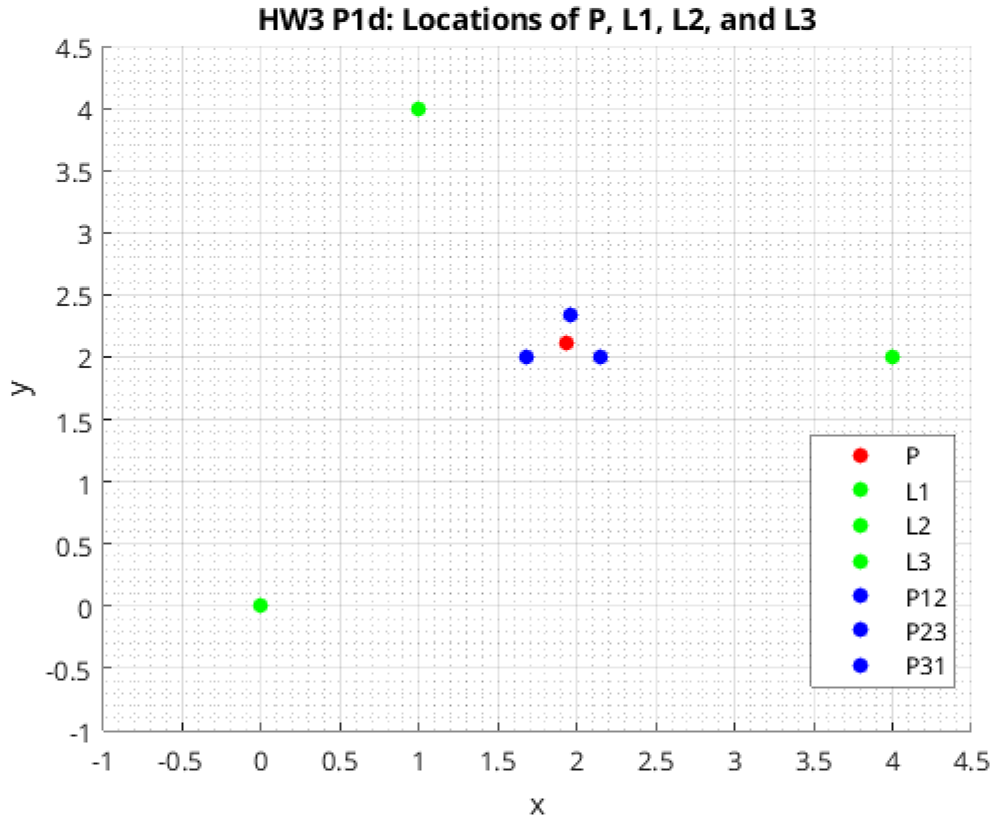
error std = 0.5168

//

```
clc; clear; close all;

% part c
L1 = [0, 0]; L2 = [4, 2];
theta1 = -165; theta2 = 150;
T1 = T(theta1); T2 = T(theta2);

A = [T1, -1;
    T2, -1];
B = [T1*L1(1) - L1(2);
    T2*L2(1) - L2(2)];

P = A\B;

% plot part c
figure;
hold on;
scatter(P(1), P(2), 35, 'r', 'filled', 'DisplayName', 'P');
scatter(L1(1), L1(2), 35, 'g', 'filled', 'DisplayName', 'L1');
scatter(L2(1), L2(2), 35, 'g', 'filled', 'DisplayName', 'L2');
xlim([-1, 6.5]); ylim([-1, 6.5]);
xlabel('x'); ylabel('y');
grid on; grid minor;
legend('Location', 'best');
title('HW3 P1c: Locations of P, L1, and L2');

% part d
L1 = [0, 0]; L2 = [4, 2]; L3 = [1, 4];
theta1 = -140; theta2 = 90; theta3 = -30;
T1 = T(theta1); T2 = T(theta2); T3 = T(theta3);

A12 = [T1, -1;
    T2, -1];
B12 = [T1*L1(1) - L1(2);
    T2*L2(1) - L2(2)];
P12 = A12\B12;

A23 = [T2, -1;
    T3, -1];
B23 = [T2*L2(1) - L2(2);
    T3*L3(1) - L3(2)];
P23 = A23\B23;

A31 = [T3, -1;
    T1, -1];
B31 = [T3*L3(1) - L3(2);
    T1*L1(1) - L1(2)];
P31 = A31\B31;

P = (P12 + P23 + P31)/3;

% plot part d
```

```
figure;
hold on;
scatter(P(1), P(2), 35, 'r', 'filled', 'DisplayName', 'P');
scatter(L1(1), L1(2), 35, 'g', 'filled', 'DisplayName', 'L1');
scatter(L2(1), L2(2), 35, 'g', 'filled', 'DisplayName', 'L2');
scatter(L3(1), L3(2), 35, 'g', 'filled', 'DisplayName', 'L3');
scatter(P12(1), P12(2), 35, 'b', 'filled', 'DisplayName', 'P12');
scatter(P23(1), P23(2), 35, 'b', 'filled', 'DisplayName', 'P23');
scatter(P31(1), P31(2), 35, 'b', 'filled', 'DisplayName', 'P31');
xlim([-1, 4.5]); ylim([-1, 4.5]);
xlabel('x'); ylabel('y');
grid on; grid minor;
legend('Location', 'best');
title('HW3 P1d: Locations of P, L1, L2, and L3');


function res = T(theta)
res = tan(pi/2 - deg2rad(theta));
end
```



