Table of Contents

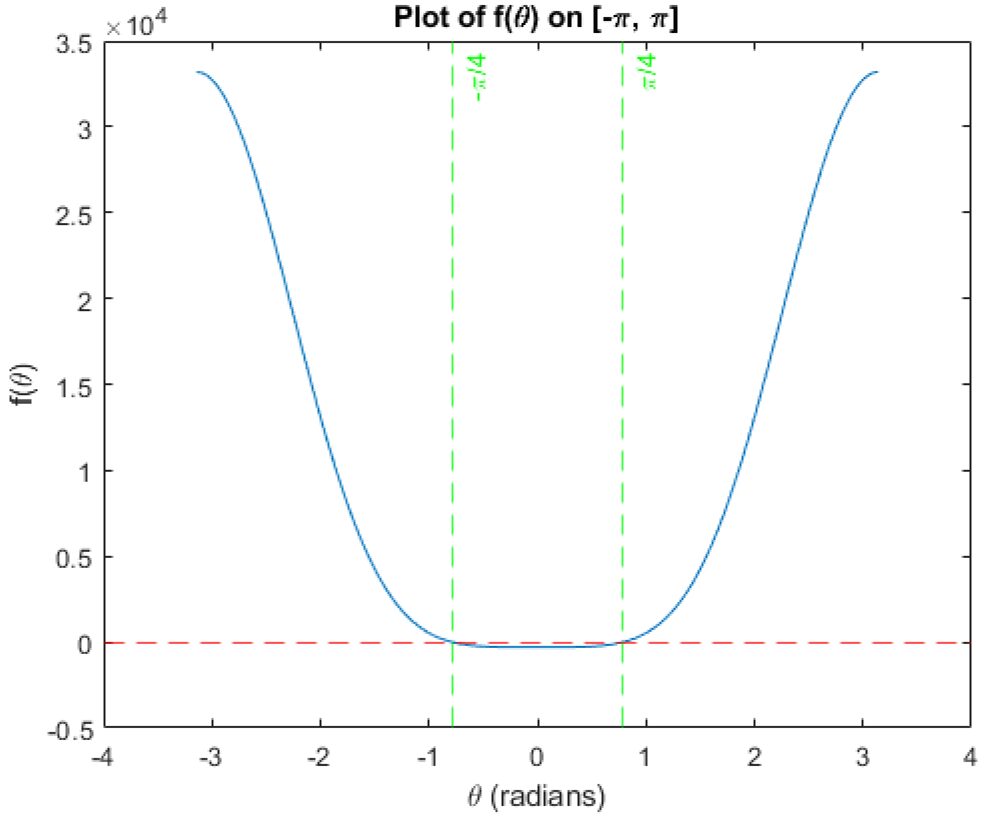

```
clear;
```

QUESTION 1:

```
% Function is at the bottom in the supporting code section
% function out = f(theta)

% Testing theta = pi/4
val1 = f(pi/4);

% Testing theta = -pi/4
val2 = f(-pi/4);

fprintf('f(pi/4) = %.10f\n', val1);
fprintf('f(-pi/4) = %.10f\n', val2);

% Both are close to 0, so we are good
```

QUESTION 2:

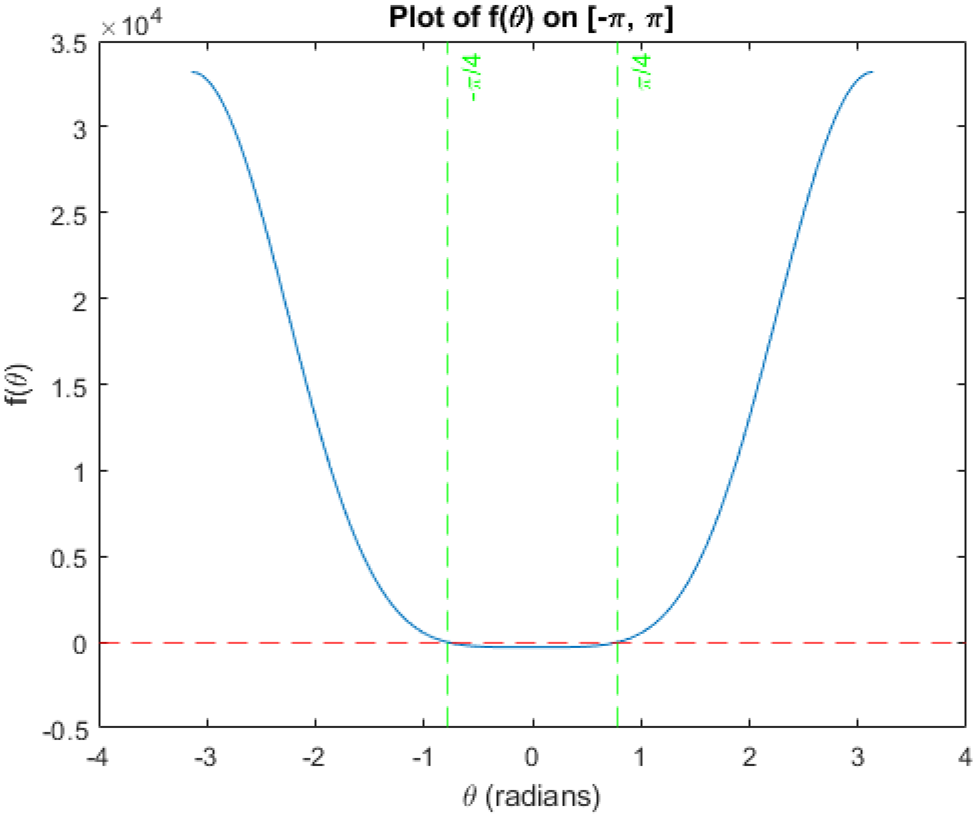
```
% Plotting f(theta) on [-pi, pi]
theta_vals = -pi:0.01:pi;

f_vals = f(theta_vals);

figure(1)
plot(theta_vals, f_vals)
xlabel('\theta (radians)')
ylabel('f(\theta)')
title('Plot of f(\theta) on [-\pi, \pi]')
yline(0, '--r');
xline(pi/4, '--g', '\pi/4');
xline(-pi/4, '--g', '-\pi/4');
drawnow;

% Plot clearly shows that there are roots at +/- pi/4
```



QUESTION 3:

```
% Pose from Figure 1.15 (a)

% Connected to (0, 0) aka (x, y)
u1 = 1; v1 = 2;

% Connected to (x1, 0)
u2 = 2; v2 = 1;

% Connected to (x2, y2)
u3 = 2; v3 = 3;

x1 = 4; x2 = 0; y2 = 4;

figure(2)
plot([u1 u2 u3 u1], [v1 v2 v3 v1], 'r'); hold on          % Platform triangle
plot([0 x1 x2], [0 0 y2], 'bo')                           % Base anchors
plot([u1 u2 u3], [v1 v2 v3], 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r')
% Platform joints
plot([u1 0], [v1 0], 'k--')    % p1
plot([u2 x1], [v2 0], 'k--')   % p2
plot([u3 x2], [v3 y2], 'k--')  % p3
title('Figure 1.15 (a)')
xlabel('x')
```

```
ylabel('y')
drawnow;

% Pose from Figure 1.15 (b)

% Connected to (0, 0) aka (x, y)
u1 = 2; v1 = 1;

% Connected to (x1, 0)
u2 = 3; v2 = 2;

% Connected to (x2, y2)
u3 = 1; v3 = 2;

x1 = 4; x2 = 0; y2 = 4;

figure(3)
plot([u1 u2 u3 u1], [v1 v2 v3 v1], 'r'); hold on          % Platform triangle
plot([0 x1 x2], [0 0 y2], 'bo')                          % Base anchors
plot([u1 u2 u3], [v1 v2 v3], 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r')
% Platform joints
plot([u1 0], [v1 0], 'k--')     % p1
plot([u2 x1], [v2 0], 'k--')    % p2
plot([u3 x2], [v3 y2], 'k--')   % p3
title('Figure 1.15 (b)')
xlabel('x')
ylabel('y')
drawnow;

% Here, we're just reproducing Figure 1.15 (a) and (b)
```


Plot of f(θ) on [-π, π]
×10^4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
f(θ)
-π/4
π/4
-4
-3
-2
-1
0
1
2
3
4
θ (radians)

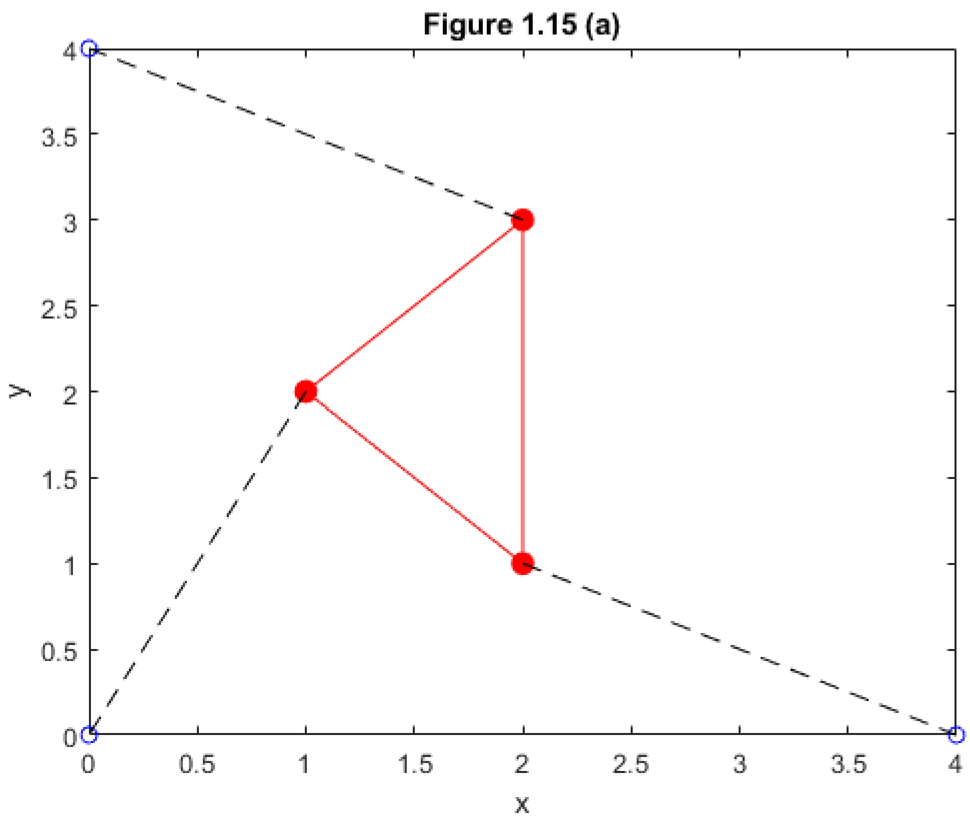
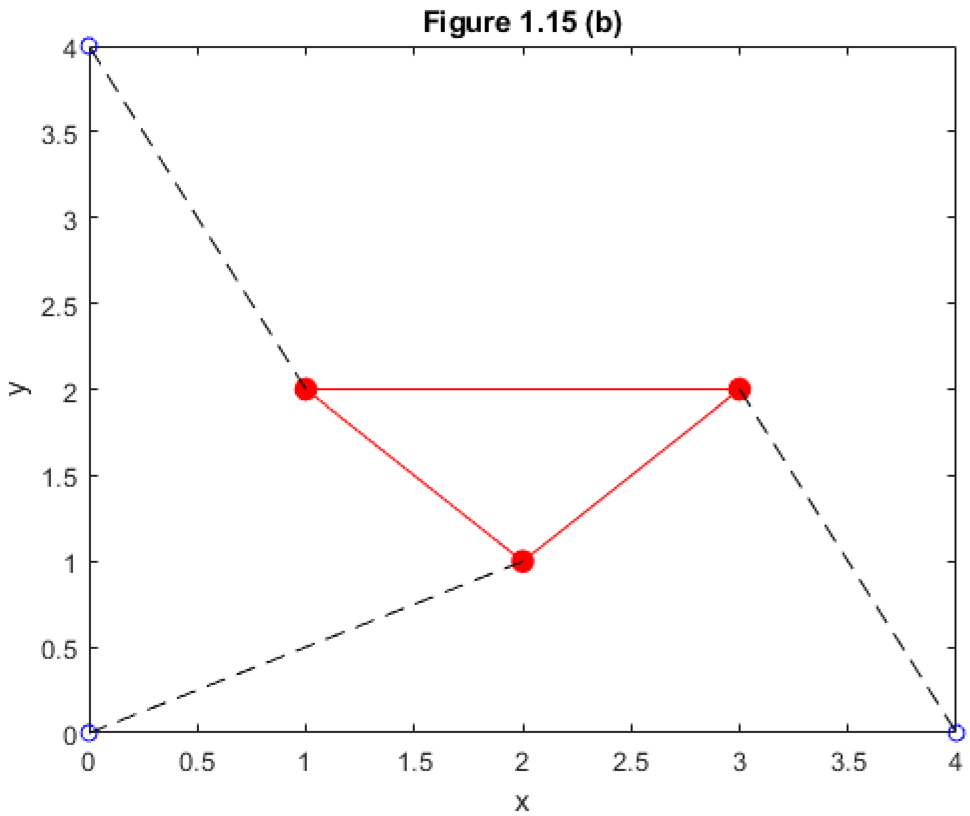


