

**MSE 4401—Robotic Manipulators**  
**Assignment #4**

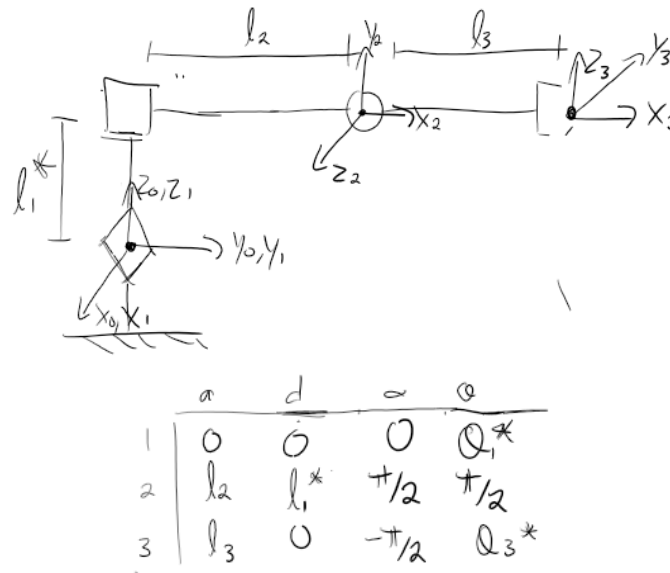
Dr. Ana Luisa Trejos  
Alexander Porrone - 250733678  
Due: November 9th, 2018

## Introduction

The purpose of this assignment is to develop a path planning algorithm that can generate a path that avoids obstacles for the RPR manipulator that was featured in our midterm. The path planning algorithm is to use attractive and repulsive fields to generate a path for the RPR manipulator.

## Robot Configuration

The configuration on the manipulator along with its DH parameters can be found below:



## Functions

Before an algorithm could be developed to plan a trajectory for the RPR manipulator MATLAB functions needed to be generated to calculate important values for the manipulator in each configuration.

### **DHMatrix**

A function was developed to calculate the DH matrix when provided with the DH parameters of the manipulator. This function was used by the forward kinematics function to perform its calculation. See DHMatrix.txt attached for the MATLAB implementation.

### **fkinRPR**

A function was developed to calculate the forward kinematics of the RPR manipulator when fed in the position of each joint. The function would output a transformation matrix from the base frame to the frame of each joint. See fkinRPR.txt attached for the MATLAB implementation

**invkinRPR**

A function was developed to calculate the inverse kinematics of the RPR manipulator when provided with the end effector position of the robot. The inverse kinematics relationships calculated for the midterm were used to implement this function. See invkinRPR.txt attached for the MATLAB implementation.

**JaconRPR**

A function was developed to calculate the jacobian of the RPR manipulator when provided with the position of each joint on the robot. The jacobian calculation implemented the forward kinematics function to calculate the jacobian between the base frame and each joint. See JaconRPR.txt for the MATLAB implementation.

**WorkspaceGen**

A function was developed to generate the workspace that the manipulator would operate in as well as populate the workspace with obstacles. The function would generate a 3d array of zeros with ones representing the objects in the workspace. See WorkspaceGen.txt for the MATLAB implementation.

**computeAttraction**

The computeAttraction function would calculate the force on each joint of the manipulator in the x, y, and z axis. It would take in the starting and ending position of the end effector as well as the attractive field constant and the distance at which each joint would transition from the far range to close range calculation of force. This function implemented the equation shown below:

$$F_{att,i}(q) = -\nabla U_{att,i}(q)$$

$$= \begin{cases} -\xi_i(o_i(q) - o_i(q_f)) & \text{for } \|o_i(q) - o_i(q_f)\| \leq d \\ -d\xi_i \frac{(o_i(q) - o_i(q_f))}{\|o_i(q) - o_i(q_f)\|} & \text{for } \|o_i(q) - o_i(q_f)\| > d \end{cases}$$

Please see computeAttraction.txt for my implementation of the function.

**computeRepulsion**

The computeRepulsion function would calculate the force on each joint of the manipulator in the x, y, and z axis. It would take in the position of the end effector as well as the repelling field

constant, the workspace and the range at which the repelling field would become active. The function implemented the equation shown below:

$$F_{rep,i}(q) = \begin{cases} \eta_i \left( \frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0} \right) \frac{1}{\rho(o_i(q))^2} \nabla \rho(o_i(q)) & \text{for } \rho(o_i(q)) \leq \rho_0 \\ 0 & \text{for } \rho(o_i(q)) > \rho_0 \end{cases}$$

Please see computeRepulsion.txt for my implementation of the function.

