Group 8
Deepak Yadav - AM.EN.U4AIE19024
Gourav Singh Bajeli - AM.EN.U4AIE19031
Indraraj Biswas- AM.EN.U4AIE19034
Shashank Priyadarshi - AM.EN.U4AIE19060

Robotic Manipulator using MATLAB

Submission date: 10th December 2020

ABSTRACT

In this project we are going to present a robotic arm with a motion planner and a 3D viewer. In order to do this task, a motion planner, which deals with constraints, i.e. collisions among the arm joints and objects, is used. Also a 3D viewer, called Simulink 3D, is used to visualize the motion of the arm. Here we are going to deal with a robot having 2 degrees of freedom (DOF) arms. The simulation of a custom robotic arm was done in order to gain experience with some nice features of MATLAB and Simulink for robot programming, we can build a scalable robot simulation to prototype, test concept models, and debug inexpensively. Then you can use the high-fidelity models for validation while keeping the rest of the algorithms in the same simulation environment. Just a little change in the mechanism model is enough to make a big change in the simulation.

THEORY

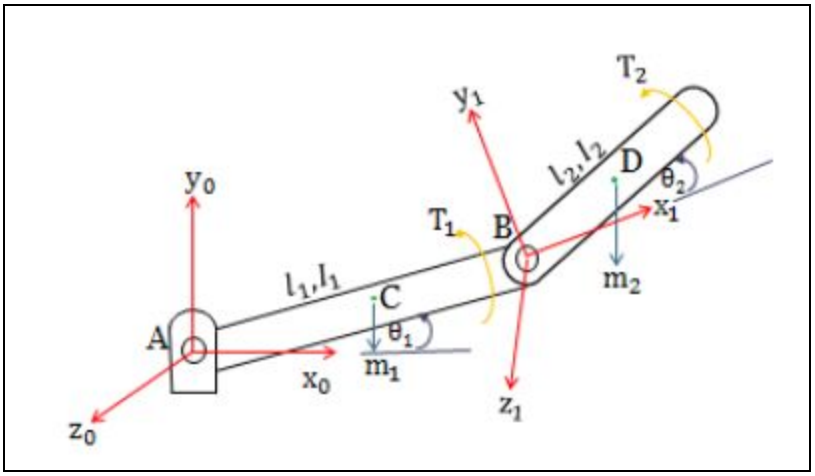
The design of two degrees of freedom serial robotic arm has been presented in this project. The analysis of the dynamic properties of the robotic arm has been presented. The solutions of the kinematics problem of any robot manipulators

have two types; the forward kinematic and inverse kinematics. In forward kinematics, the end-effector position and orientation is determined from the given set of joint angles whereas in inverse kinematics, the vice versa is done.

1. Forward Kinematics :

   Forward kinematics is the mathematical model that relates the known positions of each of the joints and their relationship with the Cartesian axes. The DH convention describes the behavior of the l(i) link with respect to the l(i-1)link, with 4 important transformations described with the below parameter :

   - Θi, angle of the joint, with which the orthogonal plane is projected with respect to the normal anterior plane.
   - di, displacement of the joint, length between the links with respect to their joints.
   - ai, length of the link between the common perpendiculars.
   - αi, torsion angle between the orthogonal projections of the Z axis in a perpendicular plane.

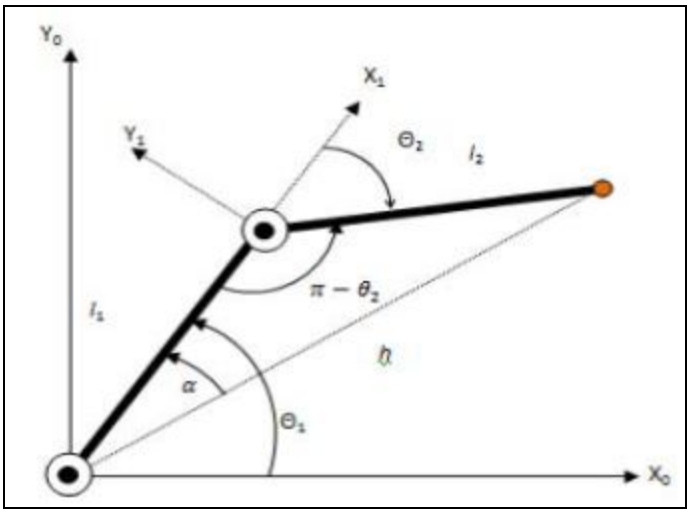


   The homogeneous matrix that represents those 4 transformations previous parameters is :

$$A_1^0 = \begin{bmatrix} \cos(\theta) & -sen(\theta) & sen(\theta) & a * \cos(\theta) \\ sen(\theta) & \cos(\theta) & -\cos(\theta) & a * sen(\theta) \\ 0 & sen(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2. Inverse Kinematics :