Figure 1.15 (a)
4
3.5
3
2.5
2
1.5
1
0.5
0
y
0
0.5
1
1.5
2
2.5
3
3.5
4
x



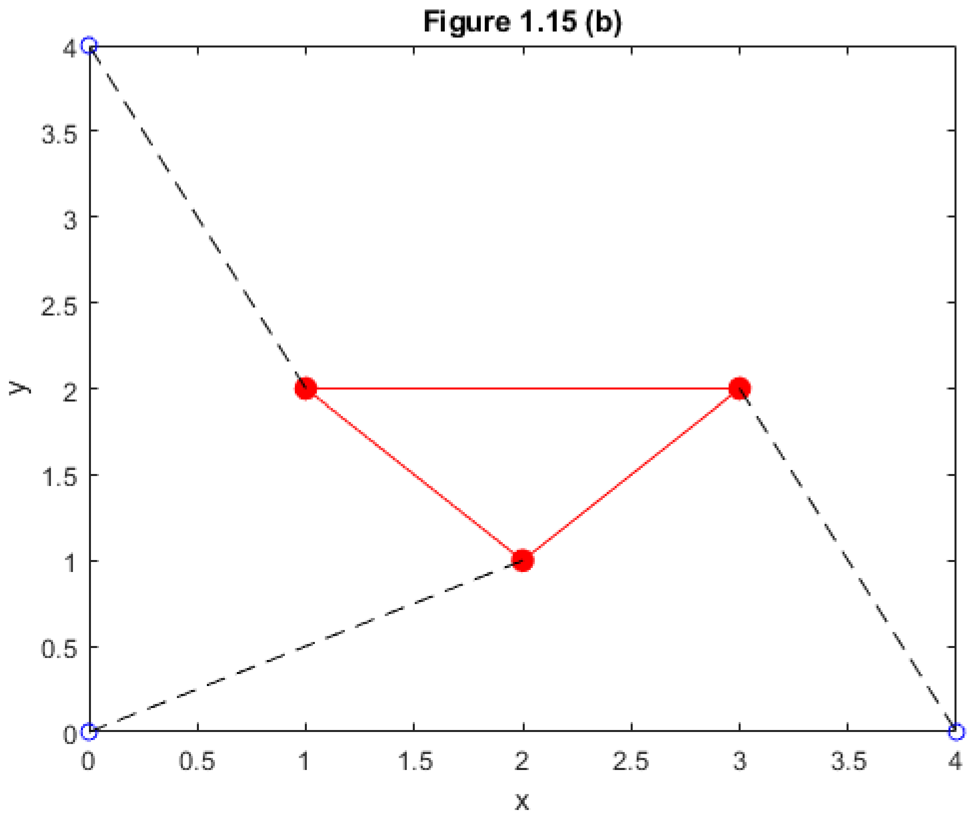
QUESTION 4:

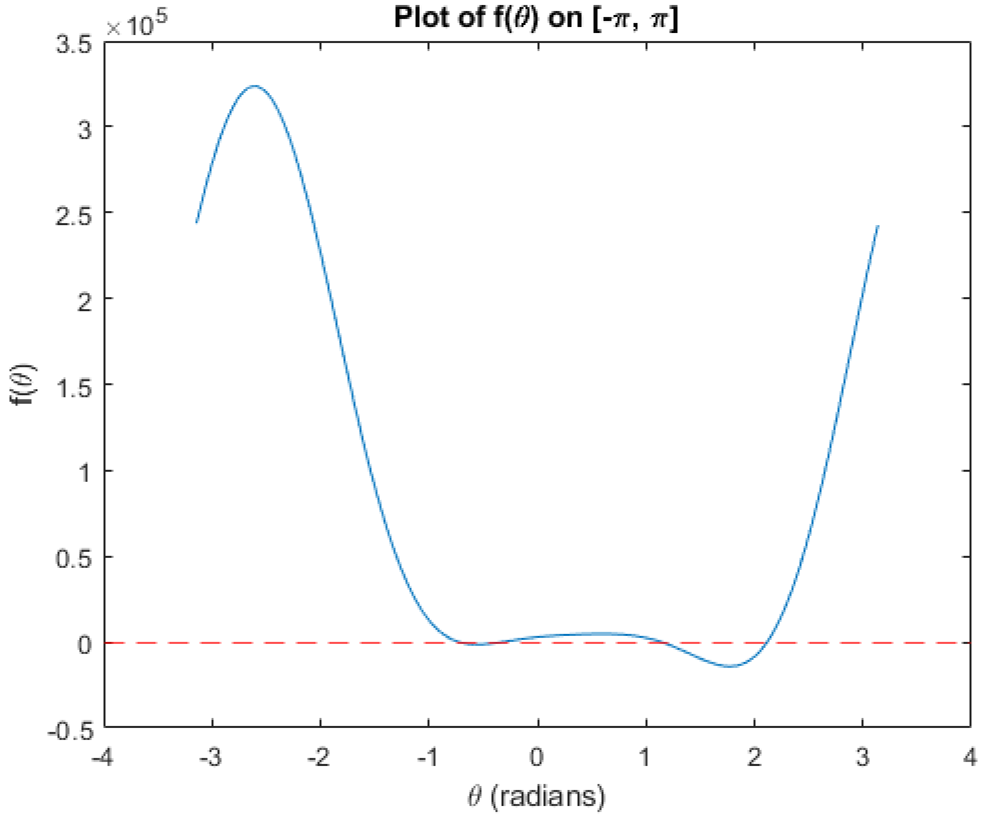
```
% Forward kinematics is when we compute (x, y) and theta for each given p1,
% p2, and p3

% The inverse kinematic problem is when we find p1, p2, p3, given x, y, and
% theta

% The new f(theta) function is in the supporting functions section at the
% bottom (f_4(theta))

% Plotting f_4(theta) on [-pi, pi]
theta_vals = -pi:0.01:pi;

f_vals = f_variable_p2(theta_vals, 5); % p2=5

figure(4)
plot(theta_vals, f_vals)
xlabel('\theta (radians)')
ylabel('f(\theta)')
title('Plot of f(\theta) on [-\pi, \pi]')
yline(0, '--r');
drawnow;

% Finding the four theta values (guesses are from eyeballing the graph)
```

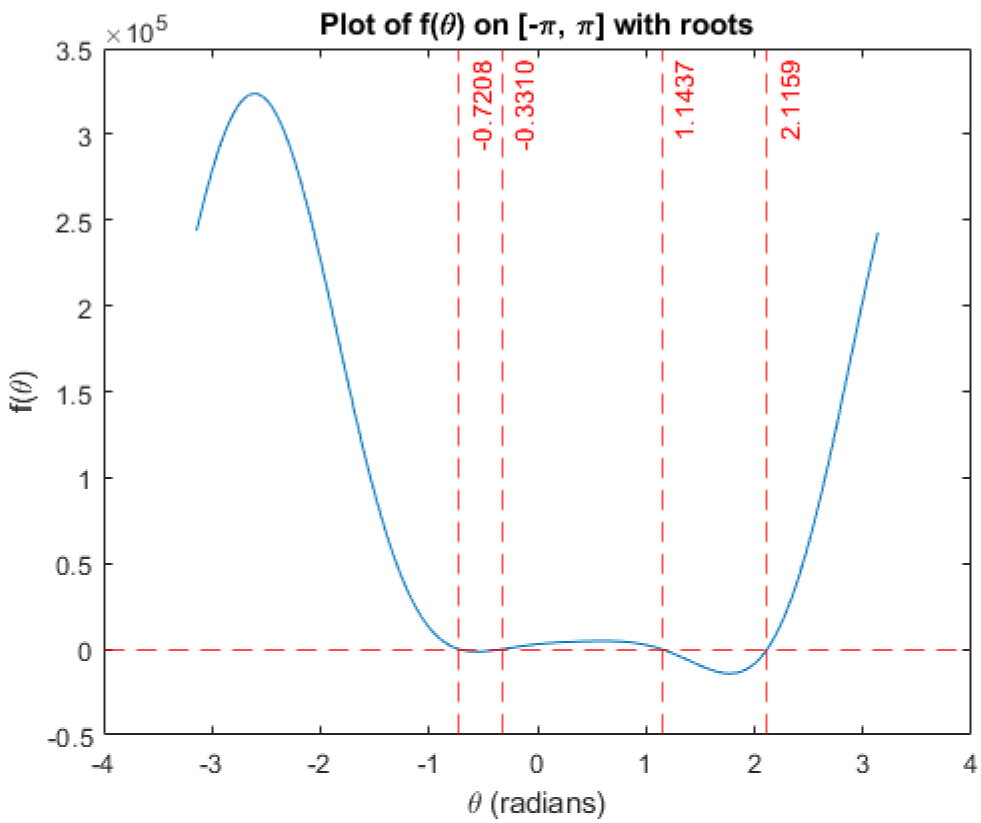
```
p2 = 5;
f_p2 = @(theta) f_variable_p2(theta, p2);

theta1 = fzero(f_p2, -0.72);
theta2 = fzero(f_p2, -0.33);
theta3 = fzero(f_p2, 1.14);
theta4 = fzero(f_p2, 2.11);
thetas = [theta1 theta2 theta3 theta4];

% From the above it appears that our roots are at:
% theta = -0.7208, -0.3310, 1.1437, and 2.1159 radians
figure(5)
plot(theta_vals, f_vals)
xlabel('\theta (radians)')
ylabel('f(\theta)')
title('Plot of f(\theta) on [-\pi, \pi] with roots')
yline(0, '--r');
xline(theta1, '--r', '-0.7208');
xline(theta2, '--r', '-0.3310');
xline(theta3, '--r', '1.1437');
xline(theta4, '--r', '2.1159');
drawnow;

% Since we're asked to solve the forward kinematics problem, we need to
% solve for x and y now (we just solved for theta)

% Finding the x and y coordinates for the four poses
% Created a new function at the bottom called
% forward_kinematics_variable_p2
[x_1 y_1] = forward_kinematics_variable_p2(theta1, p2);
[x_2 y_2] = forward_kinematics_variable_p2(theta2, p2);
[x_3 y_3] = forward_kinematics_variable_p2(theta3, p2);
[x_4 y_4] = forward_kinematics_variable_p2(theta4, p2);
xs = [x_1 x_2 x_3 x_4];
ys = [y_1 y_2 y_3 y_4];

% It was found that
% (x_1, y_1) = (-1.3784, 4.8063)
% (x_2, y_2) = (-0.9147, 4.9156)
% (x_3, y_3) = (4.4818, 2.2167)
% (x_4, y_4) = (4.5718, 2.0244)

% Now we need to plot the four poses
% Helper function is in the supporting functions section

for i = 1:4
    draw_pose(5+i, xs(i), ys(i), thetas(i), 4, i);
    drawnow;
end

% The strut lengths are correct!!!
```

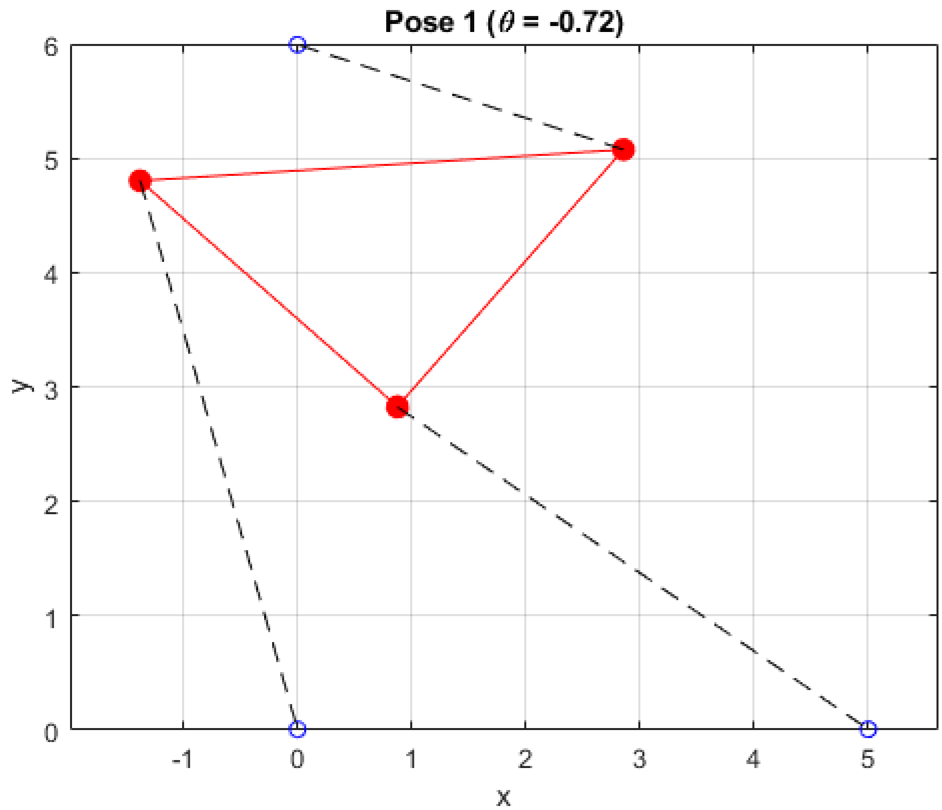
```
Pose 1: p1 = 5.0000, p2 = 5.0000, p3 = 3.0000
Pose 2: p1 = 5.0000, p2 = 5.0000, p3 = 3.0000
```

