Project No. 1
"Reality Check 1, Numerical Analysis by Timothy Sauer"
(3rd Edition, Pages 70-73)

Kinematics of the Stewart Platform

October 22, 2022

Submitted by:
Liteshwar Rao

Numerical Methods, AI-5003
Topics in Scientific Computing, MAT-5402
Fall 2022
Yeshiva University

Contents

1 Abstract

This project explores two numerical methods of finding roots (x) of non-linear equations which satisfy the equation $f(x) = 0$. We use Bisection and Newton-Raphson methods to find roots of the given equation , and we state reasons why we prefer one method over another. We also discuss briefly, why we are not using Fixed-point iteration method to solve the given problem. This project uses the non-linear equations derived for *Stewart Platform* titled as *Kinematics of the Stewart Platform* presented in Chapter 1 of the book by *Timothy Sauer, 3rd edition*. We see that we can get approximate roots with fewer iterations if we are able to plot the function and find the intervals containing true roots.

2 Introduction[1]

A Stewart platform consists of six variable length struts, or prismatic joints, supporting a payload. Prismatic joints operate by changing the length of the strut, usually pneumatically or hydraulically. As a six-degree-of-freedom robot, the Stewart platform can be placed at any point and inclination in three-dimensional space that is within its reach. To simplify matters, the project concerns a two-dimensional version of the Stewart platform. It will model a manipulator composed of a triangular platform in a fixed plane controlled by three struts, as shown in Figure 1.14. The inner triangle represents

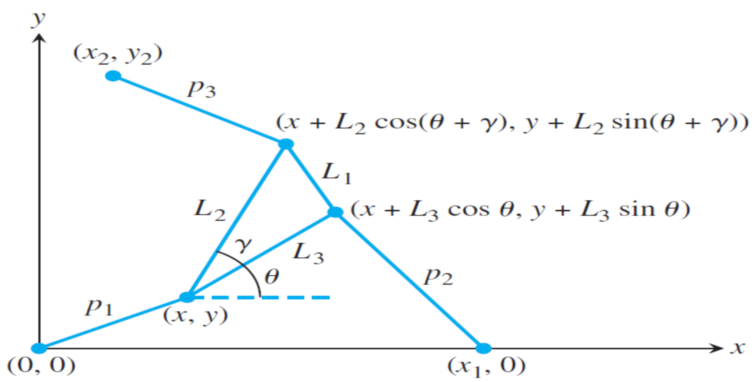


Figure 1.14 Schematic of planar stewart platform.

the planar Stewart platform whose dimensions are defined by the three lengths L_1, L_2 and L_3. Let γ denote the angle across from side L_1. The position of the platform is controlled by the three numbers p_1, p_2, and p_3, the variable lengths of the three struts. Finding the position of the platform, given the three strut lengths, is called the forward, or direct, kinematics problem for this manipulator. Namely, the problem is to compute (x, y) and θ for each given p_1, p_2, p_3.

2.1 Equations to be solved

We need to solve for (x, y) and θ by using following equations for given values of $x_1, x_2, y_2, L_1, L_2, L_3, p_1, p_2, p_3, \gamma$.

$$A_2 = L_3 \cos(\theta) - x_1 \tag{1}$$
$$B_2 = L_3 \sin(\theta) \tag{2}$$
$$A_3 = L_2 \cos(\gamma + \theta) - x_2 \tag{3}$$
$$B_3 = L_2 \sin(\gamma + \theta) - y_2 \tag{4}$$
$$D = 2(A_2 B_3 - B_2 A_3) \tag{5}$$
$$N_1 = B_3(p_2^2 - p_1^2 - A_2^2 - B_2^2) - B_2(p_3^2 - p_1^2 - A_3^2 - B_3^2) \tag{6}$$
$$N_2 = -A_3(p_2^2 - p_1^2 - A_2^2 - B_2^2) + A_2(p_3^2 - p_1^2 - A_3^2 - B_3^2) \tag{7}$$
$$x = \frac{N_1}{D} \tag{8}$$

[1]Reality Check 1, Kinematics of the Stewart Platform, page 70, Timothy Sauer, 3rd edition

$$y = \frac{N_2}{D} \tag{9}$$

$$f = N_1^2 + N_2^2 - p_1^2 D^2 = 0 \tag{10}$$

2.2 Finding roots

From the equations $1, 2, \ldots, 10$, above, we can write the non-linear equation $f(\theta)$ to be solve for θ such that $f(\theta) = 0$. We need to find roots (θ) for the following equation:

$$\begin{aligned} f(\theta) = &-p_1^2(-2L_3(L_2\cos(\gamma+\theta) - x_2)\sin(\theta) + 2(L_2\sin(\gamma+\theta) - y_2)(L_3\cos(\theta) - x_1))^2 + \\ &((-L_2\cos(\gamma+\theta) + x_2)\left(-L_3^2\sin^2(\theta) - p_1^2 + p_2^2 - (L_3\cos(\theta) - x_1)^2\right) + (L_3\cos(\theta) - x_1) \\ &\left(-p_1^2 + p_3^2 - (L_2\sin(\gamma+\theta) - y_2)^2 - (L_2\cos(\gamma+\theta) - x_2)^2\right))^2 + \\ &(-L_3\left(-p_1^2 + p_3^2 - (L_2\sin(\gamma+\theta) - y_2)^2 - (L_2\cos(\gamma+\theta) - x_2)^2\right)\sin(\theta) \\ &+ (L_2\sin(\gamma+\theta) - y_2)\left(-L_3^2\sin^2(\theta) - p_1^2 + p_2^2 - (L_3\cos(\theta) - x_1)^2\right))^2 \end{aligned} \tag{11}$$

Note: Since $f(\theta)$ is a polynomial in $\sin(\theta)$ and $\cos(\theta)$, so, for any given root (θ) there exist other roots in the form of $(\theta + 2\pi k)$ leading to same solutions. Therefore, we are restricting our domain for (θ) such that $\theta \in [-\pi, \pi]$.