To solve the angles the arms need to be at in order to achieve a desired location of the end effector, inverse kinematics is used. Inverse kinematics calculates the required angles of the arms,θ(1) & θ(2) in order for the end effector to reach its desired coordinate point.



$$\theta_1 = \tan^{-1}\frac{y}{x} - \tan^{-1}\frac{l_2 \sin q_2}{l_1 + l_2 \cos q_2},$$

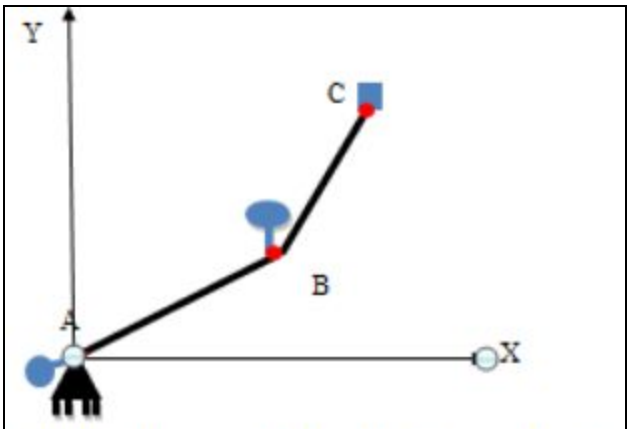
$$\alpha = \cos^{-1}\left(\frac{l_1^2 + h^2 - l_2^2}{2 l_1 h}\right),$$

$$\theta_2 = \pi - \cos^{-1}\left(\frac{l_1^2 + l_2^2 - h^2}{2 l_2 l_1}\right),$$

The end effector is represented by the orange circle. (x0, y0) is the coordinate frame for arm l1, and (x1,y1) the coordinate frame for arm l2. The above equations we can see that the direct and inverse kinematics of the manipulator can be described in closed form.

3. Manipulators :

The 2-DOF robot manipulator is a mechanical system that uses several computer-controller serial chains to support a single platform, or end-effector. It becomes progressively less rigid with more components.



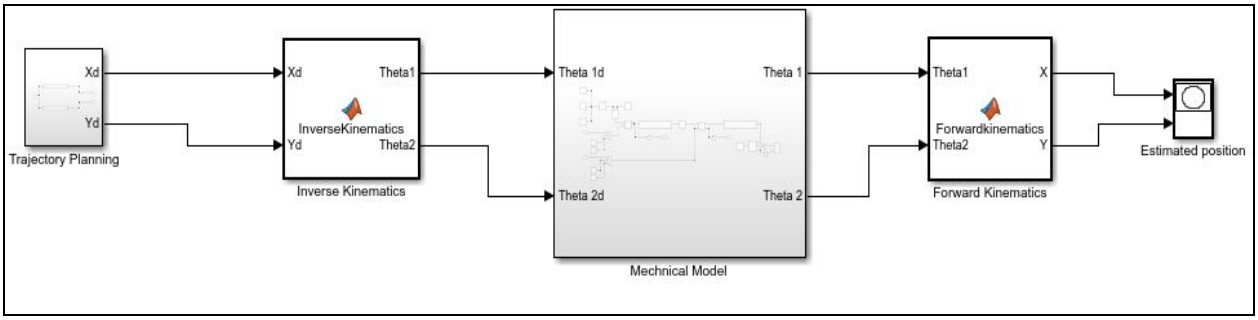
2-DOF robot manipulator Mechanism Sketch

They can be first acting in comparison to serial manipulator. 2-DOF robot manipulators are usually limited in the workspace; for instance, they generally cannot reach around obstacles. The calculations involved in performing a desired manipulation (inverse kinematics) are also usually more difficult and can lead to multiple solutions.

