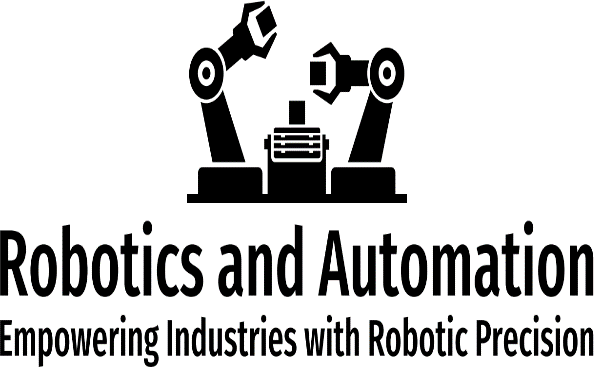


RO23421 - Mechanisms and Robotics laboratory

### (Regulation 2023)

Department of Robotics and Automation RO23421 - Mechanisms and Robotics laboratory



# Fourth Semester, Second Year

**Laboratory Manual**

RO23421 - Mechanisms and Robotics laboratory

Name

Batch

Roll No.

Department of Robotics and Automation RO23421 - Mechanisms and Robotics laboratory

Manual for Mechanisms and Robotics laboratory

# Quality Statements

#### COLLEGE VISION

* To be an institution of excellence in Engineering, Technology and Management Education & Research.
* To provide competent and ethical professionals with a concern for society.

#### COLLEGE MISSION

* To impart quality technical education imbibed with proficiency and humane values.
* To provide right ambience and opportunities for the students to develop into creative, talented, and globally competent professionals.
* To promote research and development in technology and management for the benefit of the society.

#### DEPARTMENT VISION

To be a department of excellence in academics, research and technological advancement in Robotics and Automation with a concern for society.

#### DEPARTMENT MISSION:

* To impart high technical knowledge, strong fundamentals, practical skills and creative
* knowledge for making successful professionals in Robotics and Automation.
* To foster students by infusing leadership qualities to become successful Engineer.
* To inculcate the entrepreneurial qualities for creating, developing and managing global engineering ventures.

**PROGRAMME EDUCATIONAL OBJECTIVES (PEOs):**

PEO I

To impart students with strong and comprehensive knowledge in the analytical, scientific and engineering fundamentals for solving engineering problems.

PEO II

To disseminate students with necessary skills, knowledge and leadership qualities for successful careers in industry.

PEO III

To instil students with technical expertise, Ethical practices and Team spirit and a concern towards greener society.

Manual for Robotics laboratory

**PROGRAM OUTCOMES (POs):**

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

Manual for Robotics laboratory

1. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
2. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
3. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

PSO 1: Design and develop efficient Automation system to enhance the quality of life by applying fundamentals of Basic Science, Mechanical and Mechatronics Engineering

PSO 2: Analyse and improve the performance of Manufacturing and Production system by implementing the Soft and hard Computing methods

PSO 3: Manage and lead a professional or an entrepreneur career in industries by applying modern Engineering, Management principles and best practices.

Manual for Robotics laboratory

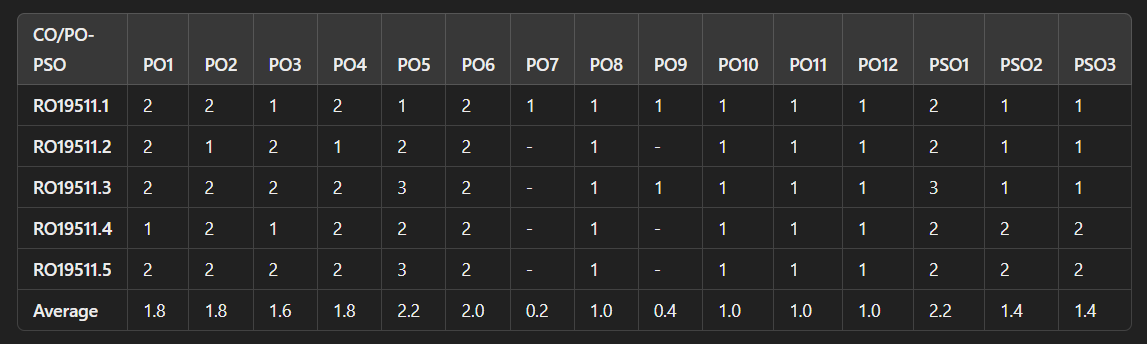
### Course Outcomes and CO-PO-PSO Mapping:

The following are the course outcomes for the laboratory course **RO19511.1**: Perform modelling of robot based on the link positions. **RO19511.2**: Determine the orientation and position of robot.

**RO19511.3**: Develop pick and place application system.

**RO19511.4**: Integrate PLC and Robot.

**RO19511.5**: Determine the joint torques of a robot.

The PO-CO mapping for the lab course is given as follows:

Manual for Robotics laboratory

**Safety Instructions for Robotics Laboratory**

**General Safety**

1. **No Food or Drinks: Do not bring food or drinks into the robotics lab to avoid spills and potential damage to equipment.**
2. **Clean Hands: Ensure hands are clean and dry before using any equipment to maintain hygiene and avoid damage.**
3. **Report Malfunctions: Immediately report any malfunctioning equipment or hazards to the lab supervisor.**

**Robot Safety (Mitsubishi & Dobot)**

1. **Operator Awareness: Always be aware of the robot's movements and ensure that no one is in the robot's operational area.**
2. **Emergency Stop: Know the location and operation of the emergency stop button for the robots and conveyor systems. Ensure it is functional before starting any task.**
3. **Safety Barriers: Do not bypass or remove safety barriers or guards around the robots.**
4. **Proper Handling: When manually handling or setting up the robots, use proper lifting techniques and follow manufacturer guidelines.**
5. **Personal Protective Equipment (PPE): Wear appropriate PPE, such as gloves and safety glasses, when working with robots and moving parts.**

**Conveyor Safety**

1. **Clear Pathways: Ensure that the area around the conveyor is clear of obstacles to prevent tripping or accidents.**
2. **No Loose Clothing: Avoid wearing loose clothing or accessories that could get caught in the conveyor or robot mechanisms.**
3. **Monitor Conveyor Operation: Stay attentive to the operation of the conveyor, and do not attempt to adjust or fix it while it is in motion.**

**Computer Safety**

1. **Proper Plug Handling: Always plug and unplug equipment by grasping the plug, not the cord, to prevent damage and electric shocks.**
2. **Avoid Overloading Outlets: Do not overload electrical outlets with multiple devices to prevent electrical fires.**
3. **Handle Equipment with Care: Do not force or roughly handle any computer components or**

Manual for Robotics laboratory

**accessories.**

1. **Save Work Frequently: Save your work regularly to avoid data loss in case of power outages or system crashes.**
2. **Use Approved Software: Only use software that is approved and installed by the lab administrator to prevent malware infections.**

**Emergency Procedures**

1. **Know Emergency Exits: Familiarize yourself with the location of emergency exits, fire extinguishers, and first aid kits.**
2. **Fire Safety: In case of a fire, evacuate immediately using the nearest exit and do not use elevators.**
3. **Report Injuries: Immediately report any injuries or accidents to the lab supervisor.**

**Security**

1. **Personal Belongings: Keep personal belongings secure and do not leave them unattended in the lab.**
2. **Log Out: Always log out of your accounts before leaving the workstation to protect your personal information.**

**COVID-19 and Hygiene**

1. **Sanitize Hands: Use hand sanitizer before and after using equipment.**
2. **Disinfect Equipment: Use disinfectant wipes to clean keyboards, mice, and other equipment before and after use.**
3. **Maintain Distance: Follow any posted social distancing guidelines and sit only at designated workstations.**

**Additional Tips**

* + **Be Considerate: Keep noise levels low to maintain a productive environment for everyone.**
  + **Follow Lab Rules: Adhere to any additional rules and guidelines specific to your robotics lab.**

Manual for Mechanisms and Robotics laboratory

# Table of Contents

**RO23421 - Mechanisms and Robotics laboratory**

|  |  |  |
| --- | --- | --- |
| **Experiment Number** | **Experiment Name** | **Page No** |
| 1 | DESIGN OF SLIDER CRANK MECHANISM USING ADAMS |  |
| 2 | DESIGN OF FOUR BAR MECHANISM USING ADAMS |  |
| 3 | DESIGN OF TOGGLE MECHANISM USING ADAMS |  |
| 4 | DETERMINATION OF MAXIMUM AND MINIMUM POSITION OF LINKS |  |
| 5 | TRAJECTORY CONTROL MODELING WITH INVERSE KINEMATICS USING SIMULINK |  |
| 6 | TRAJECTORY PATH PLANING OF 2R MANIPULATOR |  |
| 7 | CHECK FOR ENVIRONMENTAL COLLISIONS WITH MANIPULATOR |  |
| 8 | CHECK FOR WORLD COLLISION PAIR WITH 7-AXIS KINOVA GEN3 ROBOT |  |
| 9 | ROBOT PROGRAMMING AND SIMULATION FOR PICK AND PLACE |  |
| 10 | ROBOT PROGRAMMING AND SIMULATION FOR PICK AND PLACE – USING PROXIMITY SENSOR BY BLOCKLY METHOD |  |
| 11 | ROBOT PROGRAMMING AND SIMULATION FOR PICK AND PLACE – USING PROXIMITY SENSOR BY SCRIPT METHOD |  |
| 12 | DETERMINATION OF MAXIMUM AND MINIMUM POSITION OF LINKS – SPATIAL MANIPULATOR |  |
| 13 | INVERSE KINEMATICS OF KINOVA GEN3 ROBOT TO DRAW A CIRCULAR PATH |  |
| 14 | INVERSE KINEMATICS OF KINOVA GEN3 ROBOT TO DRAW A TRIANGULAR PATH |  |
| 15 | 6-AXIS MITSUBISHI ROBOT (INDUSTRIAL ROBOT) PROGRAMMING FOR PICK AND PLACE |  |

**Exp no: 1**

**DESIGN OF SLIDER CRANK MECHANISM USING ADAMS**

### Aim:

To design and develop slider crank mechanism using Adams

### Software Used:

* Adams View
* Adams Post Processor

### Procedure:

**Step 1. Creating the Model**

1. Start Adams/View
2. Create a new model.
3. Resize the working grid.
4. Open Coordinate Window
5. Create crank part from point (60, 0, 0) to (150, 0, 0)
6. Rename. slider crank. crank

### Step 2. Creating Revolution

1. Select Rigid Body: Revolution
2. Click the points
3. Right-click too close
4. Rename. slider crank. cylinder.

### Step 3. Creating Joints

1. Select Rigid body: Cylinder
2. Create piston part. Rename. slider crank. piston
3. Create revolute joints between crank and ground
4. Create spherical joint between cylinder and ground
5. Create translational joint between piston and cylinder.
6. Create Hooke joint between crank and piston
7. Add Rotational joint motion to revolute joint with function = -30deg - 60 \* time.

### Step 4. Create Point-to-Point Measure

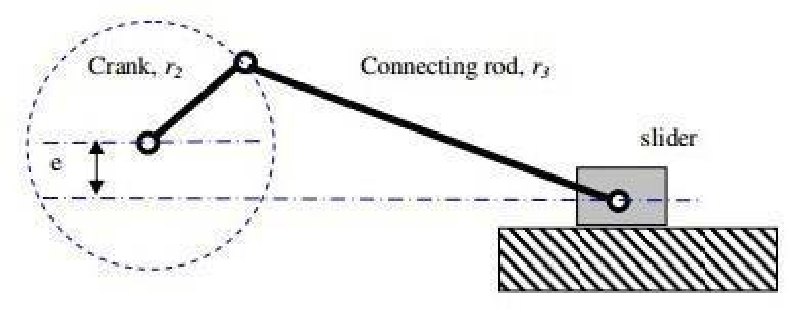
1. From Design Exploration ribbon, select Point to-Point Measure
2. Select Displacement as Characteristic
3. Select Global as Component.
4. Select the Marker at the left end of cylinder and the Marker at the left end of crank
5. Rename it as MEA\_PT2PT\_R

### Step 5. Create Point Measure

1. Under the piston tree in the Model Browser, nightsticks and select Measure
2. Select Translational Velocity and select Component.
3. Enter cylinder.cm as Represent coordinates
4. Select any Marker belongs to the ground as DO time derivatives in.
5. Repeat the above steps to create a translational acceleration

### Step 6. Create Angle about Axis Measure

1. In the Bodies tree, right-click the spherical joint between cylinder and the ground
2. Select info and remember the name of I Marker and Marker.
3. Close the info window.
4. Select Function Measure
5. Select Angle about Z under Displacement and enter the marker name in Step b.
6. cSelect angle as units



### Step 7. Testing the Model

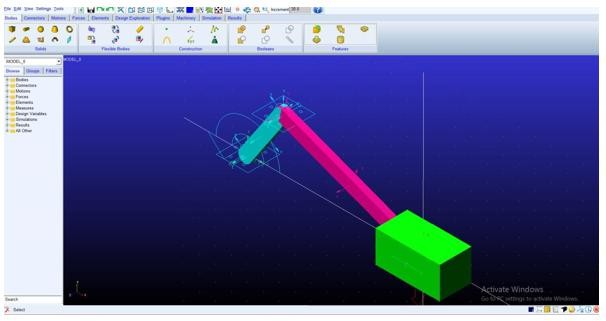
1. From Simulation ribbon, select Run an Interactive Simulation
2. Set End Time to 0.01 and Step Size to 0.001, and then click Start
3. Click Plotting
4. Use the Plot tracking tool
5. Follow the plot curve. Find the size measurement at X= 0.0

### Theory:

The Slider-crank mechanism is used to transform rotational motion into translational motion by means of a rotating driving beam, a connection rod and a sliding body. In the present example, a flexible body is used for the connection rod. The sliding mass is not allowed to rotate and three revolute joints are used to connect the bodies. While each body has six degrees of freedom in space, the kinematical conditions lead to one degree of freedom for thewhole system.

A slider crank mechanism converts circular motion of the crank into linear motion of the slider. In order for the crank to rotate fully the condition L> R+E must be satisfied where R isthe crank length, L is the length of the link connecting crank and slider and E is the offset of slider. A slider crank is a RRRP type of mechanism i.e. It has three revolute joints and 1 prismatic joint. The total distance covered by the slider between its two extreme positions is called the path length. Kinematic inversion of slier crank mechanisms produces ordinary a white work quick return mechanism

Fig: slide crank chain mechanism diagram



**ADAMS SIMULATION FOR SLIDER CRANK MECHANISM**

**OUTPUT:**

****

## RESULT:

The design and analysis of the Slider-Crank Mechanism were successfully verified using Adams simulation. The simulation results, including the displacement, velocity, and acceleration of the slider and crank components, were consistent with the theoretical expectations.

**Exp no: 2**

**DESIGN OF FOUR BAR MECHANISM USING ADAMS**

### Aim:

To design and develop four bar mechanism using Adams

### Software Used:

* Adams View
* Adams Post Processor

### Procedure:

**Step 1. Creating the Model**

1. Start Adams/View.
2. Create a new model. (Model Name = Four bar, Units= mms, Gravity = none)
3. Modify the spacing of the Working Grid (X = 10mm, Y= 10mm)
4. Click Units from Settings menu
5. Select Radian from Angle pull down menu
6. Click OK.

### Step 2. Create a Marker

1. Press F4 to Open Coordinate Window
2. From Bodies ribbon, select Construction Geometry: Marker
3. Create a marker at each of the following coordinates: O (0, 0, 0); A (-60, 80, 0); B (180,

180,0); C (180, 0, 0)

### Step 3. Create Links and Joints

1. From Bodies ribbon, double click Rigid Body Link
2. Create links OA, AB, and BC, using the markers as end points.
3. From Connectors ribbon, double click Create a Revolute joint
4. Make revolute joints between two links at points and B, and between link and ground at Oand C.

### Step 4. Add Motion

1. From Motions ribbon, select Rotational Joint Motion
2. Enter (1rad) in Rot. Speed text field
3. Select joint at point O

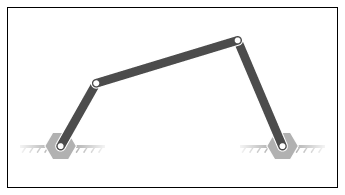
### Step 5. Testing the Model

1. From Simulation ribbon, select Run an Interactive Simulation
2. Set End Time to 10 and Step Size to 0.1
3. Click Start,
4. Click Plotting
5. Create a CM position plot for link OA in X component
6. Create a CM angular velocity plot for Linkable unlink in mag component
7. Use the Plot tracking tool
8. Follow the plot curve. Find the angular velocity at X =0.0

### Theory:

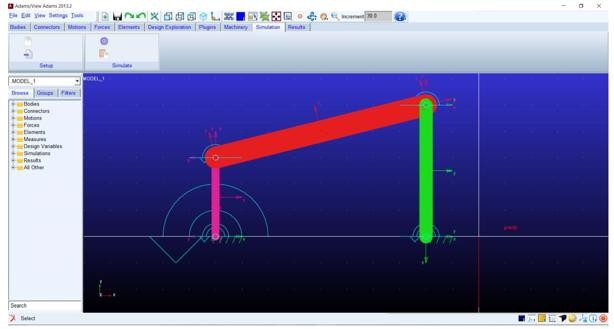
A **four-bar linkage**, also called a **four-bar**, is the simplest movable closed chain linkage. It consists of four bodies, called bars or links, connected in a loop by four joints. Generally, thejoints are configured so the links move in parallel planes, and the assembly is called a planarfour bar linkage.

