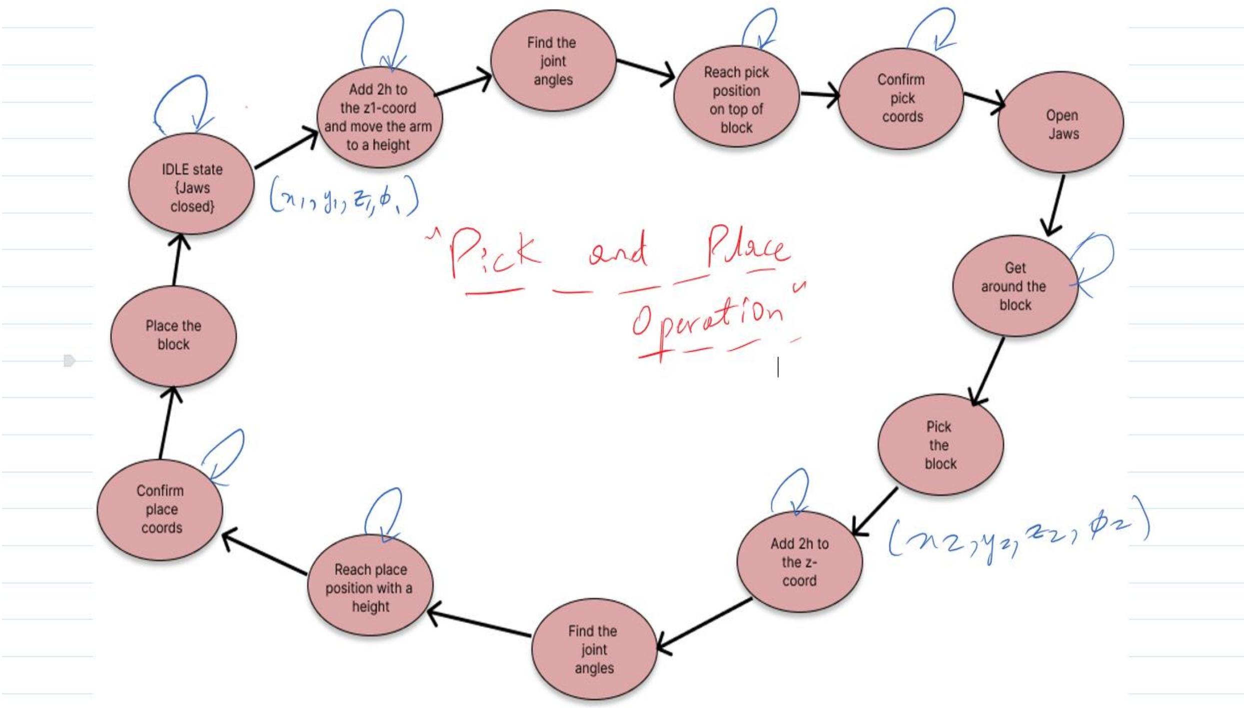
Task 6.1	FSM (30 points)

Draw a state-transition diagram of an FSM corresponding to the following scenario:

- System is in idle state till it receives pick location, (x_1, y_1, z_1, ϕ_1) and place location, (x_2, y_2, z_2, ϕ_2).
- Geometry of the object to be picked and placed, including its orientation, is known before hand.
- The locations can be assumed to lie in the interior of the manipulator's workspace, and the object is in an orientation so that it can be picked.
- System should verify the final placement location, before determining that the task has concluded.
- Smooth motion and accurate placement[a] is desirable.

[a] You'll have to plan your gripper picking and releasing strategy, considering the accuracy of your system, determined in earlier labs.



Task 6.2 FSM Implementation (50 points)

Implement the system described by the previous FSM in MATLAB[a] for Phantom X Pincher and the cube object. In addition to your implementation code, submit an explanation of your strategy, especially functions that were not developed previously, a video of your best execution, and identify and comment on points of improvement.

[a]You can implement the FSM using usual text-based programming, or Stateflow, a graphical programming environment for implementing FSMs in MATLAB. To learn further about Staeflow, see https://www.mathworks.com/help/stateflow/gs/finite-state-machines.html.