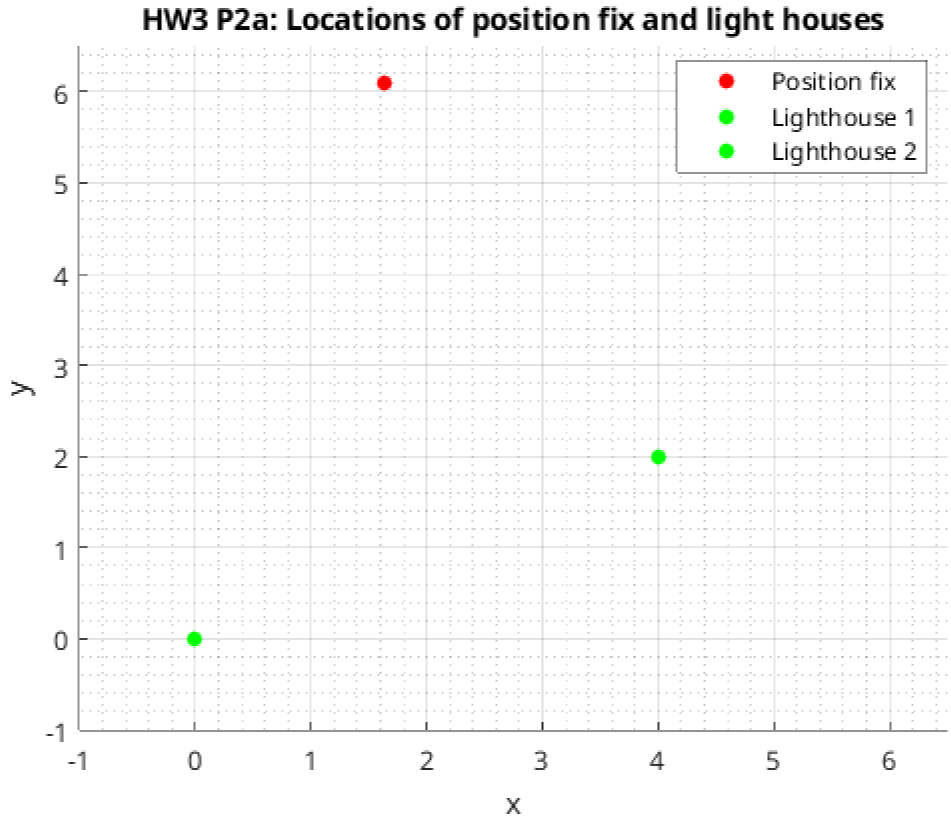


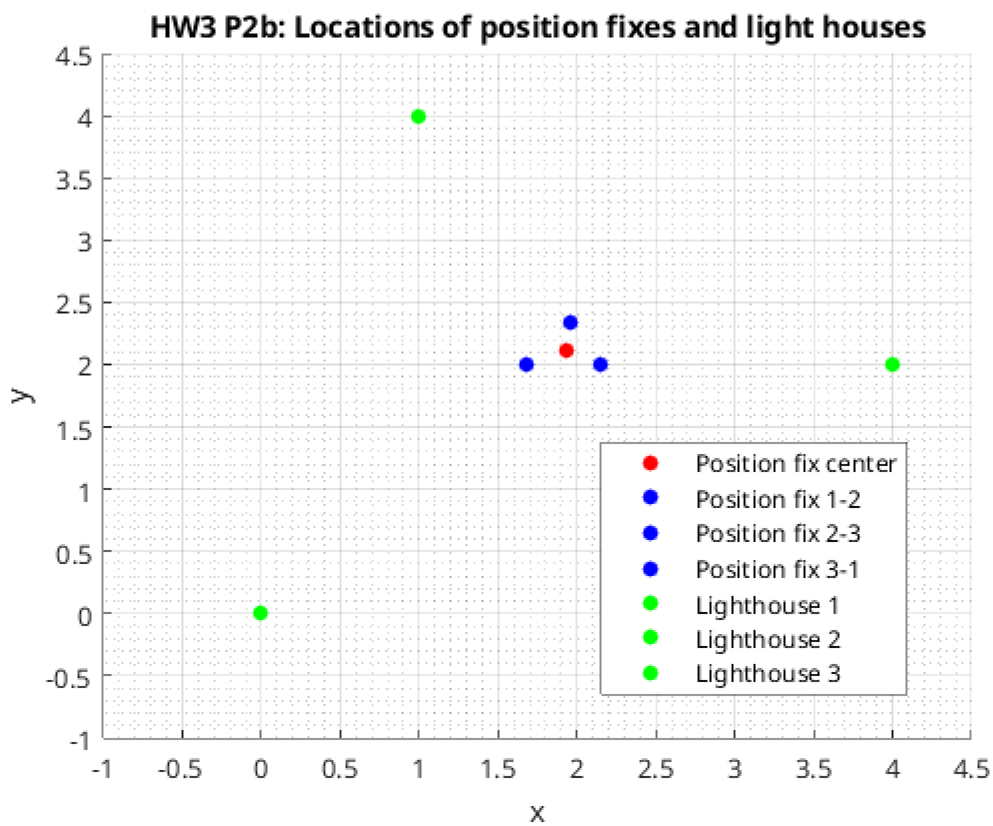
Published with MATLAB® R2023b

```
clc; clear; close all;


% subtended(0, 10)
% subtended(-10, 10)
% subtended(-10, -10)
% subtended(10, -10)
% subtended(10, 10)
% subtended(10, 370)
% subtended(370, 10)
% subtended(150, -165)
% subtended(-165, 150)
% subtended(150, 50)
% subtended(50, 150)
% subtended(170, -170)
% subtended(-170, 170)
% subtended(190, 200)
% subtended(190, 170)
% subtended(-10, -20)
% subtended(90, -30);

% part a
L1 = [0; 0]; L2 = [4; 2]; % lighthouses
theta1 = -165; theta2 = 150; % bearing to the lighthouses
theta12 = 45; % subtended angle between lighthouses
S = [0; 1]; % vector pointing of star
% psi = theta2; % bearing from star to L2

guess_range = -2:2:10;
pos_fix = get_pos_fix(S, L1, L2, theta12, theta2, guess_range);

figure;
hold on;
scatter(pos_fix(1), pos_fix(2), 35, 'r', 'filled', 'DisplayName', 'Position
fix');
scatter(L1(1), L1(2), 35, 'g', 'filled', 'DisplayName', 'Lighthouse 1');
scatter(L2(1), L2(2), 35, 'g', 'filled', 'DisplayName', 'Lighthouse 2');
xlim([-1, 6.5]); ylim([-1, 6.5]);
xlabel('x'); ylabel('y');
grid on; grid minor;
legend('Location', 'best');
title('HW3 P2a: Locations of position fix and light houses');

% part b
L1 = [0;0]; L2 = [4;2]; L3 = [1;4]; % lighthouses
theta1 = -140; theta2 = 90; theta3 = -30; % bearing to the lighthouses
theta12 = 130; theta23 = 120; theta31 = 110; % subtended angle between
lighthouses
S = [0; 1]; % vector pointing of star

guess_range = -2:2:10;
pos_fix_12 = get_pos_fix(S, L1, L2, theta12, theta2, guess_range);
```

```
pos_fix_23 = get_pos_fix(S, L2, L3, theta23, theta3, guess_range);
pos_fix_31 = get_pos_fix(S, L3, L1, theta31, theta1, guess_range);
pos_fix_final = (pos_fix_12 + pos_fix_23 + pos_fix_31)/3;

figure;
hold on;
scatter(pos_fix_final(1), pos_fix_final(2), 35, 'r', 'filled', 'DisplayName',
'Position fix center');
scatter(pos_fix_12(1), pos_fix_12(2), 35, 'b', 'filled', 'DisplayName',
'Position fix 1-2');
scatter(pos_fix_23(1), pos_fix_23(2), 35, 'b', 'filled', 'DisplayName',
'Position fix 2-3');
scatter(pos_fix_31(1), pos_fix_31(2), 35, 'b', 'filled', 'DisplayName',
'Position fix 3-1');
scatter(L1(1), L1(2), 35, 'g', 'filled', 'DisplayName', 'Lighthouse 1');
scatter(L2(1), L2(2), 35, 'g', 'filled', 'DisplayName', 'Lighthouse 2');