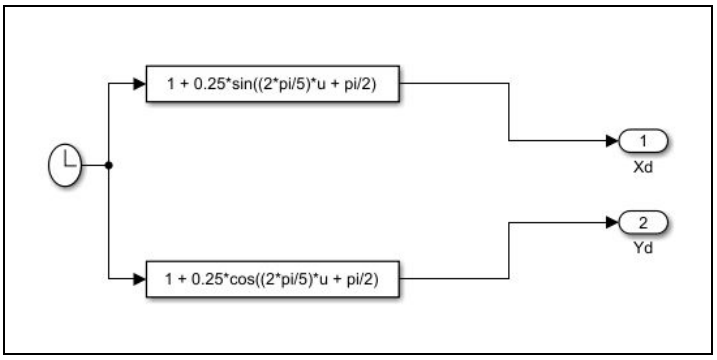
Multi-degree-of-freedom mechanisms are open loop kinematics chains used for reaching and positioning, such as robotics arms and back hoes. In general, multi-degree-of-freedom linkages offer greater ability to precisely position a link.

Code and SIMULINK Diagrams:

Main:



Trajectory equation:



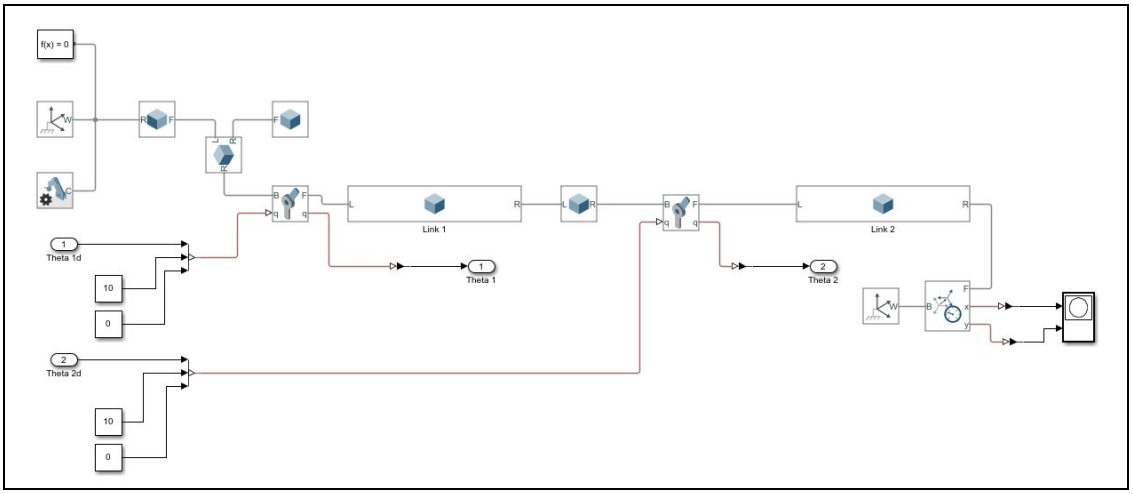
Inverse Kinematics Code:

```
function [Theta1,Theta2] = InverseKinematics(Xd,Yd)
l1 = 1;
l2 = 1;
Theta2 = acos((Xd^2+Yd^2 - l1^2 - l2^2)/(2*l1*l2));
s_Theta2 = sin(Theta2);
c_Theta2 = cos(Theta2);
Theta1 = atan2(Yd,Xd) - atan2(l2*s_Theta2, (l1+l2*c_Theta2));
```

Forward Kinematics Code:

```
function [X,Y] = Forwardkinematics(Theta1, Theta2)
l1 = 1;
l2 = 1;
T10 = [cos(Theta1), -sin(Theta1), 0, l1*cos(Theta1);
    sin(Theta1), cos(Theta1), 0, l1*sin(Theta1);
    0, 0, 1, 0;
    0, 0, 0, 1];
T21 = [cos(Theta2), -sin(Theta2), 0, l1*cos(Theta2);
    sin(Theta2), cos(Theta2), 0, l1*sin(Theta2);
    0, 0, 1, 0;
    0, 0, 0, 1];
T20 = T10*T21;
X = T20(1,4);
Y = T20(2,4);
```

Mechanical Model:



Explanation(Mechanical Model):

