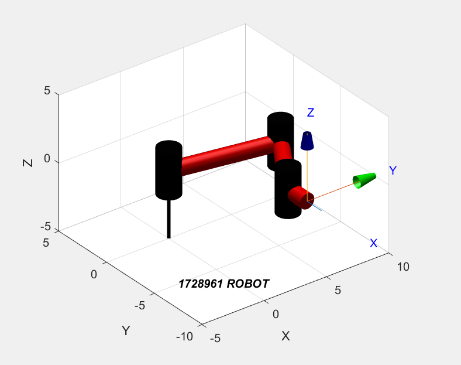


ROBOTICS PROJECT

MCTE 4352



DONE BY:

IBRAHIM MAHMUD SUHAIMI

1728961

WITH SPECIAL ACKNOWLEDGEMENTS TO:

TAHSIN FUAD HASAN

1716197

**Introduction**

Repetition. Producing a product, cleaning a room, performing calculations, etc. are all repetitive tasks that consume a lot of time. Robots are devices/machines that are specifically designed to resolve this issue. A set of instructions are programmed into them to perform the task, but unlike people robots do not get tired and are not capable of performing tasks of greater thought.

Cyclical tasks often require a definite set of movements, so robots usually have a fixed number of joints with a limited range of motion. Therefore, most robots are designed to perform a specific task. Highly intelligent robots (humanoids) are coming into the market; however, they are expensive and are often less efficient to their human counterparts.

**Objectives and conditions**

Project objectives:

The project requires the calculations to be done manually using pen & paper and Microsoft Excel and separately using MATLAB.

* The result of the two methods must be compared.
* They should produce a very similar result.

There is one strict condition in the design of the robot.

* The robot must be a ***3DOF planar robot***.
  + Three joints (Chosen, NOT a requirement)
    - 3 DOF is NOT equal to 3 joints.
  + All of them revolute (can only rotate)
  + The joints cannot extend or subtend (NO prismatic joints)
  + Three links connecting the joints

Note: Because the robot/manipulator is planar and has a fixed base with the other end being free, its 3DOF directly translates to 3 revolute joints.

Robot objectives:

The sole objective of the robot is to move in such a manner that it traces out ‘SUHAIMI’ (my last name) in its workspace.

* The size of the workspace must be no larger than an A4 paper.

**Trajectory planning**

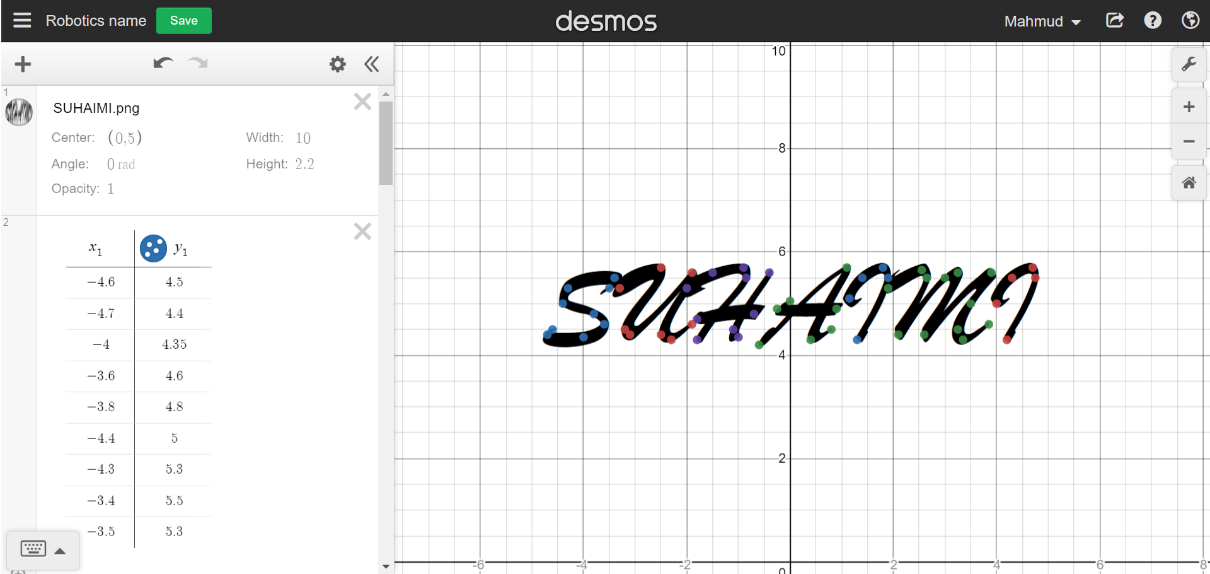
The path is a complicated one with a lot of curves that cannot be defined by functions. The solution was to define the path using several points, the shape would then be created by joining these points two at a time.

[Desmos](https://www.desmos.com/) graphing calculator was used to place the points.

Procedure:

1. Wrote SUHAIMI in Brush Script MT on Microsoft paint with a font size of 95. Font was black with a white background.
2. The PNG file was then uploaded to Desmos.
3. The centre of the image was shifted to (0,5) of the Desmos graph.
4. Tables were created and points were placed with distinct colours for each of the letters. (Figure 1)
5. These points were copied to a Microsoft Excel sheet.
6. Forward kinematics calculations were done on paper.
7. The translation components were extracted from the homogenous transformation matrix relating the end-effector to the base.
8. Inverse kinematics calculations were conducted to find the joint angles in terms of known values.
9. The path was sketched using Microsoft Excel. This was done by choosing the X-Y scatter plot with linear interpolation between the points. (Figure 2)

The forward and inverse kinematic calculations done manually are shown at the end of the report.



**Figure 1** Point placement for shape definition in Desmos graphing calculator

**Figure 2** The points shown using an MS Excel chart. It shows the entire trajectory of the robot; including the lines that will not be plotted by the robot.

**Design of the robot**

The robot was designed by using a trial-and-error method.

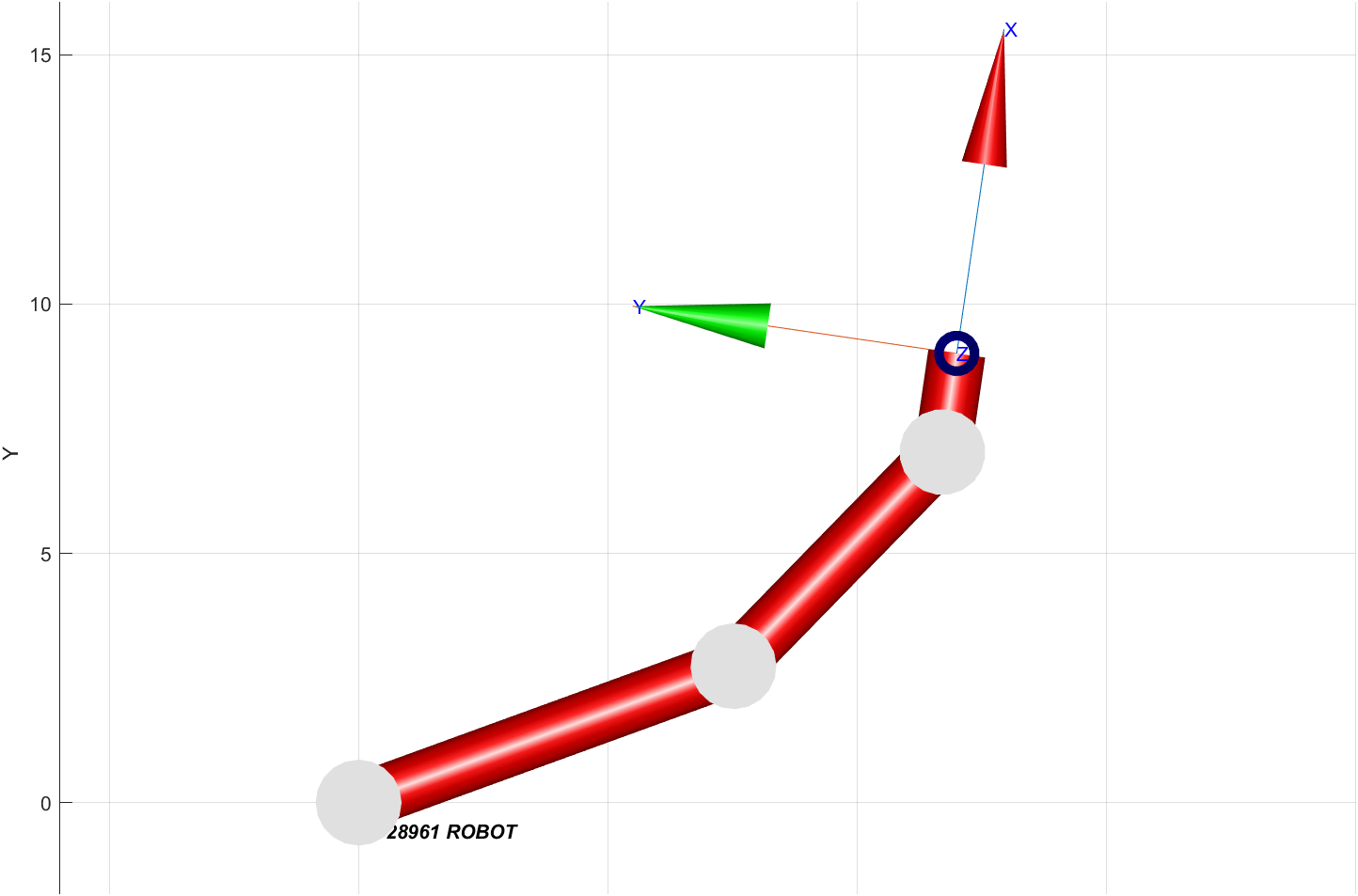
Procedure:

