ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZET
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

SZAKDOLGOZAT

OE-KVK
2023.

Sánta Szabolcs
T008791/FI12904/K

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Műszertechnikai és Automatizálási Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Sánta Szabolcs
Szakdolgozat száma: SZD2309040923379500[redacted]
Törzskönyvi száma: T008791/FI12904/K **Neptun kódja:** [redacted]

Szak: villamosmérnöki (magyar nyelven)
Specializáció: Műszer-automatika specializáció

A dolgozat címe: Csuklókaros robot inverz kinematika megoldó algoritmusa
A dolgozat címe angolul: Inverse kinematics solver algorithm of an articulated robot
A feladat részletezése: Mutassa be és elemezze a módszereket, amelyekkel megvalósítható egy csuklókaros robot inverz kinematika megoldó algoritmusa, és egy kiválasztott módszerrel tervezzen algoritmust!

Intézményi konzulens neve: Sándor Tamás

A kiadott téma elévülési határideje:2025. 12. 31.
Beadási határidő: 2023. 12. 15.

A szakdolgozat: Nem titkos.

Kiadva: Budapest, 2023. 12. 12.

Óbudai Egyetem Kandó Kálmán Villamosmérnöki Kar Műszertechnikai és Automatizálási Intézet
P.H

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZETI
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Ceglédbercel, 2023. 12. 15.

Sánta Szabolcs

hallgató aláírása

Köszönetnyilvánítás

Sok köszönettel tartozom azoknak, akik lehetővé tették a robot projekt és a szakdolgozatom megvalósulását. Köszönöm a konzulensemnek, Sándor Tamásak a rengeteg segítséget és útmutatást. Szintén köszönet Dr. Schuster Györgynek a jól irányzott hasznos tanácsokat. Hálásan köszönöm a szaktársamnak és egyben munkatársamnak, Homoki Bencének az elképesztően eredményes, eddig tizenhat hónapon át tartó csapatmunkát a közös robot projektben. Köszönöm a PowerQuattro Zrt-nek az általa biztosított támogatást, amely lehetővé teszi a robot projekt megvalósulását. Köszönet a PowerQuattro Zrt. fejlesztőmérnökének, Miháczi Viktornak, hogy vállalta a szakdolgozatom bírálatát.

Tartalomjegyzék

Bevezetés

2022 szeptemberében a szaktársammal és egyben kollégámmal, Homoki Bencével megálmodtunk egy robotkar projektet, amely elkísér minket a szakdolgozatig és talán azon is túl. Így lett. Egy éven át kutattunk, terveztünk és számításokat végeztünk, hogy idén, 2023 szeptemberében a prototípus csuklókaros robot fizikai formát ölthessen. Az építés az utolsó szakaszában jár mostanra. Ahhoz, hogy megmozduljon és elvárt módon működjön, szükség van a mozgásirányító rendszer szoftveres megvalósítására, ami egy rendkívül összetett és sok részből álló feladat. A robotok mozgásirányításának legfontosabb, legalapvetőbb művelete és egyben kihívása az úgynevezett inverz kinematikai feladat megoldása, amely bonyolult matematikát igényel. Jelen szakdolgozatban azt a célt tűztem magam elé, hogy a prototípus csuklókaros robot inverz kinematika megoldó algoritmusát szoftveresen megvalósítom. Természetesen ezt a lehetséges módszerek kutatása és tesztelése előzi meg. A robotika interdiszciplináris tudományág, amely komplex automatizálási problémák megoldásával foglalkozik, éppen ez a komplexitás és a vele járó kihívások jelentik azt, amit szeretek benne. Összesíti mindazt a műszaki tudást, ami kiemelkedően érdekel.

1 Robot kinematikai alapfogalmak

1.1 Bevezetés

Jelen fejezetben ismertetem és szemléltetem a robot manipulátorok kinematikájának fontos alapfogalmait, hogy a későbbi fejezetekben ezen ismeretekre alapozva az olvasó számára érthetően, többek között matematikai formában tárgyalhassam a robotok kinematikáját, kiemelt fókuszt helyezve a robot inverz kinematikájával kapcsolatos legfontosabb kérdésekre és összefüggésekre, mely felsoroltak ismerete és figyelembe vétele nélkülözhetetlen az inverz kinematika megoldó algoritmus szoftveres megvalósításának előkészületei során.

1.2 Az ipari robot fogalma

Az “MSZ EN ISO 10218-1:2011 Robotok és robotszerkezetek. Ipari robotok biztonsági követelményei 1. rész Robotok” szabvány a 2011. november elsején közzétett angol nyelvű változatának 2019. március elsején kiadott magyar nyelvű változata. Alapjául a “Robots and robotic devices. Safety requirements of industrial robots. Part 1: Robots (ISO 10218:2011)” máig érvényes nemzetközi szabvány szolgál. Az MSZ EN ISO 10218-1:2011 szabvány definiálja, mit nevezünk ipari robotnak (industrial robot):

“*Automatikusan vezérelt, újra programozható többcélú manipulátor, amelynek három vagy több tengelye programozható, helyhez kötött vagy mozgó ipari automatizálási alkalmazásokhoz használható*” [1]

Egy robot manipulátor kinematikai szempontból merev testek olyan rendszere, amelyben az egyes testek rugalmas kapcsolatban állnak, és egymáshoz viszonyított helyzetük változik az idő függvényében a robot működése során, tehát mozgást végeznek. E merev testek összetett mozgása egyenes vonalú, tehát haladó, valamint forgó mozgásokból tevődik össze. Ezek a merev testek a robotszegmensek. [2]

1.3 Működési módok és biztonság

Az MSZ EN ISO 10218-1:2011 szabvány definiálja az ipari robotok biztonságkritikus működési módjait és körülményeit. Az ipari robotok és irányító rendszereik

biztonságkritikus rendszerek, mert a legtöbb esetben nagy erejű gépekről van szó, ezért kiemelt fontosságúak az emberi élet védelmére szolgáló biztonsági intézkedések. Különösen fontos a szabvány szerinti biztonsági kritériumoknak megfelelni, amennyiben az ipari alkalmazás olyan, hogy ember tartózkodhat a robot munkaterületén.

A szabvány az alábbi meghatározásokat közli a működési módokkal kapcsolatban:

Kézi üzemmód (manual mode): *"Vezérlési állapot, amely engedélyezi a kezelőszemély általi közvetlen irányítást"* [1]

Automatikus működés (automatic operation): *"Olyan állapot, amelyben a robot a programozott feladatot szándék szerint hajtja végre"* [1]

Kollaboratív működés (collaborative operation): *"Olyan állapot, ahol az erre tervezett robotok dolgoznak közvetlen együttműködésben egy emberrel egy meghatározott munkaterületen"* [1]

Belátható, hogy a kollaboratív működés esetében nagyobb szigorral kell kezelni a biztonsági kérdéseket, mert a munkaterületen, a robot közvetlen környezetében ember tartózkodik ilyenkor. A szabvány az alábbi sorokkal is felhívja a figyelmet erre:

"A robotnak le kell állnia, amikor személy lép a kollaboratív munkaterületre" [1]

"A robot visszatérhet az automatikus működésbe, amikor a személy elhagyta a kollaboratív munkaterületet" [1]

A személyi biztonság mellett még fontos az anyagi biztonság is, különösen, ha több robot dolgozik azonos munkaterületen, egyidejűleg, ekkor a robotmozgások pályáit üzembiztosan össze kell hangolni. Ez az egyidejű mozgás (simultaneous motion) esete, melyet a szabvány a következőképpen határoz meg:

"Két vagy több robot egyetlen vezérlőállomással vezérelt egyidejű mozgatása, amely koordinálható vagy szinkronizálható közös matematikai korreláció alkalmazásával" [1]

1.4 Szabadsági fok

Egy kinematikai rendszer szabadsági foka (Degree of Freedom - DoF) egy n, pozitív egész szám, amely megadja, hogy hány független tengely szerint végezhet egyszerű mozgást az adott rendszer. [2]

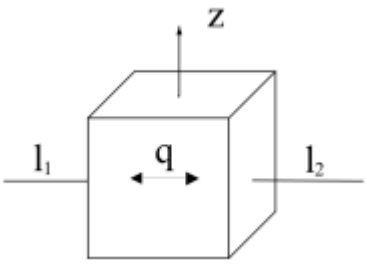
1.5 Robotcsuklók

A robotszegmenseket robotcsuklók kötik össze. A robotcsuklók valósítják meg a szomszédos robotszegmensek egymáshoz viszonyított haladó vagy forgó mozgását. A robotcsuklóknak tehát két fajtája különböztethető meg: [2] [3]

- a transzlációs csuklók,
- és a rotációs csuklók.

1.5.1 Transzlációs csukló

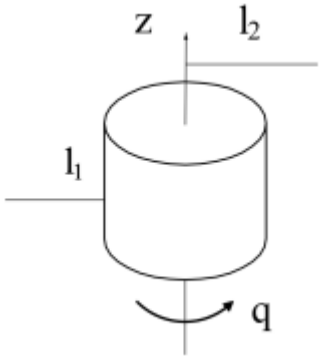
A transzlációs csuklók egy szegmensnek egy másikhoz képest az egyenes vonalú térbeli eltolását valósítják meg. Az ilyen csuklókat T szimbólummal jelöljük. Egy T csukló sematikus ábrája axonometrikus nézetben egy az egyenes szakasszal jelölt szegmensek között elhelyezett négyzet alapú hasáb. Az alábbi ábra szemlélteti a T típusú robotcsukló sematikus ábrázolását: [2]



1. ábra - Transzlációs csukló sematikus ábrája [2]

1.5.2 Rotációs csukló

Rotációs csukló felhasználásával egy robotszegmens forgó mozgása hozható létre egy másik szegmens körül. Ennek a típusnak R a jele. Az R típusú csuklók axonometrikus nézetű sematikus árbája egyenes körhenger, a T típussal azonosan ez is a szegmensek közé helyezendő. Az alábbi ábra szemlélteti az R csuklókkal kapcsolatban leírtakat: [2]



2. ábra - Rotációs csukló sematikus ábrája [2]

1.6 Robotszegmensek

A már korábban tett említések alapján a robotszegmensek tehát a csuklókat összekötő merev testek, tagok, amelyeknek kinematikai és dinamikai paramétereik vannak. Ezek a paraméterek segítenek leírni a robotot alkotó merev testek mozgás- és erőtani viselkedését. [2]

Az egyes robotszegmensek kinematikai paraméterei a következők: [2]

- a szegmens L_i hossza, amely a szegmenshez kapcsolódó csuklók közötti tengelytávolság, továbbá
- a csukló-tengelyek egymáshoz viszonyított, bezárt szöge koordináta tengelyek szerint.

E kinematikai paraméterek csukló-típusoktól és robot alapkonfigurációktól függően változó és állandó értékek is lehetnek. Ezeknek a paramétereknek a koordináta-transzformációknál lesz fontos jelentősége.

A robotszegmensek dinamikai paraméterei pedig: [2]

- a szegmens m_i tömege és
- a J_i tehetetlenségi nyomatéka.

A dinamikai paraméterek függenek a kinematikai paraméterektől, továbbá e paramétereknek mind a két fajtája végső soron a robot konfiguráció geometriájától függ. Ezeknek a paramétereknek a robot és aktuátorainak terhelhetősége, a sebesség- és gyorsulásviszonyok szempontjából van jelentősége.

1.7 Kinematikai pár

A kinematikai pár olyan kinematikai rendszer, amely két merev test rugalmas összekapcsolásával jön létre, tehát két robot szegmenst egy csukló köt össze. Léteznek egy és több szabadsági fokú kinematikai párok. Két szegmens gömbcsuklós kapcsolata például egynél több szabadsági fokú kinematikai pár. A későbbiekben egy szabadsági fokú kinematikai párokat tárgyalok és a belőlük felépülő kinematikai láncokat. [2]

1.8 Kinematikai lánc

A kinematikai lánc egymást követő kinematikai párokból felépített láncolat. Létezik egyszerű és összetett kinematikai lánc, valamint lehet zárt lánc, ha az első és az utolsó tag kapcsolódik, és ezzel hurkot képeznek, vagy lehet nyílt lánc, amely nem képez hurkot. Egy kinematikai lánc egynél több szabadsági fokú. Az ipari robotikában használatos robotkar konfigurációk különböző kinematikai láncokként modellezhetőek. [2] A továbbiakban csak egyszerű, nyílt kinematikai láncokat vizsgálok.

1.9 A robotok koordináta-rendszerei

1.9.1 Világkoordináták és az effektor helyzete

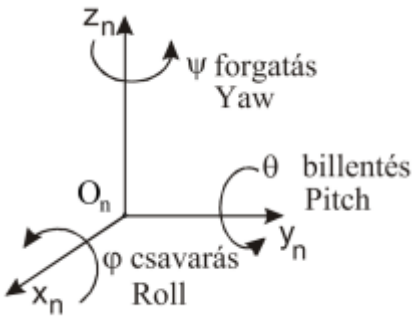
Képzeljünk el egy robotkart. A robotkar rögzített alapzatán vegyünk fel egy a robot bázisközéppontjával egybeeső origójú, Descartes féle derékszögű koordináta-rendszert. Abban az esetben, ha a robot egy planáris manipulátor, tehát minden mozgás egyetlen jól meghatározott síkban zajlik, akkor elegendő egy sík-koordinátarendszer, ha viszont a mozgás a térben történik, például az előbb említett sík elforog a térben egy harmadik tengely körül, akkor három koordinátára van szükség, tehát háromdimenziós Descartes koordináta-rendszer kell. Az imént létrejött O központú, globális értelmezésű, nyugvó O(x, y, z) vonatkoztatási koordináta-rendszer lesz nevezetesen a világ-koordinátarendszer. E vonatkoztatási rendszerhez képest vesszük fel az O_i (i = 1; 2; …; n) csuklóközpontú, Descartes féle lokális koordinátarendszereket, melyek a nyugvó világ-koordinátarendszerhez képest haladó és forgó mozgást végeznek az idő függvényében. [2]

A robotkar kinematikai lánca két részre bontható. Az első (alsó) három szabadsági fok felelős a főmozgásért, pontosabban szólva az effektor pozicionálásáért. A robot ezen részét alapkonfigurációnak vagy alapgépnek nevezzük. Az ezen felüli szabadsági fokok (tipikusan még három) az effektor orientációjának beállítására hivatottak, és a lánc ezen része az effektorhoz tartozik. A rögzítési ponttól indulva a legutolsó, O_n központú n-edik lokális koordináta-rendszer az effektor- vagy szerszám-koordinátarendszer. [2] [3]

Az effektor, vagy szabványos kifejezéssel élve a végberendezés (end-effector) MSZ EN ISO 10218-1:2011 szabvány szerinti meghatározása a következő: *“Kifejezetten a mechanikus felülethez való csatlakozásra tervezett eszköz, amely lehetővé teszi a robot számára a feladat elvégzését”* [1]

Az robot manipulátor effektorának helyzete megadható világ- és csuklókoordinátákban. Ha az effektor helyzetét világkoordinátákban adjuk meg, akkor az a következők szerint történik.

A szerszámközéppont (Tool Centre Point - TCP) a robot effektor-koordinátarendszerének O_n origója. Az MSZ EN ISO 10218-1:2011 szabvány a következő módon definiálja: *"Meghatározott pont az adott alkalmazáshoz, figyelembe véve a mechanikus interfész koordináta-rendszerét"* [1] A szerszámközéppont térbeli helyzete az effektor-koordinátarendszer origójának világ-koordinátarendszerbeli helyzeteként írható le. Az effektor világ-koordinátarendszerbeli helyzetét tehát pozíció és orientáció adja meg. A TCP pozíciója az x, y és z koordinátákkal írható le, melyek valójában x(t), y(t) és z(t) pozíció-időfüggvények. Az effektor orientációját a ψ (Yaw – forgatási vagy ingatási szög), θ (Pitch – bólintási vagy billentési szög) és φ (Roll – csavarási szög), módosított Euler-szögekkel lehet meghatározni, melyek szintén időfüggő mennyiségek. [2] [3]



3. ábra - Módosított Euler-szögek

A 3. ábra szemlélteti a Roll, Pitch és Yaw szögeket.

Az s, világkoordináta vektor tartalmazza az effektor három tengely szerinti pozíció koordinátáit és módosított Euler-szögeit az alábbiak szerint: [2]

$$s = \begin{bmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \varphi \end{bmatrix} \qquad (1.1)$$

1.9.2 Csuklókoordináták

A csuklók által megvalósított mozgásokat a q_1 … q_n csuklóváltozók, vagy gyakoribb néven csuklókoordináták reprezentálják. Rotációs csukló esetén q_i (i = 1; 2; 3; … ; n) értéke szögmennyiségként értelmezett, míg transzlációs csukló esetén a q_i értéke távolság. Működés közben minden $q_i(t)$ csuklókoordináta az idő függvényében változik. A q vektor tartalmazza a kinematikai lánc csuklókoordinátáit. A q vektor dimenziószáma egyenlő a rendszer szabadsági fokainak számával, tehát n szabadsági fokú kinematikai lánc csuklókoordinátáinak vektora n dimenziós. Alább látható a q vektor: [2]

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \qquad (1.2)$$

1.9.3 Munkatér

Egy robotkar műveleti tere vagy munkatere (Workspace vagy Task Space) a világkoordináta rendszer azon térbeli koordinátáinak halmazát jelenti, amelyeket a robot effektorának szerszámközéppontja üzemszerűen elérhet, pozícióként felvehet. A TCP tehát csak a munkatér határfelületei által bezárt térrészben tartózkodhat. Egyszerűen kifejezve a munkatér a robotkar térbeli mozgástartományát és annak határait jellemzi, az összes lehetséges, megengedett póz térigényét tartalmazza. Robot konfigurációtól függően a munkatér eltérő alakú és méretű. [2] [3]

1.9.4 Holttér

A holttér a munkatér halmazának komplementere, tehát a tiltott, vagy geometriából adódóan elérhetetlen helyzeteket magába foglaló térrész. A robot manipulátor mozgékonyságának növelése érdekében maximalizálni kell a munkateret és minimalizálni a holtteret. [2]

1.9.5 Csuklótér

Egy robotkar csuklótere (Joint Space) egy olyan képzeletbeli n dimenziós koordináta rendszer, amelyet n darab csuklókoordináta feszít ki. A csuklótér dimenziószáma megegyezik a kinematikai lánc szabadsági fokainak számával. Egy szabadsági fokú kinematikai pár csuklótere egydimenziós, n szabadsági fokú kinematikai lánc csuklótere n dimenziós. A csuklótérben a lehetséges q_i csuklóbeállítások korlátozottak, azok q_{imin} és q_{imax} csuklókoordináta határok között változhatnak, azokat nem haladhatják meg. [2] [3]

$$q_{i_{min}} \leq q_i \leq q_{i_{max}} \qquad (1.3)$$

Csuklókoordinátánként e határokon túl is egyfajta holttér található, de nem ugyanaz, mint a munkatér esetében! A határhelyzetek geometriai kényszerek, melyek például a szegmensek ütközésének elkerülését hivatottak biztosítani.

1.10 A robotkarok alapkonfigurációi

Adott, hogy az egy szabadsági fokú kinematikai párok robotcsuklói vagy R típusúak (rotációs csuklók), vagy T típusúak (transzlációs csuklók) lehetnek. Ez kétfajta lehetséges mozgás. Adott továbbá, hogy egy kinematikai lánc n szabadsági fokú lehet. Említettem, hogy egy robot manipulátor első három szabadsági foka, a pozicionáló mozgásokat megvalósító alapkonfigurációhoz tartozik.

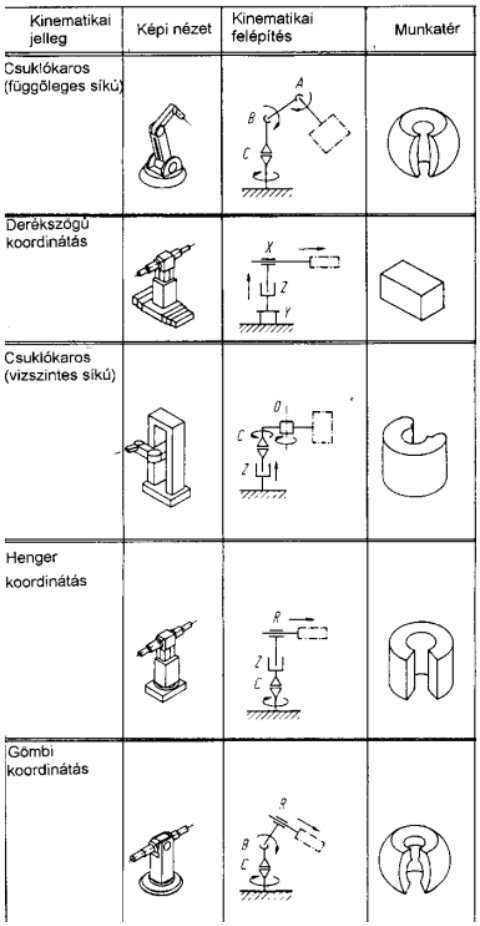
Ezek alapján a robot manipulátoroknak $2^3 = 8$ fajta alapkonfigurációja lehet, tehát kinematikailag a robotszegmensek ennyi féleképpen kapcsolódhatnak össze. A lehetséges három szabadsági fokú alapkonfigurációk tehát a következőek: [2] [3]

RRR, RRT, RTT, RTR, TRR, TTR, TRT és TTT.

Néhány alapkonfiguráció munkatere:

1.1. táblázat - Alapkonfigurációk és munkatereik [2]

Jelölés:	Alkalmazott csuklók:	Munkatér alakja:
TTT	Három transzlációs csukló alkotja.	Téglatest, hasáb, vagy kocka.
RTT	Egy rotációs és két transzlációs csuklóból áll.	Hengergyűrű.
RRT	Két rotációs és egy transzlációs csuklóból áll.	Üreges gömb.
RRR	Három rotációs csuklóból épül fel.	Gömb.

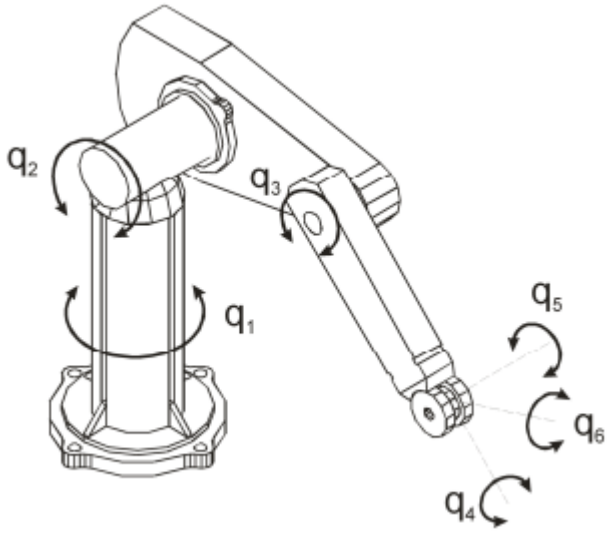


4. ábra - Alapkonfigurációk munkaterei [3]

Egységnyi szegmenshosszok és robotcsuklónként azonos (±180°) határszögek esetén az alapkonfigurációk közül az RRR konfiguráció munkatere arányosan a legnagyobb. [2]

Az ipari körülmények között alkalmazott robot manipulátorok esetén a legelterjedtebb rotációs csuklókból felépített struktúra, mert villamos hajtás esetén leggyakrabban szervomotorokat alkalmaznak, melyek irányítása könnyedén megoldható (PWM-mel – impulzusszélesség modulációval), viszont forgó mozgásukat haladó mozgássá alakítani speciális mechanikai kialakításokat igényel és ezért körülményes lenne a legtöbb esetben. [2]

A kizárólag rotációs robotcsuklók felhasználásával felépített robot manipulátort csuklókaros robotnak nevezzük. [2] A későbbi fejezetekben csak a csuklókaros kialakítást vizsgálom, de a pontosság kedvéért némely esetekben említést teszek még a transzlációs csuklókról, amennyiben az éppen tárgyalt témával kapcsolatban fontosnak tartom megemlíteni.



