Report 03

Forward and Inverse Kinematics for RP Robot

MCT 621

Motion Control and Servo Systems

Fall 2021

Submitted By/

Barakat Ibrahim Hasan Ibrahim

2002603

Introduction

In this Report, 4 blocks are introduced regarding the RP robot ARM studied in the lectures for MCT621-Fall 2021, the Four blocks are:

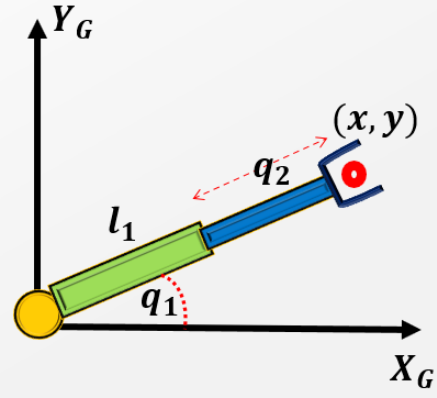
1. Forward Position Kinematics
2. Inverse Position Kinematics
3. Forward Velocity Kinematics
4. Inverse Velocity Kinematics

These Blocks are described in detail with the aid of mathematical equations behind them. Then simulation is done on each of them to get the desired output to test. And finally, each two blocks are tested among each other, so that the input is given to Forward Kinematics Block then gets out of it to the Inverse kinematics Block, and if the output is the identical to the input so that the two blocks are correct.

All simulations are done using MATLAB/Simulink, and the blocks are written as a MATLAB functions.

RP Robot Overview

The robot shown in the following figure Is a 2 DOF robot it's called RP Robot, R stands for Revolute joint, which is a motor, P stands for Prismatic which means linear actuator.



In this study q1 is the angle moved by the revolute joint measured in degrees. And q2 is the distance moved by the prismatic actuator measured in meter.

L1 is the joint length which is fixed as L1 = 0.5 m.

Position Kinematics

Position Kinematics means to calculate the End-Effector position from known actuators actions as angle for revolute and distance of actuation for prismatic for Forward. Or vice versa for reverse.

1. Forward Position Kinematics

From calculations the following 2 equations describes the Forward position kinematics for the RP robot.

$$X_{EE} = (q_1 + L_1)\cos(q_2)$$

$$Y_{EE} = (q_1 + L_1)\sin(q_2)$$

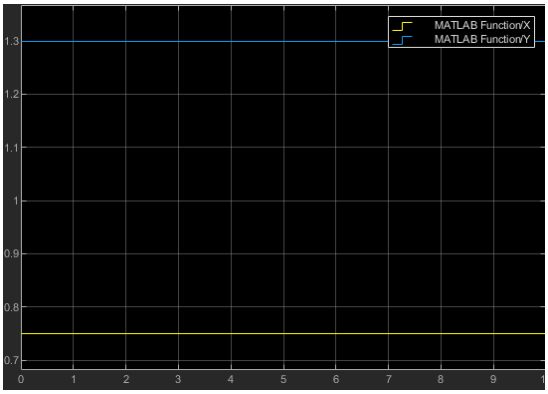
MATLAB Function code.

```
function [X,Y] = RP_POSITION_FW(q1,q2)
L1 = 0.5;
X = (L1+q2) * cosd(q1);
Y = (L1+q2) * sind(q1);
end
```

Simulink Testing Block Diagram.

The block is tested with inputs q1 = 60, q2 = 1m as follows

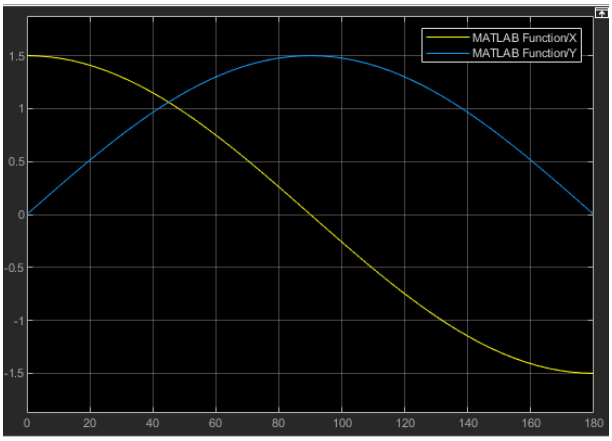
As seen in the opposite figure the End Effector has (X,Y) of (1.3 , 0.65) meter



Testing the Block for q1 as a ramp input, q2 = 1m as follows

As seen in the opposite figure the End Effector has forms a half circle as seen for x from 1.5 in the right to -1.5 in the left also for y starting from 0 up to 1.5 then to 0.

This is seen clearly in the XY Graph.



