**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: A close-up of a signature

AI-generated content may be incorrect. 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. **Attach reference frames to the robot following the DH convention.**

A mechanical arm with text and words

AI-generated content may be incorrect.

Figure: Assignment of reference frames to the robot joints

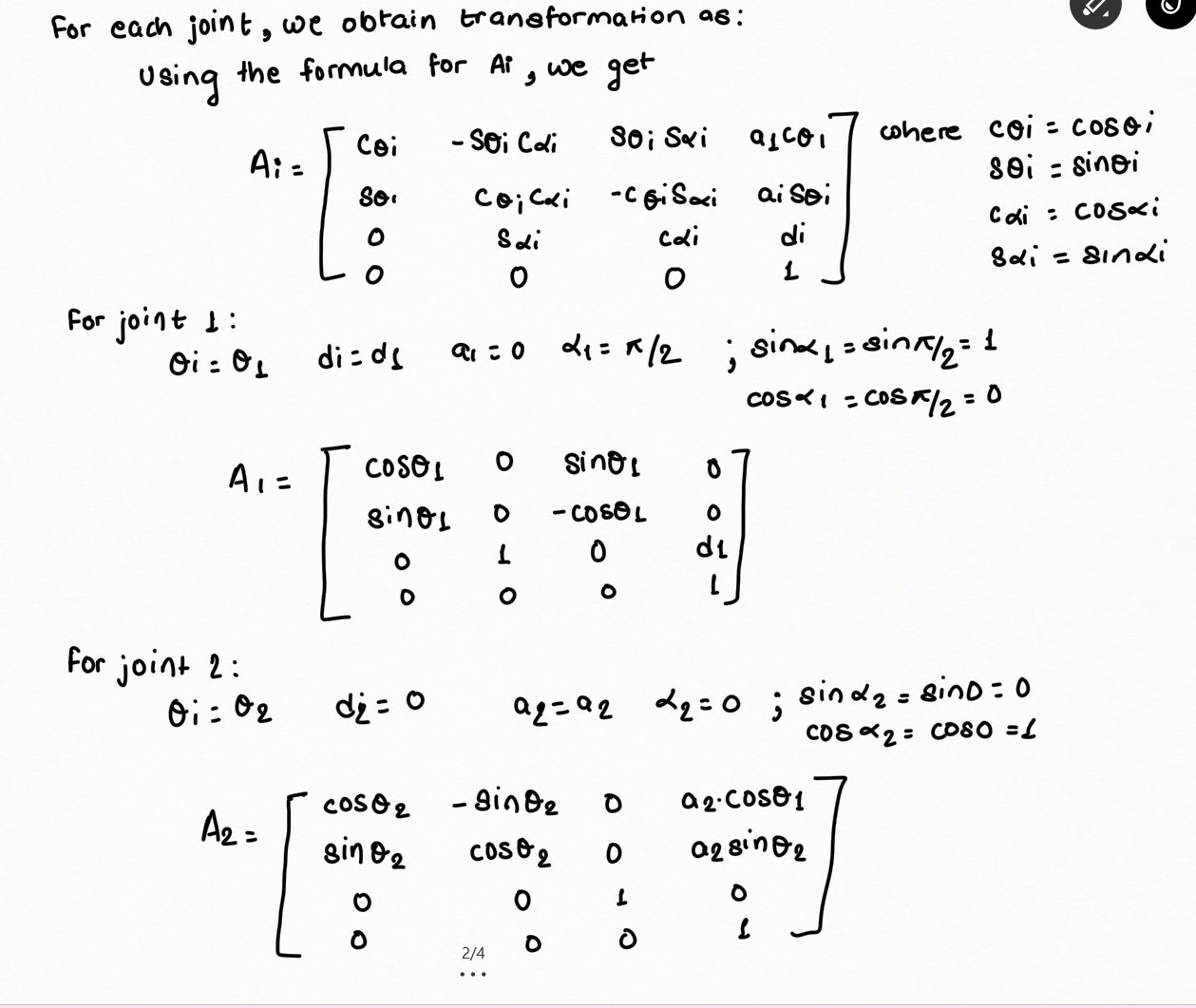
* 1. **Write down the corresponding DH parameters.**

