Reality Check 1

MATH411

Nathan Lunceford

Reality Check 1

Stewart Platform Overview

The Stewart platform is a mechanical system consisting of six adjustable prismatic joints (struts) that provide six degrees of freedom (three translations and three rotations). It is widely used in applications like robotics, flight simulators, and precision machining, where precise control over the platform's position and orientation is critical. Accurately solving for the platform's position and orientation is vital in these fields to ensure precision and functionality.

In this project, we simplify the platform to a two-dimensional model. This 2D Stewart platform consists of a triangular platform connected to fixed anchor points by three adjustable struts. The objective is to solve the **forward kinematics problem**, where we determine the position (x, y) and orientation θ of the platform, given the lengths of the three struts (p_1, p_2, p_3). These poses or configurations represent valid states of the platform in space.

Unlike the **inverse kinematics problem**, which calculates the strut lengths given a specific platform position, the forward kinematics problem has no closed-form solution due to the complexity of the geometry involved. As a result, we reduce the problem to a single nonlinear equation in θ, denoted as $f(\theta)$. Solving this equation numerically reveals the valid poses (configurations) of the platform, which is essential for its application in robotics and simulations.

Inverse Kinematics Equations

To derive the forward kinematics equations, we start by considering the inverse kinematics of the planar Stewart platform. The system consists of three struts connecting fixed anchor points to the vertices of the triangular platform. The relationship between the platform's configuration and the strut lengths is described by the following equations:

$$p_1^2 = x^2 + y^2$$

$$p_2^2 = (x + A_2)^2 + (y + B_2)^2$$

$$p_3^2 = (x + A_3)^2 + (y + B_3)^2$$

Here, p_1, p_2, and p_3 are the lengths of the struts, and the constants A_2, B_2, A_3, and B_3 are related to the platform's orientation and the fixed anchor points.

Breakdown of Constants

The constants A_2, B_2, A_3, and B_3 are derived from the platform's geometry and orientation:

- $A_2 = L_3 \cos(\theta) - x_1$
- $B_2 = L_3 \sin(\theta)$
- $A_3 = L_2 \cos(\theta + \gamma) - x_2$
- $B_3 = L_2 \sin(\theta + \gamma) - y_2$

Where:

- L_1, L_2, and L_3 are the lengths of the sides of the triangular platform.
- γ is the angle opposite side L_1.
- $(x_1, 0)$ and (x_2, y_2) are the positions of the anchor points.

These constants are necessary for expressing the inverse kinematics problem, which solves for the strut lengths (p_1, p_2, p_3) given the platform's position (x, y, θ). In the forward kinematics problem, we aim to determine (x, y, θ) from the known strut lengths.

From Inverse to Forward Kinematics

To solve the forward kinematics problem, we eliminate the unknowns x and y from the inverse kinematics equations. This involves expanding and rearranging the equations to express x and y in terms of the constants and θ. The final equation for θ is derived as:

$$x = \frac{N_1}{D} = \frac{B_3(p_2^2 - p_1^2 - A_2^2 - B_2^2) - B_2(p_3^2 - p_1^2 - A_3^2 - B_3^2)}{2(A_2 B_3 - B_2 A_3)}$$

$$y = \frac{N_2}{D} = \frac{-A_3(p_2^2 - p_1^2 - A_2^2 - B_2^2) + A_2(p_3^2 - p_1^2 - A_3^2 - B_3^2)}{2(A_2 B_3 - B_2 A_3)}$$

These expressions for x and y hold as long as the denominator $D = 2(A_2 B_3 - B_2 A_3)$ is nonzero. Substituting these values into the first inverse kinematics equation yields the final equation for θ:

$$f(\theta) = N_1^2 + N_2^2 - p_1^2 D^2 = 0$$

Here, N_1 and N_2 are intermediate expressions based on the system of equations. Solving this equation gives the valid values of θ corresponding to the platform's possible poses.

Problem 1

Objective: Write a python function for f(θ). The parameters $L_1, L_2, L_3, \gamma, x_1, x_2, y_2$ are fixed constants, and the strut lengths p_1, p_2, p_3 will be known for a given pose. To test your code, set the parameters $L_1 = 2$, $L_2 = L_3 = \sqrt{2}$, $\gamma = \pi/2$, and $p_1 = p_2 = p_3 = \sqrt{5}$. Then, substituting $\theta = -\pi/4$ or $\theta = \pi/4$, should make $f(\theta) = 0$.

