Slovenská Technická Univerzita – Fakulta Elektrotechniky a Informatiky

Ilkovičova 3, 812 19 Bratislava

Zadanie č.1

Vizualizácia priamej kinematickej úlohy

ROB

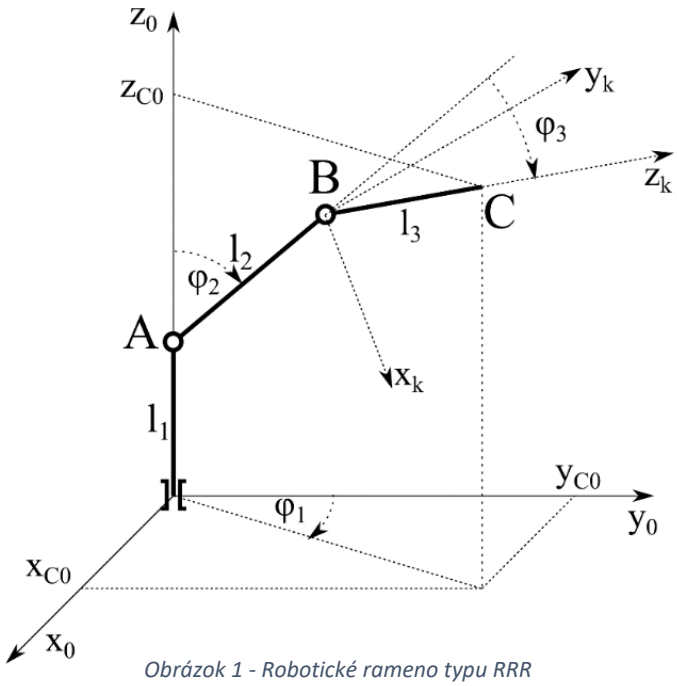
Filip Dubovský

2024

OBSAH

1. ZADANIE

Navrhnite a realizujte vizualizáciu robotického ramena uvedeného na Obrázok 1. Na tomto type zadania by ste si mali precvičiť implementáciu homogénnych transformácií a zafixovať tak preberané učivo Priamej kinematickej úlohy trojramenného manipulátora typu RRR. Všetko potrebné bolo odvodené na cvičeniach.



Obrázok 1 - Robotické rameno typu RRR

Parametre manipulátora:

$l_1 = 300mm$

$l_2 = 300mm$

$l_3 = 200mm$

$\varphi_1 =< -160°, 160° >$

$\varphi_2 =< -50°, 130° >$

$\varphi_3 =< -30°, 60° >$

V rámci riešenia zadania sa zamerajte na nasledovné úlohy:

1. Vykreslenie manipulátora v 3D priestore v zvolenej konfigurácii, vyskúšajte rôzne hodnoty uhlov φ.
2. Vykreslenie jednotlivých pomocných súradných systémov 0-k (x-červenou farbou, y zelenou, z-modrou).
3. Vykreslenie obálky pracovného priestoru daného manipulátora v bázovej rovine manipulátora Y_0Z_0, tiež aj v rovine X_0Y_0.

2. ROZBOR

Cieľom tohto projektu bolo vykresliť robotický manipulátor typu RRR a obálku jeho pracovných priestorov v súlade so zadanými parametrami. Okrem toho bolo za úlohu vykresliť jednotlivé pomocné súradnicové systémy kĺbov A, B a C.

Na vykreslenie manipulátora boli identifikované a vypočítané súradnice jednotlivých kĺbov. Pre zistenie polohy koncového bodu (bodu C) v rôznych polohách v priestore boli vykonané výpočty, ktoré budú neskôr podrobnejšie popísané v tomto dokumente.

