A screenshot of a computer

Description automatically generatedA logo of a company

Description automatically generated

Robotic-Arm Mathematical Model Calculations

By:

Amr Osama Abd-Allah

A computer screen shot of a computer screen

Description automatically generatedCalculations:

L1 = 54.00 mm

L2 = 109.990 mm

L3 = 69.607 mm

L4 = 116.826 mm

A screenshot of a computer

Description automatically generatedfunction [theta1, theta2, theta3, theta4] = inverse\_kinematics(x, y, z, phi)

% Link lengths in mm

L1 = 54.00;

L2 = 109.990;

L3 = 69.607;

A screenshot of a computer

Description automatically generated L4 = 116.826;

% Calculate theta1

theta1 = atan2(y, x);

% Calculate the wrist center position

wx = x - L4 \* cos(phi) \* cos(theta1);

wy = y - L4 \* cos(phi) \* sin(theta1);

wz = z - L4 \* sin(phi);

% Calculate the distance from the base to the wrist center

r = sqrt(wx^2 + wy^2);

s = wz - L1;

% Calculate theta2 and theta3 using the law of cosines

D = (r^2 + s^2 - L2^2 - L3^2) / (2 \* L2 \* L3);

% Clamp D to the range [-1, 1] to avoid errors due to numerical issues

D = min(1, max(-1, D));

theta3 = atan2(sqrt(1 - D^2), D);

% Intermediate angle for theta2 calculation

beta = atan2(L3 \* sin(theta3), L2 + L3 \* cos(theta3));

theta2 = atan2(s, r) - beta;

% Calculate theta4 based on the desired orientation phi

theta4 = phi - theta2 - theta3;

end

#Then call Function and Apply it

% Define the link lengths

a1 = 54.00; % in mm

a2 = 109.990; % in mm

a3 = 69.607; % in mm

a4 = 116.826; % in mm

% Define symbolic joint angles

theta = sym('theta', [1 4]); % symbolic joint angles

a = [a1, a2, a3, a4]; % link lengths

% Construct transformation matrices using DH parameters

T = cell(1, 4);

for i = 1:4

T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)\*cos(theta(i));

sin(theta(i)), cos(theta(i)), 0, a(i)\*sin(theta(i));

0, 0, 1, 0;

0, 0, 0, 1];

end

% Define a function for the full transformation from base to end-effector

full\_transformation = @(th) double(subs(T{1}, theta(1), th(1)) \* ...

subs(T{2}, theta(2), th(2)) \* ...

subs(T{3}, theta(3), th(3)) \* ...

subs(T{4}, theta(4), th(4)));

% Predefined end-effector position (X, Y, Z) and orientation (phi in degrees)

positions = [

150, 100, 50, 20;

];

num\_movements = 1; % Number of movements

for move = 1:num\_movements

% Use predefined end-effector positions and solve for joint angles

target\_position = positions(move, 1:3);

phi = deg2rad(positions(move, 4)); % Convert to radians

[theta1, theta2, theta3, theta4] = inverse\_kinematics(target\_position(1), target\_position(2), target\_position(3), phi);

theta\_rad = [theta1, theta2, theta3, theta4];

theta\_deg = rad2deg(theta\_rad); % Convert to degrees

% Display results

disp('Computed Joint Angles (degrees):');

disp(theta\_deg);

% Plot the manipulator

T\_full = full\_transformation(theta\_rad);

position\_values = T\_full(1:3, 4);

orientation\_value = atan2d(T\_full(2, 1), T\_full(1, 1));

figure(1); clf; hold on;

plot3([0, T\_full(1, 4)], [0, T\_full(2, 4)], [0, T\_full(3, 4)], 'k-o', 'LineWidth', 2);

plot3(T\_full(1, 4), T\_full(2, 4), T\_full(3, 4), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');

title(sprintf('End-Effector Position: [%.2f, %.2f, %.2f] mm', target\_position));

xlabel('X Position (mm)');

ylabel('Y Position (mm)');

zlabel('Z Position (mm)');

grid on;

axis equal;

xlim([-400, 400]);

ylim([-400, 400]);

zlim([-400, 400]);

pause(1); % Pause for 1 second between movements

end

Out:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

First let’s Assume that we know the Angles

We can adjust angles at the code to find End Effector at the space with matrix using MATLAB Code:

% Define the link lengths

a1 = 54.00; % in mm

a2 = 109.990; % in mm

a3 = 69.607; % in mm

a4 = 116.826; % in mm

% Symbolic joint angles

theta = sym('theta', [1 4]); % symbolic joint angles

d = [0, 0, 0, 0]; % for planar manipulator

a = [a1, a2, a3, a4]; % link lengths

alpha = [0, 0, 0, 0]; % for planar manipulator

% Construct transformation matrices using DH parameters

T = cell(1, 4);

for i = 1:4

T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)\*cos(theta(i));

sin(theta(i)), cos(theta(i)), 0, a(i)\*sin(theta(i));

0, 0, 1, d(i);

0, 0, 0, 1];

end

% Multiply the matrices to get the transformation from base to end-effector

T\_0\_4 = T{1} \* T{2} \* T{3} \* T{4};

% Extract end-effector position and orientation

end\_effector\_position = T\_0\_4(1:3, 4);

end\_effector\_orientation = atan2(T\_0\_4(2, 1), T\_0\_4(1, 1));

% Substitute specific values for theta1, theta2, theta3, and theta4

theta\_values = [deg2rad(90), deg2rad(0), deg2rad(0), deg2rad(0)]; % example values in radians

position\_values = double(subs(end\_effector\_position, theta, theta\_values));

orientation\_value = double(subs(end\_effector\_orientation, theta, theta\_values));

disp('End-effector Position (numeric):');

disp(position\_values);

disp('End-effector Orientation (phi, numeric):');

disp(rad2deg(orientation\_value)); % convert to degrees

A screenshot of a computer

Description automatically generatedSo at the last code we adjust theta1=90 and the rest of angles = 0 that means that the movement at Y axis and both of X,Z will be Zero as you see the output 🡪

First let’s Assume that we Don’t know the Angles

Code:

% Define the link lengths

a1 = 54.00; % in mm

a2 = 109.990; % in mm

a3 = 69.607; % in mm

a4 = 116.826; % in mm

% Symbolic joint angles

theta = sym('theta', [1 4]); % symbolic joint angles

d = [0, 0, 0, 0]; % for planar manipulator

a = [a1, a2, a3, a4]; % link lengths

alpha = [0, 0, 0, 0]; % for planar manipulator

% Construct transformation matrices using DH parameters

T = cell(1, 4);

for i = 1:4

T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)\*cos(theta(i));

sin(theta(i)), cos(theta(i)), 0, a(i)\*sin(theta(i));

0, 0, 1, d(i);

0, 0, 0, 1];

end

% Multiply the matrices to get the transformation from base to end-effector

T\_0\_4 = T{1} \* T{2} \* T{3} \* T{4};

% Extract end-effector position and orientation

end\_effector\_position = T\_0\_4(1:3, 4);

end\_effector\_orientation = atan2(T\_0\_4(2, 1), T\_0\_4(1, 1));

% Define a range of angles for simulation (0 to 180 degrees -> 0 to pi radians)

theta1\_range = linspace(0, pi, 5); % range for theta1

theta2\_range = linspace(0, pi, 5); % range for theta2

theta3\_range = linspace(0, pi, 5); % range for theta3

theta4\_range = linspace(0, pi, 5); % range for theta4

% Initialize a matrix to store end-effector positions

positions = [];

% Iterate through the angle ranges and compute positions

for th1 = theta1\_range

for th2 = theta2\_range

for th3 = theta3\_range

for th4 = theta4\_range

theta\_values\_rad = [th1, th2, th3, th4];

theta\_values\_deg = rad2deg(theta\_values\_rad); % Convert to degrees

position\_values = double(subs(end\_effector\_position, theta, theta\_values\_rad));

orientation\_value = double(subs(end\_effector\_orientation, theta, theta\_values\_rad));

% Store the positions and orientations

positions = [positions; theta\_values\_deg, position\_values', rad2deg(orientation\_value)];

end

end

end

end

% Display the results

disp('Theta values (degrees) and corresponding End-effector Position and Orientation:');

disp('Theta1 Theta2 Theta3 Theta4 X(mm) Y(mm) Z(mm) Orientation(degrees)');

disp(positions);

% Plot the end-effector positions

figure;

scatter(positions(:, 5), positions(:, 6), 'filled');

title('End-Effector Positions (0 to 180 Degrees)');

xlabel('X Position (mm)');

ylabel('Y Position (mm)');

grid on;