5. ábra - PUMA: 6 szabadsági fokú csuklókaros robot [2]

1.11 Direkt kinematikai feladat

A direkt kinematika (Direct Kinematics vagy Forward Kinematics) problémakörének lényege az, hogy a robot manipulátor, mint kinematikai lánc csuklókoordináta

beállításai ismertek, közvetlenül határozzuk meg őket, tehát csuklókoordinátákban adjuk meg az effektor helyzetét, és ezeknek a csuklókoordinátáknak a függvényében keressük a TCP világkoordinátáit, amelyet a kinematikai lánc csuklókoordinátáinak meghatározott beállítása eredményez. A direkt kinematikai feladat tehát a csuklótérből a munkatérbe, tehát csuklókoordinátákról a világkoordinátákra irányuló koordináta-transzformáció. Ehhez meg kell találnunk egy olyan függvénykapcsolatot, amely ezt az átalakítást egyértelműen leírja a következő forma szerint: [2]

$$s = f(q) \qquad (1.4)$$

ahol:

- s – a keresett világkoordináták vektora,
- q – a megválasztott csuklókoordináták vektora,
- f(q) – pedig egy $\mathbb{R}^n \rightarrow \mathbb{R}^m$ hozzárendelésű, nemlineáris, folytonosan differenciálható, vektor-vektor függvény, amely a csuklókoordinátákat világkoordinátákká transzformálja, valamint ennek a folyamatnak a nemlineáris egyenletrendszerét tömöríti, és ahol:
  - n – a csuklókoordináták vektorának dimenziószáma, amely megegyezik a kinematikai lánc szabadsági fokainak számával,
  - m – pedig a világkoordináták vektorának dimenziószáma.

Az s = f(q) hozzárendelés nyílt kinematikai láncú robot manipulátorra minden esetben egyértélmű, tehát a direkt kinematikai feladat az analitikus geometria módszereivel egyértelműen megoldható. [2] [4]

1.12 Inverz kinematikai feladat

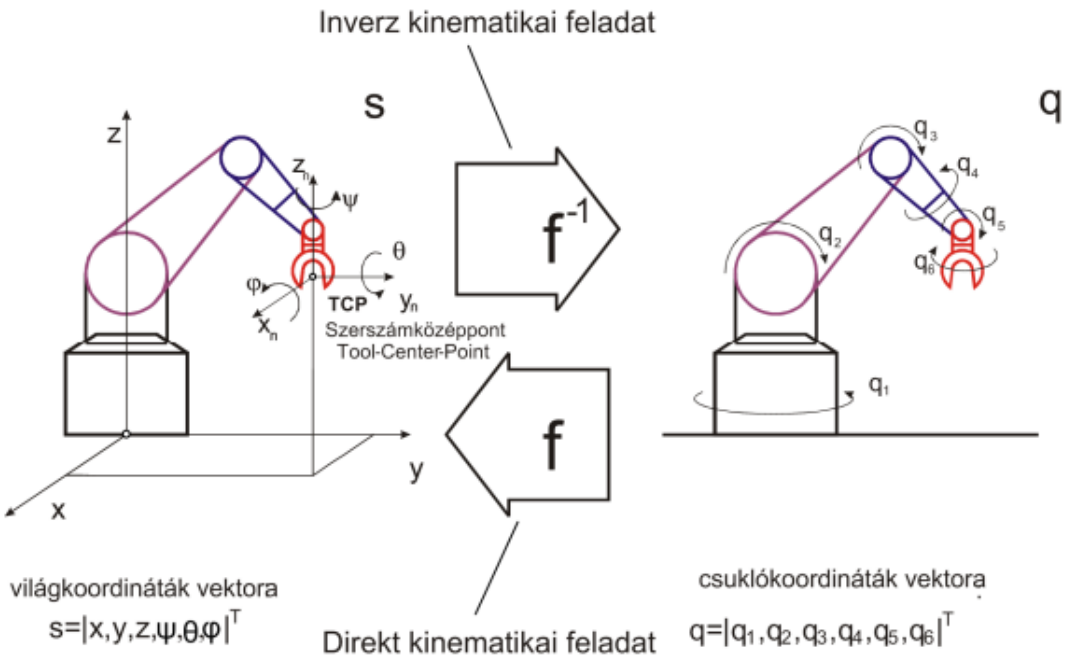
Az inverz kinematikai (Inverse Kinematics) feladat esetében az előbb ismertetett direkt kinematikai művelet megfordításáról, tehát inverz műveletéről van szó. Az inverz kinematikai problémakör lényege, hogy az effektor szerszámközéppontja által elérni kívánt helyzetet világkoordinátákban adjuk meg, és keressük azokat az ismeretlen csuklókoordinátákat, amelyek beállításával elérhető az említett célhelyzet a

munkatérben. A direkt kinematikánál ismertetett függvénykapcsolatnak az inverzét kell, hogy képezzük, és ennek megfelelően rendezzük a kinematikai egyenletet: [2]

$$q = f^{-1}(s) \qquad (1.5)$$

ahol az $f^{-1}(s)$ függvény az f(q) inverz függvénye, és a vektor-vektor hozzárendelés iránya megfordul: $\mathbb{R}^m \rightarrow \mathbb{R}^n$. [2]

A következő ábra szemlélteti a direkt és az inverz kinematikai feladatok koordináta transzformációját, hat szabadsági fokú példa manipulátorral:



6. ábra - Direkt és inverz kinematika kapcsolata [2]

A nehézség az inverz kinematikával kapcsolatban az, hogy a robotkar geometriájától és konfigurációjától függően a fentebb leírt visszafelé transzformáció analitikus módszerrel a legtöbb esetben nem oldható meg egyértelműen, ami azt jelenti, hogy több lehetséges megoldás létezik. A kinematikai redundancia fogalma ad némi betekintést az említett probléma természetébe. [2] [3] [4]

1.13 Kinematikai redundancia

A korábban említett n és m dimenziószámok nagy mértékben befolyásolják a robot manipulátor kinematikai feladatainak kiszámíthatóságát, különös tekintettel az inverz kinematikai feladat megoldhatóságára.

Az n dimenziószámról már kiderült, hogy a szabadsági fokok számával, valamint a q csuklókoordináták vektorának dimenziószámával azonos, az m dimenziószám pedig az s világkoordináták vektorának dimenziószáma.

Az m és n relációjában három eset lehetséges:

- m > n → A manipulátor kevesebb szabadsági fokú, mint amennyit a TCP teljes pozicionálása és orientálása megkíván, tehát a robot nem képes minden helyzetet elérni, a kinematikai lánc alulhatározott, ezesetben pedig az inverz kinematika általánosságban könnyen megoldható. [2] [3] [4]
- m = n → A robotkar kinematikai lánca nem redundáns, tehát éppen annyi szabadsági foka van a kinematikai láncnak, amennyi a TCP teljes pozicionálásához és orientációjának teljes beállításához feltétlenül szükséges, de nem több. Abban az esetben, ha a világkoordináta rendszer háromdimenziós Descartes féle tér, és ezáltal a világkoordináták vektora hatdimenziós (három lineáris és három anguláris koordináta), akkor a teljes pozicionálás és orientálás hat szabadsági fokot követel meg. [2] [3] [4]
- m < n → A manipulátor kinematikailag redundáns, vagy másként kifejezve túlhatározott, tehát több szabadsági fok áll rendelkezésre, mint amennyi a teljes mozgatáshoz szükséges, ilyenkor az inverz kinematikai feladatnak számos megoldása létezik, melyek megtalálása az analitikus geometria módszerével igen bonyolult. [2] [3] [4]

2 Direkt kinematikai számítás

2.1 Denavit-Hartenberg féle koordináta-transzformáció

Korábban már volt szó koordináta-transzformációról abban az értelmezésben, hogy a csuklótér csuklókoordinátái és a munkatér világkoordinátái között teremtsünk matematikai kapcsolatot, továbbá volt szó nyugvó, vonatkoztatási világkoordináta rendszerről és a hozzá képest haladó és forgó mozgást végző, a robotcsuklókhoz rögzített, lokális koordinátarendszerekről. A Denavit-Hartenberg féle koordináta-transzformációs eljárás során úgynevezett homogén transzformációs mátrix segítségével a csuklókoordináták világkoordinátákká alakíthatóak, tehát ez a mátrix tömöríti tulajdonképpen a direkt kinematikai függvénykapcsolatot, amelyről már volt szó. Továbbá lehetővé teszi a kinematikai lánc összetett mozgásának általánosított, egynemű leírását.

2.2 Rotációs mátrixok

Adott két koordináta-rendszer, melyek origói közötti eltolást nem vesszük figyelembe, nullának tekintjük, hogy egybeessenek, mert most csak az orientációjukat vizsgáljuk:

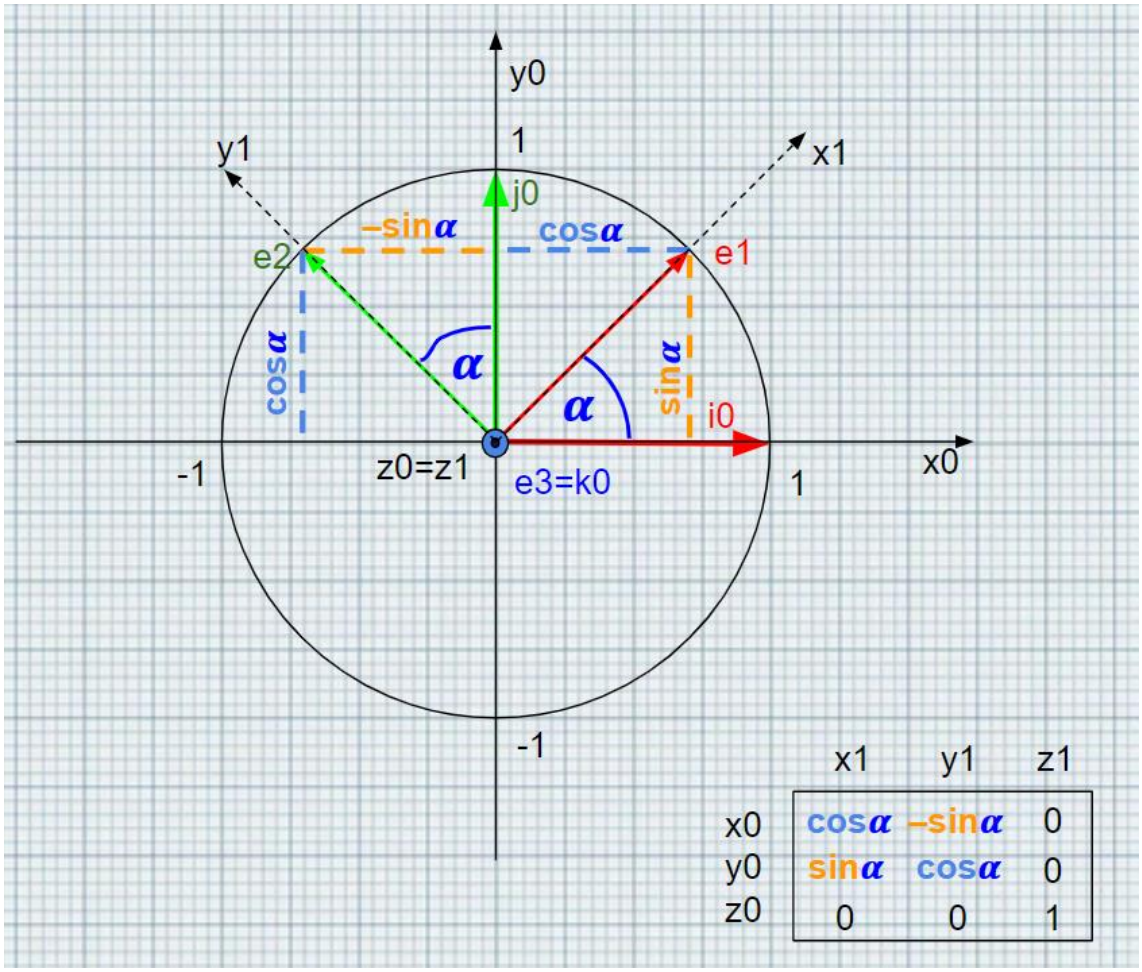
- legyen az $O(x_0, y_0, z_0)$ a nyugvó vonatkoztatási koordináta-rendszer,
- és legyen az $O_1(x_1, y_1, z_1)$ a mozgó, lokális koordináta-rendszer, amely az O központú rendszerhez képest eltérő orientációjú.

Reprezentálják a koordináta-rendszereket tengelyenként azok egységhosszúságú bázisvektorai:

- $O(i_0, j_0, k_0)$
- $O_1(e_1, e_2, e_3)$

Ekkor a két koordináta-rendszer azonos tengelyeire értelmezett bázisvektorok szögeket zárnak be egymással (i_0 és e_1, j_0 és e_2, k_0 és e_3). Ahhoz, hogy megkapjuk az O_1 koordináta rendszer O-hoz viszonyított orientációját, az e_1, e_2 és e_3 bázisvektorokat egyenként rá kell vetíteni az i_0, j_0 és k_0 bázisvektorokra, ez háromszor három, azaz kilenc vetületértéket jelent, melyek nulla és egy közé esnek. Ha ezeket a vetületeket

tengelypáronként sorokba és oszlopokba rendezzük, egy 3×3 méretű mátrixot kapunk, amely így relatív orientációt tartalmaz, és rotációs, avagy forgatómátrixnak nevezzük és R betűvel jelöljük. [2] [5]



7. ábra - Rz rotációs mátrix jelentése

Az ábra egy z tengely körüli forgatómátrix előállítását szemlélteti a fentebb részletezett módszerrel. Akkor van szükség z tengely körüli R_z rotációs mátrixra, amennyiben az O_1 koordináta rendszer előállítható az O koordináta rendszerből tisztán a z tengely körüli α szögű elforgatással, utóbbi pedig azt feltételezi, hogy a z_0 és z_1 tengelyek egybeesnek és azonos irányúak. Az R_z rotációs mátrix harmadik sorának harmadik oszlopa 1-et tartalmaz, mivel a z_0 és z_1 egybeesnek, így a e_3 bázisvektor k_0-ra vetített képe egységhosszú. Az R_z mátrixban található nullák azt mutatják, hogy mivel z merőleges x-re és y-ra, így az e_1 és e_2 bázisvektorok zérus vetületűek a z tengelyre, valamint az e_3

bázisvektor is zérus vetületű x és y tengelyekre. A trigonometrikus elemek az $x_1 \rightarrow x_0$, $x_1 \rightarrow y_0$, $y_1 \rightarrow x_0$ és $y_1 \rightarrow y_0$ vetítéseket reprezentálják. [2] [5]

$$R_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.1)$$

Az R_z rotációs mátrixhoz hasonlóan, azonos logikával vezethető le R_y és R_x rotációs mátrixok felírása is: [2] [5]

$$R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \qquad (2.2)$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \qquad (2.3)$$

Ha két koordinátarendszer teljesen azonos orientációjú, azt a kapcsolatot egységmátrix fejezi ki: [5]

$$R = I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.4)$$

Amennyiben például minden tengely egybeesik, de mind ellentétes irányú a vetítőpárjával akkor a rotációs mátrix negatív főátlójú egységmátrix: [5]

$$R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \qquad (2.5)$$

Robotcsuklóról-robotcsuklóra külön-külön rotációs mátrixok definiálhatóak. Az effektor lokális orientációját a következő egyenletek szerint, az egyes rész-rotációs mátrixokat sorrendben összeszorozva, át lehet alakítani világkoordináta rendszerbeli effektor orientációvá. [2]

Az eredő rotációs mátrix nyílt formulában: [2] [5]

$$ {}^{0}R_n = {}^{0}R_1 \cdot {}^{1}R_2 \cdot {}^{2}R_3 \cdot \ldots \cdot {}^{n-1}R_n \qquad (2.6) $$

és zárt formulában:

$$ {}^{0}R_n = \prod_{i=1}^{n} {}^{i-1}R_i \qquad (2.7) $$

Megjegyzem az indexeléssel kapcsolatban, hogy a Denavit-Hartenberg módszerben hivatalosan bal felső indexben jelöljük az aktuális transzformáció vonatkoztatási koordináta-rendszerének számjelét, jobb alsó indexben pedig a vonatkoztatott koordináta-rendszer számjelét. Ezt az indexelési módszert követi Mester Gyula Robotika című könyve, és a szakdolgozatomban én is ezt használom. [2] A szakirodalmi kutatással kapcsolatos tapasztalataim szerint más módok is vannak alkalmazásban az indexek elhelyezésére [5] [6], de az előbb említett mód a leggyakoribb.

Az eredő ${}^{0}R_n$ rotációs mátrix tartalmazza az orientációs kapcsolatot a nyugvó világ-koordinátarendszer és a mozgó, TCP (O_n) központú effektor-koordinátarendszer között. Egy P munkatérbeli pont helyzetmeghatározása a következőképpen történik meg: [2]

$$ r = {}^{0}R_n \cdot p + k \qquad (2.8) $$

ahol:

- r - az $O(x_0, y_0, z_0)$ globális koordináta-rendszerbeli helyzetvektora a P munkatérbeli pontnak,
- ${}^0R_n$ - az $O(x_0, y_0, z_0)$ és az $O_n(x_n, y_n, z_n)$ koordináta-rendszerek közötti forgási kapcsolatot leíró eredő rotációs mátrix,
- p - az $O_n(x_n, y_n, z_n)$ lokális koordináta-rendszerbeli helyzetvektora a P munkatérbeli pontnak,
- k - az O és az O_n origók közötti eltolási vektor, tehát az O_n origó $O(x_0, y_0, z_0)$ koordináta-rendszerbeli helyzetvektora.

Ugyanez kifejtve koordinátákkal és a rotációs mátrix vetületelemeivel: [2]

$$\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} e_{1x} & e_{2x} & e_{3x} \\ e_{1y} & e_{2y} & e_{3y} \\ e_{1z} & e_{2z} & e_{3z} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \qquad (2.9)$$

ahol a rotációs mátrixban foglalt, indexekkel ellátott e vetületek az első indexű e bázisvektor második indexű tengelyre vetített képei.

2.3 Homogén transzformációs mátrix és direkt kinematika

Az előző egyenlet egy negyedik, egy értékű vektorkoordináta bevezetésével, valamint a rotációs mátrix és az eltolási vektor összevonásával, úgynevezett homogén koordinátás alakra hozható, így a vektoregyenlet egyszerűsödik: [2] [5]

$$r = {}^0H_n \cdot p \qquad (2.10)$$

ahol ${}^0H_n$ a 4×4 méretű homogén transzformációs mátrix, amely immár nem csak a forgási kapcsolatot írja le a koordináta-rendszerek között, hanem tartalmazza az origók közötti eltolás mértékét is. A homogén transzformációs mátrix általános, kompakt alakja: [5]

$$ {}^{0}H_{n} = \begin{bmatrix} {}^{0}R_{n} & k \\ 0^{T} & 1 \end{bmatrix} \qquad (2.11) $$

A helyzetvektorok kifejtett vektoregyenlete: [2]

$$ \begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = \begin{bmatrix} e_{1x} & e_{2x} & e_{3x} & k_x \\ e_{1y} & e_{2y} & e_{3y} & k_y \\ e_{1z} & e_{2z} & e_{3z} & k_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \qquad (2.12) $$

Mivel a teljes kinematikai láncra érvényes ${}^{0}H_{n}$ transzformációs mátrix függvénye a csuklókoordináták q vektorának, ezért alkalmas arra, hogy direkt kinematikai függvénykapcsolatot valósítson meg a csuklótér és a munkatér között, általa tehát megoldható a direkt kinematikai feladat. [2] [4] [5]

2.4 Denavit-Hartenberg paraméterek

A homogén transzformációs mátrixot a Denavit-Hartenberg paraméterek határozzák meg, amelyek a Denavit-Hartenberg transzformáció szabályain és a robotcsuklókhoz rögzített lokális koordináta rendszerek felvételén alapszanak. Maga a Denavit-Hartenberg módszer a névadók azon észrevételén alapszik, hogy két különböző, három tengelyű Descartes féle koordináta rendszer közül valamelyik, mindössze két transzlációval és két rotációval a másik koordináta rendszerré alakítható, tehát fedésbe hozhatóak. Ezt a két transzlációt és két rotációt fejezi ki a négy Denavit-Hartenberg paraméter. [2]

A kinematikai lánc csuklóihoz rögzített koordináta rendszerek felvételének szabályai a következők: [2] [5] [6]

- A láncban minden i-edik és i+1-edik robotcsuklón derékszögű koordináta rendszereket veszünk fel.
- A z tengelyeket minden csukló hatásos irányába választjuk meg, tehát
    - rotációs csukló esetén jobbsodrású forgástengelynek,

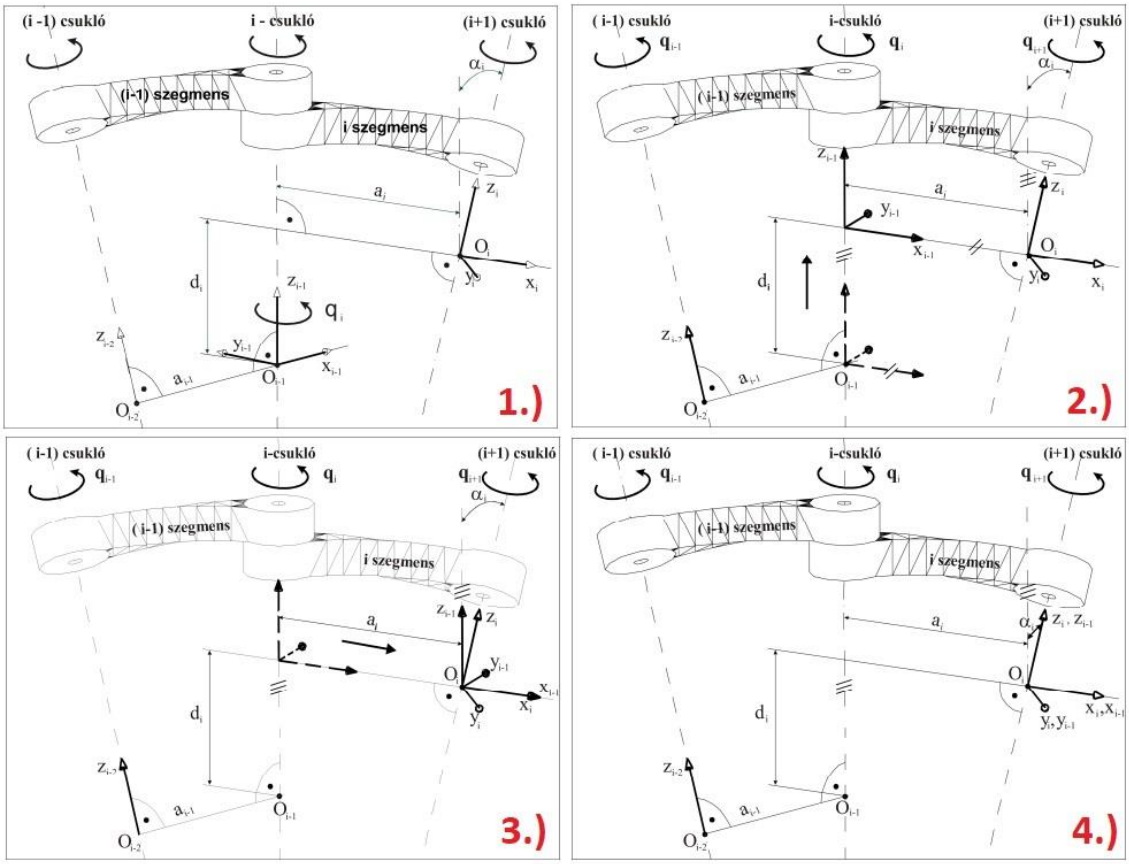
    - transzlációs csukló esetén az eltolás pozitív irányának.
- Az i+1-edik csuklón kijelöljük az $O_i(x_i, y_i, z_i)$ koordináta rendszert.
- A z_i tengely azonos irányú az i+1-edik csukló tengelyével, a fentiek értelmében.
- Az x_i tengely egybeesik az i-edik és i+1-edik csuklótengely közös normálisával és az i-ediktől az i+1-edik felé mutat.
- Az y_i tengely irányát úgy kell megválasztani, hogy az $O_i(x_i, y_i, z_i)$ koordináta rendszer jobbsodrású vektorrendszer legyen, tehát $x_i \times y_i$ kereszt-szorzat := z_i.
- Az i-edik csuklón kijelöljük az $O_{i-1}(x_{i-1}, y_{i-1}, z_{i-1})$ koordináta rendszert.
- A z_{i-1} tengely azonos irányú az i-edik csukló tengelyével.
- Az x_{i-1} tengely egybeesik az i-1-edik és i-edik csuklótengely közös normálisával és az i-1-ediktől az i-edik felé mutat.
- Az y_{i-1} tengely irányát úgy kell megválasztani, hogy az $O_{i-1}(x_{i-1}, y_{i-1}, z_{i-1})$ koordináta rendszer jobbsodrású vektorrendszer legyen, tehát $x_{i-1} \times y_{i-1}$ kereszt-szorzat := z_{i-1}.