scatter(L3(1), L3(2), 35, 'g', 'filled', 'DisplayName', 'Lighthouse 3');
xlim([-1, 4.5]); ylim([-1, 4.5]);
xlabel('x'); ylabel('y');
grid on; grid minor;
legend('Location', 'best');
title('HW3 P2b: Locations of position fixes and light houses');



% function theta_subtended = subtended(theta1, theta2)
% % if theta1 < 0
% %     theta1 = -theta1 + 360;
% % end
% % if theta2 < 0
% %     theta2 = -theta2 + 360;
% % end
% % theta_subtended = mod(abs(theta1 - theta2), 180);
% if theta1 <= 0 && theta2 > 0
% elseif theta2 <= 0 && theta1 > 0
%     temp = abs(theta1) + abs(theta2);
% else
%     temp = theta1 - theta2;
% end
%
% theta_subtended = mod(temp, 180);
% end

function cost = costfn(x, S, L1, L2, theta, psi)
% setup
x = x'; % fix x's dims as used by fminunc
r_S_Y = S;
r_Y_L1 = x-L1;
r_L2_L1 = L2-L1;
r_L2_Y = L2-x;
theta = deg2rad(theta);
psi = deg2rad(psi);

% eq2
```

```
l2 = dot(r_Y_L1, r_L2_L1);
r2 = - norm(r_Y_L1) * norm(r_L2_Y) * cos(theta) + norm(r_Y_L1)^2;
err_subt_angle = l2-r2;

% eq3
l3 = dot(r_L2_Y, r_S_Y);
r3 = norm(r_L2_Y) * cos(psi);
err_bearing = l3-r3;

cost = err_subt_angle^2 + err_bearing^2;
end

function pos = get_pos_fix(S, L1, L2, theta, psi, guess_range)
p_inits = zeros(length(guess_range)^2, 2);
for i = 1:length(guess_range)
    for j = 1:length(guess_range)
        p_inits((i-1)*length(guess_range)+j, :) = [guess_range(i),
guess_range(j)];
    end
end

costs = zeros(size(p_inits, 1), 1);
pos_fixes = zeros(size(p_inits));
options = optimoptions('fminunc', 'OptimalityTolerance', 1e-12, 'Display',
'off');
for i = 1:size(p_inits, 1)
    [x, fval] = fminunc(@(x) costfn(x, S, L1, L2, theta, psi), p_inits(i, :),
options);
    costs(i) = fval;
    pos_fixes(i, :) = x;
end

% pick point with the lowest cost that isnt closest to a lighthouse
epsilon = 0.01;
pos_fix = [100, 100]; % large number
pos_fix_cost = 100; % large number
for i = 1:size(pos_fixes, 1)
    if costs(i) < pos_fix_cost
        dist_L1 = norm(pos_fixes(i, :) - L1');
        dist_L2 = norm(pos_fixes(i, :) - L2');
        if dist_L1 > epsilon && dist_L2 > epsilon
            pos_fix = pos_fixes(i, :);
            pos_fix_cost = costs(i);
        end
    end
end
pos = pos_fix;
end
```


HW3 P2a: Locations of position fix and light houses
Position fix
Lighthouse 1
Lighthouse 2
x
y