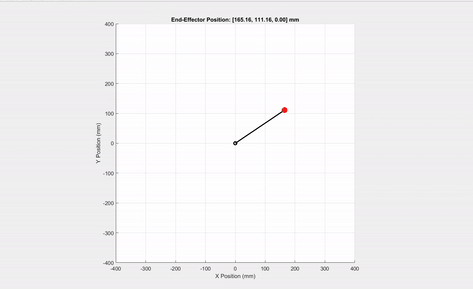
The code here depends on the Lengths we have adjust of Our robot and apply DH matrix to calculate the Theta and X,Y,Z positions so we can determine the End Effector positions at space, but will all possibilities

Of Theta and the axis

A screen shot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedHere’s the out example:

A screenshot of a computer

Description automatically generated

User can choose How Many Movement & Simulate it

Code:

% Define the link lengths

a1 = 54.00; % in mm

a2 = 109.990; % in mm

a3 = 69.607; % in mm

a4 = 116.826; % in mm

% Symbolic joint angles

theta = sym('theta', [1 4]); % symbolic joint angles

d = [0, 0, 0, 0]; % for planar manipulator

a = [a1, a2, a3, a4]; % link lengths

alpha = [0, 0, 0, 0]; % for planar manipulator

% Construct transformation matrices using DH parameters

T = cell(1, 4);

for i = 1:4

T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)\*cos(theta(i));

sin(theta(i)), cos(theta(i)), 0, a(i)\*sin(theta(i));

0, 0, 1, d(i);

0, 0, 0, 1];

end

% Define a function for the full transformation from base to end-effector

full\_transformation = @(th) double(subs(T{1}, theta(1), th(1)) \* ...

subs(T{2}, theta(2), th(2)) \* ...

subs(T{3}, theta(3), th(3)) \* ...

subs(T{4}, theta(4), th(4)));

% Get the number of movements to simulate from the user

num\_movements = input('Enter the number of movements to simulate: ');

% Define a range of angles for simulation (0 to 180 degrees -> 0 to pi radians)

theta1\_range = linspace(0, pi, 5); % range for theta1

theta2\_range = linspace(0, pi, 5); % range for theta2

theta3\_range = linspace(0, pi, 5); % range for theta3

theta4\_range = linspace(0, pi, 5); % range for theta4

% Initialize a matrix to store end-effector positions and orientations

positions = [];

% Counter for the number of simulated movements

count = 0;

% Iterate through the angle ranges and compute positions

for th1 = theta1\_range

for th2 = theta2\_range

for th3 = theta3\_range

for th4 = theta4\_range

if count >= num\_movements

break; % Exit loop if the desired number of movements is reached

end

theta\_values\_rad = [th1, th2, th3, th4];

theta\_values\_deg = rad2deg(theta\_values\_rad); % Convert to degrees

% Compute the full transformation matrix for the current angles

T\_full = full\_transformation(theta\_values\_rad);

position\_values = T\_full(1:3, 4);

orientation\_value = atan2d(T\_full(2, 1), T\_full(1, 1));

% Store the positions and orientations

positions = [positions; theta\_values\_deg, position\_values', orientation\_value];

% Plot the manipulator in 3D space

figure(1); clf; hold on;

plot3([0, T\_full(1, 4)], [0, T\_full(2, 4)], [0, T\_full(3, 4)], 'k-o', 'LineWidth', 2);

plot3(T\_full(1, 4), T\_full(2, 4), T\_full(3, 4), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');

title(sprintf('End-Effector Position: [%.2f, %.2f, %.2f] mm', position\_values));

xlabel('X Position (mm)');

ylabel('Y Position (mm)');

zlabel('Z Position (mm)');

grid on;

axis equal;

xlim([-400, 400]);

ylim([-400, 400]);

zlim([-400, 400]);

drawnow;

count = count + 1; % Increment the counter

pause(1); % Pause for 1 second between movements

end

if count >= num\_movements

break; % Exit loop if the desired number of movements is reached

end

end

if count >= num\_movements

break; % Exit loop if the desired number of movements is reached

end

end

if count >= num\_movements

break; % Exit loop if the desired number of movements is reached

end

end

% Display the results

disp('Theta values (degrees) and corresponding End-effector Position and Orientation:');

disp('Theta1 Theta2 Theta3 Theta4 X(mm) Y(mm) Z(mm) Orientation (degrees)');

disp(positions);