If the linkage has four hinged joints with axes angled to intersect in a single point, then the links move on concentric spheres and the assembly is called a *spherical four-bar linkage* is aspatial four- bar linkage with hinged joints that have their axes angled in a particular way that



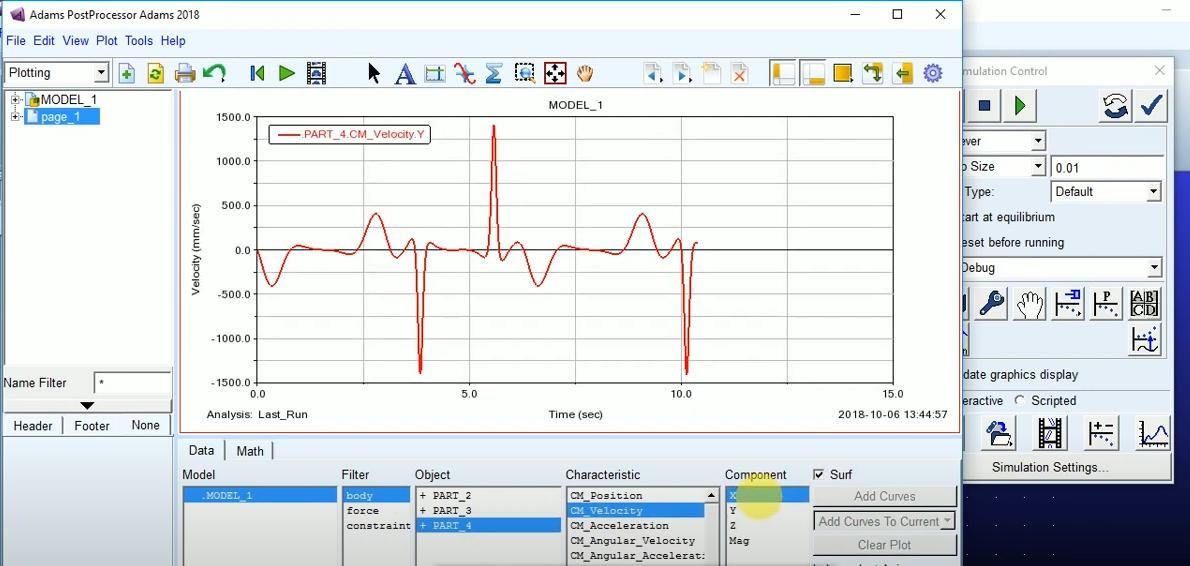
makes the system movable.

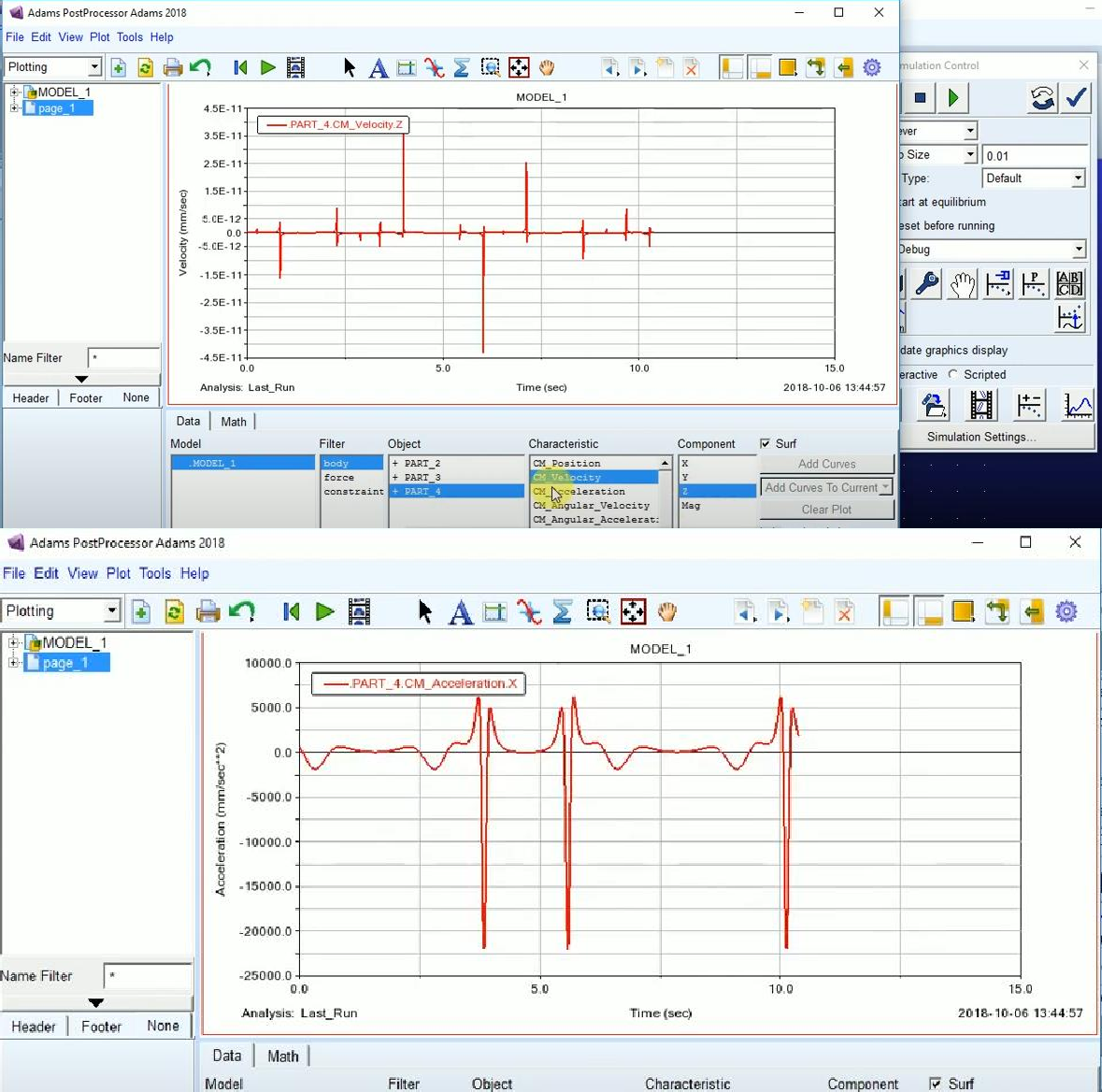
Chain bar mechanism drawing

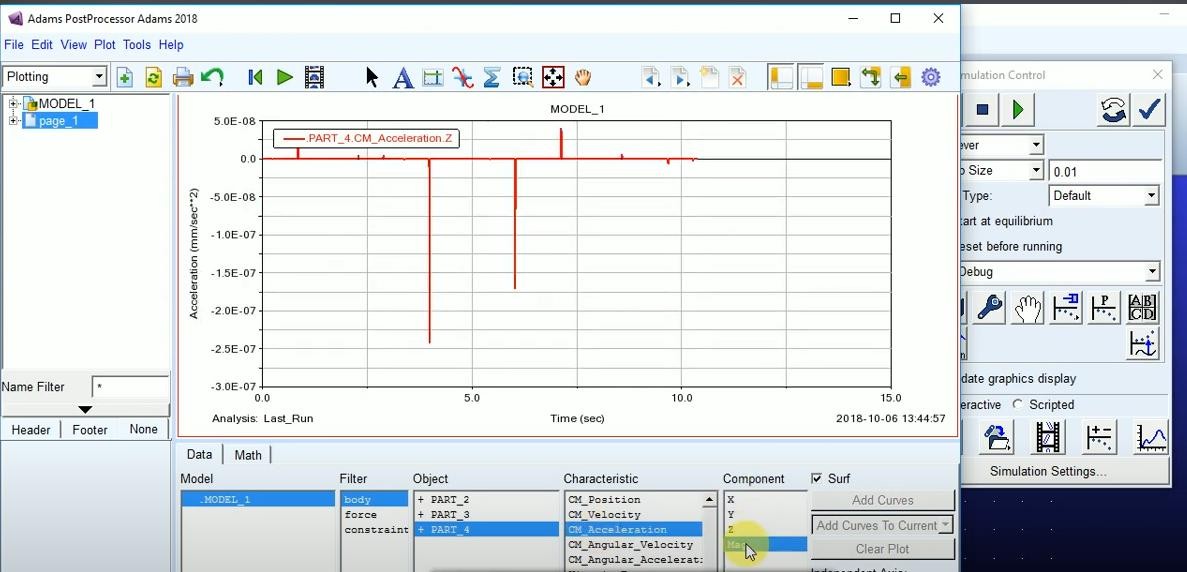


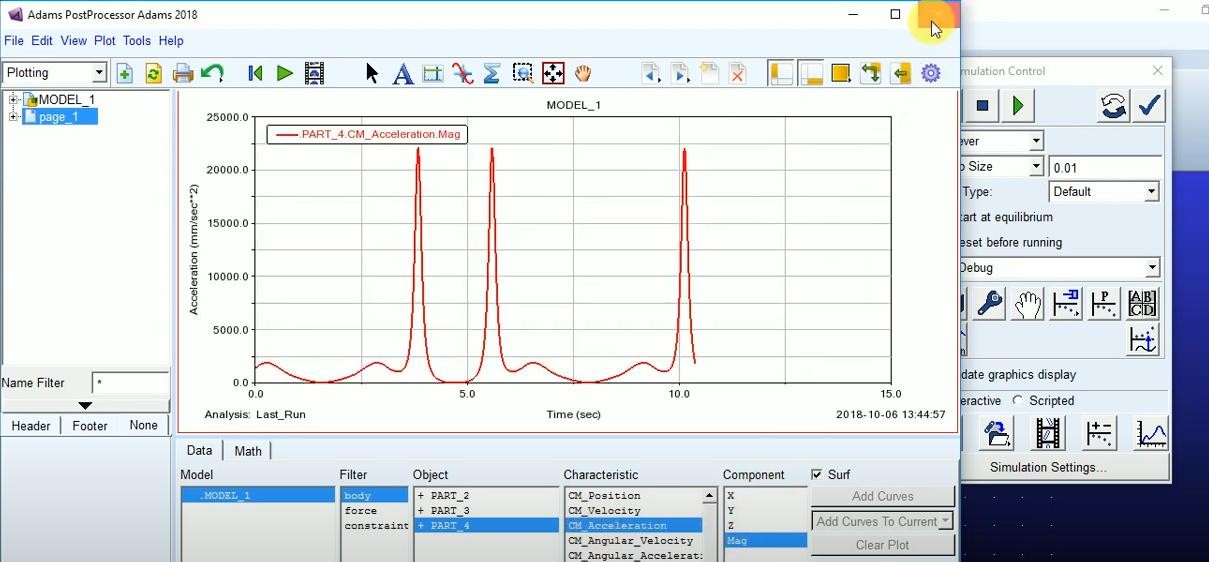
**ADAMS SIMULATION FOR FOUR BAR MECHANISM**

#### OUTPUT:

****



****



**RESULT:**

The design and analysis of the Four-Bar Mechanism were successfully verified using Adams simulation. The simulation results, including the angular displacements, velocities, and accelerations of the links, align with the expected theoretical values, confirming the accuracy of the design.

**Exp no: 3**

**DESIGN OF TOGGLE MECHANISM USING ADAMS**

### Objective:

To simulate the toggle mechanism in Adams software and analyze its motion to understand its kinematic and dynamic behavior.

### Apparatus Required:

1. Adams software
2. Computer system with required specifications

### Theory:

A **toggle mechanism** consists of two arms connected by a pin joint and attached to a fixed frame at one end. The other end is moved using an input force or displacement, causing the toggle to extend or retract. These mechanisms are commonly used in presses and clamps for generating large forces with small input motion.

### Applications:

Toggle mechanisms are used in applications such as:

1. Punching presses.
2. Mechanical clamps.
3. Force amplifiers in machinery.

### Procedure:

1. **Modeling the Toggle Mechanism in Adams:**
   * Open Adams software and create a new model.
   * Define the parts:
     + Create links representing the fixed base, input arm, coupler, and output arm.
   * Assemble the parts using joints:
     + Revolute joints for pin connections.
     + Fixed joint for the ground link.
   * Specify the dimensions of the links.

### Applying Motion and Forces:

* + Define the input motion (rotational or translational) to the driving arm.
  + Apply any forces or torques as needed.

### Running the Simulation:

* + Run a kinematic simulation to observe the motion of the mechanism.
  + Perform a dynamic analysis to determine forces and torques during operation.

### Extracting Results:

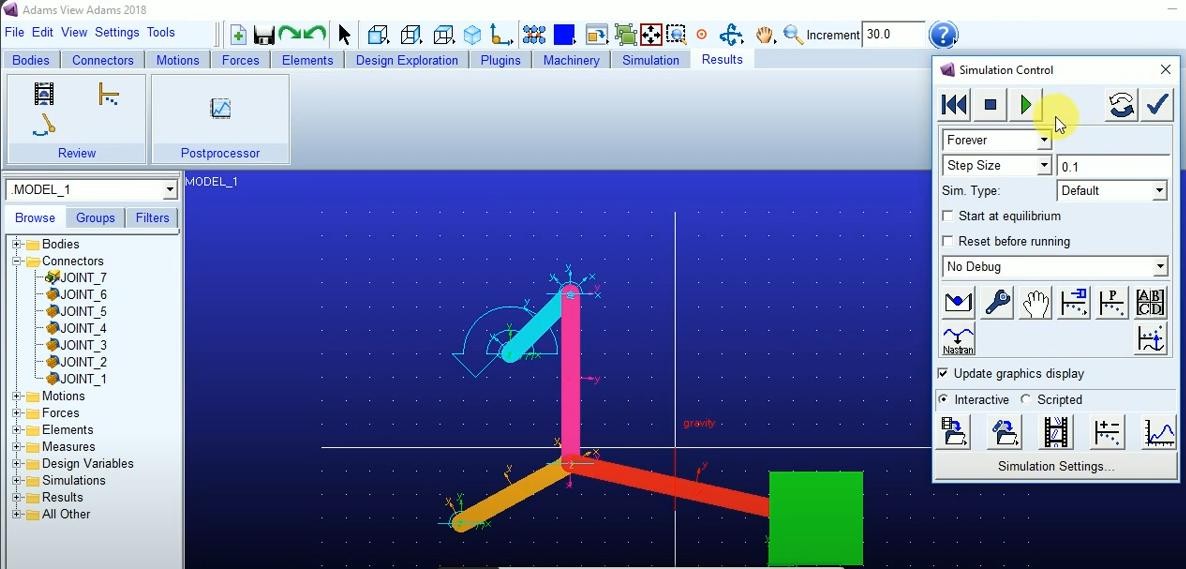
* + Plot the displacement, velocity, and acceleration of the links.
  + Record the forces at joints and the reaction forces.

### Observations:

1. Record the input motion and corresponding output displacement.
2. Observe the relationship between the input force and the output force.
3. Analyze the efficiency of force amplification.

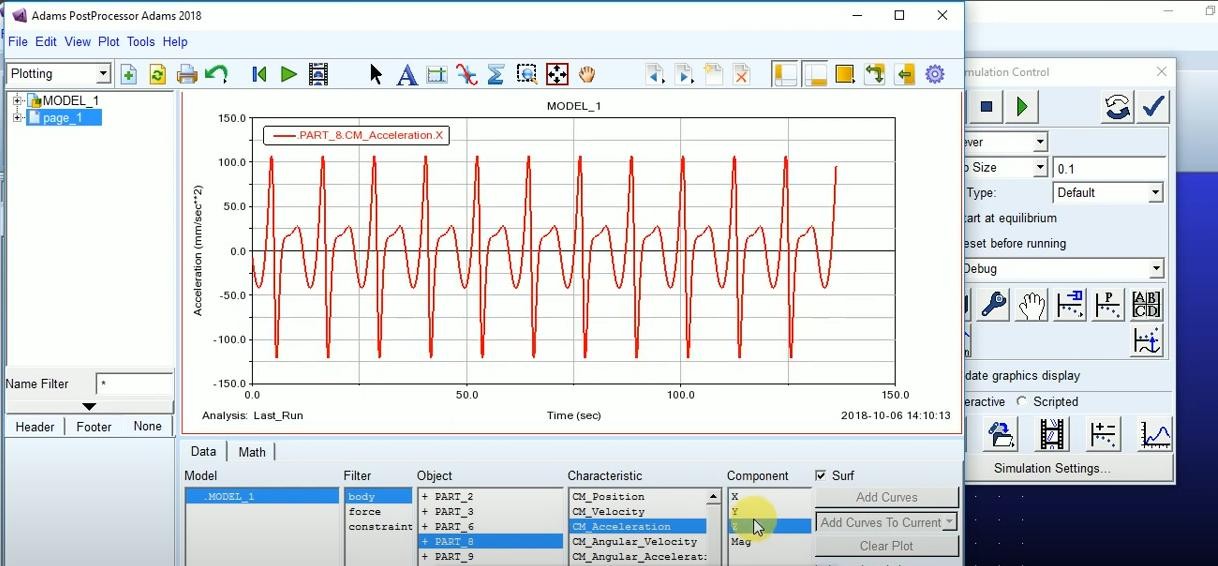
### Calculations (Manual Verification):

1. Use vector loop equations to calculate the kinematic parameters of the toggle mechanism.
2. Verify Adams results by comparing the displacement, velocity, and acceleration of the links.



## SIMULATION OF TOGGLE MECHANISM

**OUTPUT:**

****

### Result:

The simulation and analysis of the toggle mechanism were successfully completed. The kinematic and dynamic parameters, including displacement, velocity, acceleration, and force analysis, were obtained and verified.

**Exp no: 4**

[**DETERMINATION**](https://www.rajalakshmicolleges.net/moodle/course/view.php?id=75&section-1) **OF MAXIMUM AND MINIMUM POSITION OF LINKS**

**AIM**:

To determine the maximum and minimum position of links using MATLAB.