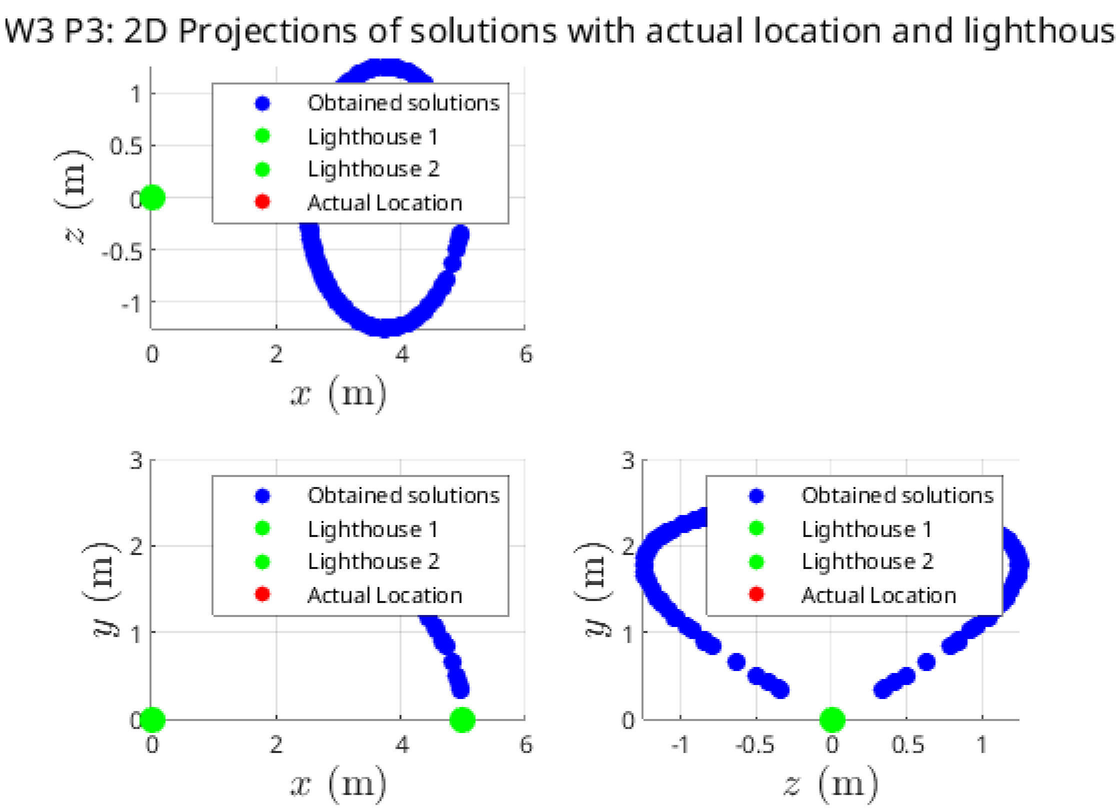

HW3 P2b: Locations of position fixes and light houses
Position fix center
Position fix 1-2
Position fix 2-3
Position fix 3-1
Lighthouse 1
Lighthouse 2
Lighthouse 3
x
y

