Lab 1: Design of Robotic Systems

Wu, Yichen
504294181

Collaborator:
Yaofang Zhang
004446568

EE183D

January 26, 2017

1 Introduction

In this lab, we modeled the mechanical linkage of a human leg, with 3 degrees of freedom at hip and 1 degree of freedom at knee. We first extracted the D-H parameters of each joint and computed the transformation matrices for forward kinematics. And then we calculated the Jacobian matrix and completed inverse kinematics calculation on a simple linear motion with iterative method.

2 Methods

We first need to establish a model of human leg.

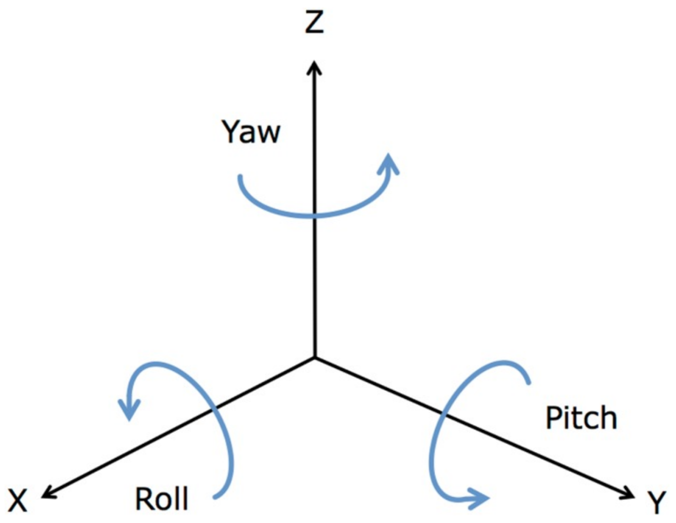


Figure 1: world frame coordinate used in the following images

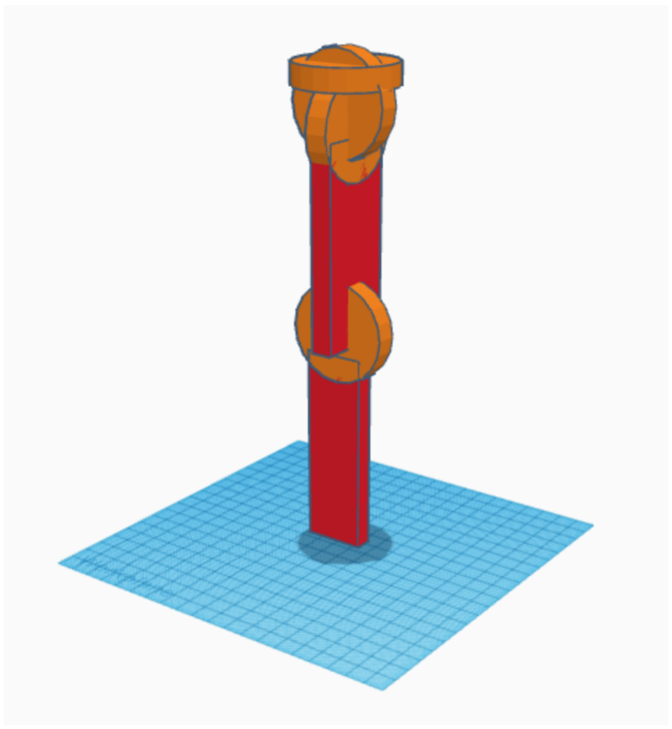
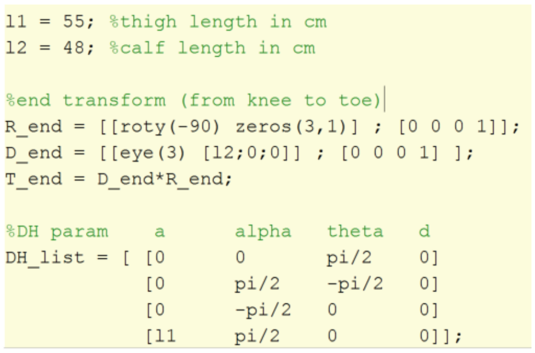


Figure 2: 4 DOF (From top to bottom): Yaw at hip (+ counterclockwise), Roll at hip (+ counterclockwise), Pitch at hip (+ counterclockwise), Roll at knee (+ counterclockwise).