A négy Denavit-Hartenberg paraméter, amelyek a két-két említett transzlációt és rotációt fejezik ki, és amelyek segítségével egy koordinátarendszer a másikba alakítható:

- d_i - transzláció - az az i-edik csukló z_{i-1} tengelye mentén mért távolság, amely a kiválasztott csuklótengely két normálisa (a_{i-1} és a_i) között van, rotációs csukló esetén konstans érték, transzlációs csukló esetén változó érték, mert ez maga a haladó mozgást kifejező csuklókoordináta, és csakis ilyenkor ezt a paramétert q_i-vel jelöljük, [2]
- a_i - transzláció - az i-edik és i+1-edik csuklótengelyek közös normálisának x_i tengelyen mért hossza, ez a paraméter transzlációs csukló esetén zérus értékű, [2]
- α_i - rotáció - az a jobbsodrású szög, amelyet az i-edik csukló z_{i-1} tengelye és az i+1-edik csukló z_i tengelye az a_i-re merőleges síkban mérve egymással bezár, [2]
- q_i - rotáció - az a jobbsodrású szög, amelyet az x_{i-1} és x_i tengelyek zárnak be, rotációs csukló esetén ez változó és maga a

csuklókoordináta, transzlációs csukló esetén viszont egy rendszerfüggő konstans θ_i szög, mert a q_i jelölést a haladó mozgású csuklókoordináta kapja meg. [2]

A 8. ábra a négy Denavit-Hartenberg paraméter jelentését, a koordinátarendszerek helyes felvételét és a homogén koordináta-transzformáció elemi lépéseit szemlélteti:



8. ábra - A homogén koordináta transzformáció lépései [2]

A Denavit-Hartenberg paraméterek segítségével egyik koordinátarendszernek a másikba való homogén transzformációjának lépései a következők:

1) Az i-edik csukló $O_{i-1}(x_{i-1}, y_{i-1}, z_{i-1})$ koordinátarendszerét z_{i-1} tengely körül elforgatjuk q_i jobbsodrású szöggel addig, amíg az x_{i-1} tengely az x_i tengellyel párhuzamossá nem válik. Ezt a forgatást a $D(q_i)$ mátrix írja le, amely a z körüli q_i szögű forgatás kifejezendően egy R_z típusú rotációs mátrixot tartalmaz, és amelynek az eltolási vektora helyén nullvektor van, mivel nincs eltolás. [2]

$$D(q_i) = \begin{bmatrix} \cos q_i & -\sin q_i & 0 & 0 \\ \sin q_i & \cos q_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.13)$$

2) Az $O_{i-1}(x_{i-1}, y_{i-1}, z_{i-1})$ koordinátarendszert z_{i-1} tengely mentén d_i mértékkel eltoljuk, az O_{i-1} origó ekkor a z_{i-1} és az x_i tengelyek metszéspontjára kerül, az x_{i-1} és x_i tengelyek pedig egy egyenesbe esnek. Ezt az eltolást a $D(d_i)$ mátrix írja le, amelyben a rotációs mátrix R = I egységmátrix, továbbá a jobb oldalon az eltolási vektor k_z koordinátája di értékű, a k_x és k_y pedig zérus. [2]

$$D(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.14)$$

3) Az $O_{i-1}(x_{i-1}, y_{i-1}, z_{i-1})$ koordinátarendszert x_i tengely mentén a_i mértékkel eltolva bevontatjuk az O_i origóba. Ezt az eltolást a $D(a_i)$ mátrix írja le, amelyben az x irányú eltolást az eltolási vektor a_i értékű k_x koordinátája reprezentálja, az orientáció változatlanságát pedig az R = I egység-rotációs mátrix. [2]

$$D(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.15)$$

4) Végül, mivel az előző lépések eredményeképpen immár közös az origó és az x tengelyek is, így már csak α_i jobbsodrású szöggel kell az $O_{i-1}(x_{i-1}, y_{i-1}, z_{i-1})$ koordinátarendszert a közös x tengely mentén elforgatni addig, amíg az x-re

merőleges síkban lévő y és z tengelyek páronként fedésbe kerülnek. Ezt a forgatást a $D(\alpha_i)$ mátrix írja le, melyben mivel nincs eltolás, ezért a k eltolási vektor nullvektor, valamint egy R_x típusú rotációs mátrix fejezi ki az x körüli α_i szögű forgatást. [2]

$$D(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.16)$$

2.5 A kinematikai lánc eredő Denavit-Hartenberg transzformációs mátrixa

A kinematikai lánc egy szegmensére érvényes homogén transzformációs mátrixot az egyes lépésekben ismertetett elemi transzformációs mátrixok alábbi sorrend szerinti szorzataként lehet meghatározni.

Rotációs csukló esetén: [2]

$$^{i-1}D_i = D(q_i) \cdot D(d_i) \cdot D(a_i) \cdot D(\alpha_i) \qquad (2.17)$$

$$^{i-1}D_i = \begin{bmatrix} \cos q_i & -\sin q_i \cos\alpha_i & \sin q_i \sin\alpha_i & a_i \cos q_i \\ \sin q_i & \cos q_i \cos\alpha_i & -\cos q_i \sin\alpha_i & a_i \sin q_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.18)$$

Transzlációs csukló esetén: [2]

$$^{i-1}D_i = D(\theta_i) \cdot D(q_i) \cdot D(a_i) \cdot D(\alpha_i) \qquad (2.19)$$

$$^{i-1}D_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & 0 \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & q_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.20)$$

A teljes kinematikai láncra érvényes Denavit-Hartenberg transzformációs mátrix, amely összekapcsolja a munkatérben a nyugvó vonatkoztatási koordináta-rendszert és az effektor koordináta-rendszerét, az egyes szegmensek homogén transzformációs mátrixainak helyes sorrendű szorzataként számítható. [2] [4] [6] [7]

Nyílt formulában:

$$^{0}T_n = {}^{0}D_1 \cdot {}^{1}D_2 \cdot {}^{2}D_3 \cdot \ldots \cdot {}^{n-1}D_n \qquad (2.21)$$

Zárt formulában:

$$^{0}T_n = \prod_{i=1}^{n} {}^{i-1}D_i \qquad (2.22)$$

Megjegyzés: az R, H, D és T mátrixok indexelésének szabálya az, hogy bal felső indexbe mindig az aktuális vonatkoztató koordináta-rendszer száma, jobb alsó indexbe pedig az épp vonatkoztatott koordináta-rendszer száma írandó. Más féle indexelési módokkal is lehet találkozni a robotikával foglalkozó különböző szakirodalmakban és forrásokban, viszont az említett mód a legelterjedtebb. [2]

2.6 Relatív és abszolút csuklóorientációk jellemzői

A Denavit-Hartenberg eljárásnál a kinematikai lánc egyes csuklóihoz rögzített lokális koordináta-rendszerek mozgása között hierarchia van, tehát függőségi viszony. A kinematikai lánc rögzített alapjához közelebbi csuklók mozgása befolyásolja a távolabbi csuklók pozícióját és az általuk mozgatott szegmensek orientációját, tehát ebben az esetben, ha a példa kedvéért minden csukló rotációs típusú, akkor a legalsón kívül minden csuklókoordináta relatív szög, szomszédos szegmensek bezárt szöge, és nem abszolút szögek, nem a világ-koordinátarendszerbeli abszolút orientációk. A gyakorlat nyelvére fordítva ezt, képzeljünk el egy csuklókaros robot manipulátort, amely villamos hajtású, és a következő szegmenseket forgató robotcsuklók meghajtását az előző szegmensekre rögzített motorok biztosítják. Ennek a kinematikai következménye az, hogy relatív csuklószögekkel és hierarchiával kell számolni. A statikai és dinamikai következmény pedig az, hogy az alsó motorokat nagyobb nyomatéktartalékúra kell méretezni, mert azok a felettük rögzített összes motorral együtt kell, hogy üzembiztosan mozgassák a robotszegmenseket. A kinematikai hierarchia fennállása tehát a robot manipulátor mechanikai kialakításától függ, megvalósítható például olyan robot manipulátor is, amely kizárólag abszolútszögű csuklókkal rendelkezik. Utóbbi a gyakorlatban például úgy jelenhet meg, hogy a rotációs robotcsuklókat meghajtó motorok nem a szegmensekre vannak rögzítve, hanem egy felépítményen helyezkednek el, a tartó- és hajtónyomaték áttétele a csuklótengelyekre pedig szíjtárcsás vagy lánckerekes hajtóműveken keresztül valósul meg. A tervezési tapasztalatok azt mutatják, hogy ez térigényes megoldás, viszont statikailag és dinamikailag erőforrásgazdaságosabb, mert kisebb nyomatékú motorok is elegendőek a csökkent önteher és az esetlegesen nyomatéknövelő áttételképzés következtében. Előnye a megoldásnak még, hogy itt minden rotációs csuklónak a csuklókoordinátája világ-koordinátarendszerbeli abszolút orientáció. Ebben az esetben a Denavit-Hartenberg féle transzformáció közvetlenül nem alkalmazható, mert mint említettem, az relatív szögekkel operál, de így az alkalmazása nem is szükségszerű, mert egy ilyen robot manipulátor direkt és inverz kinematikája matematikailag egyszerűbben is leírható. Utóbbi a szükséges számítási teljesítmény szempontjából is előnyösebb, mert az egyszerűbb számítási módszerek általánosságban kevesebb erőforrást igényelnek.

3 Csuklókaros robotok inverz kinematikai számítási módszerei

3.1 A számítás nehézségei

Korábban már tettem arra említést, hogy a robot manipulátorok inverz kinematikai feladata analitikus módszerrel nehezen oldható meg a direkt kinematikai feladathoz képest, mert a legtöbb esetben több megoldás létezik és az inverz transzformációs folyamat hozzárendelése általában nem egyértelmű. Az inverz kinematikai feladat során trigonometrikus egyenletrendszert kellene több ismeretlenre megoldani, amelyet a kinematikai redundancia esetleges fennállása még inkább megnehezít, ugyanis ezesetben kevesebb trigonometrikus egyenlet áll rendelkezésre a rendszerben, mint amennyi ismeretlen csuklókoordináta, tehát szabadsági fok van.

Az inverz kinematikai feladatnak az analitikus módszerrel való megoldása mellett számos, eltérő gyorsaságú és pontosságú, numerikus, közelítő megoldási módszere létezik. Az említettek közül néhány bemutatásra kerül jelen fejezet hátralévő részében, annak elemzése érdekében, hogy a prototípus robotkar mozgásirányító szoftvere számára melyik alkalmazása célszerű.

3.2 Analitikus geometriai inverz kinematika

Ez a módszer az analitikus geometria eszközeit használja fel az inverz kinematika pontos megoldásának megtalálására, amely megoldásból adott esetben több is létezik, vagy paraméteres alakban kapjuk meg. Az analitikus geometriai inverz kinematikai feladat megoldásához az alábbi eszközökre és teendőkre van szükség: [2] [8]

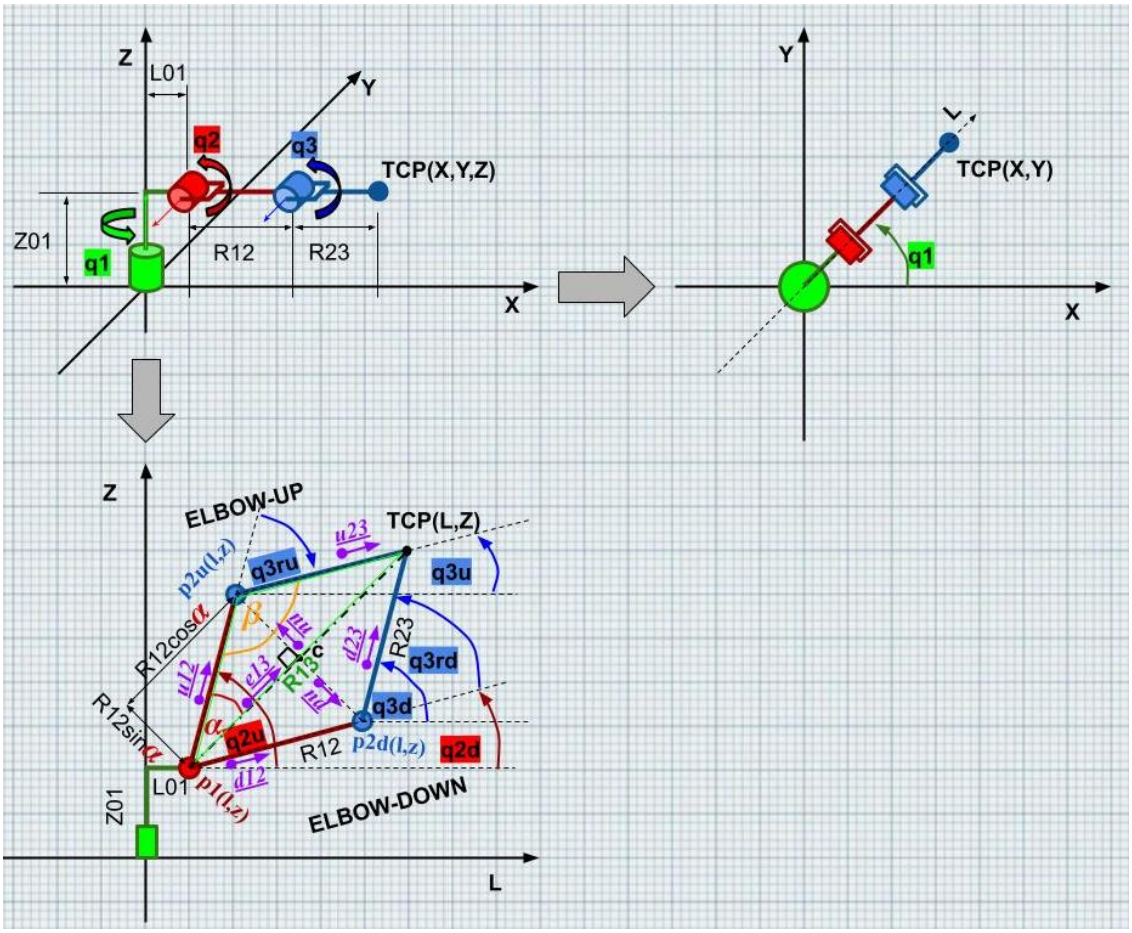
- Derékszögű és általános háromszögek felrajzolására a kinematikai diagramon a csuklók között. A kinematikai diagramnak, amely a robot manipulátort sematikusan ábrázolja a szabványos szimbólumokkal, több fajta nézetében is háromszögeket, vagy általánosabban szólva ismert geometriai tulajdonságokkal rendelkező sokszögeket kell keresni.
- Az euklideszi geometria axiómái, valamint a trigonometria tételei, összefüggései, szögfüggvényei és arcus-szögfüggvényei felhasználásával a

kinematikai lánc egyes csuklókoordinátái kifejezhetőek a világkoordináták, a robot manipulátor méretei és a csuklók elrendezésének függvényeiben.

- Háromszögek belső szögei közötti összefüggések.
- Általános háromszögekre a koszinusztétel, amely a háromszögek oldalai és szögei közötti általános összefüggést írja le.
- Derékszögű háromszögekre Pitagorasz tétele, mint a koszinusztétel speciális esete.
- Szinusz, koszinusz és tangens szögfüggvények.
- A szögek visszafejtésére alkalmas inverz szögfüggvények, tehát az arcus függvények
- A koordináta-geometria eszköztára. [8]

Ez a módszer minden esetben egyenletrendszer megoldását igényli, amely a legtöbb esetben trigonometrikus egyenleteket tartalmaz. Az analitikus geometriai inverz kinematikai feladat megoldhatósága és egyáltalán az alkalmazhatósága erősen függ a robot alapkonfigurációtól, tehát a robotcsuklók típusától, sorrendjétől és alaphelyzetétől, továbbá a robot geometriai kényszereitől. [2] [6] [8]

Lássunk egy példát a módszerre egy RRR csuklókaros alapkonfiguráció esetére.



9. ábra - RRR alapkonfiguráció kinematikai diagramja és vetületei

A mellékelt ábrán bal felül egy RRR alapkonfiguráció kinematikai diagramja látható a Descartes-féle világ-koordinátarendszerben. Jobbra ennek felülnézeti vetületén, az X-Y vízszintes síkbeli koordináta-rendszerben a q_1 szög meghatározása van ábrázolva. Az ábra alsó részén látható a térben a Z tengely körül q_1 szöggel elforgatott L-Z függőleges sík, amely egy általános oldalnézeti vetülete a világ-koordinátarendszernek, ahol az L tengely egy általánosított Z-re merőleges vízszintes tengely, amely q_1 szöget zár be az X tengellyel. Az L koordináták az X és Y koordinátákból Pitagorasz tételének alkalmazásával származtathatóak. A q_2 és q_3 szögek által orientált szegmensek minden körülmények között az L-Z síkban forognak, mert az említett robotcsuklók forgástengelyei az L-Z síkra merőlegesek.

A q_1 kiszámítható az alábbi trigonometrikus összefüggéssel, a célkoordináták felhasználásával: [8]

$$q_1 = \tan^{-1}\left(\frac{Y_{TCP}}{X_{TCP}}\right) \qquad (3.1)$$

A maradék két szög meghatározása már bonyolultabb, mert a függőleges, forgó, L-Z síkbeli koordináta-rendszerben, azonos síkban forgó két robotcsukló két féle szögbeállítással teheti lehetővé a TCP pont célpozícióra helyezését. Az egyik lehetőség az úgynevezett „elbow-up" [8] (felső könyökű) konfiguráció, a másik lehetőség az „elbow-down" [8] (alsó könyökű) konfiguráció. A szögek tekintetében a két konfiguráció eltérő eredményeket szolgáltat. Az elbow-up esetben a q_3 szög által irányított szegmens relatív orientációja a q_2 szög által orientált szegmenshez képest negatív szög, a könyökhajlat lefelé néz, míg elbow-down esetben a q_3 szög által irányított szegmens relatív orientációja a q_2 szög által orientált szegmenshez képest pozitív szög, tehát a könyökhajlat felfelé néz.

Az alábbiakban bemutatok egy lehetséges módot q_2 és q_3 szögek visszafejtésére. Az L-Z síkra tekintve, az elbow-up és elbow-down konfigurációk egy deltoidot határolnak, amely deltoid oldalhosszúságai a megfelelő szegmensek hosszával egyeznek meg (R_{12} és R_{23}). A deltoid főátlója (amely szimmetria tengely is egyben) a p_1 és p_3 (TCP) csuklók között húzódik, és a hossza a $v_{13} = p_3 - p_1$ irányvektor euklideszi normája: $\|v_{13}\| = R_{13}$.

$$v_{13} = p_3 - p_1 = \begin{bmatrix} L_{p3} - L_{p1} \\ Z_{p3} - Z_{p1} \end{bmatrix} \qquad (3.2)$$

A vektor normájának meghatározása: [9]

$$R_{13} = \|v_{13}\| = \|p_3 - p_1\| = \sqrt{\left(L_{p3} - L_{p1}\right)^2 + \left(Z_{p3} - Z_{p1}\right)^2} \qquad (3.3)$$

A p_2 csuklónak a fentiek értelmében két lehetséges pozíciója van, egy felső: p_{2u} és egy alsó: p_{2d}. A felsorolt négy pont feszíti ki a deltoidot. R_{12}, R_{23} és R_{13} két általános háromszöget alkotnak a deltoid szimmetria tengelyének felső és alsó oldalán, melyekre érvényes a koszinusztétel. Az ábrán α szimbólummal jelölt szög meghatározása a koszinusztétel megfordításával: [8]

$$R_{23}^2 = R_{12}^2 + R_{13}^2 - 2R_{12}R_{13}\cos\alpha \qquad (3.4)$$

$$\alpha = \cos^{-1}\left(\frac{R_{12}^2 + R_{13}^2 - R_{23}^2}{2R_{12}R_{13}}\right) \qquad (3.5)$$

A β szög meghatározása hasonló: [8]

$$R_{13}^2 = R_{12}^2 + R_{23}^2 - 2R_{12}R_{23}\cos\beta \qquad (3.6)$$

$$\beta = \cos^{-1}\left(\frac{R_{12}^2 + R_{23}^2 - R_{13}^2}{2R_{12}R_{23}}\right) \qquad (3.7)$$

Megállapítható a következő:

$$q_{3ru} = -q_{3rd} \qquad (3.8)$$

$$|q_{3ru}| = |q_{3rd}| = 180° - \beta \qquad (3.9)$$

Előjel szempontjából a q_3 relatív szög az elbow-up konfigurációban (q_{3ru}) és az elbow-down konfigurációban (q_{3rd}) ellentétesek, de abszolút értékük megegyezik, valamint elmondható, hogy a β mindkét szög esetében kiegészítő szög. [8]

Az abszolút szögek meghatározásához először meg kell találni a két lehetséges könyök pozíciót (p_{2u} és p_{2d}). Ebben az α szög szolgál segítségül. Az R_{12} hosszúságú szegmens α szerinti vetületeire és a p_3 és p_1 csuklópozíciók v_{13} különbségvektorára lesz szükség. A v13 vektort elosztva a normájával v13 irányba mutató e13 egységvektort kapunk. [9] [10] [11]

$$e_{13} = \frac{v_{13}}{R_{13}} \qquad (3.10)$$

A deltoidnak a c középpont pozícióvektorát úgy kapjuk, hogy p_1-ből indulva eltolást kell végrehajtani e_{13} irányba, R_{12}-nek a szimmetriatengelyre képzett vetületének megfelelő hosszúságban:

$$c = p_1 + e_{13} R_{12} \cos\alpha \qquad (3.11)$$

Ezt követően az α szöggel szemközti R_{12} vetület hosszúságban két eltolást kell végrehajtani az e_{13} két normálvektorának irányába (n_u: felső könyök felé mutató normálvektor és n_d: alsó könyök felé mutató normálvektor), hogy megkapjuk p_{2u} és p_{2d} pozícióvektorokat.

$$e_{13} = \begin{bmatrix} L_{e13} \\ Z_{e13} \end{bmatrix} \qquad (3.12)$$

$$n_u = \begin{bmatrix} -Z_{e13} \\ L_{e13} \end{bmatrix} \tag{3.13}$$

$$n_d = \begin{bmatrix} Z_{e13} \\ -L_{e13} \end{bmatrix} \tag{3.14}$$

Végül a könyökpozíciók meghatározása a deltoid középpontból induló normálvektoros eltolásokkal:

$$p_{2u} = c + n_u R_{12} \sin\alpha \tag{3.15}$$

$$p_{2d} = c + n_d R_{12} \sin\alpha \tag{3.16}$$

Innentől a csuklópozíciók különbségvektoraiból képzett egységvektorok kódolják a q_2 és q_3 szögek elbow-up (q_{2u} és q_{3u}) és elbow-down (q_{2d} és q_{3d}) változatait.

Az elbow-up egységvektorok:

$$u_{12} = \frac{v_{12u}}{\|v_{12u}\|} = \frac{p_{2u} - p_1}{R_{12}} = \begin{bmatrix} L_{u12} \\ Z_{u12} \end{bmatrix} \tag{3.17}$$

$$u_{23} = \frac{v_{23u}}{\|v_{23u}\|} = \frac{p_3 - p_{2u}}{R_{23}} = \begin{bmatrix} L_{u23} \\ Z_{u23} \end{bmatrix} \tag{3.18}$$

Az elbow-up egységvektorok:

$$d_{12} = \frac{v_{12d}}{\|v_{12d}\|} = \frac{p_{2d} - p_1}{R_{12}} = \begin{bmatrix} L_{d12} \\ Z_{d12} \end{bmatrix} \quad (3.19)$$

$$d_{23} = \frac{v_{23d}}{\|v_{23d}\|} = \frac{p_3 - p_{2d}}{R_{23}} = \begin{bmatrix} L_{d23} \\ Z_{d23} \end{bmatrix} \quad (3.20)$$

Végül az abszolút szögek meghatározása elbow-up konfigurációra, amennyiben egyik L koordináta sem nulla:

$$q_{2u} = \tan^{-1}\left(\frac{Z_{u12}}{L_{u12}}\right) \quad (3.21)$$

$$q_{3u} = \tan^{-1}\left(\frac{Z_{u23}}{L_{u23}}\right) \quad (3.22)$$

Elbow-down szögek:

$$q_{2d} = \tan^{-1}\left(\frac{Z_{d12}}{L_{d12}}\right) \quad (3.23)$$

$$q_{3d} = \tan^{-1}\left(\frac{Z_{d23}}{L_{d23}}\right) \quad (3.24)$$

A megoldások két könyök konfigurációja közül a pozícionáláshoz a kedvezőbbet kell kiválasztani a csukló- és munkatérbeli geometriai kényszerekhez viszonyítva. [8]

3.3 Az inverz kinematikai feladat numerikus megoldásai

Adott esetben megtehetjük azt, hogy az inverz kinematikai feladat pontos megoldását elengedjük, és helyette valamely numerikus módszerrel keresünk egy véges pontosságú közelítő becslést a csuklókoordináták vektorára. Ilyen esetekben általában szükség van egy q^0 kezdeti becslésre is, mint bemeneti paraméterre, amelyből elindulhat a közelítés. Vannak az inverz kinematika numerikus megoldására iteratív módszerek, melyek több, ismétlődő lépesben közelítik véges pontossággal a megoldást. Általánosan az alábbi összefüggés jellemzi a numerikus inverz kinematikát: [2]

$$q = f^{-1}(s, q^0) \tag{3.25}$$

3.3.1 Inverz sebességkinematika Newton módszerrel

A numerikus inverz kinematika legelterjedtebb megközelítése a Newton módszeren alapul, amely a differenciálszámítást hívja segítségül.