2.1. Postup riešenia

Výpočet polohového vektora koncového bodu:

Výsledná transformačná matica medzi nultým a k-tým súradnicovým systémom $T_{0,k}$, bola definovaná postupným súčinom transformačných matíc, ktoré boli spojené s kĺbmi robotického manipulátora. Každý kĺb bol spojený s jedným súradnicovým systémom, kde každé konštrukčné posunutie alebo natočenie zodpovedalo jednému kĺbu. Tieto súradnicové systémy boli usporiadané od svetového (nultého) súradnicového systému až po koncový člen kinematickej štruktúry.

$$T = T_{0,k} = \prod_{i=1}^{k} T_{i-1,i} = T_{0,1} * T_{1,2} * T_{2,3} \dots$$

Rovnica 1 - Všeobecný zápis súčinu transformačných matíc

Polohový vektor koncového bodu v nultom súradnicovom systéme bol vypočítaný z polohového vektora P_k (bodu P v k-tom súradnicovom systéme)

$$P_o = T_{0,k} \cdot P_k$$

Rovnica 2 - Polohový vektor koncového bodu v nultom súradnicovom systéme

Výpočtom sme získali analytické rovnice závislosti svetových súradníc koncového bodu od kĺbových premenných, ktoré sme používali na porovnanie a kontrolu výpočtu.

$$X_{Co} = \sin(\varphi_1) \cdot [l_2 \cdot \sin(\varphi_2) + l_3 \cdot \sin(\varphi_2 + \varphi_3)]$$

$$Y_{Co} = \cos(\varphi_1) \cdot [l_2 \cdot \sin(\varphi_2) + l_3 \cdot \sin(\varphi_2 + \varphi_3)]$$

$$Z_{Co} = l_1 + l_2 \cdot \cos(\varphi_2) + l_2 \cdot \cos(\varphi_2 + \varphi_3)$$

Rovnica 3 - Analytické rovnice

Výpočet súradníc jednotlivých kĺbov:

Polohy jednotlivých kĺbov vo svetovom súradnicovom systéme boli vypočítané postupným súčinom homogénnych transformačných matíc s počiatočným bodom(Rovnica 7, Rovnica 8, Rovnica 9). Pri koncovom bode C bol vykonaný súčin homogénnych transformačných matíc s polohovým vektorom p5 (Rovnica 10), ktorý reprezentoval vzdialenosť koncového bodu od počiatku piateho súradnicového systému. Homogénne transformačné matice boli odvodené v rámci cvičení a ich výsledné tvary boli možné vidieť v Rovnica 4 a Rovnica 5.

$$Rx(\varphi)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos(\varphi) & -\sin(\varphi) & 0\\ 0 & \sin(\varphi) & \cos(\varphi) & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\ Ry(\varphi)=\begin{pmatrix}\cos(\varphi) & 0 & -\sin(\varphi) & 0\\ 0 & 1 & 0 & 0\\ \sin(\varphi) & 0 & \cos(\varphi) & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\ Rz(\varphi)=\begin{pmatrix}\cos(\varphi) & -\sin(\varphi) & 0 & 0\\ \sin(\varphi) & \cos(\varphi) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Rovnica 4 - Homogénne rotačné matice

$$Tx(l)=\begin{pmatrix}1 & 0 & 0 & l\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\ Ty(l)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & l\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\ Tz(l)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & l\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Rovnica 5 - Homogénne translačné matice

$$p0=\begin{pmatrix}0\\ 0\\ 0\\ 1\end{pmatrix}$$

Rovnica 6 - Polohový vektor p0

$$A=Rz(\varphi_1)\cdot Tz(l_1)\cdot p0$$

Rovnica 7 - Výpočet polohy kĺbu A v nultom súradnicovom systéme

$$B=Rz(\varphi_1)\cdot Tz(l_1)\cdot Rx(\varphi_2)\cdot Tz(l_2)\cdot p0$$

Rovnica 8 - Výpočet polohy kĺbu B v nultom súradnicovom systéme

$$C=Rz(\varphi_1)\cdot Tz(l_1)\cdot Rx(\varphi_2)\cdot Tz(l_2)\cdot Rx(\varphi_3)\cdot Tz(l_3)\cdot p0=T_{0,5}\cdot P_5$$

Rovnica 9 - Výpočet polohy kĺbu C v nultom súradnicovom systéme

$$p5=\begin{pmatrix}0\\ 0\\ l_3\\ 1\end{pmatrix}$$

Rovnica 10 - Polohový vektor p5

Výpočet súradnicových systémov 0 až k:

Základný súradnicový systém sme určili s počiatkom v bode Po a jednotkovou veľkosťou. Jednotlivé súradnicové systémy sme vykresľovali rotáciou základného súradnicového systému pomocou homogénnych transformačných matíc. S každým kĺbom bol spojený jeden súradnicový systém, kde každé konštrukčné posunutie alebo natočenie zodpovedalo jednému kĺbu.