Then we extract the D-H parameters from this model.



```
l1 = 55; %thigh length in cm
l2 = 48; %calf length in cm

%end transform (from knee to toe)
R_end = [[roty(-90) zeros(3,1)] ; [0 0 0 1]];
D_end = [[eye(3) [l2;0;0]] ; [0 0 0 1] ];
T_end = D_end*R_end;

%DH param     a       alpha    theta    d
DH_list = [ [0       0        pi/2     0]
            [0       pi/2     -pi/2    0]
            [0       -pi/2    0        0]
            [l1      pi/2     0        0]];
```

Figure 3: D-H parameters extracted from a human right leg.

After we compute the transformation matrices using equation:

```
T(:,:,i) = [[cos(theta(i))               -sin(theta(i))               0               a(i)]
            [sin(theta(i))*cos(alpha(i))  cos(theta(i))*cos(alpha(i)) -sin(alpha(i))  -d(i)*sin(alpha(i))]
            [sin(theta(i))*sin(alpha(i))  cos(theta(i))*sin(alpha(i)) cos(alpha(i))   d(i)*cos(alpha(i))]
            [0                            0                           0               1]];
```

We test this forward kinematics model with controlled configuration state.

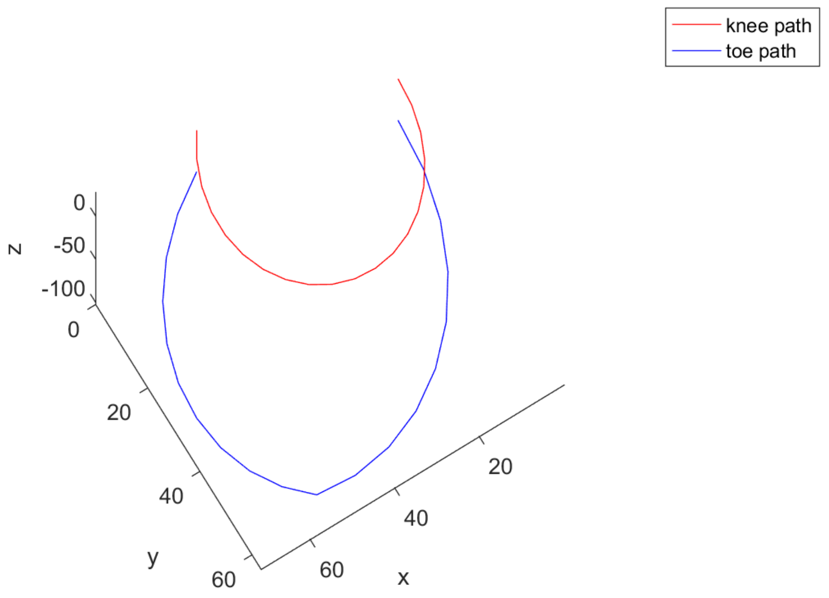
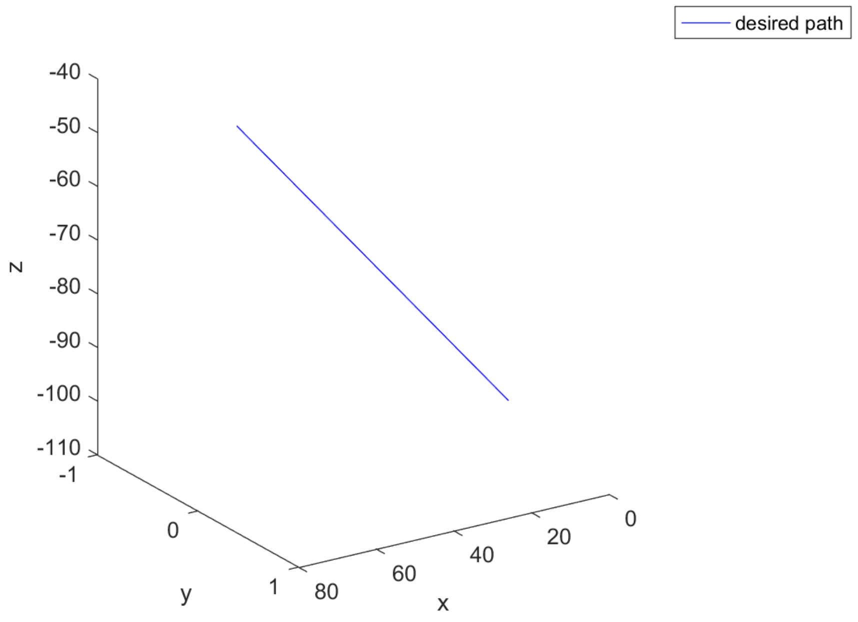


Figure 4: A sample kick-and-retract-to-the-side movement.