Adott a direkt kinematikai egyenlet a csuklókoordinátákkal és a világkoordinátákkal: [2]

$$s = f(q) \tag{3.26}$$

Mint már esett róla szó, s és q időfüggő vektorok (s(t) és q(t)). Differenciáljuk tehát az egyenletet az idő szerint: [2] [6]

$$\dot{s} = \left(\frac{\partial f}{\partial q}\right) \dot{q} \tag{3.27}$$

Ekkor a világkoordináták sebességvektora, azaz a pályasebesség-vektor és a csuklókoordináták sebességvektora, azaz a csuklósebesség-vektor közötti kinematikai összefüggést látjuk, ahol: [2] [6]

$$\frac{\partial f}{\partial q} = J(q) \qquad (3.28)$$

a Jacobi-mátrix, amely az f vektor-vektor függvénynek az összes csuklókoordináta szerinti első rendű parciális deriváltjait tartalmazza. Mivel f egy m-dimenziós függvényvektor, q pedig n-dimenziós vektor, így a Jacobi-mátrix m×n méretű, és a következőképpen alakul: [2] [6]

$$J(q) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} & \cdots & \frac{\partial f_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \frac{\partial f_m}{\partial q_2} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} \qquad (3.29)$$

A Jacobi-mátrix alakja nagyban függ a világ- és csuklókoordináták vektoraitól, valamint komoly jelentősége van az inverz kinematika numerikus megoldásában (természetesen a numerikus matematikát alkalmazó módszerek esetében), továbbá a robot dinamikájában és a pályatervezésben. [2]

Az előző sebességszintű direkt kinematikai egyenletre visszatérve és J(q)-val felírva: [2] [6]

$$\dot{s} = J(q)\dot{q} \qquad (3.30)$$

Mint, ahogy korábban volt rá példa, úgy most is rendezzük a direkt kinematikai egyenletet úgy, hogy megkapjuk az inverz kinematikai egyenletet. J(q)-val nem lehet csak úgy elosztani mindkét oldalt, mert mátrix, ezért szorozzuk meg mindkét oldalt balról $J^{-1}(q)$-val, tehát az inverz Jacobi-mátrixszal, amennyiben az inverz létezik: [2] [6]

$$J^{-1}(q)\,\dot{s} = J^{-1}(q)\,J(q)\,\dot{q} \qquad (3.31)$$

Ekkor a jobb oldalon a Jacobi mátrix és az inverzének szorzata egységmátrixot eredményez, amelyet a csuklósebesség-vektorral szorozva, a vektor önazonos marad, így az egyenlet egyszerűsödik: [2] [6]

$$J^{-1}(q)\,\dot{s} = \dot{q} \qquad (3.32)$$

Immár megoldható sebesség szinten az inverz kinematikai feladat és megkaphatóak a csuklósebességek. A csuklósebességek és az eltelt idő ismeretében pedig paraméteresen meghatározhatóvá válnak a csuklókoordináták adott pályapontokban.

A Jacobi-mátrix inverzének meghatározhatóságán áll vagy bukik e módszer. Az inverz akkor létezik, ha a mátrix kvadratikus, tehát négyzetes, valamint nem szinguláris, tehát a determinánsa nem zérus. Márpedig, ha a függvényvektor (és a világkoordináta-vektor) dimenziószáma eltér a szabadsági fokok számától, tehát a kinematikai lánc alul- vagy túlhatározott, akkor a Jacobi-mátrix nem lesz négyzetes, helyette téglalap-mátrix lesz. Ha a Jacobi-mátrix inverze nem létezik, akkor egy lehetséges orvoslata ennek a Moore-Penrose féle pszeudo-inverz meghatározása, amely így már csak közelítő megoldást eredményezhet, pontosat nem. [4]

A pszeudo-inverz számítás a következőképpen történik egy A tetszőleges mátrix esetében: [12]

$$A^{+} = (A^{T} \cdot A)^{-1} \cdot A^{T} \qquad (3.33)$$

ahol A-nak A^T a transzponáltja, A^+ pedig a pszeudo-inverz, avagy álinverz.

Matlab környezetben a mátrix bemeneti paraméterű inv(…); függvény segítségével képezhető inverz mátrix, valamint a pinv(…); függvény használatával képezhető pszeudo-inverz.

Az ipari robotikában a legtöbb esetben a Newton-módszert alkalmazzák az igen elterjedt, hat szabadsági fokú robotok inverz sebességkinematikájának megoldására, melyeknél 6×6 méretű a Jacobi-mátrix. Ez a módszer azonban, mint említettem, a Jacobi-mátix használata miatt érzékeny a robot konfigurációjára, amely a mátrix alakját

befolyásolja az m és n dimenziószámokon keresztül, valamint érzékeny a szingularitásra. [2] [6] A munkatérbeli szingularitások detektálhatóak, ha a Jacobi-mátrix determinánsa az adott pályasebességek mellett zérus. Az MSZ EN ISO 10218-1:2011 szabvány a következőképpen írja le a szingularitás (singularity) jelenségét:

"Olyan esemény, amikor a Jacobi-mátrix rangja kisebb lesz, mint a teljes rang. MEGJEGYZÉS: Matematikailag, egy szinguláris konfigurációban, a csuklósebesség a közös térben végtelenné válhat a Descartes-féle sebesség fenntartása érdekében. A tényleges működés során a Descartes-féle térben meghatározott mozgások, amelyek szingularitások közelében haladnak, nagy tengelysebességet eredményezhetnek. Ezek a nagy sebességek váratlanul érhetik a kezelőszemélyt" [1]

A szingularitáshoz közelítő Jacobi-mátrix inverzével való szorzás ahhoz hasonló jelenséget eredményez vektorok esetében, mint amikor egy skalárt egy végtelenül kicsi abszolút értékű (nullához tartó) számmal osztunk, melynek következtében a hányados végtelenhez tart.

3.3.2 Newton-Raphson iteráció

A Newton-Raphson eljárás egy iteratív, tehát ismétlődő lépésekből álló, numerikus módszer egy bonyolult, nemlineáris egyenlet gyökének véges pontosságú közelítésére. A közelítés végrehajtására a következőképpen kell eljárnunk: [13]

1) adott egy egyenlet, melynek az x gyökét keressük:
   bal oldali kifejezés = jobb oldali kifejezés,
2) nullára rendezzük az egyenletet:
   kifejezés = 0,
3) ekkor a kifejezés az x ismeretlennek valamilyen f(x) függvénye:
   f(x) = 0,
4) az átrendezett egyenlet alapján elmondható, hogy f(x) zérushelye lesz az egyenlet keresett gyöke, de az egyenlet klasszikus analízissel nem megoldható, tehát numerikus analízist kell alkalmazni,
5) helyettesítsünk be a függvénybe egy x_0 kezdeti becslést (initial guess), ekkor legnagyobb valószínűséggel azt tapasztaljuk, hogy: $f(x_0) \neq 0$,

6) deriváljuk a függvényt az ismeretlen szerint: df(x)/dx, majd a deriváltba is helyettesítsük be a kezdeti becslést: $df(x_0)/dx$, és az erre kapott érték lesz az $f(x_0)$ pont érintőjének meredeksége a függvény görbéjén,
7) Most kezdődjék az iteráció, közelítsük a becslést a Newton-Raphson iterációs egyenlet maximum n-szer ismételt alkalmazásával: [13]

$$x_{i+1} = x_i - \left(\frac{df(x_i)}{dx}\right)^{-1} \cdot f(x_i); \rightarrow i \in \{0; 1; 2; \dots; n\} \qquad (3.34)$$

ahol:

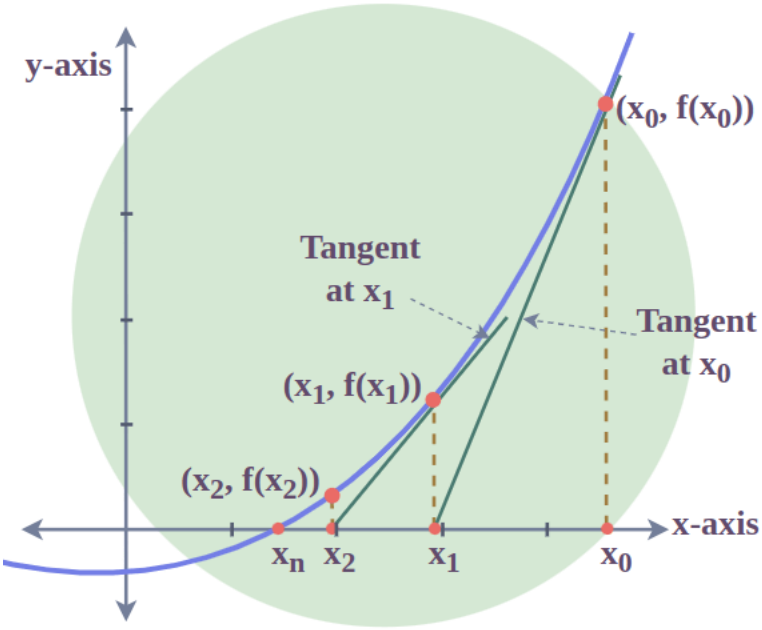
- i - az iterációs lépés sorszáma,
- n - az iterációs algoritmus maximális, megengedett lépésismétlési száma, ha a sorszám eléri a maximális ismétlési számot, akkor mindenképpen befejeződik a folyamat,
- x_i - a gyök aktuális becslése, a legelső lépésben ez az x_0 kezdeti becslés,
- x_{i+1} - a gyök következő, közelebbi becslése, a következő iterációs lépésben ez az érték kerül x_i helyére az iterációs egyenletben,
- az aktuális becsléssel behelyettesített függvényt minden iteráció során osztani kell az aktuális becsléssel behelyettesített deriválttal.

8) Az iterációs algoritmus célja a minél pontosabb közelítés, és mint minden iteratív folyamatnak, így ennek is kell kilépési feltétel. A Newton-Raphson algoritmus kilépési feltétele három feltétel logikai kapcsolatából áll össze: [13]

$$HA\left(\left((|f(x_i)| < \varepsilon)\text{ÉS}(|x_i - x_{i-1}| < \delta)\right) VAGY (i = n)\right) = IGAZ \qquad (3.35)$$

ahol: [13]

- ε - a függvény becsült zérushelyi értékének maximális, megengedett hibájának abszolút értéke, röviden a függő változó tűréshatára, zérustól való eltérése (error value),
- δ - az iterációs folyamatban az utolsó két zérushely becslés egymástól való maximális, megengedett eltérésének abszolút értéke, tehát a független változó tűréshatára (difference).
- a legpontatlanabb esetben a ciklust akkor szakítjuk meg, amikor az i ismétlési sorszám eléri az n maximális ismétlési számot, ekkor a zérushely becslés n-edik állapota, tehát x_n lesz az egyenlet közelítő megoldása, abban az esetben viszont, ha még az n maximális ismétlési szám elérése előtt mindkét fajta hiba kisebb lesz, mint a hibahatárok (ε és δ), akkor véget vetünk a közelítésnek, a zérushely becslésnek pedig i-edik állapota, tehát x_i lesz a közelítő megoldás.



10. ábra - Newton-Raphson módszer grafikus szemléltetése
A kép forrása:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.geeksforgeeks.org%2Fnewton-raphson-method%2F&psig=AOvVaw2PjGppVB57Jf5Fd4xmq-i-

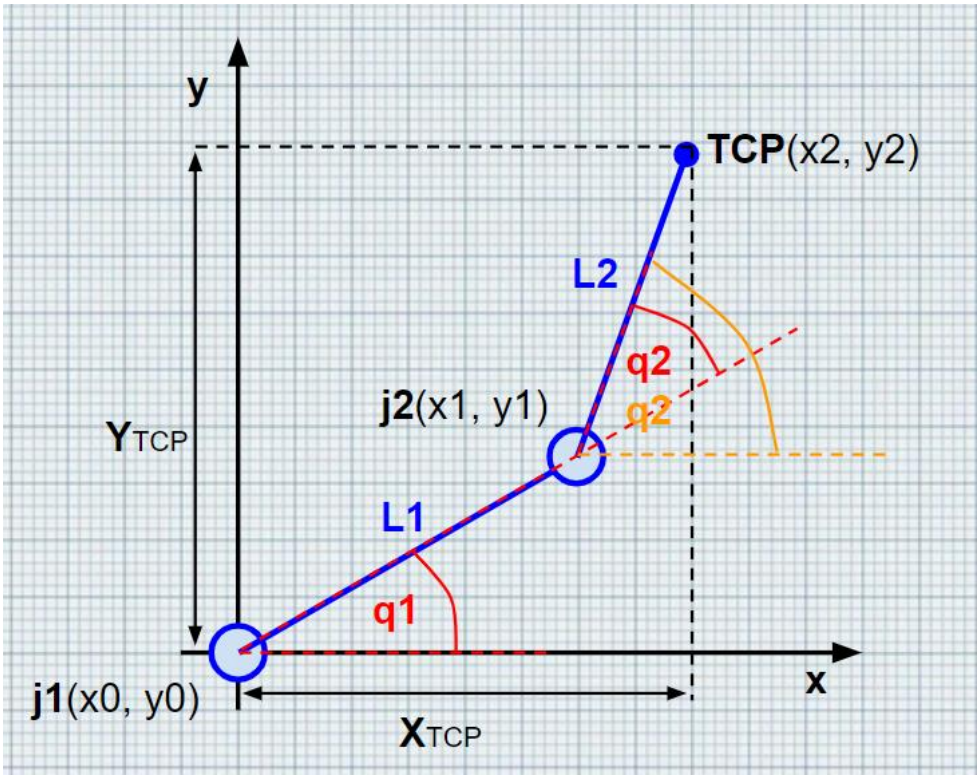
&ust=1700852220257000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCKjmiLXm2oIDFQAAAAAdAAAAABAE

Az iteratív algoritmus sikeres végeztével a végső becslés zérushelytől való eltérése a tűréshatárokon belülre kerül és kijelenthető, hogy megtaláltuk a lehetőségekhez mért legpontosabb közelített gyököt.

A Newton-Raphson eljárás az eddig ismertetett skaláris formájában nem alkalmazható az inverz kinematikai feladat megoldására, mert vektor-vektor függvényekre kell azt általánosítani. Az elv ugyanaz lesz, mint eddig, az iterációs egyenlet és az abban szereplő változók viszont szükségszerűen megváltoznak. [13]

Ebben az esetben több ismeretlenes, trigonometrikus egyenletrendszert kell megoldani, amelyet vektor egyenlet formájában írunk fel és rendezünk nullára.

Lássuk a módszer alkalmazását egy két szabadsági fokú, planáris manipulátor esetében.



11. ábra - 2DoF planáris manipulátor kinematikai diagramja

A mellékelt ábrán látható L_1 és L_2 jelölések a szegmensek hosszúságát jelentik, j_1 és j_2 pontok pedig a csukó pozíciókat. A TCP pozícióját a relatív értelmezésű (a síkbeli kinematikai diagramon piros színnel megjelenített q_2 a második csukló relatív szöge az azt megelőző szegmens irányához viszonyítva) csuklókoordináták függvényében leíró, két direkt kinematikai egyenletből álló egyenletrendszer a következő: [13]

$$\begin{cases} L_1 \cos q_1 + L_2 \cos(q_1 + q_2) = X_{TCP} \\ L_1 \sin q_1 + L_2 \sin(q_1 + q_2) = Y_{TCP} \end{cases} \quad (3.36)$$

A létrejött egyenletrendszert nullára kell rendezni: [13]

$$\begin{cases} L_1 \cos q_1 + L_2 \cos(q_1 + q_2) - X_{TCP} = 0 \\ L_1 \sin q_1 + L_2 \sin(q_1 + q_2) - Y_{TCP} = 0 \end{cases} \quad (3.37)$$

Abszolút értelmezésű szögekre (narancssárga színnel jelölt q_2 a mellékelt síkbeli kinematikai diagramon) a következő módon egyszerűsödik az eredeti és a nullára rendezett egyenletrendszer:

$$\begin{cases} L_1 \cos q_1 + L_2 \cos q_2 = X_{TCP} \\ L_1 \sin q_1 + L_2 \sin q_2 = Y_{TCP} \end{cases} \quad (3.38)$$

$$\begin{cases} L_1 \cos q_1 + L_2 \cos q_2 - X_{TCP} = 0 \\ L_1 \sin q_1 + L_2 \sin q_2 - Y_{TCP} = 0 \end{cases} \quad (3.39)$$

A Denavit-Hartenberg eljárásnál már tettem említést a relatív és abszolút értelmezésű szögek jelentőségéről. Ezek lényege a robot mechanikai kialakításában van, és abban, hogy van-e hierarchiája a kinematikai lánc csuklóinak. Robotirányítási szempontból ez egy fontos megkülönböztetés.

Szóval az első egyenlet bal oldali kifejezése legyen mostantól az $f_1(q_1, q_2)$ kétváltozós függvény, a második egyenlet bal oldali kifejezése pedig legyen $f_2(q_1, q_2)$ kétváltozós függvény. Tömörítsük a leírást azzal, hogy a függvények bemeneti paramétere legyen a q csuklókoordináták vektora. Így az egyenletrendszer: [13]

$$\begin{cases} f_1(q) = 0 \\ f_2(q) = 0 \end{cases} \tag{3.40}$$

ahol: [13]

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{3.41}$$

Legyen f(q) vektor az $\{f_1(q), f_2(q)\}$ függvényrendszer vektora, ezáltal nemlineáris $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ vektor-vektor függvény, amely folytonosan deriválható. [13]

$$f(q) = \begin{bmatrix} f_1(q) \\ f_2(q) \end{bmatrix} \tag{3.42}$$

A végső, kompakt alakú vektor egyenlet tehát: [13]

$$f(q) = 0 \tag{3.43}$$

ahol a q zérushelyek vektorát keressük.

Az eddigiek fényében az iterációs egyenlet vektor-vektor függvényekre általánosított formája a következőképpen alakul: [13]

$$q_{i+1} = q_i - \left(\frac{\partial f(q_i)}{\partial q}\right)^{-1} \cdot f(q_i); \rightarrow i \in \{0; 1; 2; \dots; n\} \tag{3.44}$$

Figyelem! Az i indexelés itt nem a csuklókoordináták indexét jelenti, hanem az iterációs folyamat ismétlési sorszámát, tehát a csuklókoordináták vektorának iterációit jelöli, a kettő nem ugyanaz! Az egyenletben látható differenciál kifejezés pedig a J(q) mátrix, tehát a korábbiakban már megismert Jacobi-mátrix, amelynek az egyenletben az inverzét kell, hogy képezzük. Így az iterációs egyenlet: [13]

$$q_{i+1} = q_i - J(q_i)^{-1} \cdot f(q_i); \rightarrow i \in \{0; 1; 2; \dots; n\} \qquad (3.45)$$

ahol: [13]

$$J(q_i) = \frac{\partial f(q_i)}{\partial q} = \begin{bmatrix} \frac{\partial f_1(q_{1i}, q_{2i})}{\partial q_1} & \frac{\partial f_1(q_{1i}, q_{2i})}{\partial q_2} \\ \frac{\partial f_2(q_{1i}, q_{2i})}{\partial q_1} & \frac{\partial f_2(q_{1i}, q_{2i})}{\partial q_2} \end{bmatrix} \qquad (3.46)$$

A Jacobi-mátrix tartalmazza a függvényrendszer minden függvényének, minden csuklókoordináta szerinti parciális deriváltjait. Visszatekintve a két szabadsági fokú, planáris manipulátor eredeti egyenletrendszereire, és alkalmazva a trigonometrikus függvények deriválási szabályait, felírható a Jacobi-mátrix: [13]

relatív szögekre:

$$J(q) = \begin{bmatrix} -L_1 \sin q_1 - L_2 \sin(q_1 + q_2) & -L_2 \sin(q_1 + q_2) \\ L_1 \cos q_1 + L_2 \cos(q_1 + q_2) & L_2 \cos(q_1 + q_2) \end{bmatrix} \qquad (3.47)$$

és abszolút szögekre:

$$J(q) = \begin{bmatrix} -L_1 \sin q_1 & -L_2 \sin q_2 \\ L_1 \cos q_1 & L_2 \cos q_2 \end{bmatrix} \qquad (3.48)$$

Immár elméletileg minden adott a Newton-Raphson algoritmus megvalósításához és két szabadsági fokú, planáris manipulátor inverz kinematikai problémájának megoldására

való alkalmazásához. Az iteratív közelítési folyamat a kilépési feltételrendszer kielégülésével véget ér és megkapjuk a célhelyzet elérése érdekében beállítandó csuklókoordináták q vektorának véges pontosságú becslését, ezzel megoldva az inverz kinematikai feladatot.

A Newton-Raphson algoritmust általánosan az jellemzi, hogy meglehetősen gyors és pontos, a tapasztalatok azt mutatják, hogy csak nagyon ritka esetekben éri el az ismétlési határt, az esetek legnagyobb többségében jóval előbb teljesül a kilépési feltétel a tűréshatárok kielégülésével. Előny, hogy nem kell az inverz kinematikai problémát sebességkinematikai szintre emelni, tehát nem az idő szerinti deriváltakat kell képezni, ami némileg egyszerűsíti a számítandó egyenleteket. [13]

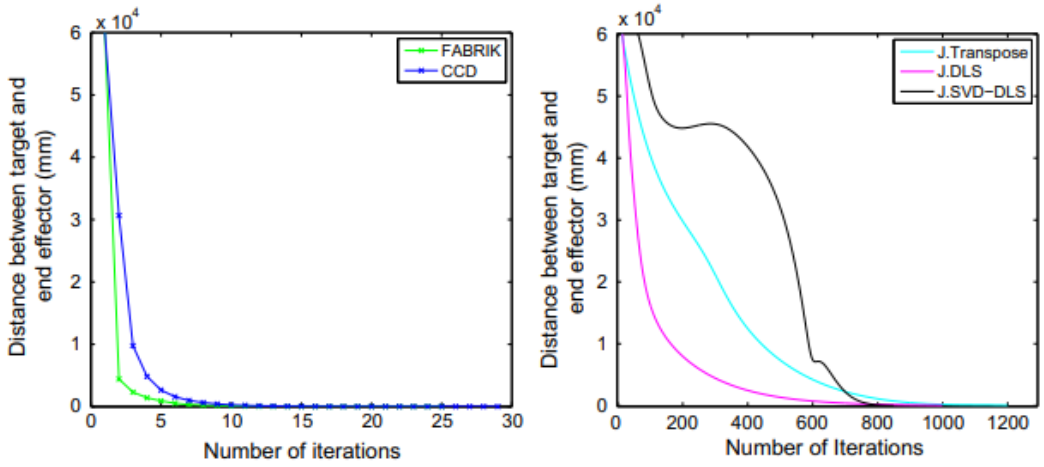
A Newton-Raphson módszernek azonban vannak kellemetlen, általános tulajdonságai is, és ezek a következők:

- az iterációs közelítési folyamat időtartama és stabilitása erősen függ a kezdeti becsléstől, ha túl távoli a becslés, akkor sok iteráció, tehát hosszú közelítési idő, vagy rosszabb esetben divergencia léphet fel, tehát az algoritmus nem talál megfelelő közelítő megoldást, [13]
- Az algoritmus vonzódik a függvény lokális minimumaihoz, ezeknél elakad és divergál, [13]
- Az algoritmus becslései hurokba tudnak ragadni, és a becslés értéke oszcillálni kezd a függvény inflexiós pontjainak környezetében, [13]
- ha a közelítés a függvény (monoton függvény) egy konstans szakaszára ér, akkor nullázódik a deriváltja, ennek következményeként pedig az iterációs egyenlet használhatatlanná válik, megreked a folyamat a nullázódás előtti becslésnél, és nem képes új becsléseket kitermelni, [13]
- ha a Jacobi-mátrix szinguláris válik a zérus Jacobi-determináns miatt az iterációs folyamat közben, akkor az inverze nem létezik, az ilyen esetekre tekintettel egy ciklusba ágyazott, feltételes elágazással detektálni kell a szingularitást, és biztosítani kell, hogy ilyen esetben az inverz helyett pszeudo-inverz Jacobi-mátrix számítás történjen, ha erre kerül sor, az pontatlanabb eredményhez vezet. [4]

Összegezve: a Newton-Raphson algoritmus érzékeny a kezdeti becslés(ek)re, a függvény(ek) alakjára és a szingularitásra, a differenciálképzés és az inverz mátrix képzés miatt.

