Scientific Computing Project 1: Robotic Arm

Xingyu Liu

February 2021

1 Introduction

A Stewart platform consists of six variable length struts, or prismatic joints, supporting a payload. The project concerns a two-dimensional version of the Stewart platform. In this Stewart platform, three struts are given fixed lengths, while three struts are variable lengths. As shown in the figure below:

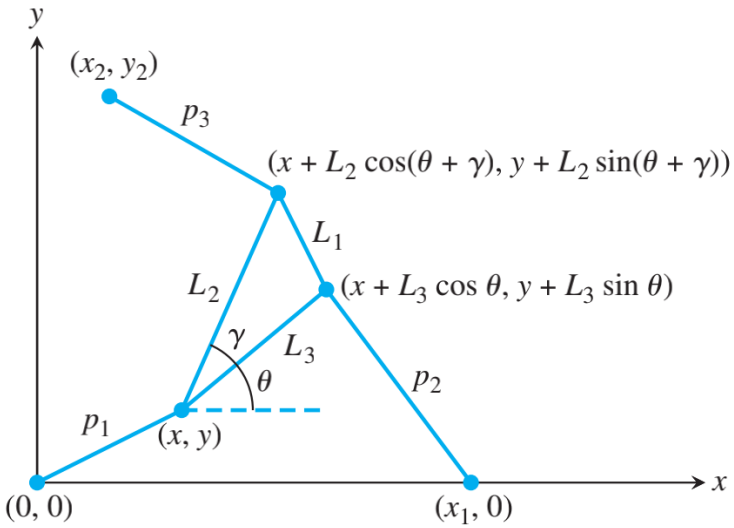

Figure 1: An image of a planar Stewart platform

In the figure, the lengths of L_1, L_2, and L_3 are supposed to be given and fixed, the lengths of P_1, P_2 and P_3 are supposed to be variable lengths. The values of x_1, x_2 and x_3 are supposed to be known, the values of x and y are supposed to be unknown. The angle γ is formed by sides L_2 and L_3, the angle θ is formed by line parallel to x-axis and side L_3.

There are six joint points, $(0,0)$, $(x_1,0)$, (x,y), (x_2,y_2), $(x+L_2cos(\theta+\gamma),(y+L_2sin(\theta+\gamma))$, $(x+L_3cos(\theta),y+L_3sin(\theta))$

The problem addressed is that, could we compute (x,y,θ) for each given P_1, P_2, P_3, L_1, L_2, L_3, x_1, x_2, y_2 and γ?

In the Project outline, we are given the following notations and relations:

$A_2 = -x_1 + L_3cos(\theta)$
$B_2 = L_3sin(\theta)$
$A_3 = -x_2 + L_2cos(\theta)cos(\gamma) - L_2sin(\theta)sin(\gamma)$
$B_3 = -y_2 + L_2sin(\theta)cos(\gamma) + L_2cos(\theta)sin(\gamma)$

$x = \frac{N_1}{D} = \frac{B_3(P_2^2-P_1^2-A_2^2-B_2^2)-B_2(P_3^2-P_1^2-A_3^2-B_3^2)}{2(A_2B_3-A_3B_2)}$

$y = \frac{N_2}{D} = \frac{-A_3(P_2^2-P_1^2-A_2^2-B_2^2)+A_2(P_3^2-P_1^2-A_3^2-B_3^2)}{2(A_2B_3-A_3B_2)}$

Since $P_1^2 = x^2 + y^2$, that means $P_1^2 = \frac{N_1^2}{D^2} + \frac{N_2^2}{D^2}$

Expand the above equation I get $N_1^2 + N_2^2 - P_1^2D^2 = 0$

In A_2, A_3, B_2, B_3 and consequently in N_1, N_2, D, the only unknown is θ.

Then we get a function of θ such that $f(\theta) = N_1^2 + N_2^2 - P_1^2D^2$

In the Project, we are required to write a function whose input is L_1, L_2, L_3, x_1, x_2, y_2, γ, and whose output is $f(\theta)$, plot $f(\theta)$ for θ in $[-\pi,\pi]$, approximately localize the roots of $f(\theta) = 0$. Solve for θ using one of the equation solvers. At last, using all the values we got solving for x and y given different sets of L_1, L_2, L_3, P_1, P_2, P_3, x_1, x_2 and y_2 in textbook.