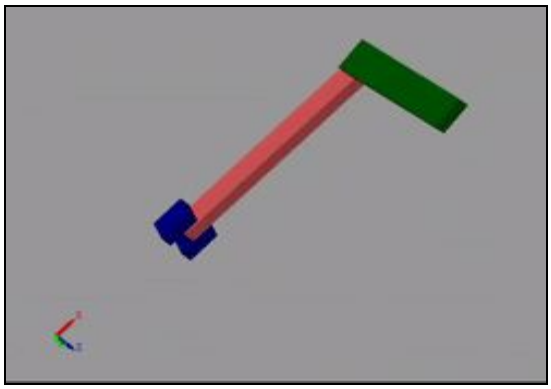
So our project has five main modules to handle the motion of the robotic arm. These are -

1) Trajectory model - So the trajectory model actually contains the trajectory of our robotic arm. It takes the input from the clock and using the sin and cos function it defines the x and y path for our robotic motion.
2) Inverse Kinematics - It contains the matrices and matlab function which takes the input as Xd and Yd from the trajectory model and feeds it to the inverse function which then calculates the theta and the required metrics and feed it to the Mechanical model of our robotic arm.
3) Mechanical Model - It contains the mechanical model of our robotic arm, from the arm, leg to each joint and support in the robot. Simulink provides very handy tools such as joints, sensors, links, functions, and constant support to manage all kinds of motions.
4) Forward Kinematics - Similar to inverse it also contains all the matrices and functions which take the input as theta from the mechanical model and the output as a calculated X and Y to move the arm.
5) Estimated position - It plots the graph in which the arm is actually moving. As initially we took some sin and cos function for the trajectory so now it will plot the actual and expected path of the robotic arm.

Result

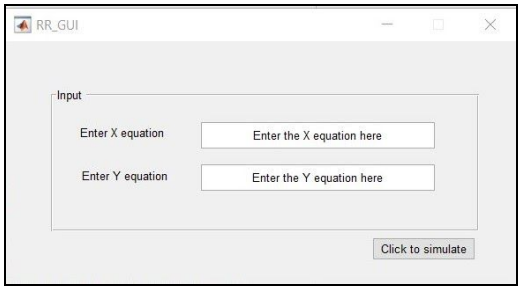
A 2-DoF robot arm simulation has been successfully implemented for kinematics learning. The manipulator robot is a simple 2-degree-of-freedom planar manipulator with revolute joints which is created in MATLAB. A trajectory is created in a 2-D plane and given as points to the inverse kinematics solver. The solver calculates the required joint positions to achieve this trajectory. Finally, the robot is animated to show the robot configurations that achieve the circular trajectory. Also we can check the expected path by the help of a plotted graph and validating it against the actual graph.

ROBOT ARM MOTION:

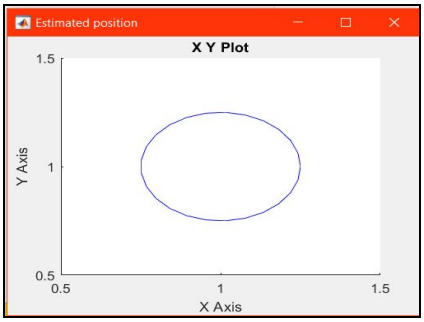


GUI Support -

As of now we created a basic GUI using the Guide library in Matlab which provides some handy tools to create easy UI. But unfortunately due to the lack in interoperability between slx and Guide files, made it hard to implement a dedicated GUI support to control the whole arm.



Estimated Position Plot:



CONCLUSION

The design and calculation of this mechanism is based on its own parameters and only tends to 2-DOF so that the required degree of freedom of different kinds of parameters can be designed and improved according to these mechanism's different characteristics. In future in this project, we will present GUI Robots, a software framework that enables the creation of tangible user interfaces for existing, unmodified GUI applications. This approach can be used to create a variety of tangible user interfaces, and can be used to enhance a wide range of existing software applications, including web browsers and 3D modeling tools. In future we will try to extend this project by adding a proper GUI support using the Guide library of MATLAB. So as of now we created the GUI but because of the lack of extensions in Guide and slx file which made interoperability difficult to handle.

REFERENCE

1) https://www.mathworks.com/help/fuzzy/modeling-inverse-kinematics-in-a-robotic-arm.html
2) https://www.researchgate.net/publication/341070239_Design_of_2-DOF_Robot_Manipulator/link/5eaba79692851cb267692aa4/download
3) https://www.mathworks.com/products/matlab.html
4) https://www.mathworks.com/matlabcentral/fileexchange/69756-forward-dynamics-of-planar-2-dof-robot-manipulator
5) https://www.ijeert.org/papers/v6-i11/3.pdf