1. A set of link lengths (a1, a2, a3) and the orientation of the end-effector (Φ) was assigned values.
2. *The cartesian co-ordinates of the path (name) were jotted down*
3. *Then inverse kinematics calculations were performed to find the corresponding joint angles.*
4. If the joint angles of two consecutive co-ordinates were far apart; one positive and one negative, then it could be said that the trajectory would not be accurate.
5. If step 4 was true, then step 1 would be repeated and then changes would be observed.

* *The cursive steps were done before hand in the trajectory planning.*

Through this trial-and-error process, the final design of the robot was made to be:

* a1 = 8
* a2 = 6
* a3 = 2
* Φ = 1.5π



**Figure 3** Top view of the robot on MATLAB.

The orientation of the final link is not accurate in figure 3. The figure is just to demonstrate the link lengths.

The tables below (Figure 4) show the final results.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | phi |  |  |  |  |
| 8 | 6 | 2 | 4.712389 |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Letter | X3 | Y3 | X2 | Y2 | theta1\_rad | theta2\_rad | theta3\_rad |
| S | -4.60 | 4.50 | -4.60 | 6.50 | -0.1843 | -1.9618 | 6.8585 |
| -4.70 | 4.40 | -4.70 | 6.40 | -0.1656 | -1.9659 | 6.8439 |
| -4.00 | 4.35 | -4.00 | 6.35 | -0.2164 | -2.0431 | 6.9719 |
| -3.60 | 4.60 | -3.60 | 6.60 | -0.2798 | -2.0408 | 7.0330 |
| -3.80 | 4.80 | -3.80 | 6.80 | -0.2821 | -1.9928 | 6.9872 |
| -4.40 | 5.00 | -4.40 | 7.00 | -0.2548 | -1.9067 | 6.8739 |
| -4.30 | 5.30 | -4.30 | 7.30 | -0.2949 | -1.8692 | 6.8765 |
| -3.40 | 5.50 | -3.40 | 7.50 | -0.3886 | -1.9127 | 7.0137 |
| -3.70 | 5.20 | -3.70 | 7.20 | -0.3322 | -1.9381 | 6.9826 |
| U | -3.30 | 5.30 | -3.30 | 7.30 | -0.3780 | -1.9532 | 7.0436 |
| -2.50 | 5.70 | -2.50 | 7.70 | -0.4930 | -1.9379 | 7.1433 |
| -3.20 | 4.50 | -3.20 | 6.50 | -0.3101 | -2.0885 | 7.1110 |
| -3.10 | 4.40 | -3.10 | 6.40 | -0.3112 | -2.1117 | 7.1352 |
| -1.90 | 5.60 | -1.90 | 7.60 | -0.5488 | -1.9849 | 7.2461 |
| -2.50 | 4.40 | -2.50 | 6.40 | -0.3810 | -2.1530 | 7.2464 |
| -2.30 | 4.30 | -2.30 | 6.30 | -0.3979 | -2.1811 | 7.2914 |
| -1.90 | 4.60 | -1.90 | 6.60 | -0.4730 | -2.1535 | 7.3389 |
| H | -2.00 | 5.30 | -2.00 | 7.30 | -0.5140 | -2.0319 | 7.2582 |
| -1.50 | 5.60 | -1.50 | 7.60 | -0.5947 | -2.0005 | 7.3076 |
| -0.90 | 5.70 | -0.90 | 7.70 | -0.6735 | -1.9994 | 7.3853 |
| -0.85 | 5.50 | -0.85 | 7.50 | -0.6676 | -2.0356 | 7.4155 |
| -1.80 | 4.30 | -1.80 | 6.30 | -0.4648 | -2.2074 | 7.3846 |
| -1.80 | 4.70 | -1.80 | 6.70 | -0.4933 | -2.1416 | 7.3473 |
| -0.70 | 4.80 | -0.70 | 6.80 | -0.6496 | -2.1590 | 7.5210 |
| -1.00 | 4.35 | -1.00 | 6.35 | -0.5834 | -2.2284 | 7.5242 |
| -1.10 | 4.50 | -1.10 | 6.50 | -0.5767 | -2.2006 | 7.4896 |
| -0.40 | 5.60 | -0.40 | 7.60 | -0.7307 | -2.0245 | 7.4676 |
| A | -0.60 | 4.20 | -0.60 | 6.20 | -0.6381 | -2.2620 | 7.6125 |
| 1.10 | 5.70 | 1.10 | 7.70 | -0.9330 | -1.9948 | 7.6402 |
| 0.40 | 4.30 | 0.40 | 6.30 | -0.8000 | -2.2479 | 7.7603 |
| 0.80 | 4.50 | 0.80 | 6.50 | -0.8655 | -2.2079 | 7.7858 |
| 0.00 | 5.05 | 0.00 | 7.05 | -0.7599 | -2.1223 | 7.5946 |
| -0.25 | 4.90 | -0.25 | 6.90 | -0.7184 | -2.1473 | 7.5780 |
| 0.90 | 4.90 | 0.90 | 6.90 | -0.8862 | -2.1380 | 7.7366 |
| I | 1.15 | 5.10 | 1.15 | 7.10 | -0.9260 | -2.0976 | 7.7360 |
| 1.40 | 5.50 | 1.40 | 7.50 | -0.9687 | -2.0212 | 7.7023 |
| 1.80 | 5.70 | 1.80 | 7.70 | -1.0270 | -1.9718 | 7.7112 |
| 1.90 | 5.50 | 1.90 | 7.50 | -1.0373 | -2.0022 | 7.7518 |
| 1.30 | 4.30 | 1.30 | 6.30 | -0.9432 | -2.2276 | 7.8833 |
| M | 1.90 | 5.30 | 1.90 | 7.30 | -1.0349 | -2.0364 | 7.7836 |
| 2.55 | 5.65 | 2.55 | 7.65 | -1.1270 | -1.9437 | 7.7831 |
| 2.65 | 5.50 | 2.65 | 7.50 | -1.1394 | -1.9634 | 7.8151 |
| 2.10 | 4.40 | 2.10 | 6.40 | -1.0659 | -2.1762 | 7.9544 |
| 3.00 | 5.50 | 3.00 | 7.50 | -1.1865 | -1.9412 | 7.8401 |
| 3.25 | 5.60 | 3.25 | 7.60 | -1.2200 | -1.9071 | 7.8394 |
| 2.60 | 4.40 | 2.60 | 6.40 | -1.1406 | -2.1467 | 7.9997 |
| 3.90 | 5.60 | 3.90 | 7.60 | -1.3053 | -1.8562 | 7.8739 |
| 3.50 | 5.00 | 3.50 | 7.00 | -1.2571 | -1.9863 | 7.9557 |
| 3.25 | 4.50 | 3.25 | 6.50 | -1.2321 | -2.0846 | 8.0291 |
| 3.35 | 4.30 | 3.35 | 6.30 | -1.2519 | -2.1075 | 8.0718 |
| 3.85 | 4.60 | 3.85 | 6.60 | -1.3127 | -2.0192 | 8.0443 |
| I | 4.00 | 5.00 | 4.00 | 7.00 | -1.3243 | -1.9440 | 7.9807 |
| 4.30 | 5.50 | 4.30 | 7.50 | -1.3577 | -1.8371 | 7.9072 |
| 4.70 | 5.70 | 4.70 | 7.70 | -1.4080 | -1.7660 | 7.8864 |
| 4.75 | 5.50 | 4.75 | 7.50 | -1.4156 | -1.7933 | 7.9213 |
| 4.20 | 4.30 | 4.20 | 6.30 | -1.3695 | -2.0314 | 8.1133 |

**Figure 4** The worksheet with all the calculated values. The grey shadings indicate jumps where the robot should not sketch.