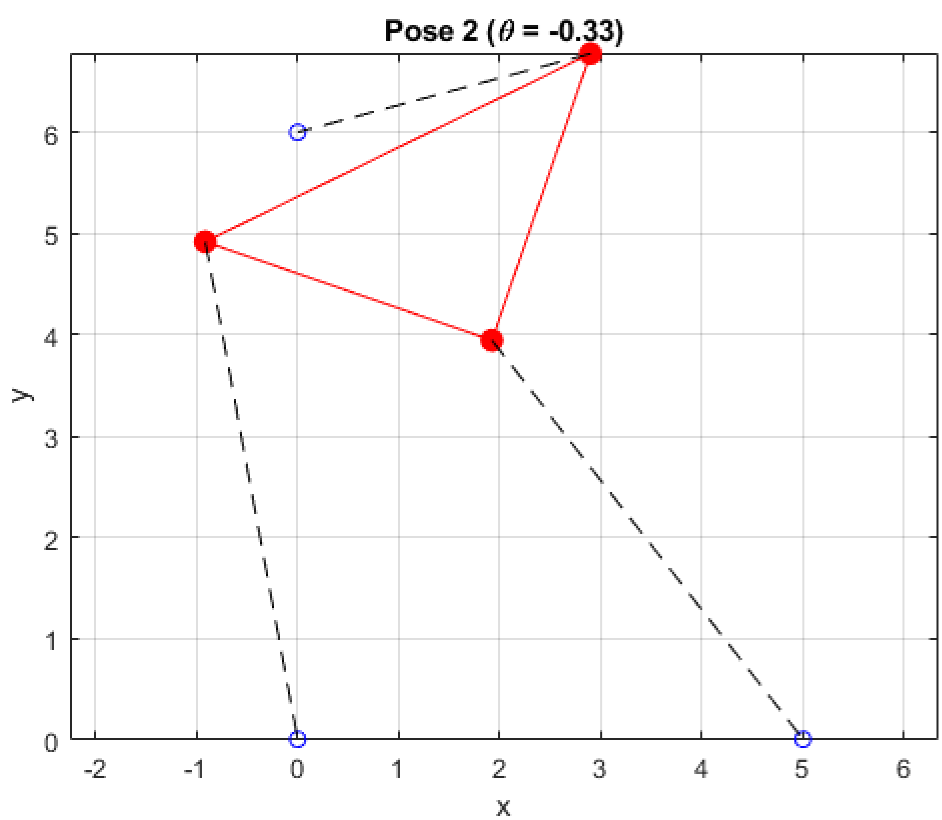
```
Pose 3: p1 = 5.0000, p2 = 5.0000, p3 = 3.0000
Pose 4: p1 = 5.0000, p2 = 5.0000, p3 = 3.0000
```

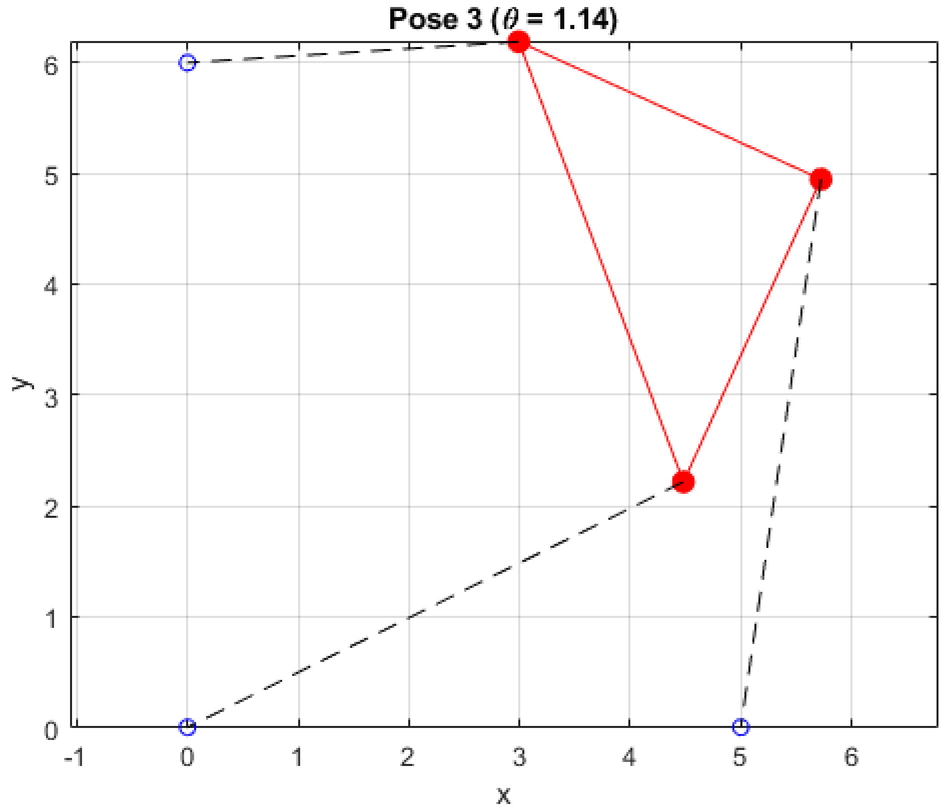


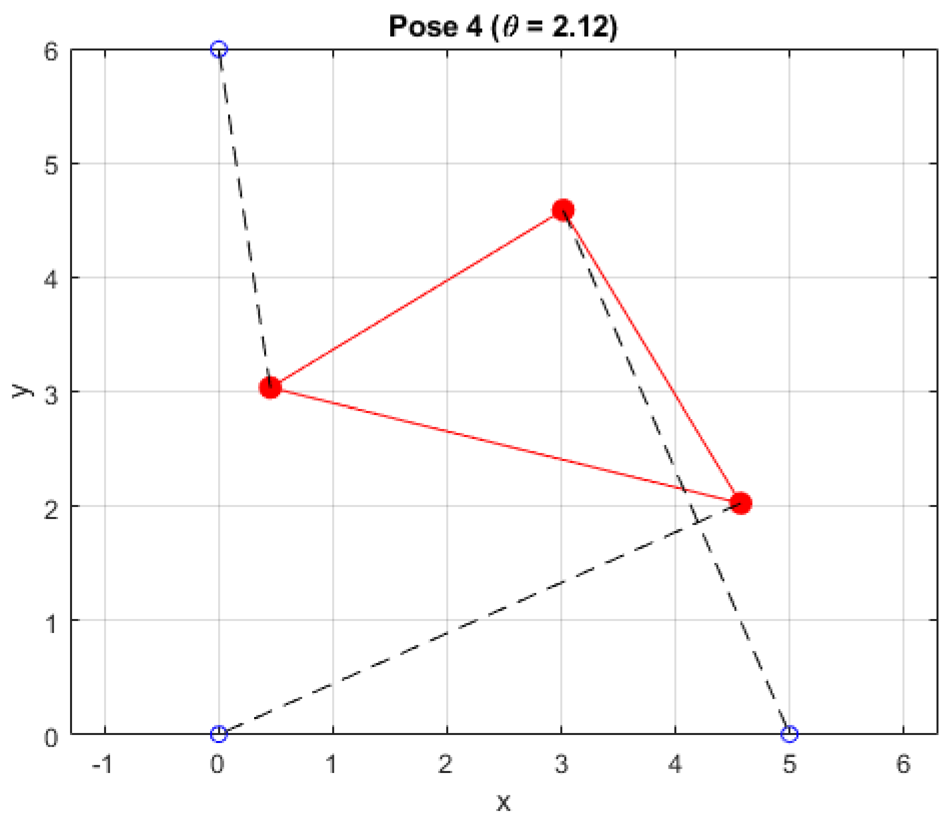

Plot of f(θ) on [-π, π]
×10^5
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-4
-3
-2
-1
0
1
2
3
4
f(θ)
θ (radians)


Plot of f(θ) on [-π, π] with roots
×10^5
-0.7208
-0.3310
1.1437
2.1159
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-4
-3
-2
-1
0
1
2
3
4
f(θ)
θ (radians)


Pose 1 (θ = -0.72)
y
x
6
5
4
3
2
1
0
-1
0
1
2
3
4
5


Pose 2 (θ = -0.33)
y
x
6
5
4
3
2
1
0
-2
-1
0
1
2
3
4
5
6


Pose 3 (θ = 1.14)
y
x
6
5
4
3
2
1
0
-1
0
1
2
3
4
5
6

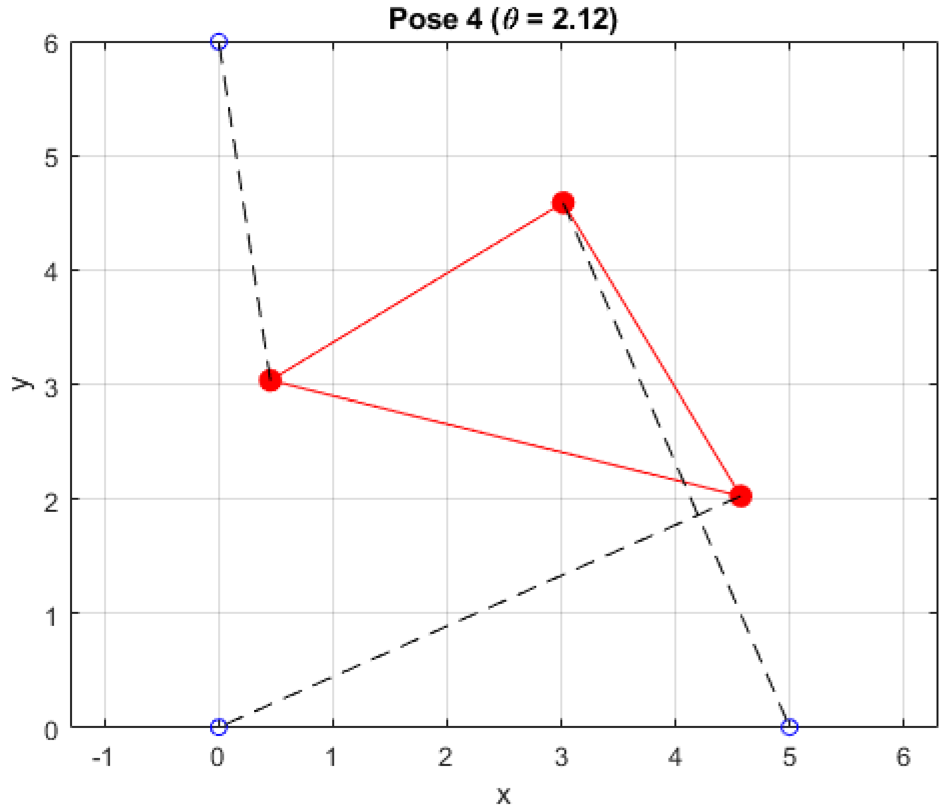

Pose 4 (θ = 2.12)
y
x
6
5
4
3
2
1
0
-1
0
1
2
3
4
5
6

QUESTION 5:

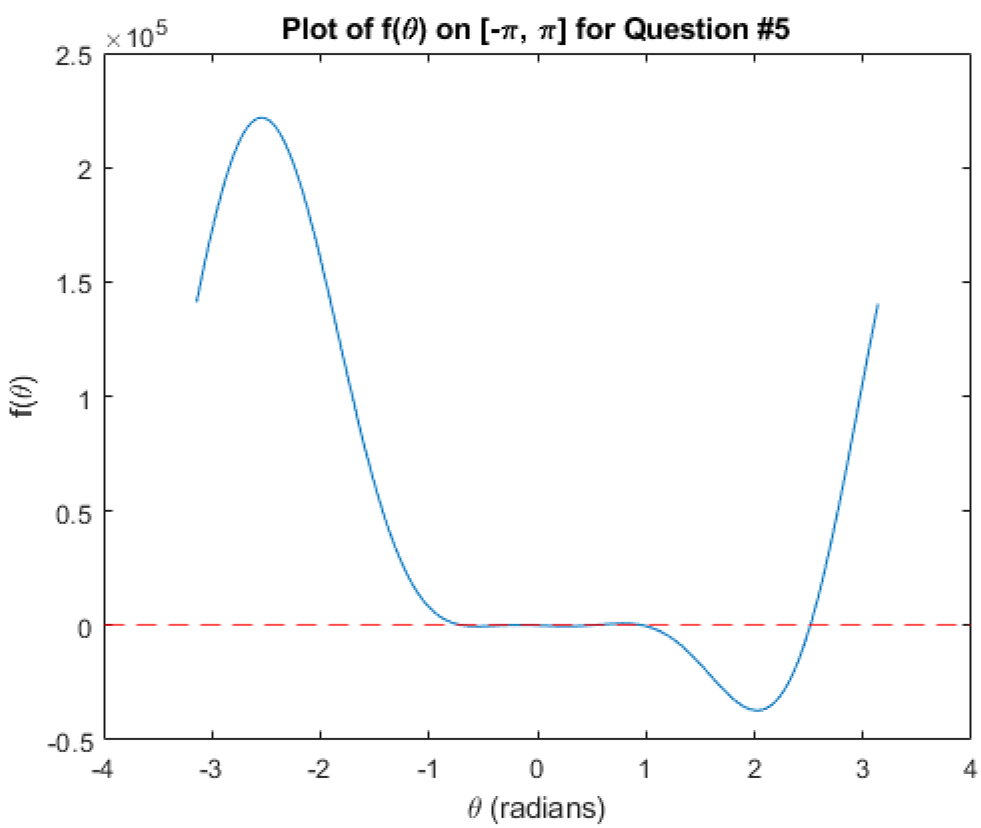
```
% Here we are changing p2 to 7 and resolving problem 4

% Plotting f(theta) on [-pi, pi]
theta_vals = -pi:0.01:pi;

f_vals = f_variable_p2(theta_vals, 7); % p2=7

figure(10)
plot(theta_vals, f_vals)
xlabel('\theta (radians)')
ylabel('f(\theta)')
title('Plot of f(\theta) on [-\pi, \pi] for Question #5')
yline(0, '--r');
drawnow;

% The problem states that there are now six poses

% Finding the six theta values (guesses are from eyeballing the graph)
p2 = 7;
f_p2 = @(theta) f_variable_p2(theta, p2);

theta1 = fzero(f_p2, -0.68);
theta2 = fzero(f_p2, -0.36);
theta3 = fzero(f_p2, 0.03);
theta4 = fzero(f_p2, 0.44);
theta5 = fzero(f_p2, 0.97);
theta6 = fzero(f_p2, 2.5);

thetas = [theta1 theta2 theta3 theta4 theta5 theta6];

% From the above it appears that our roots are at:
% theta = -0.6732, -0.3547, 0.0378, 0.4589, 0.9777, and 2.5139 rad
figure(11)
plot(theta_vals, f_vals)
xlabel('\theta (radians)')
ylabel('f(\theta)')
title('Plot of f(\theta) on [-\pi, \pi] with roots')
yline(0, '--r');
xline(theta1, '--r', '-0.6732');
xline(theta2, '--r', '-0.3547');
xline(theta3, '--r', '0.0378');
xline(theta4, '--r', '0.4589');
xline(theta5, '--r', '0.9777');
xline(theta6, '--r', '2.5139');
drawnow;

% Since we're asked to solve the forward kinematics problem, we need to
% solve for x and y now (we just solved for theta)

% Finding the x and y coordinates for the four poses
[x_1 y_1] = forward_kinematics_variable_p2(theta1, p2);
```

```
[x_2 y_2] = forward_kinematics_variable_p2(theta2, p2);
[x_3 y_3] = forward_kinematics_variable_p2(theta3, p2);
[x_4 y_4] = forward_kinematics_variable_p2(theta4, p2);
[x_5 y_5] = forward_kinematics_variable_p2(theta5, p2);
[x_6 y_6] = forward_kinematics_variable_p2(theta6, p2);

xs = [x_1 x_2 x_3 x_4 x_5 x_6];
ys = [y_1 y_2 y_3 y_4 y_5 y_6];

% It was found that
% (x_1, y_1) = (-4.3148, 2.5264)
% (x_2, y_2) = (-4.8049, 1.3831)
% (x_3, y_3) = (-4.9490, 0.7121)
% (x_4, y_4) = (-0.8198, 4.9323)
% (x_5, y_6) = (2.3036, 4.4378)
% (x_5, y_6) = (3.2157, 3.8287)

% Now we need to plot the four poses
% Helper function is in the supporting functions section

for i = 1:6
    draw_pose(11+i, xs(i), ys(i), thetas(i), 5, i);
    drawnow;
end

% The strut lengths are correct!!!
```

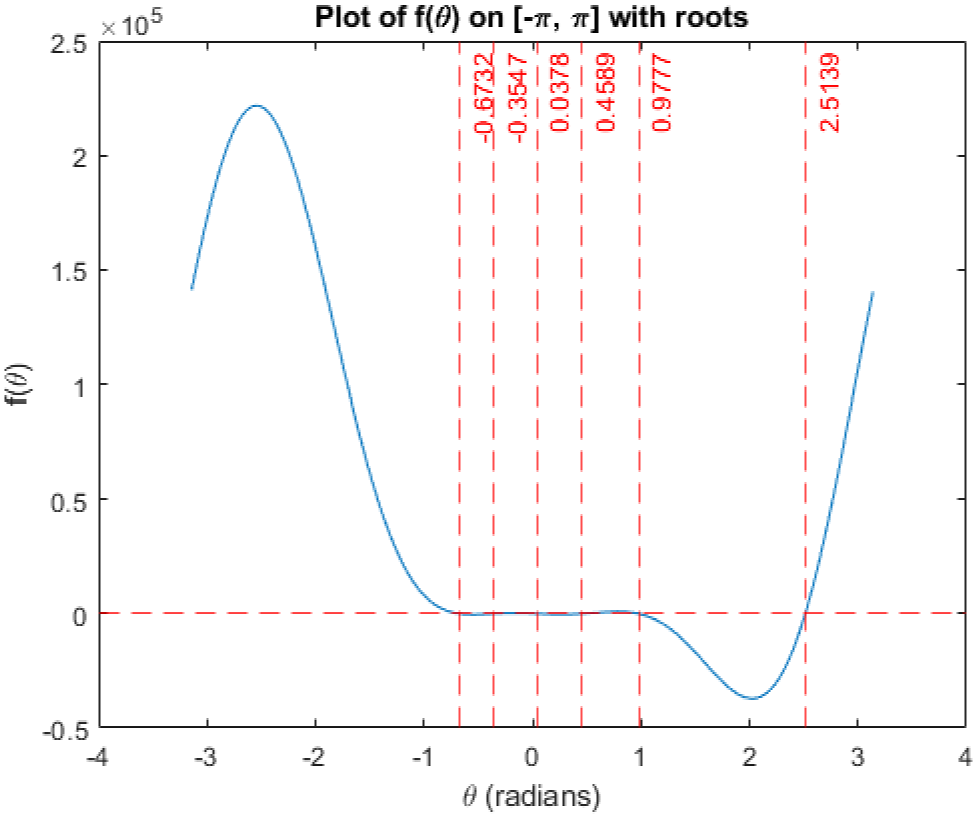
```
Pose 1: p1 = 5.0000, p2 = 7.0000, p3 = 3.0000
Pose 2: p1 = 5.0000, p2 = 7.0000, p3 = 3.0000
Pose 3: p1 = 5.0000, p2 = 7.0000, p3 = 3.0000
Pose 4: p1 = 5.0000, p2 = 7.0000, p3 = 3.0000
Pose 5: p1 = 5.0000, p2 = 7.0000, p3 = 3.0000
Pose 6: p1 = 5.0000, p2 = 7.0000, p3 = 3.0000
```