### **repelDistance**

The repelDistance function calculated the distance to the nearest object as well as the unit vector direction to the nearest object from all current joint locations. It took in the joint location and the workspace. Please see repelDistance.txt for my implementation of the function.

### **Workspace**

The workspace for this virtual robot is a cylinder with a radius of 50 units. L2 for the robot is 30 units and L3 for the robot is 20 units. As there are no joint limits assigned to the prismatic joint the robot could in theory move up and down within this cylinder to infinity. If joint limits were assigned to the prismatic joint, it would be a cylinder with a half sphere for its top. Please see Exhibit 1 for a depiction of the workspace with objects and the manipulator in it.

### **Path Planning**

The path planning algorithm was relatively simple to product after all the functions described above were created. Gradient decent was used to calculate the path for the manipulator and it was implemented using the following algorithm:

1.  $0 \rightarrow i, q_s \rightarrow q^0$
2. if  $\|q^i - q_f\| > \varepsilon$ 

$$q^{i+1} = q^i + \alpha^i \frac{\tau(q^i)}{\|\tau(q^i)\|}$$

$$i++$$
- else
 
$$\text{return } \langle q^0, q^1, \dots, q^i \rangle$$
3. goto 2

Limits were set as to the number of loops that could be performed and values for the repulsive field, attractive field, repulsive distance, attractive transition point and step size were assigned to ensure optimal movement around the workspace. See Assignment4.txt for implementation of this code.

### **Start and End Variation**

As shown in Exhibit 2 and 3 the path planning algorithm implemented in this model can work with various start and end points and all joints avoid objects along the way. As seen in Exhibit X the manipulator clearly avoided the obstacles with both its end effector and joint 2. The start and end position of each configuration can be found below:

#### Configuration 1

```
startPos = [5;45;20]
goalPos = [15;-10;37]
```

#### Configuration 2

```
startPos = [10;35;10]
goalPos = [20;-5;35]
```

### **Repulsive Field Variation**

As shown in Exhibit 4 and 5 the variation of the repulsive distance correlates to the distance around the objects each joint stay. When the repulsive distance was increased from 10 to 500 you can see that the joints remain farther from each object throughout their motion. Below were the two configurations used for the distance manipulation.

#### Configuration 1

```
repulsiveDistance = 10;
```

#### Configuration 2

```
repulsiveDistance = 500;
```

**Attractive and Repulsive Constant Variation**

As shown in Exhibit 6 and 7 the variation of the attractive and repulsive constants completely changes the manipulators path through the attractive and repulsive fields. Through the variation of the attractive and repulsive constants you completely manipulate the magnitude of the attractive and repulsive fields causing a gradient that pushes the robot joints in a different direction than the original path. As shown in the two configurations below the attractive influence on each joint was broken into x, y, and z components so that they could be independently manipulated.