3 Question 1

3.1 $\theta = -\pi/4$

From the figure 1.15 (a), we see that value of $x_1 = 4, x_2 = 0, y_2 = 4$. The other parameters ($L_1 = 2, L_2 = L_3 = \sqrt{2}, \gamma = \pi/2, p_1 = p_2 = p_3 = \sqrt{5}$) are given in the question. By putting these values along with $\theta = -\pi/4$ in equation (11) above, we get the value of $f(\theta)$.

3.2 $\theta = \pi/4$

Similarly, from the figure 1.15 (b), we see that value of $x_1 = 4, x_2 = 0, y_2 = 4$.We derive the another value of $f(\theta)$ from equation (11) with the same value of each parameters as in subsection (3.1) but with $\theta = \pi/4$.

3.3 *Results*

Results of the calculations are presented in the following table #1.

Table 1: Values of $f(\theta), x, y$ for two different values of θ

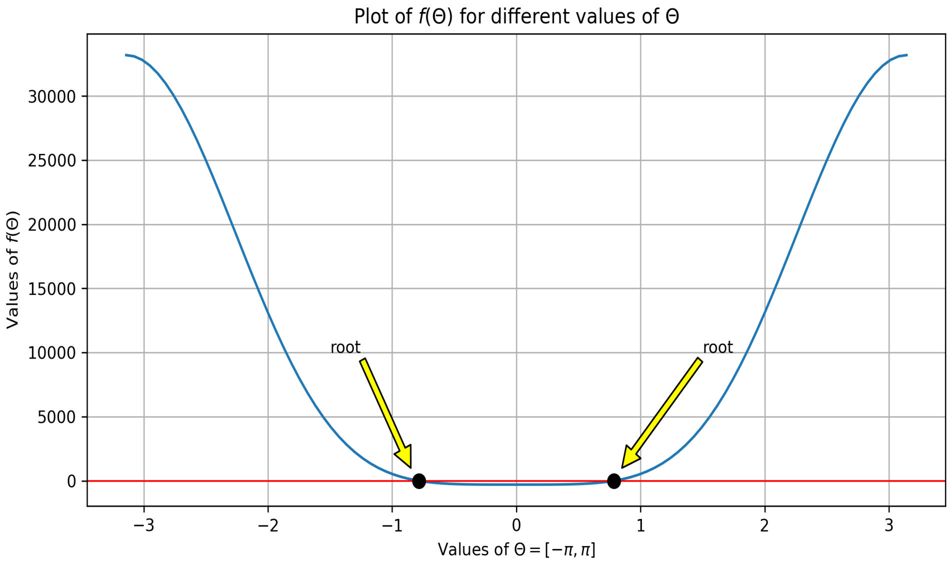
Sr. No.	θ	$f(\theta)$	x	y
1	$-\pi/4$	$-4.54747350e-13$	1	2
2	$\pi/4$	$-4.54747350e-13$	2	1

The value of $f(\theta) \approx 0$ when $\theta = -\pi/4, \pi/4$.

4 Question 2

The plot of $f(\theta)$ for a sequence of values of θ is shown below.

Plot 1 function $f(\theta)$



Numerically, we have seen in results of question 1 above that at $\theta = -\pi/4, \pi/4$ this $f(\theta) \approx 0$. From the plot above, we can see and confirm that at *blackdots* the value of $f(\theta) \approx 0$. Moreover, the value of blackdots representing $\theta = -\pi/4 = -0.78539816339744483$ and $\theta = \pi/4 = 0.78539816339744483$ can be seen in the above plot.

We check our results by solving for struts length P_1, P_2, and P_3 for $\theta = \pi/4$ and $\theta = -\pi/4$ with the corresponding x and y values $(2, 1)$ and $(1, 2)$ respectively. We see that all three struts lengths are approximately equal to $\sqrt{5}$.

5 Question 3

To reproduce the figure 1.15, we need to find $x, y, x_1, y_2, x_2, p_1, p_2, p_3, \theta, \gamma$. We are given $p_1 = p_2 = p_3 = \sqrt{5}, L_1 = 2, L_2 = L_3 = \sqrt{2}$, and $\gamma = \pi/2$. From the figures 1.15 (a) we can see that value of $x_1 = 4, x_2 = 0, y_2 = 4, x = 1, y = 2$ and similarly, from figure 1.15 (b) it can be seen $x_1 = 4, x_2 = 0, y_2 = 4, x = 2, y = 1$.

5.1 Calculations of θ for figure 1.15 (a) and 1.15 (b)

Zooming in on inner triangles in figure 1.15(a) and 1.15(b) to find θ, we get following two plots.

