# FACULTY OF SCIENCE AND TECHNOLOGY
## Assignment (Laboratory) Coversheet

| | |
|---|---|
| Student ID number | U3312590 |
| Student Name | Jasmine Bajracharya |
| Unit name | Introduction to Computer Engineering |
| Unit number | 10089 |
| Name of lecturer/tutor | Dr. Julio Romero |
| Assignment topic | Forward Kinematics |
| Due date | 4th November, 2025 |
| Word Count | 2440 |

***You must keep a photocopy or electronic copy of your assignment.***

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details. Such referencing may need to meet unit-specific requirements as to format and style.

I give permission for my assignment to be copied, submitted and retained for the electronic checking of plagiarism.

**Signature of student:** **Date: 30th October 2025**

(Students submitting work electronically can type their name in the space for signature above, but must produce a signed copy of this coversheet on request.)

**Date of submission: 30th October 2025**

# 1. Theoretical Framework First FK Model (25 marks)

**1.1 Attach reference frames to the robot following the DH convention.**

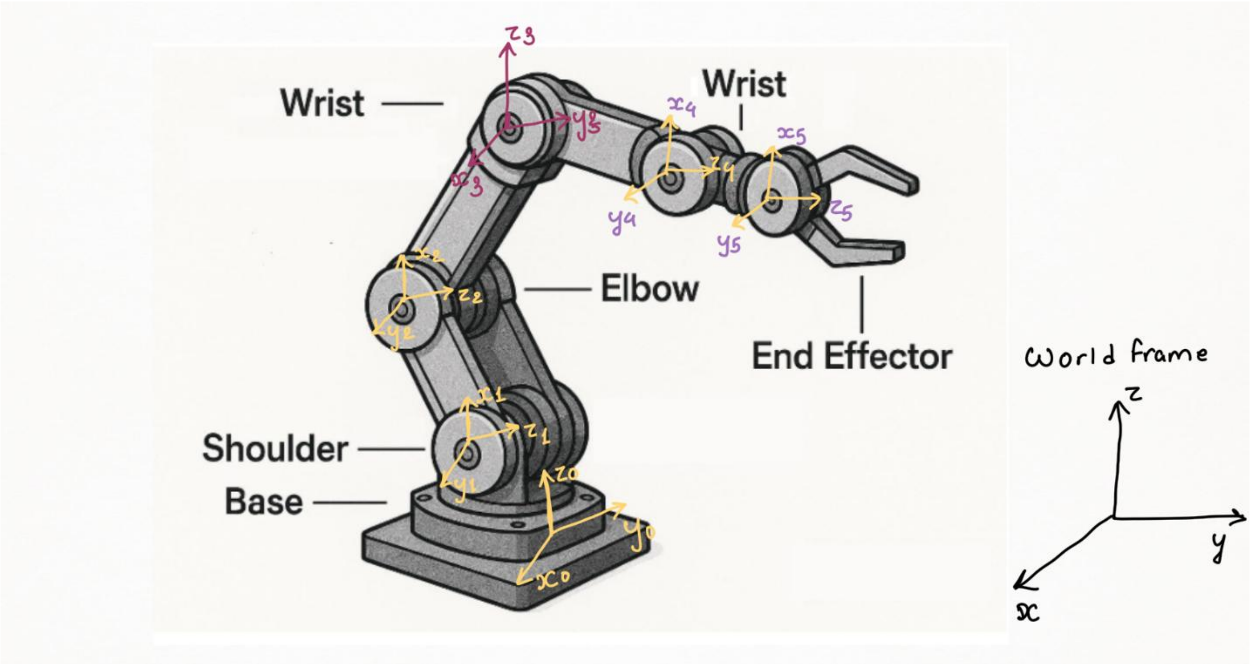



Figure: Assignment of reference frames to the robot joints

**1.2 Write down the corresponding DH parameters.**

As per the frames attached, the DH convention can be obtained as:

| I | Theta (θ) | Di | Ai | α |
|---|---|---|---|---|
| 1 | Q1 | D1 | 0 | Pi/2 |
| 2 | Q2 | 0 | A2 | 0 |
| 3 | Q3 | 0 | A3 | 0 |
| 4 | Q4 | 0 | 0 | Pi/2 |
| 5 | Q5 | 0 | 0 | -pi/2 |
| 6 | Q6 | D6 | 0 | 0 |

**1.3 Write down corresponding transformation matrices.**

The corresponding transformation matrices are given as:

For each joint, we obtain transformation as:

Using the formula for $A_i$, we get

$$A_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_1 C\theta_1 \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $C\theta_i = \cos\theta_i$, $S\theta_i = \sin\theta_i$, $C\alpha_i = \cos\alpha_i$, $S\alpha_i = \sin\alpha_i$

For joint 1:

$\theta_i = \theta_1 \quad d_i = d_1 \quad a_1 = 0 \quad \alpha_1 = \pi/2$ ; $\sin\alpha_1 = \sin\pi/2 = 1$

$\cos\alpha_1 = \cos\pi/2 = 0$

$$A_1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For joint 2:

$\theta_i = \theta_2 \quad d_2 = 0 \quad a_2 = a_2 \quad \alpha_2 = 0$ ; $\sin\alpha_2 = \sin 0 = 0$

$\cos\alpha_2 = \cos 0 = 1$

$$A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2 \cdot \cos\theta_1 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2 \sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2/4

For joint 3:

$\theta_3 = \theta_3 \quad d_3 = 0 \quad a_3 = a_3 \quad \alpha_3 = 0 \quad ; \sin\alpha_3 = \sin 0 = 0$

$\cos\alpha_3 = \cos 0 = 1$

$$A_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_3\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & a_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For joint 4:

$\theta_4 = \theta_4 \quad d_4 = 0 \quad a_4 = 0 \quad \alpha_4 = \pi/2 \quad ; \sin\alpha = \sin\pi/2 = 1$

$\cos\alpha = \cos\pi/2 = 0$

$$A_4 = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For joint 5.