## SOFTWARE REQUIRED:

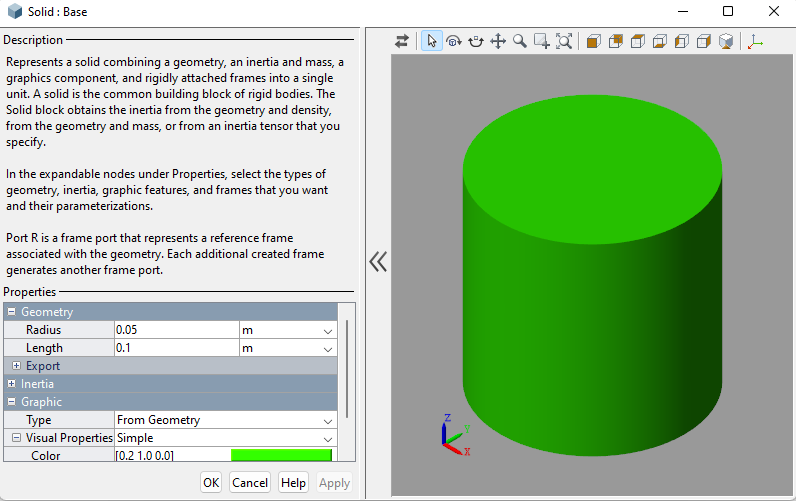
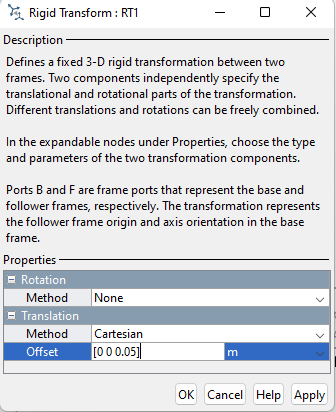
MATLAB R2021 (Robotics system toolbox and Simscape toolbox).

## PROCEDURE:

1. Open Matlab.
2. In home select Simulink>Blank workspace>Create Model.
3. Open Library browser.
4. Select Simscape> Foundation Library>Utilities>PS-Simulink converter(2), Simulink – PS Converter(2), Solver Configuration(1).
5. Select Simscape>Multibody>Utilities>Mechanism Configuration.
6. Select Simscape>Multibody>Frames and transforms>World Frame(1), Rigid transform(7).
7. Select Simscape>Multibody>Body elements>Brick Solid(3), Cylindrical solid(1).
8. Select Simscape>Multibody>Joints>Revolute joint(2), Weld joint(1).
9. Rename and connect the components as shown in the figure(1).

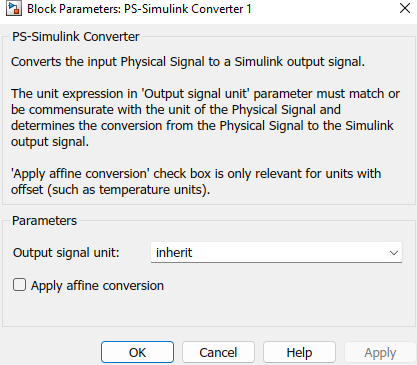
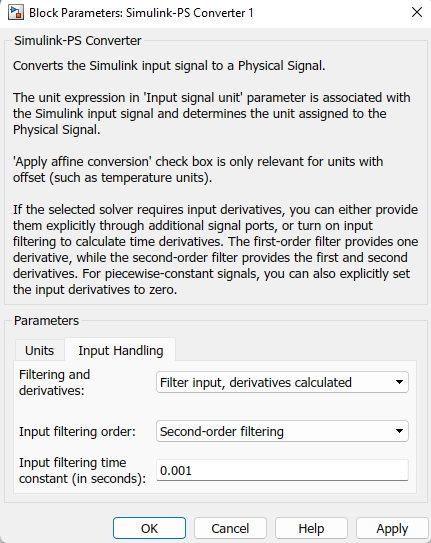
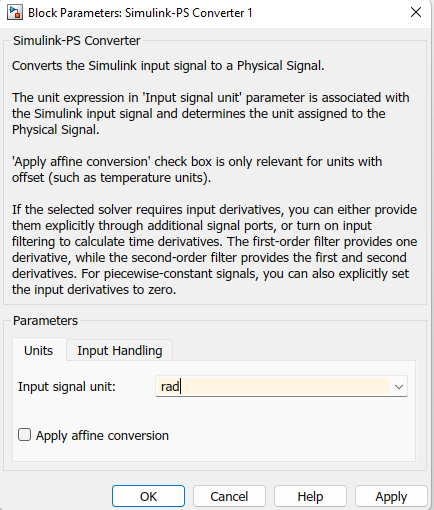
Note: Invert the component (Base, Link – 1,2,3) using ctrl+I command.



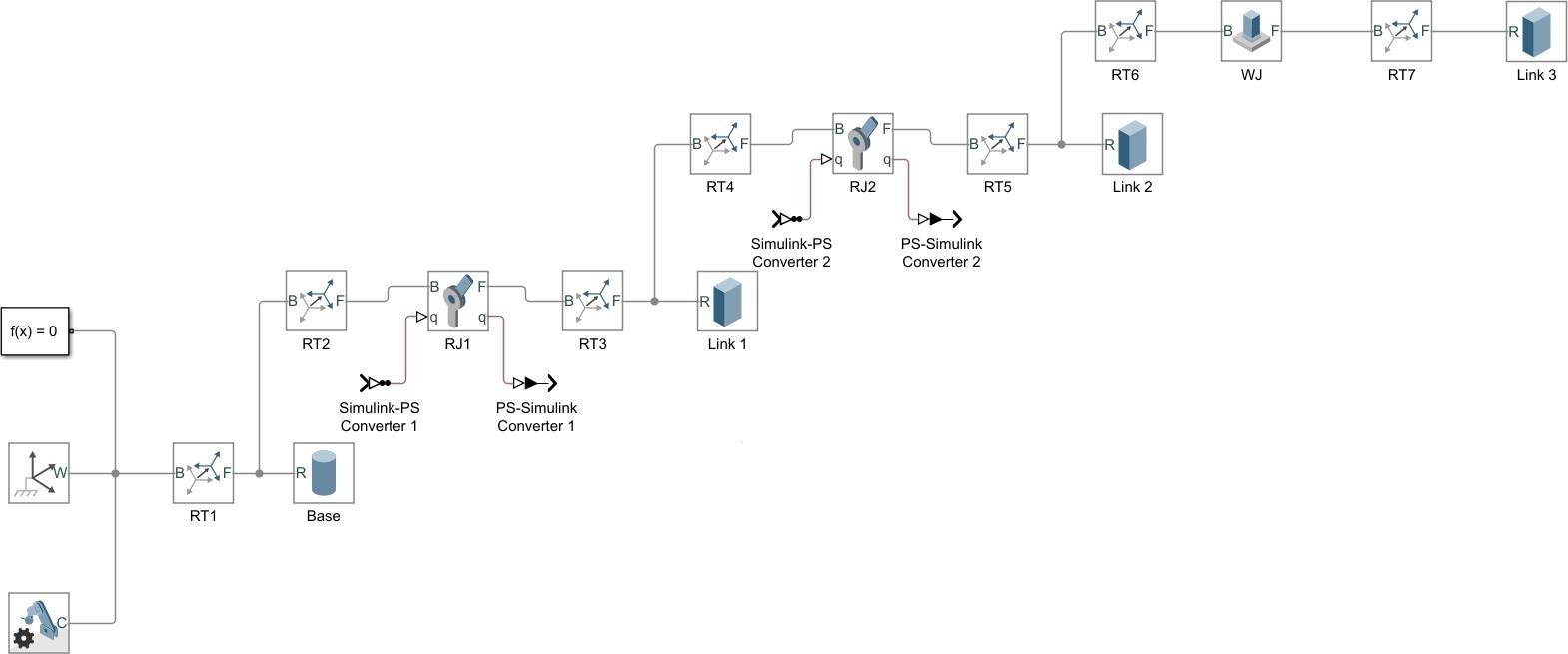
1. Double click the RT1. Under the translation section change the offset as [0 0 0.05].
2. Double click the Base. Under the geometry section change the radius as 0.05m and length as 0.1m. Choose any colour from Graphic >Visual Properties>Colour to distinguish between links. Similarly change the parameters of the links with respect to their dimensions.
3. Similarly change the parameters of other rigid transforms corresponding to the dimensions.
4. Double click RJ1. Under Limits>Specify the bound for lower limit as -90 and upper limit as 90 Under actuation> Torque – Automatically computed and Motion – Provided by input

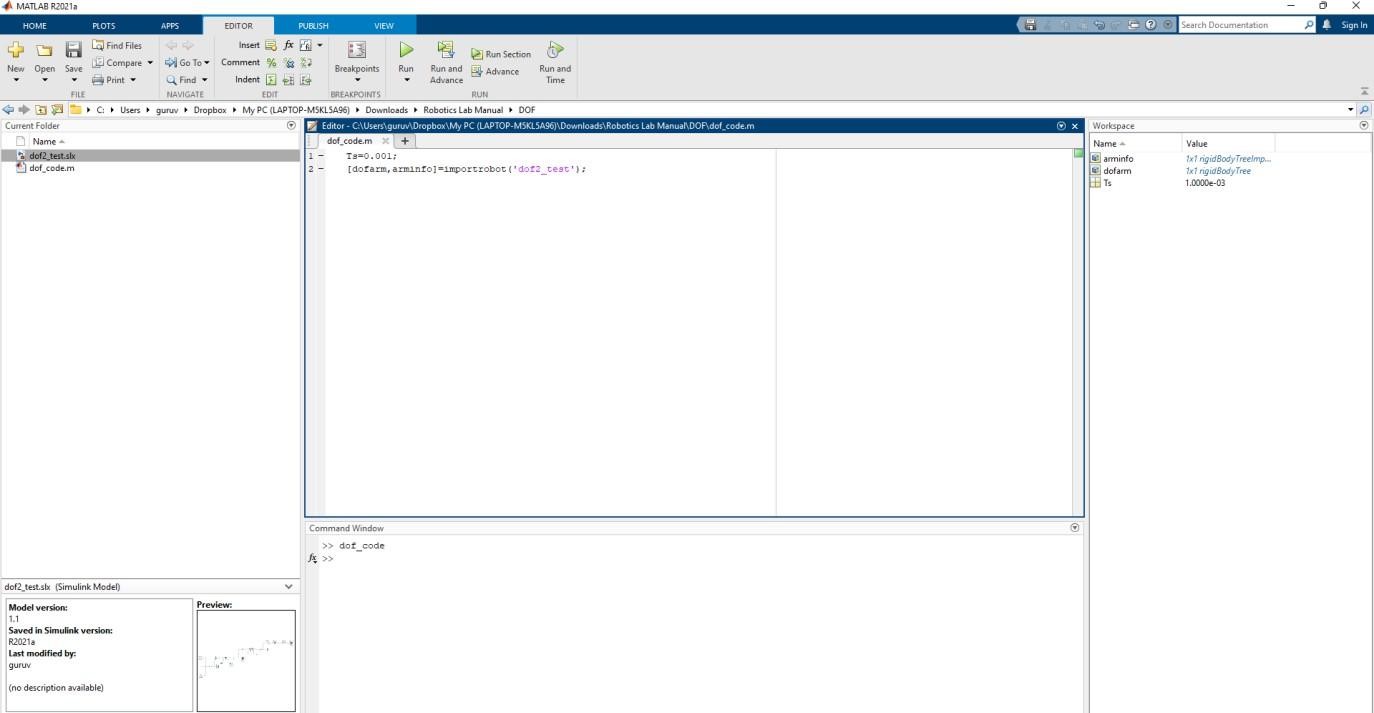
Under sensing> Check only the position Select apply>Ok

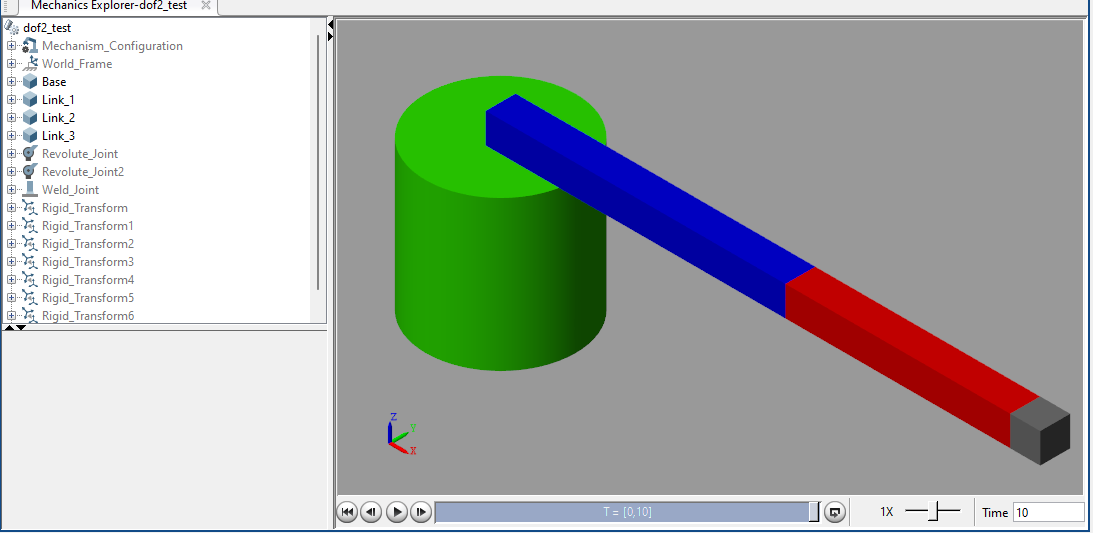
1. Similarly for RJ2, set the bound lower limit as -60 and upper limit as 60.
2. Change the parameters for Simulink-PS converter and PS-Simulink converter.



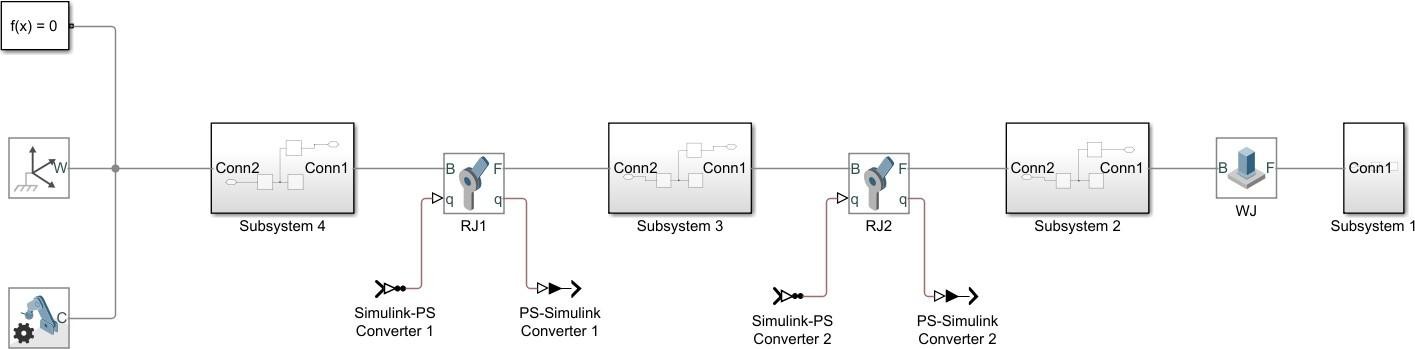
1. Connect input of RJ to the Simulink-PS converter and the output to PS-Simulink converter.



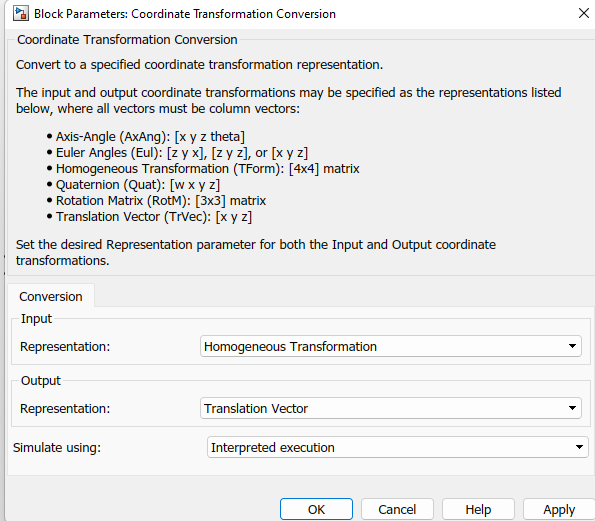
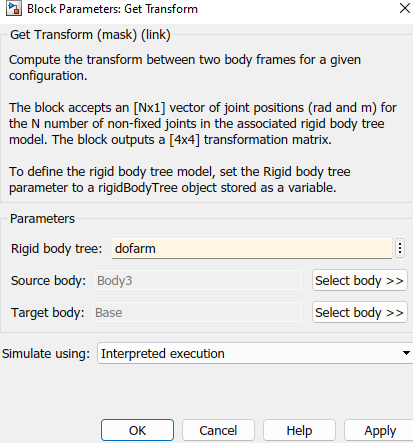
1. Save the file in .slx format.
2. Open the Matlab> Load the location of saved Simulink file> New Script> enter the code given below in the editor window. In importrobot() command, enter the name of the Simulink file you stored previously.
3. Save the code in .m format and run, also run the Simulink file.
4. Observe the graphical output of the robot link in various standard views.