- **MATLAB Codes:**
    - **Pick_and_place function:**

```
function pick_and_place_cube(pick_coords,place_coords)
  current_state = 0;
  % removing offset in the z-coord
  pick_coords(3) = pick_coords(3) + 2;
  place_coords(3) = place_coords(3) + 2;

  height_of_block = 2.8; % cm


while 1
  % pick phenomenon
  current_state
  if current_state == 0
    arb = Arbotix('port', 'COM18', 'nservos', 5);
    arb.setpos([0,0,-pi/4,-pi/2,0],[25,25,25,25,25]);
    pause(5);
    error_ = find_error([0,0,0,0,0],[arb.getpos]);
    if error_ == false
      current_state = 1;
    else
```

```
            current_state = 0;
        end
    current_state
    elseif current_state == 1
        pick_coords(3) = pick_coords(3) + 2*height_of_block;
        coords_1 =
findOptimalsoln(pick_coords(1),pick_coords(2),pick_coords(3),pick_coords(4),pick_coords(
5));

        current_state = 2;
    current_state
    elseif current_state == 2
        setPosition(coords_1,0);
        pause(5);
        arb = Arbotix('port', 'COM18', 'nservos', 5);
        curr_pos = arb.getpos;
        error_1 = find_error([coords_1,0],[curr_pos]);
        if error_1 == false
            current_state = 3;
        else
            current_state = 2;
        end
    current_state
    elseif current_state == 3
        pick_coords(3) = pick_coords(3) - 4*height_of_block;
        coords_2 =
findOptimalsoln(pick_coords(1),pick_coords(2),pick_coords(3),pick_coords(4),pick_coords(
5));

        setPosition(coords_2,0);
        pause(5);
        arb = Arbotix('port', 'COM18', 'nservos', 5);
```

```
        curr_pos = arb.getpos();
        error_1 = find_error([coords_2,0],[curr_pos]);

        if error_1 == false
            current_state = 4;
        else
            current_state = 3;
        end
    current_state
    elseif current_state == 4
        setPosition(coords_2,1.2);
        pause(5);
        arb = Arbotix('port', 'COM18', 'nservos', 5);
         if arb.getpos(5) > 0.9
            current_state = 5;
        else
            current_state = 4;
         end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
    % place phenomenon
    current_state
     elseif current_state == 5
        place_coords(3) = place_coords(3) + 2*height_of_block;
        coords_3 =
findOptimalsoln(place_coords(1),place_coords(2),place_coords(3),place_coords(4),place_co
ords(5));
        current_state = 6;
    current_state
    elseif current_state == 6
        setPosition(coords_3,1.2);
```

```
        pause(5);
        arb = Arbotix('port', 'COM18', 'nservos', 5);
        curr_pos_ = arb.getpos
        error_3 = find_error([coords_3,1.1556],[curr_pos_]);
        if error_3 == false
            current_state = 7;
        else
            current_state = 6;
        end
    current_state
    elseif current_state == 7
        place_coords(3) = place_coords(3) - 4*height_of_block - 1;
        coords_4_ =
findOptimalsoln(place_coords(1),place_coords(2),place_coords(3),place_coords(4),place_co
ords(5));

        setPosition(coords_4_,1.2);
        pause(5);
        arb = Arbotix('port', 'COM18', 'nservos', 5);
        curr_pos = arb.getpos();
        error_4 = find_error([coords_4_,1.2],[curr_pos]);

        if error_4 == false
            current_state = 8;
        else
            current_state = 7;
        end
    current_state
    elseif current_state == 8
        setPosition(coords_4_,0.8);
        pause(5);
```

```
        arb = Arbotix('port', 'COM18', 'nservos', 5);
        current_state = 0;
    end
end
end
```

- **Helper Function (finding error):**

```
function margin_error = find_error(actual_angles,motor_angles)
    margin_error = false;
    for k = 1:5
        if (abs(motor_angles(k) - actual_angles(k))) < 0.05
            margin_error = false;

        else
            margin_error = true;
        end
    end
End
```

- **Tutorial of pick and place:**
    - **Command passed:**

pick_and_place_cube([0,21.6,0,-pi/2,0],[22,3,2,-7*pi/18,0])

Click here: https://youtu.be/WBjxYtIvbZk

Task 6.3 Manipulator Jacobian (20 points)

Use the DH parameters and homogeneous transformation, ${}^0T_4$, obtained in the previous lab to find the Jacobian for the manipulator in the lab.

