NEUMANN JÁNOS
INFORMATIKAI KAR

Diplomamunka

OE-NIK Hallgató neve: **Schmidt László**
2022 Hallgató törzskönyvi száma: **T-1220/FI38878/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Biomatika és Alkalmazott Mesterséges Intelligencia Intézet

DIPLOMAMUNKA FELADATLAP

Hallgató neve: **Schmidt László**
Törzskönyvi száma: T/005381/FI12904/N
Neptun kódja: [redacted]

A diplomamunka címe:

Robotkarok pályatervezése és megjelenítése virtuális valóság környezetben
Motion planning and visualization of robotic arms in virtual reality

Intézményi konzulens: Dr. Drexler Dániel András
Külső konzulens:

Beadási határidő: 2022. május 15.

A feladat

A jelölt első feladata Matlab-ban implementált pályatervező algoritmusok illetsztése robotok szimulációjára használt v-rep virtuális valóság eszközhöz. A v-repben megadott robot/robotok pályatervező algoritmusai Matlab alatt futnak, míg a szimuláció v-rep-ben történik. A jelölt második feladata különböző pályatervező algoritmusok implementálása Matlab környezetben, és az összeillesztett eszköz tesztelésének elvégzése.

A diplomamunkának tartalmaznia kell:

- az irodalomkutatás során megismert, a megvalósításhoz felhasznált keretrendszerek leírását,
- a megvalósítandó szoftver részletes tervét, az egyes szoftverkomponensek leírását,
- a megvalósított pályatervező algoritmusok leírását, összehasonlítását,
- az elkészült szoftver dokumentációját,
- az elkészült szoftver tesztelését, annak értékelését.

Ph.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

.....
Dr. Eigner György
intézetigazgató

A diplomamunka elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A diplomamunkát beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

7. melléklet

HALLGATÓI NYILATKOZAT

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022. 05. 11.

.....

hallgató aláírása

9. melléklet

ÓBUDAI EGYETEM

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: SCHMIDT CSABA | Neptun Kód: [redacted] | Tagozat: ESTI

Telefon: [redacted] | Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[7] címe magyarul:
Robotkarok pályatervezése és megjelenítése virtuális valóság környezetben

Szakdolgozat / Diplomamunka[8] címe angolul:
Motion planning and visualization of robotic arms in virtual reality

Intézményi konzulens: Dr. Drexler Dániel András | Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2018.03.20.	TÉMA KIADÁSA	Dr. DDA
2.	2018.04.10.	~~KÖZ~~ IRODALOM ÁTBESZÉLÉSE	Dr. DDA
3.	2018.05.03.	ELŐREHALADÁS MONITOROZÁSA	Dr. DDA
4.	2018.05.18.	BESZÁMOLÓ ÁTBESZÉLÉSE	Dr. DDA

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[9] tantárgy követelményét teljesítette, beszámolóra / védésre[10] bocsátható.

Dr. DDA

Intézményi konzulens

Budapest, 2018.05.18.

---

[7] Megfelelő aláhúzandó!

[8] Megfelelő aláhúzandó!

[9] Megfelelő aláhúzandó!

[10] Megfelelő aláhúzandó!

9. melléklet

ÓBUDAI EGYETEM

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: SCHMIDT LÁSZLÓ

Neptun Kód: [redacted]

Tagozat: ESTI

Telefon:

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[7] címe magyarul:
Robotkarok pályatervezése és megjelenítése virtuális valóság környezetben

Szakdolgozat / Diplomamunka[8] címe angolul:
Motion planning and visualization of robotic arms in virtual reality

Intézményi konzulens: Dr. Drexler Dániel András

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2018.03.20.	TÉMA ÁTBESZÉLÉSE	
2.	2018.04.10.	IMPLEMENTÁCIÓ MEGTERVEZÉSE	
3.	2018.05.08.	ELŐREHALADÁS MONITOROZÁSA	
4.	2018.05.18.	BESZÁMOLÓ ÁTBESZÉLÉSE	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4 [9] tantárgy követelményét teljesítette, beszámolóra / védésre [10]bocsátható.

Intézményi konzulens

Budapest, 20.18.05.18.

[7] Megfelelő aláhúzandó!

[8] Megfelelő aláhúzandó!

[9] Megfelelő aláhúzandó!

[10] Megfelelő aláhúzandó!

9. melléklet

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: SCHMIDT LÁSZLÓ

Neptun Kód: [redacted]

Tagozat: ESTI

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[6] címe magyarul:
ROBOT KAROK PÁLYATERVEZÉSE ÉS MEGJELENÍTÉSE VIRTUÁLIS VALÓSÁG KÖRNYEZETBEN

Szakdolgozat / Diplomamunka[7] címe angolul:
MOTION PLANNING AND VISUALIZATION OF ROBOTIC ARMS IN VIRTUAL REALITY

Intézményi konzulens: DR. DRÉXLER DÁNIEL ANDRÁS

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.02.25.	EDDIGI EREDMÉNYEK ÁTTEKINTÉSE	
2.	2022.03.18.	IMPLEMENTÁCIÓ ELLENŐRZÉSE	
3.	2022.04.22.	TESZTELÉSI EREDMÉNYEK MEGVITATÁSA	
4.	2022.05.06.	DOKUMENTÁCIÓ ÁTTEKINTÉSE	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4 [8] tantárgy követelményét teljesítette, beszámolóra / védésre [9] bocsátható.

Intézményi konzulens

Budapest, 2022.05.09

[6] Megfelelő aláhúzandó!

[7] Megfelelő aláhúzandó!

[8] Megfelelő aláhúzandó!

[9] Megfelelő aláhúzandó!

Tartalomjegyzék

Ábrajegyzék

Robotkarok pályatervezése és megjelenítése virtuális valóság környezetben

Absztrakt

Az ipari robotkarok pályatervezése összetett mérnöki probléma, melyre nincs egy általánosan legjobbnak mondható megoldás. Az eszközök költségei miatt valósághű szimulációs lehetőségek kulcsfontosságúvá váltak a robotika világában dolgozók számára. Erre a két problémára keresek elfogadható megoldást a dolgozatomban.

A diplomamunka fejezetei áttekintik a robotok általános és a pályatervezés problémakörének elméleti hátterét. Kiemelt hangsúlyt kap az inverz kinematikai feladat megoldásához szükséges Jacobi-mátrix. Bemutatom a megoldást CLIK algoritmussal, ami a pályakövetési hiba kiküszöbölésére szolgál. Emellett kitérek a szinguláris konfigurációk kiszűrésének jelentőségére. A szimulációs lehetőségek közül részletesen megvizsgálom a CoppeliaSim alkalmazást.

Megvalósításra kerül egy pályatervező algoritmus, amely egyszerű és összetett referencia pályákat készít egy UR5-ös robotkar számára. A CLIK algoritmus is implementálásra kerül és segítségével történik a referencia pálya bejárása. A szinguláris esetek kiküszöbölésére a létrehozott rendszer részeként kap helyet a Jacobi-mátrix kondíciószám vizsgálata és a determináns figyelése. A szimulációs probléma feloldásaként létrehozásra kerül egy interfész a MatLab és a CoppeliaSim alkalmazások között, illetve egy a számolásokból érkező adatokat fogadó szimulációs jelenet a CoppeliaSim oldalán. Ezután az elkészült eseteket tesztelésnek vetem alá, amit grafikonok segítségével részletesen kielemzek.

A diplomamunka során megvalósított algoritmus tesztelése megmutatta, hogy a CLIK algoritmus alacsony hibaértéken tudja tartani a robotkarok pályabejárását. A tesztestek rávilágítanak, hogy a szinguláris konfigurációk elkerülésére nem a determináns, hanem a kondíciószámok figyelése hoz pontosabb eredményt. A valósághű szimulációra jó megoldást nyújt a CoppeliaSim alkalmazással történi integráció.

Abstract

The path planning of industrial robotic arms is a complex engineering problem which has no general best solution. Because of the cost of the devices, realistic simulation capabilities have become key tools for those working in the world of robotics. I am looking for an acceptable solution to these two problems during my thesis.
The chapters of the thesis review the general background of robotics and path planning. Emphasis is placed on the Jacobian matrix required to solve the inverse kinematic problem. I present the CLIK algorithm, which is used to eliminate the trajectory tracking error. I also address the importance of excluding singular configurations. Among the simulation options, I examine CoppeliaSim in detail.
A path planning algorithm is implemented that creates simple and complex reference paths for an UR5 robotic arm. The CLIK algorithm is also implemented to help following the reference path. In order to eliminate singular cases, the monitoring of the condition number and the determinant of the Jacobian matrix are part of the established system. To solve the simulation problem, an interface is created between MatLab and CoppeliaSim, and a simulation scene on the CoppeliaSim receives data from the calculations. I tested the created cases, which I analyzed in detail using graphs.
Testing of the algorithm implemented during the thesis showed that the CLIK algorithm can keep the robot path error at a low value. Test cases show that monitoring of condition numbers rather than the determinant to avoid singular configurations brings more reliable results. Integration with CoppeliaSim provides a good solution for realistic simulation.

1 Bevezetés

A robotok mára a mindennapi életünk részévé váltak. Szinte minden munkahelyen szóba kerül, hogy mely humán feladatokat lehetne kiváltani robotok segítségével. Ezek a feladatok szinte kizárólag mozgatási feladatok, amihez egy meghatározott célpontra kell mozgatni az eszközöket. Az ipari robotok alkalmazásában az egyik legáltalánosabb problémakör a pályatervezés és pályakövetés. A dolgozat célja, hogy megoldást adjon egyszerű és összetett pályák tervezésére, illetve hogy megvalósítson egy pályakövetési megoldást.

A robot elterjedésével párhuzamosan megnőtt az igény a tanulmányozásuk, tervezésük, irányításuk iránt. Mivel ezek a gépek még nagyon költségesek, a megfelelő tudást jelenleg csak néhány jól felszerelt laborban lehet megszerezni és kamatoztatni. Jelen diplomamunka célja, hogy ezt a szűk keresztmetszetet enyhítse és eszközt biztosítson a robotok szimulálására, korszerű eszközökkel.

A 2. fejezetben áttekintjük a robotirányítás elméleti hátterét. A feladatmegoldás során használt fontosabb algoritmusok levezetése is ebben a szekcióban kap helyet. Ezután következik a 3. fejezet a feladat megoldásának logikai felépítésével és a 2. fejezetben taglalt elméleti képletek gyakorlatba történő átültetésével. A feladatban az oktatásban elterjedt MatLab programban végzett pályatervezést egy korszerű virtualizációs alkalmazásban, a CoppeliaSim-ben jelenítem meg. A MatLab Robotics System Toolbox kiegészítője is tartalmaz egy egyszerűbb megjelenítőt, de a CoppeliaSim ennél magasabb minőséget képvisel. A megoldásban ki kell alakítani egy interfészt a két alkalmazás között és a megvalósított pályabejárásokat a virtuális valóság környezetben is meg kell jeleníteni.

Majd elérkezünk a 4. fejezethez, ami a megoldás tesztje során összegyűjtött adatok kiértékelését és bemutatását tartalmazza. Az 5. és 6. fejezetben a munka során levont következtetések rövid összegzése található magyar és angol nyelven.

2 Robot irányítás elméleti háttere

A feladat megoldása során kizárólag ipari robotkarokat feltételezünk, rotációs csuklókkal ellátva. Tekintsük át a robotkarok irányításának elméletét.

2.1 Robotok általános elmélete

Rögzített talpú, nyíltláncú robotokat modellezek. Szabadságfokok tekintetében, általánosan elmondható, hogy a legtöbb példa 6 szabadságfokú robotra értendő, de ettől lehetnek eltérések.

Ismerkedjünk meg néhány elengedhetetlen definícióval, melyekre a későbbiek megértése kapcsán szükségünk lesz. Egy robotkart egy **merev testekből álló kinematikus láncként** lehet modellezni, amelyek csuklókkal vannak összekötve. A csukló lehet rotációs és transzlációs. A kar minden csuklója egy újabb szabadságfokot jelent. A **szabadságfok** általános jelölése: DOF (degree of freedom), 6 szabadságfok esetén 6-DOF. A robotkar teljes szerkezetének mozgása az elemi részek mozgásainak egymáshoz gyakorolt hatásaiból áll össze.

A kinematikus lánc végén helyezkedik el az **end-effektor**, amivel manipulálni tudjuk a kívánt tárgyakat. Az end-effektornál meg kell említenünk a **szerszámközéppontot**, vagy másnéven TCP-t (Tool Center Point). Ezt a pontot próbáljuk eljuttatni bizonyos pozíciókba. Nem egy fizikai pont. A hozzá rögzített koordináta rendszert TCP koordinátarendszernek nevezzük.

Ahhoz, hogy a robotkarral a térben feladatokat tudjunk végezni meg kell tudnunk határozni az end-effektor pozícióját és orientációját. A **pozíció** az origó pozícióját, az **orientáció** pedig a bázisvektor irányát jelenti. A kettőt együttesen **póznak** nevezzük.
A pozíciót és orientációt a direkt kinematikai feladat elvégzésével tudjuk meghatározni.