Solution Process:

1. **Mathematical Approach**:
    - The function $f(\theta)$ relates the orientation angle θ and the strut lengths of the platform. It is derived using trigonometric relationships between the platform's geometry and anchor points.
    - The goal is to simplify these relationships into a single equation for $f(\theta)$, which can be solved to find valid platform poses.
2. **Implementation in Code**:
    - I implemented the function $f(\theta)$ to calculate the value based on given constants and strut lengths. I tested the function using specific values for θ, verifying that the results were consistent with expected configurations where $f(\theta) = 0$.
    - The test confirmed that the function correctly represents the forward kinematics of the Stewart platform, with results near zero indicating valid solutions.

Results:

For the test values of $\theta = \pm\frac{\pi}{4}$, the function returned:

$$f\left(\pm\frac{\pi}{4}\right) = -4.547473508864641 \times 10^{-13}$$

This value is effectively zero, within the limits of numerical precision. This confirms that the function $f(\theta)$ works correctly and that $\theta = \pm\frac{\pi}{4}$ are valid solutions for the given platform configuration.

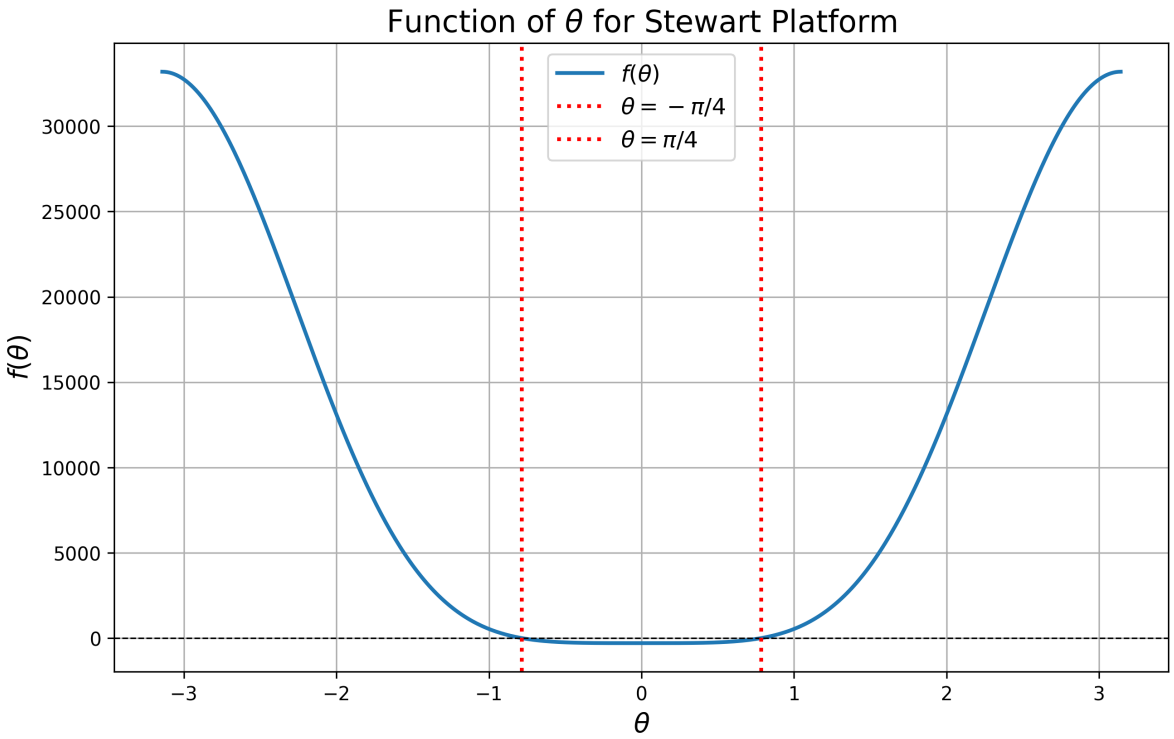
Problem 2

Objective: Plot $f(\theta)$ over the interval $[-\pi, \pi]$.

Solution Process:

1. **Mathematical Approach**:
    - Visualizing the function $f(\theta)$ is essential to understanding the behavior of the Stewart platform. The roots of this function correspond to valid configurations (poses) of the platform, where $f(\theta) = 0$. Identifying these roots graphically helps to confirm the expected solutions and provides insight into how the platform behaves across different angles θ.
2. **Implementation in Code**:
    - I first generated a range of θ values between $-\pi$ and π using `np.linspace()` to create 400 evenly spaced points. These points provide a sufficiently detailed view of the function's behavior without causing excessive computational overhead.
    - For each value of θ, I computed $f(\theta)$ by passing it through the previously defined function. This gave me an array of function values corresponding to the array of θ values.
    - I then plotted the function using `matplotlib`, where the x-axis represents θ and the y-axis represents the value of $f(\theta)$. I also included horizontal and vertical reference lines to highlight where $f(\theta) = 0$ and the specific points where $\theta = \pm\pi/4$, as identified in **Problem 1**.
    - The resulting plot clearly shows the roots of the function, which align with the values found in the previous problem. These roots represent the valid poses of the platform, where the lengths of the struts and the angle θ satisfy the system of equations.