Pre názornú ukážku sme uviediem výpočet súradníc prvého a druhého súradnicového systému.

$$x0 = \begin{pmatrix} d \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad y0 = \begin{pmatrix} 0 \\ d \\ 0 \\ 1 \end{pmatrix} \quad z0 = \begin{pmatrix} 0 \\ 0 \\ d \\ 1 \end{pmatrix}$$

Rovnica 11 - Súradnice koncových bodov nultého súradnicového systému

$$x1 = Rz(\varphi_1) \cdot x0$$

$$y1 = Rz(\varphi_1) \cdot y0$$

$$z1 = Rz(\varphi_1) \cdot z0 = z0$$

Rovnica 12 - Výpočet súradníc koncových bodov prvého súradnicového systému

$$x2 = Tz(l_1) \cdot x1$$

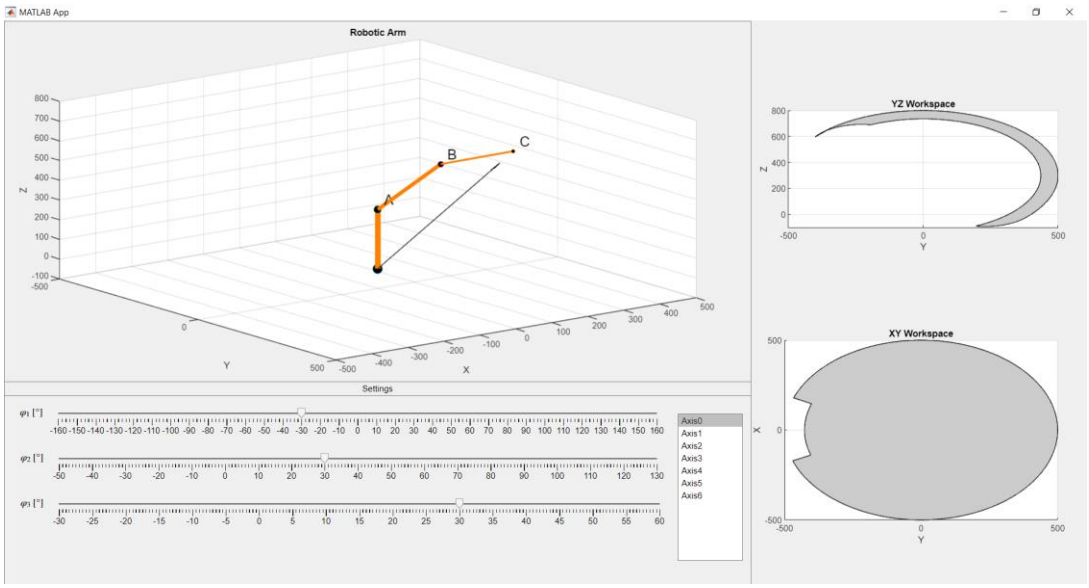
$$y2 = Tz(l_1) \cdot y1$$

$$z2 = Tz(l_1) \cdot z1$$

Rovnica 13 - Výpočet súradníc koncových bodov druhého súradnicového systému

3. PROGRAM A APLIKÁCIA

Aplikácia bola pôvodne implementovaná v prostredí MATLAB s dôrazom na jednoduchosť riešenia. Následne bola prepracovaná do prostredia MATLAB Application, čo umožnilo vylepšenú vizualizáciu a priame vykresľovanie s dynamickou možnosťou menenia hodnôt.