A manipulátor szegmenseinek végpontjai, a csuklók, nem vehetnek fel tetszőleges x,y,z koordinátákat egymástól függetlenül, mivel a szegmenseknek fix méretük van. A szegmensek együttesen felvehető helyzetét hívjuk **konfigurációnak**. Egy robotkar által felvehető összes konfigurációt **konfigurációs térnek** nevezzük. Léteznek tiltott konfigurációk, amikor a robotkar ütközne önmagával, vagy a környezetével.

A tér feladatközpontú leírását, mely tartalmazza a lehetséges pozíciókat és orientációkat **munkatérnek** nevezzük. Ezen belül található a robotkar end-effektora által elérhető tartomány. Hatnál kevesebb szabadságfokkal rendelkező manipulátorok, nem vehetnek fel tetszőleges pozíciót és orientációt a térben. A nagyobb szabadságfokkal rendelkező robotok kinematikusan redundánsak lehetnek, ami azt jeleneti, hogy egy adott end-effektor pozíciót több konfigurációban is fel tudnak venni.

Descartes-koordinátarendszeres leírás
Jellemzői:

- Itt határozhatjuk meg a TCP helyzetét és orientációját.
- Euklidészi térben értelmezhetjük az erőket és nyomatékokat.
- Itt vizsgálhatjuk az esetleges ütközéseket és elkerülendő akadályokat.
- A felsorolt lehetőségeknek megjelenhetnek geometria vagy műszaki korlátai. Például: a robotkar nem ér el egy geometriai pontot, vagy nem tudja elérni a kívánt sebességet.

Csukló tér
A robot helyzetét csuklóinak állapotával írjuk le.
Jellemzői:

- Ebben a megközelítésben beszélhetünk a jelenlegi és kívánt csuklóváltozókról.

A robotkar bizonyos irányokba már nem tud tovább mozogni, például ha minden eleme függőleges irányban áll, akkor nem tudod felfelé haladni. Ezt nevezzük **szinguláris helyzetnek**.

2.1.1 Robotok modellezési szintjei

A robot modellezésben három szintet különböztethetünk meg:

- Geometriai szint
- Kinematikai szint
- Dinamikai szint

A feladatoknak két formája van: direkt és inverz. Ezeket külön definiáljuk a 3 szinten:

Direkt geometriai feladat definíciója
Adott csuklóváltozók esetén megadja a robot TCP-jéhez rendelt koordinátarendszernek a koordinátáit a világkoordinátarendszerben.

Inverz geometriai feladat definíciója
Megadja, hogy milyen csuklóváltozó értékekkel érhető el az end-effektor megadott pozíciója és orientációja.

Direkt kinematikai feladat definíciója
Megadja, hogy csuklóváltozókkal leírt kis mozgások (sebességek) milyen end-effektor sebességeket eredményeznek.

Inverz kinematikai feladat definíciója
Megadja, hogy kisméretű end-effektor mozgáshoz milyen csukló mozgások szükségesek. Az inverz kinematikai feladat komplexebb, mint a direkt kinematikai, mivel ezt nem egy analitikus kifejezés adja meg. Adott tulajdonságokkal rendelkező ipari robotok esetén lényegesen egyszerűbb meghatározni az inverz problémát, de itt több eredmény lehetséges, és nem mindig egyértelmű, hogy melyik lesz az optimális mozgással járó.

Direkt dinamikai feladat definíciója
Megadja, hogy adott terhelések és csuklónyomatékok mellett, milyen csuklógyorsulások következnek be.

Inverz dinamikai feladat definíciója
Megadja, hogy adott csuklógyorsulásokhoz, sebességekhez és pályákhoz mekkora csuklónyomatékra van szükség.

2.1.2 Jacobi-mátrix

A pályatervezéshez a fent felsoroltak közül az Inverz kinematikai feladat megoldására lesz szükségünk, melyhez meg kell ismernünk a Jacobi-mátrix fogalmát.

A TCP pont sebessége és a szögsebessége lineárisan függ a csuklósebességektől:

$$v_{TCP} = \sum_{i=l}^{m} J_{vi}\big(\mathrm{q(t)}\big) q_l(\mathrm{t}) = \mathrm{J}_v(q(t)) \cdot q(t)$$
$$\omega(\mathrm{t}) = \sum_{i=l}^{m} J_{\omega i}\big(\mathrm{q(t)}\big) q_l(\mathrm{t}) = \mathrm{J}_\omega(q(t)) \cdot q(t)$$

A robotkar munkapontjának sebességállapota:

$$\begin{bmatrix} v_{TCP} \\ \omega \end{bmatrix} = \underbrace{\begin{bmatrix} J_v(q) \\ J_\omega(q) \end{bmatrix}}_{J(q_1,\dots,q_m)} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_m \end{bmatrix}$$

A fenti egyenlet $J(q_1,q_2,\dots,q_m)$ mátrixát nevezzük Jacobi-mátrixnak.

A Jacobi-mátrix tehát megadja a sebességek, szögsebességek és a csuklósebességek közötti kapcsolatot. Ha fel tudjuk írni a Jacobi-mátrixot, akkor megkapjuk a direkt kinematikai feladat megoldását.
Inverz kinematikai feladat, megoldásakor kívánt sebességek és szögsebességek eléréséhez szükséges csuklósebességeket úgy kaphatjuk meg, hogy megoldjuk a direkt kinematikai feladatban adott lineáris egyenletrendszert, 6-DOF robot esetén a Jacobi-mátrixot invertáltjuk:

$$\dot{q} = J(q_1,\dots,q_m)^{-1} \cdot \begin{bmatrix} v_{TCP} \\ \omega \end{bmatrix}.$$

Előállhatnak olyan esetek, hogy a mátrix nem invertálható, azaz

- a mátrix kvadratikus, de szinguláris helyzetbe került a kar, emiatt lecsökkent a mátrix rangja,
- a mátrix nem kvadratikus, mert redundáns a kar.

Ezekben az esetekben alkalmazhatjuk a Moore-Penrose-féle pszeudoinverzet:

$$\dot{q} = \underbrace{J^T(JJ^T)^{-1}}_{=J^+} \begin{bmatrix} v_{TCP} \\ \omega \end{bmatrix},$$

vagy használhatjuk a Jacobi-mátrix transzponáltját:

$$\dot{q} = J^T \begin{bmatrix} v_{TCP} \\ \omega \end{bmatrix}.$$[1]

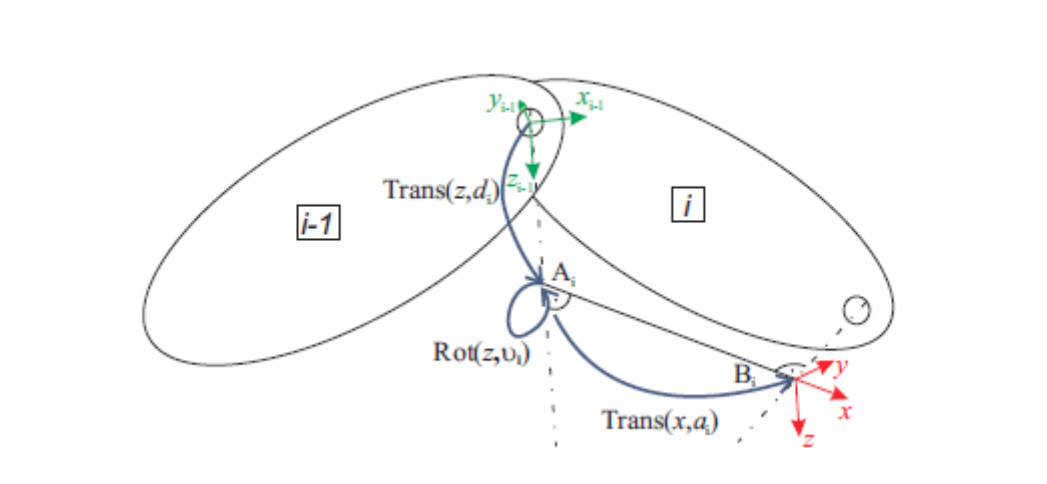
2.1.3 Kondíciószám

Ha egy mátrix determinánsa 0, akkor a mátrix szinguláris. Ekkor az analitikus megközelítés szerint egyenletrendszernek nincs véges megoldása. Numerikusan viszont, nem a determináns, hanem a kondíciószám határozza meg, hogy van-e véges megoldás és mennyire nagy. Ha a kondíciószám kiugróan nagy, a mátrixot rosszul kondícionáltnak nevezzük, ilyenkor a mátrix szingularitásközelében van. Ha a robotkar a pályatervezés során szinguláritás közelébe kerül, akkor a pályatervezés instabillá válik.
A kondíciószám matematikai definíciója: Az A mátrix kondíciószáma megegyezik a legnagyobb és a legkisebb szinguláris értékének hányadosával.

$$cond(A) = \|A\| \cdot \|A^{-1}\| = \frac{\sigma_1}{\sigma_n}$$

2.1.4 Denavit-Hartenberg konvenció

Az 1955-ben publikált eljárás megmutatta, hogy csuklótengelyekkel összekötött merev testek esetén egy szegmenshez rögzített koordinátarendszert két forgó és két haladó mozgással átvihetünk a szomszédos szegmenshez rögzített koordinátarendszerbe, és ezeknek a transzformációknak a paramétereivel egy robotkar geometriája jellemezhető. Tekintsük át egy példán keresztül a transzformáció működését



1. ábra – Koordinátarendszer átvitele Denavit-Hartenberg konvenció alapján

1. Legyenek a tengelyekre merőleges szakasz talppontjai A_i és B_i
2. Az origó eltolása A_i pontba: Trans(z, d_i)
3. Az x tengely elforgatása A_iB_i irányba: Rot(z, ϕ_i)
4. Az origó eltolása B_i pontba: Trans(x, a_i)
5. Alpha szöggel elforgatás az x tengely körül.

A Denavit-Hartenberg paraméterek alapján bármelyik robotkart definiálni tudjuk. Például a feladatban használt UR5 robotkar DH paraméterei:

Csukló	Típus	a	α	d	θ	Offset
1	Rotációs	0.00000	π/2	0.089159	q1	0.00
2	Rotációs	-0.42500	0.00	0.00000	q2	-π/2
3	Rotációs	-0.39225	0.00	0.00000	q3	0.00
4	Rotációs	0.00000	π/2	0.10915	q4	-π/2
5	Rotációs	0.00000	-π/2	0.09465	q5	0.00
6	Rotációs	0.00000	0.00	0.0823	q6	0.00

2.2 Pályatervezés elmélete

A robotkarok pályatervezése az egyik legalapvetőbb robotikai feladat, hiszen az autonóm mozgatásban rejlik a hasznuk. Ennek ellenére nagyon összetett feladat és nincs egy univerzális megoldás rá. Az alapvető elvárás egy robottal szemben, hogy egy kezdeti end-effektor pozícióból el tudjon jutni egy cél pozícióba. Ezt a feladatot még tovább bonyolítja, ha olyan elvárásunk is van, hogy egy adott útvonalon végig haladva érje el a cél pozíciót. [2]
Soros manipulátorok pályatervezésekor beszélhetünk csuklókoordinátás tervezésről, illetve világkoordinátás tervezésről. Ebből adódóan a feladat elvégezhető csuklókoordinátában vagy világkoordinátákban.
A pályatervezés célja lehet: az end-effektor mozgását meghatározó pontok pozíciójának és orientációjának megadása vagy adott pontok közötti pályameghatározás. [3]
A megtervezett pálya bejárásához az inverz kinematikai feladatot meg kell oldani az összes pózra, így megkapjuk a csuklóváltozó vektorokat minden lépésre.

Closed-loop inverse kinematics (CLIK)
Az eljárás az pályakövetési hibák kiküszöbölésére szolgáló eljárás.
A k-adik ütemre kiszámoljuk az e(k) hibát oly módon, hogy a pozíció és orientáció referencia vektorokból kivonjuk a mért párjukat:

$$e(k) = \begin{pmatrix} P_{ref} \\ \sigma_{ref} \end{pmatrix} - \begin{pmatrix} P_{mért} \\ \sigma_{mért} \end{pmatrix},$$

majd a Δq-t, a csuklóváltozók változását megkapjuk, ha az adott q(k) csuklóváltozó vektorra kiszámoljuk az inverz Jacobi-mátrixot és ezt megszorozzuk az imént megkapott e(k) hiba értékével:

$$\Delta q(k) = J^{-1}\big(q(k)\big) * e(k).$$

Ezután a következő lépés korrigált q(k+1) csuklóváltozó vektorrát úgy tudjuk kiszámolni, hogy a q(k)-hoz hozzáadjuk az imént megkapott Δq-t megszorozva a Ts mintavételi idővel:

$$q(k+1) = q(k) + Ts * \Delta q(k).$$

2.3 Felhasznált szoftver eszközök

2.3.1 Matlab Robotics System Toolbox

A MatLab alkalmazás fogja elvégezni a létrehozott rendszerünk pályatervezéséhez szükséges számításokat. A Peter Corke neve alatt futó Robotics System Toolbox számos, a robotikában szükséges eljárást tesz könnyebbé a MatLab rendszerében. Az ingyenesen elérhető kiegészítő csomag segítséget nyújthat a pályatervezésben, pályakövetésben illetve az inverz kinematikai feladatok megoldásában is.
A feladat megoldásához a 10.2-es verziót használom. A csomag lehetővé teszi a robot alkalmazások fejlesztését. Interaktív módon tesztelhetjük az alkalmazásokat szimulációkkal, illetve valós robotokon minimális kódmódosításokkal. C++ kódokat is képes generálni a MatLab-ból. Kompatibilis a ROS (Robot Operation System) operációs rendszerrel, így a ROS-ban készült napló fájlokat is elemezni tudja.