2. Inverse Position Kinematics

From calculations using the trigonometric approach the following 2 equations describes the Forward position kinematics for the RP robot.

$$q_1 = \tan^{-1}(\frac{Y}{X})$$

$$q_2 = \sqrt{X^2 + Y^2} - L_1$$

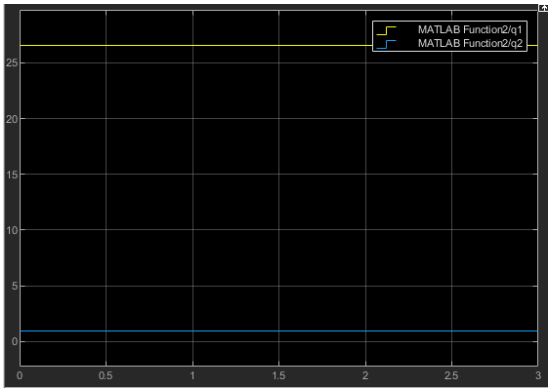
MATLAB Function code.

```
function [q1,q2] = RP_POSITION_INV(X,Y)
L1 = 0.5;
q1 = atand(Y/X);
q2 = sqrt(X^2 + Y^2) - L1;
end
```

Simulink Testing Block Diagram.

The block is tested with inputs q1 = 60, q2 = 1m as follows

As seen in the opposite figure the End Effector has the output of the previous inputs

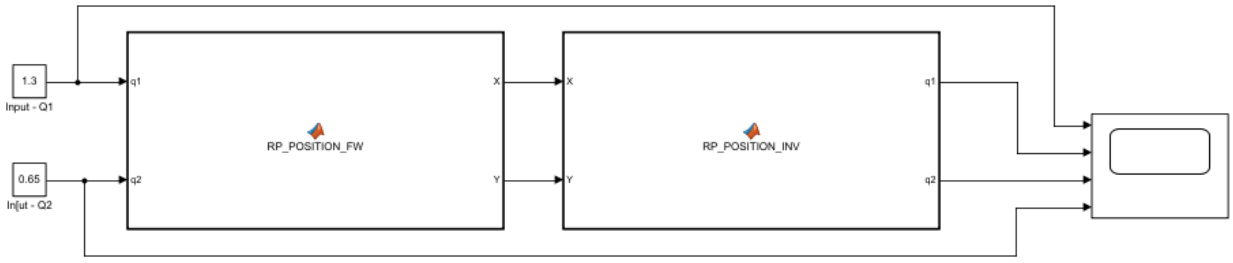


3. Forward & Inverse Position Kinematics

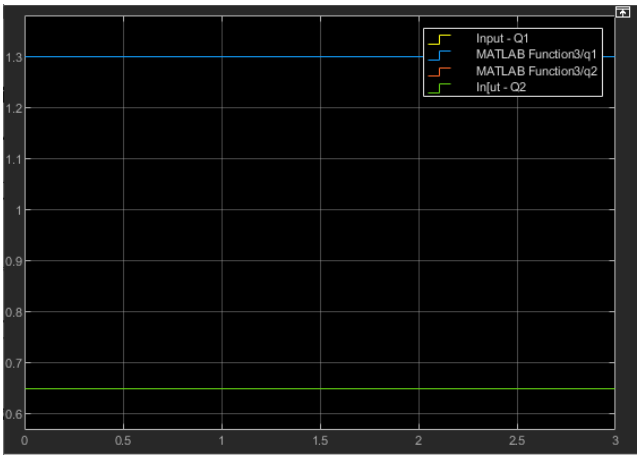
In the following part the 2 blocks are put back-to-back to test if they are correct or not.

Simulink Testing Block Diagram.

The block is tested with inputs q1 = 60, q2 = 1m as follows



As shown in the opposite figure the input is identical to the output, so that the two blocks are correct.



Velocity Kinematics

Velocity Kinematics means to calculate the End-Effector Velocities in X and Y directions from known actuators actions as angle's rate of change for revolute actuator and distance rate of change of for prismatic actuator for Forward kinematics. Or vice versa for reverse.

1. Forward Velocity Kinematics

From calculations using trigonometric approach with partial derivatives, the following 2 equations describes the Forward position kinematics for the RP robot.

$$\dot{X_{EE}} = -(L_1 + q_2)\sin(q_1)\dot{q_1} + \cos(q_1)\dot{q_2}$$

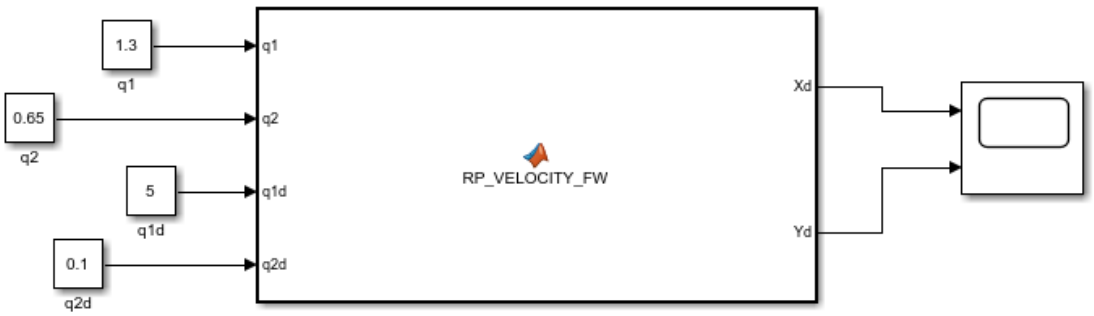
$$\dot{Y_{EE}} = (L_1 + q_2)\cos(q_1)\dot{q_1} + \cos(q_1)\dot{q_2}$$

MATLAB Function code.