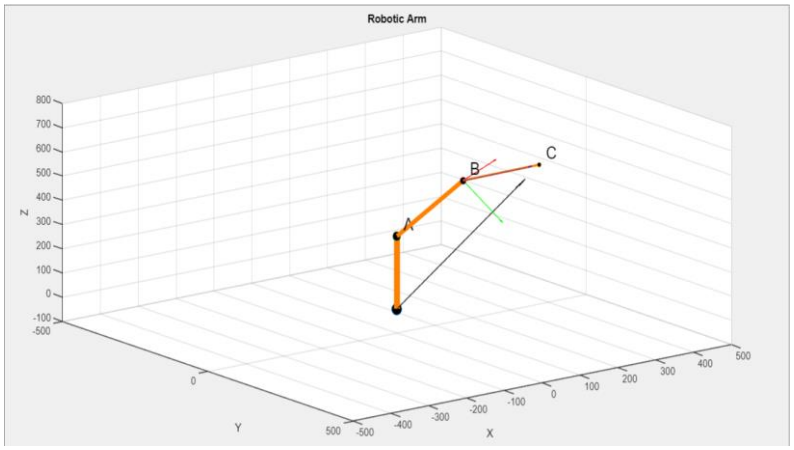


Obrázok 2 - MATLAB App design

Ako vidieť na Obrázok 2 Aplikácia pozostáva z troch modulov na vykresľovanie. Prvý modul slúži na vykreslenie 3D manipulátora v rôznych uhloch, zatiaľ čo druhý a tretí modul sa venujú vykresleniu pracovných priestorov na rovine Y_0Z_0 a na rovine X_0Y_0.

Ďalej obsahuje tri posuvníky, ktoré umožňujú nastavenie hodnôt pre premennú φ a definujú ich maximálny a minimálny rozsah pre rameno.

Na záver je k dispozícii komponent typu "Listbox", ktorý umožňuje výber požadovaných pomocných súradnicových systémov, ktoré majú byť zobrazené.



Obrázok 3 Vykreslenie 5-teho pomocného súradnicového systému

4. KÓD

4.1. Kód aplikácie

Kód aplikácie bol automaticky preddefinovaný s možnosťou upravovať iba určité časti kódu. V nasledujúcom texte budú zahrnuté len úpravy v týchto konkrétnych častiach kódu.

```
properties (Access = private)
    manipulator
end
```

Po spustení kódu sú inicializované rozmery manipulátora a vykoná sa spustenie všetkých funkcií na vykreslenie

```
% Code that executes after component creation
function startupFcn(app)
    % Creating manipulator objeckt
    app.manipulator = Manipulator(300, 300, 200);
    UIAxesButtonDown(app)
    UIAxes2ButtonDown(app)
    UIAxes3ButtonDown(app)
end
```

Funkcia, ktorá využíva hodnoty zo posuvníkov na výpočet pozície koncového bodu a jednotlivých segmentov. Následne vykonáva vykreslenie v 3D priestore.

```
function UIAxesButtonDown(app, event)
    % Reading Slider Values
    fi1 = app.varphi_1Slider.Value;
    fi2 = app.varphi_2Slider.Value;
    fi3 = app.varphi_3Slider.Value;

    % Calculate end effector position
    [Xc0, Yc0, Zc0] = app.manipulator.calculateEndEffectorPosition(-fi1, fi2, fi3);

    % Calculate segments position
    [P, A, B, C] = app.manipulator.calculateRoboticSegmentPosition( -fi1, fi2, fi3);

    % Clear previous plot
    cla(app.UIAxes);

    % Displaying End Effector Position
    plot3(app.UIAxes, 0, 0, 0, 'o', 'MarkerSize', 10, 'MarkerFaceColor', 'black');
    hold(app.UIAxes, 'on');
    xlabel(app.UIAxes, 'X');
    ylabel(app.UIAxes, 'Y');
    zlabel(app.UIAxes, 'Z');
    grid(app.UIAxes, 'on');

    % Display Vector
```

```
    quiver3(app.UIAxes, 0, 0, 0, Xc0, Yc0, Zc0,'black');
    plot3(app.UIAxes, Xc0, Yc0, Zc0, 'o', 'MarkerSize', 4, 'MarkerFaceColor',
    'black');
    text(app.UIAxes, Xc0+10, Yc0+10, Zc0+50, 'C', 'FontSize',
    20,'LineWidth',2, 'HorizontalAlignment', 'left');

    % Display first segment
    plot3(app.UIAxes, [P(1) A(1)], [P(2) A(2)], [P(3)  A(3)],'Color', [1, 0.5,
    0],'LineWidth',6);
    plot3(app.UIAxes, A(1), A(2), A(3), 'o', 'MarkerSize', 8, 'MarkerFaceColor',
    'black');
    text(app.UIAxes, A(1)+10, A(2)+10, A(3)+50, 'A', 'FontSize', 20,'LineWidth',2,
    'HorizontalAlignment', 'left');

    % Display second segment
    plot3(app.UIAxes, [A(1) A(1)+B(1)], [A(2) B(2)+A(2)], [A(3)B(3)+A(3)],'Color',
    [1, 0.5, 0],'LineWidth',4);
    plot3(app.UIAxes, A(1)+B(1), A(2)+B(2), A(3)+B(3), 'o', 'MarkerSize', 6,
    'MarkerFaceColor', 'black');
    text(app.UIAxes, A(1)+B(1)+10, A(2)+B(2)+10, A(3)+B(3)+50, 'B','FontSize',
    20,'LineWidth',2, 'HorizontalAlignment', 'left');

    % Display third segment
    plot3(app.UIAxes, [A(1)+B(1) A(1)+B(1)+C(1)], [A(2)+B(2) A(2)+B(2)+C(2)],
    [A(3)+B(3) A(3)+B(3)+C(3)],'Color', [1, 0.5, 0],'LineWidth',2);

end
```