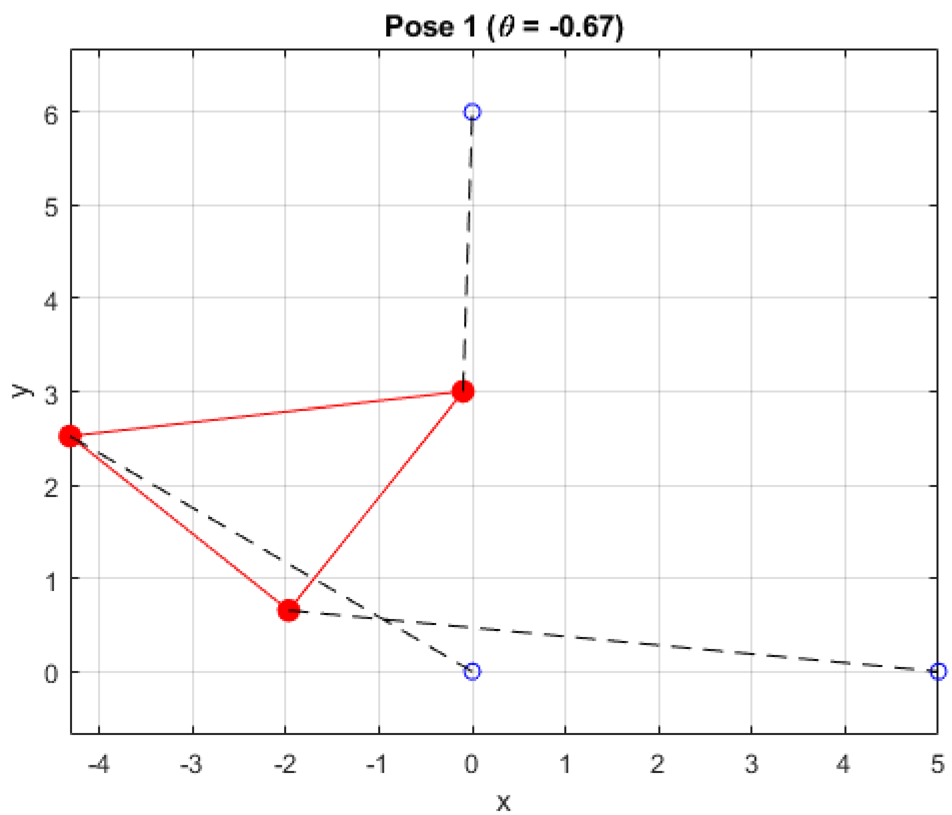


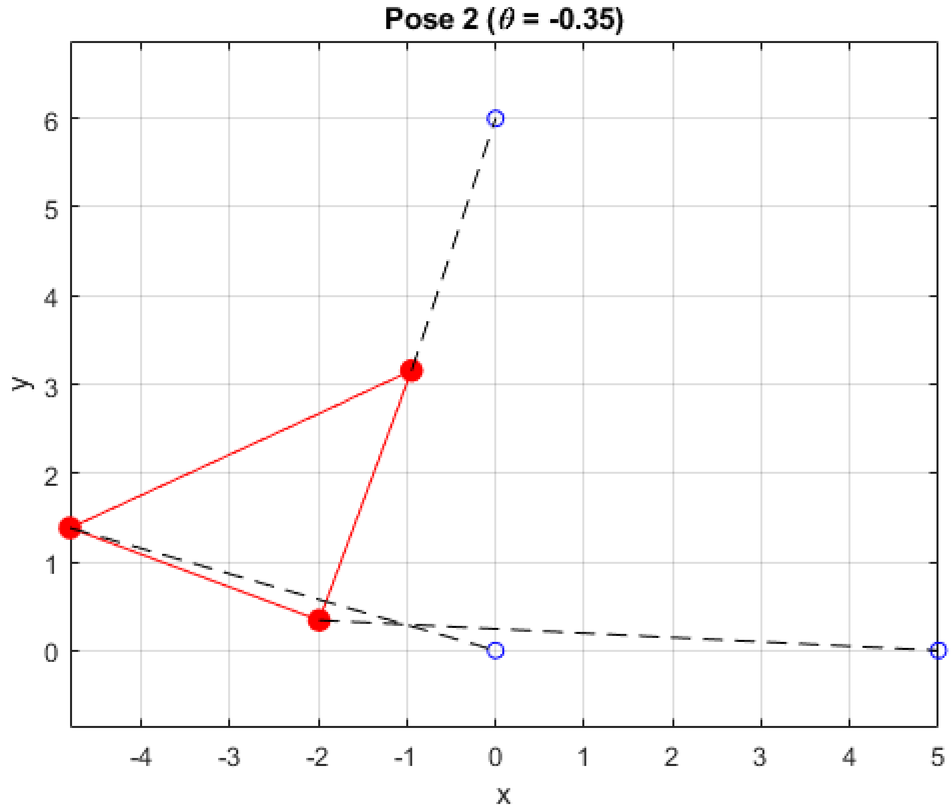
Pose 4 (θ = 2.12)
y
x

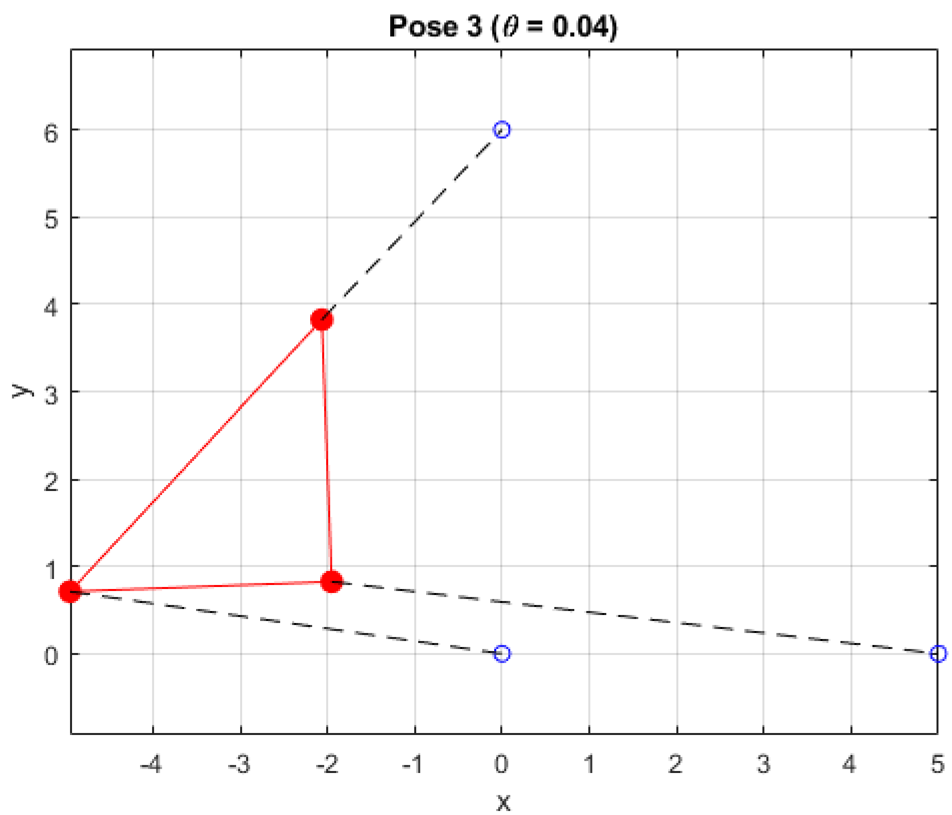


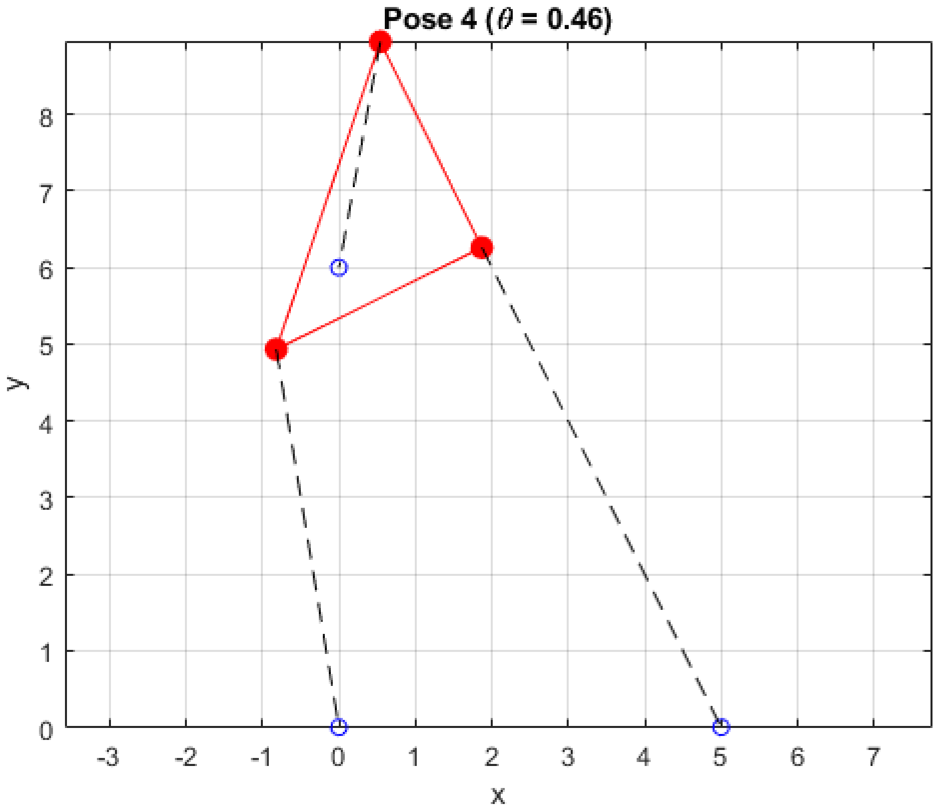
Plot of f(θ) on [-π, π] for Question #5
$\times 10^5$
f(θ)
θ (radians)


Plot of f(θ) on [-π, π] with roots
×10^5
-0.6732
-0.3547
0.0378
0.4589
0.9777
2.5139
f(θ)
θ (radians)