Results:

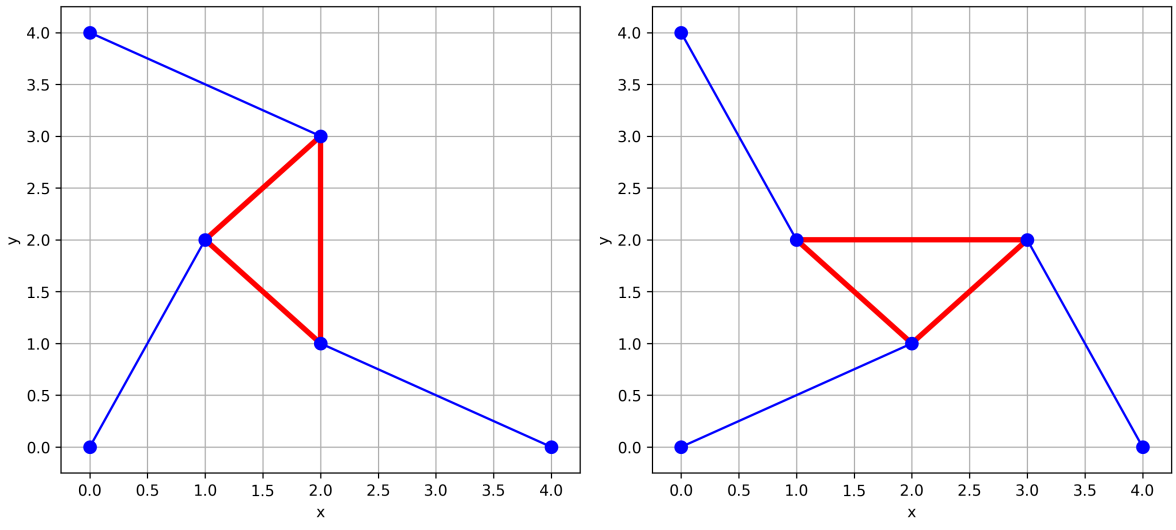

Function of θ for Stewart Platform
$f(\theta)$
$\theta = -\pi/4$
$\theta = \pi/4$
30000
25000
20000
15000
10000
5000
0
$f(\theta)$
−3
−2
−1
0
1
2
3
θ

Problem 3

Objective: Reproduce a figure of the platform with a triangle representing the platform and small circles for the anchor points.

1. **Implementation in Code**:
    - I developed a helper function `get_x_y()` to calculate the x and y coordinates of one vertex of the platform based on the angle θ and the provided constants. This function applies the trigonometric relationships derived from the platform's geometry.
    - After determining the position of this key vertex, I used another helper function, `get_points()`, to compute the remaining two vertices of the triangular platform. This function takes the initial vertex, the angle θ, and the constants (such as the strut lengths) to determine the other two points of the triangle.
    - The `get_anchor_points()` function returns the fixed positions of the anchor points, which define the platform's fixed geometry and are essential for calculating the platform's overall configuration.
    - To visualize the platform, I implemented the `plot_triangle()` function, which plots the triangle's vertices and anchor points. Blue lines connect the anchor points to the vertices, while red lines highlight the edges of the triangle.
    - Using these functions, I generated two plots: one for $\theta = \pi/4$ and another for $\theta = -\pi/4$, showing the two distinct configurations of the Stewart platform.

Results:

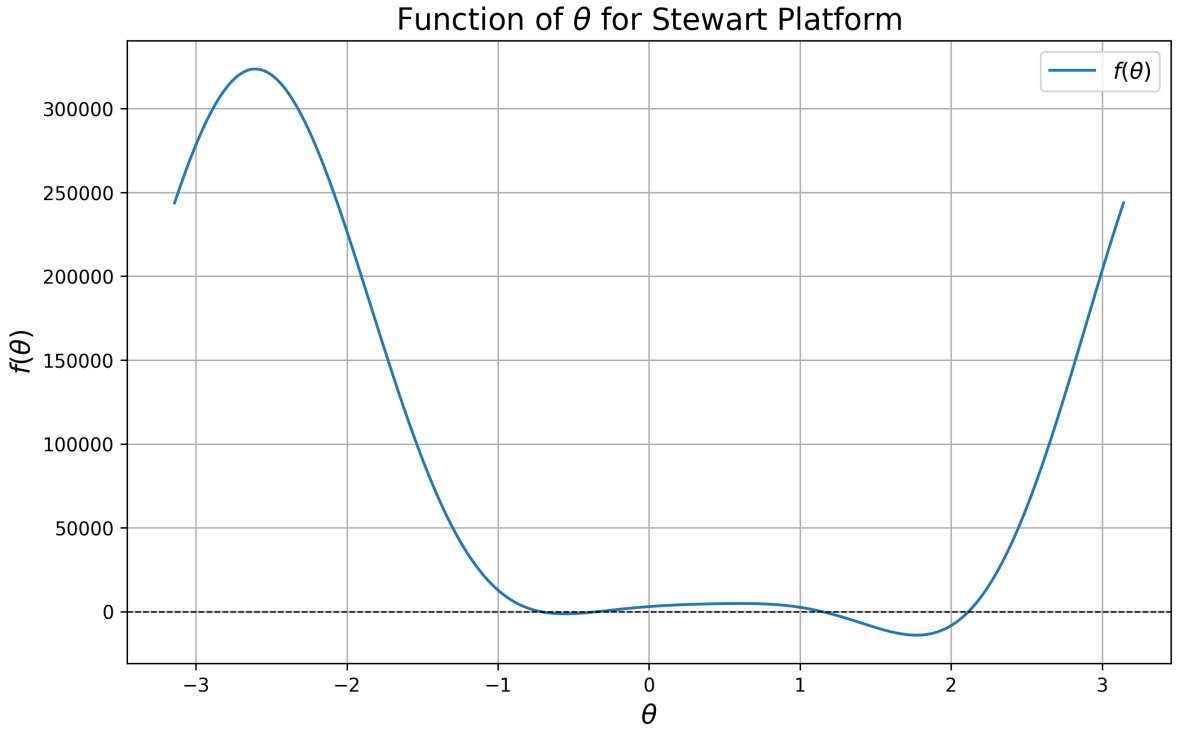


Problem 4

Objective: Solve the forward kinematics problem for the planar Stewart platform specified by $x_1 = 5$, $(x_2, y_2) = (0, 6)$, $L_1 = L_3 = 3$, $L_2 = 3\sqrt{2}$, $\gamma = \pi/4$, $p_1 = p_2 = 5$, $p_3 = 3$.

Solution Process:

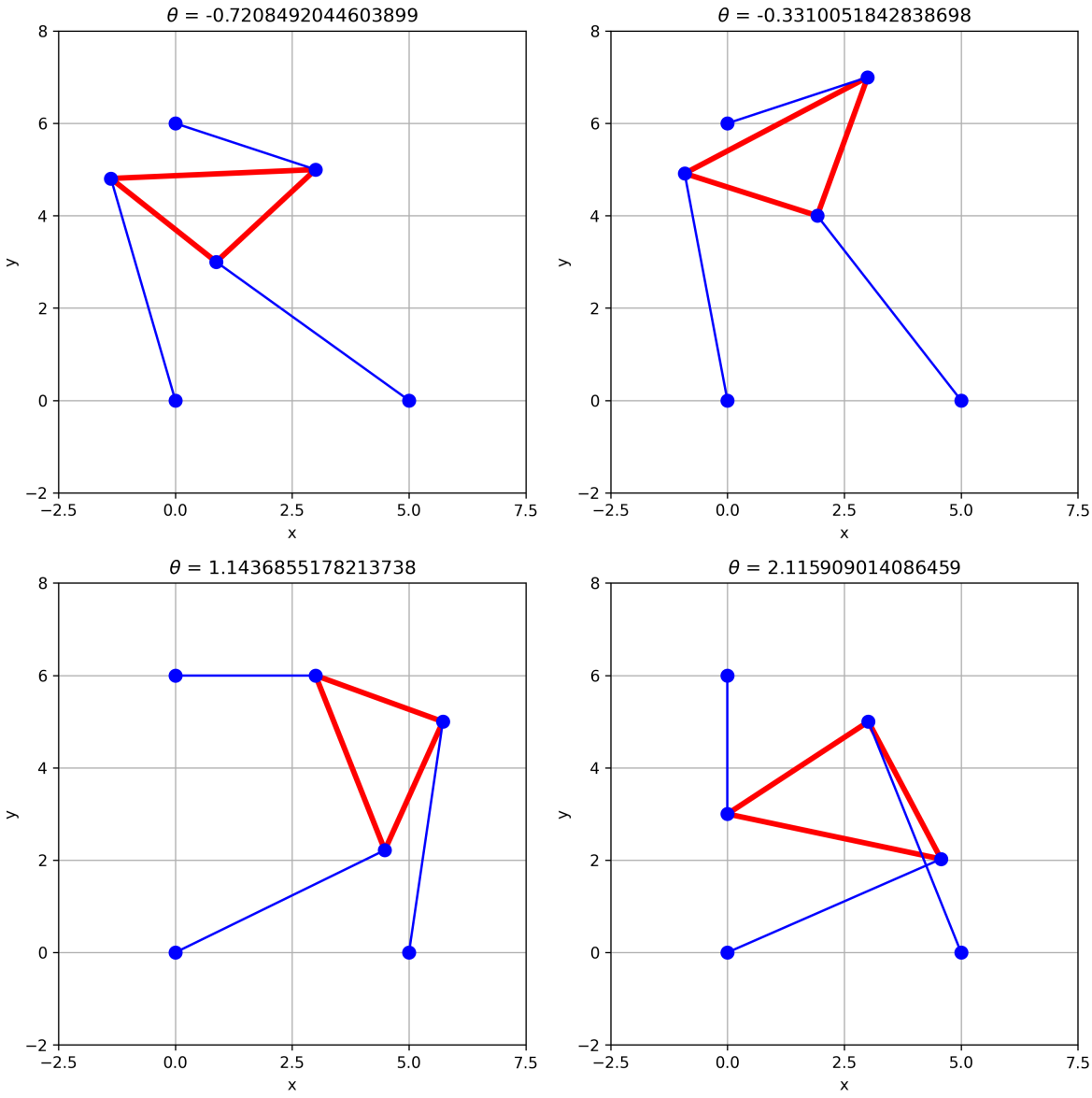
1. **Mathematical Approach**:
    - The goal was to find the roots of $f(\theta)$ to determine the platform's valid poses based on the given parameters. Each root of the equation corresponds to a distinct configuration (pose) of the Stewart platform.
    - I calculated the platform's position (x, y) and orientation θ for each root, which represents a valid solution for the platform's configuration.