Pre každý posuvník je zadefinovaná funkcia, ktorá pri zmene posuvníka spustí funkciu na vykreslenie ramena.

```
function varphi_1SliderValueChanged(app, event)
    UIAxesButtonDown(app, event)
end
```

Funkcia, ktorá inicializuje vektory a následne ukladá do nich hodnoty krajnej trajektórie pracovného priestoru v rovine Y_0Z_0.

```
function UIAxes2ButtonDown(app, event)
    fi1 = 0;
    fi2 = -50; % Starting position for YZ trajectory
    fi3 = -10;
    size = 30+180+60+180+90-40-20;

    % Define arrays to store end effector positions
    X = zeros(size, 1);
    Y = zeros(size, 1);
    Z = zeros(size, 1);

    % Calculate end effector trajectory
    for i = 1:size
            if fi3 < 0 && fi2 == -50
                fi3 = fi3 + 1;
```

```
            elseif fi3 == 0 && fi2 >= -50 && fi2 < 130
                fi2 = fi2 + 1;
            elseif fi2 == 130 && fi3 >= 0 && fi3 < 60
                fi3 = fi3 + 1;
            elseif fi3 == 60 && fi2 <= 130 && fi2 > -50
                fi2 = fi2 - 1;
            else
                fi3 = fi3 - 1;
        end

        % Calculate end effector position
        [X(i), Y(i), Z(i)] = app.manipulator.calculateEndEffectorPosition(-
                             fi1, fi2, fi3);
    end

    % Plot the end effector trajectory
    plot(app.UIAxes2, Y, Z,'black');
    xlabel(app.UIAxes2, 'Y');
    ylabel(app.UIAxes2, 'Z');
    grid(app.UIAxes2, 'on');

    % Fill the plotted area
    fill(Y, Z, [0.8 0.8 0.8], 'Parent', app.UIAxes2);
end
```

Funkcia, ktorá inicializuje vektory a následne ukladá do nich hodnoty vnútornej a potom vonkajšej krajnej trajektórie pracovného priestoru v rovine X_0Y_0.

```
function UIAxes3ButtonDown(app, event)
    fi1 = -160;
    fi2 = 90; % Starting position for XY trajectory
    fi3 = 0;
    size = 160*2;

    % Define arrays to store end effector positions
    X = zeros(size, 1);
    Y = zeros(size, 1);
    Z = zeros(size, 1);

            % Calculate end effector trajectory
    for i = 1:size
        fi1 = fi1 + 1;

        % Calculate end effector position
        [X(i), Y(i), Z(i)] = app.manipulator.calculateEndEffectorPosition(-
                             fi1, fi2, fi3);
    end

    fi1 = -20;
    fi2 = -50; % Starting position for XY trajectory
    fi3 = -30;
    size1 = 40;
    X1 = zeros(size1, 1);
```

```
    Y1 = zeros(size1, 1);
    Z1 = zeros(size1, 1);

    for i = 1:size1
        fi1 = fi1 + 1;

        % Calculate end effector position
        [X1(i), Y1(i), Z1(i)] = app.manipulator.calculateEndEffectorPosition(
                                -fi1, fi2, fi3);
    end


    % Plot the end effector trajectory
    plot(app.UIAxes3, Y, X,'black');
    hold(app.UIAxes3, 'on');
    plot(app.UIAxes3, Y1, X1,'black');
    plot(app.UIAxes3, [Y(1) Y1(size1)], [X(1) X1(size1)],'black');
    plot(app.UIAxes3, [Y(size) Y1(1)], [X(size) X1(1)],'black');
    xlabel(app.UIAxes3, 'Y');
    ylabel(app.UIAxes3, 'X');
    grid(app.UIAxes3, 'on');

    % Fill the plotted area
    fill([Y; Y1; Y(1)], [X; X1; X(1)], [0.8 0.8 0.8], 'Parent', app.UIAxes3);
end
```