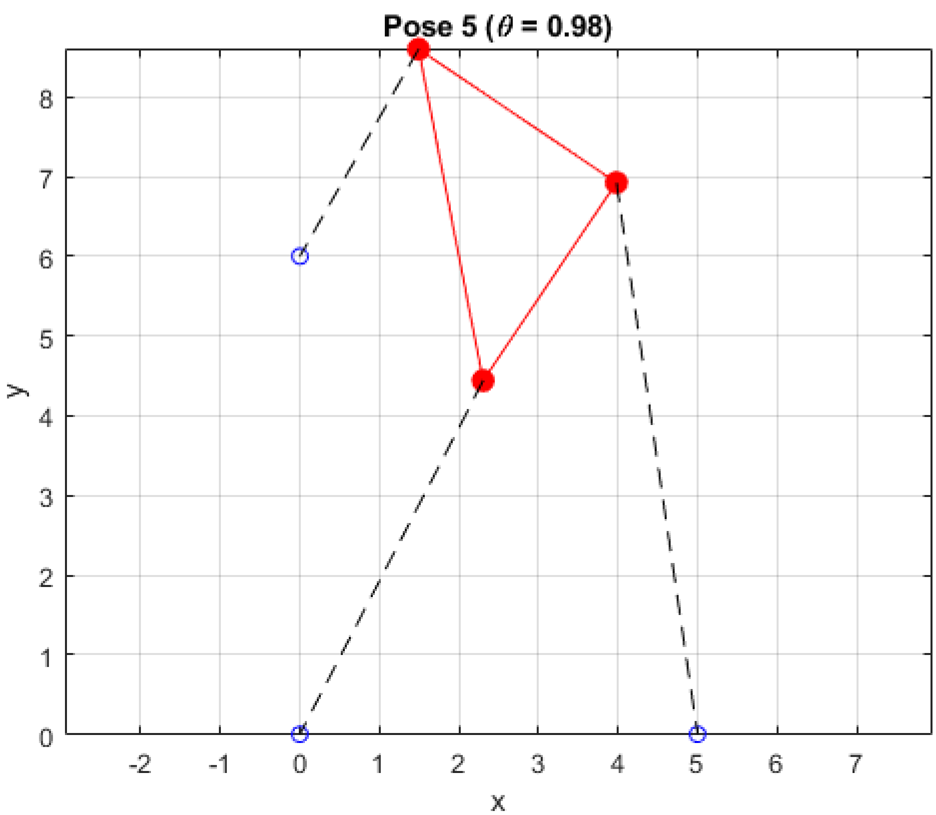

Pose 1 (θ = -0.67)
y
x


Pose 2 (θ = -0.35)
y
x

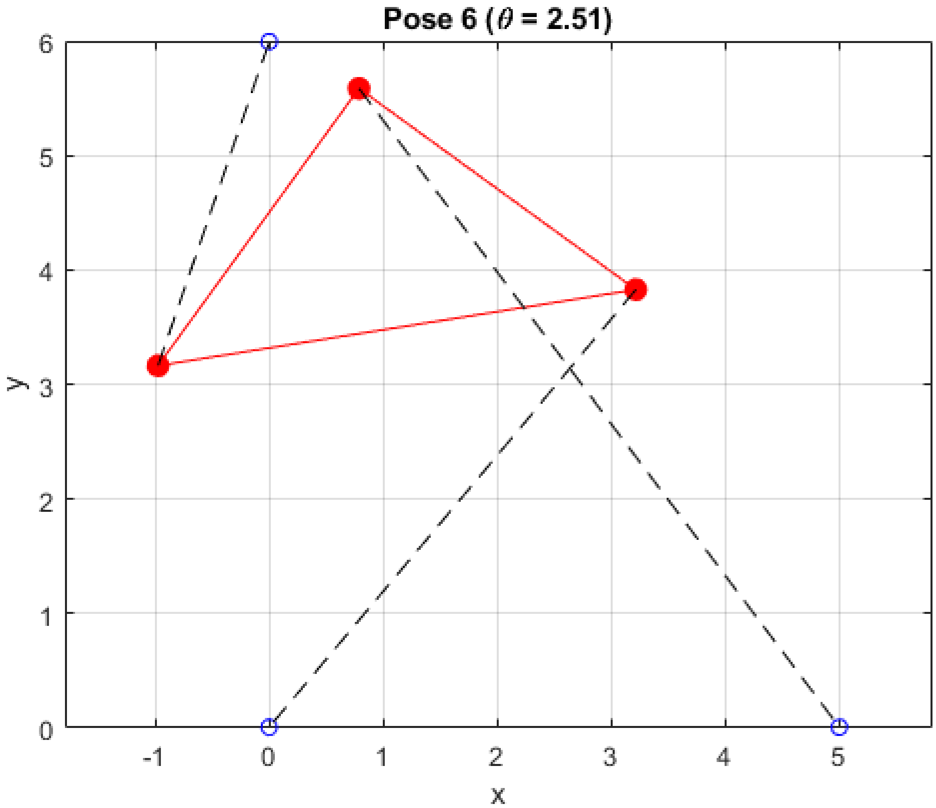

Pose 3 (θ = 0.04)
y
x



Pose 4 (θ = 0.46)
y
8
7
6
5
4
3
2
1
0
-3
-2
-1
0
1
2
3
4
5
6
7
x



Pose 5 (θ = 0.98)
y
8
7
6
5
4
3
2
1
0
-2
-1
0
1
2
3
4
5
6
7
x

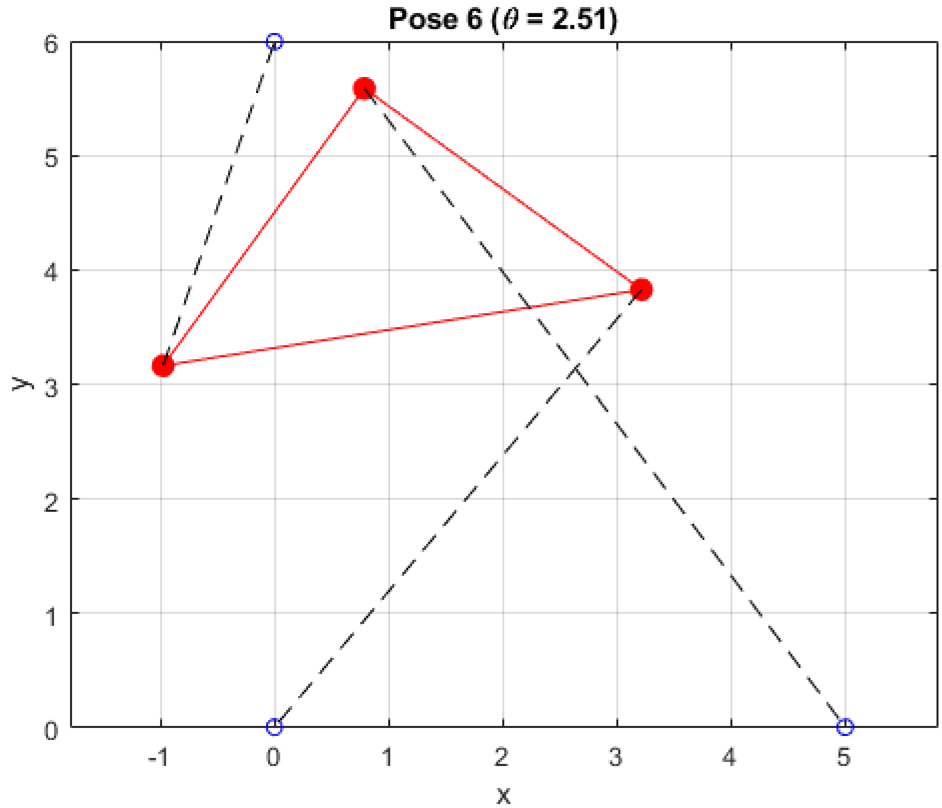


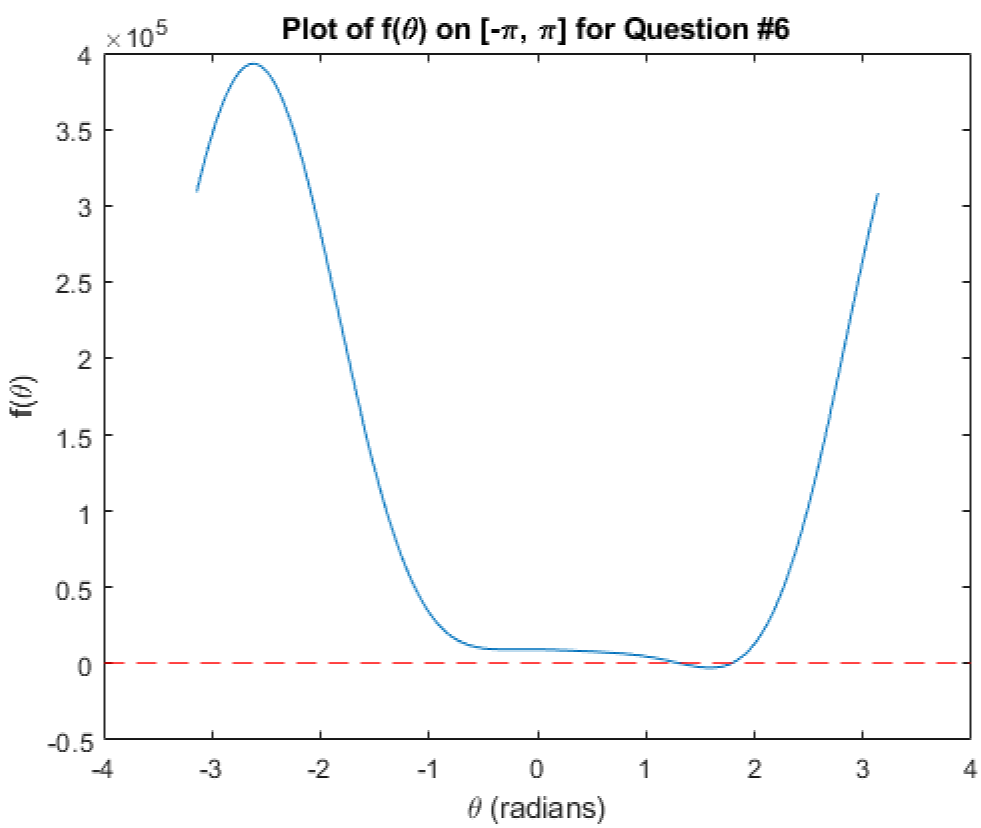
QUESTION 6:

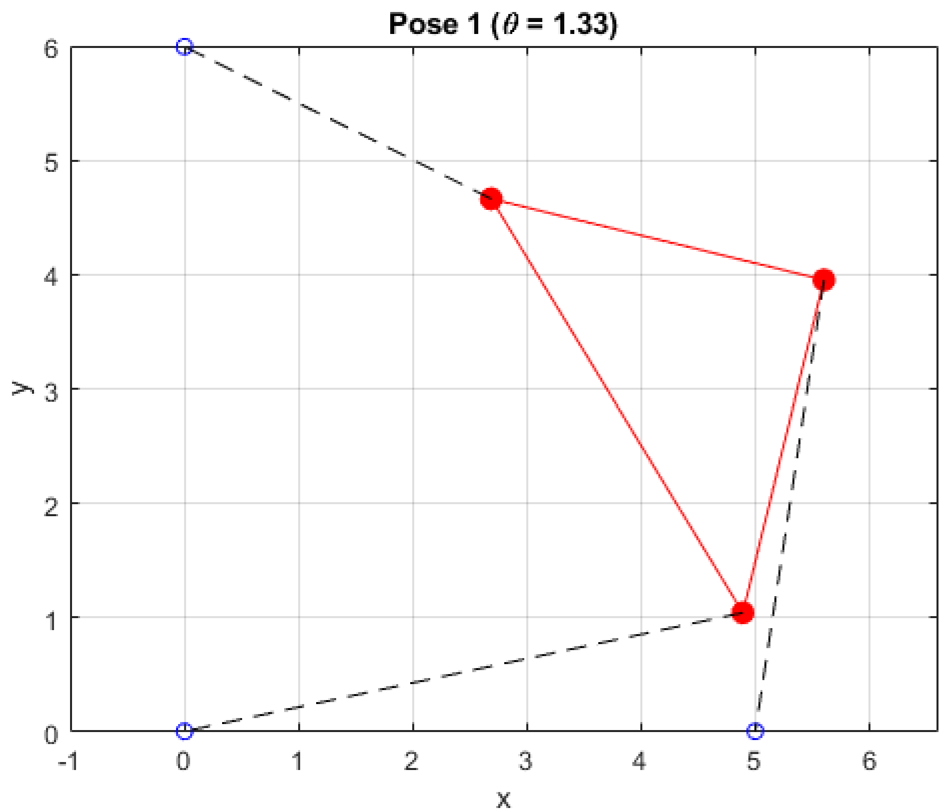
```
% Now we need to find a strut length p2, for which there are only two
% poses. It is found that when p2=4,  there are only two poses

theta_vals = -pi:0.01:pi;
p2_range = 1:7;  % Test p2 values between 1 and 7
target_num_roots = 2;
found = false;

for p2 = p2_range
    f_vals = f_variable_p2(theta_vals, p2);
    sign_changes = sum(abs(diff(sign(f_vals))) == 2);  % # of times f(theta)
crosses 0
    if sign_changes == target_num_roots
        fprintf("Found p2 = %.2f with exactly %d poses\n", p2,
target_num_roots);
        found = true;
        break
    end
end

if ~found
    fprintf("No p2 in range [%0.2f, %0.2f] gives exactly %d poses\n", ...
        p2_range(1), p2_range(end), target_num_roots);
```

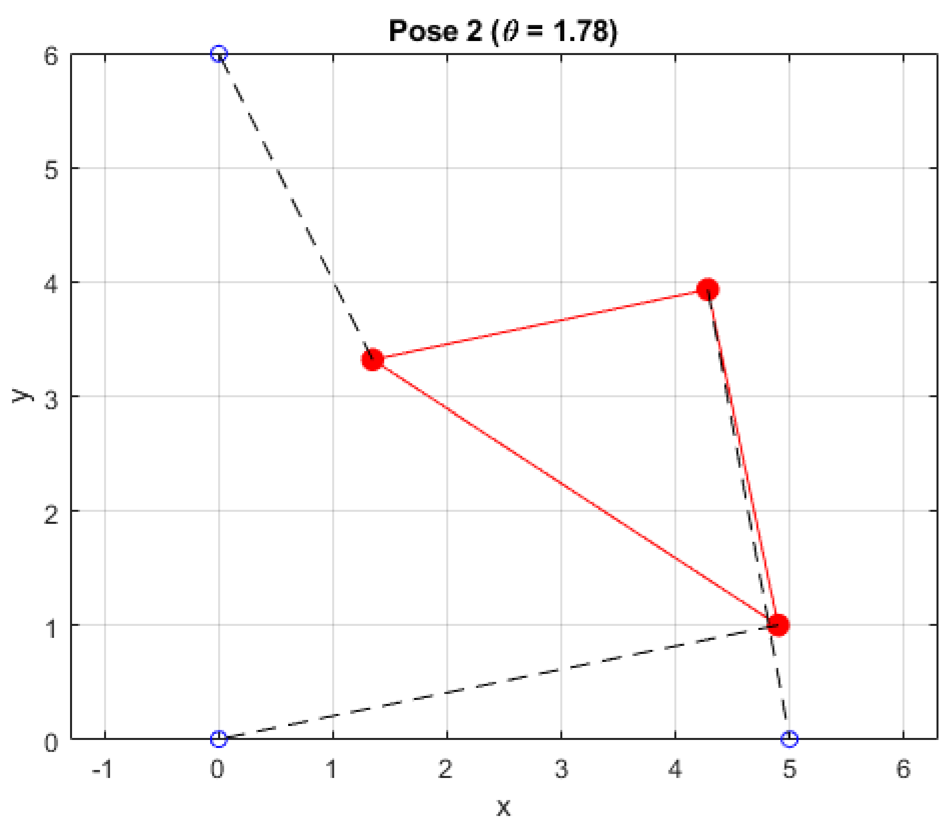
```
end

% Here we are changing p2 to 4

% Plotting f(theta) on [-pi, pi]
theta_vals = -pi:0.01:pi;

f_vals = f_variable_p2(theta_vals, 4); % p2=4

figure(18)
plot(theta_vals, f_vals)
xlabel('\theta (radians)')
ylabel('f(\theta)')
title('Plot of f(\theta) on [-\pi, \pi] for Question #6')
yline(0, '--r');
drawnow;

% Finding the six theta values (guesses are from eyeballing the graph)
p2 = 4;
f_p2 = @(theta) f_variable_p2(theta, p2);

theta1 = fzero(f_p2, 1.32);
theta2 = fzero(f_p2, 1.77);

thetas = [theta1 theta2];

% theta vals are 1.3316 and 1.7775 rad

% Since we're asked to solve the forward kinematics problem, we need to
% solve for x and y now (we just solved for theta)

% Finding the x and y coordinates for the four poses
[x_1 y_1] = forward_kinematics_variable_p2(theta1, p2);
[x_2 y_2] = forward_kinematics_variable_p2(theta2, p2);

xs = [x_1 x_2];
ys = [y_1 y_2];

% It was found that
% (x_1, y_1) = (4.8907, 1.0399)
% (x_2, y_2) = (4.8992, 0.9992)

% Now we need to plot the four poses
% Helper function is in the supporting functions section

for i = 1:2
    draw_pose(18+i, xs(i), ys(i), thetas(i), 6, i);
    drawnow;
end
```

Found p2 = 4.00 with exactly 2 poses
Pose 1: p1 = 5.0000, p2 = 4.0000, p3 = 3.0000
Pose 2: p1 = 5.0000, p2 = 4.0000, p3 = 3.0000


Pose 6 (θ = 2.51)
y
x
0
1
2
3
4
5
6
-1
0
1
2
3
4
5

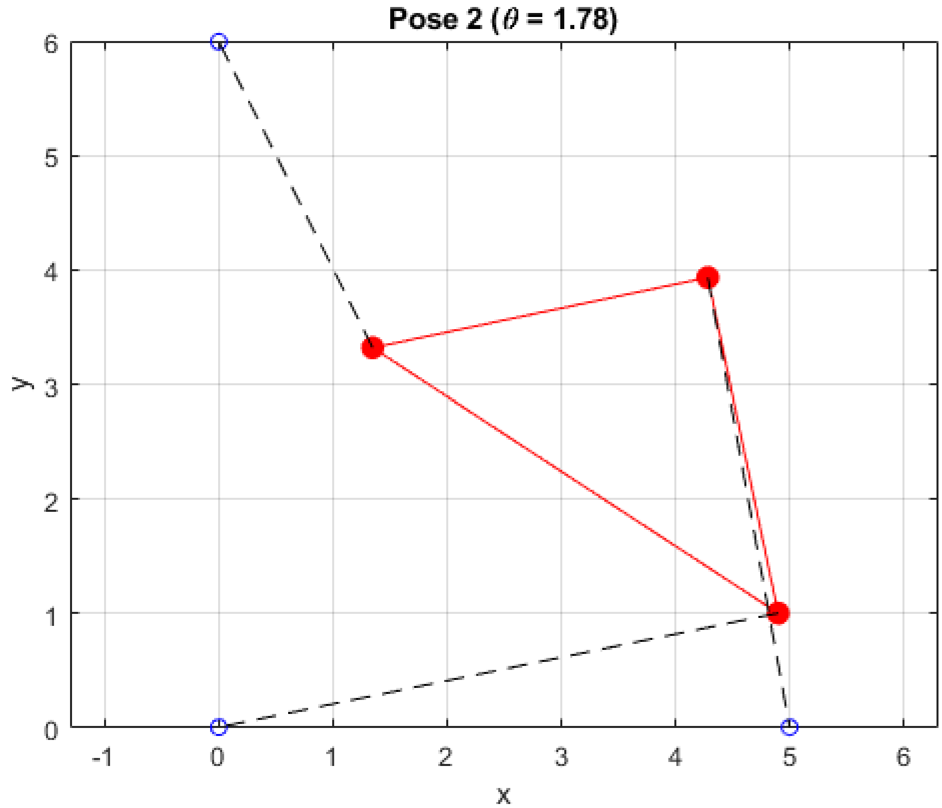

Plot of f(θ) on [-π, π] for Question #6
$\times 10^5$
f(θ)
θ (radians)
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-4
-3
-2
-1
0
1
2
3
4