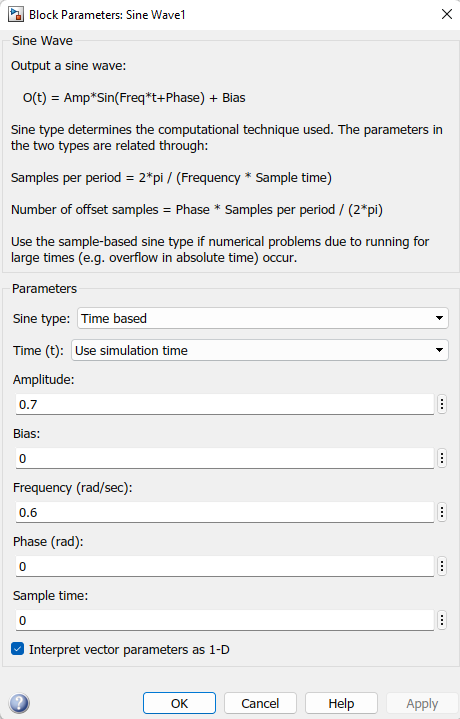
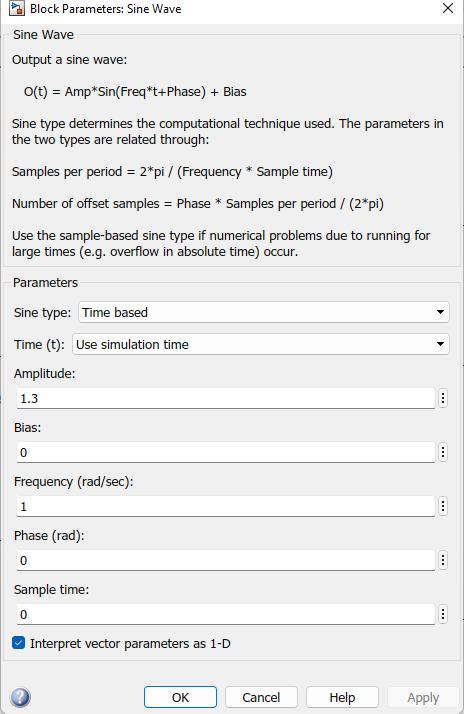
1. Open new Simulink file and copy-paste all the components from the previous Simulink file.
2. Group RT7 and Link3 using Ctrl+G command. A subsystem will be created.
3. Group RT5, Link 2, RT6.
4. Similarly create subsystem for Link 1 and Base.



1. Create a new subsystem grouping all the components in the system. Rename the subsystem as ‘Robot’, input as Q1 and Q2, output as Q1M and Q2M.
2. Open library browser>Robotics system toolbox>Manipulator Algorithm>Get transform.
3. Open library browser>Robotics system toolbox>Utilities>Coordinate transformation conversion.
4. Change the Parameters of the get transform and coordinate transformation conversion.



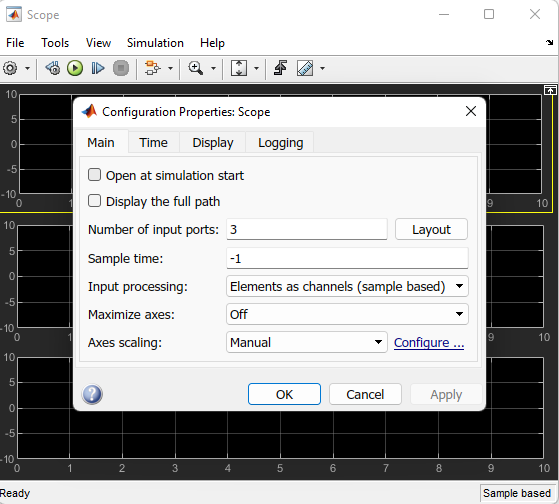
1. Connect the get transform and coordinate transformation conversion and group them, name the subsystem as ‘Forward Kinematics’.
2. Search Sine Wave(2), Mux(3), Scope(1).
3. Change the Paramaters of Sine Wave.



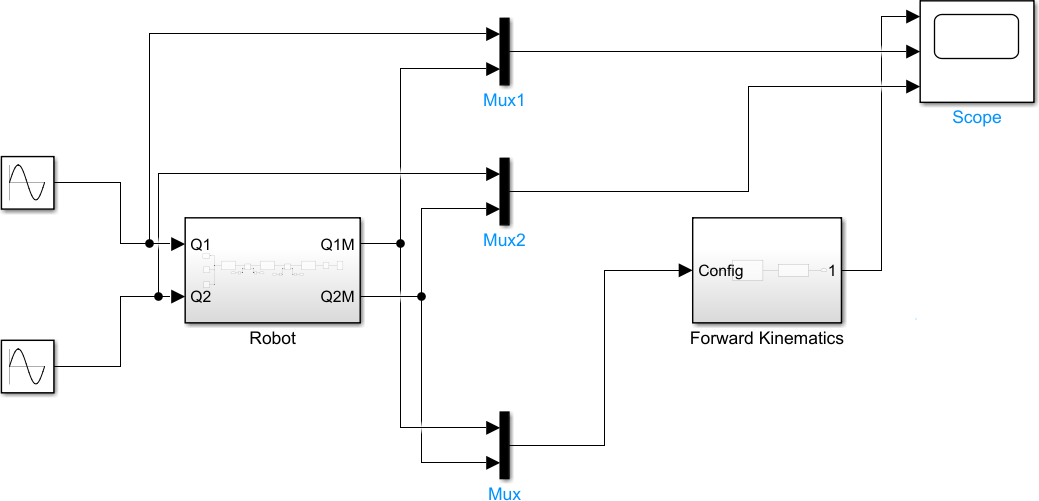
1. Change the properties of scope by double clicking the scope and selecting the setting icon.

Change

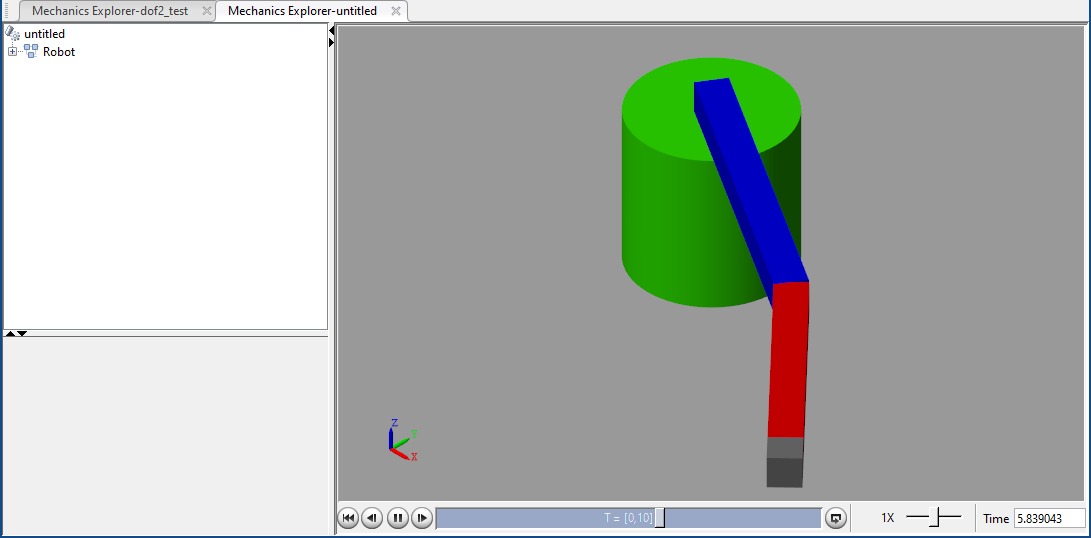
the Number of input ports (3 inputs).



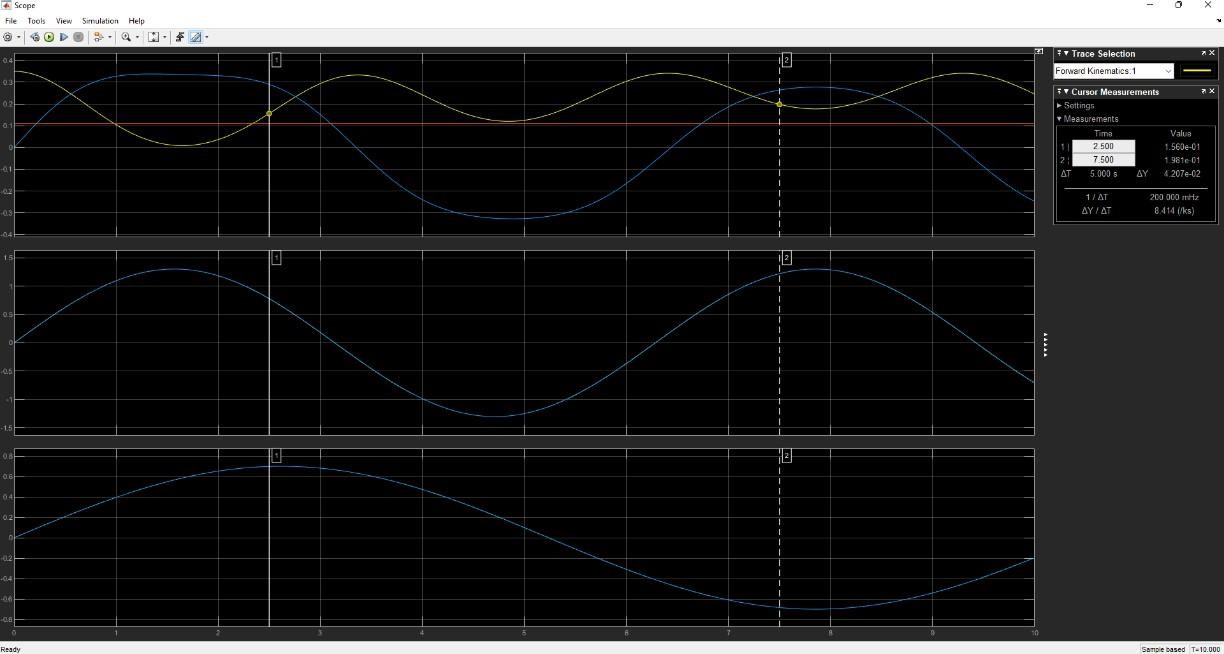
1. Make the connections as shown in the figure.



1. Run the Simulink and observe the forward kinematics of the robot.



1. Run scope and the observe the graph, Click the scale icon to obtain output at the particular instant.



## RESULT:

Thus the determination of maximum and minimum position of links using the MATLAB R2021a was done successfully.

**TRAJECTORY CONTROL MODELING WITH**

**Exp no: 5**

**INVERSE KINEMATICS USING SIMULINK**

## AIM:

To design a trajectory control modeling with inverse kinematics using MATLAB.

## SOFTWARE REQUIRED:

MATLAB R2021a.

## PROCEDURE:

1. Take rigid body tree from the forward kinematics.
2. Insert Signal Builder(1), MUX(1), Coordinate Transformation Conversion(1), Constant(2), Inverse Kinematics(1), Demux(1), Terminator(1).
3. Double click Signal Builder.
4. Delete the pre-existing Signal 1.
5. Insert a new signal by:

Signal> New>Custom>Enter Time values=[0 1 2 3 4 5]; Y values=[0.35 0.25 0.25 0.15 0.15

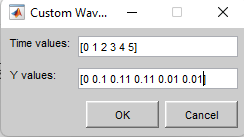
0.25].

Rename the signal as ‘X’



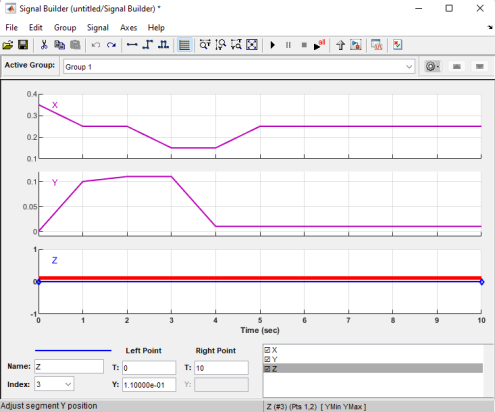
1. Insert another signal by:

Signal>New Custom>Enter Time values=[0 1 2 3 4 5]; Y values=[0 0.1 0.11 0.11 0.01 0.01]. Rename the signal as ‘Y’

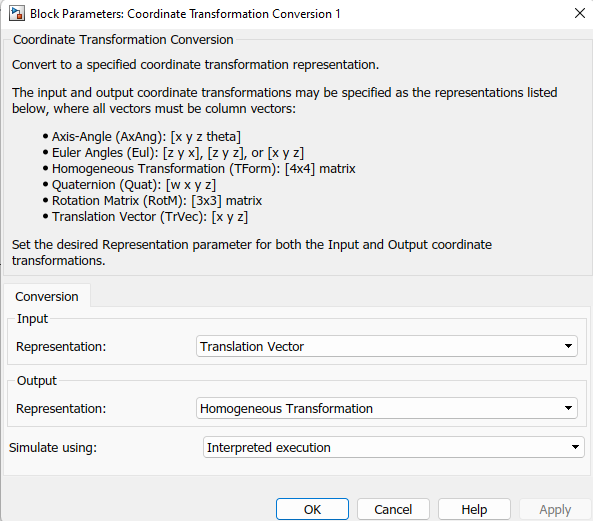


1. Insert another signal by:

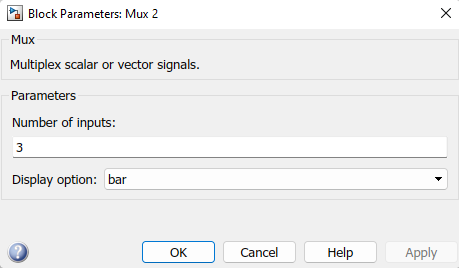
Signal>New >Constant>Rename it as ‘Z’. Left Point>Y: 0.11 ; Right Point>T: 10.



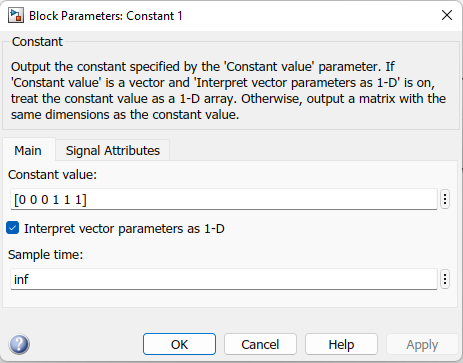
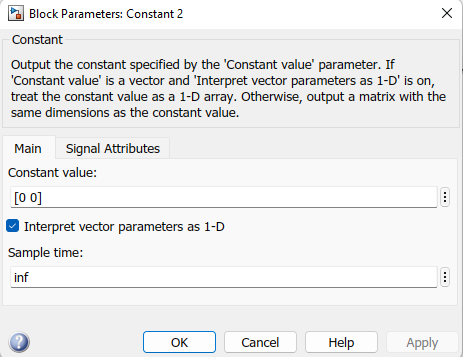
1. Change the parameters of Coordinate Transform Conversion as shown in figure.



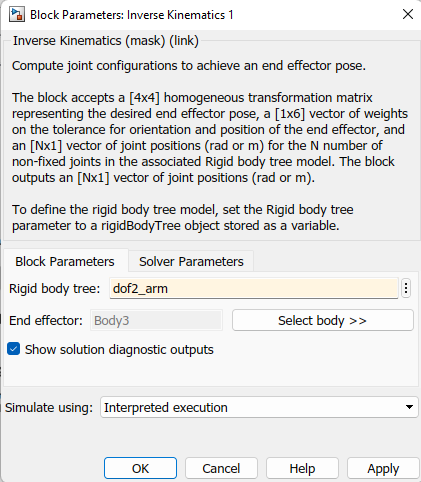
1. Change the no of inputs of the MUX(3 inputs).



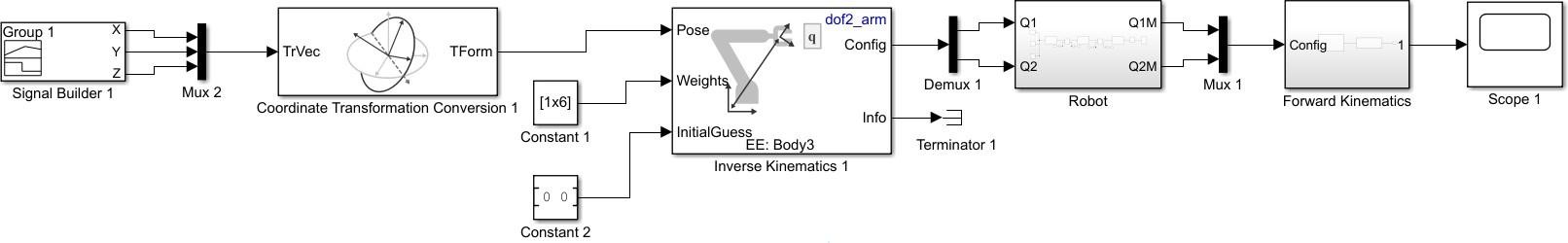
1. Rename the constants as Constant 1 and Constant 2 and change the values respectively.

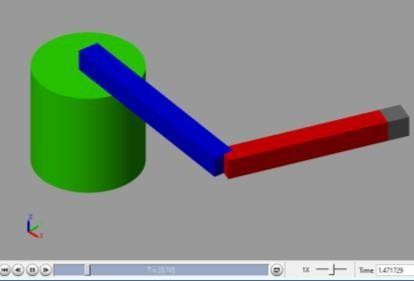
1. Change the parameters of Inverse Kinematics as shown in figure.



1. After changing the parameters of all the components, modify the connections as shown in the figure.



1. Run the Simulink and observe the output.



## RESULT:

Thus the trajectory control modeling with inverse kinematics using MATLAB R2021a was done successfully.

**Exp no: 6**

**TRAJECTORY PATH PLANING OF 2R MANIPULATOR**

## AIM:

To design trajectory path planning of 2R manipulator using MATLAB.

## SOFTWARE REQUIRED:

MATLAB R2021a.

## PROCEDURE:

1. Open the MATLAB R2021a software.
2. Go to Home > New script. A new editor window will open.
3. Type the program in the editor window.
4. Save the MATLAB file.
5. Go to editor > Run.
6. Observe the output.