**Simulation and results**

The results found using MS Excel, shown in Figure 2, are plotted using the cartesian co-ordinates found from the Desmos graphing calculator. It is not a result of any formula or equation. It is basically the raw data resketched to check for accuracy.

The graph shown in Figure 5; however, is a result of the inverse kinematic calculations done in the MS Excel sheet. The joint angles found were used in the code that produced the drawing.

Code:

Note: Code copied from here and pasted onto MATLAB may not work due to the format changes caused by MS Word. The file is provided separately (*Robotics\_project.m*).

%% Robotics project

clear;

clc;

% Specifying the link lengths of the robot

% 3 DOF with three revolute joints

L1 = Link('revolute', 'd', 0, 'a', 8, 'alpha', 0);

L2 = Link('revolute', 'd', 0, 'a', 6, 'alpha', 0);

L3 = Link('revolute', 'd', 0, 'a', 2, 'alpha', 0);

% Creating the robot model

IMS = SerialLink([L1 L2 L3]);

IMS.name = '1728961 ROBOT';

% Time interval and step size

% 5 steps with total time of 1s

t=0:0.2:1;

%% S

% The three values in the array are the three joint angles

% i.e. Sn=[theta1 theta2 theta3]; where n is an integer

S0=[-0.1843 -1.9618 6.8585];

S1=[-0.1656 -1.9659 6.8439];

S2=[-0.2164 -2.0431 6.9719];

S3=[-0.2798 -2.0408 7.0330];

S4=[-0.2821 -1.9928 6.9872];

S5=[-0.2548 -1.9067 6.8739];

S6=[-0.2949 -1.8692 6.8765];

S7=[-0.3886 -1.9127 7.0137];

S8=[-0.3322 -1.9381 6.9826];

% Trajectory planning for the letter S

Straj1 = jtraj(S0,S1,t);

Straj2 = jtraj(S1,S2,t);

Straj3 = jtraj(S2,S3,t);

Straj4 = jtraj(S3,S4,t);

Straj5 = jtraj(S4,S5,t);

Straj6 = jtraj(S5,S6,t);

Straj7 = jtraj(S6,S7,t);

Straj8 = jtraj(S7,S8,t);

hold on

% Initialising the variable

Ibra=zeros(4,4);

% Specifying the workspace

xlim([-5,5])

ylim([-10,-8])

zlim([-1,5])

% The view is set to [180,90] to straighten the sketch

% No absolute starting position was chosen

% All movements are relative to the robot's base and understanding

% So the sketch will be done based on the robot's workspace and orientation

% Move from origin to starting point

IMS.plot(jtraj([0 0 0],S0,t),'scale',0.6,'linkcolor','r','jointcolor','k','view',[180,90])

% 1st segment

for i=1:1:length(t)

Ibra=IMS.fkine(Straj1(i,:));

S\_letter(i,:)=transl(Ibra);

% Extract the translation component of the pose

% matrix (3Element column vector) stored in JTA

% vector array

jta=S\_letter;

% Draw track points

plot2(jta(i,:),'r.')

% Trajectory animation

IMS.plot(Straj1(i,:))

% Draw a trajectory line

plot2(S\_letter,'b','linewidth',2)

end

% 2nd segment

for i=1:1:length(t)

Ibra2=IMS.fkine(Straj2(i,:));

S\_letter2(i,:)=transl(Ibra2);

jta2=S\_letter2;

plot2(jta2(i,:),'r.')

IMS.plot(Straj2(i,:))

plot2(S\_letter2,'b','linewidth',2)

end

% 3rd segment

for i=1:1:length(t)

Ibra3=IMS.fkine(Straj3(i,:));

S\_letter3(i,:)=transl(Ibra3);

jta3=S\_letter3;

plot2(jta3(i,:),'r.')

IMS.plot(Straj3(i,:))

plot2(S\_letter3,'b','linewidth',2)

end

% 4th segment

for i=1:1:length(t)

Ibra4=IMS.fkine(Straj4(i,:));

S\_letter4(i,:)=transl(Ibra4);

jta4=S\_letter4;

plot2(jta4(i,:),'r.')

IMS.plot(Straj4(i,:))

plot2(S\_letter4,'b','linewidth',2)

end

% 5th segment

for i=1:1:length(t)

Ibra5=IMS.fkine(Straj5(i,:));

S\_letter5(i,:)=transl(Ibra5);

jta5=S\_letter5;

plot2(jta5(i,:),'r.')

IMS.plot(Straj5(i,:))

plot2(S\_letter5,'b','linewidth',2)

end

% 6th segment

for i=1:1:length(t)

Ibra6=IMS.fkine(Straj6(i,:));

S\_letter6(i,:)=transl(Ibra6);

jta6=S\_letter6;

plot2(jta6(i,:),'r.')

IMS.plot(Straj6(i,:))

plot2(S\_letter6,'b','linewidth',2)

end

% 7th segment

for i=1:1:length(t)

Ibra7=IMS.fkine(Straj7(i,:));

S\_letter7(i,:)=transl(Ibra7);

jta7=S\_letter7;

plot2(jta7(i,:),'r.')

IMS.plot(Straj7(i,:))

plot2(S\_letter7,'b','linewidth',2)

end

% 8th segment

for i=1:1:length(t)

Ibra8=IMS.fkine(Straj8(i,:));

S\_letter8(i,:)=transl(Ibra8);

jta8=S\_letter8;

plot2(jta8(i,:),'r.')

IMS.plot(Straj8(i,:))

plot2(S\_letter8,'b','linewidth',2)

end

%% U

U0=[-0.3780 -1.9532 7.0436];

U1=[-0.4930 -1.9379 7.1433];

U2=[-0.3101 -2.0885 7.1110];

U3=[-0.3112 -2.1117 7.1352];

U4=[-0.5488 -1.9849 7.2461];

U5=[-0.3810 -2.1530 7.2464];

U6=[-0.3979 -2.1811 7.2914];

U7=[-0.4730 -2.1535 7.3389];

% Trajectory planning for the letter U

Utraj1 = jtraj(U0,U1,t);

Utraj2 = jtraj(U1,U2,t);

Utraj3 = jtraj(U2,U3,t);

Utraj4 = jtraj(U3,U4,t);

Utraj5 = jtraj(U4,U5,t);

Utraj6 = jtraj(U5,U6,t);

Utraj7 = jtraj(U6,U7,t);

hold on

% 1st segment

for i=1:1:length(t)

Ibra=IMS.fkine(Utraj1(i,:));

U\_letter(i,:)=transl(Ibra); %#ok<\*SAGROW>

jta=U\_letter;

plot2(jta(i,:),'y.')

IMS.plot(Utraj1(i,:))

plot2(U\_letter,'g','linewidth',2)

end

% 2nd segment

for i=1:1:length(t)

Ibra2=IMS.fkine(Utraj2(i,:));

U\_letter2(i,:)=transl(Ibra2);

jta2=U\_letter2;

plot2(jta2(i,:),'y.')