2 Methodology

2.1 Write the function $f(\theta)$ in Python

As the introduction has given the function of $f(\theta)$, it is easy to write the function in Python just by inserting all the notations and relations in. Also, try this function using the given values in the textbook:

$L_1 = 2$, $L_2 = L_3 = \sqrt{2}$, $\gamma = \frac{\pi}{2}$, $p_1 = p_2 = p_3 = \sqrt{5}$.

```
import math
import numpy as np
from math import cos
from math import sin
from math import pi
from matplotlib import pyplot as plt
def function(theta):
    x1 = 4
    x2 = 0
    y2 = 4
    L1 = 2
    L2 = np.sqrt(2)
    L3 = np.sqrt(2)
    P1 = np.sqrt(5)
    P2 = np.sqrt(5)
    P3 = np.sqrt(5)
    gamma = pi/2
    A2 = (-x1) + (L3)*cos(theta)
    B2 = (L3)*sin(theta)
    A3 = (-x2) + (L2)*cos(theta)*cos(gamma) - (L2)*sin(theta)*sin(gamma)
    B3 = (-y2) + (L2)*sin(theta)*cos(gamma) + (L2)*cos(theta)*sin(gamma)
    D  = 2 * (A2*B3 - A3*B2)
    N_1 = (B3) * (P2**2 - P1 **2 - A2 ** 2 - B2 ** 2) - (B2) * (P3**2 - P1 **2 - A3 ** 2 - B3 ** 2)
    N_2 = (-A3) * (P2**2 - P1 **2 - A2 ** 2 - B2 ** 2) + (A2) * (P3**2 - P1 **2 - A3 ** 2 - B3 ** 2)
    x = (N_1)/D
    y = (N_2)/D
    P_1 = (x**2) + (y**2)
    final = (N_1**2 + N_2 ** 2) - ((P1**2) * (D**2))
    return round(final)
```

Figure 2: Function of theta

Test whether the function equals to zero when $\theta = -\frac{\pi}{4}$ and $\theta = \frac{\pi}{4}$.

Since in computer arithmetic, the real number x is replaced with the string of bits $fl(x)$, for checking the function, I rounded the final result using the function $round()$ in the code.

```
In [7]: function(pi/4)
Out[7]: -0.0

In [8]: function(-pi/4)
Out[8]: -0.0
```

Figure 3: Test the function of theta

The function looks good since two functions round to the value of 0.0.

2.2 Plot the function

Using some codes to plot the function of θ in the case $\theta = \pi/4$ or $\theta = -\pi/4$ in the interval $[-\pi, \pi]$:

```
vecfunc = np.vectorize(function)
d = np.arange(-pi, pi, 0.01)
T = vecfunc(d)
plt.plot (d, T, 'ro', d, T, 'k')
plt.show()
```