For figure 1.15 (a_1) we are using the trigonometric identity to find angle θ made by L_3 with $x - axis$. Drawing a perpendicular from $C = (1, 2)$ on L_1 will bisect the length of L_1 in two equal parts since angle opposite to L_1 $= \angle ACB = \gamma = \pi/2$. Length of perpendicular $CD = 1$ and length of $AD = 1$. So, we get $\tan(\theta) = 1/1$, which in turn gives $\theta = \tan^{-1}(1)$. This is equal to $\theta = -\pi/4$ since L_3 moves counterclockwise to make angle θ with $x - axis$. By following the similar approach for figure 1.15(b_1), we get $\theta = \pi/4$ because here L_3 moves clockwise to make an angle with $x - axis$.

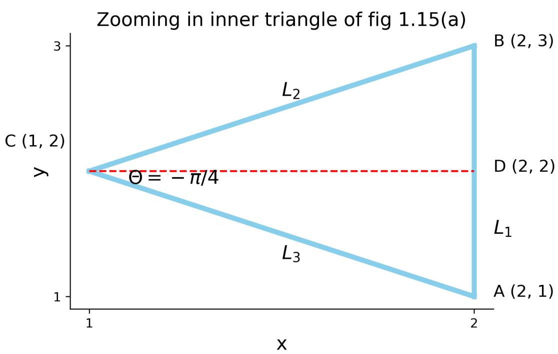


Figure 1.15 (a_1)

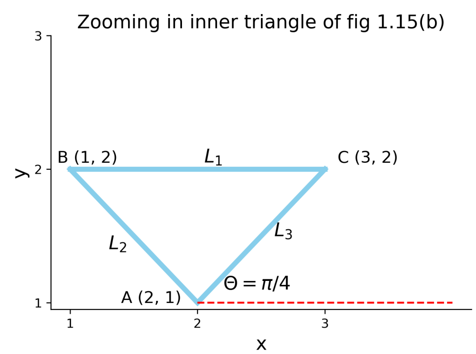
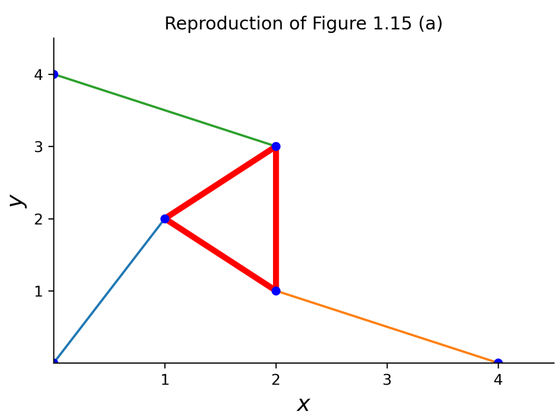
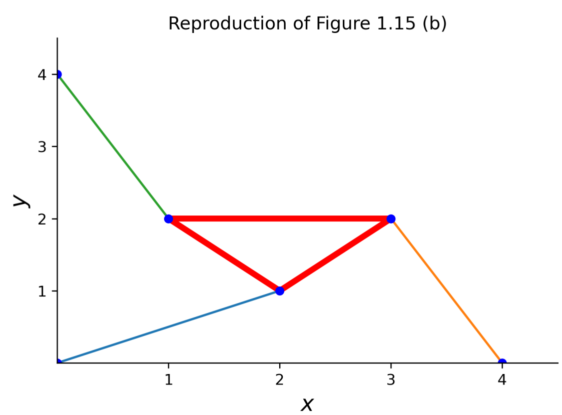


Figure 1.15 (b_1)

Zooming in inner triangles of figure 1.15

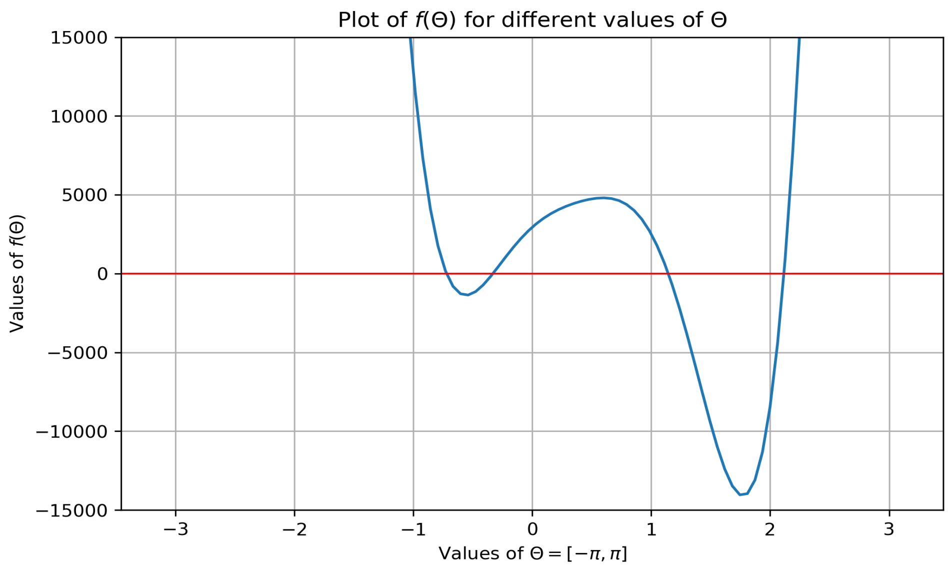
Now, to reproduce figure 1.15 (a) and (b), we have values for each parameter.





6 Question 4

We begin by plotting $f(\theta)$ to find the intervals, if exist, containing true roots.