$\theta_5 = \theta_5 \quad d_5 = 0 \quad a_5 = 0 \quad \alpha_5 = -\pi/2 \quad ; \sin\alpha = \sin(-\pi/2) = -1$

$\cos\alpha = \cos(-\pi/2) = 0$

$$A_5 = \begin{bmatrix} \cos\theta_5 & 0 & -\sin\theta_5 & 0 \\ \sin\theta_5 & 0 & \cos\theta_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For joint 6:

$\theta_6 = \theta_6 \quad , d_6 = d_6 \quad , a_6 = 0 \quad , \alpha_6 = 0 \quad ; \sin\alpha_6 = \sin 0 = 0$

$\cos\alpha_6 = \cos 0 = 1$

$$A_6 = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**1.4 Write down the matrix T of transformations (end effector relative to base)**

The final transformation matrix can be obtained as:
(position of end-effector w.r.t base):

$${}^{0}T_6 = {}^{0}T_1 \, {}^{1}T_2 \, {}^{2}T_3 \, {}^{3}T_4 \, {}^{4}T_5 \, {}^{5}T_6$$

$${}^{0}T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6$$

**1.5 Choose a set of suitable pose parameters based on the joint limits shown in the tutorial handout.**

The joint angles chosen are as follows: q_deg = [ 30, 35, -10, 20, 10, -100 ];

**1.6 Calculate the transformation matrix T based on the posed proposed in point 1.5 above. You need to use any CAD tool (such as MATLAB) to perform the matrix multiplications.**

The transformation matrix T based on the proposed angles are as follows:

The MATLAB code for the following transformation is as follows:

```
% DH parameters
d = [99 0 0 0 0 60];
a = [0 120 195 0 0 0];
alpha = [pi/2 0 0 pi/2 -pi/2 0];

pose_deg = [30 35 -10 20 10 -100];
theta = deg2rad(pose_deg);   % converting degrees to radians

% To build the matrix
dh = @(th, d, a, al)[
   cos(th) -sin(th)*cos(al)  sin(th)*sin(al) a*cos(th);
   sin(th)  cos(th)*cos(al) -cos(th)*sin(al) a*sin(th);
   0       sin(al)         cos(al)        d;
   0       0               0              1];

% transformation matrices
A1 = dh(theta(1), d(1), a(1), alpha(1));
```

```
A2 = dh(theta(2), d(2), a(2), alpha(2));
A3 = dh(theta(3), d(3), a(3), alpha(3));
A4 = dh(theta(4), d(4), a(4), alpha(4));
A5 = dh(theta(5), d(5), a(5), alpha(5));
A6 = dh(theta(6), d(6), a(6), alpha(6));

% Final transformation T0_6
T06 = A1*A2*A3*A4*A5*A6;

% Extracting position of end effector
pos = T06(1:3,4);

disp('Transformation matrix T0_6 = ')
disp(T06)
disp('End-effector position [x y z] (mm) = ')
disp(pos.')
```

The matrix obtained from the calculation is given as:

```
Transformation matrix T0_6 =
    0.4833    0.7857    0.3861  261.3455
    0.3138    0.2562   -0.9143   82.6584
   -0.8173    0.5630   -0.1228  242.8725
         0         0         0    1.0000
```

**1.7 Write down the theoretical final position of robot's end effector**.

The code for extracting the robot's end effector position with respect to the base is given by

```
% Extracting position of end effector
pos = T06(1:3,4);

disp('Transformation matrix T0_6 = ')
disp(T06)
disp('End-effector position [x y z] (mm) = ')
disp(pos.')
```

The position of the end effector is obtained as:

```
End-effector position [x y z] (mm) =
  261.3455   82.6584  242.8725
```

## 2. Theoretical Framework Second FK Model (15 marks)

**2.1 Choose a set of suitable pose parameters based on the joint limits shown in the tutorial handout.**

The chosen set of limits are
For theta1: -170 to 170
Theta 2: -90 to 90
Theta 3: -120 to 120
Theta 4: -180 to 180
Theta 5: -120 to 120
Theta 6: -180 to 180

We chose the angles to be q_deg = −45°, q2 = +30°, q3 = −60°, q4 = +90°, q5 = −30°, q6 = +60°