2.3.1.1 Felhasznált matlab függvények a megoldás során

jacob0
A jacob0 függvénnyel megtudhatjuk egy robotkar paramtéreknént megadott csuklóváltozókhoz tartozó Jacobi-mátrixát.

fkine
Az fkine függvény a direkt kinematikai (forward kinematics) feladat megoldását adja a robotkar paraméterként megadott csuklóváltozóihoz.

ikine
Az ikine függvény az inverz kinematikai (inverse kinematics) feladat megoldását adja a robotkar paraméterként megadott pózhoz.

tr2angvec
A tr2angvec függvény a rotációs mátrixból számolja ki a szögsebesség vektorát.

det
A det függvény egy mátrix determinánsát adja meg. Ez az érték szingularitás vizsgálatnál szükséges.

cond
A cond függvény egy mátrix kondíciószámait adja meg. A szinguláris hoz közelítő értékek vizsgálatához szükséges.

2.3.2 CoppeliaSim (V-REP)

A CoppeliaSim, vagy régebbi elnevezésén V-REP, egy általános célú robot szimulációs alkalmazás beépített fejlesztő környezettel.

A CoppeliaSim néhány főbb erősége:

- Keretrendszere hatékonyan egyesít külső és belső, a robotikában elterjedt könyvtárakat. Direkt- és inverz kinematikai megoldásokat, legismertebb robot típusokat, szimulációs motorokat és egyéb hasznos eszközöket is magában foglal.
- Egy jól felépített, saját script nyelvet biztosít a felhasználók számára, melyet Lua névre kereszteltek.
- Döntően nyílt forráskódú, és rengetek API-t tartalmaz. Ezt a tulajdonságát a feladat megoldásban, a MatLab kapcsolat kialakításakor tudjuk kamatoztatni.

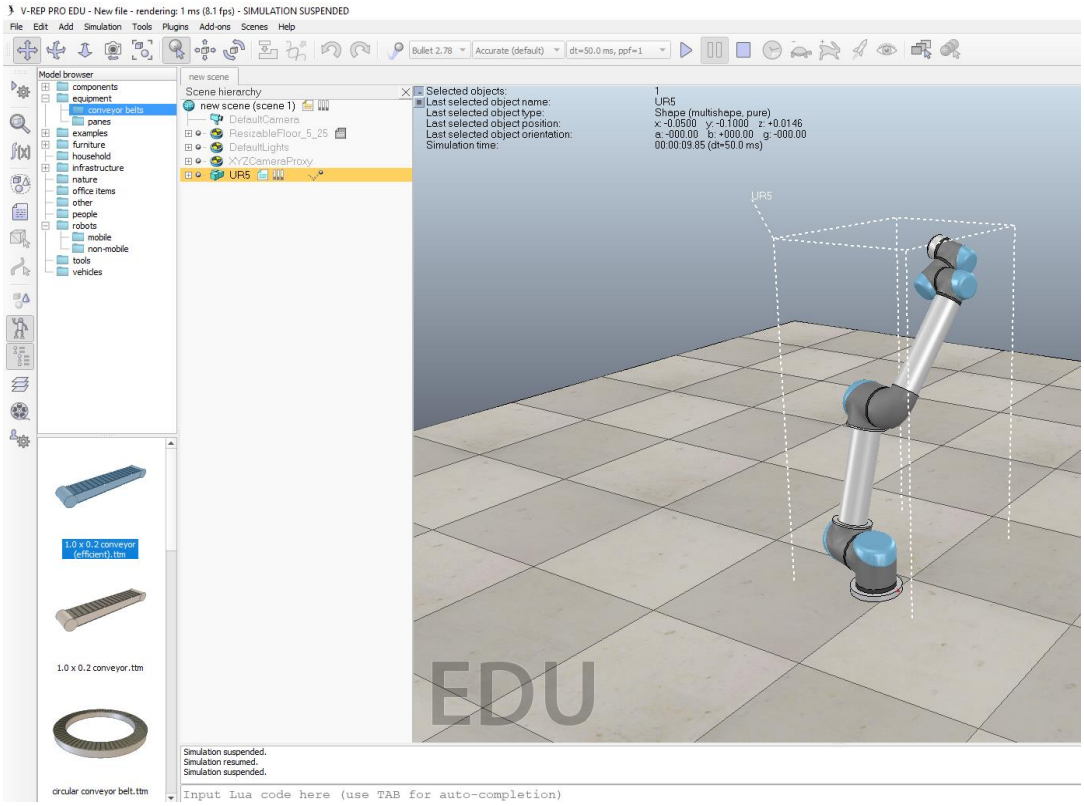
Egy egész robot munkavégzési folyamat, úgynevezett “jelenet” (scene), modellezhető a CoppeliaSim segítségével. Egy ilyen jelenet három főbb részből épül fel:

1 . Jelenet objektumai
Ide tartozik minden munkatérben található elem. Nem csak a robotkar és megmunkált tárgyak, hanem minden egyéb környezeti elem is.

2. Számolási modulok
Ide tartozik például a direkt kinematikai és inverz kinematikai műveletek megoldása. Ezt a részt nem fogom használni, mivel ezeket a számolások könnyebben elvégezhetők és elemezhetők MatLab környezetben.

3. Irányítási mechanizmusok
A különböző fizikai hatások megjelenítése a robotokon.

A CoppeliaSim alkalmazás kezelőfelületének felépítése a 2. ábrán látható.



2. ábra – CoppeliaSim alkalmazás kezelőfelületének felépítése egy UR5-ös robotkarral

Felhasznált CoppeliaSim függvények a megoldás során

simExtRemoteApiStart

A simExtRemoteApiStart függvénnyel lehet az adott CoppeliaSim jelentből kinyitni egy portot az API-s elérés számára.

simxStart

Ezzel a függvénnyel lehet a MatLab oldaláról csatlakozni a CoppeliaSim szimulációhoz, ha ott előzőleg a simExtRemoteApiStart függvénnyel kinyitották a portot

simxGetObjectHandle

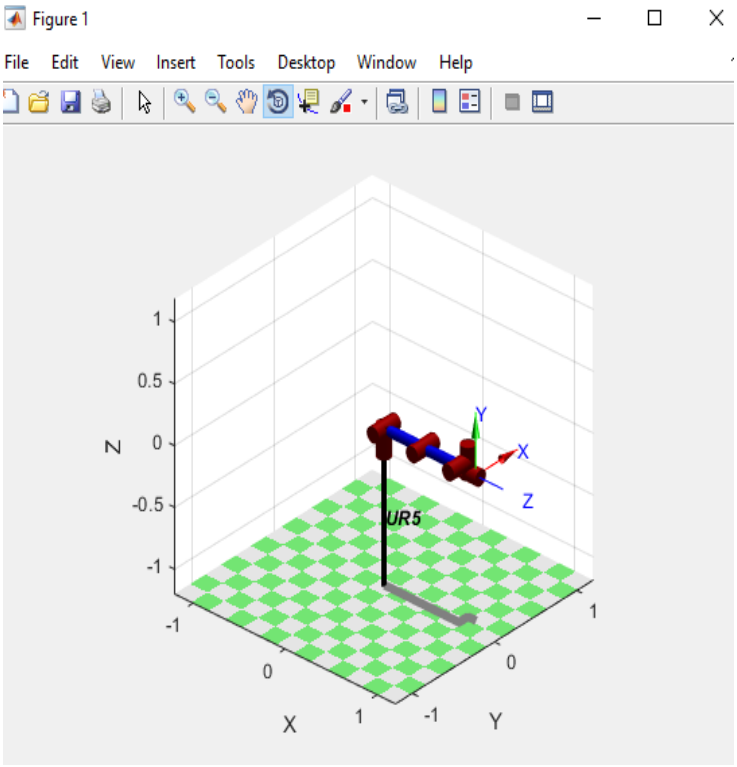
A simxGetObjectHandle függvénnyel megszólíthatjuk a szimuláció egy objektumát, hogy irányításunk alá helyezzük. A feladat megoldásában a szimulált robotkar csuklóinál lett felhasználva.

simxSetJointTargetPosition

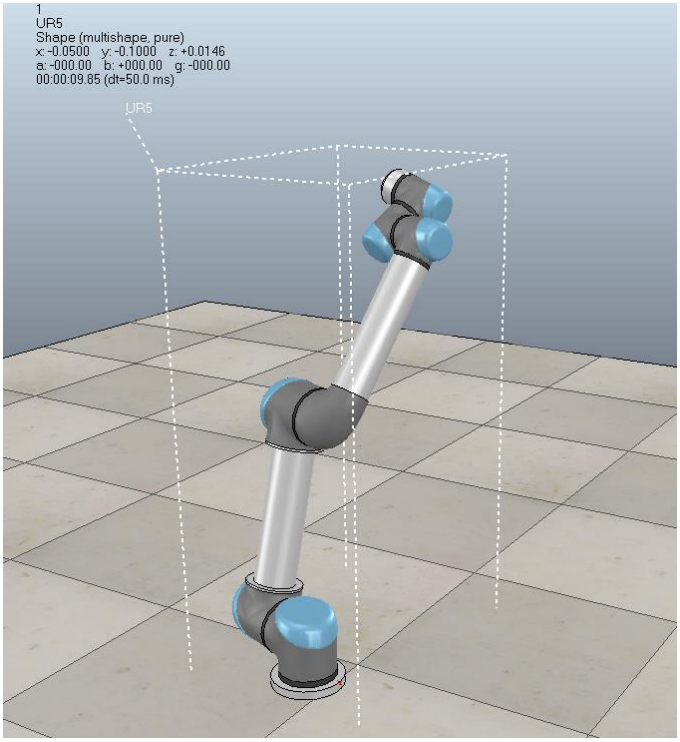
A simxSetJointTargetPosition függvénnyel megadhatjuk a robotkar csuklónak, hogy mekkora szöget zárjanak be. A feladat megoldásában ezzel a függvénnyel adtam át csuklóváltozókat a szimuláció számára.[4]

2.3.3 MatLab és CoppeliaSim megjelenítés összehasonlítása

A MatLab Robotics System Toolbox tartalmaz grafikus megjelenítést, de az így kapott kép nagyon leegyszerűsített, míg a CoppeliaSim-ben látható megjelenítés élethű mása a robotkarnak. Az alábbi ábrákon egy UR5-ös ipari robotot láthatunk a két rendszerben.



3. ábra – MatLab Robotics System Toolbox alkalmazásban az UR5 robotkar szimulációja

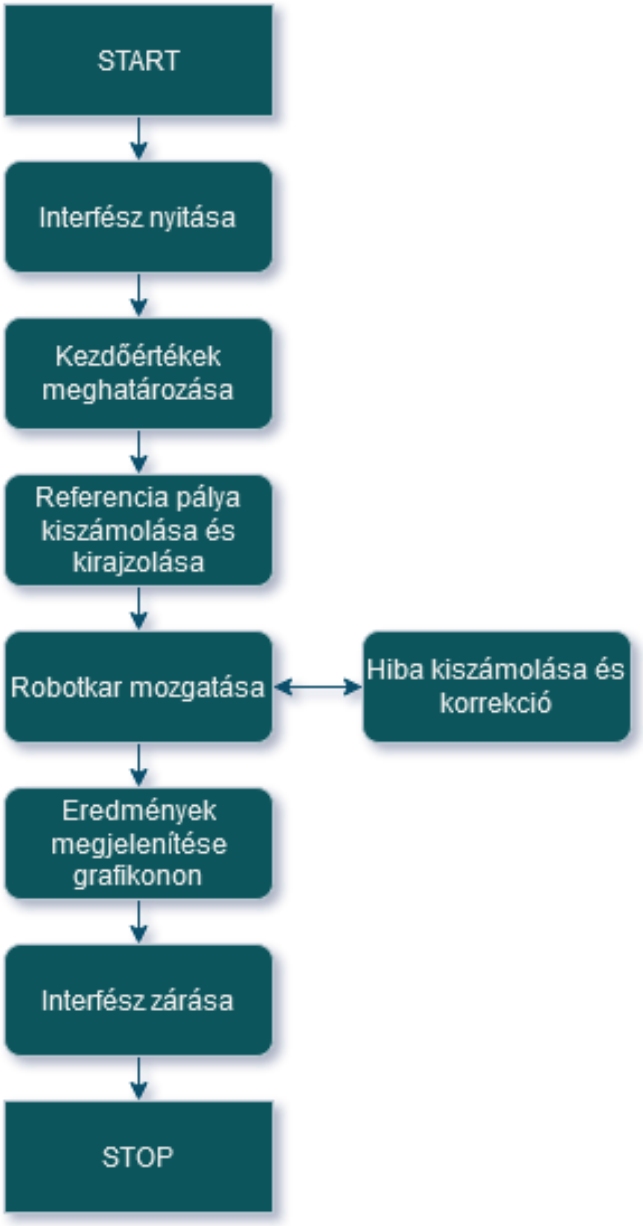


4. ábra – CoppeliaSim alkalmazásban az UR5 robotkar szimulációja

A 3. és 4. ábra alapján egyértelműen láthatjuk, hogy a CoppeliaSim virtualizációja magasabb szintet képvisel. Ezzel a megjelenítéssel pontosabb tervezéseket lehet véghezvinni. A valós munkateret is nagy közelítéssel megvalósíthatjuk, így nem lesznek fekete foltok a tervezés során. Az így kapott eredményeket fel lehet használni pályázati célokra. Az ügyfelek is láthatják, hogy pontosan mit fog csinálni a robot, ha az alkalmazást átültetjük a valós berendezésre.