From the plot above, we can say that there are four true roots of the function $f(\theta)$. We have three methods to find approximate roots, i.e. Bisection method, Fixed-point iteration method, and Newton-Raphson method.

6.1 Analysis of Fixed-point iteration method

Among these three methods, we are not using Fixed-point iteration method because of its computational complexity. We are going to use θ as a parameter to relate the general form of this method to the problem in hand. From equation (11), we know our main task is to solve for θ such that $f(\theta) = 0$. So, we have now *equation (11)=* $f(\theta)$ *=0*. We need to perform algebraic manipulation in such a way that we can express θ as a $f(\theta)$. We can do this in many different ways. First, we can keep $\sin(\theta)$ on L.H.S and move all other functional form, say $g(\theta)$, of the equation to R.H.S. followed by $\theta = \sin^{-1}(g(\theta))$. Then, its general form is given by:

$$\theta_{i+1} = g(\theta_i)$$

We chose initial value denoted by θ_0, put it in $g(\theta_0)$ and then check if $g'(\theta_0)$ is less than one. If it is, solve for θ_1 by using its general form such as $\theta_1 = g(\theta_0)$. After getting θ_1, we put back θ_1 in the general form and solve for θ_2. This process keeps continuing until we apply some stopping criteria based on predefined tolerance level. One thing note here, is that there is no unique method to find $\theta = g(\theta)$. We can choose $\cos(\theta)$ to be on L.H.S and all other functional form on R.H.S. and still get $\theta = g(\theta)$. Further, we can do any algebraic manipulation to equation (11) to get $\theta = g(\theta)$. This process can be implemented easily if $f(\theta)$ contains fewer number of parameters. However, the equation (11) contains so many functional forms that contains θ that we can use as a general form and perform iteration to get approximate roots. Moreover, we need to check the first derivative of all such functional forms to ensure it is less than one for the chosen initial value of θ. Despite all such functional forms and their corresponding first derivatives, we will have linear rate of convergence given by the following equation where ε_{k+1} denotes the error at $(k+1)^{th}$ iteration and ε_1 denotes the error at 1^{st} iteration. Thus, we have decided to keep this method aside for later use if other two methods do not give desired results.

$$\varepsilon_{k+1} = |g'(\xi)|^k.\varepsilon_1$$

6.2 Analysis of Bisection and Newton-Raphson methods

We used Bisection and Newton-Raphson methods to find approximate roots of equation (11) with assumed tolerance level as 10^{-8} defined as difference between $(\theta_{i+1} - \theta_i) < 10^{-8}$ and maximum number of iterations to achieve this level of accuracy is 100. These two conditions are defined as stopping criteria for the algorithm. We are using these two methods because they have unique functional form and apriori we know the intervals containing true roots. These two methods work well if we choose our initial conditions/roots close to the true roots. Because of plotting of $f(\theta)$ we can choose our initial conditions/roots easily to ensure we get approximate roots converging to true roots. The results of these two methods are below:

Table 2: Initial conditions, roots, iteration no., time taken

Bisection method				Newton-Raphson		
Iteration No.	θ	a	b	Iteration No.	θ	a
26	-0.7208492085337639	-1	-0.5	5	-0.720849204460389	-0.8
26	-0.331005 17839193344	-0.5	0	4	-0.331005 18428387	-0.4
25	1.143685 5122447014	1	1.25	4	1.143685 51782137	1.2
25	2.115909 017622471	2	2.25	4	2.115909 01408646	2.1
time taken		0.000403		**time taken**		0.000542

Table 3: Values of x, y, θ for each pose (from **In [19]**)

Bisection method		Newton-Raphson			
Iteration no.	θ	Iteration no.	θ	x	y
26	-0.720849 2085337639	5	-0.720849 204460389	-1.378379 630597699	4.806253 176222966
26	-0.331005 17839193344	4	-0.331005 18428387	-0.914708 7168343394	4.915618 777259611
25	1.143685 5122447014	4	1.143685 51782137	4.481750 065399037	2.216735 5167669114
25	2.115909 017622471	4	2.115909 01408646	4.571830 175332456	2.024442 8487659563

We can see that Newton-Raphson method is more efficient than Bisection method. Although, the time taken to find all four roots by each method is approximately the same (between 0.1 and 1 ms), we get the desired level of accuracy (tolerance level) within the maximum allowed iterations (100) with fewer iterations in Newton's method. Therefore, we have decided to implement Newton-Raphson method for the following exercises.

We check our results by following inverse kinematics problem of the planar Stewart platform which is to find p_1, p_2, p_3 for the given x, y, θ. Since, we have now x, y, θ from Newton-Raphson method in table 3 above and $x_1, x_2, y_2, L_2, L_3, \gamma$ given in the question, we simply plug these values in equation nos. 1,2,3 to get A_2, B_2, A_3, B_3.