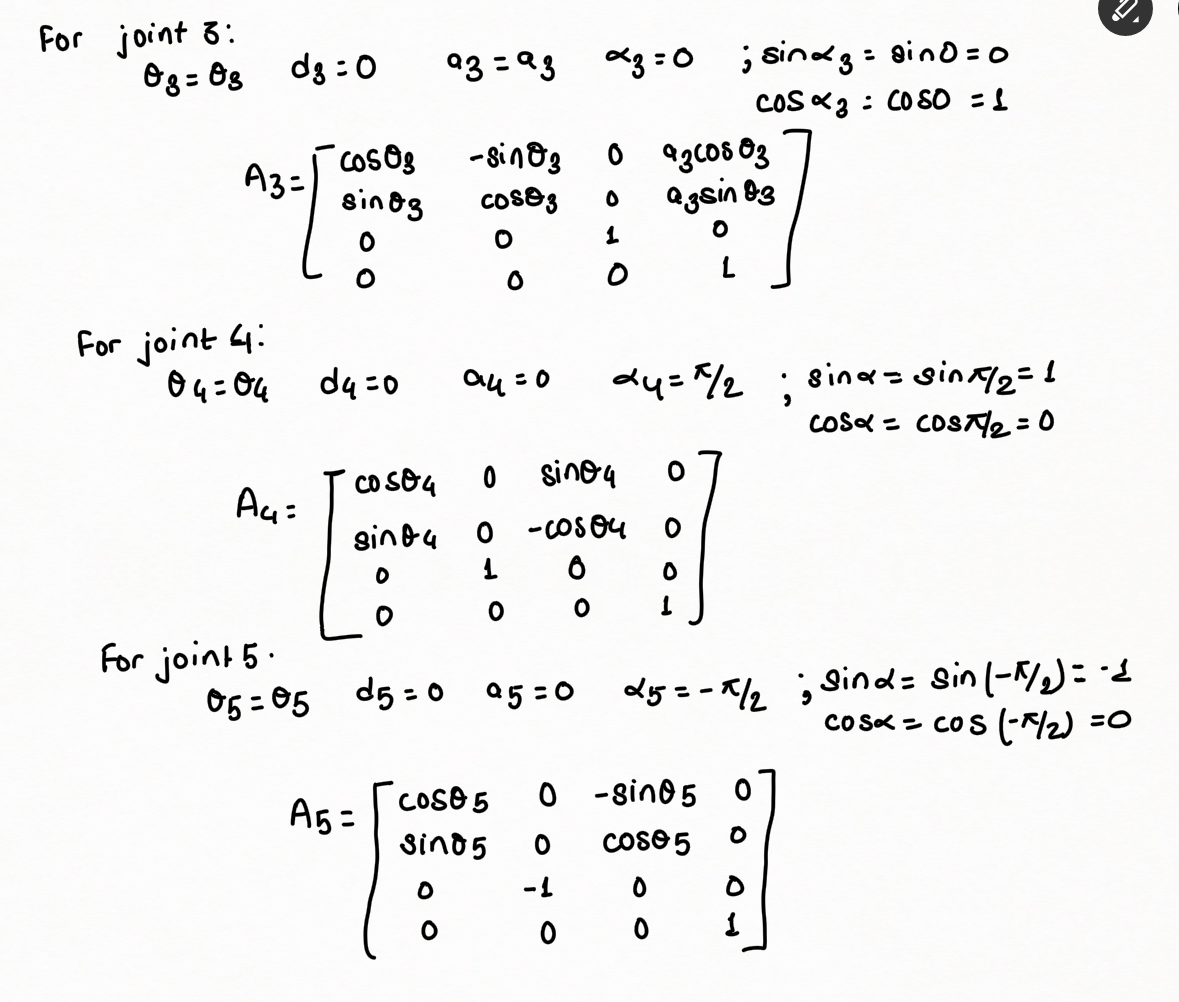
As per the frames attached, the DH convention can be obtained as:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. **Write down corresponding transformation matrices.**

The corresponding transformation matrices are given as:





A group of math equations

AI-generated content may be incorrect.

* 1. **Write down the matrix T of transformations (end effector relative to base)  
     A white paper with black text

     AI-generated content may be incorrect.**
  2. **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. **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:

A black numbers and symbols on a white background

AI-generated content may be incorrect.

* 1. **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:

A black text with black text

AI-generated content may be incorrect.

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

* 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°

* 1. **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.')

**A white background with black numbers and symbols

AI-generated content may be incorrect.**

* 1. **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)**

* 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');

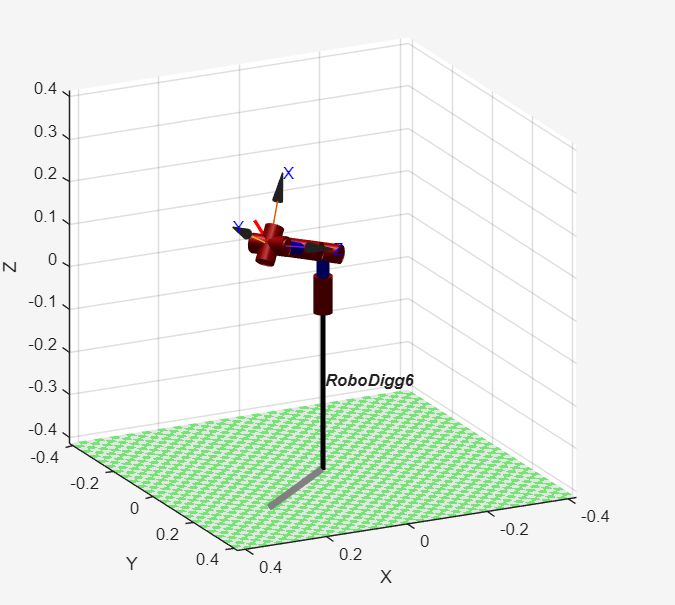
% Animation of a a small joint-space move

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});

* 1. **Display robot’s end effector position (take a screenshot from MATLAB)**



A screenshot of a computer

AI-generated content may be incorrect.