**2.2 Calculate the transformation matrix T based on the posed proposed in point 2.1 above (end effector relative to base). You need to use any CAD tool (such as MATLAB) to perform the matrix multiplications.**

```
% DH parameters
d = [99 0 0 0 0 60];
a = [0 120 195 0 0 0];
alpha = [pi/2 0 0 pi/2 -pi/2 0];

% New joint angles (degrees)
pose_deg = [-45 30 -60 90 -30 60];

% Apply joint limits
limits = [
   -170 170;
   -90  90;
   -120 120;
   -180 180;
   -120 120;
   -180 180
];

% Clamp angles to limits
pose_deg = max(min(pose_deg, limits(:,2).'), limits(:,1).');

% Convert to radians
theta = deg2rad(pose_deg);
```

```
% DH transformation function
dh = @(th, d, a, al)[
   cos(th) -sin(th)*cos(al)  sin(th)*sin(al) a*cos(th);
   sin(th)  cos(th)*cos(al) -cos(th)*sin(al) a*sin(th);
   0       sin(al)         cos(al)         d;
   0       0               0               1];

% Compute individual transformation matrices
A1 = dh(theta(1), d(1), a(1), alpha(1));
A2 = dh(theta(2), d(2), a(2), alpha(2));
A3 = dh(theta(3), d(3), a(3), alpha(3));
A4 = dh(theta(4), d(4), a(4), alpha(4));
A5 = dh(theta(5), d(5), a(5), alpha(5));
A6 = dh(theta(6), d(6), a(6), alpha(6));

% Final transformation matrix
T06 = A1*A2*A3*A4*A5*A6;

% Extract end-effector position
pos = T06(1:3,4);

disp('Transformation matrix T0_6 = ')
disp(T06)
disp('End-effector position [x y z] (mm) = ')
disp(pos.')
```

```
>> Solution_2
Transformation matrix T0_6 =
   -0.2005   -0.8775   -0.4356  166.7616
    0.5540    0.2652   -0.7891 -240.2463
    0.8080   -0.3995    0.4330   87.4808
         0         0         0    1.0000

End-effector position [x y z] (mm) =
  166.7616 -240.2463   87.4808
```

**2.3 Write down the theoretical final position of robot's end effector.**

The position of the end effector is as:
X= 166.7617 Y= -240.2463 and z=87.4808

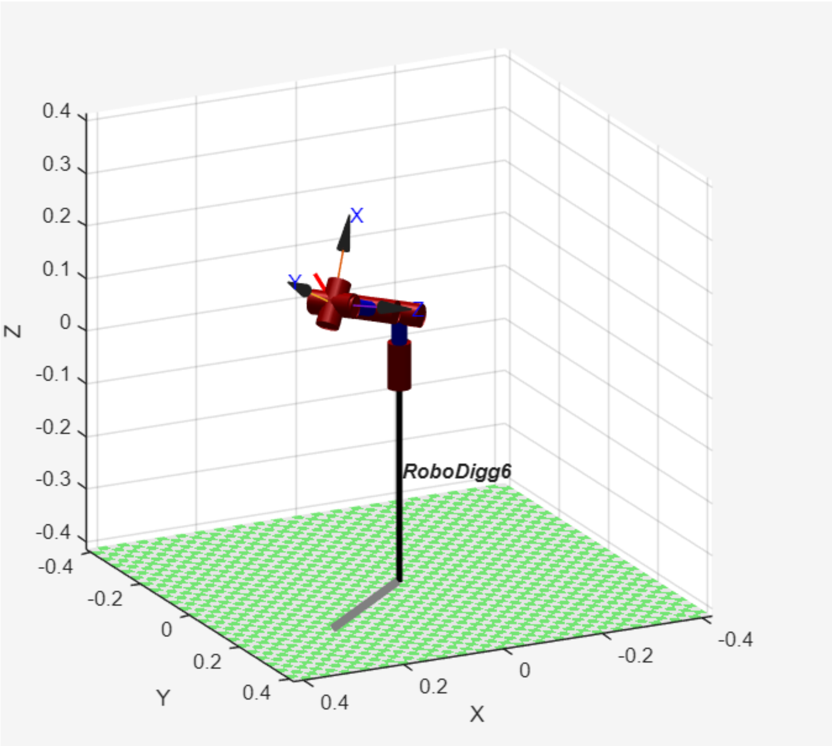
## 3. Simulation of First FK Model Using MATLAB (15 marks)

**3.1** Using the method *fkine* in MATLAB, obtain the transformation matrix T (end effector relative to base frame).

```
% Forward Kinematics

% Link lengths in metres for RTB (convert mm→m)
L1 = 99e-3;   % base to shoulder (d1)
L2 = 120e-3;  % shoulder to elbow (a2)
L3 = 195e-3;  % elbow to wrist (a3)
L6 = 60e-3;   % small tool offset (d6)

% Standard DH using Link('d',d, 'a',a, 'alpha',alpha)
L(1) =Link('d', L1, 'a', 0, 'alpha', pi/2);
L(2) =Link('d', 0, 'a', L2, 'alpha', 0);
L(3) =Link('d', 0, 'a', L3, 'alpha', 0);
L(4) =Link('d', 0, 'a', 0, 'alpha', pi/2);
L(5) =Link('d', 0, 'a', 0, 'alpha', pi/2);
L(6) =Link('d', 0, 'a', 0, 'alpha', 0 );


robot = SerialLink(L, 'name', 'RoboDigg6');

robot.qlim = deg2rad([ -170 170;  -90  90;  -120 120;  -180 180;  -120 120;  -180 180 ]);

q_deg = [ 30, 35, -10, 20, 10, -100 ];
q = deg2rad(q_deg);

% FK
T06 = robot.fkine(q);
disp('End-effector pose ^0T6:'); disp(T06);
disp('Position (x y z) in m:'); disp(transl(T06));

% Visualization
figure(1); clf; robot.plot(q, 'workspace', [-0.4 0.4 -0.4 0.4 0 0.5]);
trplot(T06, 'frame', '6', 'length', 0.05);
title('3.3 Robot configuration after applying transformation T');
view(135, 25); grid on;

hold on;
trplot(T06, 'frame', '6', 'color', 'r', 'length', 0.05);
title('3.4 End effector reference frame after applying transformation T');
```

```
% Animation of a a small joint-space move
q2 = q + deg2rad([10 -15 10 0 0 20]);
title('3.5 Animation of joint space move');
figure(2); clf; robot.plot([q; q2], 'fps', 20, 'trail', {'r', 'LineWidth', 2});
```