## PROGRAM:

clc clear all

robot = rigidBodyTree('DataFormat','column','MaxNumBodies',3);

L1 = 0.3;

L2 = 0.3;

body = rigidBody('link1');

joint = rigidBodyJoint('joint1', 'revolute'); setFixedTransform(joint,trvec2tform([0 0 0]));

joint.JointAxis = [0 0 1]; body.Joint = joint; addBody(robot, body, 'base');

body = rigidBody('link2');

joint = rigidBodyJoint('joint2','revolute'); setFixedTransform(joint, trvec2tform([L1,0,0])); joint.JointAxis = [0 0 1];

body.Joint = joint;

addBody(robot, body, 'link1');

body = rigidBody('tool');

joint = rigidBodyJoint('fix1','fixed'); setFixedTransform(joint, trvec2tform([L2, 0, 0])); body.Joint = joint;

addBody(robot, body, 'link2'); showdetails(robot)

t = (0:0.2:10)'; % Time

count = length(t); center = [0.2 0.1 0];

radius = 0.15;

theta1 = t\*(2\*pi/t(end)); theta2 = t\*(2\*pi/t(end));

points = center + radius\*[cos(theta2) sin(theta1) zeros(size(theta1))];

q0 = homeConfiguration(robot); ndof = length(q0);

qs = zeros(count, ndof);

ik = inverseKinematics('RigidBodyTree', robot); weights = [0, 0, 0, 1, 1, 0];

endEffector = 'tool';

qInitial = q0; % Use home configuration as the initial guess for i = 1:count

% Solve for the configuration satisfying the desired end effector

% position

point = points(i,:);

qSol = ik(endEffector,trvec2tform(point),weights,qInitial);

% Store the configuration qs(i,:) = qSol;

% Start from prior solution qInitial = qSol;

end

figure show(robot,qs(1,:)'); view(2)

ax = gca;

ax.Projection = 'orthographic'; hold on plot(points(:,1),points(:,2),'r')

axis([-0.1 0.9 -0.3 0.5])

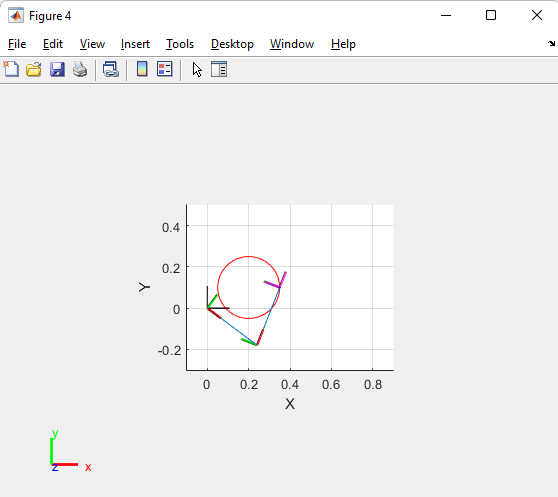
framesPerSecond = 15;

r = rateControl(framesPerSecond); for i = 1:count

show(robot,qs(i,:)','PreservePlot',false); drawnow

waitfor(r); end

## OUTPUT:

****

**RESULT:**

Thus the trajectory path planning of 2R manipulator using MATLAB R2021a was done successfully.

**CHECK FOR ENVIRONMENTAL COLLISIONS WITH**

**Exp no: 7**

**MANIPULATOR**

## AIM:

To check for the environmental collisions with manipulator using MATLAB.

## SOFTWARE REQUIRED:

MATLAB R2021a.

## PROCEDURE:

1. Open the MATLAB R2021a software.
2. Go to Home > New script. A new editor window will open.
3. Type the program in the editor window.
4. Save the MATLAB file.
5. Go to editor > Run.
6. Observe the output.

## PROGRAM:

clc clear all

% Create two platforms

platform1 = collisionBox(0.5,0.5,0.25) platform1.Pose = trvec2tform([-0.5 0.4 0.2])

platform2 = collisionBox(0.5,0.5,0.25); platform2.Pose = trvec2tform([0.5 0.2 0.2]);

% Add a light fixture, modeled as a sphere lightFixture = collisionSphere(0.1); lightFixture.Pose = trvec2tform([.2 0 1]);

% Store in a cell array for collision-checking worldCollisionArray = {platform1 platform2 lightFixture};

% ax = exa(worldCollisionArray);

% ax = exampleHelperVisualizeCollisionEnvironment(worldCollisionArray); figHandle = figure;

% Show the first object show(worldCollisionArray{1});

% Get axis properties and set hold ax = gca;

hold all;

% Show remaining objects

for i = 2:numel(worldCollisionArray) show(worldCollisionArray{i}, "Parent", ax);

end

% Set axis properties axis equal;

robot = loadrobot("kinovaGen3","DataFormat","column","Gravity",[0 0 -9.81]);

% Show the robot in the environment using the same axes as the collision objects.

% The robot base is fixed to the origin of the world.

show(robot,homeConfiguration(robot),"Parent",ax);

startPose = trvec2tform([-0.5,0.5,0.4])\*axang2tform([1 0 0 pi]);

endPose = trvec2tform([0.5,0.2,0.4])\*axang2tform([1 0 0 pi]);

% Use a fixed random seed to ensure repeatable results rng(0);

ik = inverseKinematics("RigidBodyTree",robot); weights = ones(1,6);

startConfig = ik("EndEffector\_Link",startPose,weights,robot.homeConfiguration); endConfig = ik("EndEffector\_Link",endPose,weights,robot.homeConfiguration);

% Show initial and final positions show(robot,startConfig); show(robot,endConfig);

[q,qd,qdd,t] = trapveltraj([homeConfiguration(robot),startConfig,endConfig],200,"EndTime",2);

% Loop through the other positions for i = 1:length(q)

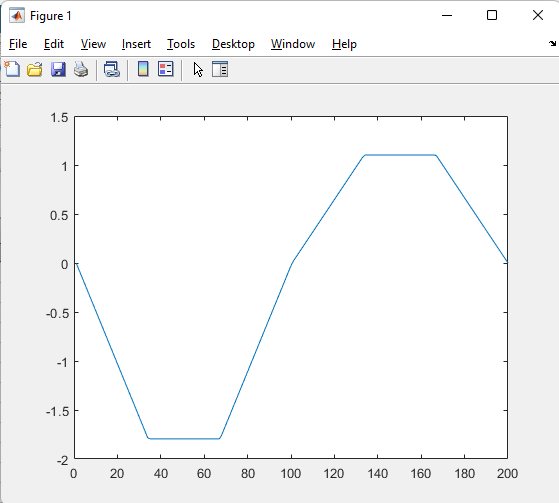
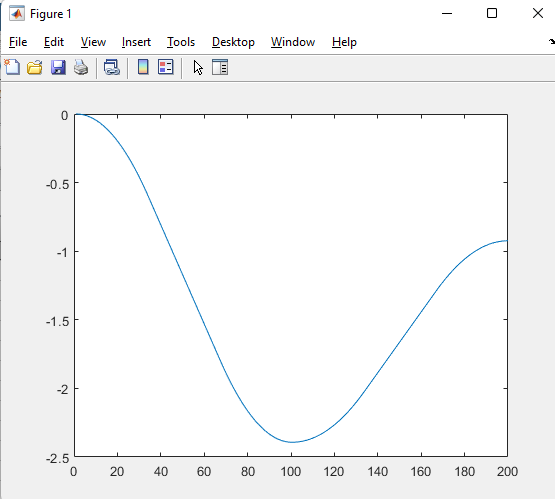
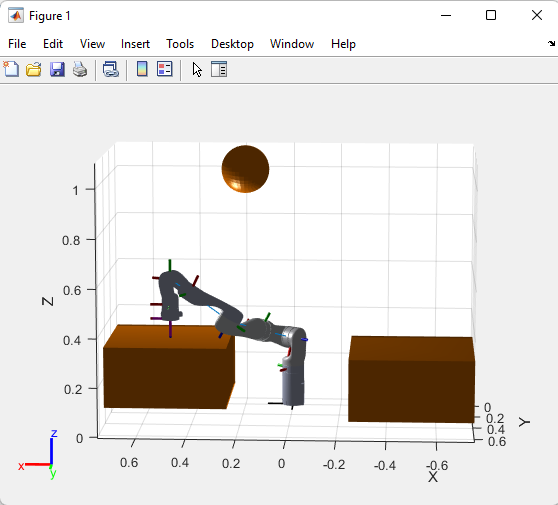
show(robot,q(:,i),"Parent",ax,"PreservePlot",false);

% Update the figure drawnow

end

disp("program over")

## OUTPUT:

****

**Plot of q Plot of qd**

****

**Plot of qdd**

## RESULT:

Thus the environmental collision with manipulator is checked using MATLAB R2021a

**CHECK FOR WORLD COLLISION PAIR WITH**

**Exp no: 8**

**7-AXIS KINOVA GEN3 ROBOT**

## AIM:

To check for world collision pair with 7-axis Kinova Gen3 Robot using MATLAB.

## SOFTWARE REQUIRED:

MATLAB R2021a.

## PROCEDURE:

1. Open the MATLAB R2021a software.
2. Go to Home > New script. A new editor window will open.
3. Type the program in the editor window.
4. Save the MATLAB file.
5. Go to editor > Run.
6. Observe the output.

## PROGRAM:

clc clear all

% Create two platforms

platform1 = collisionBox(0.5,0.5,0.25); platform1.Pose = trvec2tform([-0.5 0.4 0.2]);

platform2 = collisionBox(0.5,0.5,0.25); platform2.Pose = trvec2tform([0.5 0.2 0.2]);

% Add a light fixture, modeled as a sphere lightFixture = collisionSphere(0.1); lightFixture.Pose = trvec2tform([.2 0 1]);

% Store in a cell array for collision-checking worldCollisionArray = {platform1 platform2 lightFixture};

%Visualize the environment using a helper function that iterates through the collision array.

%ax = exampleHelperVisualizeCollisionEnvironment(worldCollisionArray); figHandle = figure;

% Show the first object

show(worldCollisionArray{1});

% Get axis properties and set hold ax = gca;

hold all;

% Show remaining objects

for i = 2:numel(worldCollisionArray) show(worldCollisionArray{i}, "Parent", ax);

end

% Set axis properties axis equal;

robot = loadrobot("kinovaGen3","DataFormat","column","Gravity",[0 0 -9.81]);

show(robot,homeConfiguration(robot),"Parent",ax);

startPose = trvec2tform([-0.5,0.5,0.4])\*axang2tform([1 0 0 pi]);

endPose = trvec2tform([0.5,0.2,0.4])\*axang2tform([1 0 0 pi]);

% Use a fixed random seed to ensure repeatable results rng(0);

ik = inverseKinematics("RigidBodyTree",robot); weights = ones(1,6);

startConfig = ik("EndEffector\_Link",startPose,weights,robot.homeConfiguration); endConfig = ik("EndEffector\_Link",endPose,weights,robot.homeConfiguration);

% Show initial and final positions show(robot,startConfig); show(robot,endConfig);

q = trapveltraj([homeConfiguration(robot),startConfig,endConfig],200,"EndTime",2);

% Initialize outputs

inCollision = false(length(q), 1); % Check whether each pose is in collision worldCollisionPairIdx = cell(length(q),1); % Provide the bodies that are in collision

for i = 1:length(q) [inCollision(i),sepDist] =

checkCollision(robot,q(:,i),worldCollisionArray,"IgnoreSelfCollision","on","Exhaustive","on");

[bodyIdx,worldCollisionObjIdx] = find(isnan(sepDist)); % Find collision pairs worldCollidingPairs = [bodyIdx,worldCollisionObjIdx]; worldCollisionPairIdx{i} = worldCollidingPairs;

end

isTrajectoryInCollision = any(inCollision)

collidingIdx1 = find(inCollision,1) collidingIdx2 = find(inCollision,1,"last")

% Identify the colliding rigid bodies.

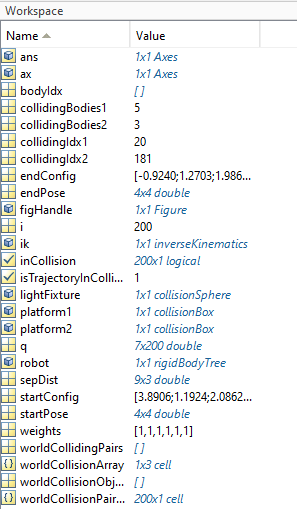
collidingBodies1 = worldCollisionPairIdx{collidingIdx1}\*[1 0]'; collidingBodies2 = worldCollisionPairIdx{collidingIdx2}\*[1 0]';

% Visualize the environment.

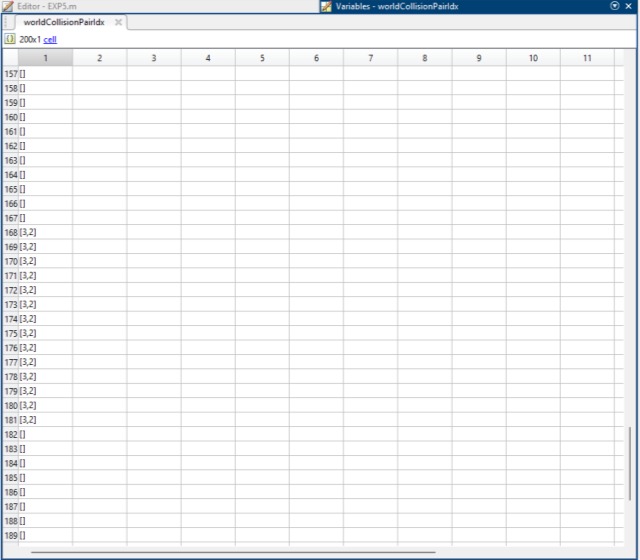
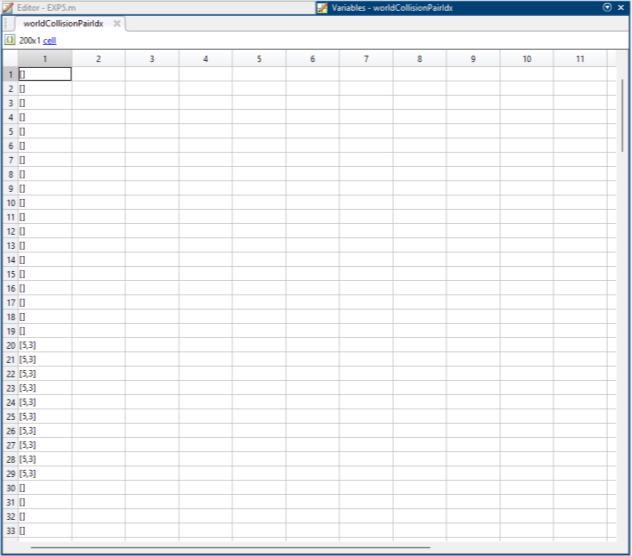
ax = exampleHelperVisualizeCollisionEnvironment(worldCollisionArray);

% Add the robotconfigurations & highlight the colliding bodies. show(robot,q(:,collidingIdx1),"Parent",ax,"PreservePlot",false); exampleHelperHighlightCollisionBodies(robot,collidingBodies1 + 1,ax); show(robot,q(:,collidingIdx2),"Parent",ax); exampleHelperHighlightCollisionBodies(robot,collidingBodies2 + 1,ax);

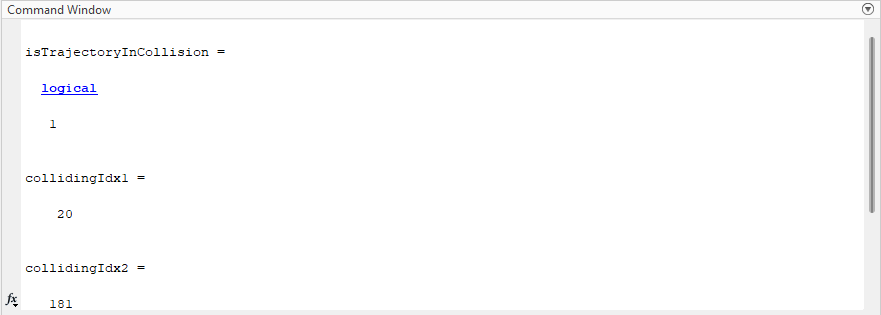
**WORKSPACE:**

****

**WORLD COLLISION PAIR:**

****

**OUTPUT:**

****

**RESULT:**

Thus the world collision pair with 7-axis Kinova Gen3 Robot is checked using MATLAB R2021a.

**ROBOT PROGRAMMING AND SIMULATION FOR**

**Exp no: 9**

**PICK AND PLACE**

## AIM:

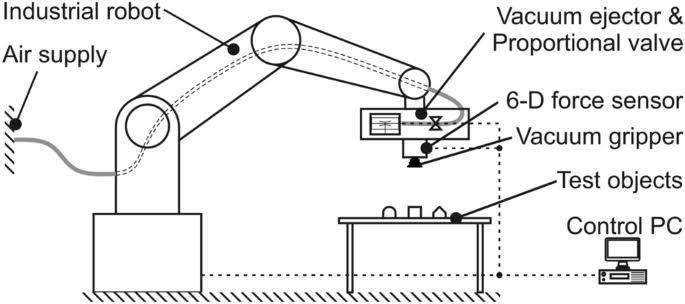
To program and simulate the robot for pick and place using Dobot.