- For convenience, a MATLAB function `createA(theta,d,a,alpha)` is available on canvas to easily create homogeneous transformations in symbolic form.
- Define your joint variables θ_i as functions of time, so that you can differentiate them.

```
syms theta_1(t) theta_2(t) theta_3(t) theta_4(t)
A1 = createA(theta_1,'d_1',0,-pi/2)
```

- The homogeneous transformation you'll obtain will be a 4×4 matrix function of t. To extract a particular entry of this matrix, you'll have to first evaluate it at a value of t and save it in an intermediate variable. For example, if `B` is a matrix symbolic function and you want to find matrix entry $(1, 2)$, then use:

```
tempVar = B(t);
entry = tempVar(1,2);
```

- You can find derivative of a symbolic expression using the MATLAB function `diff`. For example, `diff(f,x)` computes $\frac{\partial f}{\partial x}$.
- Chain rule will frequently yield simplified expressions.

```
 syms theta_1(t) theta_2(t) theta_3(t) theta_4(t)
 a = {0 ,11 ,11, 7};
 alpha = {pi/2, 0, 0, 0};
 d = {4, 0, 0, 0};
 thetas = {theta_1, theta_2, theta_3, theta_4};

 %Link offset and Link length in cm;
 T_01 =[cos(thetas{1}) -sin(thetas{1})*cos(alpha{1})
sin(thetas{1})*sin(alpha{1}) a{1}*cos(thetas{1});
         sin(thetas{1}) cos(thetas{1})*cos(alpha{1}) -
cos(thetas{1})*sin(alpha{1}) a{1}*sin(thetas{1});
         0 sin(alpha{1}) cos(alpha{1}) d{1};
         0 0 0 1];

 T_12 = [cos(thetas{2}) -sin(thetas{2})*cos(alpha{2})
sin(thetas{2})*sin(alpha{2}) a{2}*cos(thetas{2});
         sin(thetas{2}) cos(thetas{2})*cos(alpha{2}) -
cos(thetas{2})*sin(alpha{2}) a{2}*sin(thetas{2});
         0 sin(alpha{2}) cos(alpha{2}) d{2};
         0 0 0 1];

 T_23 =  [cos(thetas{3})  -sin(thetas{3})*cos(alpha{3})
sin(thetas{3})*sin(alpha{3})  a{3}*cos(thetas{3});
         sin(thetas{3})  cos(thetas{3})*cos(alpha{3})  -
cos(thetas{3})*sin(alpha{3})  a{3}*sin(thetas{3});
         0 sin(alpha{3}) cos(alpha{3}) d{3};
         0 0 0 1];

 T_34 = [cos(thetas{4})  -sin(thetas{4})*cos(alpha{4})
sin(thetas{4})*sin(alpha{4})  a{4}*cos(thetas{4});
         sin(thetas{4})  cos(thetas{4})*cos(alpha{4})  -
cos(thetas{4})*sin(alpha{4})  a{4}*sin(thetas{4});
         0 sin(alpha{4}) cos(alpha{4}) d{4};
         0 0 0 1];

 T_04 = T_01*T_12*T_23*T_34
```

```
T_04(t) =
```