3 Rendszerterv, rendszerintegráció

3.1 A feladat megvalósítás logikai egységei



5. ábra – Feladat megvalósítás logikai részei

Ahogy az 5. ábrán látható, a tervezéskor a rendszert több logikai szintre osztottam. A következő fejezetekben az ábrán megjelenő részfeladatok megoldását fejtem ki részletesen.

3.2 START-STOP

A létrehozott rendszer a MatLab és a CoppeliaSim alkalmazásokon belül fut. Az indításhoz meg kell nyitni a .m és a .ttt kiterjesztésű állományokat. Ezután a CoppeliaSim-en belül el kell indítani a szimulációt a háromszög gombbal. Ha ez megtörtént a MatLab-ban is el kell indítani a futtatást a háromszög gombbal. Ha nem fut a szimuláció az előbbiben, nem fog elindulni magában a MatLab-os számolás. A két megjelentőben egyszerre mozognak a robotkarok, hogy össze lehessen hasonlítani őket. Ha végig értek a pályán, megjelennek az adatok a grafikonokon, amiről le lehet olvasni a teszthez szükséges információkat.

Ha meg akarjuk állítani a rendszert, csak be kell zárni az ablakokat. Itt a sorrend már nem számít. Soha ne mentsük el az esetleges változásokat, mert ezek befolyásolhatják a későbbi futási eredményeket.

3.3 Interfész kialakítása az alkalmazások között

A MatLab és a CoppeliaSim kapcsolatához a CoppeliaSim beépített Remote API-t használom. A CoppeliaSim telepítése után, a program könyvtárában megtalálhatóak a szükséges fájlok:

- remoteApiProto.m
- remApi.m
- remoteApi.dll

Ügyelni kell rá, hogy a megfelelő architektúrához tartozó dll-t használjuk. Egy 64 bites MatLab nem fog jól működni a 32 bites remoteApi.dll-lel.

3.3.1 Interfész nyitása

A CoppeliaSim-ben futtatnunk kell az API-t működésbe hozó parancsot: **simExtRemoteApiStart(19999),** amiből a szám a kinyitott portot jelenti. Ez a függvény elhelyezhető az objektumokhoz tartozó kódok között, de manuálisan is beírható indítás

után a parancssorba. Ezzel a CoppeliaSim készen áll rá, hogy fogadjon más programoktól inputokat.
A feladat megvalósításakor elhelyeztem a függvényt a jelenthez tartozó LUA script fájlban, így minden szimuláció indításakor automatikusan megnyitja a portot a Matlab számára.

A MatLab-ban a következő parancsokat kellett elhelyezni a forráskódban, hogy létrejöjjön a kapcsolat:
vrep=remApi('remoteApi');
Ezzel a vrep változóval definiáltam, hogy a CoppeliaSim API-ját fogom használni.

vrep.simxFinish(-1);
Ez a függvény -1 érték esetén minden futó kommunikációs szálat megállít, felkészítve a CoppeliaSim szimulációs jelenetét a MatLab irányából érkező kommunikációra.

clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
Ezzel a függvénnyel megadható, hogy milyen IP címen és porton érhető el a CoppeliaSim. Esetünkben ez a localhost és a SimExtRemoteApiStart függvényben megadott 19999-es port.

3.3.2 Interfész zárása

A szimuláció végén újra meg kell állítani a futó kommunikációs szálakat a vrep.simxFinish(-1) függvénnyel, majd törölni kell a megnyitott API hívást a vrep.delete() paranccsal.

3.4 Kezdőértékek meghatározása

A MatLab Robotics System Toolbox tartalmaz előre definiált UR5-ös robotkart, ezért a feladat megoldásánál ezt használtam fel. Ha más robotkarra szeretnénk elvégezni a pályatervezést és ez nem állna rendelkezésre a Toolboxban könnyen létrehozhatunk egy saját konfigurációt.

Például UR5 definiálása a robot geometriai adatainak (Denavit-Hartenberg paramétereinek) megadásával történik:
L1=Revolute('a',0,'alpha',pi/2,'d',0.08916);
L2=Revolute('a',-0.425,'alpha',0,'d',0);

```
L3=Revolute('a',-0.39226,'alpha',0,'d',0);
L4=Revolute('a',0,'alpha',pi/2,'d',0.10915);
L5=Revolute('a',0,'alpha',-pi/2,'d',0.09456);
L6=Revolute('a',0,'alpha',0,'d',0.0823);
ur5=SerialLink([L1 L2 L3 L4 L5 L6],'name','UR5')
```

A kezdőértékek meghatározásához az általam meghatározott kezdő csuklóváltozókra és az inicializált robotkarra el kell végezni a direkt kinematikai számítást. Ebből már kiszámítható a pozíció, majd a pozíció alapján megkapjuk a rotációs mátrixot. A rotációs mátrix segítségével előállíthatjuk a szögsebesség vektort.
Fenti számolt értékeken kívül megadjuk az erősítés mértékét, a lépések számát és nagyságát, illetve, ha kör vagy ellipszis pályát számolunk, akkor a sugarat.

Megadott kezdőértékek

q	[0.1 0.1 0.6 0 0.1 0]
G	10
tEnd	10
Ts	0.1
N	tEnd/Ts+1
radius	0.2

3.5 Referencia pálya kiszámolása és kirajzolása

A megoldás során három referenciapályát implementáltam. Egy egyenest, ami egy egyszerűbb pályatervezés, valamint egy kört és egy ellipszist, amelyek bonyolultabbak.

A pályatervezésnél a q kezdő csuklóváltozó vektorra a solver segítségével elvégezte a direkt kinematikai feladatot, így megkaptam a pozíciót és a rotációsmátrixot. A rotációs mátrix segítségével kiszámoltam az orientációt, ez végig állandó maradt a referencia pályán. Az orientáción kívül szükségem volt, az összes lépés pozíció értékeire. Ez mindig az aktuális követendő pályáknak megfelelő függvény értékének az imént kiszámolt pozícióhoz történő hozzáadásával oldottam meg. Az így kapott Pref referencia mátrixot kirajzoltattam 3 dimenzióban a robotkar mellé, hogy látni lehessen a kijelölt pályát.

Referencia pályák paraméterezése

A referencia pályák paraméterezéséhez először kiszámoltam a q kezdő csuklóváltozó vektor alapján a direkt kinematikai feladat megoldásával a P pozíciót és az R rotációs mátrixot. Az R-ből meg tudom határozni a σ orientációt. A σ-t végig állandónak tekintem. A referencia pozíciókat pedig a következő módon határoztam meg az eltérő pályáknál.

Egyenes pálya paraméterezése:

Meghatározom, hogy mekkora eltérés akarok elérni x, y, és z irányba, majd ezt felosztom az előre meghatározott N-1 lépésszámú egységre (az első lépés a kiinduló pozíció, P_1=P) és k lépésben iterálva az egységeket hozzáadva megkapjuk a lépésekre eső pozíciókat:

$$P_{k+1} = P_k + \frac{[x, y, z]}{N-1}$$

Kör pálya paraméterezése:

A kör pálya paraméterezéséhez szükség van egy előre meghatározott s sugárra és a θ vektorra, amely tartalmazza az N lépésre osztott 2π egységet. Ezeken kívül kell még egy θ méretű m vektorra, amely csupa 1-est tartalmaz.
A kör pontjait megkapjuk:

$$P_{ref} = P + s * [m, \cos\theta, \sin\theta]$$

Ellipszis pálya paraméterezése:

Az ellipszis pálya paraméterezése megegyezik a kör pálya paraméterezésével, csak az egyik tengelyt megszorozzuk egy konstans értékkel:

$$P_{ref} = P + s * [m, \cos\theta, (\sin\theta * 0.5)]$$

3.6 Robotkar mozgatása és korrekció

A megoldás során a closed-loop inverse kinematic (CLIK) eljárást használtam fel a pályakövetési hiba kiküszöbölésére. Az alábbi lépésekben oldottam meg feladat kivitelezésekor.

Veszem a q kezdő csuklóváltozó vektort és elvégzem rá a direkt kinematikai feladatot a solver segítségével:

```
T = ur5.fkine(q(k,:));
```

Így már tudom a robotkar pózát, amiből megkapom pozíciót és a rotációs mátrixot:

```
Pmert(:,k) = T.t;
r = T.R;
```

A rotációs mátrix segítségével meg tudom határozni az orientációt:

```
[w,t] = tr2angvec(r);
szigma = w*t';
```

Így megvan a referencia és a mért pozíció és orientáció is az adott lépésre, így meg tudom határozni a hiba mértékét, ha kivonom őket egymásból:

```
error(:,k) = [Pref(:,k); szigmaref(:,k)] - [Pmert(:,k);szigma];
```

Ekkor kiszámolom a Jacobi-mátrixot az adott csuklóváltozókra:

```
J=ur5.jacob0(q(k,:));
```

és a Jacobi-mátrix inverzét megszorzom a kapott hibával, hogy megkapjam a delta q-t:

```
dq=inv(J)*error(:,k);
```

a delta q-t meg kell szorozni még a Ts mintavételi idővel és az erősítéssel és megkapjuk a korrigált csuklóváltozó vektort a következő lépésre:

```
dqts=Ts*dq;
q(k+1,:)=q(k,:)+G*reshape(dqts,[1,6])
```

Az egész folyamatot minden lépésre el kell végezni egy ciklusban.

A mozgatáshoz minden iterációban szimulálom az adott csuklóváltozókat a robotkarral:

```
ur5.plot(q(k,:))
```

3.6.1 Determináns vizsgálat

A megoldás vizsgálja, hogy az adott lépésben a Jacobi-mátrix szinguláris-e. Ha egy mátrix determinánsa 0, akkor szinguláris. Ebben az esetben a robotkar instabillá válna, így nem számol tovább az algoritmus, hanem leáll.

A vizsgálatra a MatLab beépített det() függvényét használtam fel, amire megvizsgáltam, hogy egyenlő-e nullával:

```
if det(J)==0
disp('A mátrix szinguláris')
```

Szinguláris Jacobi-mátrixnál nem folytatódik a számolás, mivel nincs értelme egy instabil, vagy elérhetetlen mozgást szimulálni.

3.6.2 Kondíciószám figyelés

Nem csak szinguláris Jacobi-mátrix esetén vállhat instabillá a robotkar, hanem abban az esetben is, ha nagyon megközelít egy szinguláris állapotot. A kondíciószám megegyezik a mátrix legnagyobb és legkisebb szinguláris értékének hányadosával. Ha a kondíciószám kiugróan alacsony vagy nagy értéke megmutatja, ha szinguláris érték közelébe kerül a robotkar.
A kondíciószám figyelésre a MatLab beépített cond() függvényét használtam. Ebben az esetben nem zártam ki a számolás folytatását, mert meg akartam vizsgálni a tesztek során a kondíciószám figyelés hatékonyságát. Eltároltam az értékeket egy vektorban és egy gráfon ábrázoltam a változásukat:

```
kond(k) = cond(J);
plot(z,kond)
```

3.7 Eredmények megjelenítése

A szimuláció során kiszámolt fontosabb adatokat grafikonon jelenítettem meg a könnyebb értelmezhetőség miatt.

3.7.1 Valódi pálya

A pályakövetés pontosságának megjelenítésére a korábban térben áttetsző színben felrajzolt referencia pályára más színnel rárajzoltam a valóban bejárt pályát így könnyen észlelhetők a különbségek.

3.7.2 Csuklóváltozók

A robotkar 6 csuklójának csuklóváltozóit külön grafikonokon jelenítem meg, így látható, hogy a szimuláció során mekkora mozgásokon estek át. A könnyebb értelmezhetőség kedvéért radiánból fokba váltottam az ábrán az értékeket.

3.7.3 Hiba

Hibáról akkor beszélhetünk, ha az end-effektor a tér valamelyik irányába eltér a kijelölt pályától. A hiba grafikonok emiatt az x, y és z irányokra külön-külön megjelennek az end-effektor referencia és valódi pályája közötti különbségek.

3.7.4 Kondíciószám

A grafikon kiugró értékei megmutatják, ha a kar közel kerül egy szinguláris értékhez. Ha instabilitást tapasztalunk a robotkar mozgása során, akkor ellenőrizni tudjuk, hogy egy szingulárisérték megközelítése okozza-e a problémát.