Then we calculate p_1, p_2, p_3 by the following equation:

$$p_1^2 = x^2 + y^2 \tag{12}$$
$$p_2^2 = (x + A_2)^2 + (y + B_2)^2 \tag{13}$$
$$p_3^2 = (x + A_3)^2 + (y + B_3)^2 \tag{14}$$

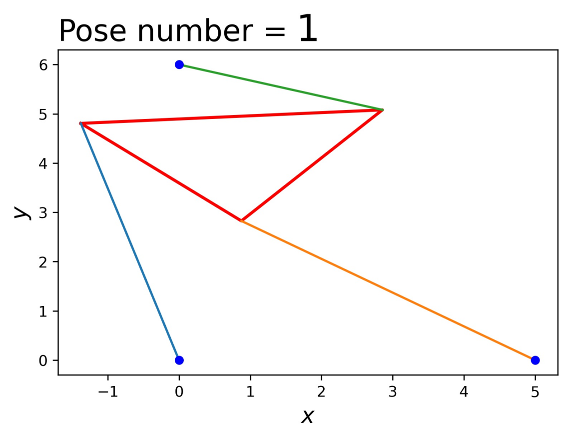
The results of our calculations are below:

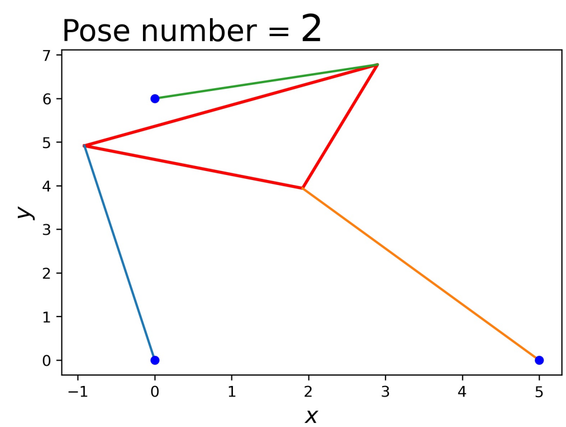
Table 4: Testing of results obtained from Solver

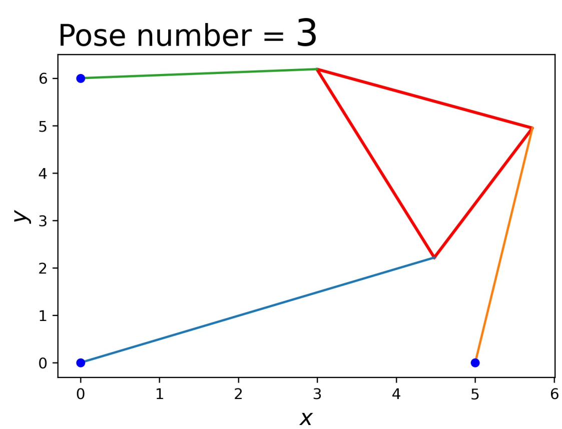
θ	p_1	p_2	p_3
-0.720849204460389	5	5	3
-0.331005 18428387	5	5	3.0000000000000004
1.143685 51782137	5	5	3.0000000000000004
2.115909 01408646	4.999999999999998	5	2.9999999999999973

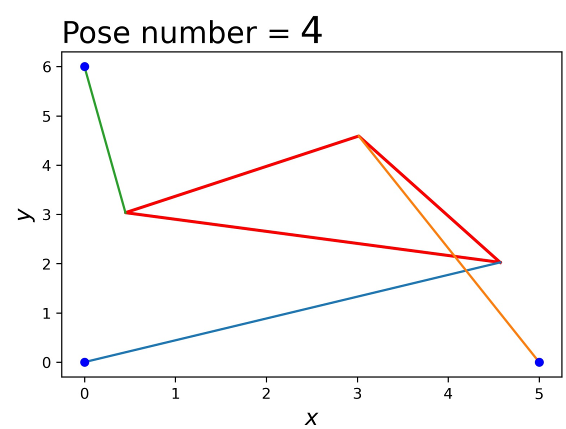
We are satisfied with these results since the error, defined as $|p_i(actual) - p_i(calculated)|$ for $i = 1, 2, 3$ is lesser than the predefined tolerance level.

Plotting all four poses



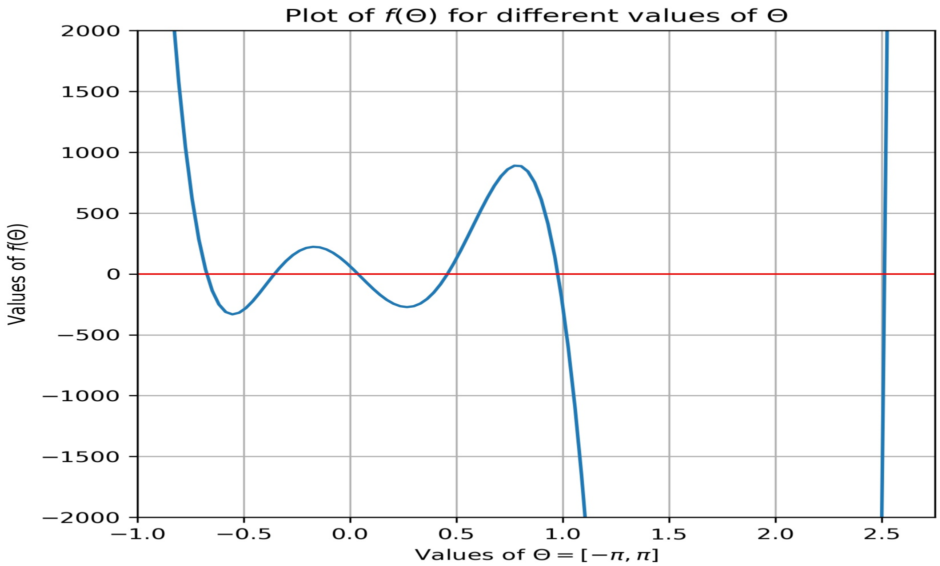






7 Question 5

As with question (4) above, we begin by plotting the function when $p_2 = 7$ and keeping everything same to see if we can identify the intervals containing true roots.