**3.2 Display robot's end effector position (take a screenshot from MATLAB)**



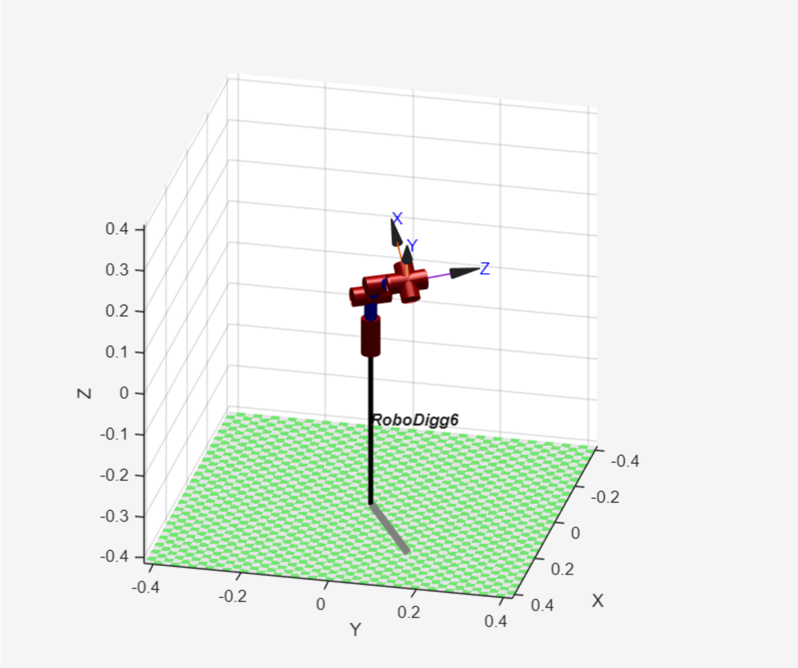

```
End-effector pose ^0T6:
    -0.7229     0.5731    -0.3861     0.2382
    -0.3825     0.1334     0.9143     0.1375
     0.5754     0.8086     0.1228     0.2502
          0          0          0          1
Position (x y z) in m:
    0.2382     0.1375     0.2502
```

**3.3 Plot the state of the robot after applying the transformation T using MATLAB.**

From the code used in 3.1 above, the following code block plots the state of the robot after applying transformations:

```
% Visualization
figure(1); clf; robot.plot(q, 'workspace', [-0.4 0.4 -0.4 0.4 0 0.5]);
trplot(T06, 'frame', '6', 'length', 0.05);
```

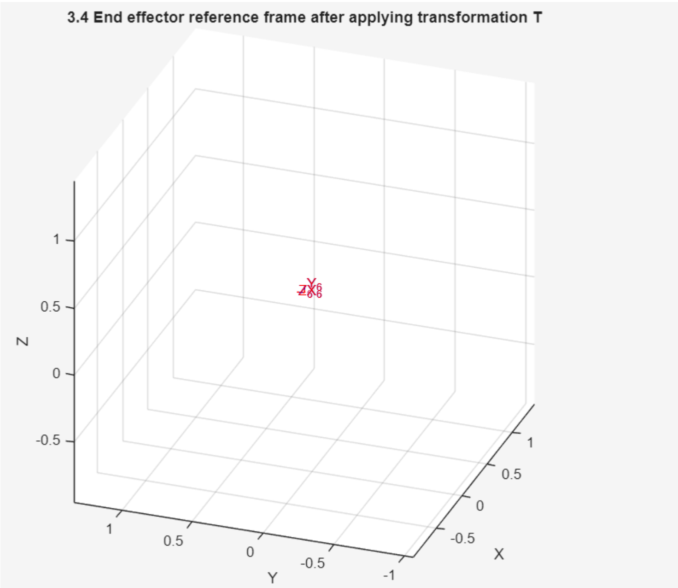
```
title('3.3 Robot configuration after applying transformation T');
view(135, 25); grid on;
```




**3.4 Plot end effector's reference frame after applying the transformation T using MATLAB.**

From the code used in 3.1 above, the following code block plots the end effector's reference after applying the transformation T.

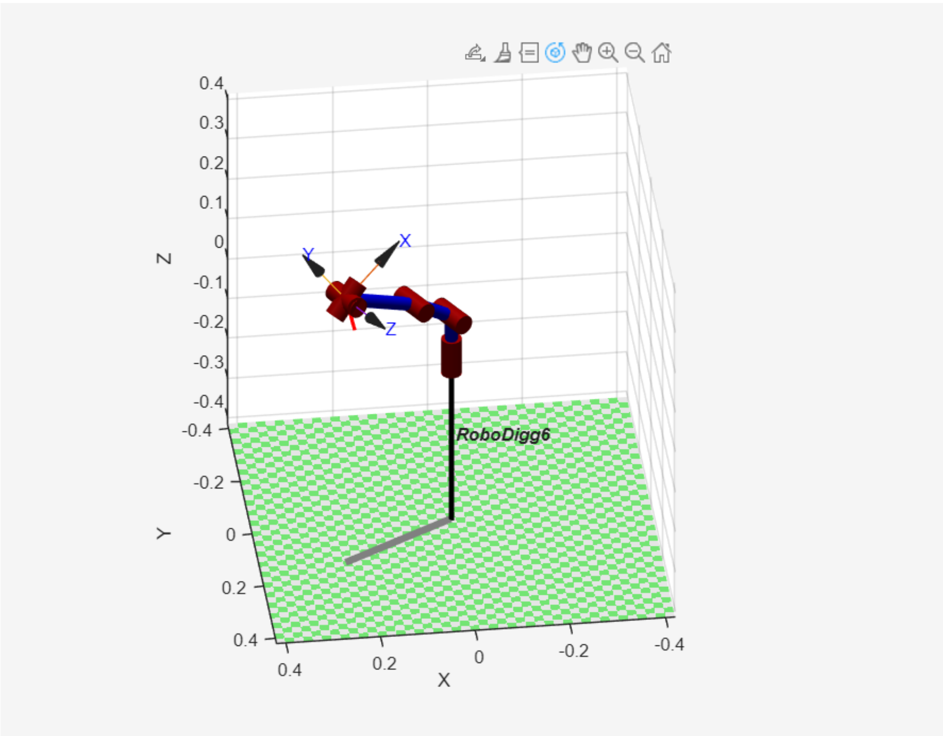
```
hold on;
trplot(T06, 'frame', '6', 'color', 'r', 'length', 0.05);
title('3.4 End effector reference frame after applying transformation T');
```