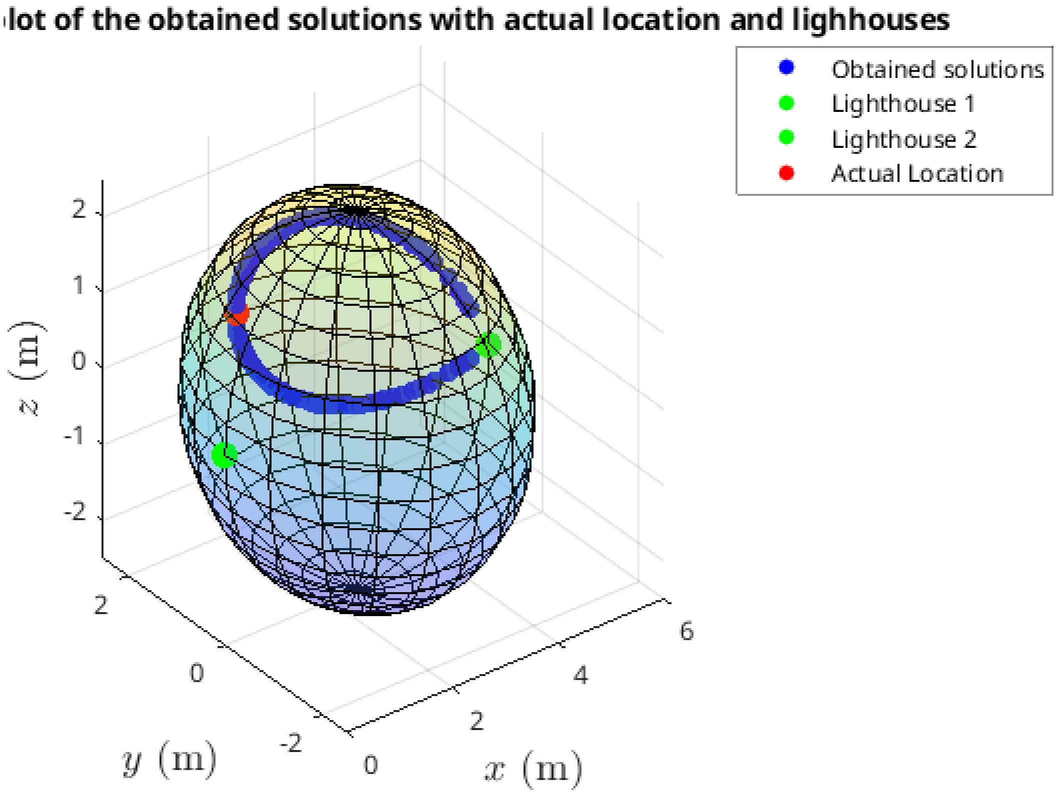
Published with MATLAB® R2023b

```
clc; clear; close all;

% part a

L1 = [0;0;0]; L2 = [5;0;0]; % lighthouses
S = [0; 1; 0]; % vector pointing of star
pos_actual = [2.5;2.5;0];
pos_fixes = get_pos_fix(S, L1, L2, 90, 135, -5:2:5);
plot_2d_proj(pos_actual, pos_fixes, L1, L2);
plot_sphere(pos_actual, pos_fixes, L1, L2);

function cost = costfn(x, S, L1, L2, theta, psi)
% setup
x = x'; % fix x's dims as used by fminunc
r_S_Y = S;
r_Y_L1 = x-L1;
r_L2_L1 = L2-L1;
r_L2_Y = L2-x;
theta = deg2rad(theta);
psi = deg2rad(psi);

% eq4
l2 = dot(r_Y_L1, r_L2_L1);
r2 = - norm(r_Y_L1) * norm(r_L2_Y) * cos(theta) + norm(r_Y_L1)^2;
err_subt_angle = l2-r2;

% eq5
l3 = dot(r_L2_Y, r_S_Y);
r3 = norm(r_L2_Y) * cos(psi);
err_bearing = l3-r3;

cost = err_subt_angle^2 + err_bearing^2;
end


function pos_fixes = get_pos_fix(S, L1, L2, theta, psi, guess_range)
p_inits = zeros(length(guess_range)^3, 3);
for i = 1:length(guess_range)
    for j = 1:length(guess_range)
        for k = 1:length(guess_range)
            p_inits((i-1)*length(guess_range)^2 + (j-1)*length(guess_range) +
k, :) = [guess_range(i), guess_range(j), guess_range(k)];
        end
    end
end

costs = zeros(size(p_inits, 1), 1);
pos_fixes = zeros(size(p_inits));
options = optimoptions('fminunc', 'OptimalityTolerance', 1e-12, 'Display',
'off');
for i = 1:size(p_inits, 1)
    [x, fval] = fminunc(@(x) costfn(x, S, L1, L2, theta, psi), p_inits(i, :),
```

```
options);
    costs(i) = fval;
    pos_fixes(i, :) = x;
end
end

function plot_2d_proj(pos_actual, pos_fixes, L1, L2)
x1 = L1(1); y1 = L1(2); z1 = L1(3);
x2 = L2(1); y2 = L2(2); z2 = L2(3);

figure;
sgtitle('HW3 P3: 2D Projections of solutions with actual location and
lighthouses')

subplot(2,2,1)
scatter(pos_fixes(:,1), pos_fixes(:,3), 50, 'b', 'filled')
hold on
scatter(x1, z1, 100, 'g', 'filled')
scatter(x2, z2, 100, 'g', 'filled')
scatter(pos_actual(1), pos_actual(3), 100, 'r', 'filled')
hold off
grid on
ylabel('$z$ (m)','interpreter','latex','fontsize',15)
xlabel('$x$ (m)','interpreter','latex','fontsize',15)
legend('Obtained solutions', 'Lighthouse 1', 'Lighthouse 2', 'Actual
Location')

subplot(2,2,3)
scatter(pos_fixes(:,1), pos_fixes(:,2), 50, 'b', 'filled')
hold on
scatter(x1, y1, 100, 'g', 'filled')
scatter(x2, y2, 100, 'g', 'filled')
scatter(pos_actual(1), pos_actual(2), 100, 'r', 'filled')
hold off
grid on
ylabel('$y$ (m)','interpreter','latex','fontsize',15)
xlabel('$x$ (m)','interpreter','latex','fontsize',15)
legend('Obtained solutions', 'Lighthouse 1', 'Lighthouse 2', 'Actual
Location')

subplot(2,2,4)
scatter(pos_fixes(:,3), pos_fixes(:,2), 50, 'b', 'filled')
hold on
scatter(z1, y1, 100, 'g', 'filled')
scatter(z2, y2, 100, 'g', 'filled')
scatter(pos_actual(3), pos_actual(2), 100, 'r', 'filled')
hold off
grid on
ylabel('$y$ (m)','interpreter','latex','fontsize',15)
xlabel('$z$ (m)','interpreter','latex','fontsize',15)
legend('Obtained solutions', 'Lighthouse 1', 'Lighthouse 2', 'Actual
Location')
end
```

```
function plot_sphere(pos_actual, pos_fixes, L1, L2)
x1 = L1(1); y1 = L1(2); z1 = L1(3);
x2 = L2(1); y2 = L2(2); z2 = L2(3);

figure;
% 3D plot
scatter3(pos_fixes(:,1), pos_fixes(:,2), pos_fixes(:,3), 50, 'b', 'filled')
hold on
scatter3(x1, y1, z1, 100, 'g', 'filled')
scatter3(x2, y2, z2, 100, 'g', 'filled')
scatter3(pos_actual(1), pos_actual(2), pos_actual(3), 100, 'r', 'filled')

% Plot the sphere
[Xsp, Ysp, Zsp] = sphere;
surf(2.5.*Xsp + 2.5, 2.5.*Ysp, 2.5.*Zsp, 'FaceAlpha', 0.25)

hold off
zlabel('$z$ (m)','interpreter','latex','fontsize',15)
ylabel('$y$ (m)','interpreter','latex','fontsize',15)
xlabel('$x$ (m)','interpreter','latex','fontsize',15)
legend('Obtained solutions', 'Lighthouse 1', 'Lighthouse 2', 'Actual
Location')
title('HW3 P3: 3D plot of the obtained solutions with actual location and
lighhouses')
end
```