## REQUIREMENTS:

Dobot, DobotStudio-V1.9.4

## PROCEDURE:

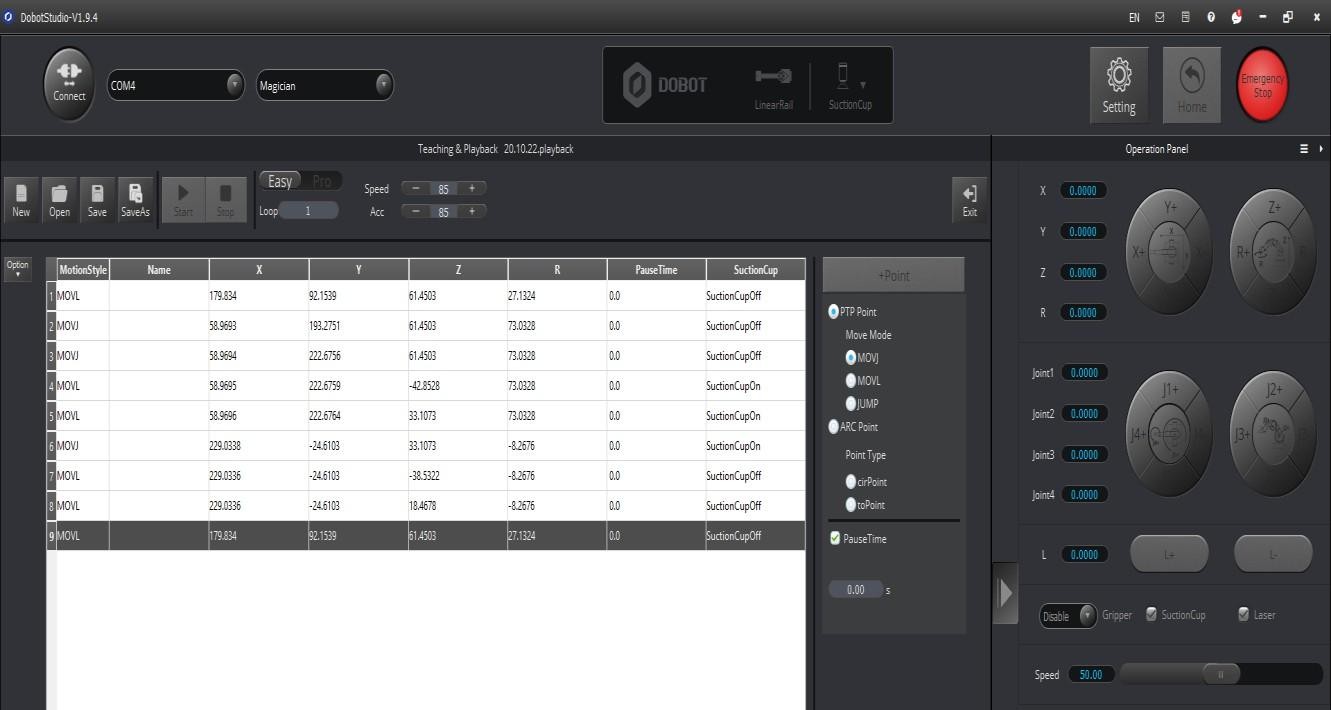
1. Check the connections of the Dobot.



1. Connect the Dobot with the DobotStudio-V1.9.4 software using the connect option in the software.
2. Set the Dobot in home position.
3. Select the suction cup tool.
4. Select the teaching and playback method.



1. Open a new workspace.
2. To manually teach a Dobot, press the lock button in the Dobot and move the arm step by step or we can control the movement by adjusting the coordinates (World and Joint Coordinates).
3. The coordinates of each movement will be recorded.
4. If it is a linear movement set the motion style as MOVL, if it is a free movement set the motion style as MOVJ.
5. Once the arm touches the object, the suction cup should be ON to pick the object, the suction cup should be OFF to place the object.
6. The end position must be set to the home position.



1. Once the coordinates are fixed, run and observe the pick and place movement.

## RESULT:

Thus, the programming and simulation of Dobot for pick and place was done successfully.

**ROBOT PROGRAMMING AND SIMULATION FOR**

**Exp no: 10**

**PICK AND PLACE – USING PROXIMITY SENSOR**

**BY BLOCKLY METHOD**

## AIM:

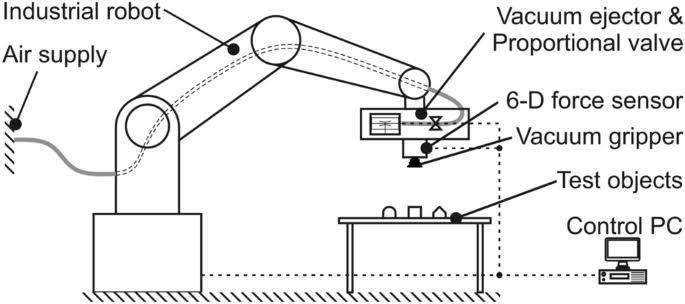
To program and simulate the robot for pick and place by blockly method using Dobot and proximity sensor.

## REQUIREMENTS:

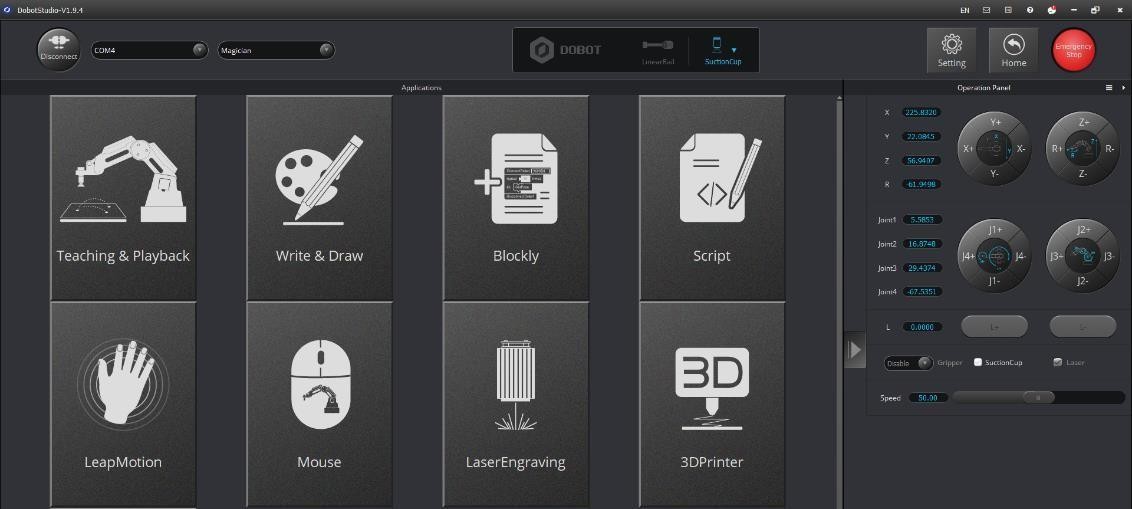
Dobot, DobotStudio-V1.9.4, Conveyor with proximity sensor.

## PROCEDURE:

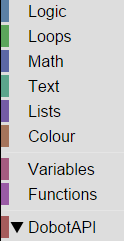
1. Connect the Dobot with conveyor.



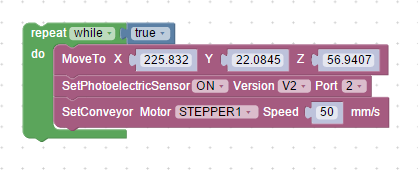
1. Connect the Dobot with DobotStudio-V1.9.4 using connect option in the software.
2. Set the Dobot to home position.
3. Select the suction cup tool.
4. Select the blockly method.



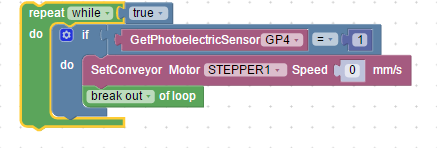
1. Open a new work space.
2. Select the required blockly module on the left pane of the blockly page to the program (logic, loops, variables).



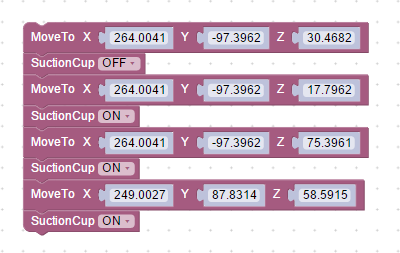
1. Add the coordinates of home position.
2. ON the photoelectric sensor.
3. Set the speed of the stepper motor of the conveyor to 50 mm/s.



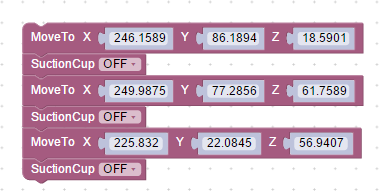
1. Once the photoelectric sensor detects the object the stepper motor speed of conveyor is set to 0 mm/s.



1. Set the coordinates of the moment manually by moving the Dobot to pick the object.

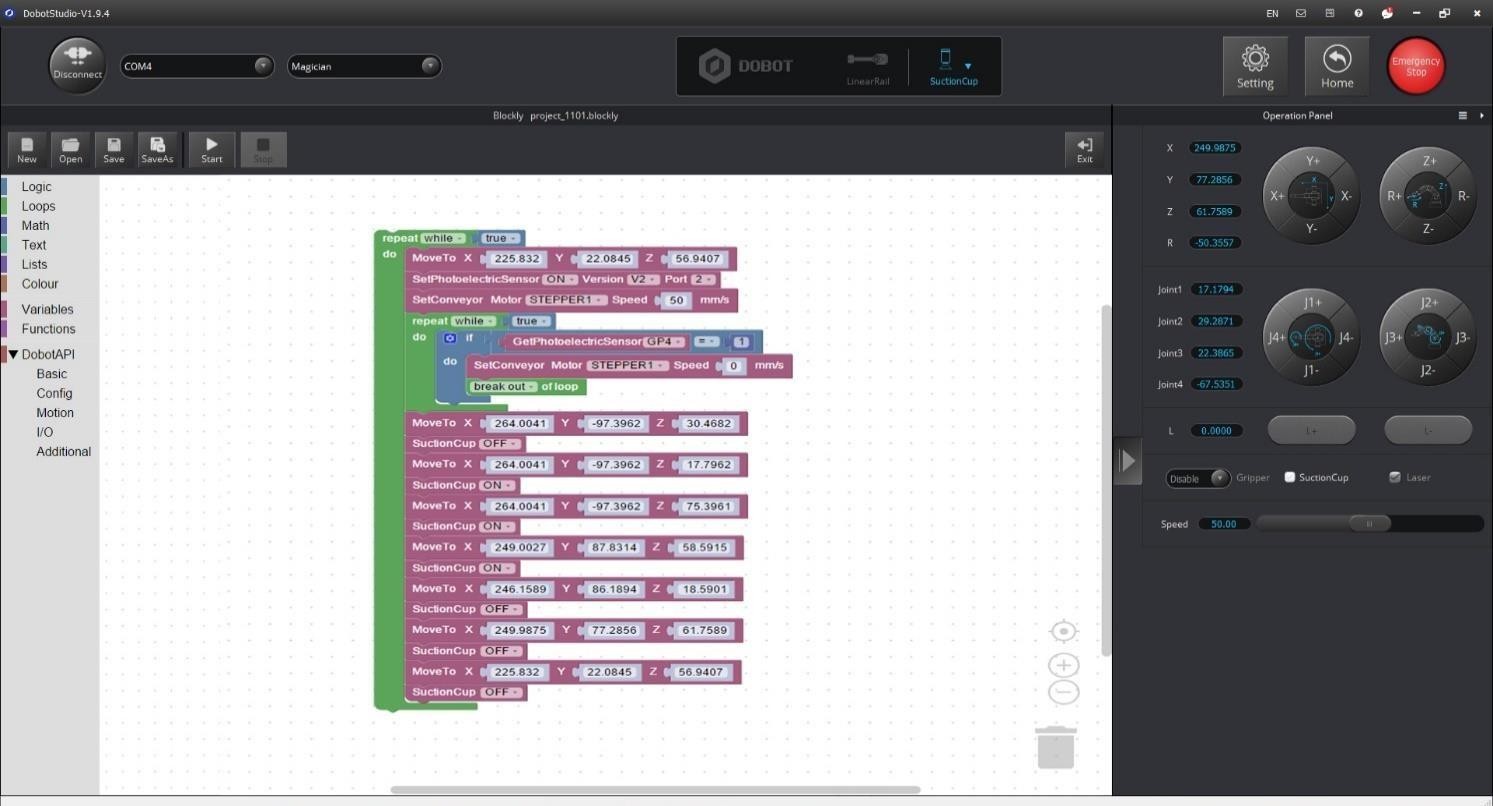


1. Set the coordinates of the movement to place the object.



1. Use a break out block to stop the loop.
2. Run and observe the pick and place movement.

## PROGRAM:

****

**RESULT:**

Thus, the programming and simulation to pick and place the object by blockly method using Dobot with conveyor and proximity sensor was done successfully.

**ROBOT PROGRAMMING AND SIMULATION FOR**

**Exp no: 11**

**PICK AND PLACE – USING PROXIMITY SENSOR**

**BY SCRIPT METHOD**

## AIM:

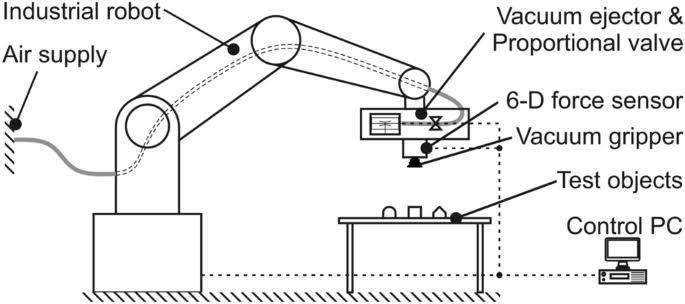
To program and simulate the robot for pick and place by script method using Dobot and proximity sensor.

## REQUIREMENTS:

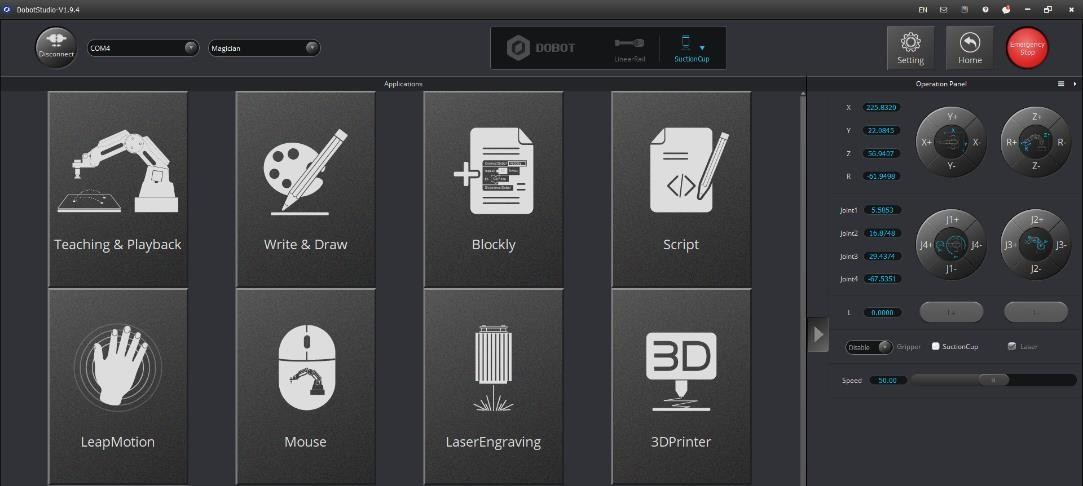
Dobot, DobotStudio-V1.9.4, Conveyor with proximity sensor.

## PROCEDURE:

1. Connect the Dobot with conveyor.

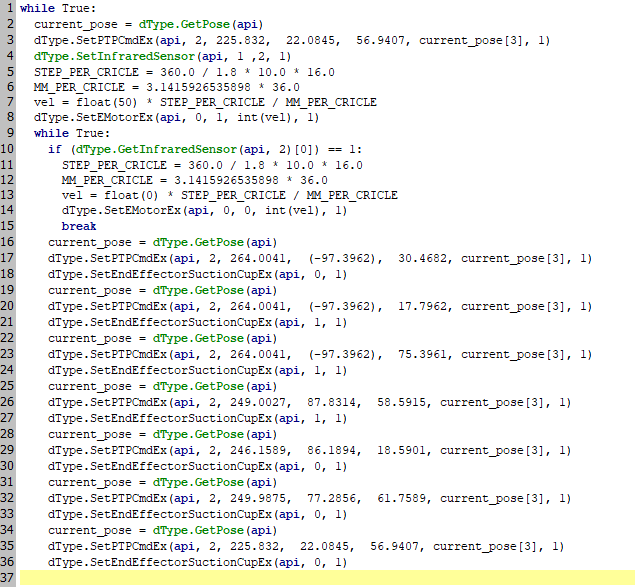


1. Connect the Dobot with DobotStudio-V1.9.4 using connect option in the software.
2. Set the Dobot to home position.
3. Select the suction cup tool.
4. Select the script method.



1. Open a new work space.
2. Type the program in the workspace.
3. Run and observe the pick and place movement.

## PROGRAM:

****

**RESULT:**

Thus, the programming and simulation to pick and place the object by script method using Dobot with conveyor and proximity sensor was done successfully.

**DETERMINATION OF MAXIMUM AND MINIMUM**

**Exp no: 12**

**POSITION OF LINKS – SPATIAL MANIPULATOR**

## AIM:

To determine the maximum and minimum position of links of a spatial manipulator.

## SOFTWARE REQUIRED:

MATLAB R2021a.

## PROCEDURE:

1. Open the MATLAB R2021a software.
2. Go to Home > New script. A new editor window will open.
3. Type the program in the editor window.
4. Save the MATLAB file.
5. Go to editor > Run.
6. Observe the output.