Figure 4: Code: Plot the function of theta

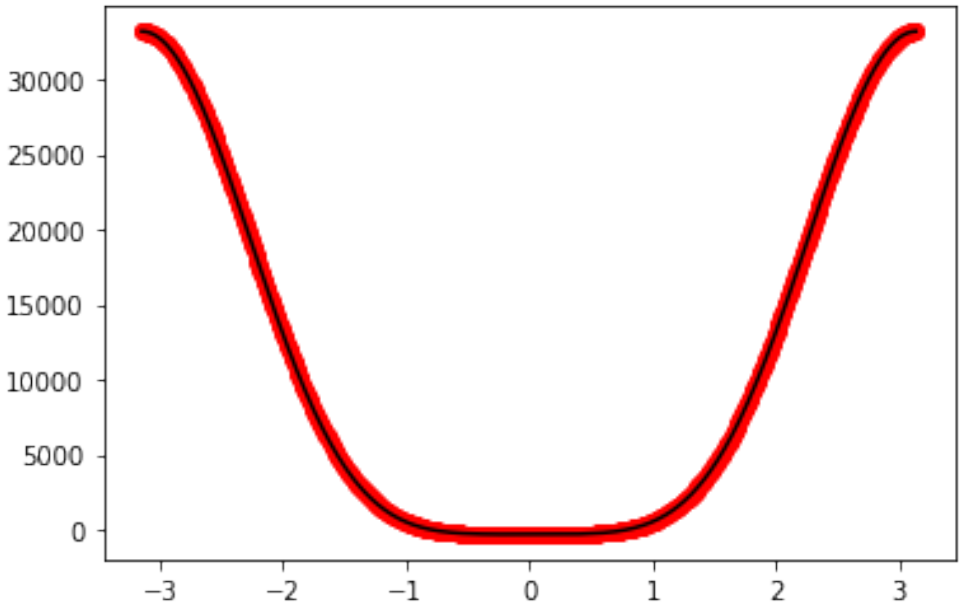

Figure 5: Plot the function of theta

From the figure above, it is kindly easy to recognize there are two solutions in $[-1, 0]$ and $[0, 1]$, for the case that $L_1 = 2$, $L_2 = L_3 = \sqrt{2}$, $\gamma = \frac{\pi}{2}$, $p_1 = p_2 = p_3 = \sqrt{5}$ and $\theta = \frac{\pi}{4}$ or $\theta = -\frac{\pi}{4}$.

2.3 Solve for θ using one of the equation solver

There are different equation solvers in Chapter 1, I will go through them one by one to check which one is most suitable for this function.

2.3.1 Newton's Method

If I want to use Newton's Method, I need to calculate the derivative of $f(\theta)$ first, but as the function of $f(\theta)$ is too complicated, the derivative of $f(\theta)$ is hard to calculate and to express, therefore, I would reject the Newton's Method.

2.3.2 Iterative fixed point Method

When I tried the iterative fixed point method, the range of θ becomes very large like that:

```
Out[45]: ([-114.0,
           239.0,
           -294.0,
           2310.0,
           18122.0,
           2178.0,
           19420.0,
           2446.0,
           9736.0,
           32263.0,
           1246.0,
           11555.0,
           -294.0,
           2310.0,
           18122.0,
           2178.0,
           19420.0,
           2446.0,
           9736.0,
           32263.0,
           1246.0,
           11555.0,
           -294.0,
           2310.0,
           18122.0,
           2178.0,
           19420.0,
           2446.0,
           9736.0,
           32263.0,
           1246.0,
           11555.0,
           -294.0,
           2310.0,
           18122.0,
           2178.0,
           19420.0,
```

Figure 6: Output the fixed point method

I think that is because, whenever θ is a solution, $\theta + 2\pi n$ for n an integer is also a solution, there are infinite solutions. So the method of iterative fixed point is not suitable.

2.4 Bisection method

Finally, I chose bisection method to solve the function, since from the plot, we can easily to lock some intervals for where roots would be around, then it is plausible to get the final answer via bisection method in Python.

First, define a bisection method function:

```
def bisection(f, a, b, TOL):
    if np.sign(f(a))*np.sign(f(b)) > 0:
        print('f(a)f(b)<0 not satisfied')
        return # stop execution
    n=1
    fa= f(a)
    fb= f(b)
    while np.abs(a-b)>TOL:
        c = (a+b)/2
        fc=f(c)
        n=n+1
        if np.isclose(f(c), 0):
            print('Approximate  root', c, 'has been obtained in', n, 'steps')
            return
        if np.sign(fc)*np.sign(fa)<0:
            b = c
            fb=fc
        else:
            a = c
            fa= fc
    c=(a+b)/2
    print('The final interval [', a, b, '] contains a root')
    print('Approximate  root', c, 'has been obtained in', n, 'steps')
```

Figure 7: Define a bisection method

Then, using the guessed interval in the problem 1 ($[-1, 0]$ and $[0, 1]$) to find the approximate root for the function of θ.

```
bisection(function,-1,0,0.5e-4)
```

```
Approximate  root -0.78515625 has been obtained in 9 steps
```