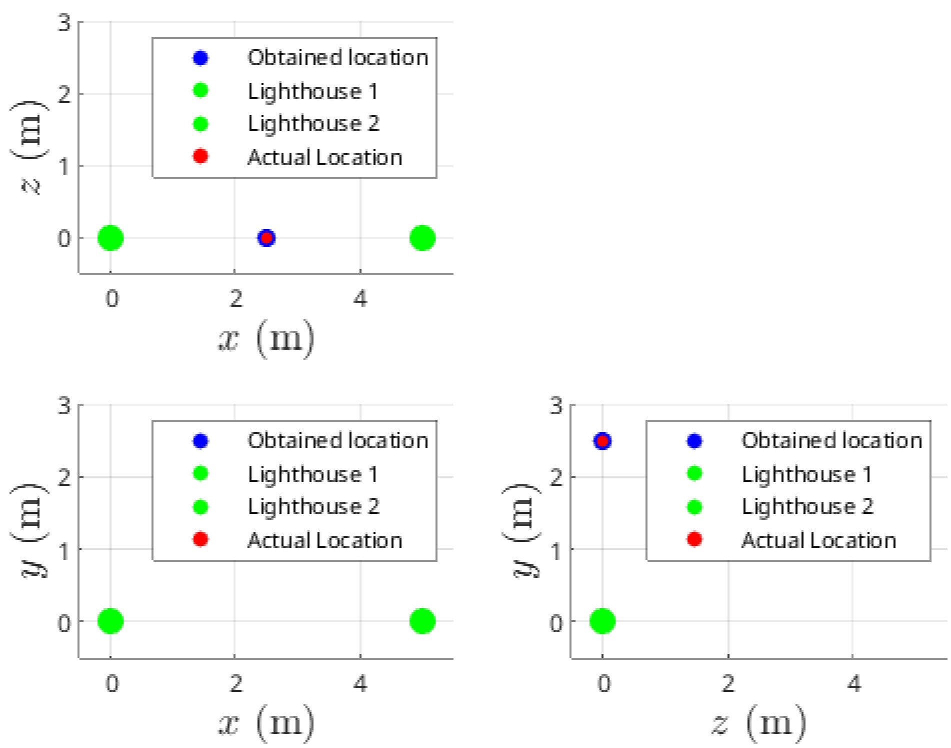
Published with MATLAB® R2023b

```
clc; clear; close all;

L1 = [0;0;0]; L2 = [5;0;0]; % lighthouses
S_r = [0; 1; 0]; S_v = [0; 0; 1]; % vector pointing of star
psi_r = 135; psi_v = 90; % relative to lighthouse 2
theta = 90; % subtended angle between lighthouses
pos_actual = [2.5;2.5;0];

pos_fix = get_pos_fix(S_r, S_v, L1, L2, theta, psi_r, psi_v, -5:2:5);
plot_2d_proj(pos_actual, pos_fix, L1, L2);

function cost = costfn(x, S1, S2, L1, L2, theta, psi1, psi2)
% setup
x = x'; % fix x's dims as used by fminunc
r_S1_Y = S1;
r_S2_Y = S2;
r_Y_L1 = x-L1;
r_L2_L1 = L2-L1;
r_L2_Y = L2-x;
theta = deg2rad(theta);
psi1 = deg2rad(psi1);
psi2 = deg2rad(psi2);

% eq4
l4 = dot(r_Y_L1, r_L2_L1);
r4 = - norm(r_Y_L1) * norm(r_L2_Y) * cos(theta) + norm(r_Y_L1)^2;
err_subt_angle = l4-r4;

% eq5
l5 = dot(r_L2_Y, r_S1_Y);
r5 = norm(r_L2_Y) * cos(psi1);
err_bearing1 = l5-r5;

% eq6
l6 = dot(r_L2_Y, r_S2_Y);
r6 = norm(r_L2_Y) * cos(psi2);
err_bearing2 = l6-r6;

cost = err_subt_angle^2 + err_bearing1^2 + err_bearing2^2;
end


function pos = get_pos_fix(S1, S2, L1, L2, theta, psi1, psi2, guess_range)
p_inits = zeros(length(guess_range)^3, 3);
for i = 1:length(guess_range)
    for j = 1:length(guess_range)
        for k = 1:length(guess_range)
            p_inits((i-1)*length(guess_range)^2 + (j-1)*length(guess_range) +
k, :) = [guess_range(i), guess_range(j), guess_range(k)];
        end
    end
end
```

```
costs = zeros(size(p_inits, 1), 1);
pos_fixes = zeros(size(p_inits));
options = optimoptions('fminunc', 'OptimalityTolerance', 1e-12, 'Display',
'off');
for i = 1:size(p_inits, 1)
    [x, fval] = fminunc(@(x) costfn(x, S1, S2, L1, L2, theta, psi1, psi2),
p_inits(i, :), options);
    costs(i) = fval;
    pos_fixes(i, :) = x;
end

% pick point with the lowest cost that isnt closest to a lighthouse
epsilon = 0.01;
pos_fix = [100, 100]; % large number
pos_fix_cost = 100; % large number
for i = 1:size(pos_fixes, 1)
    if costs(i) < pos_fix_cost
        dist_L1 = norm(pos_fixes(i, :) - L1');
        dist_L2 = norm(pos_fixes(i, :) - L2');
        if dist_L1 > epsilon && dist_L2 > epsilon
            pos_fix = pos_fixes(i, :);
            pos_fix_cost = costs(i);
        end
    end
end
pos = pos_fix;

end

function plot_2d_proj(pos_actual, pos_fix, L1, L2)
x1 = L1(1); y1 = L1(2); z1 = L1(3);
x2 = L2(1); y2 = L2(2); z2 = L2(3);

figure;
sgtitle('HW3 P4: 2D Projections of obtained location with actual location and
lighthouses')

subplot(2,2,1)
scatter(pos_fix(1), pos_fix(3), 50, 'b', 'filled')
hold on
scatter(x1, z1, 100, 'g', 'filled')
scatter(x2, z2, 100, 'g', 'filled')
scatter(pos_actual(1), pos_actual(3), 20, 'r', 'filled')
hold off
grid on
xlim([-0.5 5.5])
ylim([-0.5 3])
ylabel('$z$ (m)','interpreter','latex','fontsize',15)
xlabel('$x$ (m)','interpreter','latex','fontsize',15)
legend('Obtained location', 'Lighthouse 1', 'Lighthouse 2', 'Actual Location')

subplot(2,2,3)
scatter(pos_fix(1), pos_fix(2), 50, 'b', 'filled')
```

```
hold on
scatter(x1, y1, 100, 'g', 'filled')
scatter(x2, y2, 100, 'g', 'filled')
scatter(pos_actual(1), pos_actual(2), 20, 'r', 'filled')
hold off
grid on
xlim([-0.5 5.5])
ylim([-0.5 3])
ylabel('$y$ (m)','interpreter','latex','fontsize',15)
xlabel('$x$ (m)','interpreter','latex','fontsize',15)
legend('Obtained location', 'Lighthouse 1', 'Lighthouse 2', 'Actual Location')

subplot(2,2,4)
scatter(pos_fix(3), pos_fix(2), 50, 'b', 'filled')
hold on
scatter(z1, y1, 100, 'g', 'filled')
scatter(z2, y2, 100, 'g', 'filled')
scatter(pos_actual(3), pos_actual(2), 20, 'r', 'filled')
hold off
grid on
xlim([-0.5 5.5])
ylim([-0.5 3])
ylabel('$y$ (m)','interpreter','latex','fontsize',15)
xlabel('$z$ (m)','interpreter','latex','fontsize',15)
legend('Obtained location', 'Lighthouse 1', 'Lighthouse 2', 'Actual Location')
end
```