2. **Implementation in Code**:
    - I solved the function $f(\theta)$ numerically using `fsolve()` and an array of initial guesses to ensure that all valid roots were captured. The `find_roots()` function I created handled this process by iterating over each guess to find the unique solutions where $f(\theta) = 0$. This method successfully identified four distinct roots, corresponding to valid poses of the platform.

- To ensure the correctness of the computed configurations, I implemented a validation step. I verified that the calculated lengths of the struts p_1, p_2, and p_3 matched the expected values provided in the problem. This was done by calculating the Euclidean distances between the platform's vertices and their respective anchor points, corresponding to the strut lengths.

Results:



As shown in the plot above, I found four distinct roots corresponding to four valid poses of the Stewart platform. For each root, I plotted the platform's triangular configuration, confirming the geometry and strut lengths.

Here are the identified roots and the corresponding strut lengths:

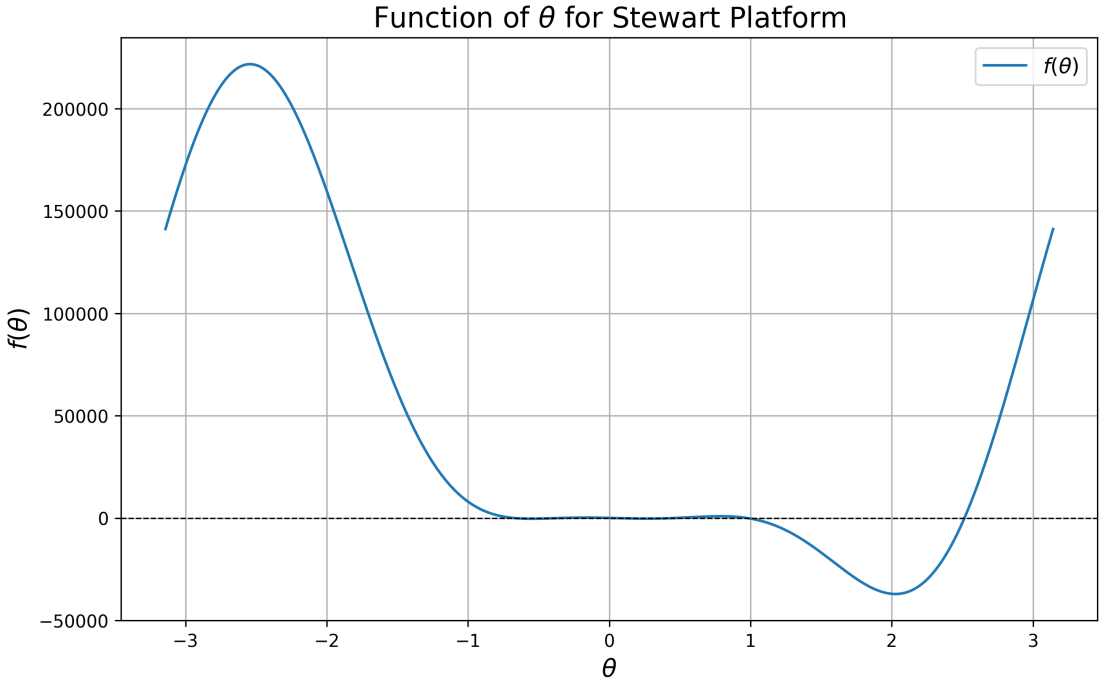
- Root 1: $\theta = -0.72084920$, strut lengths = [5, 5, 3]
- Root 2: $\theta = -0.33100518$, strut lengths = [5, 5, 3]
- Root 3: $\theta = 1.14368551$, strut lengths = [5, 5, 3]
- Root 4: $\theta = 2.11590901$, strut lengths = [5, 5, 3]