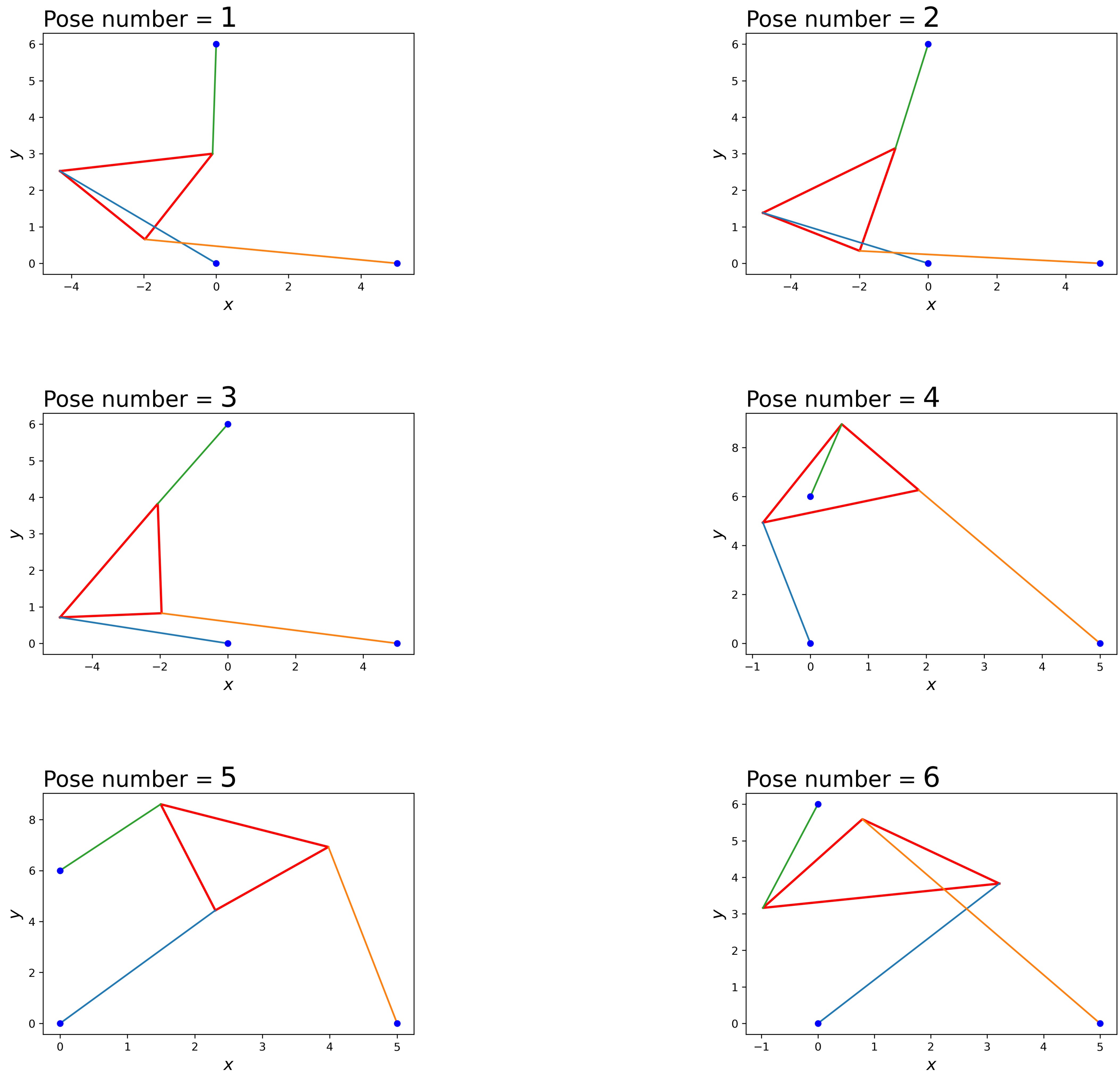
We can see that there are six roots (six poses) when $p_2 = 7$.

As we find out in question (4) above that Newton-Raphson method is more efficient than Bisection method, we are going to use the former method to find approximate roots and we adopted the similar methods to check our results by comparing the actual value of p_1, p_2, p_3 versus calculated. The initial conditions/roots, results of the algorithm with values of θ, x, y, iteration no. and calculated values of $p_i(calculated)$ for $i = 1, 2, 3$ are presented in the table below:

Table 5: Initial conditions/roots, values of θ, x, y, iteration no. and calculated values of p_i