4: 2D Projections of obtained location with actual location and lighth



Published with MATLAB® R2023b

```
clc; clear; close all;

L1 = [0;0;0]; L2 = [5;0;0]; % lighthouses
S_r = [0; 1; 0]; S_v = [0; 0; 1]; % vector pointing of star
psi_r = 135; psi_v = 90; % relative to lighthouse 2
theta = 90; % subtended angle between lighthouses
pos_actual = [2.5;2.5;0];
rv_samples = 200;
rv_mean = 0; rv_std = 2;

pos_fixes = zeros(rv_samples, 3);

for i = 1:rv_samples
    theta = theta + rv_std*randn + rv_mean;
    psi_r = psi_r + rv_std*randn + rv_mean;
    psi_v = psi_v + rv_std*randn + rv_mean;
    pos_fixes(i, :) = get_pos_fix(S_r, S_v, L1, L2, theta, psi_r, psi_v,
-1:3:5);
end
```

plot

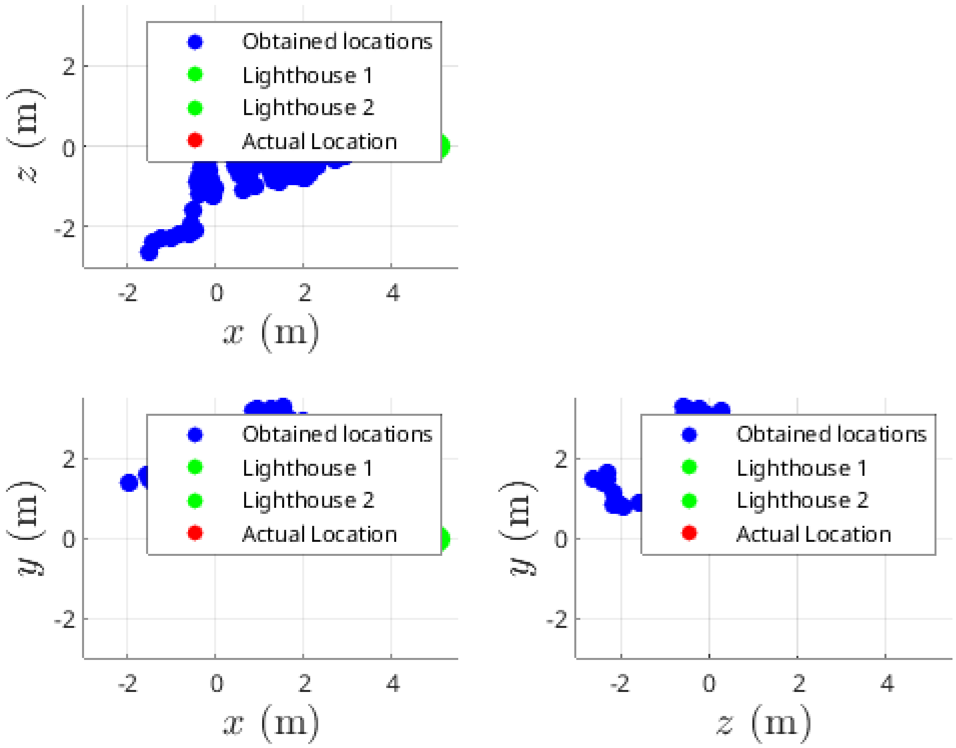
```
plot_2d_proj(pos_actual, pos_fixes, L1, L2);
plot_pos_fix_err(pos_actual, pos_fixes);

function cost = costfn(x, S1, S2, L1, L2, theta, psi1, psi2)
% setup
x = x'; % fix x's dims as used by fminunc
r_S1_Y = S1;
r_S2_Y = S2;
r_Y_L1 = x-L1;
r_L2_L1 = L2-L1;
r_L2_Y = L2-x;
theta = deg2rad(theta);
psi1 = deg2rad(psi1);
psi2 = deg2rad(psi2);

% eq4
l4 = dot(r_Y_L1, r_L2_L1);
r4 = - norm(r_Y_L1) * norm(r_L2_Y) * cos(theta) + norm(r_Y_L1)^2;
err_subt_angle = l4-r4;

% eq5
l5 = dot(r_L2_Y, r_S1_Y);
r5 = norm(r_L2_Y) * cos(psi1);
err_bearing1 = l5-r5;

% eq6
l6 = dot(r_L2_Y, r_S2_Y);
r6 = norm(r_L2_Y) * cos(psi2);
err_bearing2 = l6-r6;
```