Problem 5

Objective: Modify the strut length $p_2 = 7$ and re-solve the problem to find six distinct poses.

Solution Process:

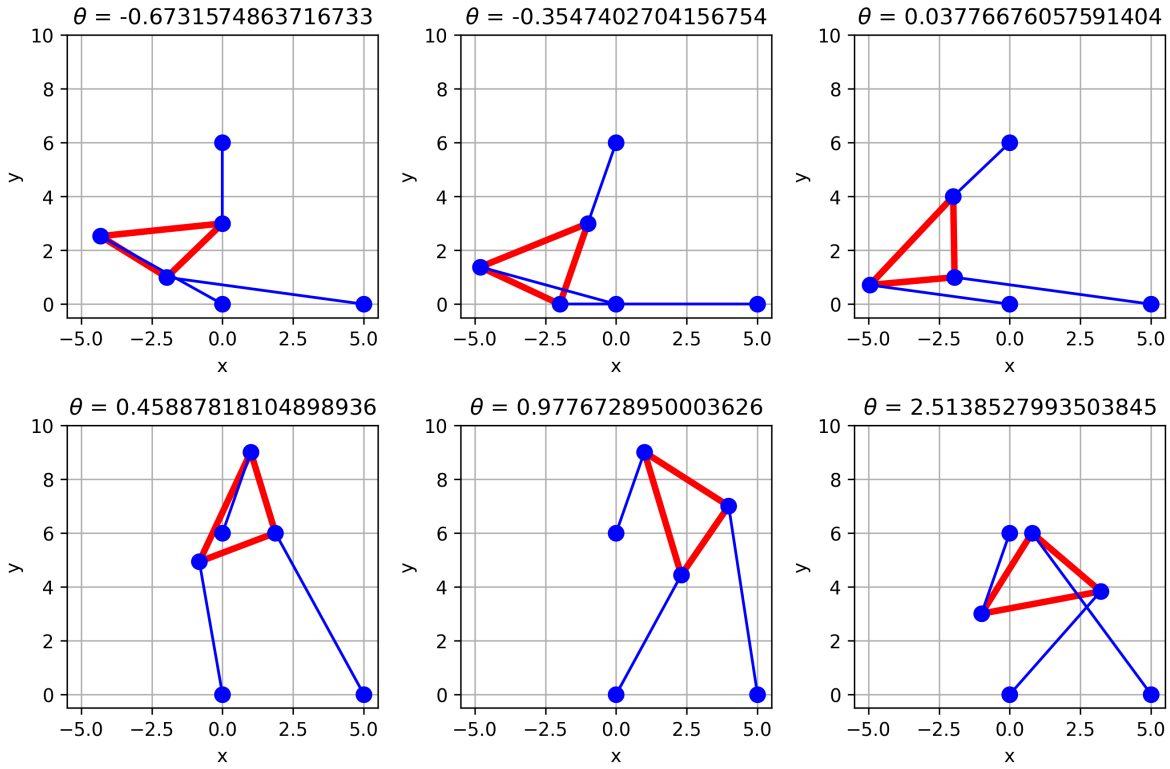
1. **Mathematical Approach**:
    - By changing the strut length $p_2 = 7$, the function $f(\theta)$ and the number of possible poses of the platform are altered. Modifying this parameter can introduce additional roots, corresponding to new configurations of the Stewart platform.
    - The goal here was to find the six possible poses by solving $f(\theta)$ for the modified system. Each root of the function $f(\theta) = 0$ corresponds to a valid configuration of the platform.



2. **Implementation in Code**:
    - After modifying the constants to reflect $p_2 = 7$, I updated the function $f(\theta)$ and plotted it over the interval $[-\pi, \pi]$ to visualize its behavior.