3.7.5 Mozgás megjelenítése CoppeliaSim alkalmazásban

A 3.2.1-es fejezetben taglalt módon alakítjuk ki az interfész kapcsolatot. Ha ez sikeresen megtörtént, a virtuális térben való mozgatáshoz már csak definiálnunk kell, hogy a megvalósított pályatervező algoritmus kimenete mely csuklókra vonatkozik a példa roboton. A teszteléshez egy UR5-ös robotot használok mintának.
Annak érdekében, hogy a robotkar által bejárt pálya jobban követhető legyen, egy Graph elemmel 3 dimenzióban rajzolva egy színes csíkkal jelzem ezt a jeleneten. A Graph elemet az alkalmazás egy pufferben tárolja. Lehetőség van akkora puffert beállítani, hogy sokáig látható legyen a pálya, viszont így a kezdeti mozgások is rajta maradnak az ábrán, ami zavaró lehet. A puffer méretét úgy választottam meg, hogy lehetőleg csak a kirajzolt alakzat maradjon a jeleneten, a szimuláció végén.

4 Tesztelés

A rendszer működtetéséhez telepíteni kell mindkét alkalmazást. Az általam használt verziók, amelyekkel biztosan jól működik a mellékelt forráskód:

MatLab R2021b 9.11

CoppeliaSim EDU 4.3.0

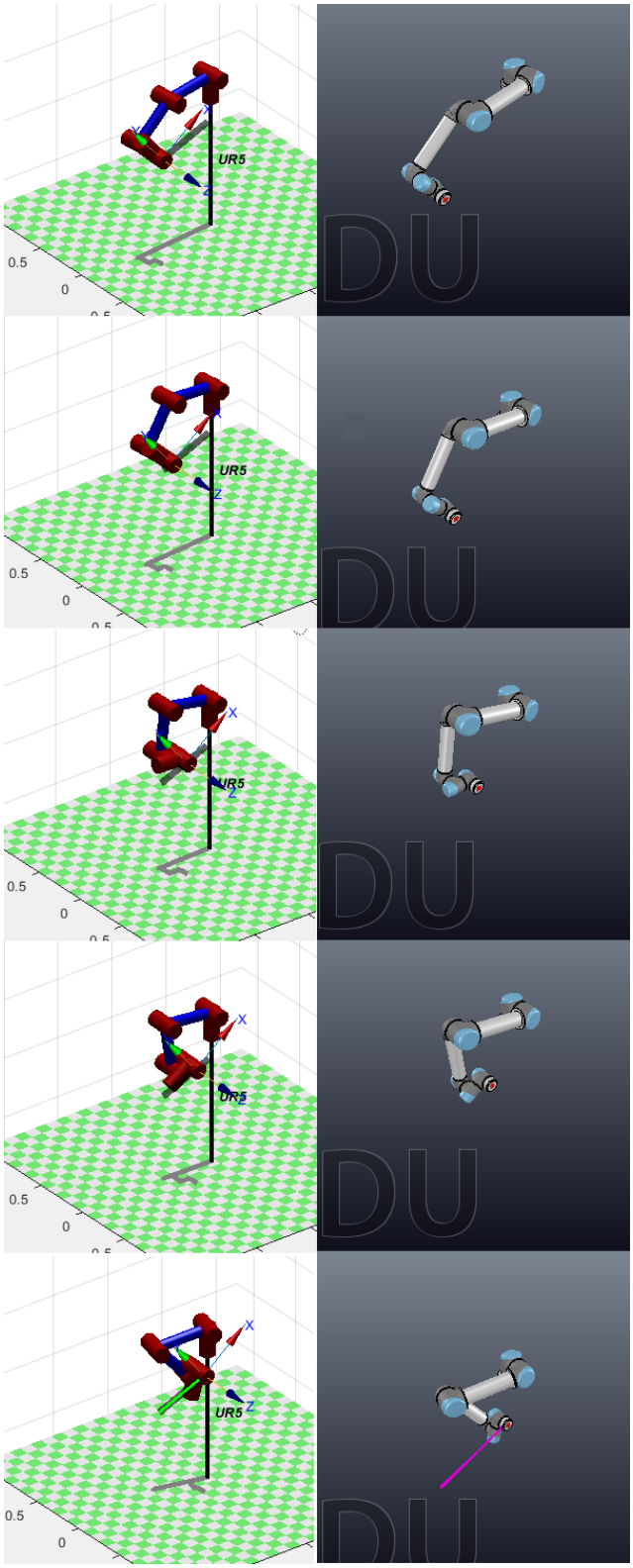
A feladat megoldásában létrehoztam három különböző típusú pályát, a robotkar viselkedését ezekkel a pályákkal teszteltem. A teszthez szükséges értékeket grafikonokon ábrázoltam, hogy könnyebben értelmezhetők legyenek. Ezen kívül összehasonlításként megjelenítem a virtuális valóság szimulációban születő eredményeket, hogy valóban követi-e a MatLab alkalmazásban kiszámolt és megjelenített pályát. A következő alfejezetekben bemutatok négy tesztesetet, amelyeken keresztül megvizsgáltam, hogy helyesen működik-e a rendszer és milyen következéseket lehet levonni az általam létrehozott rendszerből. Mindegyik alfejezet három nagy részből áll, amelyek bemutatják a pályakövetés vizuális megjelenítését, a csuklóváltozók értékeit, valamit a pályakövetési hiba mértékét és az ebből levonható következtetéseket.

4.1 Egyenes pálya

Az egyenes pálya a legegyszerűbb pálya forma. A beállítás során azt az esetet próbáltam elkerülni, amikor a választott keret valamely tengelyére párhuzamos az egyenes. Emiatt a referencia pályát úgy állítottam be, hogy mindhárom irányba legyen elmozdulás, hogy megvizsgálhassuk a hiba mértékét mindhárom tengelyen.

4.1.1 Pályakövetés

A 6. ábrán látható, hogy a robotkar kis hibával megfelelően követi a számára előírt egyenes pályát.

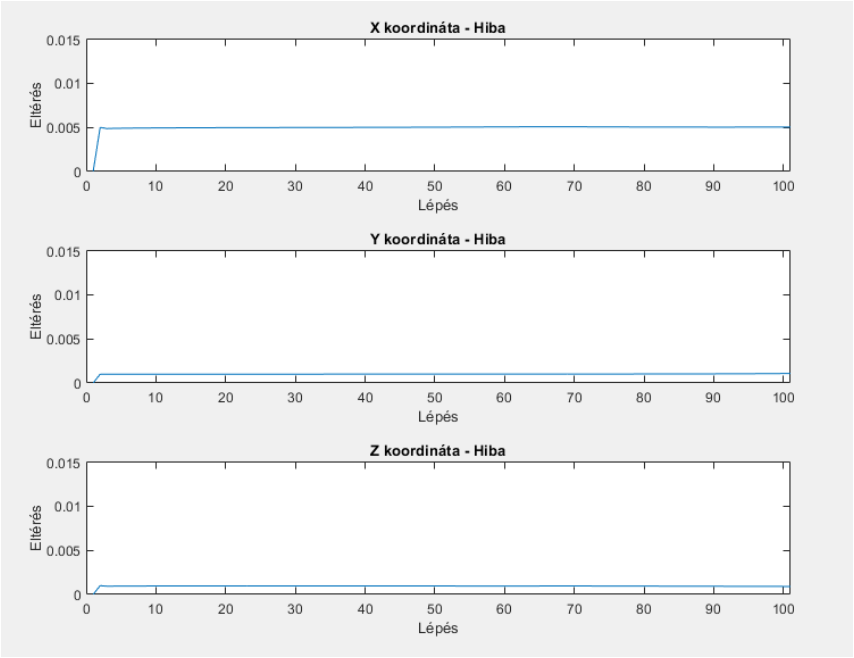


6. ábra – Egyenes pálya pályakövetés szimulációja UR5-ös robotkarral (bal oldalon MatLab, jobb oldalon CoppeliaSim alkalmazásban)

A virtuális valóság szimulációban is látható a 6. ábrán, ahogy a robotkar ugyanazt az előírt pályát teljesítette, mint a MatLab-ban. Érdemes megnézni, hogy nincsenek kezdeti ingások sem. Mivel a robotkar közvetlenül a pályáról indul, ezért már az első lépéseknél sem kell nagyon korrigálni.

4.1.2 Hiba elemzése

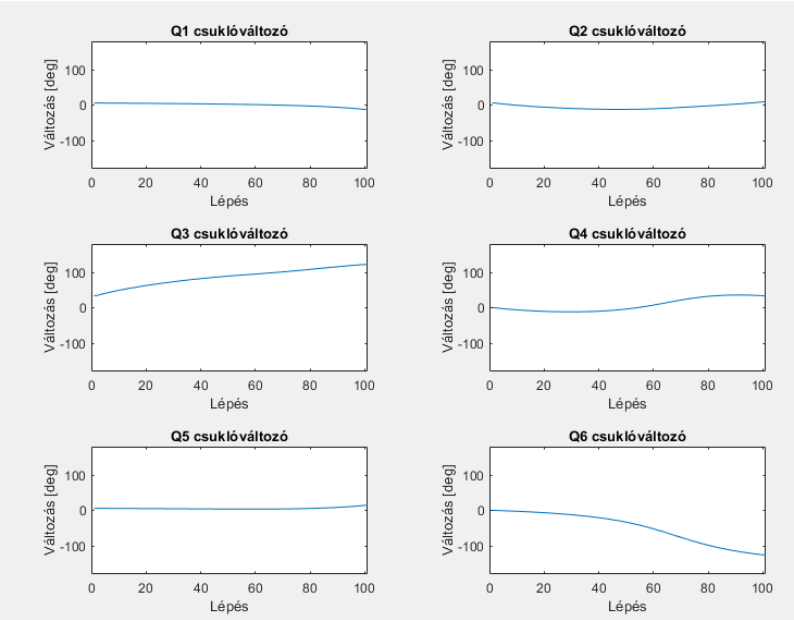
X irányba van a legnagyobb elmozdulás emiatt látható a 7. ábrán, hogy az X koordinátánál van a legnagyobb eltérés, viszont ez is nagyon alacsony érték.
Az egész mozgás során nagyon egyenletes az érték, ami annak köszönhető, hogy az egyszerű pálya miatt nincsenek nagy átforgások a csuklókban.



7. ábra – Egyenes pálya pályakövetés szimulációjának hiba értékei

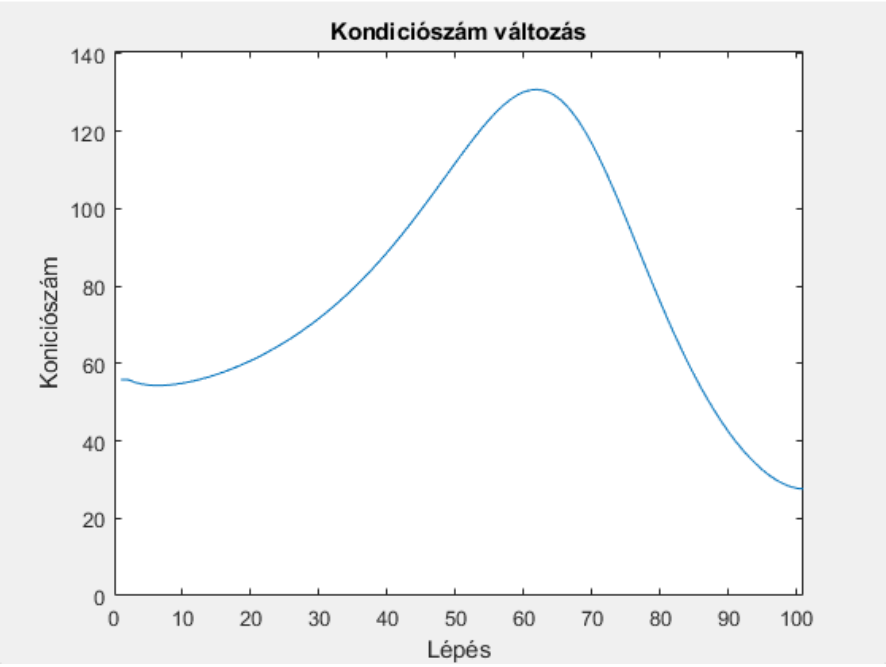
4.1.3 Csuklóváltozók elemzése

A 8. ábrán a Q3-as csuklónál láthatunk egy 120 fokos elmozdulást, ez a 6. ábrán megfigyelhető, ahogy a 3. csukló maga alá húzza az end-effektort. Ezen kívül csak a 6. csuklónál látható 120 fokos eltérés, ami a kar utolsó elemét forgatja az egyenes követési irányába. A többi négy csuklónál, csak kis eltérések figyelhetők meg, ezért a robotkar pályája nagyon simának tekinthető. Ez a fenti hibagrafikonon (7. ábrán) abban mutatkozik meg, hogy egyenletesen alacsony a hiba értéke egész idő alatt.



8. ábra – Egyenes pálya pályakövetés szimulációjának csuklóváltozó értékei

A kondíciószám végig elfogadható értéken marad, a robot nem közelít meg szinguláris helyzetet. A pályakövetés is végig stabil állapotban történik, nincsenek megingások.