Figure 8: Root 1

```
bisection(function,0,1,0.5e-4)
```

```
Approximate  root 0.78515625 has been obtained in 9 steps
```

Figure 9: Root 2

Then I would define a function to calculate the final x and y:

```
def functionforxy(x1,x2,y2,L1,L2,L3,P1,P2,P3,delta,theta):
    A2 = (-x1) + (L3)*cos(theta)
    B2 = (L3)*sin(theta)
    A3 = (-x2) + (L2)*cos(theta)*cos(delta) - (L2)*sin(theta)*sin(delta)
    B3 = (-y2) + (L2)*sin(theta)*cos(delta) + (L2)*cos(theta)*sin(delta)
    D  = 2 * (A2*B3 - A3*B2)
    N_1 = (B3) * (P2**2 - P1 **2 - A2 ** 2 - B2 ** 2) - (B2) * (P3**2 - P1 **2 - A3 ** 2 - B3 ** 2)
    N_2 = (-A3) * (P2**2 - P1 **2 - A2 ** 2 - B2 ** 2) + (A2) * (P3**2 - P1 **2 - A3 ** 2 - B3 ** 2)
    x = (N_1)/D
    y = (N_2)/D
    return (round(x),round(y))
```

Figure 10: Function to calculate x and y

Type in the values of problem 1:

```
functionforxy(4,0,4,2,np.sqrt(2),np.sqrt(2),np.sqrt(5),np.sqrt(5),np.sqrt(5),pi/2,0.78515625)
```

```
(2.0, 1.0)
```

Figure 11: Calculate x and y

Actually, given $\theta = \frac{\pi}{4}$, $L_1 = 2$, $L_2 = L_3 = \sqrt{2}$, without giving P_1, P_2, P_3 and x_0, x_1, y_1, we can still calculate the value of (x, y) using similar triangle as shown in the figure:

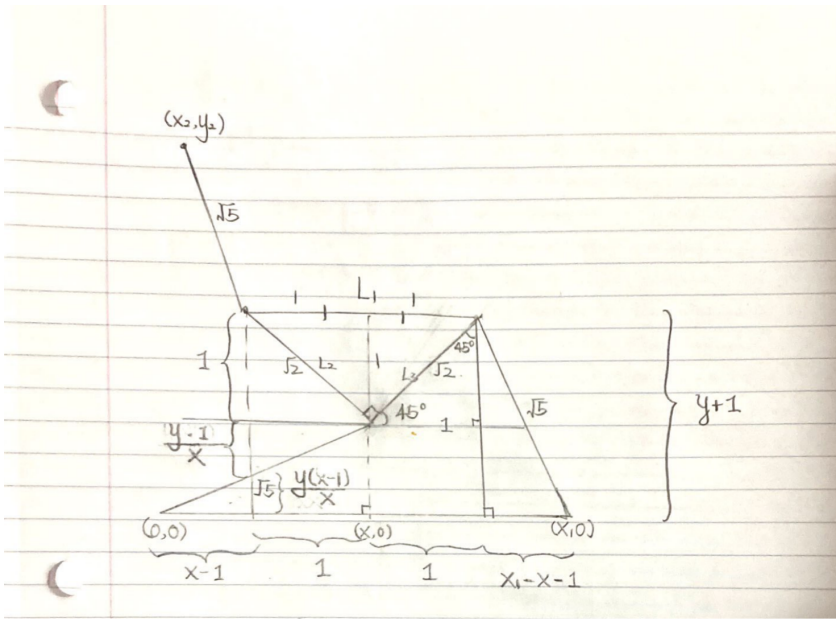

Figure 12: Calculate x and y

$\frac{y}{x} + \frac{y(x-1)}{x} + 1 = y + 1$
$\Rightarrow \frac{y+xy-1}{x} = y$
$\Rightarrow y + xy - 1 = xy$
$\Rightarrow y - 1 = 0$
$\Rightarrow y = 1$
$\Rightarrow x = \sqrt{5-1} = 2$

$(x, y) = (2, 1)$ is exactly what we get from the code.