- Using an updated set of initial guesses for the `find_roots()` function, I identified six distinct roots. These roots represent six unique poses of the Stewart platform for the modified strut length.
- For each root, I calculated and verified the corresponding configurations by plotting the triangle configurations of the platform and confirming that the strut lengths p_1, p_2, p_3 matched the expected values.

Results:



As shown, I found the following six roots of $f(\theta)$, corresponding to six distinct poses of the Stewart platform:

- Root 1: $\theta = -0.67315749$, strut lengths = [5, 7, 3]
- Root 2: $\theta = -0.35474027$, strut lengths = [5, 7, 3]
- Root 3: $\theta = 0.03776676$, strut lengths = [5, 7, 3]
- Root 4: $\theta = 0.45887818$, strut lengths = [5, 7, 3]
- Root 5: $\theta = 0.97767289$, strut lengths = [5, 7, 3]
- Root 6: $\theta = 2.51385280$, strut lengths = [5, 7, 3]

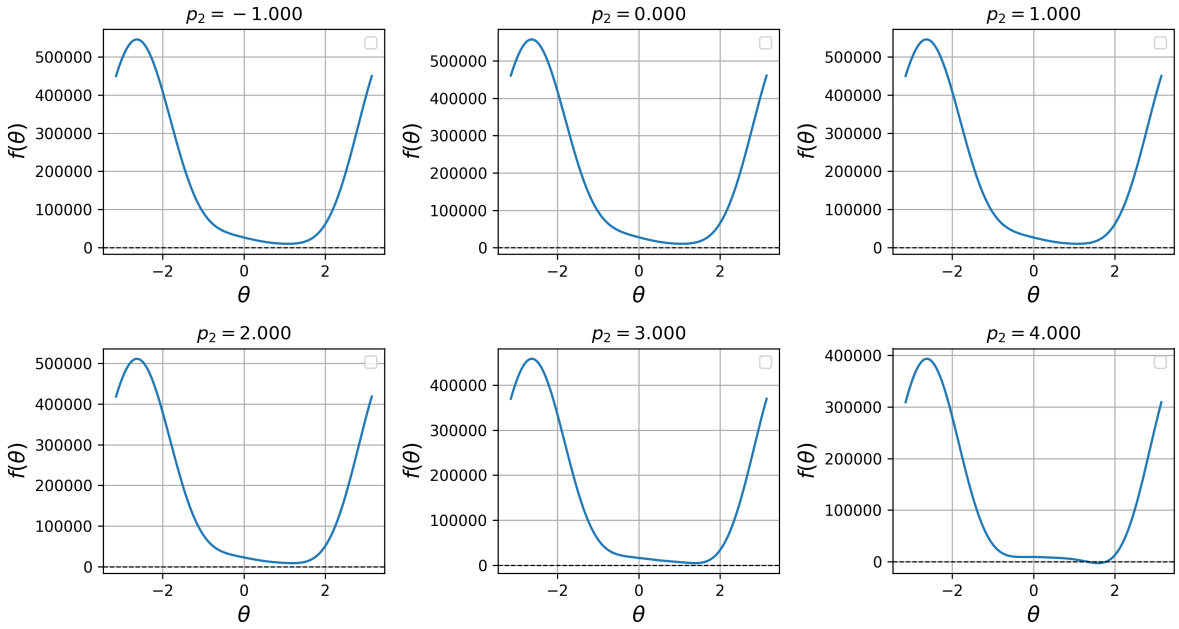
Problem 6

Objective: Find a strut length p_2 such that the platform has exactly two poses.

Solution Process:

1. **Mathematical Approach**:
    - To identify the value of p_2 that results in exactly two poses, I iteratively adjusted p_2 and solved the corresponding $f(\theta) = 0$. The goal was to determine when the number of valid solutions (poses) reduced to exactly two, indicating a unique configuration of the platform with only two possible positions.
2. **Implementation in Code**:
    - I used the `fsolve()` method with predefined initial guesses to identify the roots of $f(\theta)$. This method relies on initial guesses for the root-finding process. While fsolve can be effective, its success depends on the accuracy of these guesses. Therefore, I systematically tested different values of p_2 to check whether exactly two valid roots could be found.
    - After verifying the results through both numerical methods and graphical plotting, I confirmed that $p_2 = 4.000$ produced exactly two distinct roots.

Results:



As seen in the plots, for $p_2 = 4.000$, two valid roots were found at $\theta = 1.331642$ and $\theta = 1.777514$. The plot clearly shows the corresponding behavior of $f(\theta)$ for various values of p_2. Prior to $p_2 = 4.000$, there were no valid poses, and as p_2 increased, the two roots emerged, confirming that this was the correct value for the two-pose configuration.