$$
\begin{pmatrix}
-\cos(\theta_4(t))\,\sigma_5 - \sin(\theta_4(t))\,\sigma_3 & \sin(\theta_4(t))\,\sigma_5 - \cos(\theta_4(t))\,\sigma_3 & \sin(\theta_1(t)) & 11\cos(\theta_1(t))\cos(\theta_2(t)) - 11\cos(\theta_3(t))\,\sigma_7 - 11\sin(\theta_3(t))\,\sigma_8 - 7\cos(\theta_4(t))\,\sigma_5 - 7\sin(\theta_4(t))\,\sigma_3 - \frac{54645333600236621\sin(\theta_1(t))\sin(\theta_2(t))}{81129638414606681695789005144064} \\
\cos(\theta_4(t))\,\sigma_4 - \sin(\theta_4(t))\,\sigma_6 & -\cos(\theta_4(t))\,\sigma_6 - \sin(\theta_4(t))\,\sigma_4 & -\cos(\theta_1(t)) & \frac{54645333600236621\cos(\theta_1(t))\sin(\theta_2(t))}{81129638414606681695789005144064} + 11\cos(\theta_2(t))\sin(\theta_1(t)) + 11\cos(\theta_3(t))\,\sigma_{10} - 11\sin(\theta_3(t))\,\sigma_9 + 7\cos(\theta_4(t))\,\sigma_4 - 7\sin(\theta_4(t))\,\sigma_6 \\
\cos(\theta_4(t))\,\sigma_2 - \sin(\theta_4(t))\,\sigma_1 & -\cos(\theta_4(t))\,\sigma_1 - \sin(\theta_4(t))\,\sigma_2 & \frac{4967757600021511}{81129638414606681695789005144064} & 11\sin(\theta_2(t)) + 11\cos(\theta_2(t))\sin(\theta_3(t)) + 11\cos(\theta_3(t))\sin(\theta_2(t)) + 7\cos(\theta_4(t))\,\sigma_2 - 7\sin(\theta_4(t))\,\sigma_1 + 4 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

where

$\sigma_1 = \sin(\theta_2(t))\sin(\theta_3(t)) - \cos(\theta_2(t))\cos(\theta_3(t))$

$\sigma_2 = \cos(\theta_2(t))\sin(\theta_3(t)) + \cos(\theta_3(t))\sin(\theta_2(t))$

$\sigma_3 = \cos(\theta_3(t))\,\sigma_8 - \sin(\theta_3(t))\,\sigma_7$

$\sigma_4 = \cos(\theta_3(t))\,\sigma_{10} - \sin(\theta_3(t))\,\sigma_9$

$\sigma_5 = \cos(\theta_3(t))\,\sigma_7 + \sin(\theta_3(t))\,\sigma_8$

$\sigma_6 = \cos(\theta_3(t))\,\sigma_9 + \sin(\theta_3(t))\,\sigma_{10}$

$\sigma_7 = \frac{4967757600021511\sin(\theta_1(t))\sin(\theta_2(t))}{81129638414606681695789005144064} - \cos(\theta_1(t))\cos(\theta_2(t))$

$\sigma_8 = \cos(\theta_1(t))\sin(\theta_2(t)) + \frac{4967757600021511\cos(\theta_2(t))\sin(\theta_1(t))}{81129638414606681695789005144064}$

$\sigma_9 = \sin(\theta_1(t))\sin(\theta_2(t)) - \frac{4967757600021511\cos(\theta_1(t))\cos(\theta_2(t))}{81129638414606681695789005144064}$

$\sigma_{10} = \frac{4967757600021511\cos(\theta_1(t))\sin(\theta_2(t))}{81129638414606681695789005144064} + \cos(\theta_2(t))\sin(\theta_1(t))$

```
tempvar = T_04(t);
position = simplify(tempvar(1:3, 4))
```