In the next part, we implement the inverse kinematics algorithm. We first created a linear path of desired operational state.



Since our system is under-actuated, we only try to make the coordinates of end effector(toe) match with the path, and disregard the rotational orientation. We implemented an iterative method to find the joint states given an end coordinate. First we pick a joint state, evaluate the difference between the desired joint state and the current state, calculate the Jacobian matrix, and take a joint state step forward.
Since this motion starts at the all-zero configuration-state, we used this as a known starting point. And for each next point, we use the previous calculated joint state values as the start searching point.

$$dq = (J^T J)^{-1} J^T \times ds \tag{1}$$

In practice, We divided each row $\vec{r}$ of the Jacobian matrix by $||\vec{r}||^2$ and multiplied it with the difference δs.

3 Results

We setup a threshold of error at $0.01cm$ and it yields:

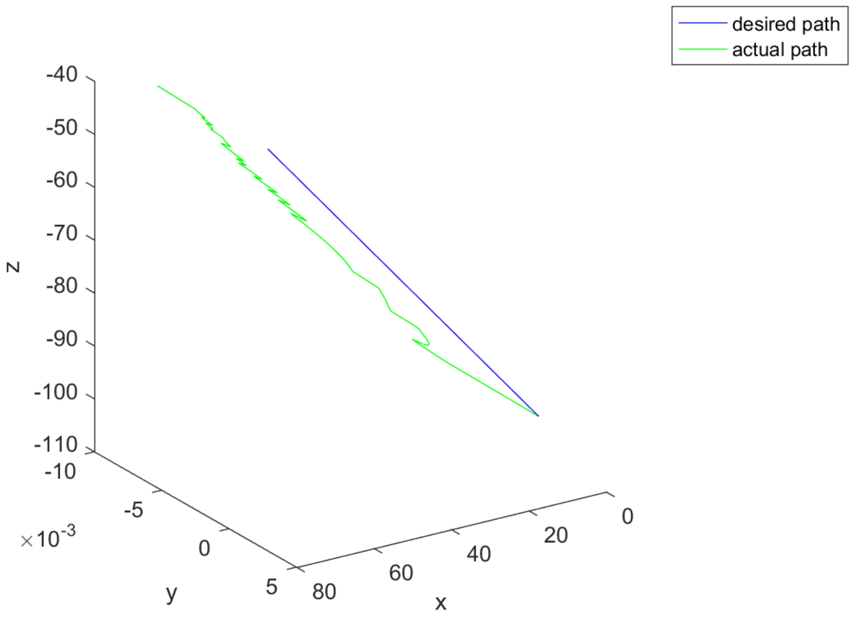


Figure 5: Calculated path with 0.01 cm tolerance.

The threshold can be set smaller to achieve a more accurate path.

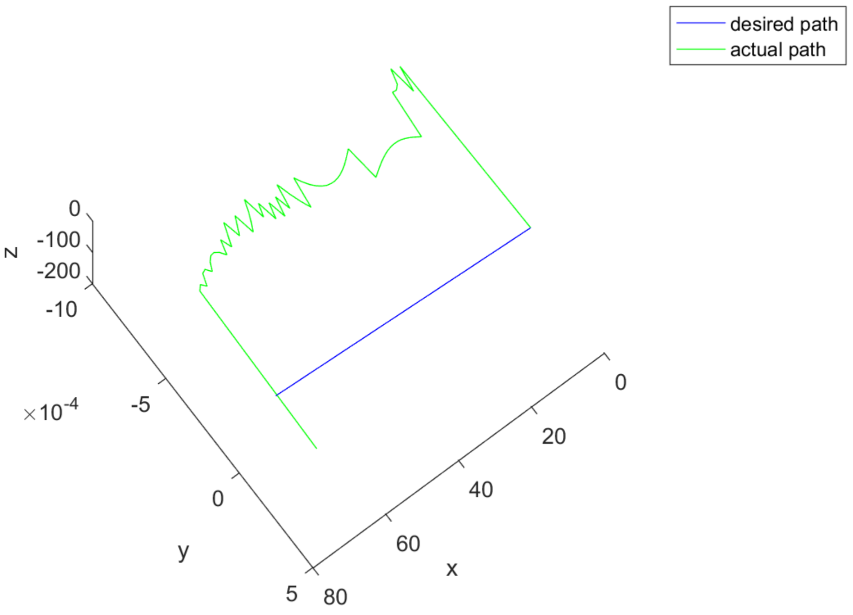


Figure 6: Calculated path with 0.001 cm tolerance.

This lab took me about 8 hours. Debugging took about half of the time.