**Configuration 1**

```
repulsiveInfluence = 5000;  
attractiveInfluence = [100; 100; 50000];
```

**Configuration 2**

```
repulsiveInfluence = 10000;  
attractiveInfluence = [300; 300; 500000];
```

**Appendix**

Exhibit 1: Workspace

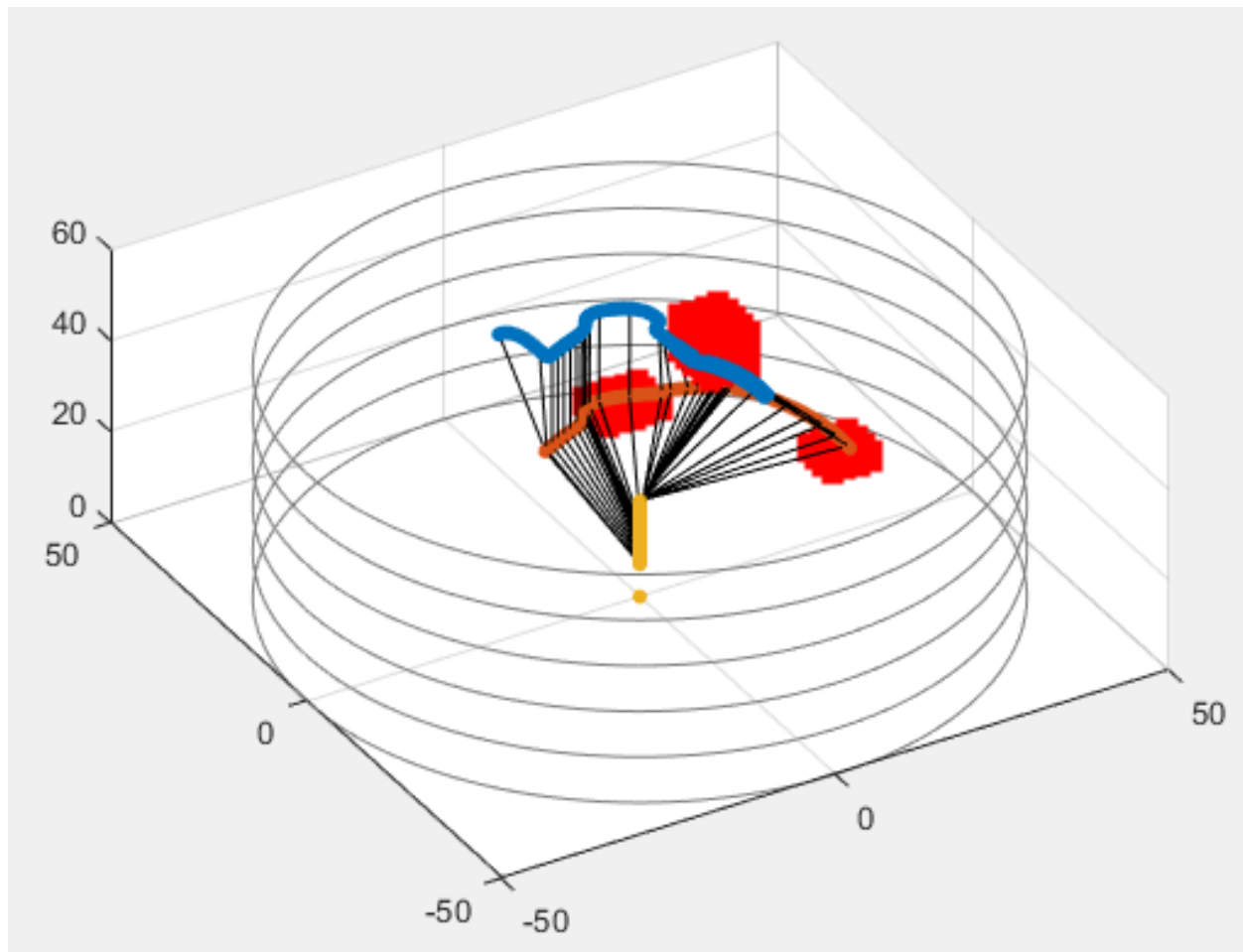