**3.5 Animate a small joint-space move using MATLAB.**

In the code used in 3.1, the following segment of code animates the robot arm.

```
% Animate a small joint-space move
q2 = q + deg2rad([10 -15 10 0 0 20]);
figure(2); clf;
while true
  % Forward move
  robot.plot([q; q2], 'fps', 20, 'trail', {'r', 'LineWidth', 2});
   pause(0.5); % short pause

   % Backward move (return to start)
    robot.plot([q2; q], 'fps', 20, 'trail', {'b', 'LineWidth', 2});
    pause(0.5);
end
```




**Note: The simulation video can be accessed through the file Robot_Arm_animation_simulation_3_5*

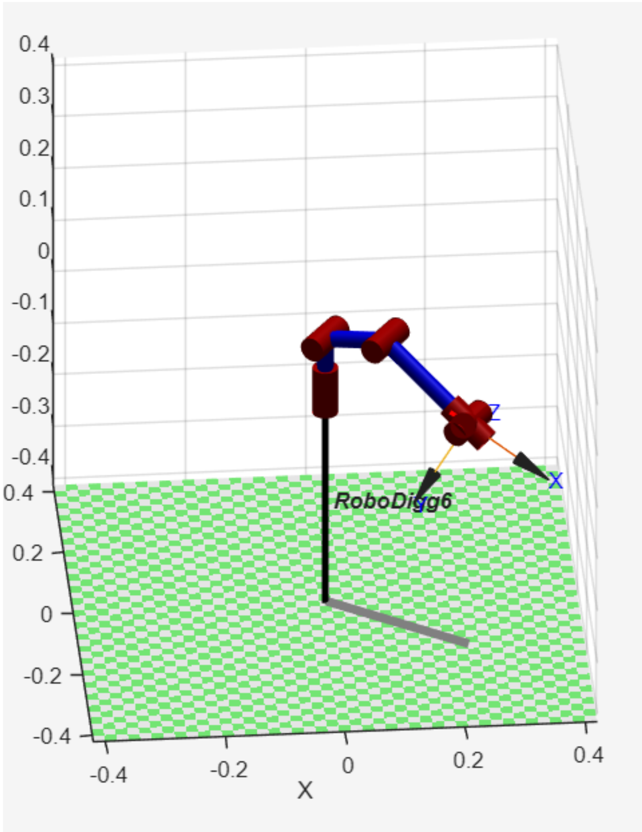
# 4. Simulation of Second FK Model Using MATLAB (15 marks)

**4.1** Using the method *fkine* in MATLAB, obtain the transformation matrix T (end effector relative to base frame).

**MATLAB code**

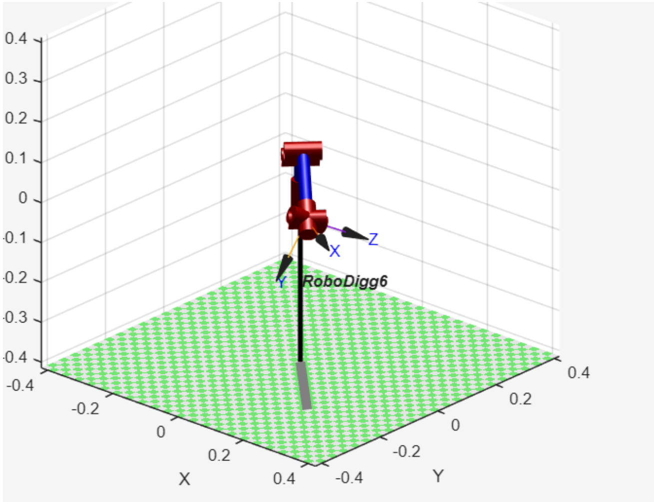
```
% Link lengths in metres for RTB
L1 = 99e-3;   % base to shoulder (d1)
L2 = 120e-3;  % shoulder to elbow (a2)
L3 = 195e-3;  % elbow to wrist (a3)
L6 = 60e-3;   % small tool offset (d6)

% Standard DH using Link('d',d, 'a',a, 'alpha',alpha)
L(1) =Link('d', L1, 'a', 0, 'alpha', pi/2);
L(2) =Link('d', 0, 'a', L2, 'alpha', 0);
L(3) =Link('d', 0, 'a', L3, 'alpha', 0);
L(4) =Link('d', 0, 'a', 0, 'alpha', pi/2);
L(5) =Link('d', 0, 'a', 0, 'alpha', pi/2);
L(6) =Link('d', 0, 'a', 0, 'alpha', 0 );


robot = SerialLink(L, 'name', 'RoboDigg6');

% Joint limits (conservative classroom limits; refine per hardware)
robot.qlim = deg2rad([ -170 170;  -90  90;  -120 120;  -180 180;  -120 120;  -180 180 ]);

% Example pose (degrees → radians)
q_deg = [ -45  +30 -60 +90 -30  +60 ];
q = deg2rad(q_deg);

% FK
T06 = robot.fkine(q);
disp('End-effector pose ^0T6:'); disp(T06);
disp('Position (x y z) in m:'); disp(transl(T06));

% Visualize by plotting the resulting frame
figure(1); clf; robot.plot(q, 'workspace', [-0.4 0.4 -0.4 0.4 0 0.5]);
trplot(T06, 'frame', '6', 'length', 0.05);
view(135, 25); grid on;

% Animate a small joint-space move
```