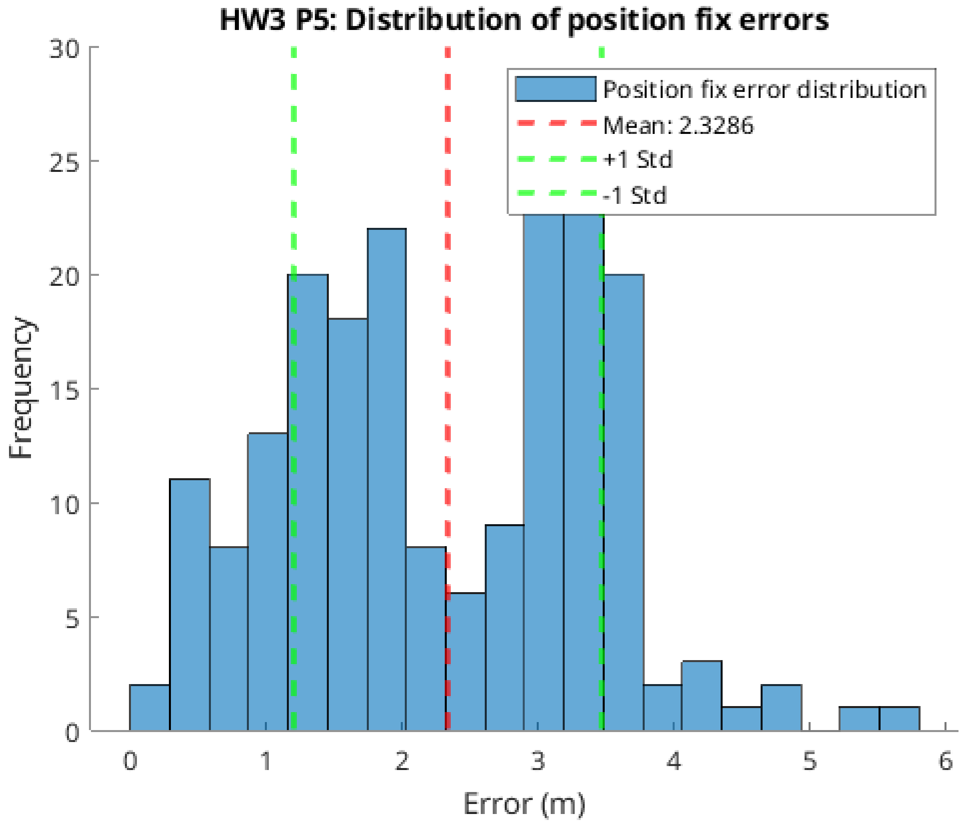
```
cost = err_subt_angle^2 + err_bearing1^2 + err_bearing2^2;
end


function pos = get_pos_fix(S1, S2, L1, L2, theta, psi1, psi2, guess_range)
p_inits = zeros(length(guess_range)^3, 3);
for i = 1:length(guess_range)
    for j = 1:length(guess_range)
        for k = 1:length(guess_range)
            p_inits((i-1)*length(guess_range)^2 + (j-1)*length(guess_range) +
k, :) = [guess_range(i), guess_range(j), guess_range(k)];
        end
    end
end

costs = zeros(size(p_inits, 1), 1);
pos_fixes = zeros(size(p_inits));
options = optimoptions('fminunc', 'OptimalityTolerance', 1e-12, 'Display',
'off');
for i = 1:size(p_inits, 1)
    [x, fval] = fminunc(@(x) costfn(x, S1, S2, L1, L2, theta, psi1, psi2),
p_inits(i, :), options);
    costs(i) = fval;
    pos_fixes(i, :) = x;
end

% pick point with the lowest cost that isnt closest to a lighthouse
epsilon = 0.01;
pos_fix = [100, 100]; % large number
pos_fix_cost = 100; % large number
for i = 1:size(pos_fixes, 1)
    if costs(i) < pos_fix_cost
        dist_L1 = norm(pos_fixes(i, :) - L1');
        dist_L2 = norm(pos_fixes(i, :) - L2');
        if dist_L1 > epsilon && dist_L2 > epsilon
            pos_fix = pos_fixes(i, :);
            pos_fix_cost = costs(i);
        end
    end
end
pos = pos_fix;

end

function plot_2d_proj(pos_actual, pos_fixes, L1, L2)
x1 = L1(1); y1 = L1(2); z1 = L1(3);
x2 = L2(1); y2 = L2(2); z2 = L2(3);

figure;
sgtitle('HW3 P5: 2D Projections of obtained locations with actual location
and lighthouses')

subplot(2,2,1)
scatter(pos_fixes(:,1), pos_fixes(:,3), 50, 'b', 'filled')
```

```
hold on
scatter(x1, z1, 100, 'g', 'filled')
scatter(x2, z2, 100, 'g', 'filled')
scatter(pos_actual(1), pos_actual(3), 100, 'r', 'filled')
hold off
grid on
xlim([-3 5.5])
ylim([-3 3.5])
ylabel('$z$ (m)','interpreter','latex','fontsize',15)
xlabel('$x$ (m)','interpreter','latex','fontsize',15)
legend('Obtained locations', 'Lighthouse 1', 'Lighthouse 2', 'Actual
Location')

subplot(2,2,3)
scatter(pos_fixes(:,1), pos_fixes(:,2), 50, 'b', 'filled')
hold on
scatter(x1, y1, 100, 'g', 'filled')
scatter(x2, y2, 100, 'g', 'filled')
scatter(pos_actual(1), pos_actual(2), 100, 'r', 'filled')
hold off
grid on
xlim([-3 5.5])
ylim([-3 3.5])
ylabel('$y$ (m)','interpreter','latex','fontsize',15)
xlabel('$x$ (m)','interpreter','latex','fontsize',15)
legend('Obtained locations', 'Lighthouse 1', 'Lighthouse 2', 'Actual
Location')

subplot(2,2,4)
scatter(pos_fixes(:,3), pos_fixes(:,2), 50, 'b', 'filled')
hold on
scatter(z1, y1, 100, 'g', 'filled')
scatter(z2, y2, 100, 'g', 'filled')
scatter(pos_actual(3), pos_actual(2), 100, 'r', 'filled')
hold off
grid on
xlim([-3 5.5])
ylim([-3 3.5])
ylabel('$y$ (m)','interpreter','latex','fontsize',15)
xlabel('$z$ (m)','interpreter','latex','fontsize',15)
legend('Obtained locations', 'Lighthouse 1', 'Lighthouse 2', 'Actual
Location')
end

function plot_pos_fix_err(pos_actual, pos_fixes)
err = vecnorm(pos_fixes - pos_actual', 2, 2); % 2 norm along rows
err_mean = mean(err);
err_std = std(err);
figure;
hold on;
histogram(err, 20, 'DisplayName', "Position fix error distribution");
xline(err_mean, '--r', 'LineWidth', 2, 'DisplayName', "Mean: " + err_mean);
xline(err_mean + err_std, '--g', 'LineWidth', 2, 'DisplayName', "+1 Std");
xline(err_mean - err_std, '--g', 'LineWidth', 2, 'DisplayName', "-1 Std");
```

```
hold off;
xlabel('Error (m)');
ylabel('Frequency');
legend show;
title('HW3 P5: Distribution of position fix errors');
end
```

5: 2D Projections of obtained locations with actual location and lightt





Published with MATLAB® R2023b