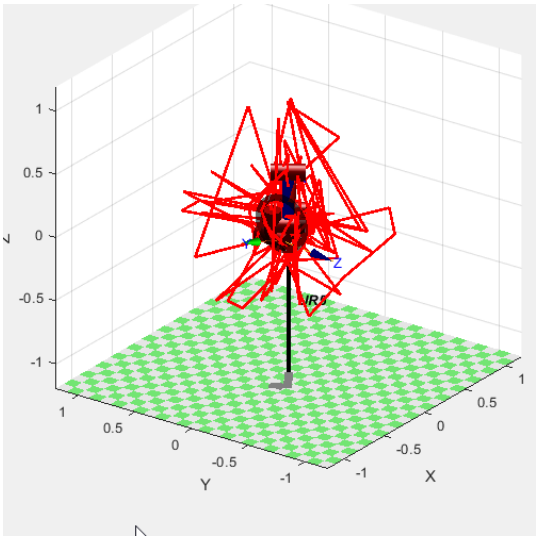
9. ábra – Egyenes pálya pályakövetés szimulációjának kondíciószámai

4.2 Kör pálya 1. eset - instabil

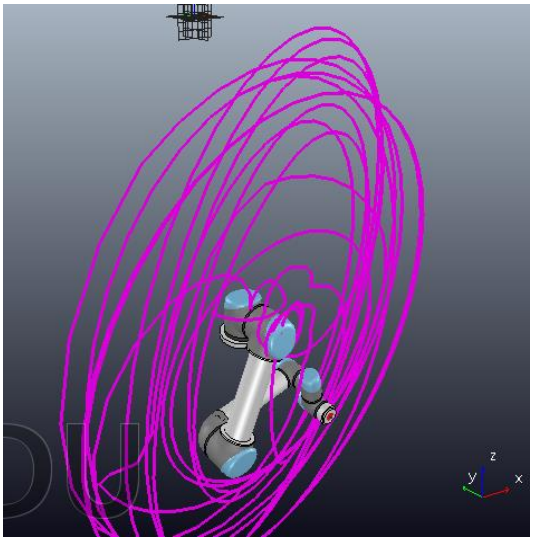
A kör pálya komplexebb feladat a robotkar számára. Ebben a beállított konfigurációban a robotkar instabillá vált, ami ellentétes a céljainkkal, de jó példa a vizsgálat szempontjából.

4.2.1 Pályakövetés

Az instabil kar a MatLab szimulációban össze-vissza csapkod, míg a CoppeliaSim-ben pörögni kezd a 2. csukló mentén.



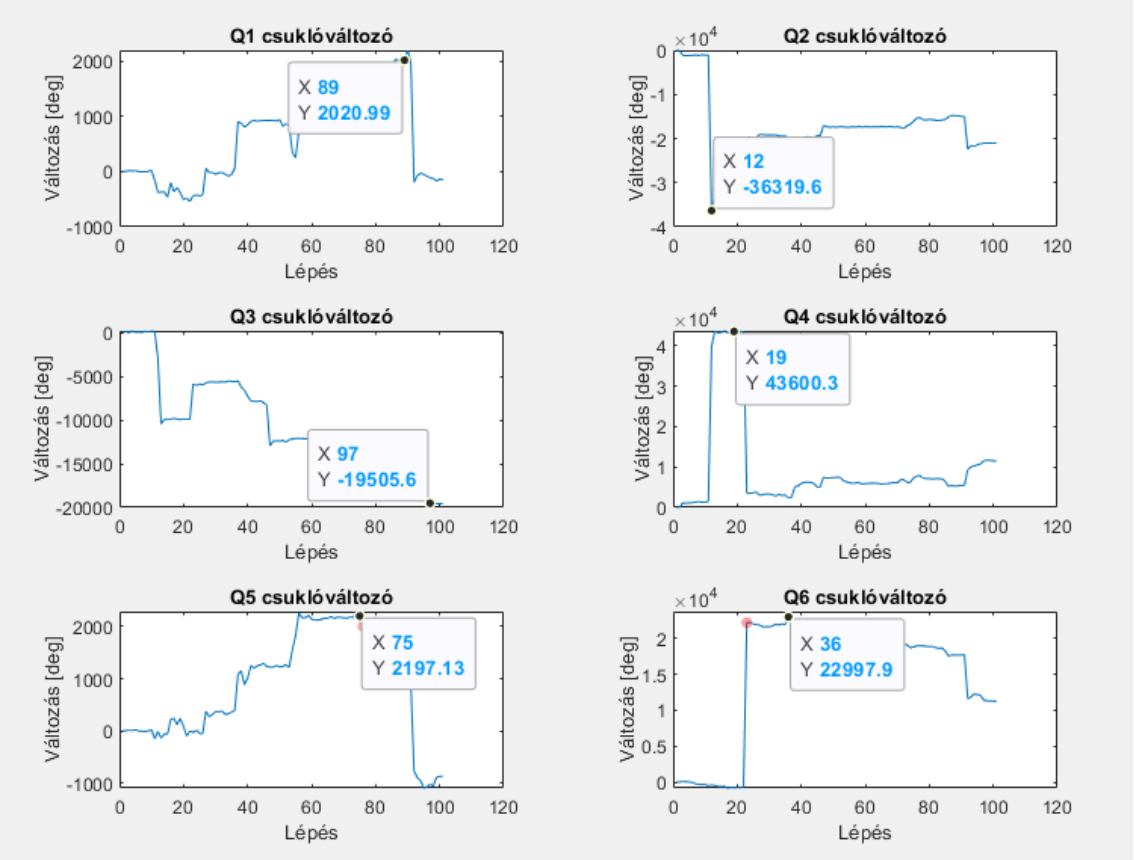
10. ábra – Kör pálya 1.eset pályakövetés szimulációja MatLab-ban instabil állapotban



11. ábra Kör pálya 1. eset pályakövetés szimulációja CoppeliaSim-ben instabil állapotban

4.2.2 Csuklóváltozók elemzése

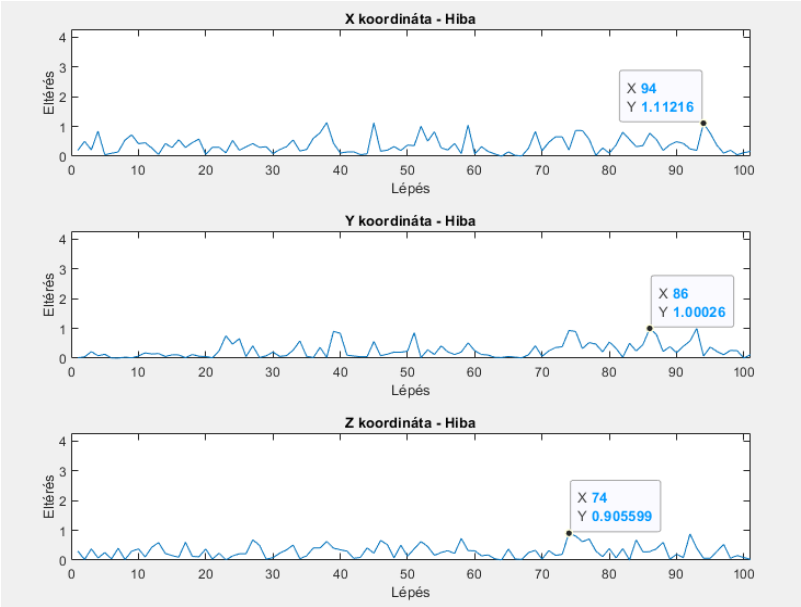
A 12. ábrán megfigyelhető, hogy a csuklóváltozók lépésenkét 360 foknál nagyobb mértékben változnak, emiatt történnek a nagy pörgő mozdulatok a szimuláció során. A legnagyobb érték a 4-es csuklón 43600 fok változás, de a többi csuklón is megfigyelhető a 20000 fok körüli hirtelen változások.



12. ábra – Kör pálya 1. eset pályakövetés szimulációjának csuklóváltozó értékei

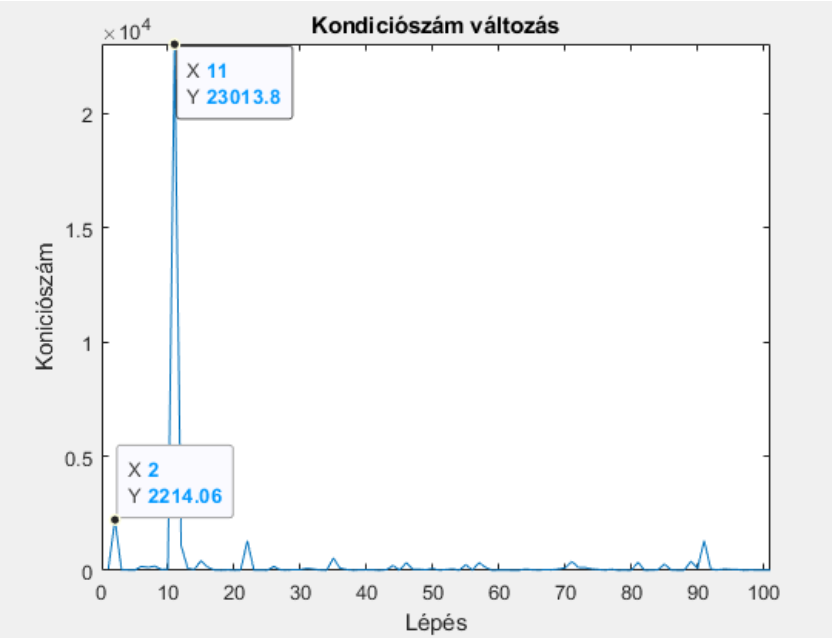
4.2.3 Hiba elemzése

A 13. ábrán látszik, hogy a megalkotott algoritmusnál általánosnak vehető 0.005-ös hibaérték helyett ebben az instabil helyzetben 1 körül ingadozik a hiba mértéke.



13. ábra – Kör pálya 1. eset pályakövetés szimulációjának hiba értékei

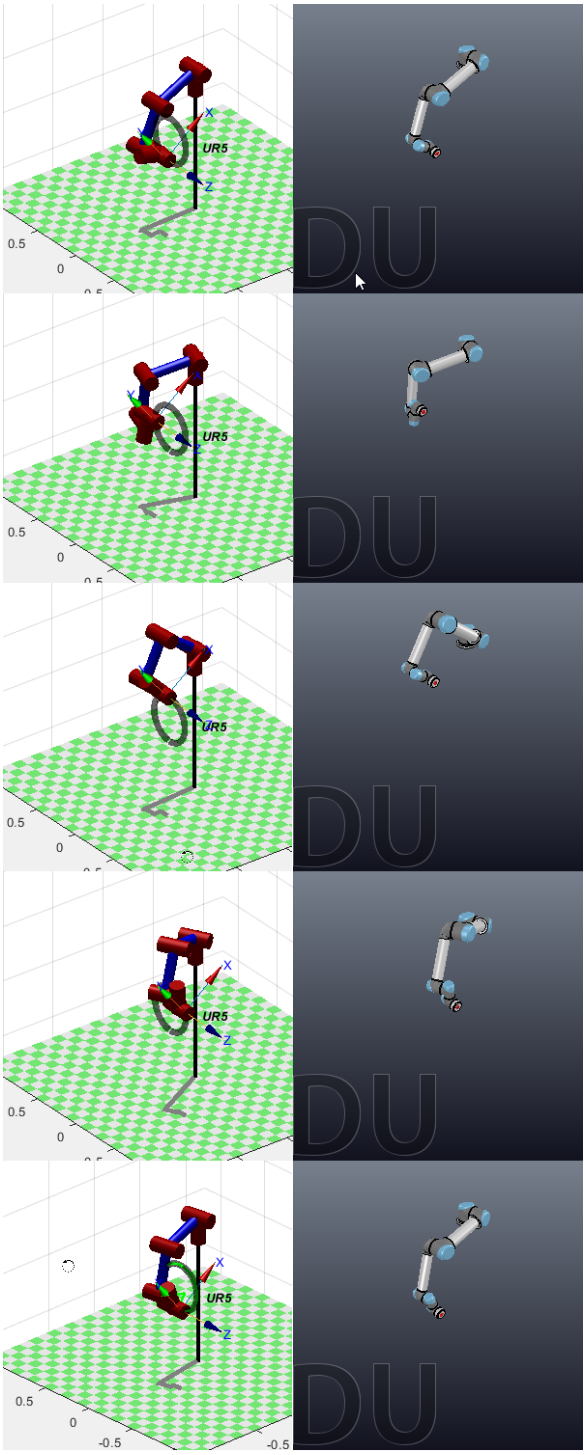
Az instabilitás oka jól látszik a 14. ábrán, a 2. lépésnél a robotkar megközelít egy szinguláris állapotot, majd a 11. lépésnél teljesen instabillá változik.