3.3.3 FABRIK módszer

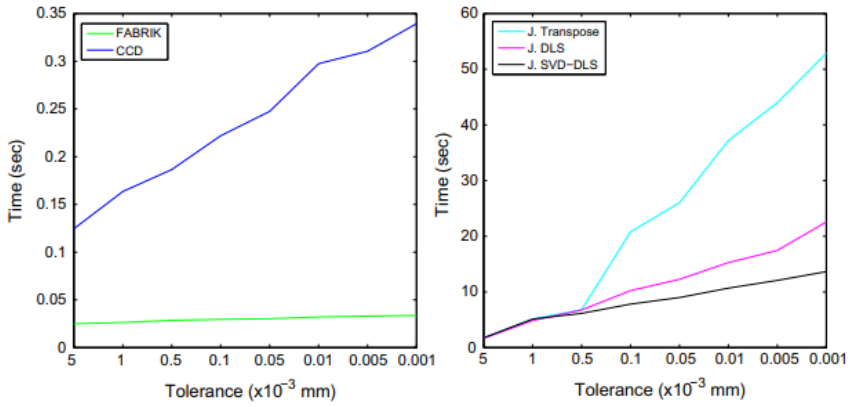
Az inverz kinematikai probléma megoldása nem kizárólagosan az ipari robotikának sajátja, hanem vannak más alkalmazási területek, mint például a számítógépes karakter-animáció, ahol szintén inverz kinematikai problémát kell megoldani az animált karakter mozgásának ábrázolásának lehetővé tétele céljából. Utóbb említett területen léteznek úgynevezett heurisztikus keresési stratégiák alapján működő iteratív IK solver (Inverz Kinematika Megoldó) algoritmusok, mint pédául a Jacobian Transpose módszer, Jacobian DLS módszer, a Jacobian pseudo-inverse DLS módszer, a háromszögelés, az FTL, azaz Follow the Leader eljárás, a CCD – Cyclic Coordinate Descent – algoritmus, továbbá a FABRIK - Forward and Backward Reaching Inverse Kinematics – eljárás, melyek közül a szükséges iterációk számának szempontjából a leggyorsabb, legpontosabb és legmegbízhatóbb módszer a FABRIK. Az alábbi ábra szemlélteti, hogy az egyes IK solverek hány iterációval, milyen pontosan képesek megközelíteni a célpozíciót: [10] [11] [14]



12. ábra - Heurisztikus IK algoritmusok közelítési sebessége [14]

Látható, hogy a FABRIK messze a legkevesebb iterációval képes teljesíteni a tolerancia alatti pontosságot. A következő ábrán látható karakterisztikák az egyes IK solverek

esetén a szükséges közelítési idő növekedésének mértékét ábrázolják a tolerancia szűkítése függvényében.



13. ábra - Heurisztikus IK solverek közelítési idejének tolerancia-érzékenysége [14]

A FABRIK eljárás közelítési ideje szinte érzéketlenséget mutat a tolerancia szűkülésére, ellentétben más IK solverekkel. Ezek az adatok és összehasonlító mérések meggyőztek arról, hogy a heurisztikus inverz kinematika megoldó algoritmusok közül idő- és pontosságkritikus alkalmazásokban egyértelműen a FABRIK módszert kell felhasználni. [14]

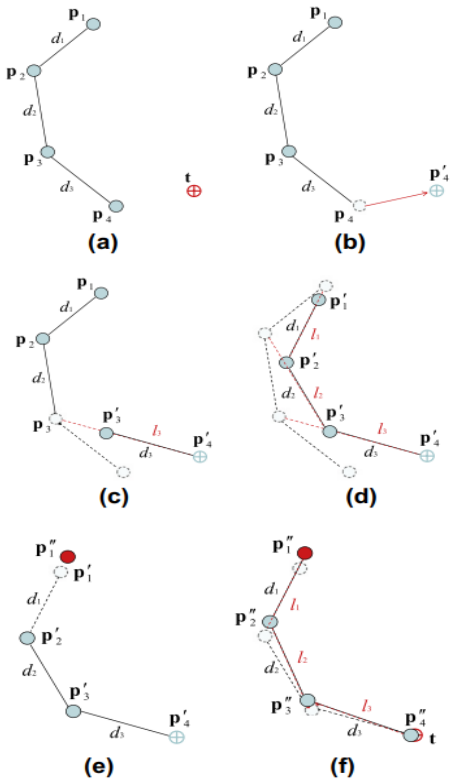
Összegezve a FABRIK jellemző tulajdonságait, a következők mondhatóak el:

- A FABRIK módszer nem alkalmaz differenciál-számítást és mátrix műveleteket, így nincs feladatunk a Jacobi-mátrixszal, sem az inverzével, sem a pszeudo-inverz Jacobi-mátrixszal, sem a szingularitásokkal. [10]
- A módszer iteratív, ismétlődő lépéssorozatokból épül fel, melyek rövidesen bemutatásra kerülnek. [10]
- Ezen eljárás is igényel egy kezdeti becslés készletet, viszont a Newton-Raphson algoritmussal ellentétben, teljes mértékben érzéketlen a közelítési idő és a közelítés stabilitása a kezdeti becslések értékeire. A FABRIK algoritmus csuklókaros robotra való alkalmazása esetében a becslések nem szögek, tehát nem a csuklókoordináták, hanem a robotcsuklóknak a nyugvó, globális koordináta-rendszerben értelmezett abszolút pozíciói, tehát a módszer mellőzi a lokális koordinátarendszerek használatát. Az előző értelmében a FABRIK

használata előnyösebb abszolút-szögű csuklókaros robot esetén, mint relatív szögű változatnál. [10]

- További előnye a Newton-Raphsonnal szemben, hogy nincs hajlama a divergenciára, elakadásra és oszcillációra, utóbbi a CCD-vel szemben is előnye, mivel az képes a cél közvetlen környezetében nagyon hosszan oszcillálni, mire teljesül a tolerancia feltétel. Ennek oka, hogy a CCD szög becslésekkel operál, míg a FABRIK pozíciókkal. Egyetlen ritka esetben tud elakadni a FABRIK, mégpedig akkor, ha nagyon közeli a cél, és e cél irányával azonos irányban van az összes kezdeti csuklópozíció, tehát ki van nyújtva a kinematikai lánc. [10] [11] [13]
- A FABRIK algoritmust nem zavarja össze a kinematikai redundancia és az inverz kinematikai feladat adott esetben végtelen sok megoldása sem, sőt bonyolult, összetett kinematikai láncokra is alkalmazható, melyeknek például több effektora van, ilyen az emberi kéz is. [14]
- Ez az algoritmus a heurisztikus keresési stratégiák között egy úgynevezett greedy best-first search módszer, tehát egy „mohó legjobbat-először kereső” módszer. [15] Ennek tényéről az alfejezet későbbi részében megbizonyosodhatunk.
- Nagyon egyszerű az alkalmazása, könnyen bővíthető, alkalmazható síkban és térben is, valamint az alkalmazás csak koordináta-geometriai és trigonometria összefüggések, továbbá a vektorműveletek ismeretét követeli meg. [10]

Az alábbi ábra bemutatja a FABRIK alkalmazását három szabadsági fokú, síkbeli csuklókaros manipulátorra.



14. ábra - FABRIK lépései [14]

A FABRIK végrehajthatóságának feltétele, hogy a cél távolsága a globális origótól nem lehet nagyobb, mint a manipulátor szegmenshosszainak összege, különben a cél túl messze van, így a manipulátor effektora azt képtelen elérni, kívül van a munkatér határfelületén. [10]

A FABRIK algoritmus minden iterációja két fázisból áll. Az adott iteráció első fázisa a forward-phase, tehát előre-fázis, amelynek a végrehajtását a backward-phase, avagy a hátra-fázis végrehajtása követi. [10]

A kiindulási csuklópozíciók legyenek a p_1, p_2, p_3, p_4 pozícióvektorok, valamint legyen t, mint target a célpozícióvektor. A pozícióvektoroknak ebben a síkbeli példában x és y világkoordinátáik vannak, továbbá legyenek d_1, d_2, d_3 az egyes szegmensek skalár hosszúságai sorban. A p_1 az origóban van, tehát nullvektor: $p_1 = [0\ 0]^T$. [10]

Az előre-fázis rögtön a cél elérésével indít, ezért is mohó az algoritmus, mert az effektort rögtön az első lépésben a célpozícióra helyezzük. Természetesen ez önmagában sértené a szegmenshosszok jelentette geometriai kényszereket, ezért az effektor előtti csuklót is el kell mozdítani az effektor irányába, a két csukló közötti szegmenshossz állandóságát fenntartva. Az ezek előtti ízületeknél is így járunk el, mindig módosítva a csuklópozíciókat, de mindig állandónak megőrizve a szegmensek hosszát, amíg el nem jutunk a legelső csuklóig. A pozíciómódosítások az effektor felől a legelső csukló felé haladnak. Az előre-fázis végeztével a tapasztalatunk az, hogy az effektor beért a célba, az első csukló viszont elemelkedett a rögzített alaptól, kvázi lebeg a manipulátor, természetesen ez nem végleges eredmény. [10]

A pozíciómódosítások menete előre-fázisban, ahol a módosított pozíciókat egy aposztróffal szokás jelölni: [10]

$$p_4' = t \tag{3.49}$$

$$p_3' = p_4' + e_{43} \cdot d_3 \tag{3.50}$$

$$p_2' = p_3' + e_{32} \cdot d_2 \tag{3.51}$$

$$p_1' = p_2' + e_{21} \cdot d_1 \tag{3.52}$$

Ahol az egyes e egységvektorok a pozíció-eltolás irányát, a d szegmenshossz skalár szorzótényezők pedig az eltolás nagyságát határozzák meg. Az egységvektorok

előállíthatóak, ha vesszük a már eltolt csukló pozícióvektorát és az éppen módosítandó pozícióvektort, és előállítjuk a különbségvektorukat, amely módosítandó csukló eredeti pozíciója felé mutat, általánosan megfogalmazva az előre-fázis esetén a kisebb indexű csuklók felé mutat, hátra-fázisnál pedig a nagyobb indexű csuklók irányába mutat. Az aktuális különbségvektornak és euklideszi normájának, tehát saját hosszának hányadosát képezzük, így megkapjuk az e egységhosszú irányvektort, amelyet az adott szegmens d hosszúságára skálázunk, ezáltal megvalósítva az eltolást. Az egységvektorok meghatározásának általános formulája, ahol a csuklóindexek i és j, melyek közül i az előzőleg módosított pozíciójú csukló indexe, j pedig az éppen módosítandó pozíciójú csukló indexe, és előre-fázisban j = i-1, hátra-fázisban pedig j = i+1. Ebben az általánosításban az aktuális egységvektor mindig a j indexű csukló pozíciója felé mutat. [10] [11]

$$e_{ij} = \frac{p_j - p_i}{\|p_j - p_i\|} = \frac{1}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \cdot \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \qquad (3.53)$$

Az euklideszi normát Matlab környezetben a norm(…); függvény segítségével lehet egyszerűen kiszámíttatni.

Az előre-fázist a hátra-fázis követi, amelynek fordított a műveleti iránya, tehát a legelső csuklót visszatesszük az eredeti pozíciójába, tehát a rögzített alaphoz, majd a korábban ismertetett pozíciómódosításokat az első csuklótól az effektor felé haladva, és a szegmensek hosszának állandóságát fenntartva végezzük el. Ezek után a manipulátor nem lebeg, az effektor pozíció pedig eltér a célpozíciótól, ám sokkal kisebb mértékben, mint a most leírt első iteráció előtt. Itt a módosított pozíciókat két aposztróffal szokás jelölni. [10]

$$p_1'' = p_1 \qquad (3.54)$$

$$p_2'' = p_1'' + e_{12} \cdot d_1 \qquad (3.55)$$

$$p_3'' = p_2'' + e_{23} \cdot d_2 \qquad (3.56)$$

$$p_4'' = p_3'' + e_{34} \cdot d_3 \qquad (3.57)$$

Az előre- és hátra-fázisok után a következő iteráció kiindulási pozícióivá tesszük jelen iteráció két aposztrófos pozícióit, ahol i a csuklóindex, k pedig a folyamat ismétlési sorszáma. [10]

$$p_i(k+1) = p_i''(k) \qquad (3.58)$$

A továbbiakban számos ilyen iteráción keresztül folyamatosan közelítjük az effektor pozícióját a célpozícióhoz egészen addig, amíg meg nem haladjuk az ismétlési limitet, vagy ez előtt el nem érjük a világkoordinátákban értelmezett tolerancia alatti pontosságot, tehát a pozícióhibát a toleranciánál kisebb értékre csökkentjük világ-koordinátánként. Jogosan merül fel a kérdés, hogy a pozíciókból hogyan lesznek a folyamat végén visszafejtett csuklószögek, azaz csuklókoordináták, tehát hogyan jutunk hozzá az inverz kinematikai feladat megoldásához. A véges pontossággal megközelített csuklószögek az utolsó lezajlott iteráció hátra-fázisának egységhosszú irányvektoraiban vannak kódolva, tulajdonképpen e vektorok abszolút orientációi az abszolút értelmű csuklókoordináták. [10]

$$q_1 = \tan^{-1}\left(\frac{y_{e_{12}}}{x_{e_{12}}}\right) \qquad (3.59)$$

$$q_2 = \tan^{-1}\left(\frac{y_{e_{23}}}{x_{e_{23}}}\right) \qquad (3.60)$$

$$q_3 = \tan^{-1}\left(\frac{y_{e_{34}}}{x_{e_{34}}}\right) \qquad (3.61)$$

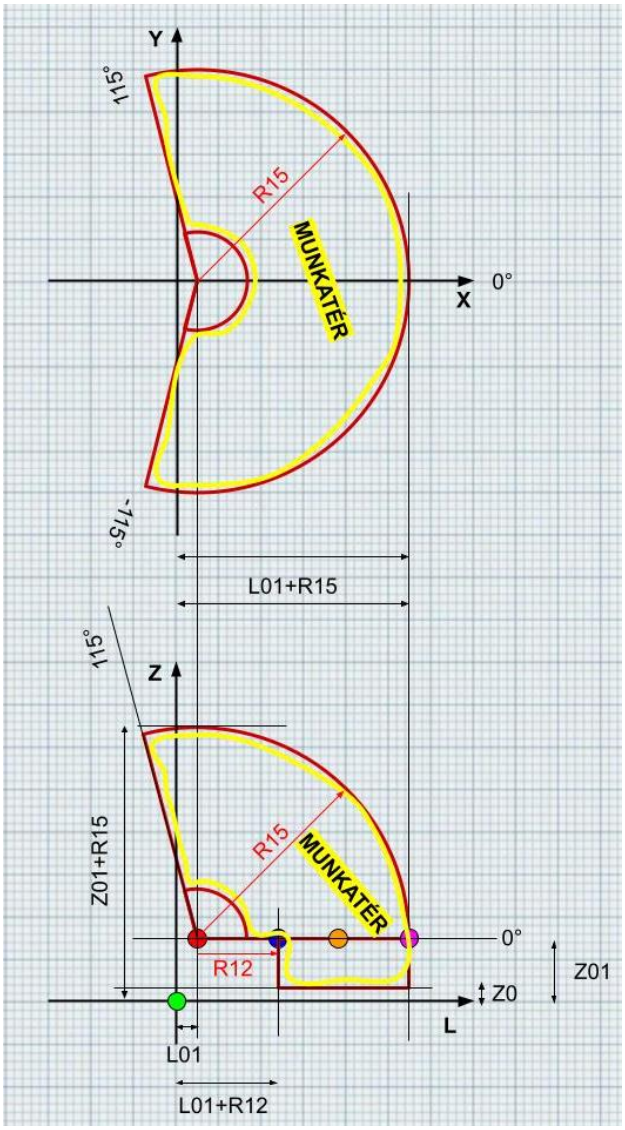
Természetesen itt az algoritmusban szükséges vizsgálni a nevezőben lévő koordinátákat, mert a zéróosztó értelmezhetetlen bemeneti paramétert eredményez az arcustangens függvénynek. Szerencsére például a Matlab szoftver atan2(y, x); függvénye el van látva a megfelelő zéróosztó- és előjeltesztelő feltételekkel, így ezen célra konyhakészen felhasználható.

4 Prototípus csuklókaros robot tervei és adatai

4.1 Általános leírás

A korábbi félévek során általunk megtervezett és mostanra majdnem teljesen megépült prototípus robot egy öt szabadsági fokú, RRR alapkonfigurációjú, csuklókaros robot, melynek tehát minden robotcsuklója rotációs típusú. A robot manipulátor előzetesen tervezett karhosszúsága fél méter, a megépült robotkar esetében a megfogó szerkezettel együtt ez 444,61 mm hasznos karhossz, mivel a hajtóművekben alkalmazott gyári szíjhosszúságok és tárcsaméretek befolyásolták a mechanikai tervezésnél a végleges tengelytávolságokat. A prototípusnak a korábbi félévek projektmunkáiban specifikált feladata, egy néhány centiméterszer néhány centiméterszer néhány centiméter méretű, hasáb vagy henger alakú munkadarab megfogása, üzembiztos mozgatása és elengedése. A szállítandó munkadarab előzetesen specifikált tömege a néhány tíz grammos tartományban volt meghatározva, ez azonban a tervezési folyamat során és a motorterhelhetőségi számítások elvégzése után elérte a száz grammos elméleti határt. A robot minden csuklója és a megfogó mechanizmus is villamos hajtású. A TCP előzetes pozícionálási pontosságára vonatkozó követelményt milliméter alatti pontosságúnak állapítottuk meg. A robot a tervek szerint világkoordinátákban megadott pályapontokkal kell, hogy programozható legyen, ez az inverz kinematikai feladat megoldását igényli, az inverz kinematika megoldó algoritmus tehát a mozgásirányítás központi műveletét testesíti meg. Jelen szakdolgozat célja ezen megoldó algoritmus tervezése és fejlesztése, amely a jövőben a mozgásirányító szoftverrendszer legfontosabb rendszerelemeként fog szolgálni.

4.2 Munkatér vetületei



15. ábra - Munkatér vetületei

Az ábrán látható két koordináta-rendszer a munkatér vetületei. Felül látható a felülnézeti X-Y sík, alul pedig a forgó oldalnézeti L-Z sík, ahol az L tengely a törzs (Z) körüli q_1 szögű forgatásra általánosított vízszintes tengely. Az L koordináták az X és Y koordinátákból Pitagorasz tételével számíthatóak, megfordítva az X koordináta $Lcosq_1$, az Y koordináta $Lsinq_1$ összefüggésekkel számítható vissza. Az L-Z sík origója egybeesik az X-Y sík origójával a térben és ezzel az X-Y-Z világ-koordinátarendszer origójával. Az L-Z sík a Z tengely körül forog, L tengely q_1 szöget zár be az X tengellyel. A q_1 szög a robot törzsének csuklókoordinátája. A zöld pont a törzs

középpontját, a piros pedig az L-Z síkra merőleges váll tengelyt jelöli, míg a többi más színű pont is a csuklótengelyeket jelöli. Az X-Y síkon a legyező szerű síkidom a legális munkatér függőleges vetülete, határait a kinyújtott kar hossza, valamint a q_1 szög abszolút orientációshatárértékei szabják meg. Az L-Z sík legyező szerű síkidomja a legális munkatér q_1 szög szerint elforgatott oldalnézeti vetülete, melynek határait a kinyújtott karhossz, a q_2 váll szög abszolút orientációs határértékei, valamint a földdel való ütközést elkerülendően a Z koordináta minimum értéke szabják meg. A munkatér ezen vetületeit az inverz kinematika megoldó szoftverben a számítási eredmények ellenőrzésére használom fel. Az alábbi táblázat tartalmazza a konkrét méreteket, amelyekkel meghatározhatóak a munkatér határai:

4.1. táblázat - Munkatérbeli méretek

Jel	**Jelentés**	**Méret [mm]**
L01	Törzs középpont és váll tengely L tengelyen mért távolsága	49,5
Z01	Törzs középpont és váll tengely Z tengelyen mért távolsága	145,8
Z0	Munkatér alsó határfelületének magassága	30
R15	Válltól TCP-ig kinyújtott karhossz	444,61
R12	Váll és könyök tengelytávolsága, felkar szegmenshossz	182
R23	Könyök és csukló-bólintó tengelytávolsága, alkar szegmenshossz	116
R35	Csukló-bólintó tengely és TCP közötti távolság	146,61

4.3 Csuklótér határszögei

Korábban már többször említettem a relatív és abszolút orientációkat megvalósító mechanikai kialakításokat, nem véletlenül. A prototípus robotkar csuklóira a léptetőmotorok által szolgáltatott tartó- és hajtónyomaték, nyomaték növelő – sebesség csökkentő áttételeket megvalósító, bordásszíjtárcsás és fogaskerekes hajtóműveken keresztül jut el. Ez azt jelenti, hogy a váll alatti felépítményen kerültek elhelyezésre a vállat, könyököt, és csukló-bólintást működtető motorok, nem a karon, tehát az említett három motor abszolút szögként vezérli a q_2, q_3 és q_4 csuklókoordinátákat a függőleges L-Z síkon, tehát ezek között a szögek között nincs az orientáció tekintetében

kinematikai hierarchia. Számítással azonban elő kell állítani q_3 és q_4 relatív orientált változatait, hogy azok felhasználásával az inverz kinematika megoldó algoritmusban a szöghatárok mérhetőek és ellenőrizhetőek legyenek, mert e két szög esetén az abszolút orientációknak nem definiálhatóak állandó határok.

Az alábbi táblázat a robotcsuklók határszögeit rögzíti:

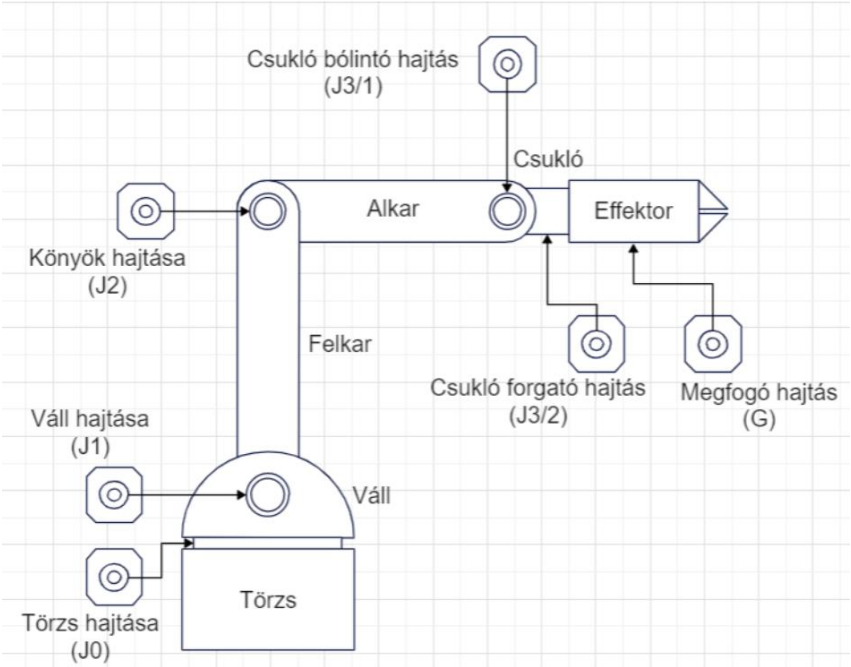
4.2. táblázat - Csuklótér határparaméterei

Minimum [°]	**Csuklókoordináta: q_i**	**Maximum [°]**
-115	q_1 (Törzs)	+115
0	q_2 (Váll)	+115
változó	Abszolút q_3 (Könyök)	változó
-115	Relatív q_{3r} (Könyök)	+115
nincs	$\|q_{3r}\| + \|q_{4r}\|$	180
változó	Abszolút q_4 (Csukló-bólintás)	változó
-115	Relatív q_{4r} (Csukló-bólintás)	+115
-90	q_5 (Csukló-csavarás)	+90

A táblázatban a citromsárgával jelölt sorban az az ellenőrző feltétel található, amely megakadályozni hivatott a kinematikai hurok képződést, tehát a szegmensek keresztezését, az önütközést. A q5 csukló-csavaró szög az egyetlen valóban relatív orientációjú szög, mert az azt vezérlő unipoláris léptetőmotor a csukló szegmensbe van beépítve.