Exhibit 2: Start and End Configuration 1

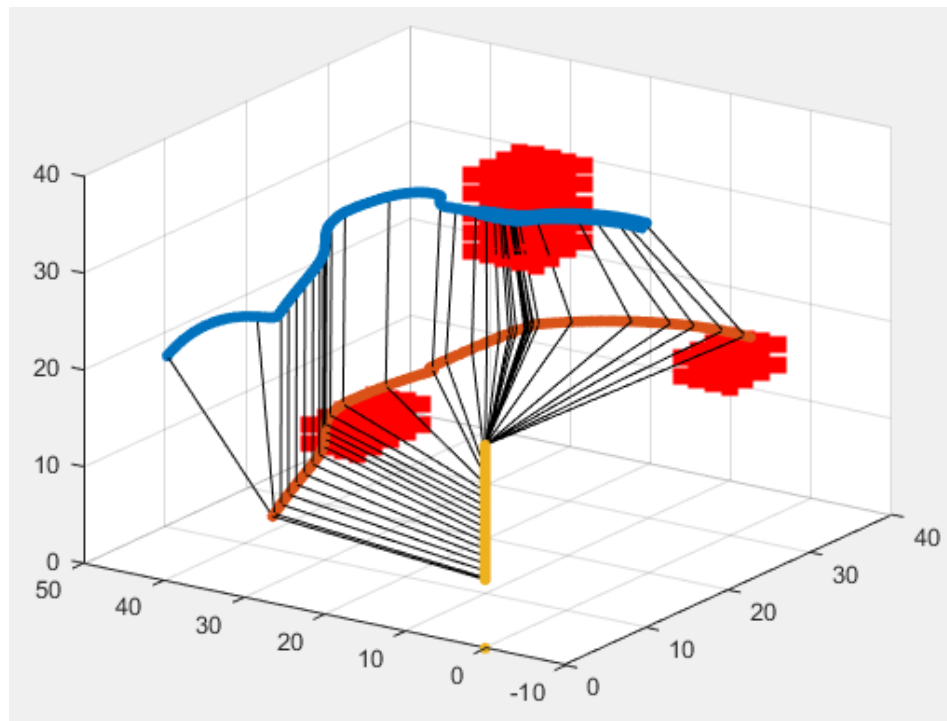


Exhibit 3: Start and End Configuration 2

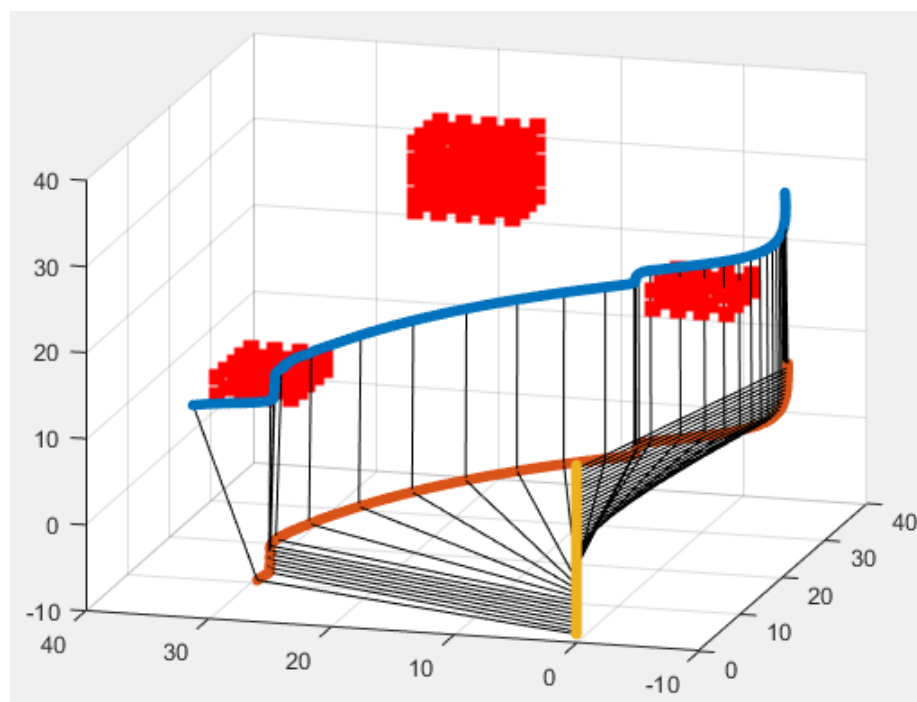




Exhibit 4: Repulsive Influence Configuration 1

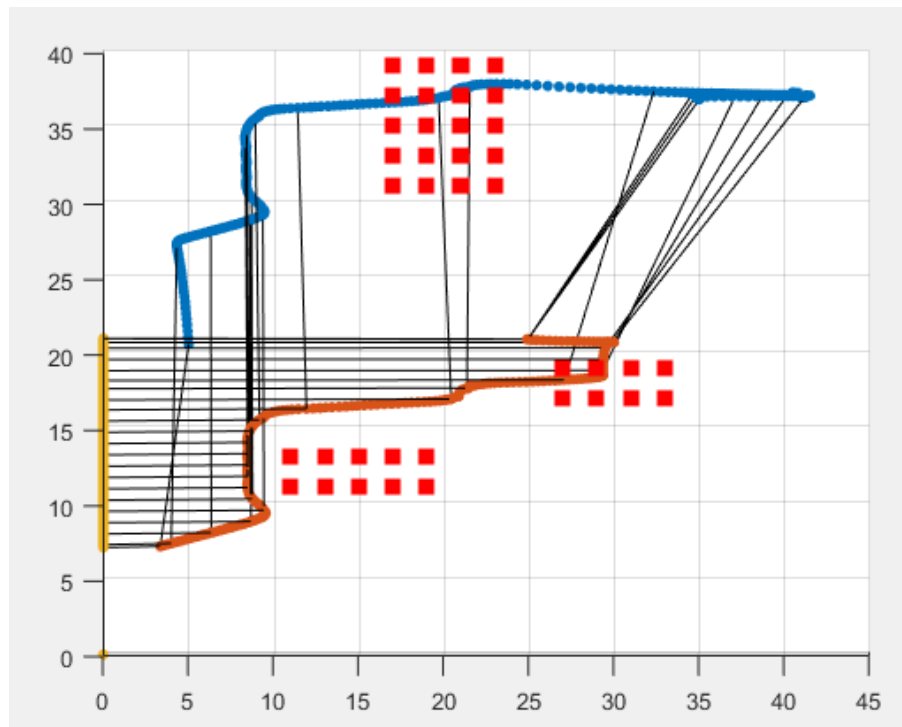