% Plot the end-effector positions in 3D

figure;

scatter3(positions(:, 5), positions(:, 6), positions(:, 8), 'filled');

title('3D Visualization of End-Effector Positions and Orientations');

xlabel('X Position (mm)');

ylabel('Y Position (mm)');

zlabel('Orientation (degrees)');

grid on;

EX: As we see 10 movements Simulation with Theta and X,Y,Z for End Effector

A screenshot of a computer

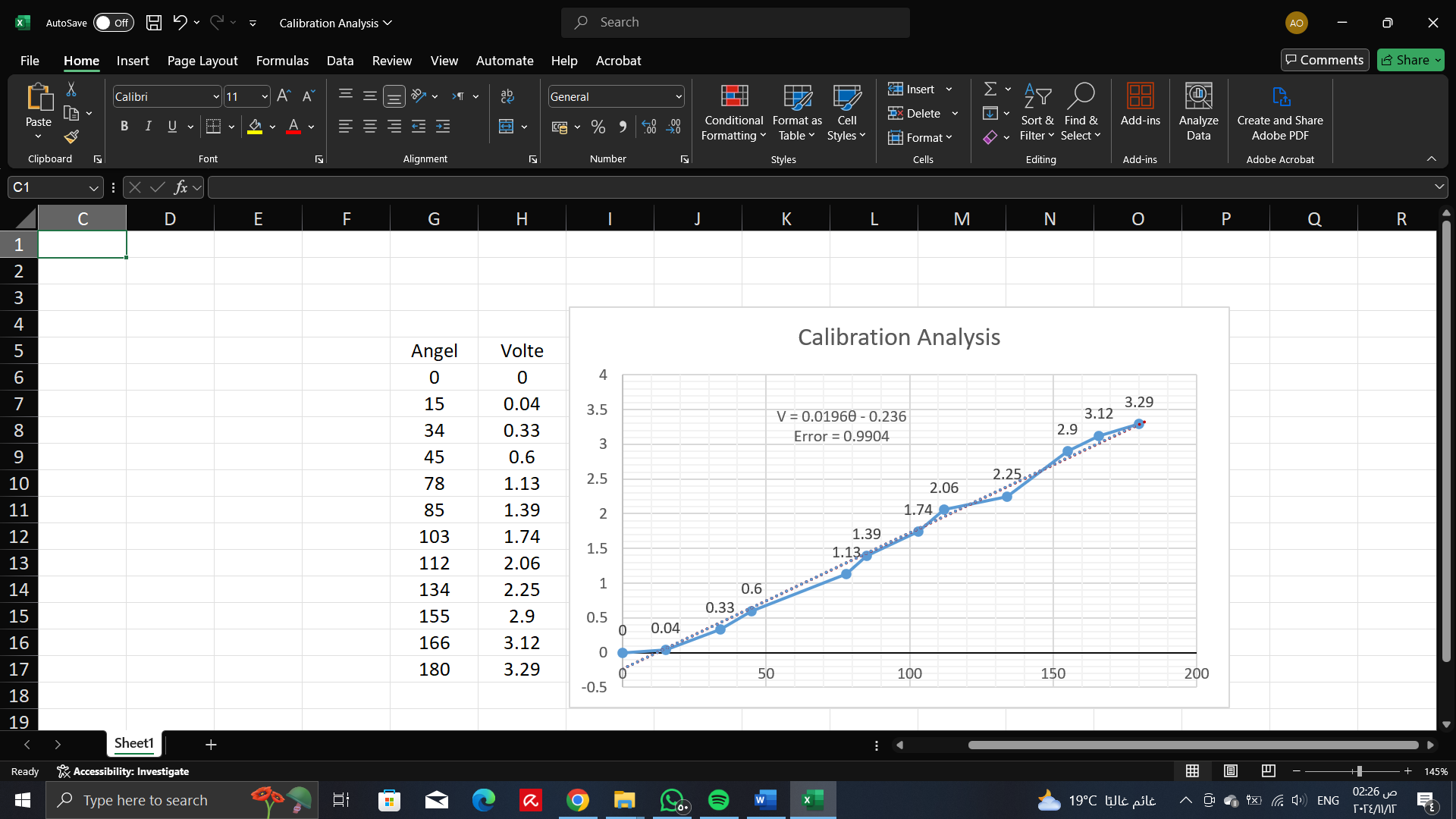
Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Calibration Analysis



So, we get a relationship between voltage and angel as you see in case we adjust a volte then we can calculate the angle of its movement.

* Error: 0.9904