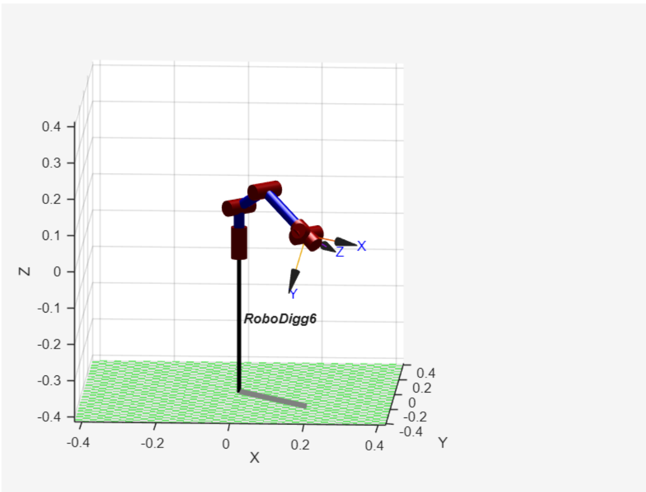
```
q2 = q + deg2rad([10 -15 10 0 0 20]);
figure(2); clf;
while true
   % Forward move
   robot.plot([q; q2], 'fps', 20, 'trail', {'r', 'LineWidth', 2});
   pause(0.5); % short pause

   % Backward move (return to start)
   robot.plot([q2; q], 'fps', 20, 'trail', {'b', 'LineWidth', 2});
   pause(0.5);
end
```



```
End-effector pose ^0T6:
       0.8602   -0.2652    0.4356    0.1929
      -0.5066   -0.3472    0.7891   -0.1929
     -0.05801   -0.8995    -0.433    0.0615
            0         0         0         1
Position (x y z) in m:
     0.1929   -0.1929    0.0615
```

**4.2** Display robot's end effector position (take a screenshot from MATLAB)

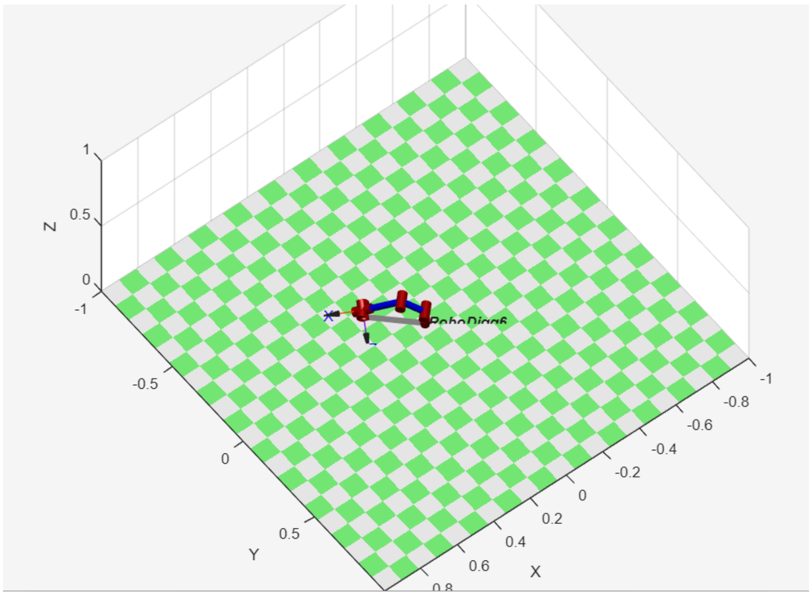


**4.3 Plot the state of the robot after applying the transformation T using MATLAB.**

The state of the robot after applying the transformation is plotted using the code block from solution 4.1 as:

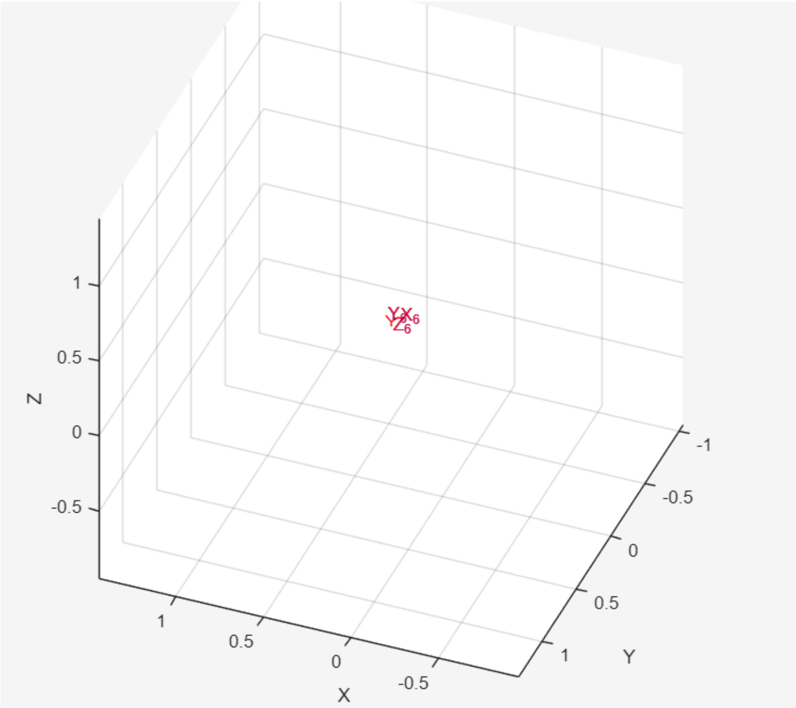
```
figure(2); clf;
robot.plot(q, 'workspace', [-1 1 -1 1 0 1]);
title('4.3 Robot configuration after applying transformation T');
grid on; view(135, 25);
```