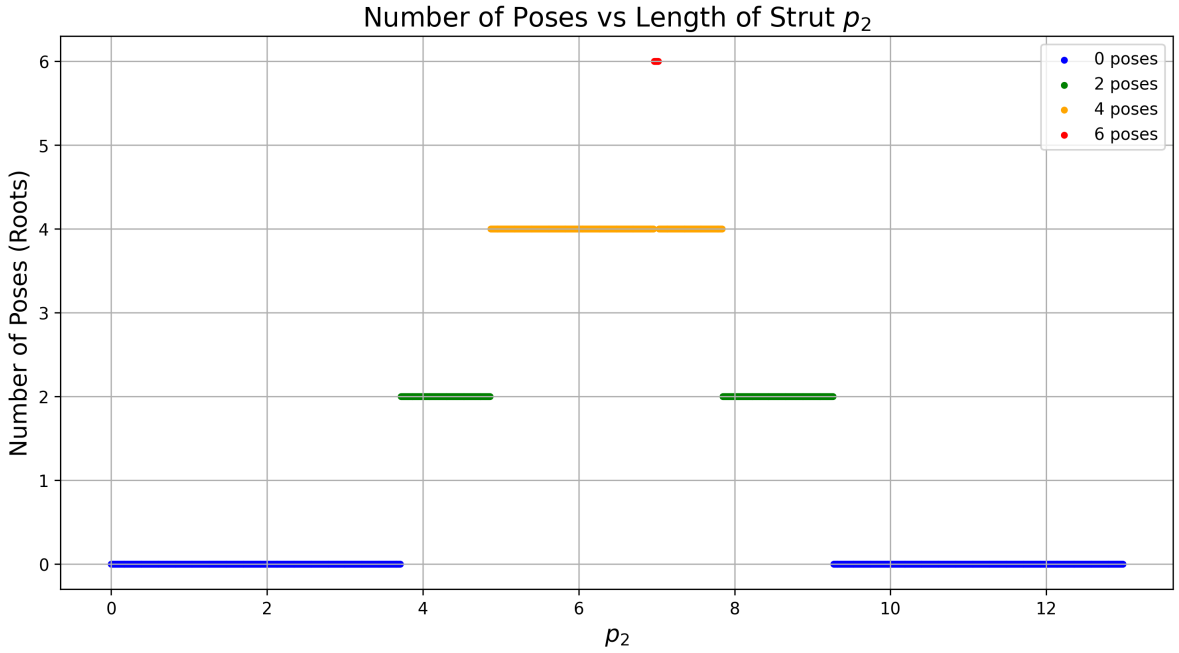
Problem 7

Objective: Calculate the intervals in p_2 where the platform has 0, 2, 4, or 6 poses.

Solution Process:

1. **Mathematical Approach**:
    - The number of valid poses changes as p_2 varies, and the goal was to determine the intervals of p_2 where the platform has different numbers of valid poses (roots).
    - By solving $f(\theta)$ for a range of p_2 values, I classified the intervals based on the number of valid solutions. Each solution corresponds to a possible configuration (pose) of the platform.
2. **Refinement of the Numerical Method**:
    - In transitioning from **Problem 6** to **Problem 7**, I realized that using `fsolve` with predefined initial guesses became unreliable. It often missed roots or identified duplicate roots due to its sensitivity to the starting points provided. This was problematic for accurately counting the number of valid poses, especially as the value of p_2 changed.
    - To overcome this, I adopted a more robust approach using the **Brent's method** (`brentq`). This method detects sign changes in $f(\theta)$ and efficiently locates roots where the function transitions from positive to negative or vice versa. This strategy improved root detection by ensuring that I captured all the valid roots without introducing duplicates or missing any solutions.
    - The precision of this method allowed me to accurately count the number of poses for each p_2 value, giving a much clearer picture of how the platform's configurations evolve as the strut length changes.
3. **Implementation in Code**:
    - I implemented a loop that systematically adjusted p_2 in small increments and employed `brentq` to find the corresponding roots of $f(\theta)$. For each value of p_2, the number of valid roots was counted, and the results were classified into intervals where the platform had 0, 2, 4, or 6 poses.
    - This method provided a reliable classification of the intervals and improved the accuracy of the results, which were then plotted for visual clarity.

Results:



Based on the computations, the following intervals for p_2 and the corresponding number of poses were identified:

- **Intervals with 0 poses:**
  - p_2 from 0.00 to 3.71
  - p_2 from 9.27 to infinity
- **Intervals with 2 poses:**
  - p_2 from 3.72 to 4.86
  - p_2 from 7.85 to 9.26
- **Intervals with 4 poses:**
  - p_2 from 4.87 to 6.96
  - p_2 from 7.03 to 7.84
- **Intervals with 6 poses:**
  - p_2 from 6.97 to 7.02