14. ábra – Kör pálya 1. eset pályakövetés szimulációjának kondíciószámai

4.3 Kör pálya 2. eset – stabil

4.3.1 Pályakövetés

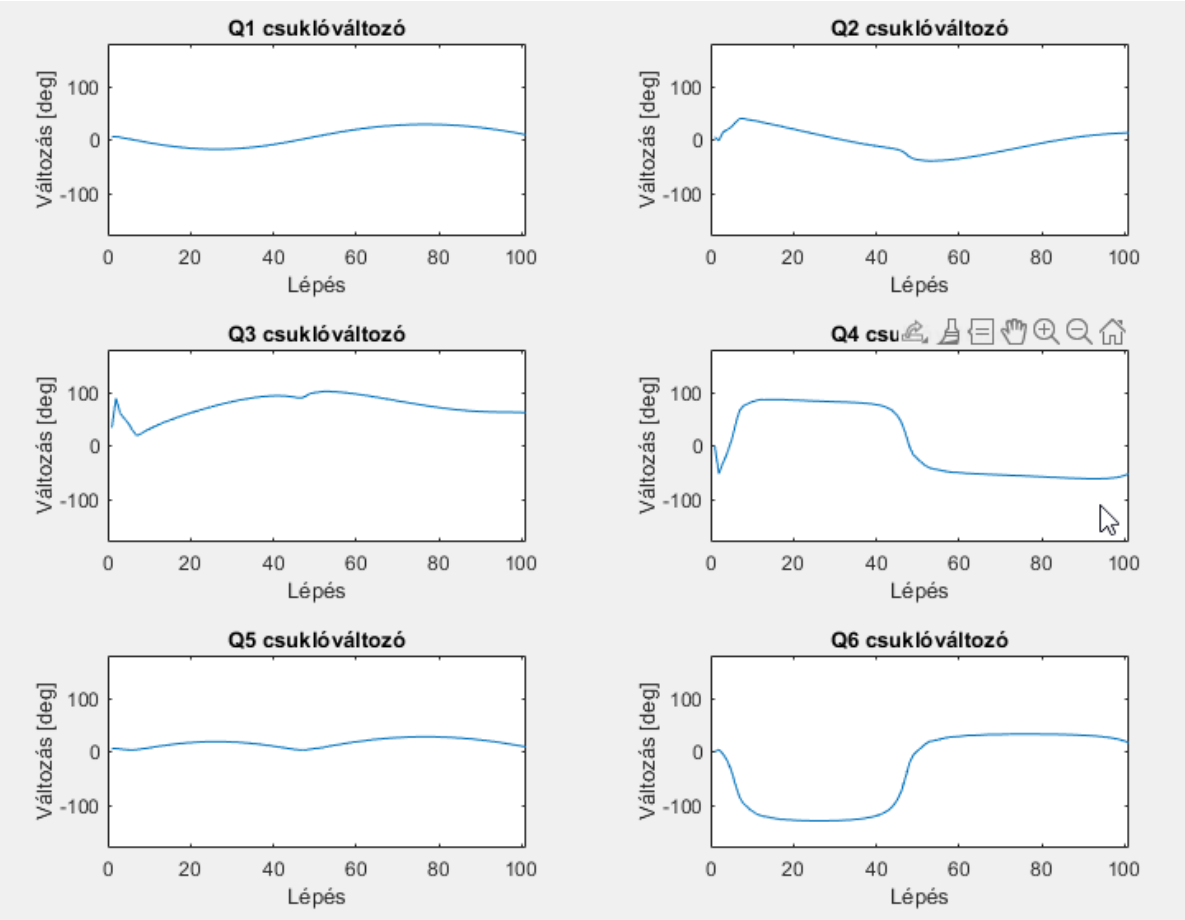


15. ábra – Kör pálya 2. eset pályakövetés szimulációja UR5 robotkarral (bal oldalon MatLab, jobb oldalon CoppeliaSim alkalmazásban)

Ennél a konfigurációnál a robotkar nem vált instabillá, kis hiba értékkel követi a kijelölt pályát. Látható a 15. ábrán, hogy a két alkalmazásban megegyeznek a robotkar által felvett pózok.

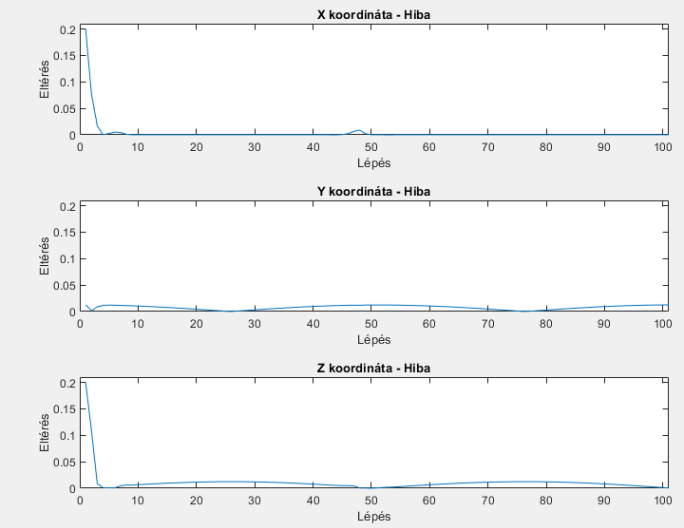
4.3.2 Csuklóváltozók elemzése

Az instabil kör tesztesethez hasonlítva itt teljesen normális csuklóváltozó értékeket láthatunk az ábrán. Sehol nem haladja meg -180/+180 fokos tartományt.



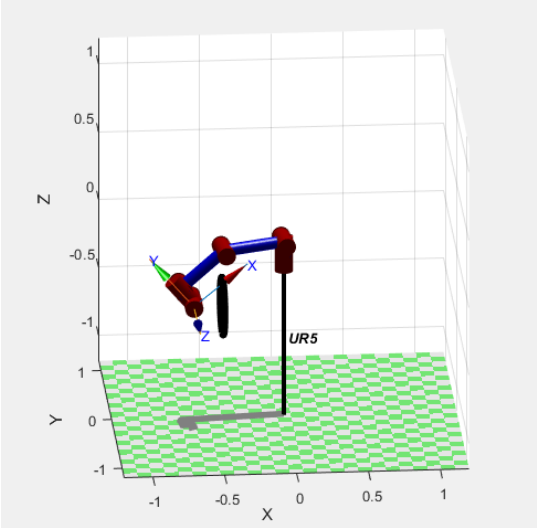
16. ábra – Kör pálya 2. eset pályakövetés szimulációjának csuklóváltozói

4.3.3 Hiba elemzése



17. ábra – Kör pálya 2. eset pályakövetés szimulációjának hibaértékei

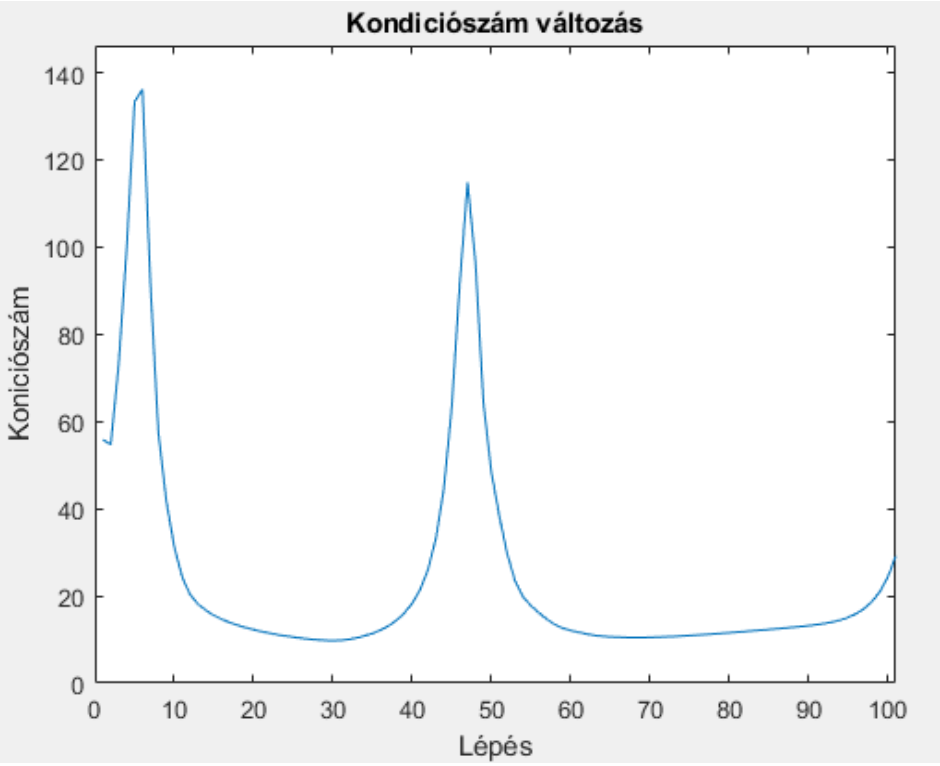
A 0.2 kezdeti hiba abból adódik, hogy a kezdő pont nem része a kör pályának, a 18. ábrán szemléletesen látszik a távolság.



18. ábra – Kör pálya 2. eset pályakövetésénél a robotkar nem a pályáról indul

Miután beáll a robotkar a pályára, látható, hogy a hiba drasztikusan lecsökken és beáll az általánosnak mondható 5 ezredes nagyságrendre. Ezek alapján elmondhatóm, hogy ha minimálisan akarjuk tartani a hibát a mozgás idejére ügyelni kell rá, hogy a kiindulópont része legyen a pályának. Ezt úgy érhetjük el, hogy a referencia pálya első pontjának koordinátáiból kiszámoljuk a hozzá tartozó csuklóváltozókat. Az így kapott csuklóváltozót adjuk meg a kezdésnél.

A 19. ábrán a kondíciószámokon látszik, hogy a mozgás végig stabil volt, sehol nem közelített meg szinguláris állapotot.

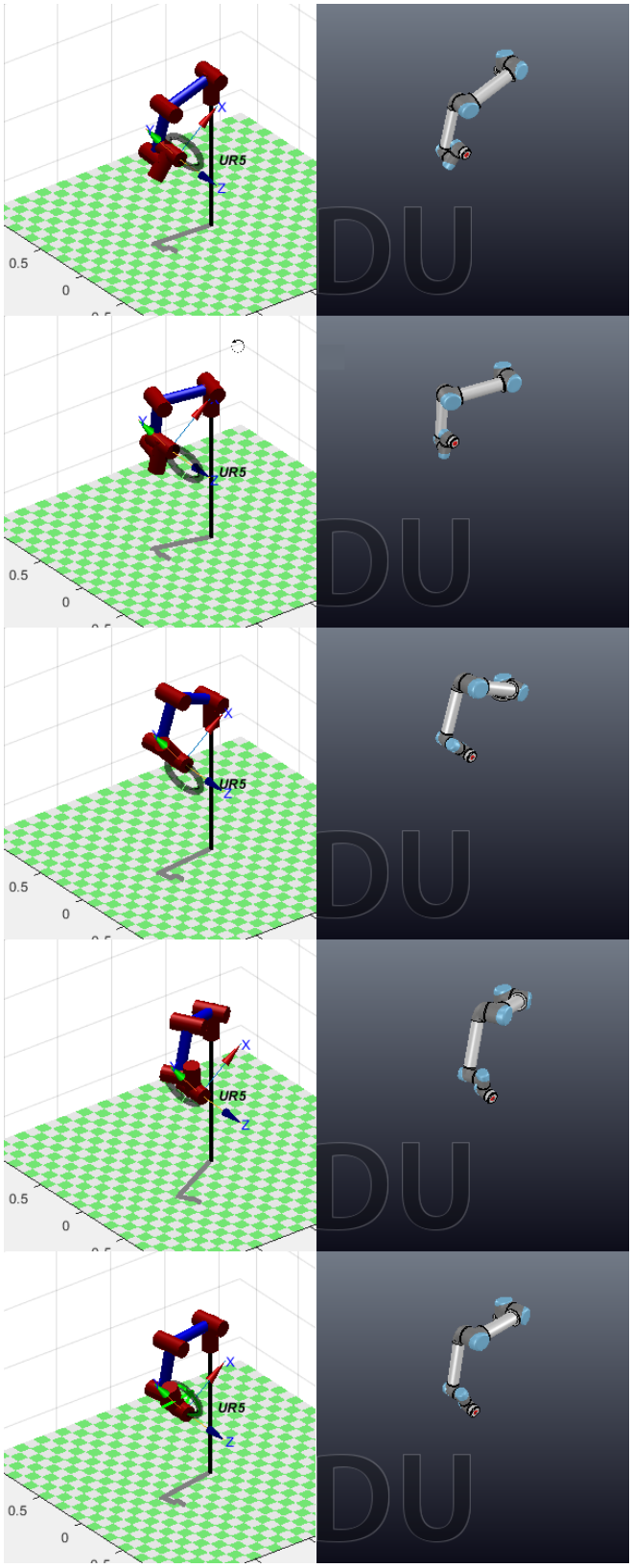


19. ábra – Kör pálya 2. eset pályakövetés szimulációjának kondíciószámai

4.4 Ellipszis pálya

4.4.1 Pályakövetés

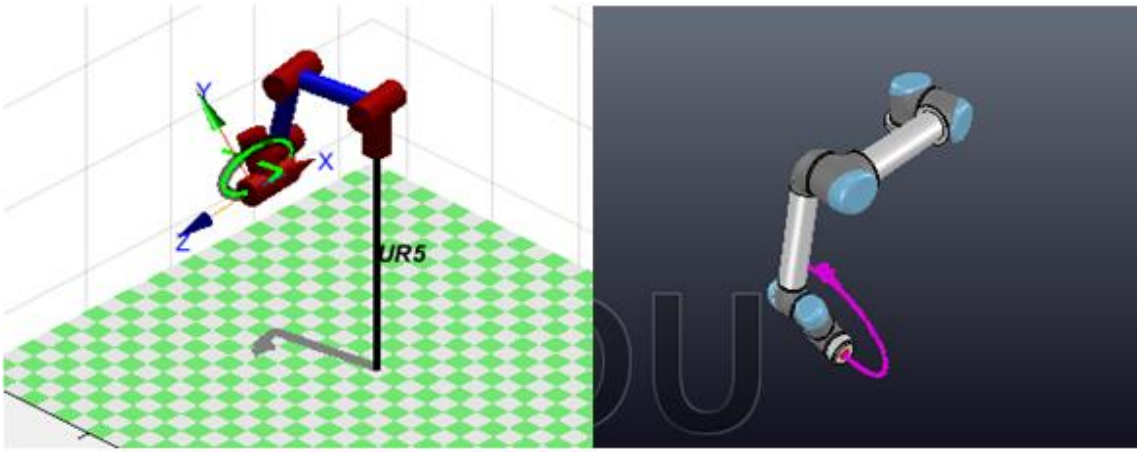
Az ellipszis pályát is a körhöz hasonlóan megfelelően tudta követni a robotkar, ahogy azt a 20. ábrán nyomon követhetjük. Látható, hogy a CoppeliaSim szimulációban is a MatLab-bal megegyezően mozog a robotkar a pályabejárás során.