**4.4 Plot end effector's reference frame after applying the transformation T using MATLAB.**

The end effector's reference frame is given using the code block from solution 4.1 is given as:

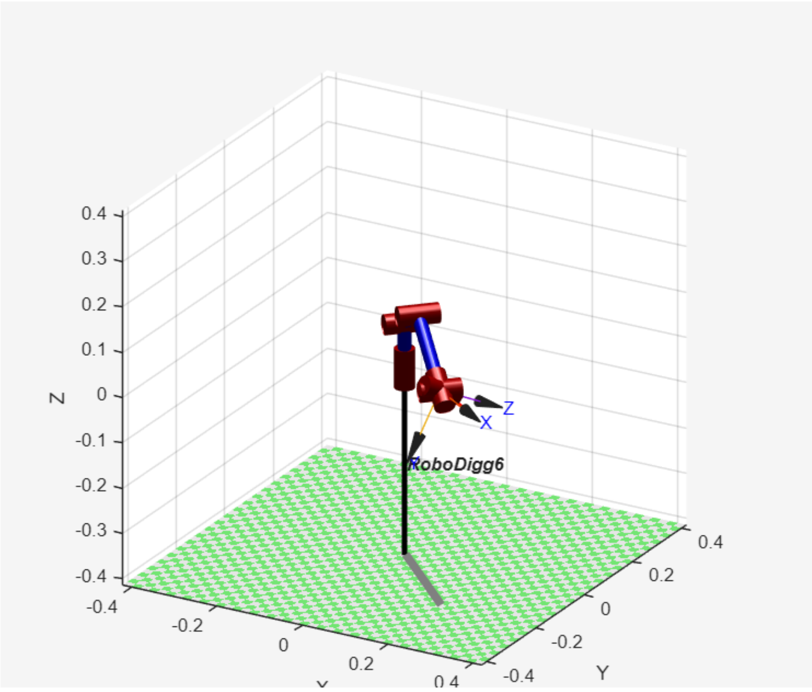
```
% Visualize by plotting the resulting frame
figure(1); clf; robot.plot(q, 'workspace', [-0.4 0.4 -0.4 0.4 0 0.5]);
trplot(T06, 'frame', '6', 'length', 0.05);
view(135, 25); grid on;
```




**4.5 Animate a small joint-space move using MATLAB.**

The following code block from the above solution 4.1 is used to animate the robot arm as:

```
% Animate a small joint-space move
q2 = q + deg2rad([10 -15 10 0 0 20]);
figure(2); clf;
while true
   % Forward move
   robot.plot([q; q2], 'fps', 20, 'trail', {'r', 'LineWidth', 2});
   pause(0.5); % short pause

   % Backward move (return to start)
   robot.plot([q2; q], 'fps', 20, 'trail', {'b', 'LineWidth', 2});
   pause(0.5);
end
```




**Note: The simulation video of the animation can be accessed through the file robotic_arm_animation_4_4*

## 5. Hardware Implementation First FK Model (15 marks)

**5.1 Using ARDUINO, configure the joints of your robot according to the parameters simulated in point 1.5 above**.

The code used in Arduino is as:

```
#include <Servo.h>

const uint8_t PIN[6] = {3, 5, 6, 10, 11, 9}; // base, shoulder, elbow, wristPitch, wristRoll, gripper
int8_t  DIR[6]     = {+1, -1, +1, +1, +1, +1};
int16_t OFFSET[6]  = { 90,  90,  90,  90,  90,  45};
int16_t MIN_ANG[6] = {  5,   5,  10,   5,   5,   0};
int16_t MAX_ANG[6] = {175, 175, 170, 175, 175, 100};
Servo    servo[6];

int16_t dhToServoDeg(int joint, float dh_deg){
  long v = (long)DIR[joint]* (long)dh_deg + (long)OFFSET[joint];
  if(v < MIN_ANG[joint]) v = MIN_ANG[joint];
  if(v > MAX_ANG[joint]) v = MAX_ANG[joint];
  return (int16_t)v;
}
```

```
void moveToPose(float q_deg[6], uint16_t ms_per_step=20, uint16_t steps=50){
 int16_t s[6], t[6];
 for(int i=0;i<6;++i){ s[i]=servo[i].read(); t[i]=dhToServoDeg(i,q_deg[i]); }
 for(uint16_t k=1;k<=steps;++k){
  float u=(float)k/steps; float e=u*u*(3-2*u); // smoothstep
  for(int i=0;i<6;++i){ int val = (int)(s[i] + e*(t[i]-s[i])); servo[i].write(val); }
  delay(ms_per_step);
 }
}

void setup(){
 for(int i=0;i<6;++i) servo[i].attach(PIN[i]);
 Serial.begin(115200);
 float q_home[6]={0,0,0,0,0,0};
 moveToPose(q_home,15,60);
 Serial.println(F("Send q in degrees as: q=a,b,c,d,e,f"));
 Serial.println(F("Example: q=30,35,-25,20,15,-40"));
}

void loop(){
 if(Serial.available()){
  String line=Serial.readStringUntil('\n'); line.trim();
  if(line.startsWith("q=")){
   float q[6]; int last=2;
   for(int i=0;i<6;++i){ int comma=line.indexOf(',',last);
    String tok=(comma==-1)? line.substring(last): line.substring(last,comma);
    q[i]=tok.toFloat(); last=comma+1; if(comma==-1) break; }
   moveToPose(q,15,60); Serial.println(F("OK"));
  }
 }
}
```

The serial monitor was used and the angles, q=30, 35, -10, 20, 10, -100 was sent through the serial monitor to obtain the position of the robot arm.