Pose 1 (θ = 1.33)
y
x
-1
0
1
2
3
4
5
6

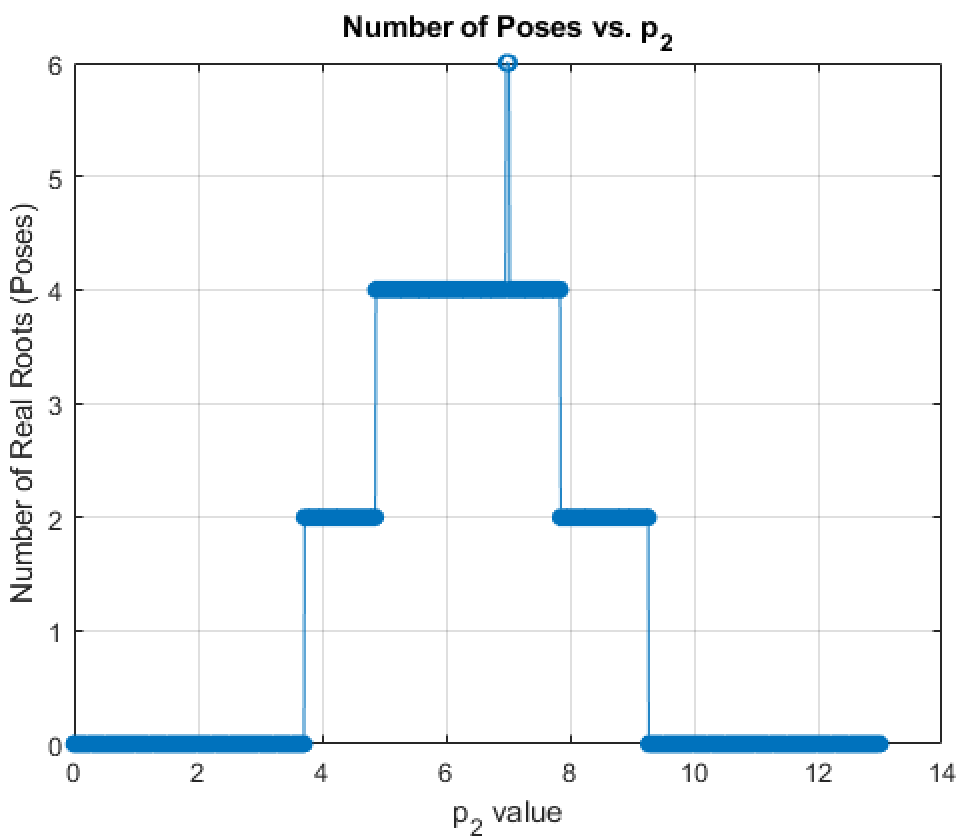

Pose 2 (θ = 1.78)
y
x
-1
0
1
2
3
4
5
6

QUESTION 7:

```
theta_vals = -pi:0.01:pi;
p2_range = 0:0.01:13;
pose_counts = zeros(size(p2_range));

for i = 1:length(p2_range)
    p2 = p2_range(i);
    f_vals = f_variable_p2(theta_vals, p2);
    pose_counts(i) = sum(abs(diff(sign(f_vals))) == 2);  % number of real
roots
end

% Plot how the number of real roots changes with p2
figure(21);
plot(p2_range, pose_counts, '-o');
xlabel('p_2 value');
ylabel('Number of Real Roots (Poses)');
title('Number of Poses vs. p_2');
grid on;

% Based on the graph, it appears that:
% there are 0 poses when p2 < 3.7 and when p2 > 9.27
% there are 2 poses when 3.7 < p2 < 4.86 and 7.85 < p2 < 9.26
% there are 4 poses when 4.87 < p2 < 6.96 and 7.03 < p2 < 7.84
% there are 6 poses when 6.97 < p2 < 7.02
```



Pose 2 (θ = 1.78)
y
x



Number of Poses vs. p_2
Number of Real Roots (Poses)
p_2 value

QUESTION 8:

```
% 1. Define base attachment points (fixed)
B = [...   % Each row is a point (xi, yi, zi)
    200, 0, 0;
    100, 173.2, 0;
   -100, 173.2, 0;
   -200, 0, 0;
   -100, -173.2, 0;
    100, -173.2, 0
];

% 2. Define platform attachment points (in platform frame)
P = [... % Each row is a point (xi, yi, zi)
     100, 0, 0;
     50, 86.6, 0;
    -50, 86.6, 0;
   -100, 0, 0;
    -50, -86.6, 0;
     50, -86.6, 0
];

% 3. Define leg lengths (example values)
L = [250; 240; 245; 260; 255; 250];  % Length of each leg

% 4. Objective function to minimize
f_obj = @(x) stewart_error(x, B, P, L);

% 5. Initial guess for pose: [x, y, z, alpha, beta, gamma]
x0 = [0; 0; 200; 0; 0; 0];  % Start near z = 200 mm, no rotation

% 6. Use fsolve to solve
options = optimoptions('fsolve', 'Display', 'iter', 'TolFun', 1e-10);
[x_sol, fval, exitflag] = fsolve(f_obj, x0, options);

disp('Estimated pose:')
disp(['x = ', num2str(x_sol(1)), ', y = ', num2str(x_sol(2)), ...
      ', z = ', num2str(x_sol(3))])
disp(['alpha = ', num2str(x_sol(4)), ', beta = ', num2str(x_sol(5)), ...
      ', gamma = ', num2str(x_sol(6))])

% visualize_stewart(B, P, x_sol);

interactive_stewart(B, P);
```

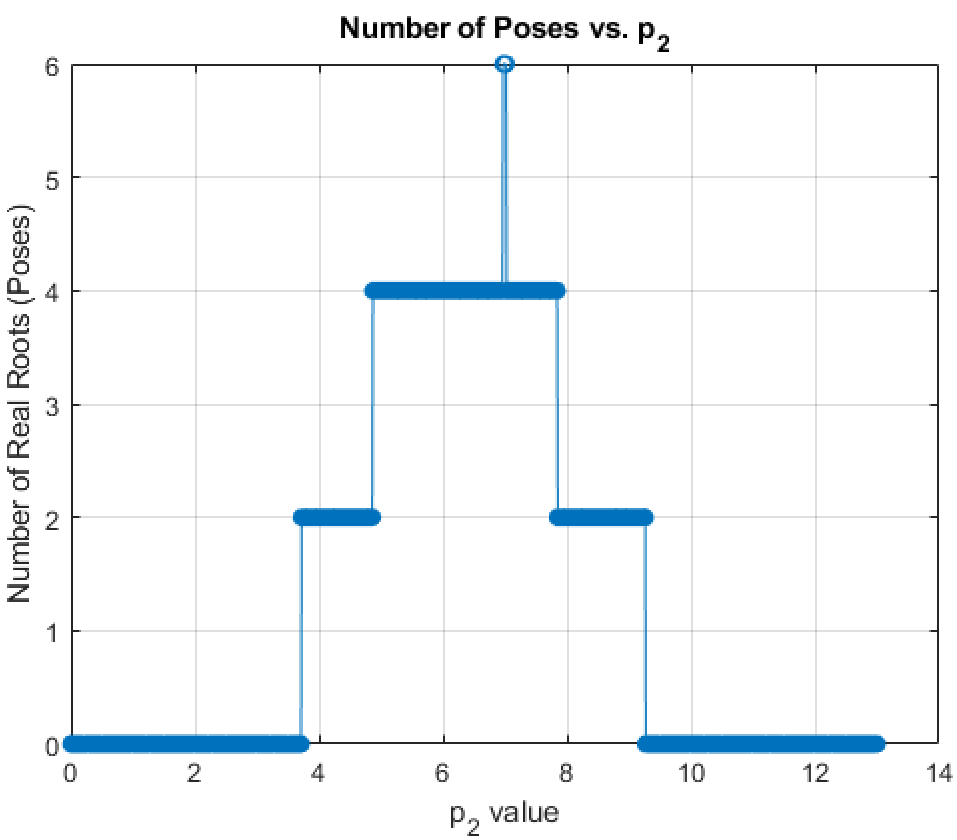
```
                                             Norm of      First-order
Trust-region
 Iteration  Func-count     ||f(x)||^2           step
optimality         radius
     0           7         1.00631e+09
3.43e+08               1
     1          14         9.48743e+08       0.0959583
```

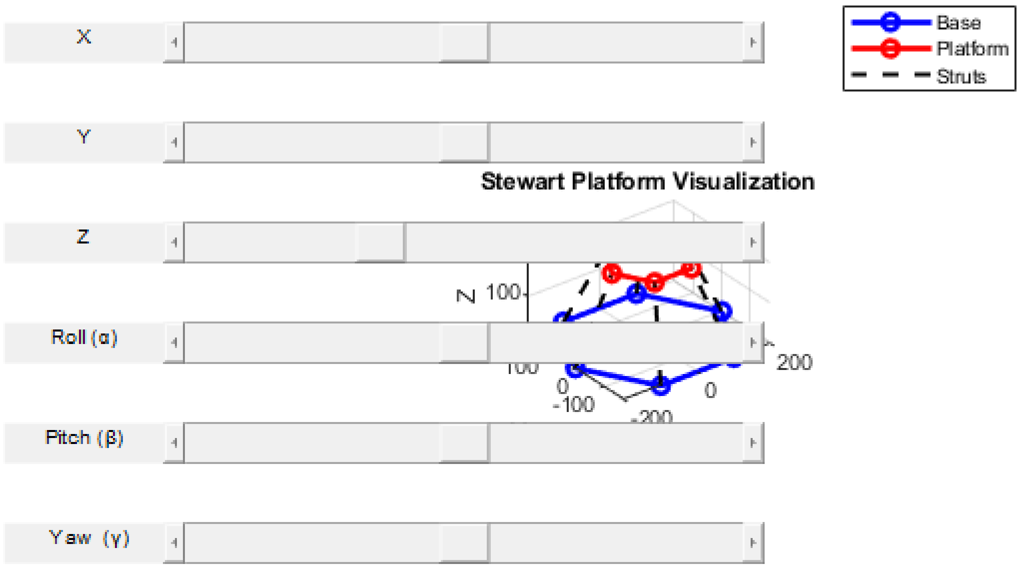
```
1.03e+08            1
     2         21          9.12547e+08              1
2.89e+08            1
     3         22          9.12547e+08            2.5
2.89e+08          2.5
     4         29          7.65115e+07          0.625
3.88e+08        0.625
     5         36          1.81799e+07         1.5625
2.3e+07          1.56
     6         43          1.80569e+07        3.90625
9.08e+06         3.91
     7         44          1.80569e+07        3.90625
9.08e+06         3.91
     8         51          1.80381e+07       0.976562
5.86e+05        0.977
     9         52          1.80381e+07        2.44141
5.86e+05         2.44
    10         53          1.80381e+07       0.610352
5.86e+05         0.61
    11         60           1.8038e+07       0.152588
9.78e+04        0.153
    12         67           1.8038e+07       0.152588
3.34e+05        0.153
    13         74          1.80379e+07       0.152588
9.78e+04        0.153
    14         81          1.80379e+07       0.152588
2.19e+05        0.153
    15         88          1.80379e+07       0.152588
9.78e+04        0.153
    16         95          1.80379e+07       0.152588
1.4e+05        0.153
    17         96          1.80379e+07       0.152588
1.4e+05        0.153
    18        103          1.80379e+07       0.038147
9.75e+04       0.0381
    19        110          1.80379e+07       0.038147
1.27e+05       0.0381
    20        117          1.80379e+07      0.0953674
1.01e+05       0.0954
    21        124          1.80379e+07      0.0953674
1.05e+05       0.0954
    22        131          1.80379e+07      0.0953674
9.71e+04       0.0954
    23        138          1.80379e+07      0.0953674
9.86e+04       0.0954
    24        145          1.80379e+07      0.0953674
9.7e+04       0.0954
    25        152          1.80379e+07      0.0953674
9.82e+04       0.0954
    26        159          1.80379e+07      0.0953674
9.68e+04       0.0954
    27        166          1.80379e+07      0.0953674
9.8e+04       0.0954
    28        173          1.80378e+07      0.0953674
```

```
9.67e+04          0.0954
    29          180          1.80378e+07       0.0953674
9.82e+04          0.0954
    30          187          1.80378e+07       0.0953674
9.7e+04          0.0954
    31          194          1.80378e+07       0.0953674
9.83e+04          0.0954
    32          201          1.80378e+07       0.0953674
9.68e+04          0.0954
    33          208          1.80378e+07       0.0953674
9.8e+04          0.0954
    34          215          1.80378e+07       0.0953674
9.67e+04          0.0954
    35          222          1.80378e+07       0.0953674
9.81e+04          0.0954
    36          229          1.80378e+07       0.0953674
9.67e+04          0.0954
    37          236          1.80378e+07       0.0953674
9.81e+04          0.0954
    38          243          1.80378e+07       0.0953674
9.67e+04          0.0954
    39          250          1.80378e+07       0.0953674
9.91e+04          0.0954
    40          257          1.80378e+07       0.0953674
9.65e+04          0.0954
    41          264          1.80378e+07       0.0953674
9.79e+04          0.0954
    42          271          1.80377e+07       0.0953674
9.65e+04          0.0954
    43          278          1.80377e+07       0.0953674
1.04e+05          0.0954
    44          285          1.80377e+07       0.0953674
9.64e+04          0.0954
    45          292          1.80377e+07       0.0953674
9.79e+04          0.0954
    46          299          1.80377e+07       0.0953674
9.64e+04          0.0954
    47          306          1.80377e+07       0.0953674
9.77e+04          0.0954
    48          313          1.80377e+07       0.0953674
9.64e+04          0.0954
    49          320          1.80377e+07       0.0953674
9.77e+04          0.0954
    50          327          1.80377e+07       0.0953674
9.64e+04          0.0954
    51          334          1.80377e+07       0.0953674
9.79e+04          0.0954
    52          341          1.80377e+07       0.0953674
9.64e+04          0.0954
    53          348          1.80377e+07       0.0953674
9.76e+04          0.0954
    54          355          1.80377e+07       0.0953674
9.62e+04          0.0954
    55          362          1.80377e+07       0.0953674
```

```
9.74e+04          0.0954
    56         369          1.80377e+07       0.0953674
9.61e+04          0.0954
    57         376          1.80377e+07       0.0953674
9.74e+04          0.0954
    58         383          1.80377e+07       0.0953674
9.6e+04          0.0954
    59         390          1.80376e+07       0.0953674
9.74e+04          0.0954
    60         397          1.80376e+07       0.0953674
9.62e+04          0.0954
    61         404          1.80376e+07       0.0953674
1.02e+05          0.0954
    62         411          1.80376e+07       0.0953674
9.64e+04          0.0954
    63         418          1.80376e+07       0.0953674
1.08e+05          0.0954
    64         425          1.80376e+07       0.0953674
9.65e+04          0.0954
    65         432          1.80376e+07       0.0953674
9.88e+04          0.0954
    66         439          1.80376e+07       0.0953674
9.64e+04          0.0954
    67         446          1.80376e+07       0.0953674
9.76e+04          0.0954
    68         453          1.80376e+07       0.0953674
9.64e+04          0.0954
    69         460          1.80376e+07       0.0953674
9.77e+04          0.0954
    70         467          1.80376e+07       0.0953674
9.64e+04          0.0954
    71         474          1.80376e+07       0.0953674
9.79e+04          0.0954
    72         481          1.80376e+07       0.0953674
9.66e+04          0.0954
    73         488          1.80376e+07       0.0953674
9.81e+04          0.0954
    74         495          1.80375e+07       0.0953674
9.66e+04          0.0954
    75         502          1.80375e+07       0.0953674
9.78e+04          0.0954
    76         509          1.80375e+07       0.0953674
9.64e+04          0.0954
    77         516          1.80375e+07       0.0953674
9.76e+04          0.0954
    78         523          1.80375e+07       0.0953674
9.63e+04          0.0954
    79         530          1.80375e+07       0.0953674
9.76e+04          0.0954
    80         537          1.80375e+07       0.0953674
9.61e+04          0.0954
    81         544          1.80375e+07       0.0953674
9.74e+04          0.0954
    82         551          1.80375e+07       0.0953674
```

```
9.59e+04          0.0954
    83        558          1.80375e+07       0.0953674
9.73e+04          0.0954
    84        565          1.80375e+07       0.0953674
9.62e+04          0.0954
    85        572          1.80375e+07       0.0953674
9.74e+04          0.0954
    86        579          1.80375e+07       0.0953674
9.62e+04          0.0954
    87        586          1.80375e+07       0.0953674
9.76e+04          0.0954
    88        593          1.80375e+07       0.0953674
9.64e+04          0.0954
    89        600          1.80374e+07       0.0953674
9.77e+04          0.0954

Solver stopped prematurely.

fsolve stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 6.000000e+02.

Estimated pose:
x = 0.052786, y = -0.99558, z = 205.3665
alpha = -0.075693, beta = 0.067383, gamma = -0.73633
```