4.4 Mechanika

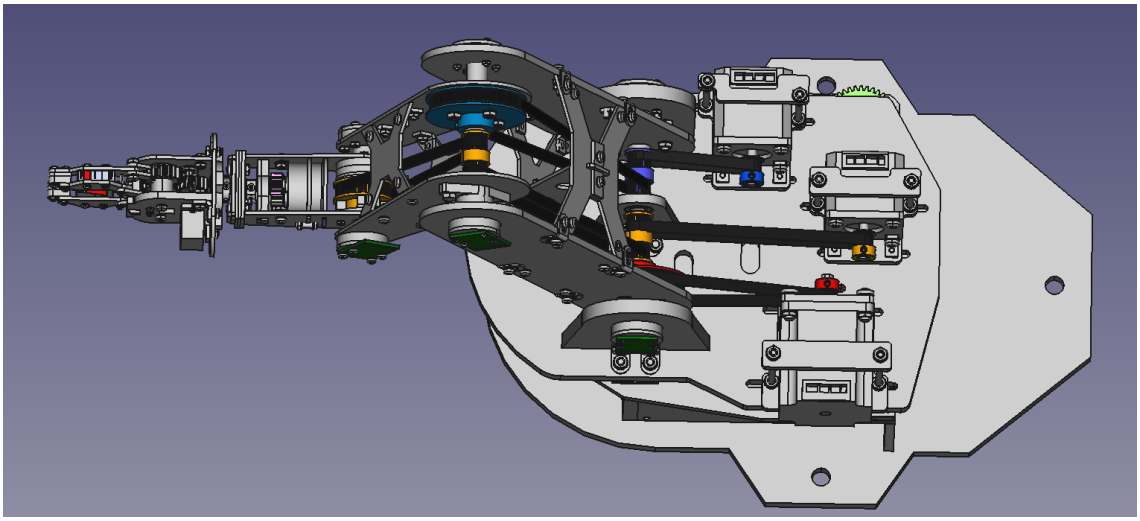
4.4.1 Mechanikai logikai rendszerterv



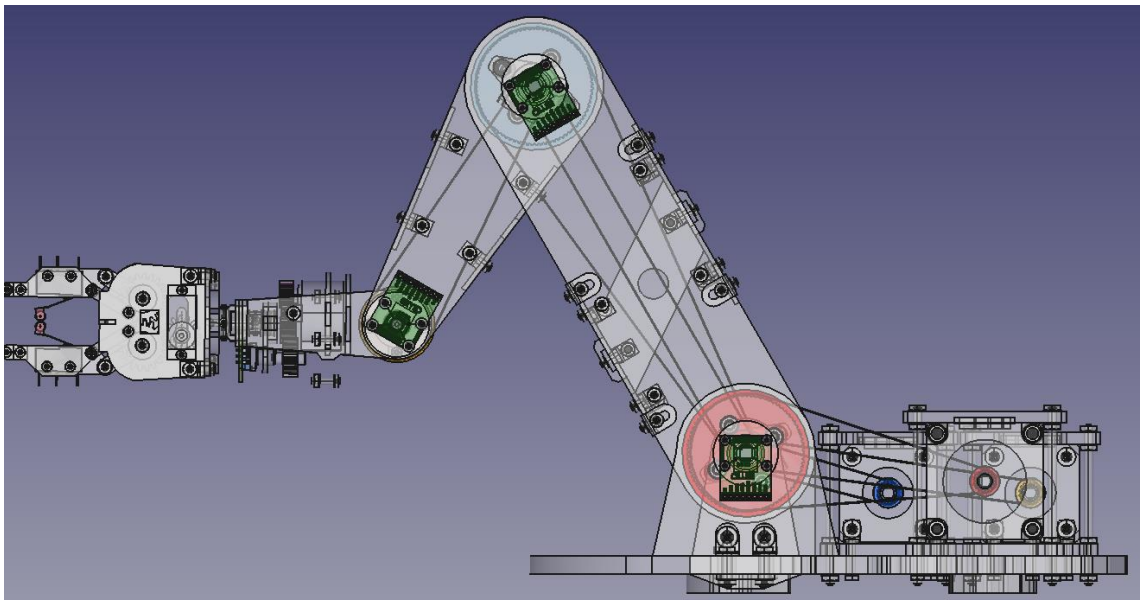
16. ábra - Prototípus robot mechanikai logikai rendszerterve

4.4.2 Mechanikai fizikai rendszerterv

A mechanikai kialakítás fizikai rendszerterve méretarányos, háromdimenziós CAD modell formájában készült el a FreeCAD nevű ingyenes CAD szoftver felhasználásával. Ez a szoftver segítségemre volt abban, hogy elvégezzem a hajtásrendszer terhelhetőségi számításait, mert a program lehetőséget biztosít az erők, tehetetlenségi nyomatékok, anyagjellemzők, sűrűségek, tömegek kezelésére. Az FC_info nevű python alapú addonja telepítése után lehetővé vált az alkatrészek súlypont koordinátáinak és a súlyponti tehetetlenségi tenzorainak a kinyerése.



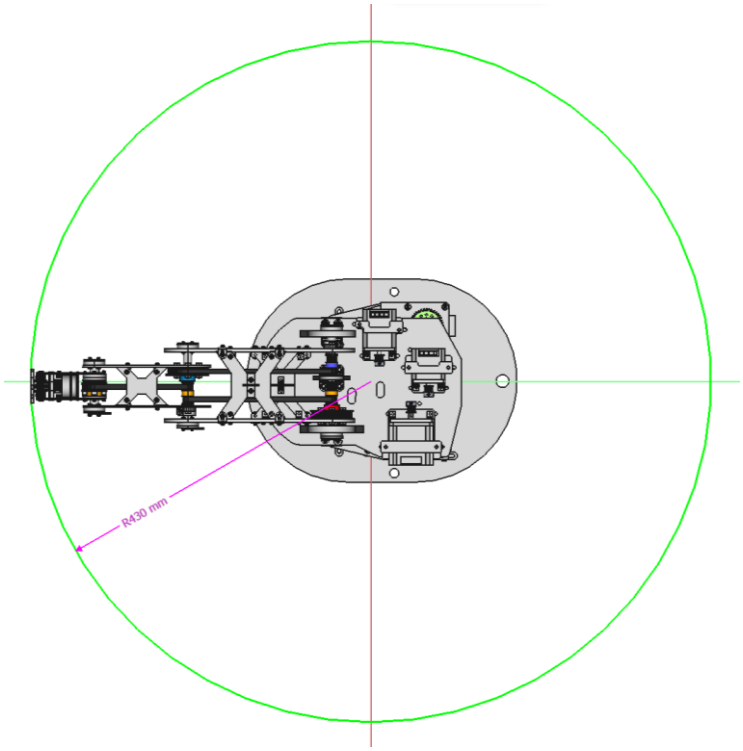
17. ábra - Prototípus robot CAD modellje



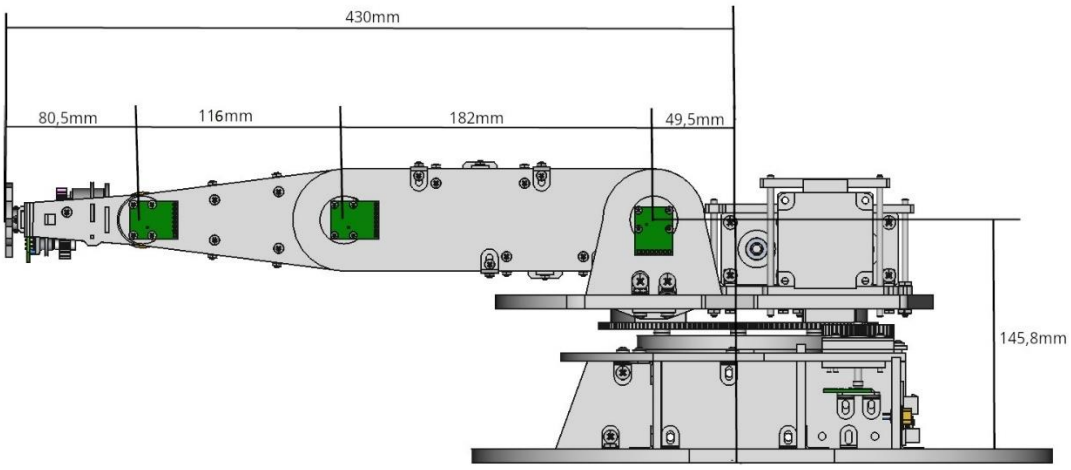
18. ábra - Prototípus robot CAD modellje

A törzs, váll, könyök és csukló-bólintó tengelyek meghajtására bipoláris léptetőmotorokat alkalmazunk, a törzs eseténben fogaskerekes hajtóművel, a váll, könyök és csukló-bólintás esetén bordásszíjtárcsás hajtóműveket, mindkét esetben nyomaték növelő – sebesség csökkentő áttételeket megvalósítva. A törzs motor az alaphoz van rögzítve, míg a váll, könyök és csukló-bólintó motorok egy a törzs hajtórendszer által forgatott felépítményen vannak rögzítve. A csukló-csavarás mechanizmusát a csukló szegmensbe épített unipoláris léptetőmotor hajtja meg

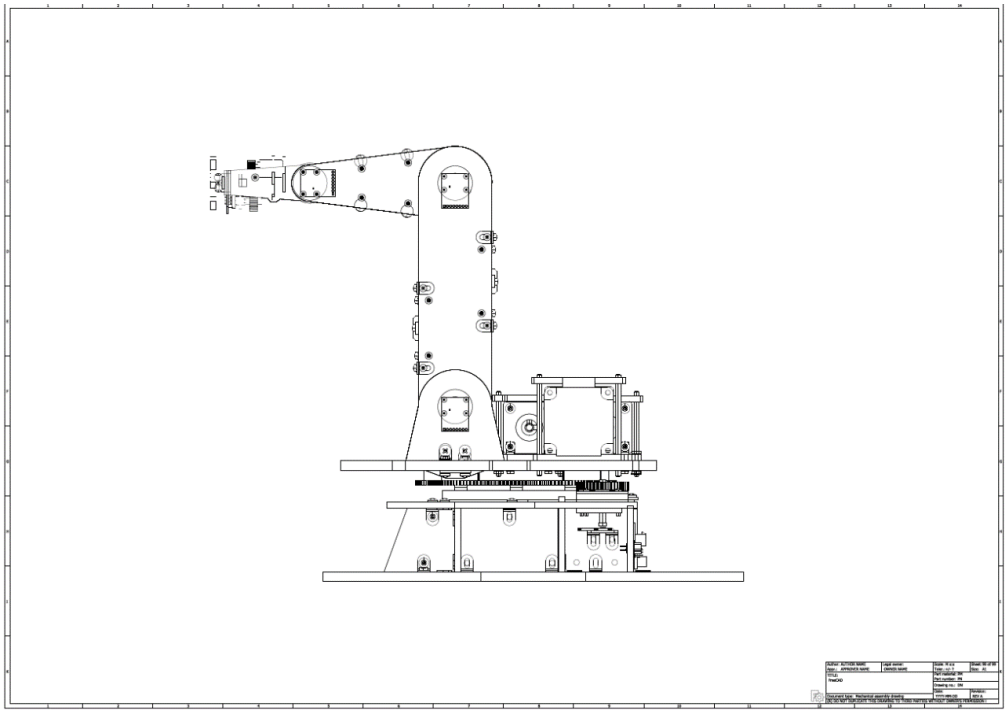
fogaskerekes áttétellel. A végberendezés megfogó mechanizmusát egy mikroszervo motor működteti szintén fogaskerekes áttétellel, úgynevezett erőzáró megfogást lehetővé téve.



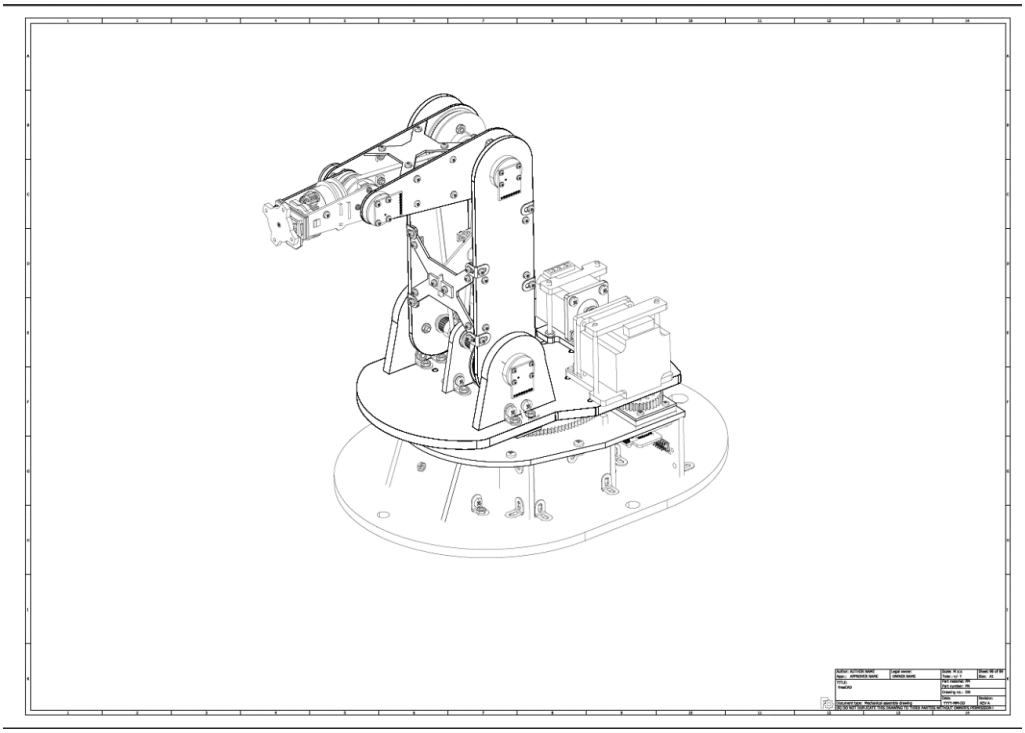
19. ábra - Effektor nélküli karhossz



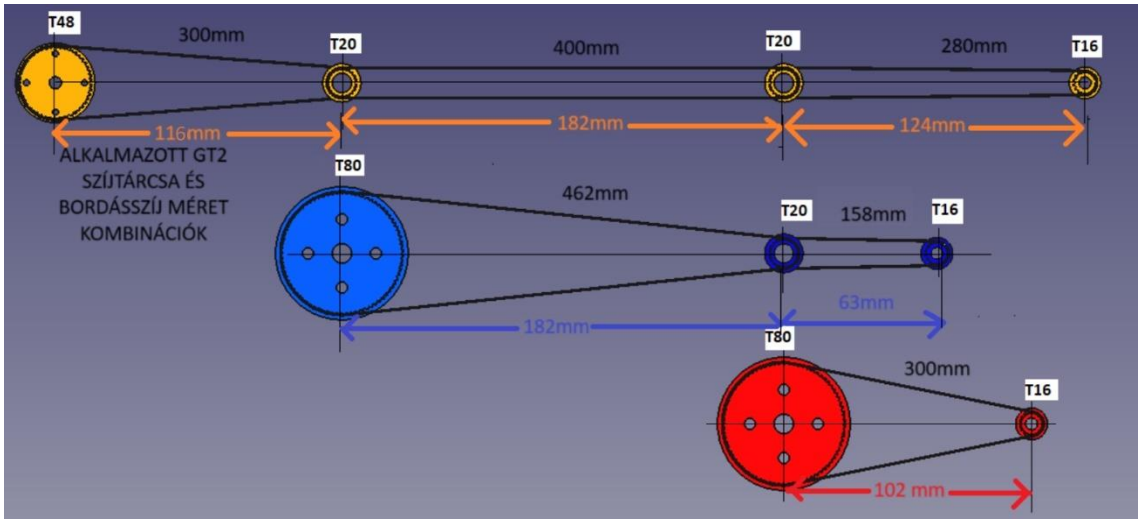
20. ábra - Effektor nélküli karméretek



21. ábra - CAD modell oldalnézete



22. ábra - CAD modell effektor nélkül



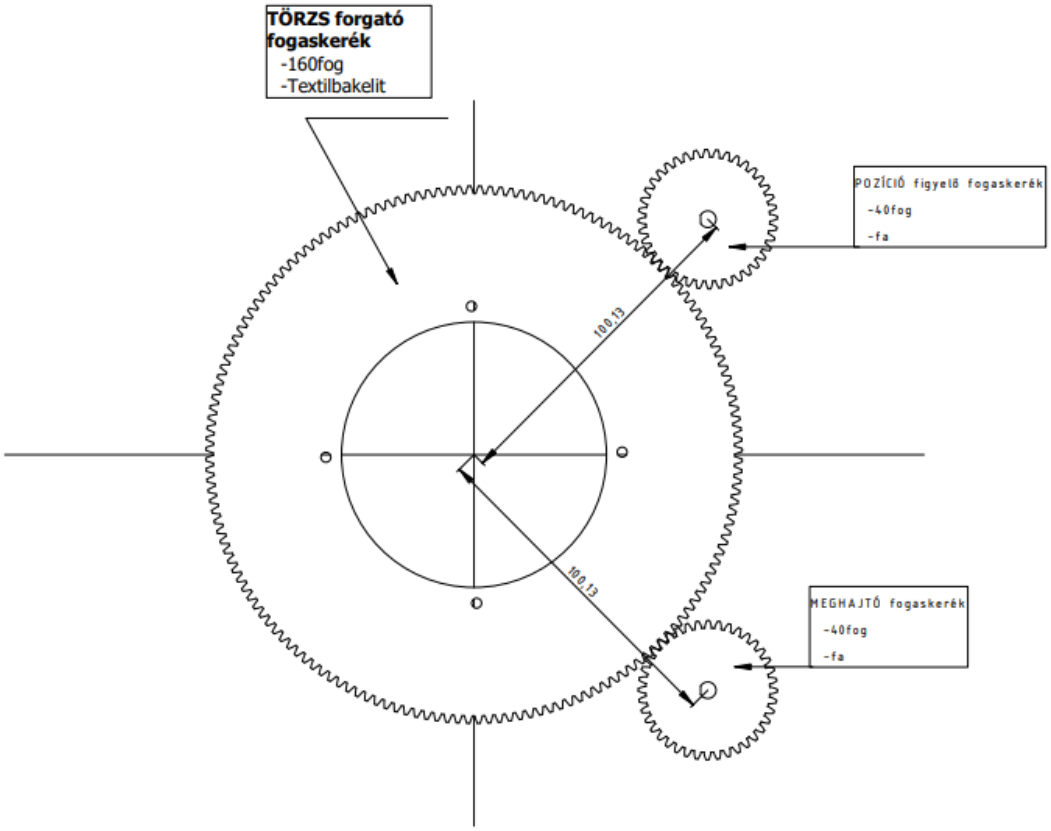
23. ábra - Bordásszíjtárcsás csuklóhajtóművek

A szíjtárcsás hajtóművek a felépítményen és a kar belsejében futnak.

A piros hajtómű a váll hajtóműve, amelynek a hajtó oldalán a motortengelyen egy 16 fogú szíjtárcsa, a hajtott oldalon pedig egy 80 fogú szíjtárcsa van. Ez 80/16 = 5-szörös nyomatéknövelő áttételt jelent, továbbá a vezérlési oldalon a valós szögnél ötször nagyobb elfordulás szükséges. [16]

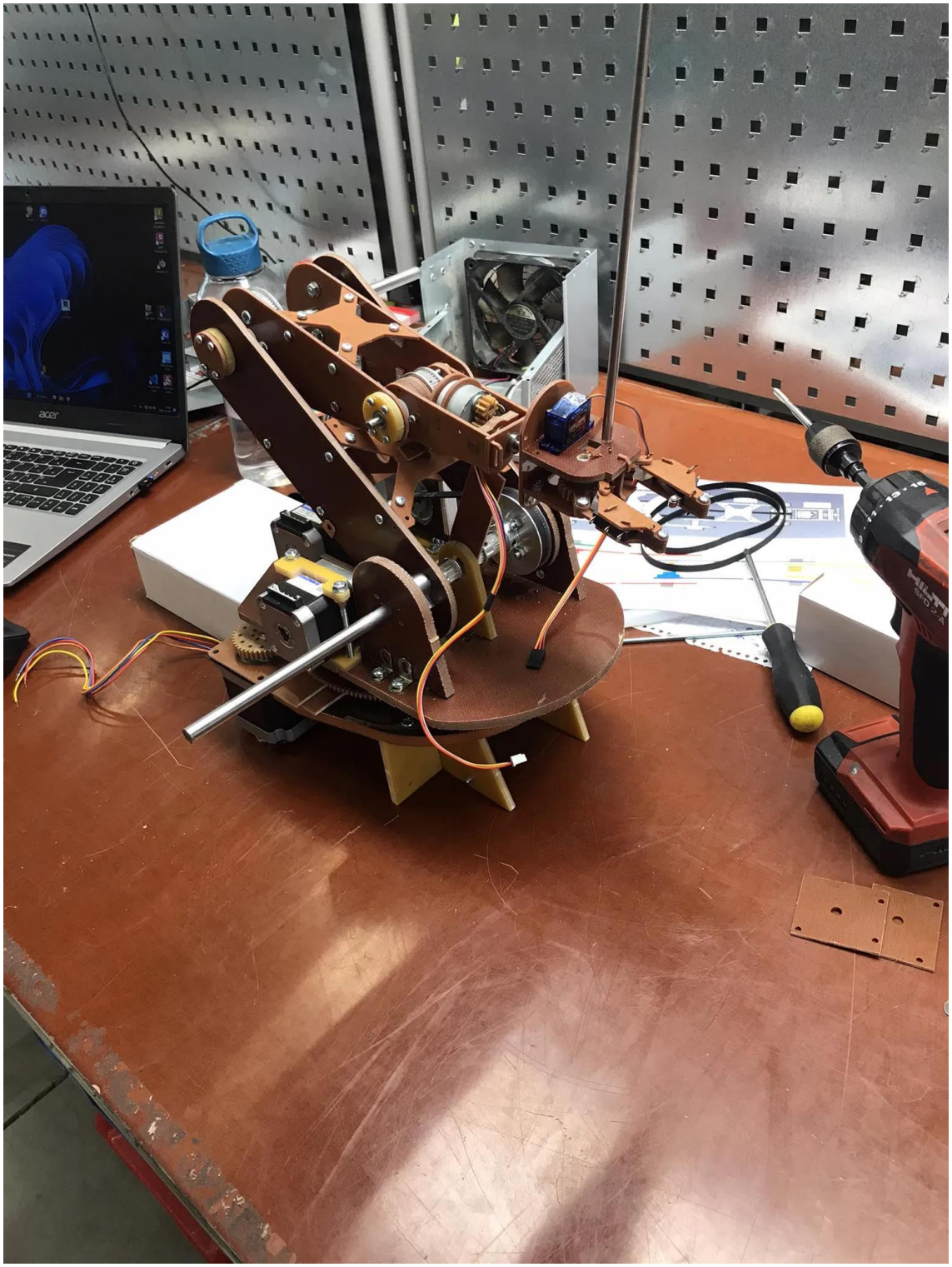
A kék hajtómű a könyöké. Hajtó oldalon 16 fogú tárcsa van, ami egy 20 fogú dupla tárcsán keresztül kapcsolódik a váll tengelyen át a könyökre rögzített 80 fogú hajtott tárcsara. Ez 80/20 * 20/16 = 5-szörös nyomatéknövelő áttételt jelent, továbbá a vezérlési oldalon a valós szögnél ötször nagyobb elfordulás szükséges. [16]

A sárga hajtómű a csukló-bólintást működteti. A hajtó tárcsa 16 fogú itt is, a váll és könyök tengelyeken átjátszó dupla, 20 fogú tárcsák vannak elhelyezve, és ezeken keresztül jut el a forgatónyomaték a hajtott 48 fogú tárcsára. Ez 48/20 * 20/20 * 20/16 = 3-szoros nyomatéknövelő áttétel, háromszoros szögelfordulással kell vezérelni. [16]