## PROGRAM:

clc clear all

gen3=loadrobot("kinovaGen3"); gen3.DataFormat='column';

q\_home=[0 15 180 -130 0 55 90]'\*pi/180; eeName='EndEffector\_Link'; T\_home=getTransform(gen3,q\_home,eeName) show(gen3,q\_home);

axis auto; view([60,10]);

k=tform2trvec(getTransform(gen3,q\_home,eeName)); [isColliding,sepDist]=checkCollision(gen3,q\_home,'Exhaustive','on') ik=inverseKinematics('RigidBodyTree',gen3); ik.SolverParameters.AllowRandomRestart=false; weights=[1,1,1,1,1,1];

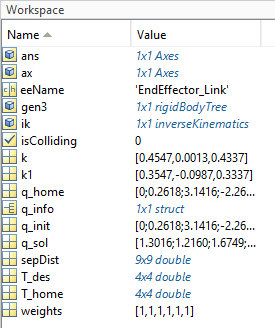
q\_init=q\_home; T\_des=T\_home;

k1=k'; figure;set(gcf,'Visible','on'); k1=k-0.1;

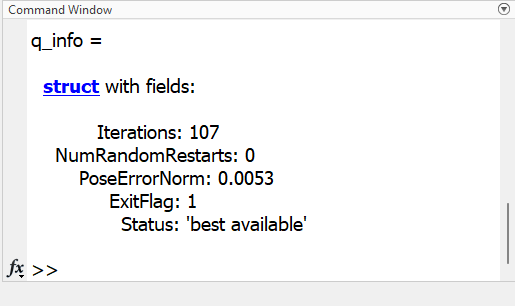
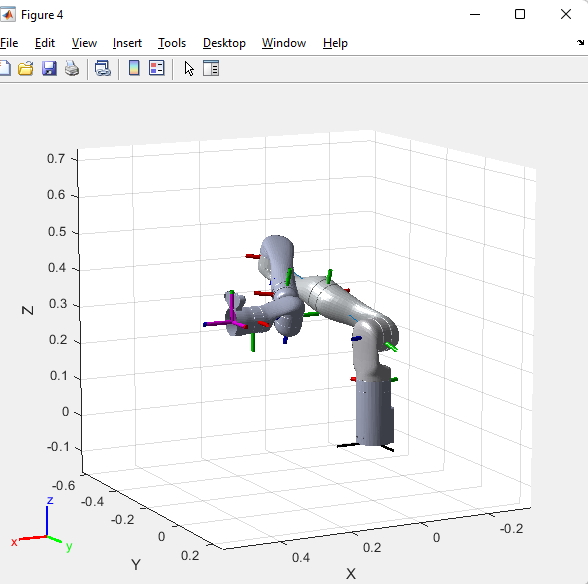
T\_des(1:3,4)=k1; [q\_sol,q\_info]=ik(eeName,T\_des,weights,q\_init) ax=show(gen3,q\_sol); ax.CameraPositionMode='auto';

view([60,10]);

## WORKSPACE:

****

**OUTPUT:**

## RESULT:

Thus the determination of maximum and minimum position of links of a spatial manipulator using MATLAB R2021a was done successfully.

**INVERSE KINEMATICS OF KINOVA GEN3 ROBOT**

**Exp no: 13**

**TO DRAW A CIRCULAR PATH**

## AIM:

To draw a circular path with Kinova Gen3 Robot using inverse kinematics.

## SOFTWARE REQUIRED:

MATLAB R2021a.

## PROCEDURE:

1. Open the MATLAB R2021a software.
2. Go to Home > New script. A new editor window will open.
3. Type the program in the editor window.
4. Save the MATLAB file.
5. Go to editor > Run.
6. Observe the output.

## PROGRAM:

clc clear all

gen3 = loadrobot("kinovaGen3"); gen3.DataFormat = 'column';

q\_home = [0 15 180 -130 0 55 90]'\*pi/180 eeName = 'EndEffector\_Link';

T\_home = getTransform(gen3, q\_home, eeName); show(gen3,q\_home);

axis auto; view([60,10]);

ik = inverseKinematics('RigidBodyTree',gen3); ik.SolverParameters.AllowRandomRestart = false; weights = [1, 1, 1, 1, 1, 1];

q\_init = q\_home;

center = [0.5 0 0.4]; %[x y z]

radius = 0.1;

dt = 0.25;

t = (0:dt:10)';

theta = t\*(2\*pi/t(end))-(pi/2); theta = t\*(2\*pi/t(end));

points = center + radius\*[0\*ones(size(theta)) cos(theta) sin(theta)]; hold on;

plot3(points(:,1),points(:,2),points(:,3),'-\*g','LineWidth', 1.5); numJoints = size(q\_home,1);

numWaypoints = size(points,1);

qs = zeros(numWaypoints,numJoints); for i = 1:numWaypoints

T\_des = T\_home; T\_des(1:3,4) = points(i,:)';

[q\_sol, q\_info] = ik(eeName, T\_des, weights, q\_init); qs(i,:) = q\_sol(1:numJoints);

q\_init = q\_sol;

end

%Visualize the Animation of the Solution figure; set(gcf,'Visible','on');

ax = show(gen3,qs(1,:)'); ax.CameraPositionMode='auto'; hold on;

% Plot waypoints plot3(points(:,1),points(:,2),points(:,3),'-g','LineWidth',2); axis auto;

view([60,10]);

grid('minor');

hold on;

title('Simulated Movement of the Robot');

% Animate framesPerSecond = 30;

r = robotics.Rate(framesPerSecond); for i = 1:numWaypoints

show(gen3, qs(i,:)','PreservePlot',false); drawnow;

waitfor(r);

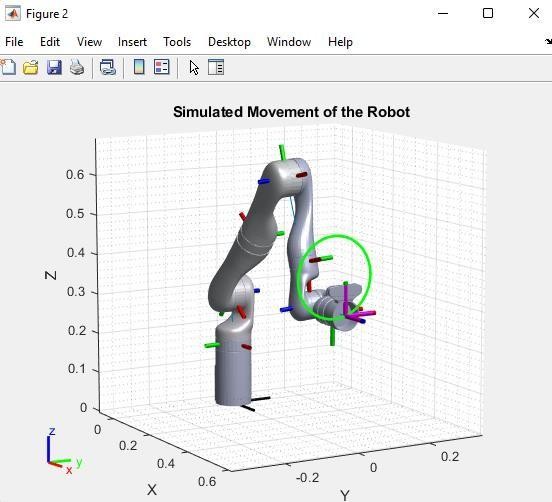
end xlabel('x');

ylabel('y');

zlabel('z'); axis auto; view([60,10]);

grid('minor');

## OUTPUT:

****

**RESULT:**

Thus, the circular path with Kinova Gen3 Robot using inverse kinematics was done successfully.

**INVERSE KINEMATICS OF KINOVA GEN3 ROBOT**

**Exp no: 14**

**TO DRAW A TRIANGULAR PATH**

## AIM:

To draw a triangular path with Kinova Gen3 Robot using inverse kinematics.

## SOFTWARE REQUIRED:

MATLAB R2021a.

## PROCEDURE:

1. Open the MATLAB R2021a software.
2. Go to Home > New script. A new editor window will open.
3. Type the program in the editor window.
4. Save the MATLAB file.
5. Go to editor > Run.
6. Observe the output.

## PROGRAM:

clc clear all

gen3 = loadrobot("kinovaGen3"); gen3.DataFormat = 'column';

q\_home = [0 15 180 -130 0 55 90]'\*pi/180; eeName = 'EndEffector\_Link';

T\_home = getTransform(gen3, q\_home, eeName); show(gen3,q\_home);

axis auto; view([60,10]);

ik = inverseKinematics('RigidBodyTree', gen3); ik.SolverParameters.AllowRandomRestart = false; weights = [1, 1, 1, 1, 1, 1];

q\_init = q\_home; points(1,:)=[0.5 0 0.4]; k=1;

for i = 0.1:0.1:0.3

k=k+1;

points (k,:)= points(1,:)+[0 i 0];

end

k1=k;

for j=0.1:0.1:0.3

k=k+1;

points (k,:)= points(k1,:)+[0 0 j];

end k2=k;

for j=0.1:0.1:0.3

k=k+1;

points (k,:)= points(k2,:)+[0 -j -j]; hold on;

plot3(points(:,1),points(:,2),points(:,3),'-\*g', 'LineWidth', 1.5);

end

numJoints = size(q\_home,1); numWaypoints = size(points,1);

qs = zeros(numWaypoints,numJoints); for i = 1:numWaypoints

T\_des = T\_home; T\_des(1:3,4) = points(i,:)';

[q\_sol, q\_info] = ik(eeName, T\_des, weights, q\_init); if q\_info.ExitFlag==2

disp('not reached the point',T\_des(:,4)) end

qs(i,:) = q\_sol(1:numJoints); q\_init = q\_sol;

end

%Visualize the Animation of the Solution figure; set(gcf,'Visible','on');

ax = show(gen3,qs(1,:)'); ax.CameraPositionMode='auto'; hold on;

% Plot waypoints plot3(points(:,1),points(:,2),points(:,3),'-g','LineWidth',2); axis auto;

view([60,10]);

grid('minor'); hold on;

title('Simulated Movement of the Robot');

% Animate

% framesPerSecond =20;

% r = robotics.Rate(framesPerSecond); for i = 1:numWaypoints

show(gen3, qs(i,:)','PreservePlot',false); drawnow;

% waitfor(0.1);

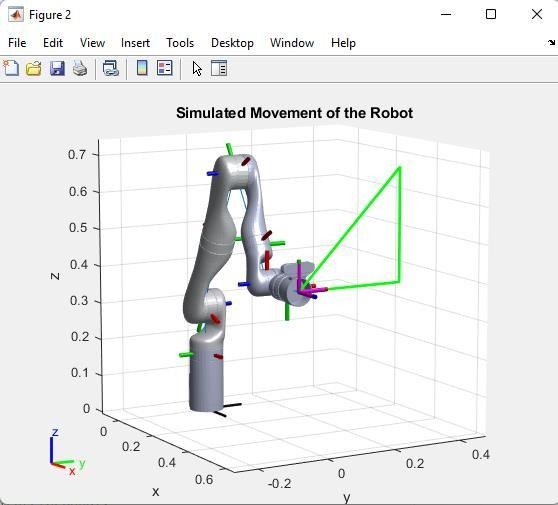
end xlabel('x');

ylabel('y');

zlabel('z'); axis auto; view([60,10]);

grid('minor');

## OUTPUT:

****

**RESULT:**

Thus, the triangular path with Kinova Gen3 Robot using inverse kinematics was done successfully.

**6-AXIS MITSUBISHI ROBOT (INDUSTRIAL ROBOT)**

**Exp no: 15**

**PROGRAMMING FOR PICK AND PLACE**

## AIM:

To program the 6-axis Mitsubishi Robot for pick and place movement.

## SOFTWARE REQUIRED:

6 – axis Mitsubishi Robot, RT ToolBox3, Teaching Pendant

## PROCEDURE:

1. Check the connections and switch on the 6-axis Mitsubishi Robot.



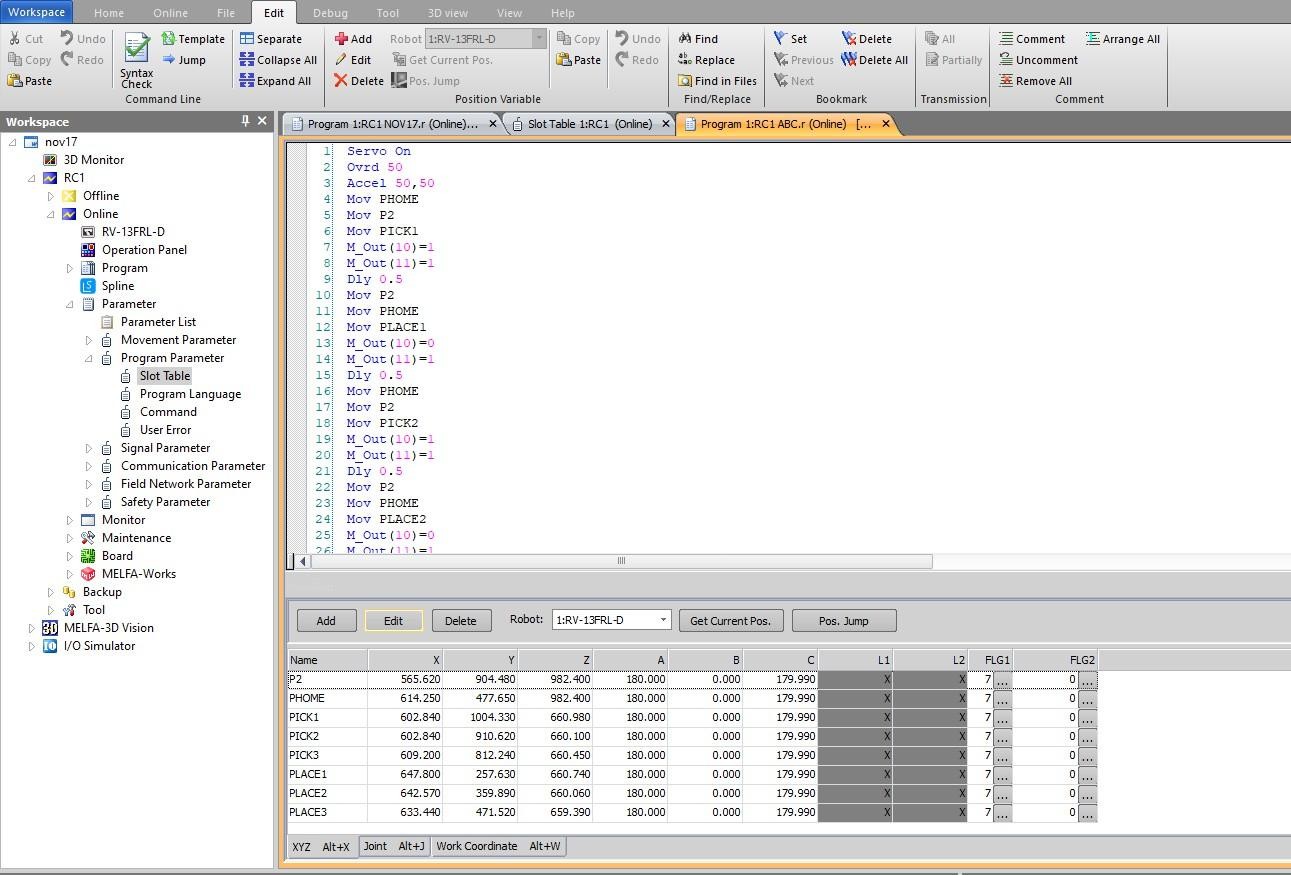
1. Connect the 6-axis Mitsubishi Robot with RT ToolBox3 software.
2. In RT ToolBox3 go to Home > Online > program > (right click) new > name the file.
3. Type the program.
4. To add the positions of each point, we need to teach the robot using teach pendant and add the position to the software.
5. First set the home position of robot using the teach pendant by the following steps:
   1. On the manual mode in the PLC.
   2. Turn on the TB enable button at the back of the teach pendant.
   3. Hold the scroll button to a limited angle and click the servo button

(The servo will be off when the scroll is over pressed or when it is released).

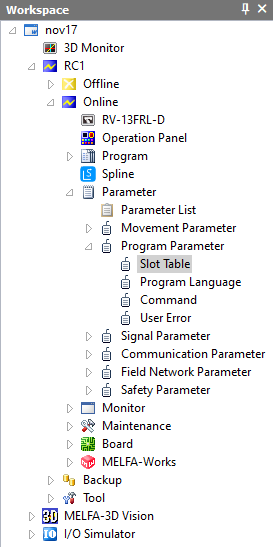
 

* 1. Click the jog button > select the appropriate function (tool or joint).
  2. Now by using the coordinates (x,y,z) set the position.

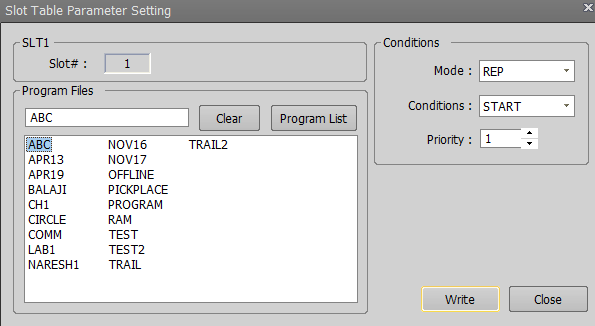
1. Now in the software click add and the coordinates will be displayed, now name the coordinates as PHOME and click get current position so that your current position will be stored.
2. Likewise move the coordinates using teach pendant and set the P1 point and store the position.
3. Repeat this process for all the positions ( PHOME, P1, P2, PICK1,PLACE1)



1. Once the positions are added, save the program.
2. To teach the program to the robot we need to do the following steps:
   1. Under workspace click parameter > program parameter > slot table.



* 1. Now a pop up box will appear, select the required program and click write.



1. Once the program is loaded, turn off the TB enable.
2. Switch to auto mode in PLC and start the cycle.
3. Observe the pick and place movement.

Note: If there is any emergency situation press the emergency button in PLC.