Funkcia, ktorá na základe vybranej položky vykresľuje pomocné osy.

```
function ListBoxValueChanged(app, event)
    value = app.ListBox.Value; value = app.ListBox.Value;

    % Reading Slider Values
    fi1 = app.varphi_1Slider.Value;
    fi2 = app.varphi_2Slider.Value;
    fi3 = app.varphi_3Slider.Value;


    if strcmp(value, 'Axis0')
        [X, Y, Z, P] = app.manipulator.axer_0();
    elseif strcmp(value, 'Axis1')
        [X, Y, Z, P] = app.manipulator.axer_1(fi1);
    elseif strcmp(value, 'Axis2')
        [X, Y, Z, P] = app.manipulator.axer_2(fi1);
    elseif strcmp(value, 'Axis3')
        [X, Y, Z, P] = app.manipulator.axer_3(fi1,-fi2);
    elseif strcmp(value, 'Axis4')
        [X, Y, Z, P] = app.manipulator.axer_4(fi1,-fi2);
    elseif strcmp(value, 'Axis5')
        [X, Y, Z, P] = app.manipulator.axer_5(fi1,-fi2,-fi3);
    elseif strcmp(value, 'Axis6')
        [X, Y, Z, P] = app.manipulator.axer_6(fi1,-fi2,-fi3);
    end
    UIAxesButtonDown(app, event)
    % Plot the vectors
```

```
    quiver3(app.UIAxes, P(1), P(2), P(3), X(1), X(2), X(3), 'r');
    quiver3(app.UIAxes, P(1), P(2), P(3), Y(1), Y(2), Y(3), 'g');
    quiver3(app.UIAxes, P(1), P(2), P(3), Z(1), Z(2), Z(3), 'b');
end
```

4.2. Kód Triedy Manipulátor

Trieda, ktorá obsahuje jednotlivé rozmery manipulátora a funkcie na prácu s manipulátorom.

```
classdef Manipulator
    properties
        l1
        l2
        l3
    end
```

Funkcia, ktorá pridelí parametre objektu:

```
methods
    function obj = Manipulator(l1, l2, l3)
        obj.l1 = l1;
        obj.l2 = l2;
        obj.l3 = l3;
    end
```

Funkcia, ktorá vypočíta vektor koncového bodu:

```
% Calculate end effector position
function [Xc0, Yc0, Zc0] = calculateEndEffectorPosition(obj, fi1, fi2, fi3)
    fi1 = fi1 * pi/180;
    fi2 = fi2 * pi/180;
    fi3 = fi3 * pi/180;

    Xc0 = sin(fi1)*(obj.l2*sin(fi2) + obj.l3*sin(fi2+fi3));
    Yc0 = cos(fi1)*(obj.l2*sin(fi2) + obj.l3*sin(fi2+fi3));
    Zc0 = obj.l1 + obj.l2*cos(fi2) + obj.l3*cos(fi2+fi3);
end
```

Funkcia, ktorá vypočíta vektory jednotlivých segmentov pomocou homogénnych matíc:

```
% Calculate robotic segment position
function [P, A, B, C] = calculateRoboticSegmentPosition(obj, fi1, fi2, fi3)
    fi1 = fi1 * pi/180;
    fi2 = fi2 * pi/180;
    fi3 = fi3 * pi/180;

    P = [0 0 0 1];
    A = P * transpose(Translation.Z(obj.l1)) * Rotation.Z(fi1);
    B = P * transpose(Translation.Z(obj.l2)) * Rotation.X(fi2) * Rotation.Z(fi1);
```

```
    C = P * transpose(Translation.Z(obj.l3)) * Rotation.X(fi2) * Rotation.X(fi3) *
        Rotation.Z(fi1);
end
```