20. ábra – Ellipszis pálya pályakövetés szimulációja lépésenként (bal oldalon MatLab, jobb oldalon CoppeliaSim alkalmazásban)

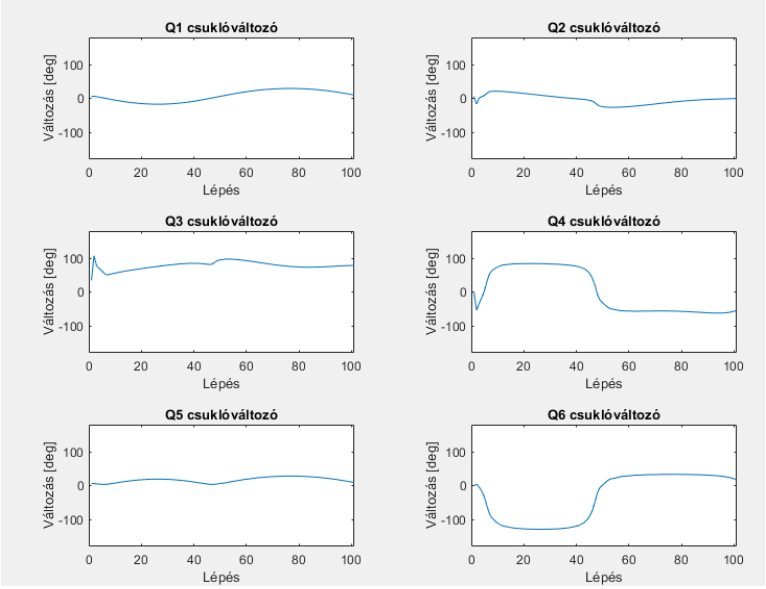
4.4.2 Csuklóváltozók elemzése

A pálya felénél, amikor a 4-es és 6-os csuklók egyszerre fordulnak át a robotkar CoppeliaSim-ben megremeg, ami látszik a kirajzolt ellipszis felső ívén a 21. ábrán, ez a MatLab-os ábrán nem jelenik meg. A különbséget a CoppeliaSim-ben szimulált erők adják, amik a MatLabnál nincsenek hatással a robotkarra.



21. ábra - Ellipszis pálya pályakövetés szimulációjának kirajzolt eredménye (bal oldalon MatLab, jobb oldalon CoppeliaSim alkalmazásban)

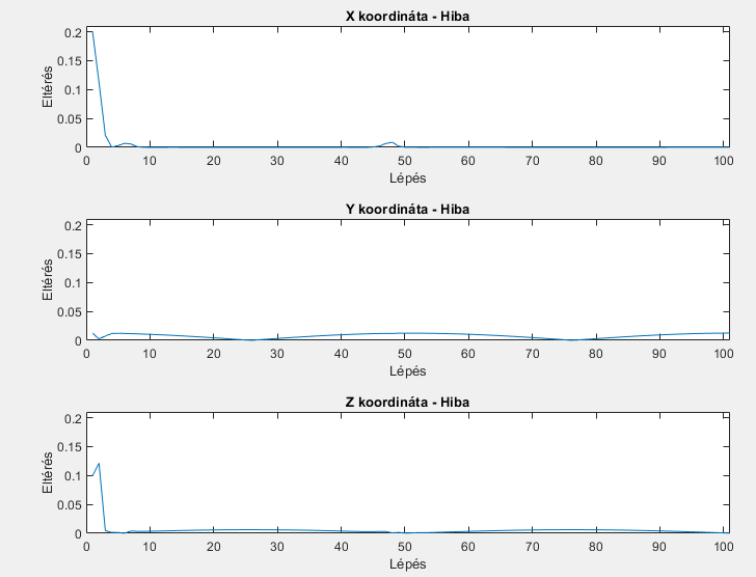
A 22. ábrán látszik, ahogy a 4-es és 6-os csuklók egyszerre fordulnak át a robotkaron, itt a grafikon meredeken változik.



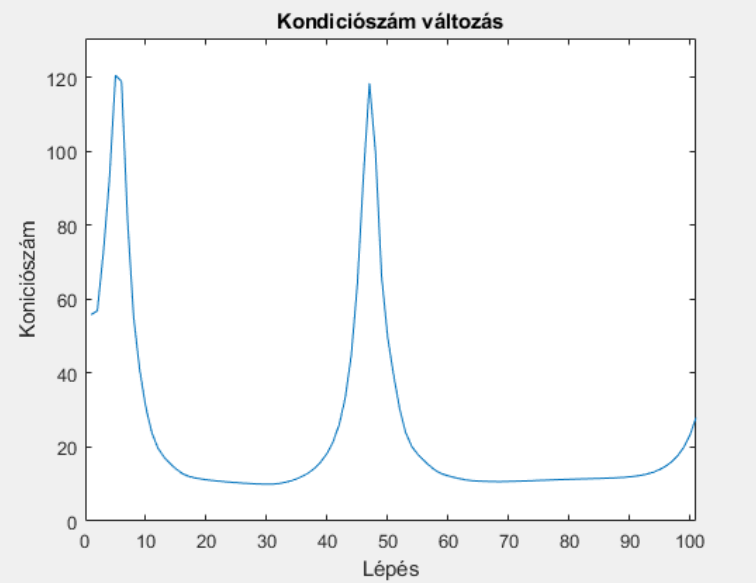
22. ábra – Ellipszis pálya pályakövetés szimulációjának csuklóváltozó értékei

4.4.3 Hiba elemzése

A körhöz hasonlóan itt is megtalálható a 0.2 indulási hiba a 23. ábrán, mert ekkora távolságra helyezkedik el a robotkar end-effektora a kijelölt pályától. Miután beáll a kijelölt pályára, ez az algoritmus is a 0.005 körüli hiba értékkel működik. A 23. ábrán láthatjuk kondíciószámokban nincs nagy kiugrás, a két átfordulás jelenik csak meg a görbén. Szinguláris állapot közelébe nem kerül.



23. ábra – Ellipszis pálya pályakövetés szimulációjának hibaértékei



24. ábra – Ellipszis pálya pályakövetés szimulációjának kondíciószámai

4.5 Általánosan felmerült hibák

- A CoppeliaSim felé lassabb az adatáramlás a rendszerben, ezért kissé szaggatott a robotkar mozgása. A MatLab önálló futása esetén a robotkar mozgása folyamatos.
- A CoppeliaSim kezelőfelületén nehézkes a robot forgatása, így nehézkes a két alkalmazás egymáshoz igazítása.
- A CoppeliaSim lassan reagál a szimuláció közben kiadott utasításokra a kezelőfelületről, ezért például egy megállítást a MatLab oldalról kell kezdeményezni.

4.6 Fejlesztési ötletek

A feladat tesztelése és összegzése során felmerült fejlesztési ötletek megvalósításával az elkészült rendszer egy magasabb szintre emelhető a jövőben.

4.6.1 MatLab grafikus felhasználói interfész megvalósítása

Ahhoz, hogy az alkalmazás széles körben használható legyen, a MatLab parancssoros bevitelét grafikus felhasználói interfészre kell cserélni. Ez lehetővé tenné, hogy azok is használhassák a rendszert, akik csak a robotkarok pályatervezésének elméletet ismerik, a programozás ismereteiben viszont járatlanok.

4.6.2 Megoldás összehasonlítása Gazebo alkalmazással

A feladatmegoldás során a CoppeliaSim alkalmazás mellett sok helyen felmerült a Gazebo alkalmazás, mint virtualizációs megoldás. Érdemes lehet összehasonlítani az elkészült rendszert, egy Gazeboban felépített hasonló rendszerrel.

4.6.3 Általános robotkar kezelés

Az eddig megvalósított rendszer csak az Universal Robots UR5-ös robotot kezeli. Bár kézzel meg lehet változtatni a csuklók kiosztását, más robotokar, de ilyen formában nem felhasználóbarát az alkalmazás. Az következő lépcső, hogy a megvalósított grafikus kezelőfelületen, egy listából tudjuk meghatározni, hogy melyik robottal szeretnénk dolgozni. Ezt követően szeretnék áttérni egy olyan megoldásra, hogy az alkalmazás az összekötő interfészen érzékelje, hogy a munkatérbe milyen robotkart helyezett el a felhasználó és automatikusan erre az eszközre végezze el a számolásokat és a beállításokat.

4.6.4 Szolgáltatás jelleg kialakítása

Ki kell dolgozni a rendszer szolgáltatásként nyújtásához szükséges alapokat. Jelenleg mindkét keretrendszerben manuális parancsokat kell megadni a működtetéshez. Ahhoz, hogy a felhasználók körében elterjedjen a megoldás, el kell fedni ezeket az átmeneteket. Az egységbe zárás után pedig ki kell dolgozni egy licensz kezelési folyamatot.

4.6.5 VR szemüveges kiegészítés

Készült egy fejlesztés a CoppeliaSim alkalmazáshoz, amivel virtuális valóság szemüveggel is működtethetővé vált az alkalmazás. Ez a megjelenítési módszer jó kiegészítője lenne a megvalósított a rendszernek.

5 Összefoglalás

A robotkarok pályatervezése összetett matematikai feladat, azonban a technológiai lehetőségnek köszönhetően jól általánosítható. A feladat megoldása során betekintést nyertem a pályatervezés kihívásaiba és matematikai megoldásaiba. Egyszerű és összetett pályák tervezésére is alkalmas megoldás született. Az egyszerűbb pályán, ami a megoldásban egy egyenes volt, a robotkar kis hiba értékkel tudott végig haladni. Az összetett pályákon, mint a kör vagy az ellipszis már megfigyelhető volt a nagyobb csuklóváltozó értékeknél kisebb megingások, illetve egy szinguláris konfiguráció létrejött a tesztelések során.

A feladatban megvalósított CLIK algoritmus a tesztelés során megmutatta, hogy valóban jól csillapítja a pályakövetési hibákat, így azt a mérnöki munkában történő alkalmazásra javasolnám. Fontos része volt a megvalósításnak a kondíciószám vizsgálata, amivel meg lehet mutatni a szinguláris konfigurációkat, ezáltal el lehet kerülni, hogy a robot instabillá váljon a pályabejárások közben.

A MatLab Robotics System Toolbox rendszerével a pályatervezében felmerülő problémákra szoftveres megoldás született. A megvalósított mozgásokat egy interfészen keresztül, a CoppeliaSim alkalmazás segítségével, virtuális környezetben jeleníti meg a kialakított rendszer, így egy valósághű megjelenítésen is meg lehet vizsgálni a pályatervezési eredményeket.

6 Summary

Motion planning for robotic arms is a complex task, but thanks to the techonlogy it can be easily generalized. While I was working on this project, I gained insight into challenges of the motion planning and I found mathematical solutions for these problems. A solution has been created that is suitable for designing both simple and complex paths. On the simpler path, which was a straight line in the solution, the robot arm was able to move along with a small error value. On complex paths such as the circle or ellipse, small oscillations for the large values of the joint variables were already observed, and a singular configuration was also created during testing.

The implemented and tested CLIK algorithm in the solution handled the path following errors well, so I recommend it for use in engineering work. An important part of the implementation was the examination of the condition number, which can be used to show the singular configurations, thus preventing the robot from becoming unstable during the trajectories.

I created MatLab program for the path planning problems with the Robotics System Toolbox. The motions of robotic arm can be simulated in V-REP virtual envrionment, so it is also possible to examine the path planning results on a realistic view.

Irodalomjegyzék

[1] Bruno Siciliano - Jacobian-based algorithms: A bridge between kinematics and control - Proceedings of the Special Celebratory Symposium In the honor of Professor Bernie Roth's 70th Birthday, pp. 4-35, - 2003

[2] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo - Robotics - Modelling, Planning and Control
Springer-Verlag - London - 2009
ISBN 978-1-84628-641-4

[3] Mester Gyula – Robotika - Szegedi Tudományegyetem Természettudományi és Informatikai Kar Műszaki Informatika Tanszék - 2011

[4] CoppeliaSim – Remote API MatLab - http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm
Utoljára megtekintve: 2022-02-10

Melléklet

Forráskód állományok:

- remApi.m
- remoteApi.dll
- remoteApiProto.m
- schmidt_laszlo_bdydvp_megoldas.m
- schmidt_laszlo_bdydvp_megoldas.ttt
- schmidt_laszlo_bdydvp_megoldas_rajz.ttt