Also, we could use the function $functionxy()$ to try other values, such as $x_1 = 5$ $x_2 = 0$, $y_2 = 6$, $L_1 = 3$, $L_2 = 3$, $L_3 = 3\sqrt{2}$, $P_1 = 3$, $P_2 = 5$, $P_3 = 5$, $\gamma = \frac{\pi}{4}$.

```
def function(theta):
    x1 = 5
    x2 = 0
    y2 = 6
    L1 = 3
    L2 = 3*np.sqrt(2)
    L3 = 3
    P1 = 5
    P2 = 5
    P3 = 5
    delta = pi/4
    A2 = (-x1) + (L3)*cos(theta)
    B2 = (L3)*sin(theta)
    A3 = (-x2) + (L2)*cos(theta)*cos(delta) - (L2)*sin(theta)*sin(delta)
    B3 = (-y2) + (L2)*sin(theta)*cos(delta) + (L2)*cos(theta)*sin(delta)
    D  = 2 * (A2*B3 - A3*B2)
    N_1 = (B3) * (P2**2 - P1 **2 - A2 ** 2 - B2 ** 2) - (B2) * (P3**2 - P1 **2 - A3 ** 2 - B3 ** 2)
    N_2 = (-A3) * (P2**2 - P1 **2 - A2 ** 2 - B2 ** 2) + (A2) * (P3**2 - P1 **2 - A3 ** 2 - B3 ** 2)
    x = (N_1)/D
    y = (N_2)/D
    P_1 = (x**2) + (y**2)
    final = (N_1**2 + N_2 ** 2) - ((P1**2) * (D**2))
    return round(final)
vecfunc = np.vectorize(function)
d = np.arange(-pi, pi, 0.01)
T = vecfunc(d)
plt.plot (d, T, 'ro', d, T, 'k')
plt.show()
```

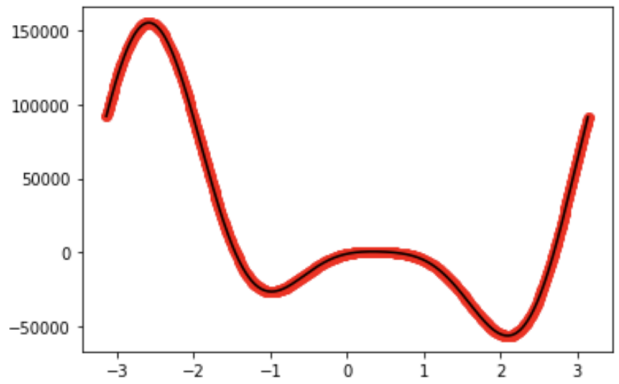

Figure 13: The second try

From the figure, we could see there are almost six roots at all, there is safely one root in the interval $[-2, -1]$, so I will use this interval for bisection method to find the one root of the function.

```
bisection(function,-2,-1,0.5e-4)
```

```
The final interval [ -1.46258544921875 -1.462554931640625 ] contains a root
Approximate  root -1.4625701904296875 has been obtained in 16 steps
```

Figure 14: The second try

$\theta = -1.4625701904296875$ by calculation from bisection method.

```
functionforxy(5,0,6,3,3,3*np.sqrt(2),3,5,5,pi/4,-1.4625701904296875)
```

```
(-0.0, 3.0)
```

Figure 15: Calculate x and y

Finally we get $(x, y) = (0, 3)$ for $\theta = -1.4625701904296875$.

Similarly, use different intervals containing the root shown in the plot of $f(\theta)$ in bisection method, we would get another root, then plug this root in $functionxy()$, we get a different (x, y).

3 Conclusion

In my project, first I define a function of variable θ, when plugging in the values of L_1, L_2, L_3, P_1, P_2, P_3, x_1, x_2, y_2 we get a function of θ, θ here is the only variable in the function. When plotting the figure of this function $f(\theta)$ in interval $[-\pi, \pi]$, we could approximately localize the roots of the the function, pick one interval which contains the root, using bisection method then, we could get the approximate value of θ, using this θ and the defined function to calculate the final (x, y).