IMS.plot(Utraj2(i,:))

plot2(U\_letter2,'g','linewidth',2)

end

% 3rd segment

for i=1:1:length(t)

Ibra3=IMS.fkine(Utraj3(i,:));

U\_letter3(i,:)=transl(Ibra3);

jta3=U\_letter3;

plot2(jta3(i,:),'y.')

IMS.plot(Utraj3(i,:))

plot2(U\_letter3,'g','linewidth',2)

end

% 4th segment

for i=1:1:length(t)

Ibra4=IMS.fkine(Utraj4(i,:));

U\_letter4(i,:)=transl(Ibra4);

jta4=U\_letter4;

plot2(jta4(i,:),'y.')

IMS.plot(Utraj4(i,:))

plot2(U\_letter4,'g','linewidth',2)

end

% 5th segment

for i=1:1:length(t)

Ibra5=IMS.fkine(Utraj5(i,:));

U\_letter5(i,:)=transl(Ibra5);

jta5=U\_letter5;

plot2(jta5(i,:),'y.')

IMS.plot(Utraj5(i,:))

plot2(U\_letter5,'g','linewidth',2)

end

% 6th segment

for i=1:1:length(t)

Ibra6=IMS.fkine(Utraj6(i,:));

U\_letter6(i,:)=transl(Ibra6);

jta6=U\_letter6;

plot2(jta6(i,:),'y.')

IMS.plot(Utraj6(i,:))

plot2(U\_letter6,'g','linewidth',2)

end

% 7th segment

for i=1:1:length(t)

Ibra7=IMS.fkine(Utraj7(i,:));

U\_letter7(i,:)=transl(Ibra7);

jta7=U\_letter7;

plot2(jta7(i,:),'y.')

IMS.plot(Utraj7(i,:))

plot2(U\_letter7,'g','linewidth',2)

end

%% H

H0=[-0.5140 -2.0319 7.2582];

H1=[-0.5947 -2.0005 7.3076];

H2=[-0.6735 -1.9994 7.3853];

H3=[-0.6676 -2.0356 7.4155];

H4=[-0.4648 -2.2074 7.3846];

H5=[-0.4933 -2.1416 7.3473];

H6=[-0.6496 -2.1590 7.5210];

H7=[-0.5834 -2.2284 7.5242];

H8=[-0.5767 -2.2006 7.4896];

H9=[-0.7307 -2.0245 7.4676];

% Trajectory planning for the letter H

Htraj1 = jtraj(H0,H1,t);

Htraj2 = jtraj(H1,H2,t);

Htraj3 = jtraj(H2,H3,t);

Htraj4 = jtraj(H3,H4,t);

% Htraj5 = jtraj(H4,H5,t); % Commented to show it is skipped

Htraj6 = jtraj(H5,H6,t);

% Htraj7 = jtraj(H6,H7,t); % Commented to show it is skipped

Htraj8 = jtraj(H7,H8,t);

Htraj9 = jtraj(H8,H9,t);

hold on

% 1st segment

for i=1:1:length(t)

Ibra=IMS.fkine(Htraj1(i,:));

H\_letter(i,:)=transl(Ibra);

jta=H\_letter;

plot2(jta(i,:),'r.')

IMS.plot(Htraj1(i,:))

plot2(H\_letter,'b','linewidth',2)

end

% 2nd segment

for i=1:1:length(t)

Ibra2=IMS.fkine(Htraj2(i,:));

H\_letter2(i,:)=transl(Ibra2);

jta2=H\_letter2;

plot2(jta2(i,:),'r.')

IMS.plot(Htraj2(i,:))

plot2(H\_letter2,'b','linewidth',2)

end

% 3rd segment

for i=1:1:length(t)

Ibra3=IMS.fkine(Htraj3(i,:));

H\_letter3(i,:)=transl(Ibra3);

jta3=H\_letter3;

plot2(jta3(i,:),'r.')

IMS.plot(Htraj3(i,:))

plot2(H\_letter3,'b','linewidth',2)

end

% 4th segment

for i=1:1:length(t)

Ibra4=IMS.fkine(Htraj4(i,:));

H\_letter4(i,:)=transl(Ibra4);

jta4=H\_letter4;

plot2(jta4(i,:),'r.')

IMS.plot(Htraj4(i,:))

plot2(H\_letter4,'b','linewidth',2)

end

% 5th segment

for i=1:1:length(t)

Ibra6=IMS.fkine(Htraj6(i,:));

H\_letter6(i,:)=transl(Ibra6);

jta6=H\_letter6;

plot2(jta6(i,:),'r.')

IMS.plot(Htraj6(i,:))

plot2(H\_letter6,'b','linewidth',2)

end

% 6th segment

for i=1:1:length(t)

Ibra8=IMS.fkine(Htraj8(i,:));

H\_letter8(i,:)=transl(Ibra8);

jta8=H\_letter8;

plot2(jta8(i,:),'r.')

IMS.plot(Htraj8(i,:))

plot2(H\_letter8,'b','linewidth',2)

end

% 7th segment

for i=1:1:length(t)

Ibra9=IMS.fkine(Htraj9(i,:));

H\_letter9(i,:)=transl(Ibra9);

jta9=H\_letter9;

plot2(jta9(i,:),'r.')

IMS.plot(Htraj9(i,:))

plot2(H\_letter9,'b','linewidth',2)

end

%% A

A0=[-0.6381 -2.2620 7.6125];

A1=[-0.9330 -1.9948 7.6402];

A2=[-0.8000 -2.2479 7.7603];

A3=[-0.8655 -2.2079 7.7858];

A4=[-0.7599 -2.1223 7.5946];

A5=[-0.7184 -2.1473 7.5780];

A6=[-0.8862 -2.1380 7.7366];

% Trajectory planning for the letter A

Atraj1 = jtraj(A0,A1,t);

Atraj2 = jtraj(A1,A2,t);

Atraj3 = jtraj(A2,A3,t);

% Atraj4 = jtraj(A3,A4,t); Commented to show it is skipped

Atraj5 = jtraj(A4,A5,t);

Atraj6 = jtraj(A5,A6,t);

hold on

% 1st segment

for i=1:1:length(t)

Ibra=IMS.fkine(Atraj1(i,:));

A\_letter(i,:)=transl(Ibra);

jta=A\_letter;

plot2(jta(i,:),'y.')

IMS.plot(Atraj1(i,:))

plot2(A\_letter,'g','linewidth',2)

end

% 2nd segment

for i=1:1:length(t)

Ibra2=IMS.fkine(Atraj2(i,:));

A\_letter2(i,:)=transl(Ibra2);

jta2=A\_letter2;

plot2(jta2(i,:),'y.')

IMS.plot(Atraj2(i,:))

plot2(A\_letter2,'g','linewidth',2)

end

% 3rd segment

for i=1:1:length(t)

Ibra3=IMS.fkine(Atraj3(i,:));

A\_letter3(i,:)=transl(Ibra3);

jta3=A\_letter3;

plot2(jta3(i,:),'y.')

IMS.plot(Atraj3(i,:))

plot2(A\_letter3,'g','linewidth',2)

end

% 4th segment

for i=1:1:length(t)

Ibra5=IMS.fkine(Atraj5(i,:));

A\_letter5(i,:)=transl(Ibra5);

jta5=A\_letter5;

plot2(jta5(i,:),'y.')

IMS.plot(Atraj5(i,:))

plot2(A\_letter5,'g','linewidth',2)

end

% 5th segment

for i=1:1:length(t)

Ibra6=IMS.fkine(Atraj6(i,:));

A\_letter6(i,:)=transl(Ibra6);

jta6=A\_letter6;

plot2(jta6(i,:),'y.')

IMS.plot(Atraj6(i,:))

plot2(A\_letter6,'g','linewidth',2)

end

%% I

I0=[-0.9260 -2.0976 7.7360];

I1=[-0.9687 -2.0212 7.7023];

I2=[-1.0270 -1.9718 7.7112];

I3=[-1.0373 -2.0022 7.7518];

I4=[-0.9432 -2.2276 7.8833];

% Trajectory planning for the letter I

Itraj1 = jtraj(I0,I1,t);

Itraj2 = jtraj(I1,I2,t);

Itraj3 = jtraj(I2,I3,t);

Itraj4 = jtraj(I3,I4,t);

hold on

% 1st segment

for i=1:1:length(t)

Ibra=IMS.fkine(Itraj1(i,:));

I\_letter(i,:)=transl(Ibra);

jta=I\_letter;

% Draw track points

plot2(jta(i,:),'r.')