* 1. **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;

A graph of a graph with a graph and a diagram

AI-generated content may be incorrect.

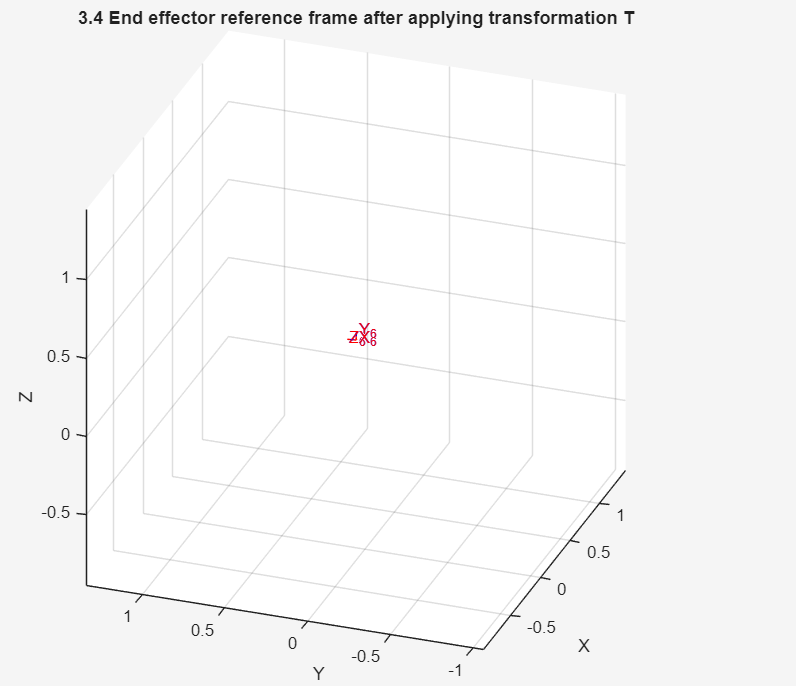
* 1. **Plot end effector’s reference frame after applying the transformation T using MATLAB.**

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');

****

* 1. **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

A screenshot of a computer

AI-generated content may be incorrect.

*\*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)**

* 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

**A diagram of a robot

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* 1. Display robot’s end effector position (take a screenshot from MATLAB)

A diagram of a robot

AI-generated content may be incorrect.

A diagram of a robot

AI-generated content may be incorrect.

* 1. **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);

**A computer generated image of a graph

AI-generated content may be incorrect.**

* 1. **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;

A graph of a graph

AI-generated content may be incorrect.

* 1. **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

A diagram of a graph

AI-generated content may be incorrect.  
  
*\*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"));

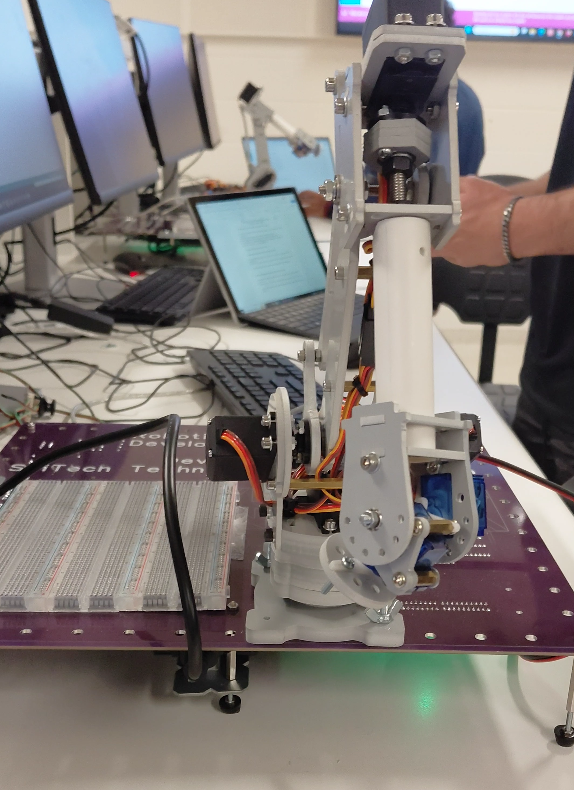
}

}

}

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*