Funkcie, ktoré vypočíta vektory k-tych pomocných súradnicových systémov pomocou homogénnych matíc:

Otočí vektory jednotlivých pomocných osi a otočí a predĺži vektor začiatku osí.

```
%% Axis plotting

function [X, Y, Z, P] = axer_0(obj)
    P = [0 0 0 1];
    X = [200 0 0 1];
    Y = [0 200 0 1];
    Z = [0 0 200 1];
end

function [X, Y, Z, P] = axer_1(obj,fi1)
    fi1 = fi1 * pi/180;
    [X, Y, Z, P] = axer_0(obj);

    X = X * transpose(Rotation.Z(fi1));
    Y = Y * transpose(Rotation.Z(fi1));
    Z = Z * transpose(Rotation.Z(fi1));
    P = P * transpose(Rotation.Z(fi1));
end

function [X, Y, Z, P] = axer_2(obj,fi1)
    fi1 = fi1 * pi/180;
    [X, Y, Z, P] = axer_0(obj);

    % Rotating XYZ
    X = X * transpose(Rotation.Z(fi1));
    Y = Y * transpose(Rotation.Z(fi1));
    Z = Z * transpose(Rotation.Z(fi1));

    % Finding starting position
    P = P * transpose(Translation.Z(obj.l1));
end

function [X, Y, Z, P] = axer_3(obj,fi1,fi2)
    fi1 = fi1 * pi/180;
    fi2 = fi2 * pi/180;
    [X, Y, Z, P] = axer_0(obj);

    % Rotating XYZ
    X = X * transpose(Rotation.X(fi2)) * transpose(Rotation.Z(fi1));
    Y = Y * transpose(Rotation.X(fi2)) * transpose(Rotation.Z(fi1));
    Z = Z * transpose(Rotation.X(fi2)) * transpose(Rotation.Z(fi1));

    % Finding starting position
    P = P * transpose(Translation.Z(obj.l1));
end
```

```
function [X, Y, Z, P] = axer_4(obj,fi1,fi2)
    fi1 = fi1 * pi/180;
    fi2 = fi2 * pi/180;
    [X, Y, Z, P] = axer_0(obj);

    % Rotating XYZ
    X = X * transpose(Rotation.X(fi2)) * transpose(Rotation.Z(fi1));
    Y = Y * transpose(Rotation.X(fi2)) * transpose(Rotation.Z(fi1));
    Z = Z * transpose(Rotation.X(fi2)) * transpose(Rotation.Z(fi1));

    % Finding starting position
    P1 = P * transpose(Translation.Z(obj.l1));
    P2 = P * transpose(Translation.Z(obj.l2)) * transpose(Rotation.X(fi2)) *
         transpose(Rotation.Z(fi1));
    P = P1 + P2;
end

function [X, Y, Z, P] = axer_5(obj,fi1,fi2,fi3)
    fi1 = fi1 * pi/180;
    fi2 = fi2 * pi/180;
    fi3 = fi3 * pi/180;
    [X, Y, Z, P] = axer_0(obj);

    % Rotating XYZ
    X = X * transpose(Rotation.X(fi3)) * transpose(Rotation.X(fi2)) *
        transpose(Rotation.Z(fi1));
    Y = Y * transpose(Rotation.X(fi3)) * transpose(Rotation.X(fi2)) *
        transpose(Rotation.Z(fi1));
    Z = Z * transpose(Rotation.X(fi3)) * transpose(Rotation.X(fi2)) *
        transpose(Rotation.Z(fi1));

    % Finding starting position
    P1 = P * transpose(Translation.Z(obj.l1));
    P2 = P * transpose(Translation.Z(obj.l2)) * transpose(Rotation.X(fi2)) *
         transpose(Rotation.Z(fi1));
    P = P1 + P2;
end

function [X, Y, Z, P] = axer_6(obj,fi1,fi2,fi3)
    fi1 = fi1 * pi/180;
    fi2 = fi2 * pi/180;
    fi3 = fi3 * pi/180;
    [X, Y, Z, P] = axer_0(obj);

    % Rotating XYZ
    X = X * transpose(Rotation.X(fi3)) * transpose(Rotation.X(fi2)) *
        transpose(Rotation.Z(fi1));
    Y = Y * transpose(Rotation.X(fi3)) * transpose(Rotation.X(fi2)) *
        transpose(Rotation.Z(fi1));
    Z = Z * transpose(Rotation.X(fi3)) * transpose(Rotation.X(fi2)) *
        transpose(Rotation.Z(fi1));

    % Finding starting position
    P1 = P * transpose(Translation.Z(obj.l1));
    P2 = P * transpose(Translation.Z(obj.l2)) * transpose(Rotation.X(fi2)) *
```

```
        transpose(Rotation.Z(fi1));
    P3 = P * transpose(Translation.Z(obj.l3)) * transpose(Rotation.X(fi3)) *
        transpose(Rotation.X(fi2)) * transpose(Rotation.Z(fi1));
    P = P1 + P2 + P3;
end
```