% Trajectory animation

IMS.plot(Itraj1(i,:))

% Draw a trajectory line

plot2(I\_letter,'b','linewidth',2)

end

% 2nd segment

for i=1:1:length(t)

Ibra2=IMS.fkine(Itraj2(i,:));

I\_letter2(i,:)=transl(Ibra2);

jta2=I\_letter2;

plot2(jta2(i,:),'r.')

IMS.plot(Itraj2(i,:))

plot2(I\_letter2,'b','linewidth',2)

end

% 3rd segment

for i=1:1:length(t)

Ibra3=IMS.fkine(Itraj3(i,:));

I\_letter3(i,:)=transl(Ibra3);

jta3=I\_letter3;

plot2(jta3(i,:),'r.')

IMS.plot(Itraj3(i,:))

plot2(I\_letter3,'b','linewidth',2)

end

% 4th segment

for i=1:1:length(t)

Ibra4=IMS.fkine(Itraj4(i,:));

I\_letter4(i,:)=transl(Ibra4);

jta4=I\_letter4;

plot2(jta4(i,:),'r.')

IMS.plot(Itraj4(i,:))

plot2(I\_letter4,'b','linewidth',2)

end

%% M

M0=[-1.0349 -2.0364 7.7836];

M1=[-1.1270 -1.9437 7.7831];

M2=[-1.1394 -1.9634 7.8151];

M3=[-1.0659 -2.1762 7.9544];

M4=[-1.1865 -1.9412 7.8401];

M5=[-1.2200 -1.9071 7.8394];

M6=[-1.1406 -2.1467 7.9997];

M7=[-1.3053 -1.8562 7.8739];

M8=[-1.2571 -1.9863 7.9557];

M9=[-1.2321 -2.0846 8.0291];

M10=[-1.2519 -2.1075 8.0718];

M11=[-1.3127 -2.0192 8.0443];

% Trajectory planning for the letter M

Mtraj1 = jtraj(M0,M1,t);

Mtraj2 = jtraj(M1,M2,t);

Mtraj3 = jtraj(M2,M3,t);

Mtraj4 = jtraj(M3,M4,t);

Mtraj5 = jtraj(M4,M5,t);

Mtraj6 = jtraj(M5,M6,t);

Mtraj7 = jtraj(M6,M7,t);

Mtraj8 = jtraj(M7,M8,t);

Mtraj9 = jtraj(M8,M9,t);

Mtraj10 = jtraj(M9,M10,t);

Mtraj11 = jtraj(M10,M11,t);

hold on

% 1st segment

for i=1:1:length(t)

Ibra=IMS.fkine(Mtraj1(i,:));

M\_letter(i,:)=transl(Ibra);

jta=M\_letter;

plot2(jta(i,:),'y.')

IMS.plot(Mtraj1(i,:))

plot2(M\_letter,'g','linewidth',2)

end

% 2nd segment

for i=1:1:length(t)

Ibra2=IMS.fkine(Mtraj2(i,:));

M\_letter2(i,:)=transl(Ibra2);

jta2=M\_letter2;

plot2(jta2(i,:),'y.')

IMS.plot(Mtraj2(i,:))

plot2(M\_letter2,'g','linewidth',2)

end

% 3rd segment

for i=1:1:length(t)

Ibra3=IMS.fkine(Mtraj3(i,:));

M\_letter3(i,:)=transl(Ibra3);

jta3=M\_letter3;

plot2(jta3(i,:),'y.')

IMS.plot(Mtraj3(i,:))

plot2(M\_letter3,'g','linewidth',2)

end

% 4th segment

for i=1:1:length(t)

Ibra4=IMS.fkine(Mtraj4(i,:));

M\_letter4(i,:)=transl(Ibra4);

jta4=M\_letter4;

plot2(jta4(i,:),'y.')

IMS.plot(Mtraj4(i,:))

plot2(M\_letter4,'g','linewidth',2)

end

% 5th segment

for i=1:1:length(t)

Ibra5=IMS.fkine(Mtraj5(i,:));

M\_letter5(i,:)=transl(Ibra5);

jta5=M\_letter5;

plot2(jta5(i,:),'y.')

IMS.plot(Mtraj5(i,:))

plot2(M\_letter5,'g','linewidth',2)

end

% 6th segment

for i=1:1:length(t)

Ibra6=IMS.fkine(Mtraj6(i,:));

M\_letter6(i,:)=transl(Ibra6);

jta6=M\_letter6;

plot2(jta6(i,:),'y.')

IMS.plot(Mtraj6(i,:))

plot2(M\_letter6,'g','linewidth',2)

end

% 7th segment

for i=1:1:length(t)

Ibra7=IMS.fkine(Mtraj7(i,:));

M\_letter7(i,:)=transl(Ibra7);

jta7=M\_letter7;

plot2(jta7(i,:),'y.')

IMS.plot(Mtraj7(i,:))

plot2(M\_letter7,'g','linewidth',2)

end

% 8th segment

for i=1:1:length(t)

Ibra8=IMS.fkine(Mtraj8(i,:));

M\_letter8(i,:)=transl(Ibra8);

jta8=M\_letter8;

plot2(jta8(i,:),'y.')

IMS.plot(Mtraj8(i,:))

plot2(M\_letter8,'g','linewidth',2)

end

% 9th segment

for i=1:1:length(t)

Ibra9=IMS.fkine(Mtraj9(i,:));

M\_letter9(i,:)=transl(Ibra9);

jta9=M\_letter9;

plot2(jta9(i,:),'y.')

IMS.plot(Mtraj9(i,:))

plot2(M\_letter9,'g','linewidth',2)

end

% 10th segment

for i=1:1:length(t)

Ibra10=IMS.fkine(Mtraj10(i,:));

M\_letter10(i,:)=transl(Ibra10);

jta10=M\_letter10;

plot2(jta10(i,:),'y.')

IMS.plot(Mtraj10(i,:))

plot2(M\_letter10,'g','linewidth',2)

end

% 11th segment

for i=1:1:length(t)

Ibra11=IMS.fkine(Mtraj11(i,:));

M\_letter11(i,:)=transl(Ibra11);

jta11=M\_letter11;

plot2(jta11(i,:),'y.')

IMS.plot(Mtraj11(i,:))

plot2(M\_letter11,'g','linewidth',2)

end

%% Ib

Ib0=[-1.3243 -1.9440 7.9807];

Ib1=[-1.3577 -1.8371 7.9072];

Ib2=[-1.4080 -1.7660 7.8864];

Ib3=[-1.4156 -1.7933 7.9213];

Ib4=[-1.3695 -2.0314 8.1133];

% Trajectory planning for the letter Ib

Ibtraj1 = jtraj(Ib0,Ib1,t);

Ibtraj2 = jtraj(Ib1,Ib2,t);

Ibtraj3 = jtraj(Ib2,Ib3,t);

Ibtraj4 = jtraj(Ib3,Ib4,t);

hold on

% 1st segment

for i=1:1:length(t)

Ibra=IMS.fkine(Ibtraj1(i,:));

Ib\_letter(i,:)=transl(Ibra);

jta=Ib\_letter;

% Draw track points

plot2(jta(i,:),'r.')

% Trajectory animation

IMS.plot(Ibtraj1(i,:))

% Draw a trajectory line

plot2(Ib\_letter,'b','linewidth',2)

end

% 2nd segment

for i=1:1:length(t)

Ibra2=IMS.fkine(Ibtraj2(i,:));

Ib\_letter2(i,:)=transl(Ibra2);

jta2=Ib\_letter2;

plot2(jta2(i,:),'r.')

IMS.plot(Ibtraj2(i,:))

plot2(Ib\_letter2,'b','linewidth',2)

end

% 3rd segment

for i=1:1:length(t)

Ibra3=IMS.fkine(Ibtraj3(i,:));

Ib\_letter3(i,:)=transl(Ibra3);

jta3=Ib\_letter3;

plot2(jta3(i,:),'r.')