position =

$$
\begin{pmatrix}
\frac{567907468902246806644826236159025\cos(\theta_1(t)+\theta_2(t)+\theta_3(t)+\theta_4(t))}{162259276829213363391578010288128} + \frac{892426022560673553299012656821325\cos(\theta_1(t)+\theta_2(t)+\theta_3(t))}{162259276829213363391578010288128} + \frac{892426022560673553299012656821325\cos(\theta_1(t)+\theta_2(t))}{162259276829213363391578010288128} + \frac{567907468902246737096219835857871\cos(\theta_2(t)-\theta_1(t)+\theta_3(t)+\theta_4(t))}{162259276829213363391578010288128} + \frac{892426022560673444008345456348083\cos(\theta_2(t)-\theta_1(t)+\theta_3(t))}{162259276829213363391578010288128} + \frac{89242602256\cdots}{162\cdots} \\
\frac{54645333600236621\cos(\theta_1(t))\sin(\theta_2(t))}{81129638414606681695789005144064} + 11\cos(\theta_2(t))\sin(\theta_1(t)) + 11\cos(\theta_3(t))\,\sigma_2 - 11\sin(\theta_3(t))\,\sigma_1 + 7\cos(\theta_4(t))\,(\cos(\theta_3(t))\,\sigma_2 - \sin(\theta_3(t))\,\sigma_1) - 7\sin(\theta_4(t))\,(\cos(\theta_3(t))\,\sigma_1 + \sin(\theta_3(t))\,\sigma_2) \\
11\sin(\theta_2(t)) + 7\sin(\theta_2(t)+\theta_3(t)+\theta_4(t)) + 11\sin(\theta_2(t)+\theta_3(t)) + 4
\end{pmatrix}
$$

where

$\sigma_1 = \sin(\theta_1(t))\sin(\theta_2(t)) - \frac{4967757600021511\cos(\theta_1(t))\cos(\theta_2(t))}{81129638414606681695789005144064}$

$\sigma_2 = \frac{4967757600021511\cos(\theta_1(t))\sin(\theta_2(t))}{81129638414606681695789005144064} + \cos(\theta_2(t))\sin(\theta_1(t))$

```
% To get the Jacobian, take the final position of T04 and differentiate it
% wrt the thetas
Jv = vpa(simplify(expand([diff(position,theta_1) diff(position,theta_2)
diff(position,theta_3) diff(position,theta_4)])),5)
```

Jv(t) =

$$\begin{pmatrix} 3.5\sin(\sigma_2)+5.5\sin(\sigma_4)-5.5\sin(\sigma_7)-3.5\sin(\sigma_3)-5.5\sin(\sigma_6)-5.5\sin(\sigma_8) & 5.5\sin(\sigma_7)-5.5\sin(\sigma_4)-3.5\sin(\sigma_2)-3.5\sin(\sigma_3)-5.5\sin(\sigma_6)-5.5\sin(\sigma_8) & -3.5\sin(\sigma_2)-5.5\sin(\sigma_4)-3.5\sin(\sigma_3)-5.5\sin(\sigma_6) & -3.5\sin(\sigma_2)-3.5\sin(\sigma_3) \\ 3.5\cos(\sigma_2)+5.5\cos(\sigma_4)+5.5\cos(\sigma_7)+3.5\cos(\sigma_3)+5.5\cos(\sigma_6)+5.5\cos(\sigma_8) & 3.5\cos(\sigma_3)-5.5\cos(\sigma_4)-5.5\cos(\sigma_7)-3.5\cos(\sigma_2)+5.5\cos(\sigma_6)+5.5\cos(\sigma_8) & 3.5\cos(\sigma_3)-5.5\cos(\sigma_4)-3.5\cos(\sigma_2)+5.5\cos(\sigma_6) & 3.5\cos(\sigma_3)-3.5\cos(\sigma_2) \\ 0 & 11.0\cos(\theta_2(t))+\sigma_1+\sigma_5 & \sigma_1+\sigma_5 & \sigma_1 \end{pmatrix}$$

where

$$\sigma_1 = 7.0\cos(\theta_2(t)+\theta_3(t)+\theta_4(t))$$

$$\sigma_2 = \theta_2(t) - 1.0\,\theta_1(t) + \theta_3(t) + \theta_4(t)$$

$$\sigma_3 = \theta_1(t) + \theta_2(t) + \theta_3(t) + \theta_4(t)$$

$$\sigma_4 = \theta_2(t) - 1.0\,\theta_1(t) + \theta_3(t)$$

$$\sigma_5 = 11.0\cos(\theta_2(t)+\theta_3(t))$$

$$\sigma_6 = \theta_1(t) + \theta_2(t) + \theta_3(t)$$

$$\sigma_7 = \theta_1(t) - 1.0\,\theta_2(t)$$

$$\sigma_8 = \theta_1(t) + \theta_2(t)$$