4.3. Kód Triedy Rotácie

Trieda, ktorá obsahuje jednotlivé rotačné homogénne matice.

```
classdef Rotation

    methods (Static)
        function Rx = X(angle)
            Rx = [1 0 0 0;
                  0 cos(angle) -sin(angle) 0;
                  0 sin(angle) cos(angle) 0;
                  0 0 0 1];
        end

        function Ry = Y(angle)
            Ry = [cos(angle) 0 sin(angle) 0;
                  0 1 0 0;
                  -sin(angle) 0 cos(angle) 0;
                  0 0 0 1];
        end

        function Rz = Z(angle)
            Rz = [cos(angle) -sin(angle) 0 0;
                  sin(angle) cos(angle) 0 0;
                  0 0 1 0;
                  0 0 0 1];
        end
    end
end
```

4.4. Kód Triedy Translácie

Trieda, ktorá obsahuje jednotlivé translačné homogénne matice.

```
classdef Translation

    methods (Static)
        function Tx = X(d)
            Tx = [1 0 0 d;
                  0 1 0 0;
                  0 0 1 0;
                  0 0 0 1];
        end

        function Ty = Y(d)
            Ty = [1 0 0 0;
```

```
                    0 1 0 d;
                    0 0 1 0;
                    0 0 0 1];
        end

        function Tz = Z(d)
            Tz = [1 0 0 0;
                  0 1 0 0;
                  0 0 1 d;
                  0 0 0 1];
        end
    end
end
```

5. ZÁVER

Cieľom tohto projektu bolo navrhnúť a implementovať robotický manipulátor a simultánne precvičiť aplikáciu homogénnych transformácií. Manipulátor bol úspešne modelovaný a vykreslený v prostredí MATLAB Application. Avšak, pôvodné riešenie využívalo funkciu "quiver3" na vykresľovanie, ktorá bola neskôr nahradená funkciou "plot3". Táto zmena v syntaxe spôsobila zvýšenú komplexitu kódu, ktorá by mohla byť ďalej optimalizovaná.

Vykresľovanie pracovných priestorov prebehlo úspešne, avšak pre zvýšenie úplnosti by bolo vhodné vykresliť nielen krajnú trajektóriu, ale celý pracovný priestor manipulátora, ktorý sme pre jednoduchosť programu zanedbali.

Vykreslenie pomocných osí prebiehalo účinne, no pre lepšiu funkcionalitu by bolo vhodné pridať možnosť vykresľovania viacerých osí súčasne.

Týmito úpravami by sa dosiahlo zvýšenie efektivity a celkového výkonu aplikácie.

„Zadanie som vypracoval sám. Čestne prehlasujem, že som ho neskopíroval a nikomu inému neposkytol. Nech mi je Isaac Asimov svedkom.“

6. ZDROJE

[1.] MathWorks. (n.d.). MATLAB Documentation. Retrieved from https://www.mathworks.com/help/matlab/

[2.] OpenAI. (n.d.). ChatGPT: A State-of-the-Art Conversational AI. Retrieved from https://openai.com/chatgpt/