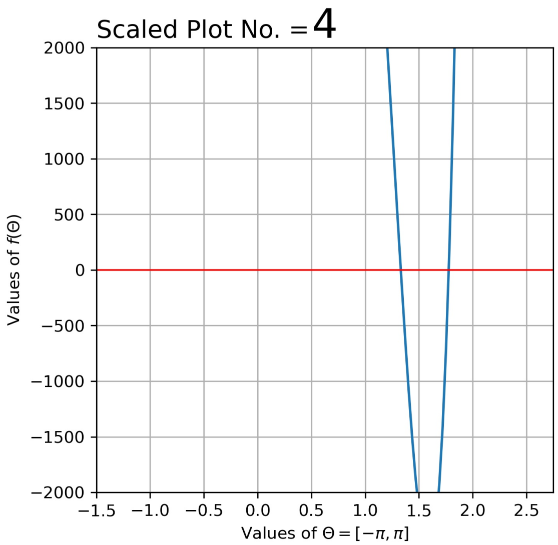
Iteration no	Initial root	θ	x	y	p_1	p_3	p_3
6	-0.8	-0.673157 486371674	-4.314759 599568257	2.526430 208403464	5.0	7.0	3.0
4	-0.4	-0.354740 27041567	-4.804896 519074754	1.383101 3849256757	5.000000 000000001	7.000000 000000001	3.000000 000000002
4	0.0	0.037766 7605759118	-4.949024 616818959	0.712148 3989450506	5.000000 000000004	7.000000 0000000036	3.000000 0000000067
4	0.5	0.458878 181048989	-0.819800 1690662414	4.932334 9118646584	4.999999 999999998	6.999999 999999998	2.999999 999999998
4	1	0.977672 895000362	2.303554 099146348	4.437751 515385477	5.000000 000000004	7.000000 000000003	3.000000 000000006
4	2.5	2.513852 79935038	3.215696 036151082	3.828746 4009884706	4.999999 999999995	6.999999 999999996	2.999999 99999999

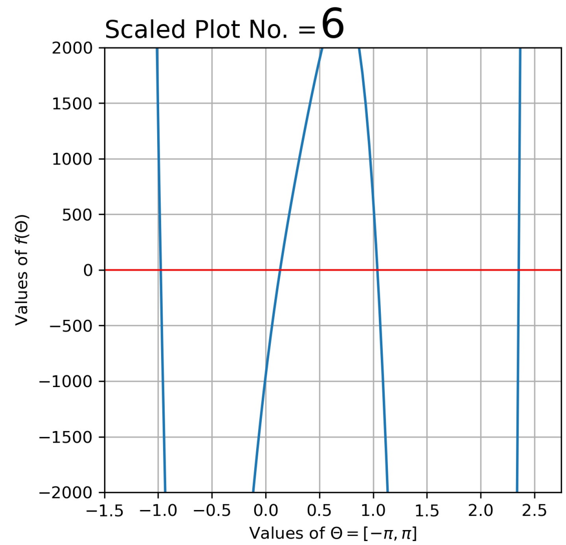
We are satisfied with these results since the error, defined as $|p_i(actual) - p_i(calculated)|$ for $i = 1, 2, 3$ is lesser than the predefined tolerance level. **Plotting all six poses**

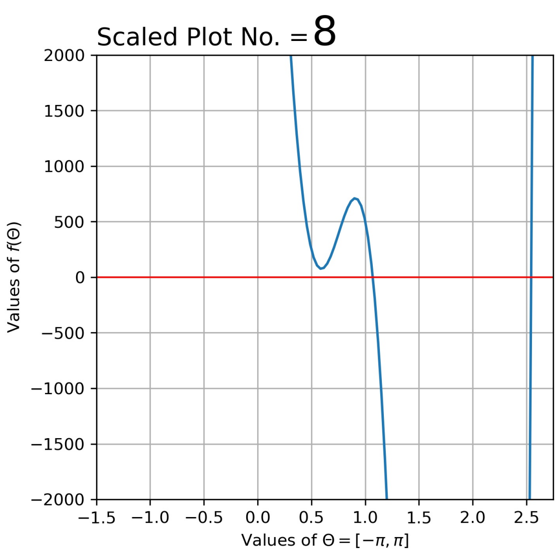


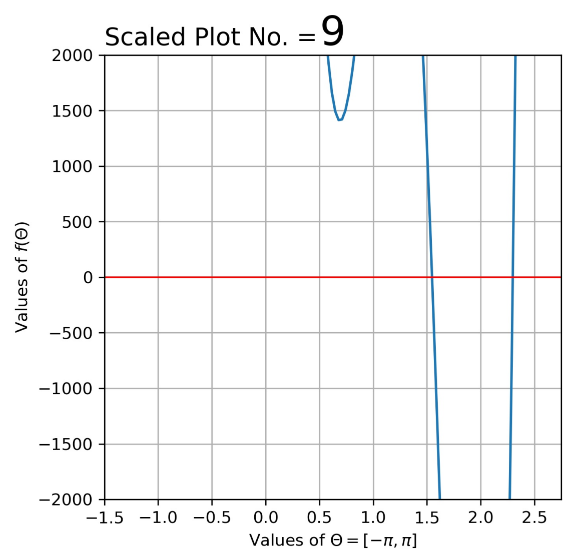
8 Question 6

We try to solve this question by iteration. We begin by plotting the $f(\theta)$ for different values of p_2 and see if we can find such a interval giving us only two roots. The plots for different values of p_2 are shown below. We are inserting only those plots which are relevant for us for comparison.