Exhibit 5: Repulsive Influence Configuration 2

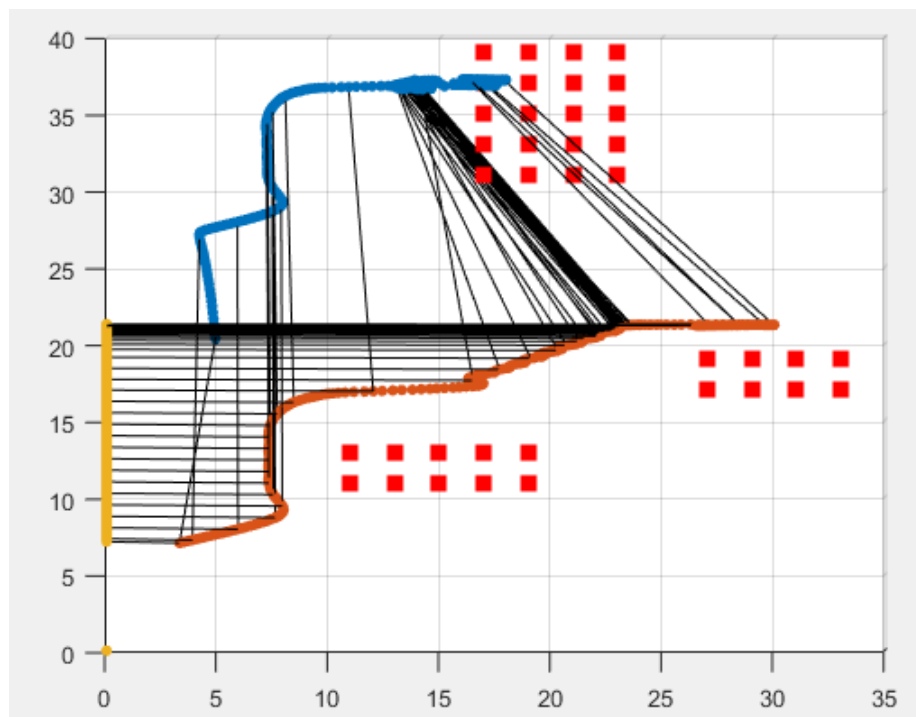


Exhibit 6: Repulsive and Attractive Constant Configuration 1

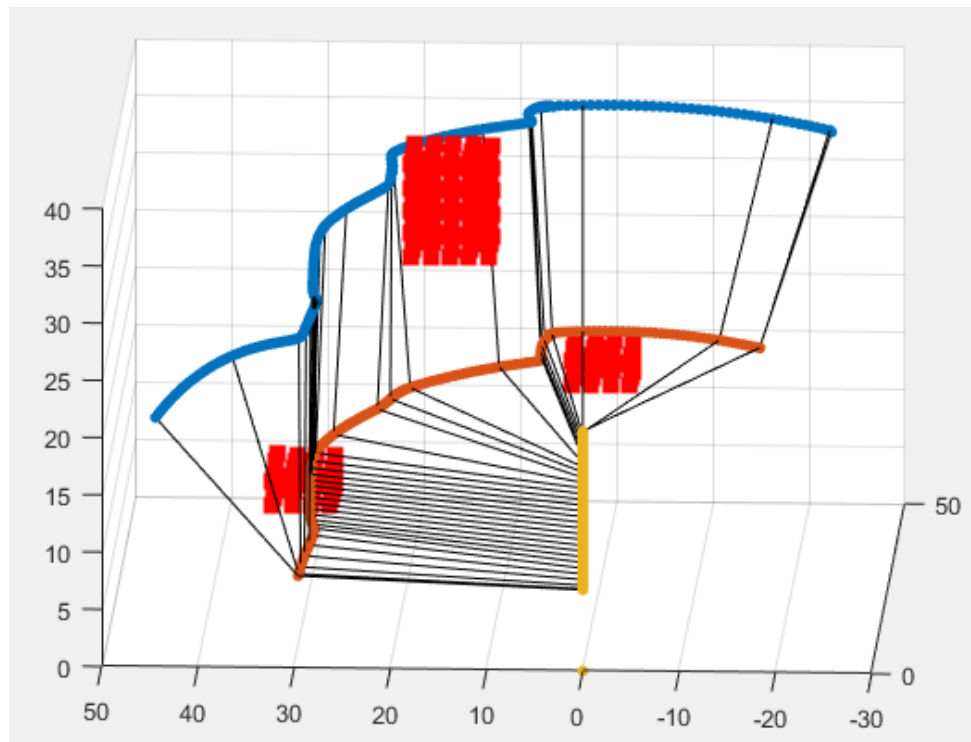


Exhibit 7: Repulsive and Attractive Constant Configuration 2