## PROGRAM:

****

**RESULT:**

Thus, the pick and place movement of 6-axis Mitsubishi Robot was done successfully

# Viva Questions and Answers

1. **Determination of Maximum and Minimum Position of Links**
   1. **Q**: What is the purpose of determining the maximum and minimum positions of links in a manipulator? **A**: To establish the reachable workspace and limits of movement for the manipulator, ensuring safe operation.
   2. **Q**: How do you calculate the maximum extension of a link? **A**: By adding the lengths of all connected links in a straight line, considering joint limits.
   3. **Q**: What is a workspace in robotics? **A**: The total volume or area that the end effector can reach, defined by the manipulator's link lengths and joint constraints.
   4. **Q**: What are joint limits? **A**: The maximum and minimum angles or displacements that a joint can achieve.
   5. **Q**: Why is it important to consider the minimum position of links? **A**: To avoid collisions and ensure that the manipulator does not exceed its mechanical limits.
   6. **Q**: What tools or software can be used to visualize the workspace? **A**: Robotics simulation software like MATLAB, ROS, or specific CAD programs.
   7. **Q**: What is the significance of the forward kinematics in this context? **A**: It allows us to determine the end effector's position based on joint angles and link lengths.
   8. **Q**: What would happen if a manipulator exceeded its maximum position? **A**: It could lead to mechanical failure or damage to the manipulator and surrounding objects.
   9. **Q**: How can you ensure safety during the experiments? **A**: By adhering to safety protocols, using emergency stops, and ensuring the area is clear before operation.
   10. **Q**: What are the typical link lengths in a 6-axis robot? **A**: Link lengths vary by design, but they are typically between 0.5 to 1.5 meters in industrial robots.
2. **Trajectory Control Modeling with Inverse Kinematics Using Simulink**
   1. **Q**: What is trajectory control in robotics? **A**: It is the process of planning and executing the path that a manipulator should follow to reach a target position smoothly.
   2. **Q**: What role does inverse kinematics play in trajectory control? **A**: Inverse kinematics calculates the required joint angles to achieve a desired position and orientation of the end effector.
   3. **Q**: How do you implement trajectory control in Simulink? **A**: By using blocks to model the robot dynamics, define the trajectory, and apply inverse kinematics to control the joints.
   4. **Q**: What are the common methods for solving inverse kinematics? **A**: Geometric methods, algebraic methods, and numerical methods are commonly used.
   5. **Q**: Why is it important to smooth the trajectory? **A**: To reduce jerk and improve the quality of motion, ensuring safer operation and preserving mechanical integrity.
   6. **Q**: What challenges can arise when using inverse kinematics? **A**: Multiple solutions, singularities, and joint limits can complicate the calculation of inverse kinematics.
   7. **Q**: Can Simulink simulate real-time control of robots? **A**: Yes, Simulink can be used for real- time simulation and control by integrating with hardware through ROS or MATLAB.
   8. **Q**: What parameters can you adjust in Simulink to modify the trajectory? **A**: You can adjust speed, acceleration, and path curvature parameters.
   9. **Q**: How does feedback control relate to trajectory tracking? **A**: Feedback control helps correct any deviations from the desired trajectory in real-time.
   10. **Q**: What is the significance of PID control in trajectory control? **A**: PID control helps minimize errors in the trajectory by adjusting control inputs based on proportional, integral, and derivative terms.
3. **Trajectory Path Planning of 2R Manipulator**
   1. **Q**: What is a 2R manipulator? **A**: A manipulator with two rotational joints (links), often used to illustrate basic concepts in robotics.
   2. **Q**: What is trajectory path planning? **A**: The process of defining a path that the end effector of the manipulator should follow over time.
   3. **Q**: How do you define the start and end points for the trajectory? **A**: By specifying the desired positions and orientations for the end effector in Cartesian space.
   4. **Q**: What are the main techniques used for trajectory planning? **A**: Linear interpolation, cubic splines, and quintic polynomials are commonly used techniques.
   5. **Q**: Why is it necessary to consider dynamics in trajectory planning? **A**: To ensure that the manipulator can follow the planned trajectory within its physical capabilities.
   6. **Q**: What is a path planning algorithm? **A**: An algorithm that determines the trajectory while avoiding obstacles and ensuring smooth movement.
   7. **Q**: How does the choice of trajectory affect the manipulator's performance? **A**: A well- planned trajectory minimizes energy consumption and maximizes efficiency while avoiding collisions.
   8. **Q**: What is the role of simulation in trajectory path planning? **A**: Simulation allows for testing and refining the trajectory before executing it on the actual manipulator.
   9. **Q**: Can you achieve the same end effector position with different joint angles? **A**: Yes, due to the redundancy in the manipulator’s configuration.
   10. **Q**: What safety considerations should be made during trajectory execution? **A**: Ensure the workspace is clear, use emergency stops, and monitor the manipulator's movements.
4. **Check for Environmental Collisions with Manipulator**
   1. **Q**: What is the importance of checking for environmental collisions? **A**: To prevent damage to the robot, surrounding equipment, and ensure operator safety.
   2. **Q**: What are common methods for detecting collisions? **A**: Using sensors, simulations, and collision detection algorithms in software.
   3. **Q**: How do you model the environment in simulation software? **A**: By defining the geometry of obstacles and the workspace where the robot operates.
   4. **Q**: What is the role of bounding boxes in collision detection? **A**: Bounding boxes help simplify the calculation of potential collisions by providing a rough estimate of object dimensions.
   5. **Q**: Can you implement soft limits to prevent collisions? **A**: Yes, soft limits can be set to keep the manipulator within a safe distance from obstacles.
   6. **Q**: What is the significance of real-time collision detection? **A**: Real-time detection allows for immediate responses to prevent collisions during operation.
   7. **Q**: How can you use simulation to test collision avoidance strategies? **A**: By running simulations with different scenarios and analyzing the robot's responses to avoid collisions.
   8. **Q**: What sensors are commonly used for collision detection? **A**: Proximity sensors, LIDAR, ultrasonic sensors, and cameras can all be used.
   9. **Q**: What is a collision avoidance algorithm? **A**: An algorithm designed to adjust the robot’s path or speed to avoid potential collisions.
   10. **Q**: How can you ensure safety in a collaborative workspace? **A**: By using safety-rated sensors and ensuring clear communication between the robot and human operators.
5. **Check for World Collision Pair with 7-Axis Kinova Gen3 Robot**
   1. **Q**: What is a 7-axis robot? **A**: A robot with seven degrees of freedom, providing greater flexibility and reach in its operations.
   2. **Q**: How do you define a world collision pair? **A**: A pair of objects or components that can potentially collide during the robot's operation in a defined workspace.
   3. **Q**: Why is it critical to analyze collision pairs? **A**: To identify and mitigate risks of accidental impacts that can cause damage or pose safety hazards.
   4. **Q**: What tools are available for analyzing collisions in robotics? **A**: Simulation software like ROS, MATLAB, and dedicated robotic simulation platforms can be used.
   5. **Q**: How do you test collision avoidance in a 7-axis robot? **A**: By simulating various movement scenarios and checking for potential collisions using collision detection algorithms.
   6. **Q**: What are the advantages of a 7-axis configuration over a 6-axis robot? **A**: The additional axis provides enhanced maneuverability and the ability to reach difficult positions.
   7. **Q**: What is the role of simulation in collision checking? **A**: Simulation allows for extensive testing of movements without risking damage to physical hardware.
   8. **Q**: How can you adjust the robot’s path to avoid world collisions? **A**: By modifying joint angles, speeds, or introducing intermediate waypoints in the trajectory.
   9. **Q**: What is the significance of workspace modeling? **A**: Accurate workspace modeling helps identify potential collision scenarios and optimize robot paths.
   10. **Q**: How do you document collision checks and outcomes? **A**: By maintaining detailed logs of tests, including configurations, encountered issues, and resolutions.
6. **Robot Programming and Simulation for Pick and Place**
   1. **Q**: What is a pick-and-place operation in robotics? **A**: It involves picking an object from a location and placing it at a designated target location.
   2. **Q**: What programming languages are commonly used for robot programming? **A**: C++, Python, and specific robot programming languages like ROS are commonly used.
   3. **Q**: How do you define the pick and place algorithm? **A**: By specifying the sequence of movements for the robot to successfully grasp and relocate an object.
   4. **Q**: What sensors are typically used for pick-and-place applications? **A**: Cameras, proximity sensors, and force/torque sensors are often used to aid in object detection and manipulation.
   5. **Q**: Why is object detection important in pick-and-place tasks? **A**: Accurate detection ensures the robot can successfully identify and grasp the correct object.
   6. **Q**: How do you simulate a pick-and-place operation? **A**: By modeling the robot's movements and the environment in simulation software to test the algorithm.
   7. **Q**: What challenges might arise during a pick-and-place operation? **A**: Misalignment, object variability, and unexpected environmental factors can pose challenges.
   8. **Q**: How can you improve the accuracy of a pick-and-place operation? **A**: By calibrating sensors, optimizing grasping techniques, and refining the trajectory planning.
   9. **Q**: What safety measures should be taken during pick-and-place operations? **A**: Ensure the workspace is clear, monitor for obstacles, and have emergency stops readily accessible.
   10. **Q**: How do you validate the success of a pick-and-place operation? **A**: By verifying that the object has been successfully picked and placed at the desired location without damage.
7. **Robot Programming and Simulation for Pick and Place – Using Proximity Sensor by Blockly Method**
   1. **Q**: What is Blockly in the context of robot programming? **A**: Blockly is a visual programming language that allows users to create code by stacking blocks that represent programming constructs.
   2. **Q**: How does using a proximity sensor enhance pick-and-place operations? **A**: It allows the robot to detect the presence of objects, enabling automated picking processes.
   3. **Q**: What are the steps to program a robot using Blockly for pick-and-place? **A**: Define object detection, initiate the grasping sequence, and program the placing action using Blockly blocks.
   4. **Q**: Why is visual programming beneficial for beginners? **A**: It simplifies the programming process and reduces the complexity of syntax errors.
   5. **Q**: How do you connect the proximity sensor to the Blockly program? **A**: By using specific blocks that represent sensor input and defining the logic for object detection.
   6. **Q**: What is the importance of calibrating the proximity sensor? **A**: Calibration ensures accurate distance measurements, improving the reliability of object detection.
   7. **Q**: Can Blockly be used for advanced robotic programming? **A**: While primarily for educational purposes, Blockly can be extended for more complex programming with additional features.
   8. **Q**: How do you troubleshoot issues in Blockly programming? **A**: By checking block connections, sensor calibration, and reviewing the logic flow of the program.
   9. **Q**: What safety considerations are needed when using proximity sensors? **A**: Ensure the sensor is correctly calibrated and positioned to avoid false readings or missed objects.
   10. **Q**: How can you test the effectiveness of the Blockly program? **A**: By running simulations and observing the robot’s behavior in picking and placing tasks.
8. **Robot Programming and Simulation for Pick and Place – Using Proximity Sensor by Script Method**
   1. **Q**: What is the script method in robot programming? **A**: It involves writing code manually using a programming language to control the robot's operations.
   2. **Q**: How do you implement proximity sensor logic in a script? **A**: By defining conditional statements to read the sensor data and trigger appropriate actions based on object detection.
   3. **Q**: What are the advantages of using scripts over visual programming? **A**: Scripts provide greater flexibility and control, allowing for more complex logic and integrations.
   4. **Q**: How do you handle errors in script-based programming? **A**: By implementing error handling techniques, such as try-catch blocks, to manage exceptions.
   5. **Q**: Why is it important to simulate the script before executing it on hardware? **A**: To identify and fix potential issues without risking damage to the robot or surrounding equipment.
   6. **Q**: What programming languages are commonly used for scripting in robotics? **A**: Python, C++, and JavaScript are frequently used for scripting robot behaviors.
   7. **Q**: How can you optimize the pick-and-place script for performance? **A**: By minimizing unnecessary calculations and optimizing the sequence of commands.
   8. **Q**: What challenges might arise when using a script for pick-and-place tasks? **A**: Syntax errors, unexpected behavior due to logic flaws, and sensor calibration issues can be challenges.
   9. **Q**: How do you document your script for future reference? **A**: By adding comments to the code explaining the functionality of each section and maintaining a change log.
   10. **Q**: What testing strategies can you employ to ensure the script works correctly? **A**: Unit testing, integration testing, and running simulations can help verify the script's accuracy and reliability.
9. **Determination of Maximum and Minimum Position of Links – Spatial Manipulator**
   1. **Q**: How does a spatial manipulator differ from a planar manipulator? **A**: A spatial manipulator operates in three-dimensional space, whereas a planar manipulator operates in a two-dimensional plane.
   2. **Q**: What additional factors must be considered for spatial manipulators? **A**: Factors like 3D joint configurations, additional degrees of freedom, and the complex interactions of movement.
   3. **Q**: Why is it important to visualize the positions of links in 3D? **A**: Visualization helps in understanding the range of motion and potential collisions in three-dimensional space.
   4. **Q**: How do you calculate the positions of links in a spatial manipulator? **A**: By using transformation matrices to account for joint angles and link lengths in three dimensions.
   5. **Q**: What role do Denavit-Hartenberg parameters play in spatial manipulators? **A**: They provide a systematic way to represent the link lengths, joint angles, and orientations for kinematic analysis.
   6. **Q**: What are the typical applications of spatial manipulators? **A**: Applications include assembly tasks, automated welding, and robotic surgery, where 3D manipulation is required.
   7. **Q**: What software tools can aid in modeling spatial manipulators? **A**: Tools like MATLAB, SolidWorks, and ROS can be used for modeling and simulation.
   8. **Q**: How can simulation help in determining the positions of links? **A**: Simulation allows for dynamic analysis and visualization of the manipulator's motion in a 3D environment.
   9. **Q**: What safety concerns are associated with spatial manipulators? **A**: Potential hazards from extended reach, unexpected movements, and interference with surrounding objects.
   10. **Q**: How do you ensure accuracy when determining positions in a spatial manipulator? **A**: By using precise measurements, accurate modeling, and validating results through simulation.
10. **Inverse Kinematics of Kinova Gen3 Robot to Draw a Circular Path**
    1. **Q**: What is inverse kinematics (IK)? **A**: IK is the process of calculating the joint configurations required for a robot to achieve a desired end effector position and orientation.
    2. **Q**: Why is drawing a circular path significant in robotics? **A**: It tests the robot's ability to perform complex motions and maintain accuracy over curved trajectories.
    3. **Q**: How do you define the parameters for a circular path? **A**: By specifying the center, radius, and the plane in which the circle lies.
    4. **Q**: What are the common challenges in solving inverse kinematics for circular paths? **A**: Multiple solutions, singularities, and ensuring smooth transitions between points can be challenging.
    5. **Q**: What algorithms can be used for IK in the Kinova Gen3? **A**: Analytical methods, numerical methods (like Newton-Raphson), and optimization techniques can be used.
    6. **Q**: How can you visualize the circular path in simulation? **A**: By plotting the desired path in the simulation software and verifying the robot's movement along it.
    7. **Q**: What is the importance of real-time feedback in executing a circular path? **A**: Real-time feedback allows for adjustments to maintain accuracy and correct deviations during motion.
    8. **Q**: How do you ensure the smooth execution of a circular path? **A**: By adjusting joint velocities and accelerations to prevent abrupt changes that can lead to jerky movements.
    9. **Q**: What safety measures should be in place during this experiment? **A**: Ensure the workspace is clear, monitor the robot's movements, and have emergency stops accessible.
    10. **Q**: How can you validate the success of the circular path execution? **A**: By checking that the end effector accurately follows the defined circular trajectory without deviations.
11. **Inverse Kinematics of Kinova Gen3 Robot to Draw a Triangular Path**
    1. **Q**: What is the objective of drawing a triangular path with a robot? **A**: To assess the robot's precision and ability to navigate between distinct points in a structured path.
    2. **Q**: How do you define the vertices of a triangular path? **A**: By specifying the coordinates of each vertex in the robot's workspace.
    3. **Q**: What challenges arise when implementing IK for a triangular path? **A**: Ensuring the robot accurately reaches each vertex and managing joint limits and configurations.
    4. **Q**: Why is trajectory planning essential for triangular path execution? **A**: It ensures smooth transitions between vertices, reducing jerk and improving motion quality.
    5. **Q**: How do you handle the transitions between straight lines in IK? **A**: By calculating intermediate waypoints and ensuring the robot's path is continuous and smooth.
    6. **Q**: What is the significance of joint limits during triangular path execution? **A**: To prevent the robot from exceeding its mechanical capabilities and risking damage.
    7. **Q**: How can you use simulation to prepare for the triangular path execution? **A**: By running simulations to test and refine the trajectory before implementing it on the physical robot.
    8. **Q**: What sensors can assist in verifying the triangle's path accuracy? **A**: Cameras and encoders can help monitor the robot's position relative to the expected path.
    9. **Q**: How do you ensure safety during the triangular path experiment? **A**: By conducting pre- operation checks, ensuring a clear workspace, and having emergency procedures in place.
    10. **Q**: What metrics can you use to evaluate the success of the triangular path execution? **A**: Position accuracy, smoothness of movement, and successful completion of the path without errors.
12. **6-Axis Mitsubishi Robot (Industrial Robot) Programming for Pick and Place**
    1. **Q**: What is the primary function of a 6-axis industrial robot? **A**: To perform tasks such as assembly, welding, painting, and pick-and-place operations with high precision.
    2. **Q**: How do you program the Mitsubishi robot for pick-and-place tasks? **A**: By using the robot's programming language or graphical interface to define movements, sensor inputs, and actions.
    3. **Q**: What are the benefits of using a 6-axis robot for industrial applications? **A**: Greater flexibility, dexterity, and the ability to reach complex positions compared to simpler configurations.
    4. **Q**: What types of sensors are typically integrated into pick-and-place robots? **A**: Vision systems, proximity sensors, and force sensors can enhance the robot's capabilities.
    5. **Q**: How do you ensure the accuracy of the robot's movements during pick-and-place? **A**: By calibrating the robot, testing the program in simulation, and adjusting parameters as necessary.
    6. **Q**: What safety features are essential in industrial robot programming? **A**: Emergency stop functions, safety interlocks, and proper guarding of the workspace are critical.
    7. **Q**: How do you validate the performance of a pick-and-place operation? **A**: By measuring the precision of object placement and the reliability of the robot's actions during tests.
    8. **Q**: What programming environment is commonly used for Mitsubishi robots? **A**: The Mitsubishi robot can be programmed using software like MELFA Basic or RobotStudio.
    9. **Q**: What challenges can occur during the pick-and-place process? **A**: Variability in object position, unexpected obstacles, and sensor malfunctions can create challenges.
    10. **Q**: How can you improve the efficiency of a pick-and-place operation? **A**: By optimizing the robot's movements, reducing cycle times, and ensuring effective sensor integration.