IMS.plot(Ibtraj3(i,:))

plot2(Ib\_letter3,'b','linewidth',2)

end

% 4th segment

for i=1:1:length(t)

Ibra4=IMS.fkine(Ibtraj4(i,:));

Ib\_letter4(i,:)=transl(Ibra4);

jta4=Ib\_letter4;

plot2(jta4(i,:),'r.')

IMS.plot(Ibtraj4(i,:))

plot2(Ib\_letter4,'b','linewidth',2)

end

% Move back to origin

IMS.plot(jtraj(Ib4,[0 0 0],t))

Discussion about the code:

Initially the joints are defined using the *link* function, where the type of joint and its Denavit-Hartenberg values are specified.

The four famous Denavit-Hartenberg parameters are:

* a = link length
* d = offset
* alpha = link offset
* theta = joint rotation (Is the joint variable and thus not used in the *link* function)

Secondly, the *SerialLink* function creates the robot/manipulator that will be responsible for sketching out the path.

Thirdly, the joint angles are assigned to specific variables. i.e., S0, S1, etc.

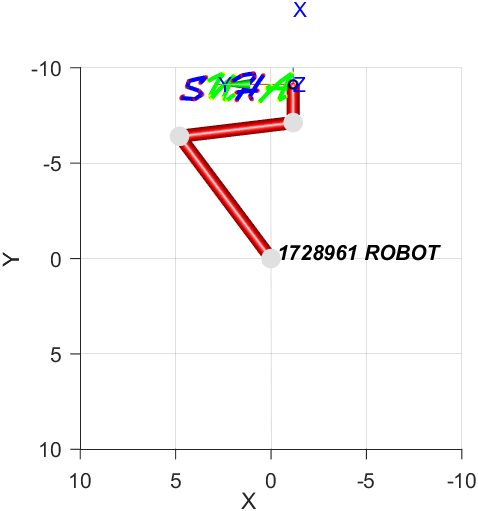
Fourthly, the function *jtraj* is used to calculate the trajectories between two points.

* To do this, a time variable was created, named t, which can be changed to increase or decrease the number of segments desired between any two points.
* In this case, 0.2s was chosen as the time step with an overall time of 1s. So, 5 steps between 2 points.

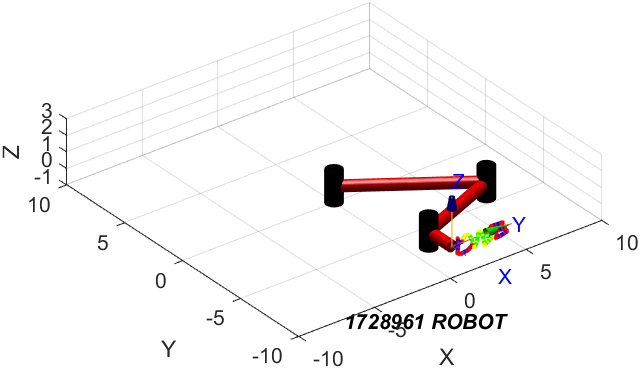
Finally, inside the ‘for’ loop, the *fkine* function is used to perform forward kinematic calculations. These calculations are then used to plot out both the trajectory and the animation of the robot.

This is repeated for each and every letter of the name. The final results are shown below. The *SerialLink.plot* function has several options for camera angle or view of the robot. Three of which are shown in Figure 5,6 and 7.

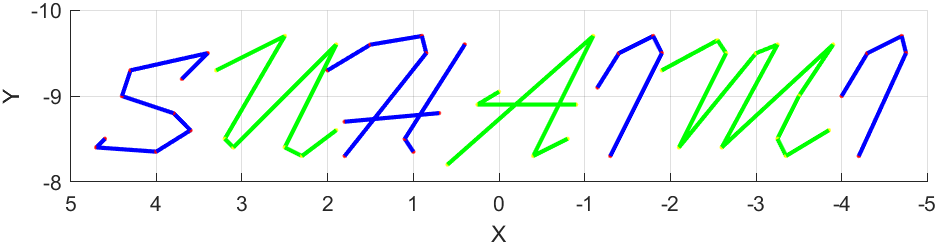
Results:



**Figure 5** Top view of the simulation showing all the links of the robot.



**Figure 6** Orthographic view of the robot during simulation.



**Figure 7** The simulation result of the name done using the Peter Corke RTB toolbox on MATLAB.

Comparison of the two methods:

Instead of one-to-one conventional approach to comparison. A different approach was used to compare between the two methods. The objective of this comparison was to double check the theory that we have learnt by performing the calculations using MATLAB.

But rather than doing the work twice; once on MS Excel and then again on MATLAB. The inverse kinematics calculations done on MS Excel were used to create the simulation on MATLAB.

Since the plots on Excel (Cartesian trajectory) and MATLAB (Joint trajectory) closely resemble each other, it can be said that the inverse kinematic formulas found manually are very much accurate.

In the MATLAB code, the *jta* variables were used to store the join translation component of the homogenous transformation matrix.

The other problem before the values is compared is the fact that the sketch on MATLAB is the points are moving relative to the base of the robot. They are not absolute movements.

The last letter I taken for comparison:

|  |  |  |  |
| --- | --- | --- | --- |
| jta (MATLAB) | | MS Excel | |
| X3 | Y3 | X3 | Y3 |
| -4.00 | -9.00 | 4.00 | 5.00 |
| -4.30 | -9.50 | 4.30 | 5.50 |
| -4.70 | -9.70 | 4.70 | 5.70 |
| -4.75 | -9.50 | 4.75 | 7.50 |
| -4.2 | -8.30 | 4.20 | 6.30 |

It can be seen that the X values are mirrored and that the Y values are shifted by negative 14.

Reasons:

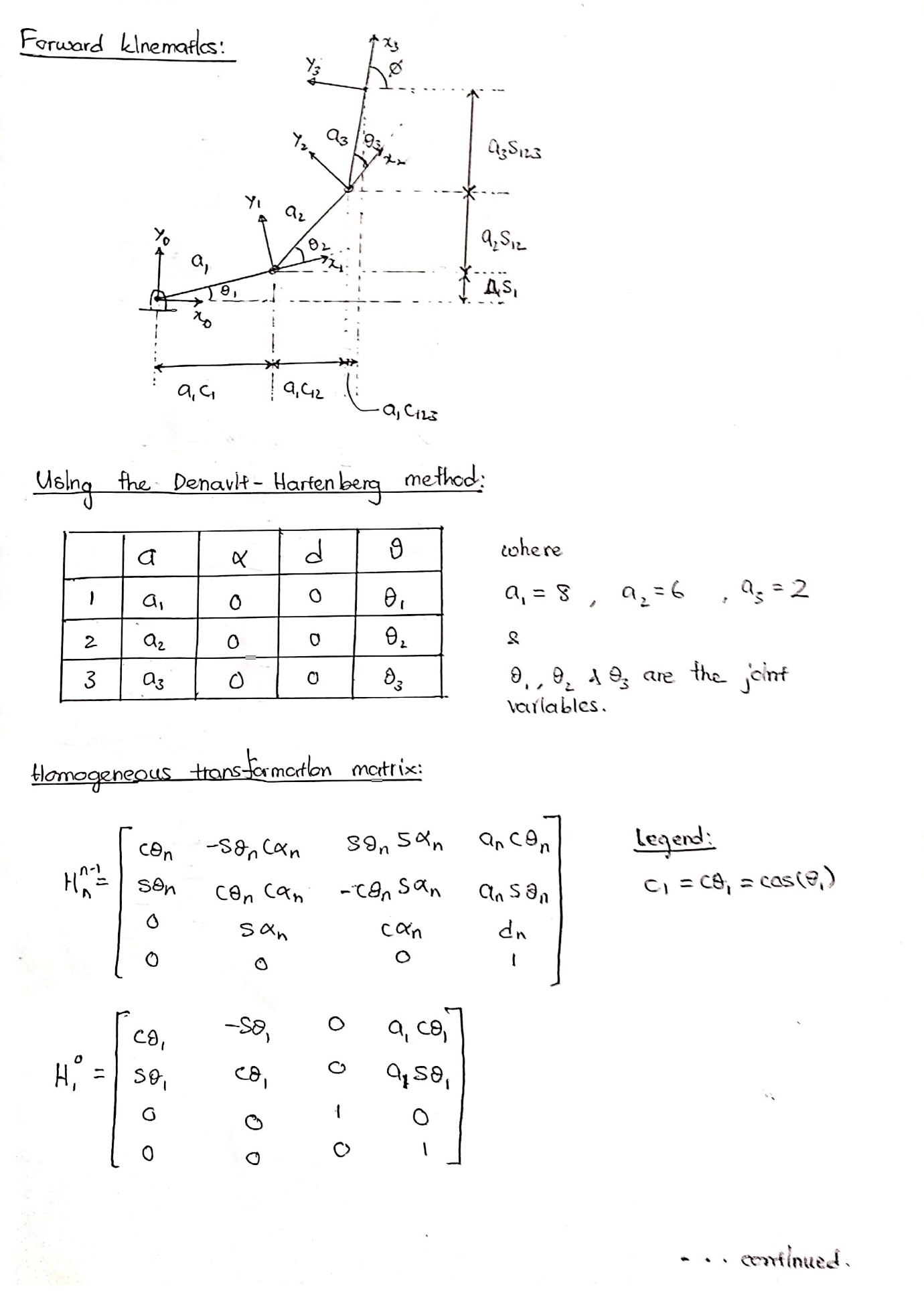
* The designed robot cannot sketch this path at that location.
* The orientation of the robot is wrong.