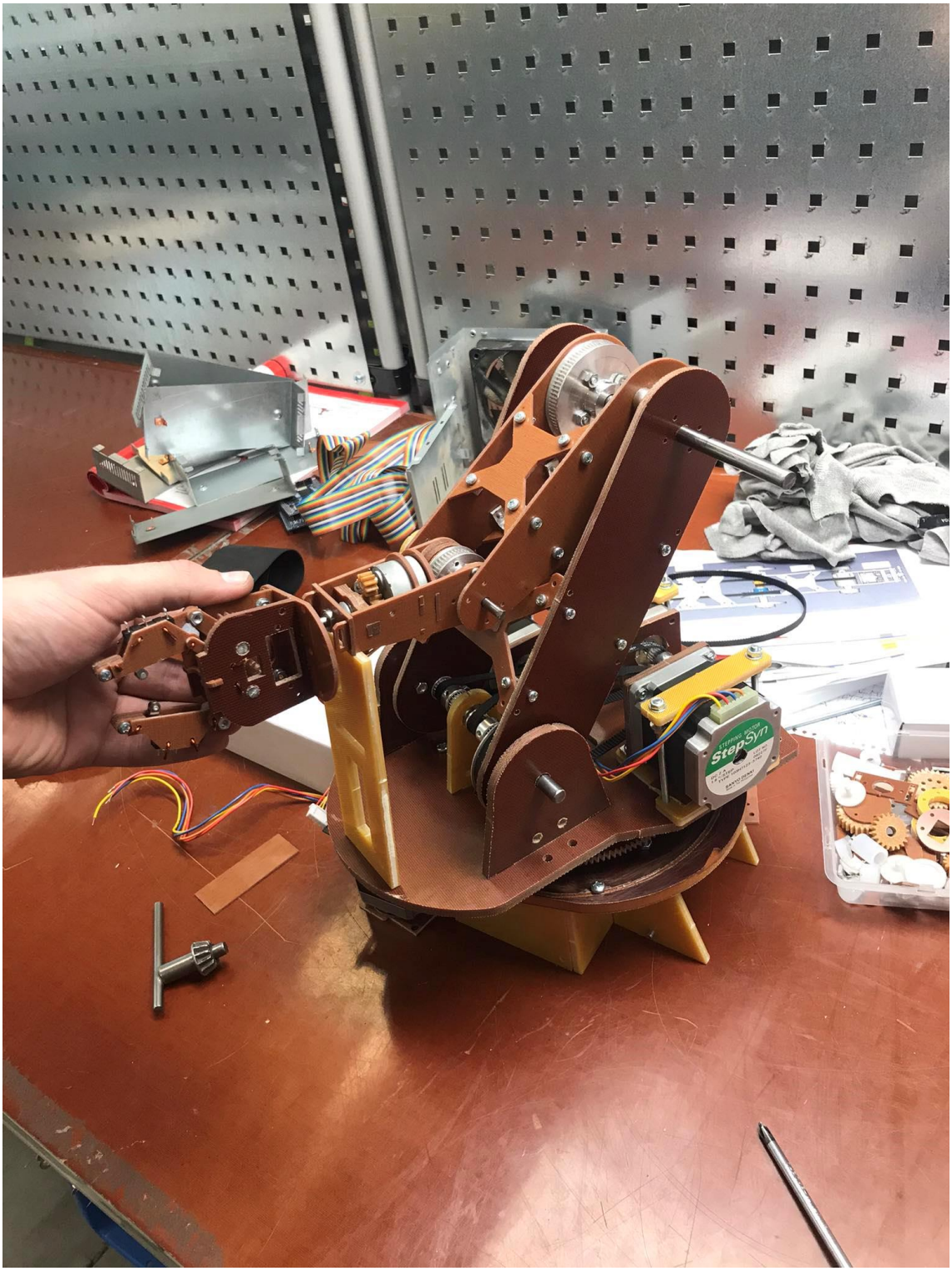


24. ábra - Törzs fogaskerekes hajtóműve

A törzs fogaskerekes hajtóművében a hajtó fogaskerék 40 fogú, a hajtott fogaskerék 160 fogú, ezzel 160/40 = 4-szeres nyomatéknövelő áttétel valósul meg. Ez azt is jelenti, hogy a q_1 törzs szög beállítása a hajtott oldalon, a hajtó oldalról $4*q_1$ szögelfordulást igényel. [16]



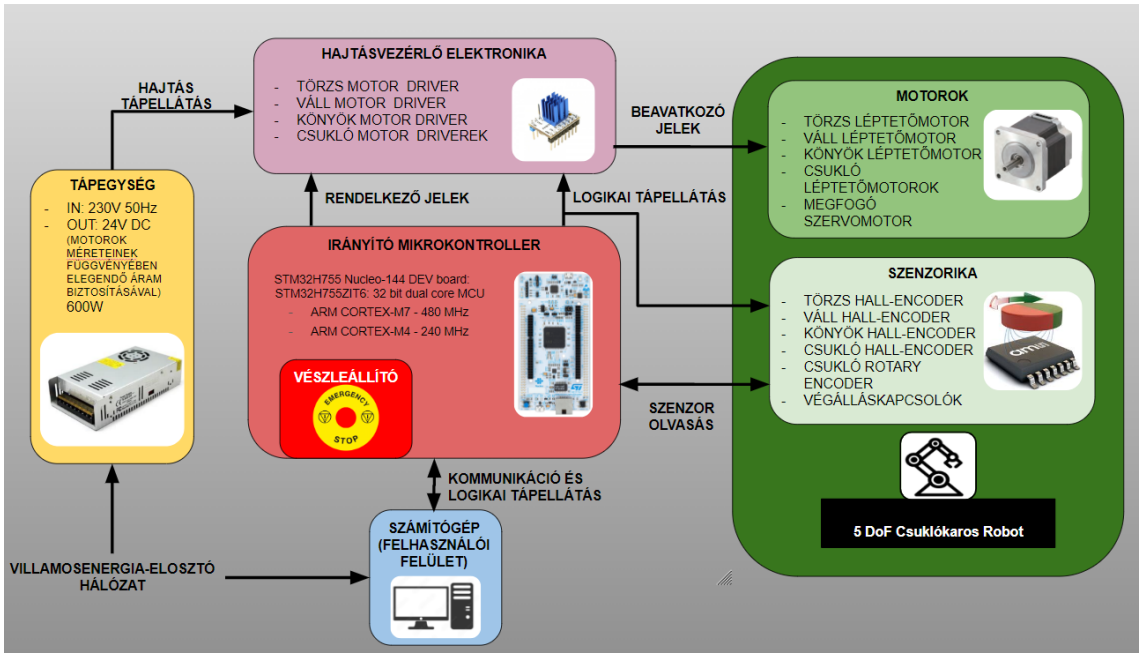
25. ábra - Fénykép az épülő robotról



26. ábra - Fénykép az épülő robotról

4.5 Elektronika

4.5.1 Elektronikai logikai rendszerterv



27. ábra - Elektronikai logikai rendszerterv

**A rajta szereplő termékfényképek:*

https://variometrum.hu/image/cache/catalog/vezerlok/Bipolar-stepper-motor-controller-2.8A-TMC2209-v3-800x800.jpg

https://sg.element14.com/productimages/standard/en_GB/4225702-40.jpg

https://www.ampul.eu/10331-medium_default/tapegyseg-24v-25a-600w.jpg

https://www.digikey.hu/-/media/Images/Product%20Highlights/A/ams/AS5048A%20and%20AS5048B%20Rotary%20Sensors/ams-as5048a-as5048b-rotary-sensors.jpg?&la=hu-HU&ts=eeb5f231-cd34-422b-9448-fd02007f2fdf

https://img.directindustry.com/images_di/photo-g/8104-9126103.webp

A hajtásvezérlő elektronikát TMC2209v4.0 bipoláris léptelőmotor-vezérlő és egy ULN2003 unipoláris léptetőmotor-vezérlő alkotják, amelyek a mikrokontroller által GPIO portokon keresztül szolgáltatott impulzussorozatoknak és logikai szinteknek

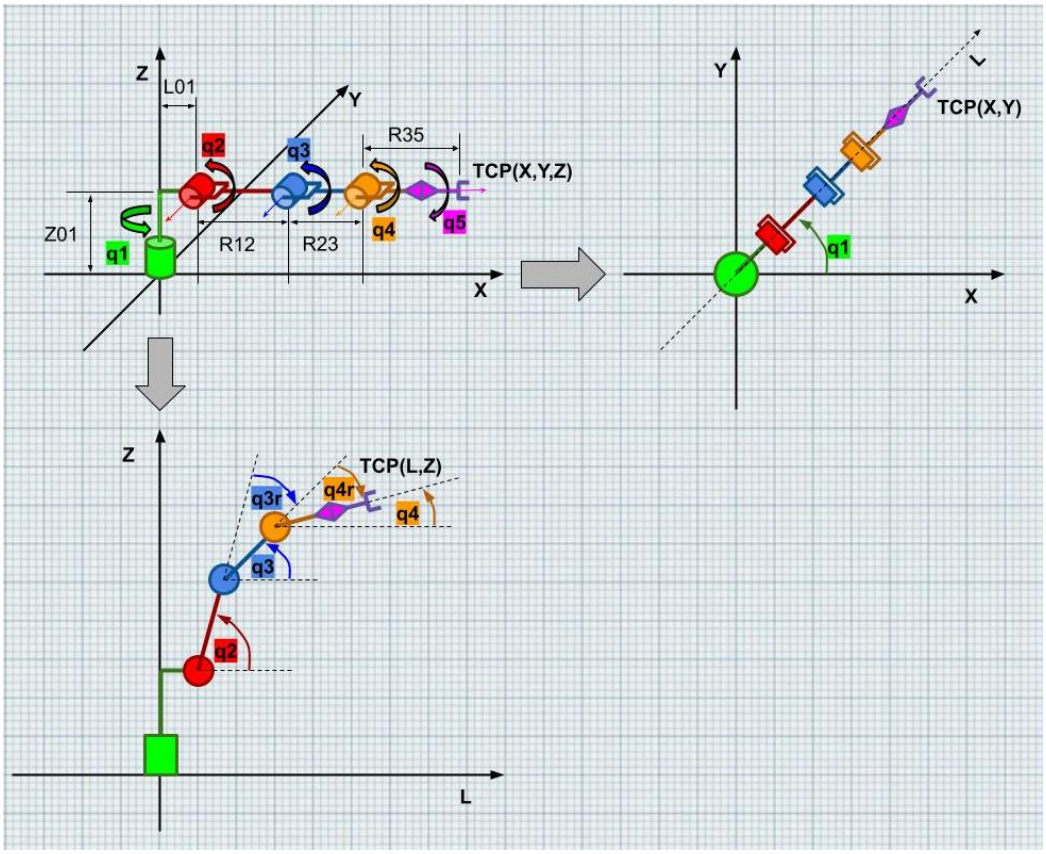
megfelelően előállítják a léptetőmotorok vezérléséhez szükséges szinuszoid fázisáram jelalakokat. Az ehhez szükséges teljesítményt a S-600-24, 24V, 25A, 600W tápegység biztosítja. A robotcsuklók tengelyeinek szögelfordulását a törzs, váll, könyök és csukló-bólintó tengelyek esetében AS5048B Hall szenzoros szögelfordulásmérő modulokkal detektáljuk, utóbbiak I^2C buszon keresztül, különböző I^2C címekkel kapcsolódnak az irányító mikrokontrollerhez. A q_5 csukló szögelfordulását egy rotary encoderrel mérjük. A megfogó szerszám zárását és nyitását a megfogó pofákba épített, gumiréteg alá rejtett két görgős végálláskapcsoló huzalozott logikai „ÉS” kapcsolata teszi érzékelhetővé. Az irányító mikrokontroller egy STM32H755ZIT6 kétmagos, ARM architektúrájú, 32 bites mikrokontoller, amely tartalmaz egy ARM Cortex-M7 processzormagot, maximum 480 MHz-es órajellel és egy ARM Cortex-M4 processzormagot, maximum 240 MHz-es órajellel, amelyek együtt szorosan csatolt multiprocesszoros rendszert alkotnak. A két mag szinkronizását hardveres szemafor teszi lehetővé. A mikrokontroller támogatja a hardveres osztást és FPU-t, azaz Floating Point Unit-ot, tehát lebegőpontos egységet tartalmaz. A mozgásirányító rendszer jövőbeli szoftveres megvalósításának kihívásai indokolttá teszik e mikrokontroller használatát, a párhuzamos feladatvégzés igénye, a lebegőpontos számábrázolás és hardveres osztás képességének igénye, a nagysebességű működés igénye és a nagy számítási teljesítmény igénye miatt.

4.5.2 Elektronikai fizikai rendszertervek

A prototípus robot manipulátor elektronikai fizikai rendszerterveit jelentő kapcsolási rajzok a szakdolgozat 1. mellékletében találhatóak.

5 Inverz kinematika megoldó algoritmus tervezése

5.1 Prototípus robotkar kinematikai modelljének és egyenleteinek kidolgozása



28. ábra - 5DoF robot kinematikai diagram és vetületei

A fenti ábra szemlélteti a bal felső sarokban a prototípus robot axonometrikus kinematikai diagramját az X-Y-Z világ-koordinátarendszerben. Jelölve vannak rajta a 4. fejezetben dokumentált szegmenshosszok, tengelytávolságok, valamint a csuklókoordináták.

A q_1 törzs szög visszafejtése egyszerű az X_{TCP} és Y_{TCP} világkoordináták alapján, ez az analitikus geometriával egyértelmű. Ezt az ábra jobb oldalán a kinematikai diagram felülnézeti vetülete mutatja be. A q1 szög egyenletének meghatározása a következő módon történik:

$$\tan q_1 = \frac{Y_{TCP}}{X_{TCP}} \qquad (5.1)$$

Az Y_{TCP} és X_{TCP} világkoordináták hányadosáról belátható, hogy a q_1 tangensével azonos. Az egyenlet mindkét oldalának arcustangensét képezve megkapjuk q_1 szöget: [8]

$$q_1 = \tan^{-1}\left(\frac{Y_{TCP}}{X_{TCP}}\right) \qquad (5.2)$$

Ha X_{TCP} nulla, akkor az arcustangens argumentuma zéróosztó miatt nem létezik, tehát ebben az esetben Y_{TCP} előjelét vizsgálva kell eldönteni, hogy q_1 értéke 90° vagy – 90°.

A q5 csukló-csavaró szög lokális roll szög, nagysága nem befolyásolja a csukló szegmens irányát, valamint a csukló-bólintó robotcsukló és a TCP pozícióját, hanem egyedül a végberendezésnek a csukló szegmens körüli orientációját határozza meg. Ennek a szögnek tehát közvetlenül a munkadarab megfogásával kapcsolatban van jelentősége. Ezt a szöget a világkoordináták egyedi alakú vektorában fogja a felhasználó definiálni.

$$s = \begin{bmatrix} X_{TCP} \\ Y_{TCP} \\ Z_{TCP} \\ \vdots \\ \boldsymbol{q_5} \end{bmatrix} \qquad (5.3)$$

A q_2, q_3 és q_4 szögek azonos, függőleges síkban forgó robotcsuklókhoz tartoznak. Ez a függőleges sík a Z tengely körül q_1 szöggel van elforgatva. A függőleges sík L-Z koordináta-rendszere szerinti vetülete a kinematikai diagramnak az ábra bal alsó részén látható. A karnak ez a középső része az L-Z síkbeli koordináta-rendszerben redundáns kinematikai lánc.

A q_2, q_3 és q_4 szögek beállítandó értéke kérdés marad és a visszafejtésük analitikusan nem egyértelmű. Az analitikus geometriai módszerrel kétértelmű megoldás kapható a q_2 és q_3 szögekre, amennyiben a q_4 csukló-bólintó szög előre definiált, felhasználó által a világkoordináták egyedi alakú vektorában rögzített.

$$s = \begin{bmatrix} X_{TCP} \\ Y_{TCP} \\ Z_{TCP} \\ \boldsymbol{q_4} \\ q_5 \end{bmatrix} \qquad (5.4)$$

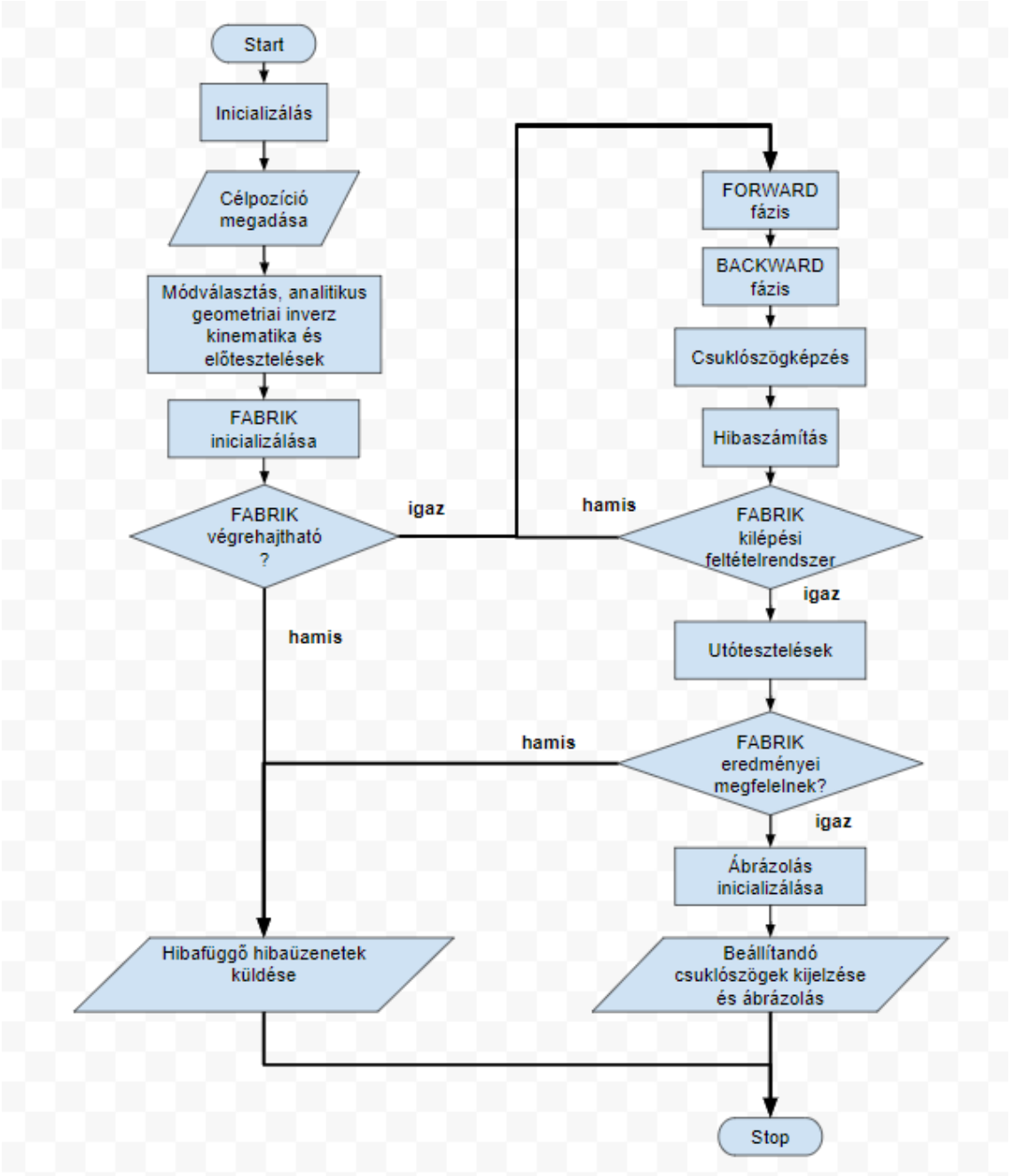
Ha nem akarjuk, hogy a q4 rögzítésétől függővé váljon a megoldó algoritmus, akkor az analitikus geometriánál erre az esetre általánosabb érvényűleg használható módszert kell alkalmazni, tehát például valamelyik iteratív közelítő algoritmust, amennyiben persze geometriai szinten és nem sebesség szinten szeretnénk megkapni az inverz kinematika megoldását. A prototípus csuklókaros robotunknál a léptetőmotoros hajtásrendszer alkalmazása miatt célszerű, ha a megoldás szögek és nem szögsebességek formájában áll rendelkezésre, mert a pontos pozícionáláshoz szükséges csuklószög beállításokhoz meghatározott lépésszámok szükségesek. A lépésszámokat a léptetőmotor-vezérlő áramkörök impulzussorozatok formájában kapják meg, amelyek alapján előállítják a léptetőmotorok vezérléséhez szükséges gerjesztési mintát. Az impulzussorozatok frekvenciája, a léptetési frekvencia felel meg a csuklósebességeknek, amelyek korlátozása, valamint a csuklógyorsulások korlátozása a robot hajtásrendszerének előzetes terhelhetőségi számításain alapulnak.

A FABRIK algoritmus kutatásaim és tapasztalataim alapján stabilabb és általánosabban használható, mint a Newton-Raphson algoritmus. A FABRIK érzéketlen azokra az eseményekre, amelyekre a Newton-Raphson kritikusan érzékeny, emellett a heurisztikus IK solverek között a leggyorsabb algoritmus. Az analitikus geometrián alapuló algoritmus teljesen pontos (a számítógépes környezetben a számábrázolás szóhosszúságához mérten maximálisan pontos) megoldáspárt eredményez az RRR alapkonfigurációan, viszont a prototípus robot q_4 szögét előre definiálni kell, míg a FABRIK könnyedén képes kezelni redundáns kinematikai láncokat is. A Jacobi-mátrix alakját is kellemetlenül befolyásolja a redundancia. Az L-Z síkon forgó három robotcsukló (q_2, q_3 és q_4 szögek) alkotta lánc, mint említettem redundáns láncnak számít, mert a c(L_{TCP}, Z_{TCP}) L-Z síkon értelmezett világkoordináta vektor (célvektor) csak két dimenziós, míg a kinematikai lánc ezen része három szabadsági fokú, másként kevesebb egyenlet áll rendelkezésünkre az egyenletrendszerben, mint amennyi

ismeretlen. A végleges inverz kinematika megoldó algoritmus alapjául tehát a FABRIK módszert fogom alkalmazni, mert így meghatározható véges pontossággal, stabilan és gyorsan a q_2 és q_3, valamint igény szerint a q_4 meghatározása is automatizálható az algoritmusban.

Három megoldó módszert megvalósító algoritmust szimuláltam Matlabban, a Newton-Raphsont, az analitikus geometriai inverz kinematika megoldót, valamint a FABRIK algoritmust, utóbbi esetben kiegészítve a szükséges ellenőrzésekkel, feltételrendszerekkel, hogy mind a munkatérben, mind a csuklótérben legális eredmény születhessék. A következő fejezetben ezek kerülnek majd bemutatásra, kiemelt figyelmet szentelve a FABRIK alapú, végleges megoldásnak.

5.2 Algoritmus működésének funkcionális terve



29. ábra - FABRIK alapú inverz kinematika megoldó algoritmus funkcionális terve

6 Tervezett inverz kinematika megoldó algoritmus megvalósítása Matlab környezetben

6.1 Bevezetés

Jelen fejezetben bemutatom, hogyan valósítottam meg szoftveresen az öt szabadsági fokú, prototípus csuklókaros robot manipulátorra alkalmazandó inverz kinematika megoldó algoritmust. Az inverz kinematika megoldó algoritmus azért felel, hogy az egyes, világkoordinátákban megadott, munkatérbeli, legális pályapontokat az effektor meghatározott vagy automatikusan kiszámított orientációban, továbbá meghatározott munkatérbeli és csuklótérbeli geometriai kényszerek alkalmazása mellett, kívánt pontossággal legyen képes elérni. A MATLAB R2023 – academic use – verziójában egy háromdimenziós, méretarányos pálcika-modellel szimulálva fejlesztettem a fent leírt funkciókat megvalósító szoftvert. Az algoritmus magába foglalja a nyers, ellenőrizetlen inverz kinematika megoldó algoritmust, valamint egy sok kényszerből álló ellenőrző rendszert, amely kizárja a nem megengedett világ- és csuklókoordinátákat, valamint e feltételek sértése esetén meghatározott hibaüzenetekkel válaszol a felhasználónak a Matlab konzoljában. Mielőtt azonban az inverz kinematika megoldó algoritmus szoftveres megvalósítása bemutatásra kerülne, azelőtt bemutatok két inverz kinematika megoldót, amelyeket kísérleti céllal fejlesztettem a korábbi fejezetekben tárgyalt módszerek tesztelésére és alkalmazására, lehetőségeinek és korlátainak felmérésére.

A szakdolgozatom címében szereplő algoritmus megvalósítását mindenképpen szimulációval terveztem, mert ez szélesebb lehetőséget ad biztonságos körülmények között kísérletezni, és nem igényli a robot hardver teljesen megépült voltát, valamint így az algoritmus fejlesztése során a felmerülő hibák bizonyosan szoftverben keresendőek, mert a szimulált környezet nem produkál mechanikai és villamos természetű hibákat, így könnyebb kiszűrni a megírt szoftver működési rendellenességeit. Azért döntöttem a Matlab szimuláció mellett, mert egyrészt széles körű matematikai függvénykészlettel bír, ami egy inverz kinematika megoldó algoritmus fejlesztés esetén kényelmes és kompakt lehetőséget biztosít a bonyolultabb matematikai kifejezések leírására és nemskalár matematikai objektumok kezelésére, másrészt a Matlab olyan függvényábrázolási lehetőségeket szolgáltat, melyekkel nagyon egyszerű az algoritmus

működését mérhetővé, tesztelhetővé, debugolhatóvá tenni, mindez igen szemléletesen végezhető el.