ALL FUNCTIONS SUPPORTING THIS CODE

```
% First f(theta) function
function out = f(theta)

    % Platform lengths
    L1 = 2;
    L2 = sqrt(2);
    L3 = sqrt(2);

    % Angle across from L1
    gamma = pi / 2;

    % Strut lengths
    p1 = sqrt(5);
    p2 = sqrt(5);
    p3 = sqrt(5);

    % Strut base positions
    % Got these from Figure 1.15
    x1 = 4;
    x2 = 0;
    y2 = 4;

    A2 = L3 * cos(theta) - x1;
```

```
    B2 = L3 * sin(theta);
    A3 = L2 * (cos(theta) * cos(gamma) - sin(theta) * sin(gamma)) - x2;
    B3 = L2 * (cos(theta) * sin(gamma) + sin(theta) * cos(gamma)) - y2;

    N1 = B3 .* (p2^2 - p1^2- A2.^2 - B2.^2) - B2 .* (p3^2 - p1^2 - A3.^2 -
B3.^2);
    N2 = -A3 .* (p2^2 - p1^2- A2.^2 - B2.^2) + A2 .* (p3^2 - p1^2 - A3.^2 -
B3.^2);

    D = 2 * (A2 .* B3 - B2 .* A3);

    out = N1.^2 + N2.^2 - p1.^2 * D.^2;

end

% f(theta) function with ability to change p2
function out = f_variable_p2(theta, p2)
    L1 = 3; L2 = 3 * sqrt(2); L3 = 3;
    gamma = pi / 4;
    p1 = 5; p3 = 3;
    x1 = 5; x2 = 0; y2 = 6;

    A2 = L3 * cos(theta) - x1;
    B2 = L3 * sin(theta);
    A3 = L2 * (cos(theta) * cos(gamma) - sin(theta) * sin(gamma)) - x2;
    B3 = L2 * (cos(theta) * sin(gamma) + sin(theta) * cos(gamma)) - y2;

    N1 = B3 .* (p2^2 - p1^2 - A2.^2 - B2.^2) - B2 .* (p3^2 - p1^2 - A3.^2 -
B3.^2);
    N2 = -A3 .* (p2^2 - p1^2 - A2.^2 - B2.^2) + A2 .* (p3^2 - p1^2 - A3.^2 -
B3.^2);
    D = 2 * (A2 .* B3 - B2 .* A3);

    out = N1.^2 + N2.^2 - p1.^2 * D.^2;
end

% Forward kinematics problem solver with variable p2
function [x, y] = forward_kinematics_variable_p2(theta, p2)

    % Platform lengths
    L1 = 3;
    L2 = 3 * sqrt(2);
    L3 = 3;

    % Angle across from L1
    gamma = pi / 4;

    % Strut lengths
    p1 = 5;
    p3 = 3;

    % Strut base positions
    x1 = 5;
    x2 = 0;
```

```
    y2 = 6;

    % Compute intermediate terms
    A2 = L3 * cos(theta) - x1;
    B2 = L3 * sin(theta);
    A3 = L2 * (cos(theta) * cos(gamma) - sin(theta) * sin(gamma)) - x2;
    B3 = L2 * (cos(theta) * sin(gamma) + sin(theta) * cos(gamma)) - y2;

    % Numerators and denominator
    N1 = B3 .* (p2^2 - p1^2 - A2.^2 - B2.^2) - B2 .* (p3^2 - p1^2 - A3.^2 -
B3.^2);
    N2 = -A3 .* (p2^2 - p1^2 - A2.^2 - B2.^2) + A2 .* (p3^2 - p1^2 - A3.^2 -
B3.^2);
    D = 2 * (A2 .* B3 - B2 .* A3);

    % Solve for x and y
    x = N1 / D;
    y = N2 / D;

end


function draw_pose(fig_num, x, y, theta, question_number, pose_index)
    % Constants
    L2 = 3 * sqrt(2);
    L3 = 3;
    gamma = pi/4;
    x1 = 5; x2 = 0; y2 = 6;

    % Triangle corner positions
    u1 = x;
    v1 = y;
    u2 = x + L3 * cos(theta);
    v2 = y + L3 * sin(theta);
    u3 = x + L2 * cos(theta + gamma);
    v3 = y + L2 * sin(theta + gamma);

    % Compute strut lengths
    p1 = norm([u1, v1] - [0, 0]);
    p2 = norm([u2, v2] - [x1, 0]);
    p3 = norm([u3, v3] - [x2, y2]);

    % Plot
    figure(fig_num)
    plot([u1 u2 u3 u1], [v1 v2 v3 v1], 'r'); hold on
    plot([0 x1 x2], [0 0 y2], 'bo')
    plot([u1 u2 u3], [v1 v2 v3], 'ro', 'MarkerSize', 8, 'MarkerFaceColor',
'r')
    plot([u1 0], [v1 0], 'k--')     % p1
    plot([u2 x1], [v2 0], 'k--')    % p2
    plot([u3 x2], [v3 y2], 'k--')  % p3

    % Pose label
    title_str = sprintf('Pose %d (\\theta = %.2f)', pose_index, theta);
```

```
    title(title_str)
    xlabel('x')
    ylabel('y')
    axis equal
    grid on

    % Print strut lengths
    fprintf("Pose %d: p1 = %.4f, p2 = %.4f, p3 = %.4f\n", pose_index, p1,
p2, p3);
end

function F = stewart_error(x, B, P, L)
    % Inputs:
    % x = [x; y; z; alpha; beta; gamma]

    pos = x(1:3);          % Translation vector
    alpha = x(4); beta = x(5); gamma = x(6);  % Roll, pitch, yaw

    % Rotation matrix (ZYX order)
    Rz = [cos(gamma) -sin(gamma) 0;
          sin(gamma)  cos(gamma) 0;
          0 0 1];
    Ry = [cos(beta) 0 sin(beta);
          0 1 0;
          -sin(beta) 0 cos(beta)];
    Rx = [1 0 0;
          0 cos(alpha) -sin(alpha);
          0 sin(alpha) cos(alpha)];

    R = Rz * Ry * Rx;

    F = zeros(6,1);
    for i = 1:6
        L_vec = pos + R * P(i,:)' - B(i,:)';
        F(i) = norm(L_vec)^2 - L(i)^2;  % Constraint: length match
    end
end

function visualize_stewart(B, P, pose, ax)
    if nargin < 4
        figure;
        ax = gca;
    end

    % Extract pose
    pos = pose(1:3);
    alpha = pose(4);
    beta  = pose(5);
    gamma = pose(6);

    % Rotation matrix
    Rz = [cos(gamma), -sin(gamma), 0;
```

```
        sin(gamma),  cos(gamma), 0;
        0,           0,          1];
    Ry = [cos(beta), 0, sin(beta);
        0,         1, 0;
       -sin(beta), 0, cos(beta)];
    Rx = [1, 0, 0;
        0, cos(alpha), -sin(alpha);
        0, sin(alpha),  cos(alpha)];
    R = Rz * Ry * Rx;

    % Transform
    P_world = (R * P')' + pos';
    B_loop = [B; B(1,:)];
    P_loop = [P_world; P_world(1,:)];

    % --- Plot in the given axes ---
    cla(ax);
    axes(ax); %#ok<LAXES>
    hold on; grid on; axis equal;
    xlabel('X'); ylabel('Y'); zlabel('Z');
    title('Stewart Platform Visualization');
    view(3);

    % Plot
    plot3(ax, B_loop(:,1), B_loop(:,2), B_loop(:,3), 'bo-', 'LineWidth', 2);
    plot3(ax, P_loop(:,1), P_loop(:,2), P_loop(:,3), 'ro-', 'LineWidth', 2);
    for i = 1:6
        plot3(ax, [B(i,1), P_world(i,1)], ...
                  [B(i,2), P_world(i,2)], ...
                  [B(i,3), P_world(i,3)], 'k--', 'LineWidth', 1.5);
    end

    legend(ax, 'Base', 'Platform', 'Struts')
end


function interactive_stewart(B, P)
    figure('Name', 'Interactive Stewart Platform');

    % Initial pose
    pose = [0; 0; 200; 0; 0; 0];  % [x y z alpha beta gamma]

    % Create sliders for each parameter
    labels = {'X', 'Y', 'Z', 'Roll (α)', 'Pitch (β)', 'Yaw (γ)'};
    mins   = [-100, -100, 150, -pi, -pi, -pi];
    maxs   = [ 100,  100, 300,  pi,  pi,  pi];

    sliders = gobjects(1,6);
    for i = 1:6
        uicontrol('Style', 'text', 'String', labels{i}, ...
            'Position', [20, 400 - 50*i, 80, 20]);
        sliders(i) = uicontrol('Style', 'slider', ...
            'Min', mins(i), 'Max', maxs(i), ...
            'Value', pose(i), ...
```

```
            'Position', [100, 400 - 50*i, 300, 20], ...
            'Callback', @(src, ~) update_plot());
    end

    % Axes for visualization
    ax = axes('Position', [0.5 0.2 0.45 0.7]);
    view(3); grid on; axis equal;
    xlabel('X'); ylabel('Y'); zlabel('Z');
    title('Stewart Platform Pose');

    % Initial plot
    update_plot();

    % Update function
    function update_plot()
        for i = 1:6
            pose(i) = sliders(i).Value;
        end
        cla(ax);
        axes(ax);
        visualize_stewart(B, P, pose, ax);
    end
end
```

```
f(pi/4) = -0.0000000000
f(-pi/4) = -0.0000000000
```

Published with MATLAB® R2024b