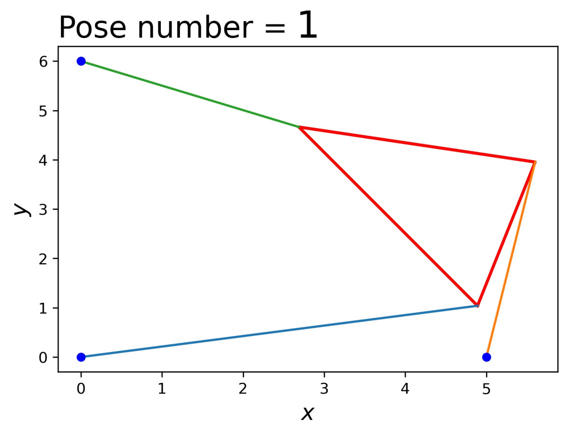
We see that when $p_2 = 4, 8, 9$ as shown in Scaled Plot No. =4,8,9, we have two roots. For all other values of p_2 we don't find two roots. We see that in those cases either we don't find roots (for $p_2 = 1, 2, 3, 10, 11, \ldots, \infty$) or we have more than roots (for $p_2 = 5, 6, 7$).

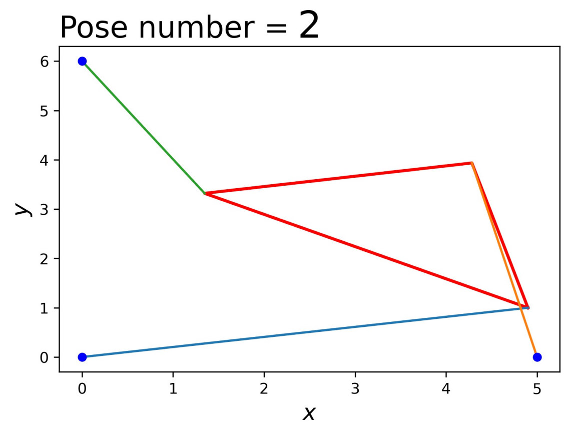
As stated in question (4) and question (5) we are using Newton-Raphson method to find approximate roots. The results of the algorithm are presented below:

Table 6: Initial conditions/roots, values of θ, x, y, iteration no. and calculated values of p_i

Iteration no	Initial root	θ	x	y	p_1	p_3	p_3
4	1.35	1.331642 20334278	4.890658 973005102	1.039930 1946595925	5.000000 000000006	4.000000 000000007	3.000000 0000000098
4	1.8	1.777513 57439986	4.899151 197793511	0.999158 4164477904	5.000000 000000005	4.000000 000000005	3.000000 0000000075

We are satisfied with these results since the error, defined as $|p_i(actual) - p_i(calculated)|$ for $i = 1, 2, 3$ is lesser than the predefined tolerance level. **Plotting two poses**





9 Question 7

We now attempt to find intervals for different values of p_2 identifying number of roots containing in each interval. Using the same approach as we adopted in question (6), we begin by plotting $f(\theta)$ for range of values of p_2 and see if we can find interval containing $0, 2, 4, 6$ roots. In this question, we don't need to calculate approximate roots, values of x, y, or check our results, or plot the poses. Therefore, we are only writing the intervals below with corresponding number of roots contained in it.

Table 7: Intervals containing $0, 2, 4, 6$ roots

Sr. No.	**Interval**	**No. of roots**
1	$(0, 3.7)$	0
2	$(3.8, 4.8)$	2
3	$(4.9, 6.9)$	4
4	(7.0)	6
5	$(7.1, 7.8)$	4
6	$(7.9, 9.2)$	2
7	$(9.3, \infty)$	0

10 Conclusion

We have seen that Newton-Raphson method gives approximate roots with fewer iterations than Bisection method. However, we must remember that Newton-Raphson method does not lead to convergence always since it is subject to the selection of initial roots closer to the true roots. Further, it may not give results at all if function has inflection points. In the given function, though, we have seen that function had inflection points, we were able to find approximate roots because by plotting we identified initial roots close to the true roots and avoid being trapped in inflection points. On the other hand, Bisection method leads to convergence always, though with more iterations, subject to the identification of two initial roots that bracket the true root. It does not get affected by the inflection points in the function. At last, Fixed-point iteration method should be used when we can find fewer number of functional form to represent $\theta = f(\theta)$. If we have many forms/equations, it would be computationally intensive and it may not lead to convergence at all since it convergence depends on the first derivative of $f(\theta)$ less than one evaluated at chosen initial root.

References

[1] Bisection method solver
https://www.geeksforgeeks.org/program-for-bisection-method/

[2] Differentiation for Newton method solver
https://en.m.wikipedia.org/wiki/Numerical_differentiation

[3] Matplotlib for plotting, labeling of axis, scientific vs default designs etc.
https://www.youtube.com/watch?v=cTJBJH8hacc&list=PLkdGijFCNuVm4IfZlsZPEt4fPJHfl-0g5&index=3

[4] Arrows props used in part (b) to highlight roots in the plot
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.arrow.html

[5] Calculating pi, trigonometric functions
https://www.youtube.com/watch?v=DcfYgePyedM&list=PLkdGijFCNuVm4IfZlsZPEt4fPJHfl-0g5&index=1

[6] Calculating the derivative used in Newton-Raphson method
https://www.youtube.com/watch?v=jmX4FOUEfgU&list=PLkdGijFCNuVm4IfZlsZPEt4fPJHfl-0g5&index=2

[7] Convergence of Bisection and Newton-Raphson method: Online lecture series by Prof. Niket S Kaisare, Week 4 and Week 5 lectures
https://nptel.ac.in/courses/127106019

[8] For loop and while loop to create simulations and sequence of iterations
https://github.com/AlphaWaveData/Jupyter-Notebooks/blob/master/Learn%20Python%20Loops.ipynb

[9] General information on root-finding algorithms
https://en.wikipedia.org/wiki/Root-finding_algorithms