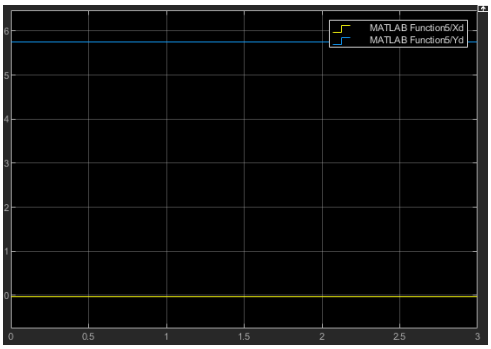
```
function [Xd,Yd] = RP_VELOCITY_FW(q1,q2,q1d,q2d)
L1 = 0.5;
Xd = -(L1+q2) * sind(q1)*q1d + cosd(q1)*q2d;
Yd = (L1+q2) * cosd(q1)*q1d + sind(q1)*q2d;
end
```

Simulink Testing Block Diagram.

The block is tested with inputs q1 = 60, q2 = 1m and q1d = 5 rad/s^2, q2d = 0.1 m/s^2



Shown in the opposite figure the output of the simulation resulting in the output velocity of the End-Effector in both X and Y directions.



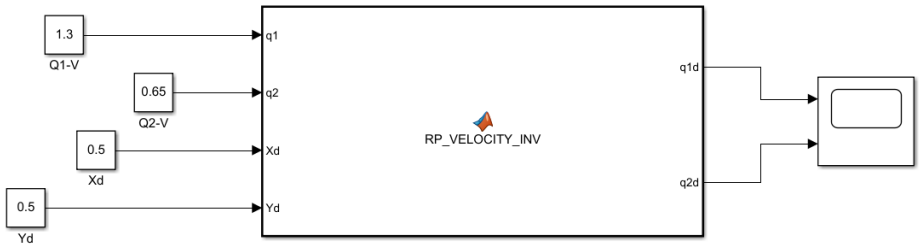
2. Inverse Velocity Kinematics

From calculations using the Jacobian matrix the following code can be generated.

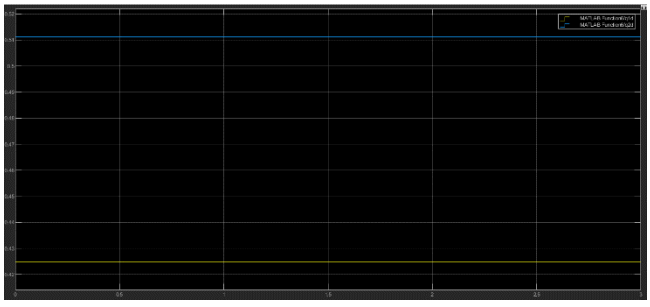
MATLAB Function code.

```
function [q1d,q2d] = RP_VELOCITY_INV(q1,q2,Xd,Yd)
L1=0.5;
J = [-(L1+q2)*sind(q1),cosd(q1);
      (L1+q2)*cosd(q1),sind(q1)];
  Joint_Vel = (J\[Xd;Yd]);
  q1d = Joint_Vel(1);
  q2d = Joint_Vel(2);
end
```

Simulink Testing Block Diagram.



As seen in the opposite figure the End Effector has the output of the previous inputs

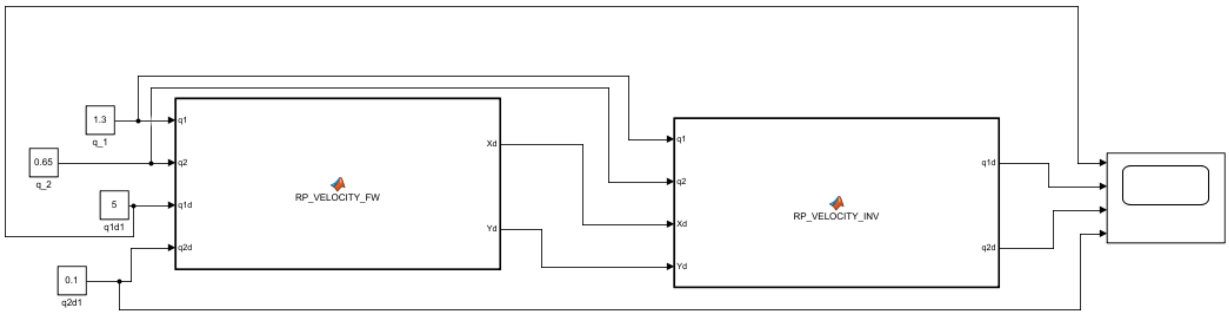


3. Forward & Inverse Velocity Kinematics

In the following part the 2 blocks are put back-to-back to test if they are correct or not.

Simulink Testing Block Diagram.

The Block is tested for the shown parameters



As shown in the opposite figure the input is identical to the output, so that the two blocks are correct.