**5.2 Verify that the final position of your robot matches the one you calculated in point 1.7 above.**

The position of the robot matches the position as obtained through the MATLAB code above.
The position of the robot is observed as:

**Note: The video of the robot arm reaching the pose can be accessed through the file Hardware_simulation_1*

*Deliverable: ARDUINO code and a short video showing the robot reaching the pose.*

## 6. Hardware Implementation Second FK Model (15 marks)

5.1 Using ARDUINO, configure the joints of your robot according to the parameters simulated in point 1.5 above.

The code used in Arduino is as:

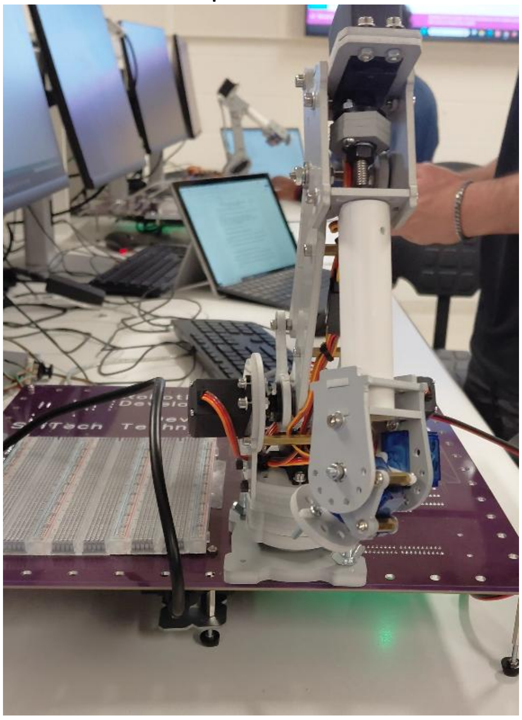
```
#include <Servo.h>

const uint8_t PIN[6] = {3, 5, 6, 10, 11, 9}; // base, shoulder, elbow, wristPitch, wristRoll, gripper
int8_t  DIR[6]     = {+1, -1, +1, +1, +1, +1};
int16_t OFFSET[6]  = { 90,  90,  90,  90,  90,  45};
int16_t MIN_ANG[6] = {  5,   5,  10,   5,   5,   0};
int16_t MAX_ANG[6] = {175, 175, 170, 175, 175, 100};
Servo    servo[6];

int16_t dhToServoDeg(int joint, float dh_deg){
  long v = (long)DIR[joint]* (long)dh_deg + (long)OFFSET[joint];
  if(v < MIN_ANG[joint]) v = MIN_ANG[joint];
  if(v > MAX_ANG[joint]) v = MAX_ANG[joint];
  return (int16_t)v;
}

void moveToPose(float q_deg[6], uint16_t ms_per_step=20, uint16_t steps=50){
  int16_t s[6], t[6];
  for(int i=0;i<6;++i){ s[i]=servo[i].read(); t[i]=dhToServoDeg(i,q_deg[i]); }
  for(uint16_t k=1;k<=steps;++k){
    float u=(float)k/steps; float e=u*u*(3-2*u); // smoothstep
    for(int i=0;i<6;++i){ int val = (int)(s[i] + e*(t[i]-s[i])); servo[i].write(val); }
    delay(ms_per_step);
  }
}

void setup(){
  for(int i=0;i<6;++i) servo[i].attach(PIN[i]);
  Serial.begin(115200);
  float q_home[6]={0,0,0,0,0,0};
  moveToPose(q_home,15,60);
  Serial.println(F("Send q in degrees as: q=a,b,c,d,e,f"));
  Serial.println(F("Example: q=30,35,-25,20,15,-40"));
}

void loop(){
```

```
 if(Serial.available()){
  String line=Serial.readStringUntil('\n'); line.trim();
  if(line.startsWith("q=")){
   float q[6]; int last=2;
   for(int i=0;i<6;++i){ int comma=line.indexOf(',',last);
    String tok=(comma==-1)? line.substring(last): line.substring(last,comma);
    q[i]=tok.toFloat(); last=comma+1; if(comma==-1) break; }
   moveToPose(q,15,60); Serial.println(F("OK"));
  }
 }
}
```

The serial monitor was used and the angles, q= -45 , 30, -60, 90, -30, 60 was sent through the serial monitor to obtain the position of the robot arm.

**5.2 Verify that the final position of your robot matches the one you calculated in point 2.3 above.**

The robot arm reaches the pose and matches the pose as shown in the MATLAB code simulation. The position of the arm is observed as:



**Note: The video of the robot arm reaching the pose can be accessed through the file Hardware_simulation_2*

## REFERENCES

[1] Corke, P. (2023). *Robotics, Vision and Control: Fundamental Algorithms in Python* (3rd ed.). Springer Nature.

[2] Corke, P., Jachimczyk, W., & Pilat, R. (2023). *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*