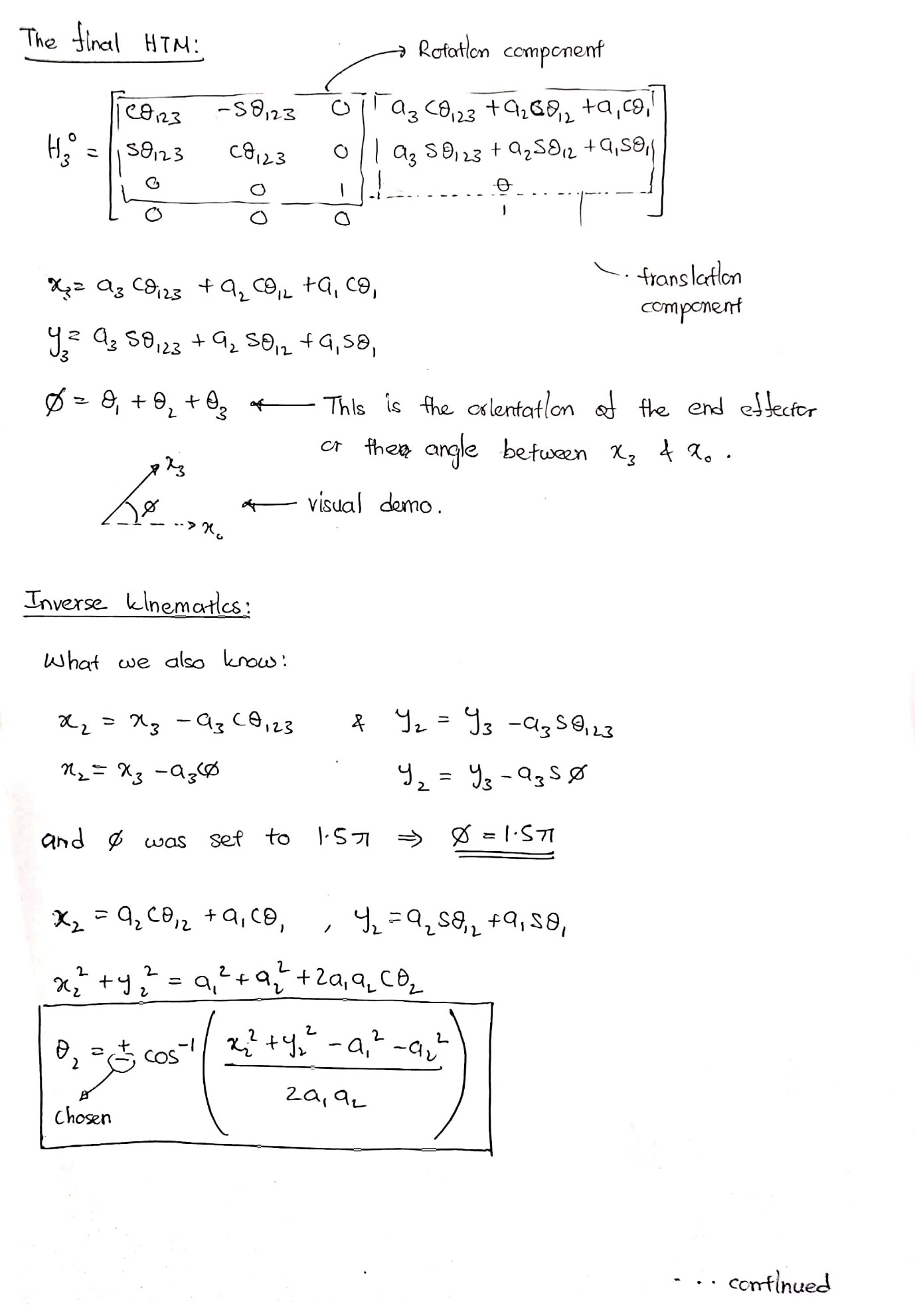
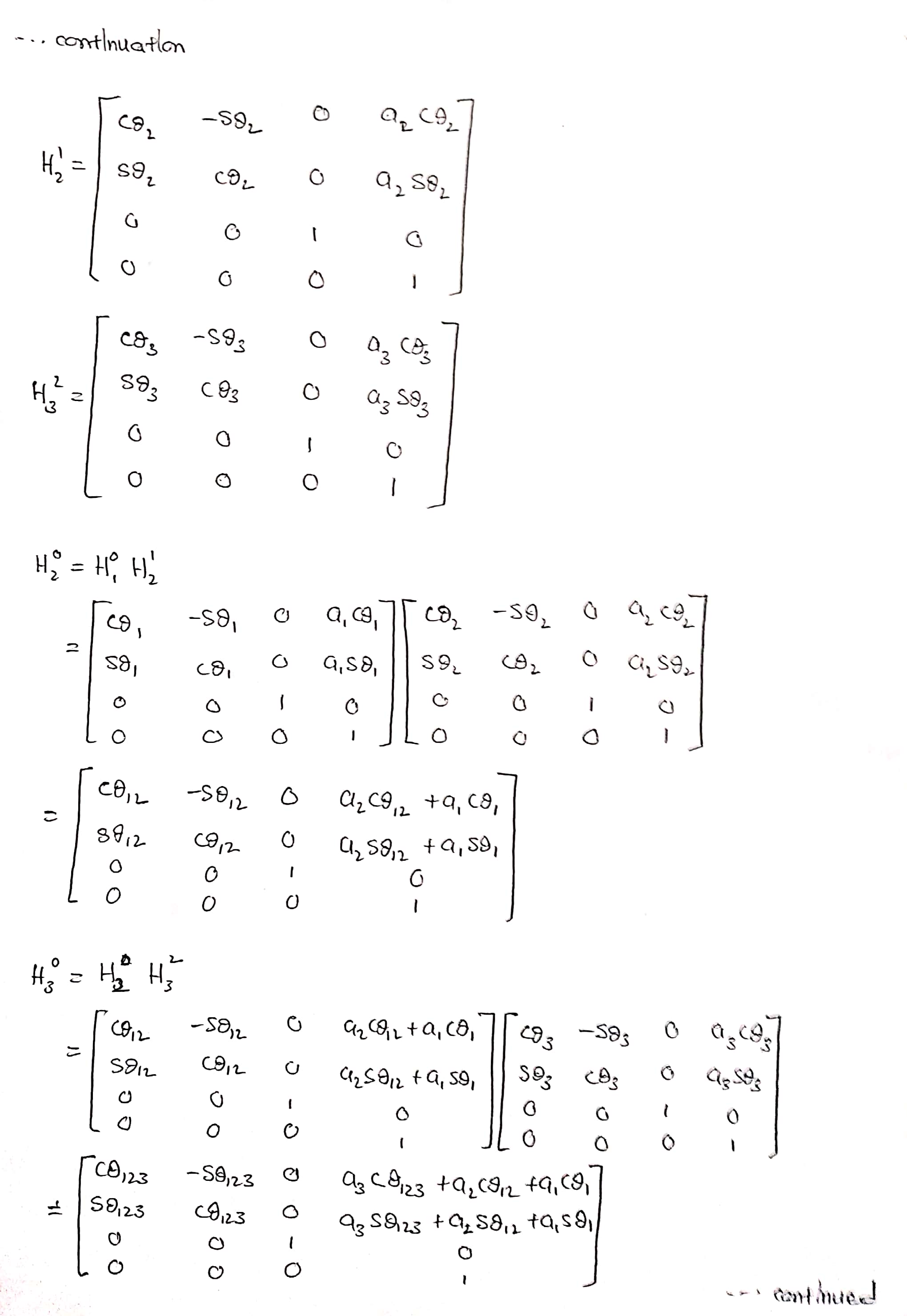
Solution:

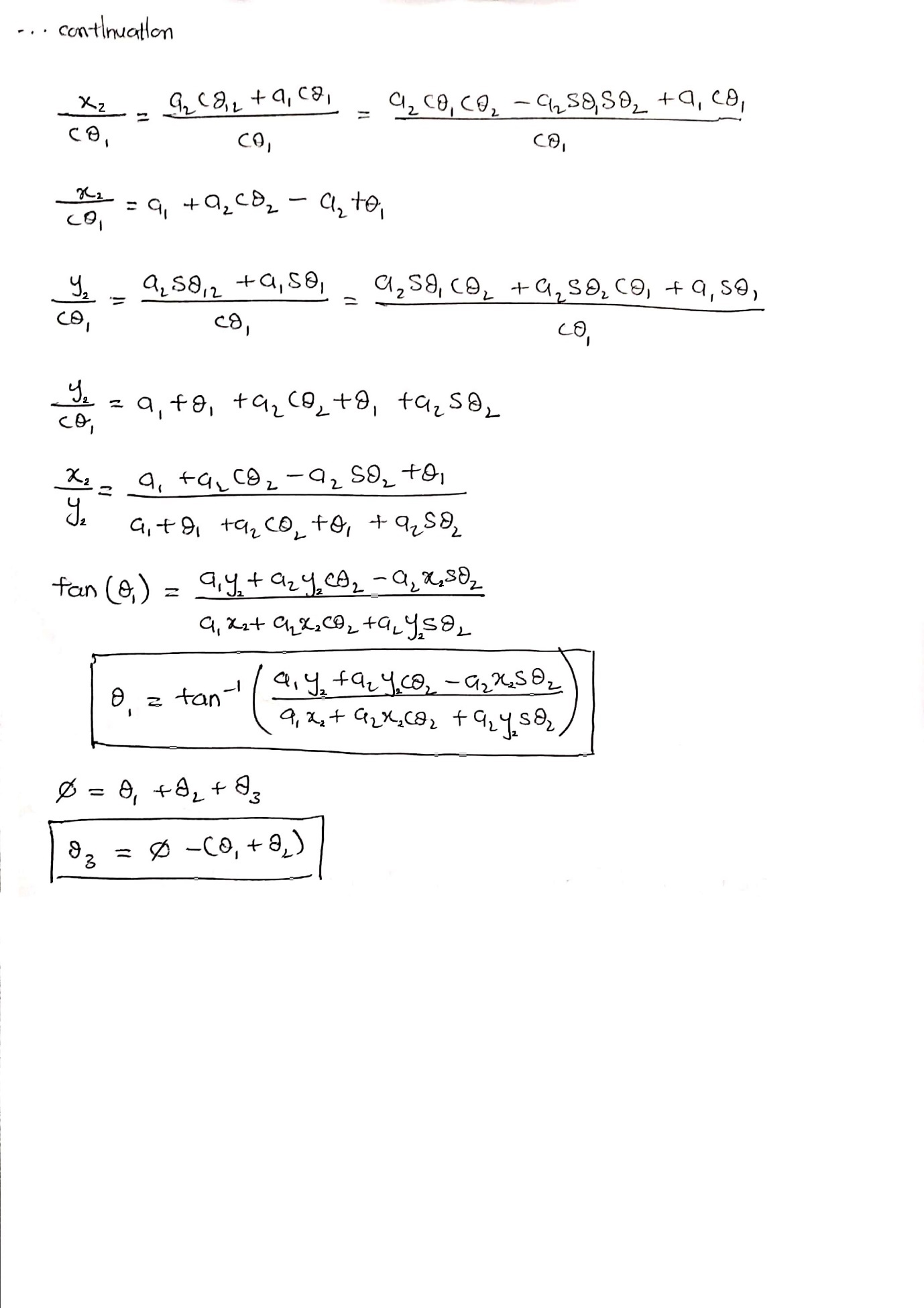
* Use the positive term when calculating for theta2.
* Shift the image/shape up by a few units.

However, the final shape of the path is not affected. The paper has to be place at a specific location but the code does not necessarily require changing.

The sketch was designed such that one unit of the graph is equivalent to 1 decimetre in the real world. This leads to a very nicely sized shape on the A4 paper.

**Manual calculations** 





Conclusion:

Modern simulation tools such as the RTB toolbox created by Peter Corke for MATLAB shows us how easily we can design robots of varying lengths and joints for a range of purposes. But the comparison methodology used in this project also shows the importance of the understanding of the theory that goes behind building these robots.

The Denavit-Hartenberg method to find the homogeneous transformation matrix and the inverse kinematic calculations to find the joint angles are extremely important to design a functional and efficient robot.

Appendix:

Key functions used in the MATLAB program:

**Link.Link**

Create robot link object

This the class constructor which has several call signatures.

**L** = Link() is a Link object with default parameters.

**SerialLink.SerialLink**

Create a SerialLink robot object

**R** = SerialLink(**links**, **options**) is a robot object defined by a vector of Link class objects which can be instances of Link, Revolute, Prismatic, RevoluteMDH or PrismaticMDH.

**jtraj**

Compute a joint space trajectory between two configurations

[**q**,**qd**,**qdd**] = jtraj(**q0**, **qf**, **m**) is a joint space trajectory **q** (MxN) where the joint coordinates vary from **q0** (1xN) to **qf** (1xN). A quintic (5th order) polynomial is used with default zero boundary conditions for velocity and acceleration. Time is assumed to vary from 0 to 1 in **m** steps. Joint velocity and acceleration can be optionally returned as **qd** (MxN) and **qdd** (MxN) respectively. The trajectory **q**, **qd** and **qdd** are MxN matrices, with one row per time step, and one column per joint.

**SerialLink.fkine**

Forward kinematics

**T** = R.fkine(**q**, **options**) is the pose of the robot end-effector as an SE(3) homogeneous transformation (4x4) for the joint configuration **q** (1xN).

If **q** is a matrix (KxN) the rows are interpreted as the generalized joint coordinates for a sequence of points along a trajectory. **q**(i,j) is the j'th joint parameter for the i'th trajectory point. In this case **T** is a 3d matrix (4x4xK) where the last subscript is the index along the path.

[**T**,**all**] = R.fkine(**q**) as above but **all** (4x4xN) is the pose of the link frames 1 to N, such that **all**(:,:,k) is the pose of link frame k.

**SerialLink.plot**

Graphical display and animation

R.plot(**q**, **options**) displays a graphical animation of a robot based on the kinematic model. A stick figure polyline joins the origins of the link coordinate frames. The robot is displayed at the joint angle **q** (1xN), or if a matrix (MxN) it is animated as the robot moves along the M-point trajectory.

**Special note:**

A much simpler code could have been created if functions such as ***mstraj*** were used. But this was avoided as it is beyond the scope of my understanding on the mathematics behind said code.

Inverse kinematics could have been done with MATLAB as well.

***jtraj*** makes a fixed number of straight segments, defined by t. This was taught in the course and is much simpler to understand.