6.2 Newton-Raphson inverz kinematika megoldó szimulációs tesztelése

6.2.1 Működési elv

A Matlab környezetben megírt Newton-Raphson algoritmus a korábban említett megfontolások értelmében a q_2 és q_3 szögek megtalálására fókuszál kizárólag, ennek következtében a Newton-Raphson módszert tárgyaló fejezetben ismertetett két szabadsági fokú síkbeli példának az abszolút csuklóorientációs változatán alapszik a kód. A Jacobi-mátrix is azonos, 2×2 méretű. Ebben a kódban tehát kizárólag a váll és a könyök szögek visszafejtését tartottam szem előtt, az 5. fejezetben tárgyaltak értelmében azzal az előzetes megfontolással, hogy q_4 és q_5 a felhasználó által megadott, q_1 pedig analitikus módon arcustangenssel számítható, q_2 és q_3 pedig a kérdéses szögek. A kód elején inicializálás, a végén diagramábrázolás található. A kód magja egy for-end ciklus, amely az iteratív műveleteket, a közelítő becslések vektorát (a kódban x-szel jelöltem, nem q-val) a függvényrendszer vektorát és a Jacobi-mátrixot és az iteratív egyenletet kezeli. A Jacobi-mátrix szingularitás vizsgálata is itt történik azzal, hogy a Jacobi-determinánst egy if-else-end szerkezettel teszteljük, hogy zérus-e. Ha det(J) nem zérus, akkor J nem szinguláris és az inverze létezik, ezért a Matlab inv(J); függvényével le is képezzük az inverzet. Ha a det(J) zérus, akkor szingularitás áll fenn, ekkor az inverz nem létezik, helyette a Moore-Penrose-féle pszeudo-inverzét képezzük a Jacobi-mátrixnak a Matlab pinv(J); függvényével. A tolerancia teljesülését és az iterációs limit elérését is if-else-end szerkezetekkel vizsgálom. Ha a tolerancia teljesül akkor az utolsó iteráció indexét eltároljuk, majd törjük a for-end ciklust a break; utasítással. A becslésvektor értékeit és a függvényrendszer vektorának értékeit, valamint zérushely hibákat és az értékhibákat a lezajlott iterációk számának megfelelő oszlopszámú tömbökben tárolom, hogy ábrázolhassam a közelítés állapotait iterációról iterációra. A toleranciák: $\varepsilon = 0{,}001$ mm és $\delta = 0{,}001$ rad. Ez volt az első algoritmus, amivel kísérleteztem, és azért követeltem tőle ekkora pontosságot, hogy felmérjem, mire képes a módszer. A Newton-Raphson algoritmus Matlab kódja a szakdolgozat 2. mellékletében található.

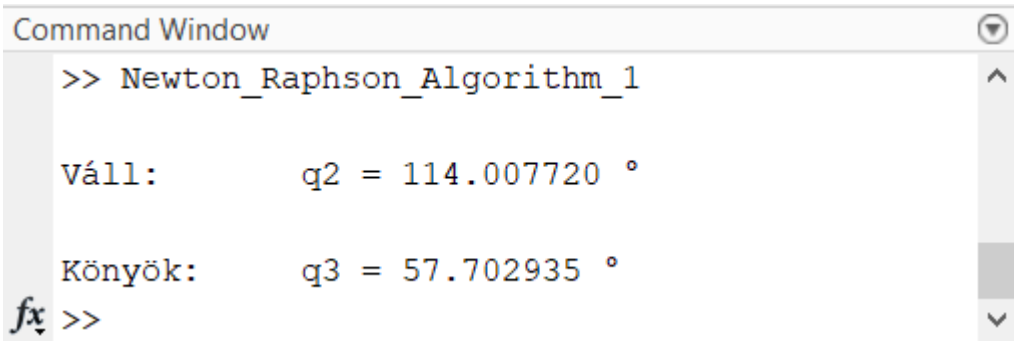
6.2.2 Teszt esetek

A kezdeti becslések minden vizsgált esetben: $q_2 = 90°$, $q_3 = 0°$.

Függőleges Z világkoordináta [mm]: 266

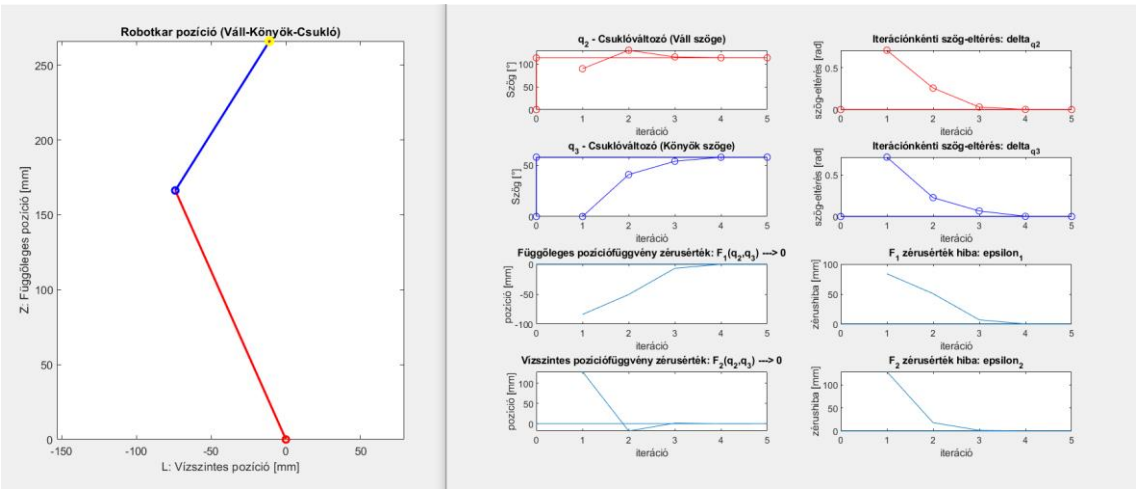
Vízszintes L világkoordináta [mm]: -11

Konzol üzenetek:



30. ábra - Newton-Raphson algoritmus konzol üzenet

Generált diagramok:

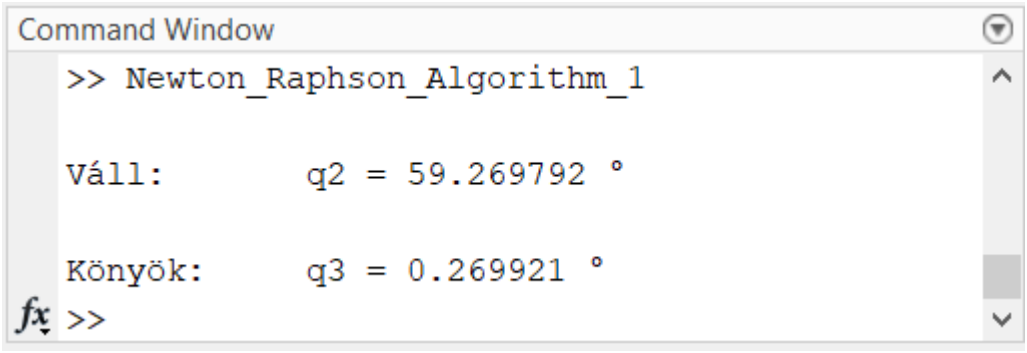


31. ábra- Newton-Raphson algoritmus diagram

Függőleges Z világkoordináta [mm]: 157

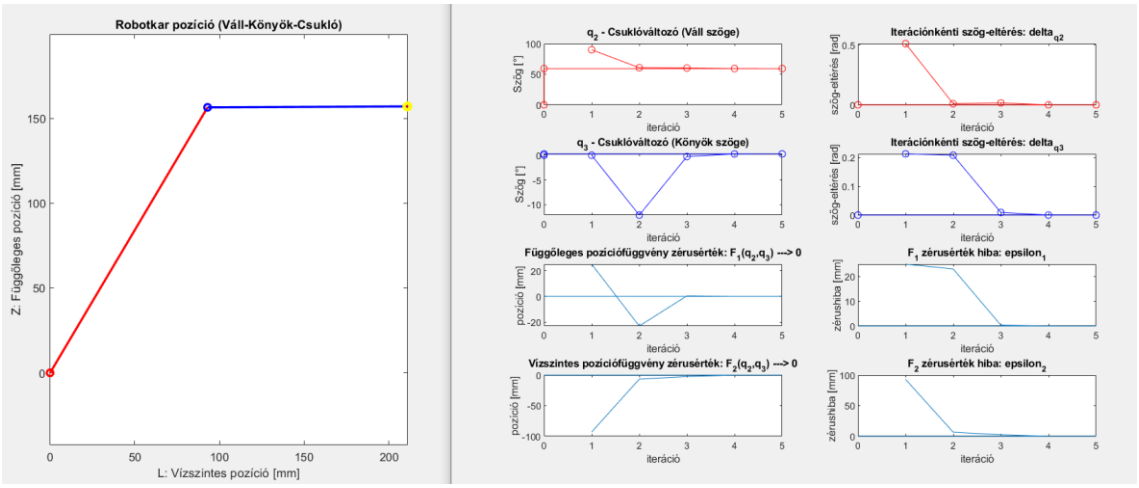
Vízszintes L világkoordináta [mm]: 211

Konzol üzenetek:



32. ábra- Newton-Raphson algoritmus konzol üzenet

Generált diagramok:

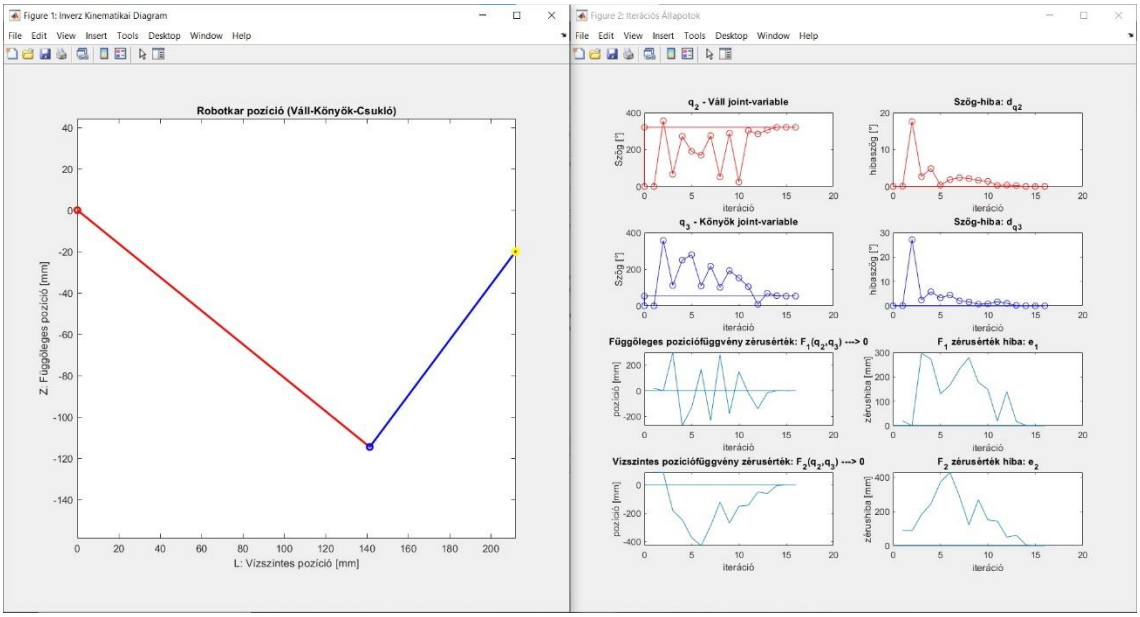


33. ábra- Newton-Raphson algoritmus diagram

Egy korábban elkapott oszcillációs jelenség, amely a 10. iterációig használhatatlan becsléseket termelt:

Függőleges Z világkoordináta [mm]: -20

Vízszintes L világkoordináta [mm]: 212



34. ábra- Newton-Raphson algoritmus oszcillációja

6.2.3 Eredmények kiértékelése

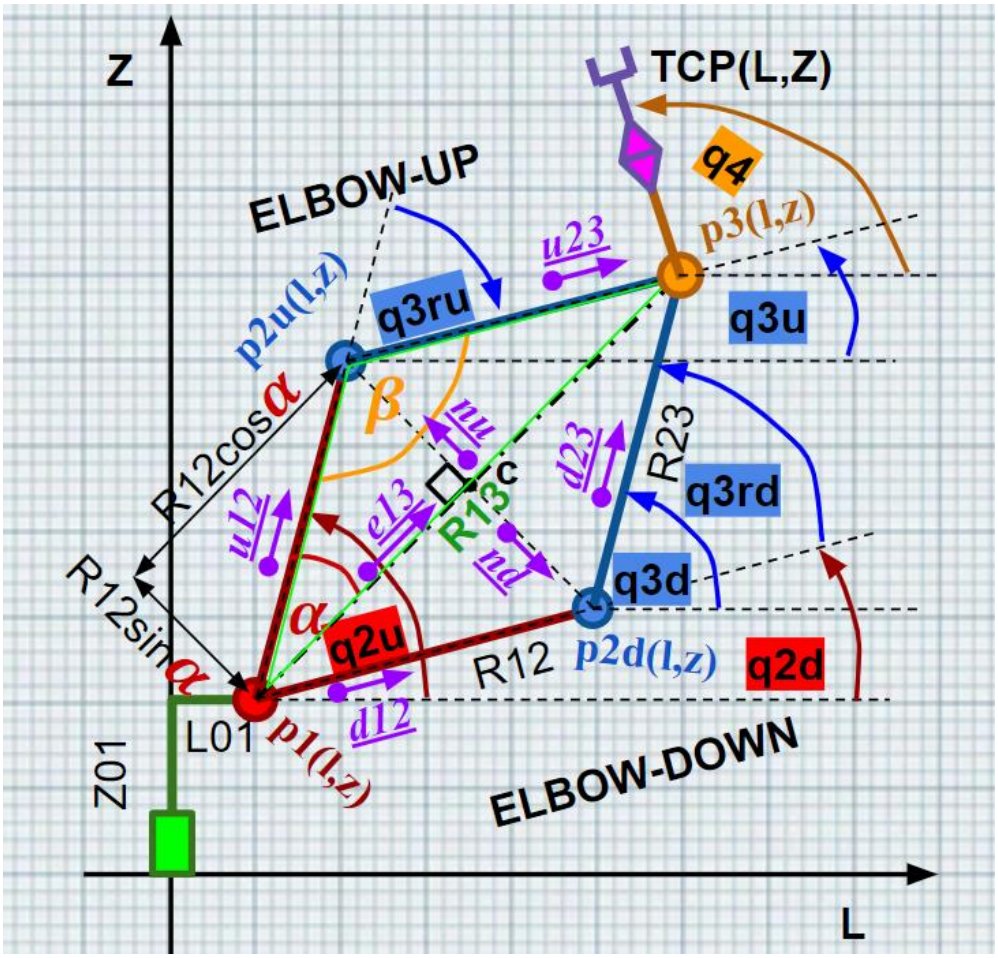
A dokumentált és nem dokumentált, vizsgált esetek a közelítés megbízhatóságát tekintve nagy szórást produkáltak. A Newton-Raphson inverz kinematika megoldó algoritmus pontos és gyors, de nem optimális és instabil eredményeket szolgáltat.

6.3 Analitikus geometriai inverz kinematika megoldó algoritmus

6.3.1 Működési elv

Az analitikus geometria alkalmazásán alapuló inverz kinematika megoldó algoritmust az L-Z síkon reprezentálva majdnem teljes mértékben a 3.2 fejezetbeli RRR példa alapján készítettem el. Kiegészítettem azzal, hogy kezelem benne a q_4 szöget is, ami megbonyolítja az inverz kinematikát, mert, mint az 5. fejezetben említettem, kinematikai redundanciát okoz az L-Z síkon. Ezért ebben az algoritmusban felhasználó adja meg a q_4 szög értékét, ezáltal a felhasználó határozza meg, hogy az L-Z síkban milyen orientációban érje el az effektor a célt. A célpozícióból tehát kell egy q_4-től és az R_{35} szegmenshossztól függő eltolás, hogy megkapjuk a csukló ízület pozícióját, ami a

3.2 fejezetben használt deltoid csúcsa lesz. A következő ábra szemlélteti L-Z síkon a megoldást:



35. ábra - 5DoF L-Z kinematikai diagram

A q_5 szöget nem kezelem ebben az algoritmusban, mivel az L-Z síkon nem ábrázolható. Az algoritmus elkészítésénel a kidolgozott megoldási módszer és az elképzelt működési elv megvalósítása volt a cél, ez az algoritmus még nem tartalmazza a végteszteléseket csuklótérre és munkatérre.

A program a geometriai konstansok és változók inicializálását követően, a 3.2 fejezetben leírt deltoidos módszerrel történik (közel azonos jelölésrendszert használva a kódban) az inverz kinematikai feladat analitikus megoldása. Tartalmaz az algoritmus néhány tesztet a teljesség igénye nélkül. A program végén az L-Z síkú diagramábrázolás kódrészlete található az effektor pozícionálás szemléltetése céljából. Az ábrázoláshoz a

Matlab figure(…); plot(…); hold on; hold off; és egyéb diagramábrázolási utasításait, függvényeit használtam. A Matlabban generált diagramokon ennél az algoritmusnál a TCP célpozícióját az L-Z síkon egy zöld üres körrel jelöltem.

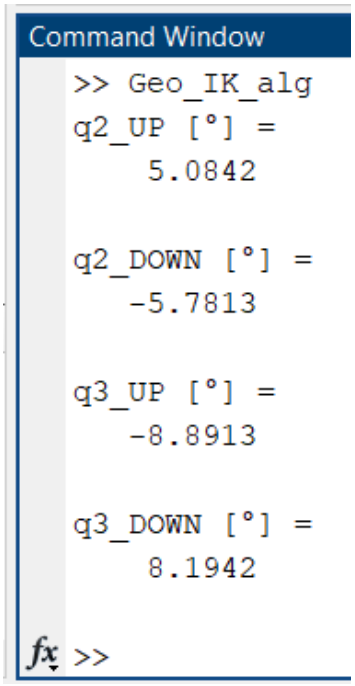
Az analitikus geometriai inverz kinematika megoldó algoritmus Matlab kódját a szakdolgozat 3. melléklete tartalmazza.

6.3.2 Teszt esetek

6.1. táblázat - Bemenet

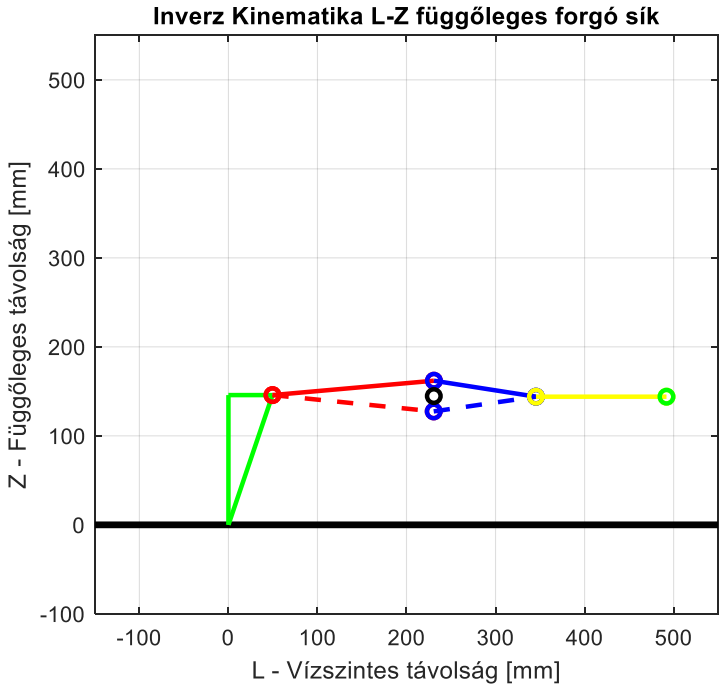
Bemeneti Világkoordináták			
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	q_{4_TCP} [°]
492	0	144	0

Kimeneti konzol üzenetek:



36. ábra - Konzol üzenet

Generált diagram:

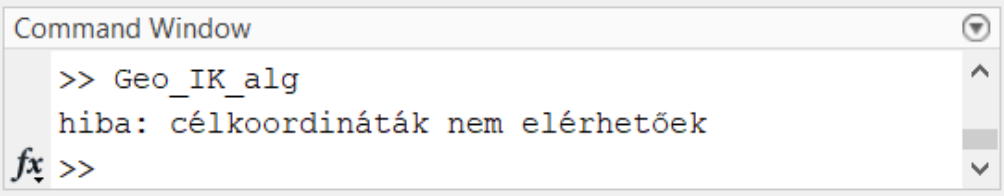


37. ábra - Kimeneti diagram

6.2. táblázat - Bemenet

Bemeneti Világkoordináták			
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	q_{4_TCP} [°]
499	0	144	0

Kimeneti konzol üzenetek:



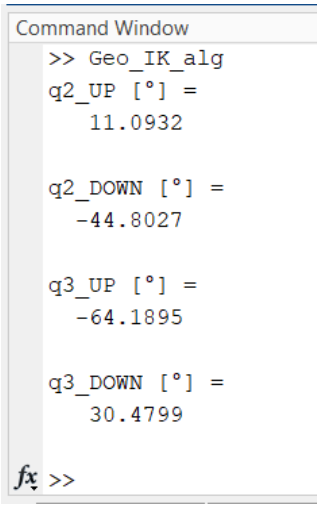
38. ábra - Konzol üzenet

Generált diagram: NINCS.

6.3. táblázat - Bemenet

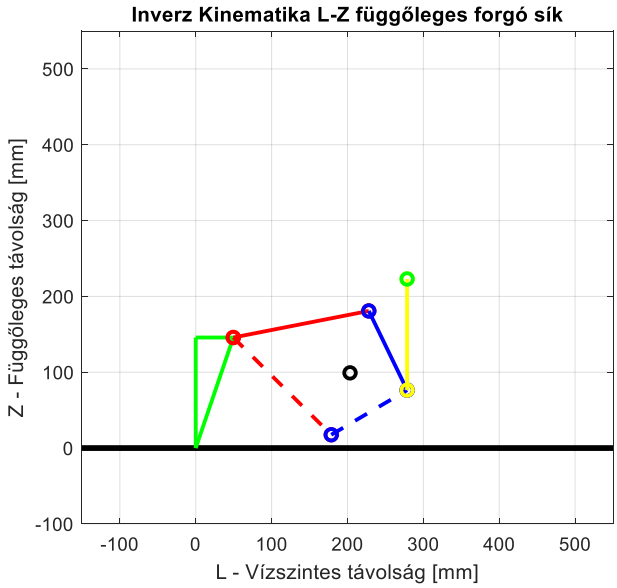
Bemeneti Világkoordináták			
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	q_{4_TCP} [°]
265	86	223	90

Kimeneti konzol üzenetek:



39. ábra - Konzol üzenet

Generált diagram:



40. ábra - Kimeneti diagram

6.4. táblázat - Bemenet

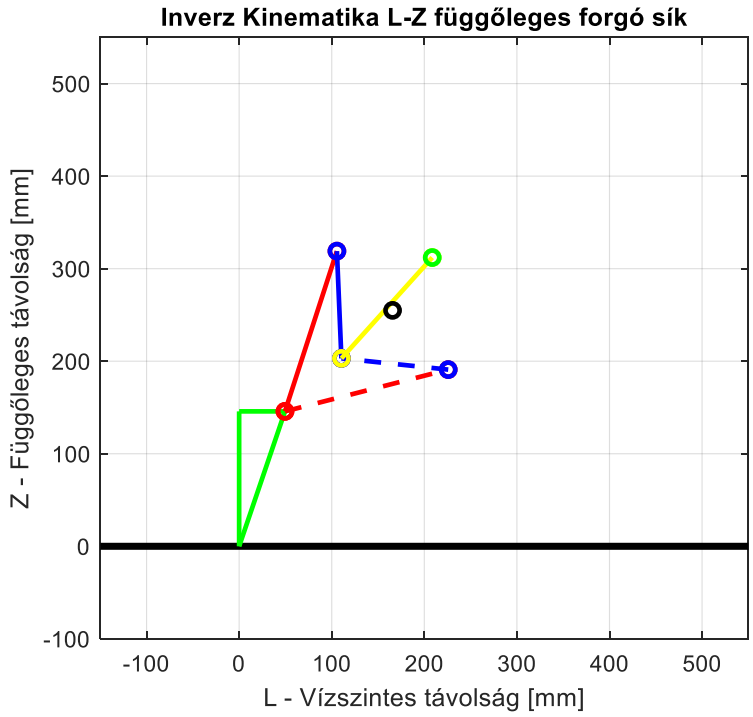
Bemeneti Világkoordináták			
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	q_{4_TCP} [°]
190	86	312	48

Kimeneti konzol üzenetek:

```
Command Window
>> Geo_IK_alg
q2_UP [°] =
   72.0537

q2_DOWN [°] =
   14.3526

q3_UP [°] =
  -87.5904

q3_DOWN [°] =
  173.9967

fx >>
```

41. ábra - Konzol üzenet

Generált diagram:



42. ábra - Kimeneti diagram

6.3.3 Eredmények kiértékelése

Ennek az algoritmusnak előnye, hogy a számítógépes környezetbeli számábrázolás szóhosszúságának pontosságával működik lévén, hogy analitikus megoldás. A szakdolgozatomban tárgyalt megoldások közül ez a legpontosabb. Mivel két megoldást ad az algoritmus, így az egyik könyök konfiguráció illegalitása esetén a másik még legális lehet, tehát ez ad egyfajta rugalmasságot, amelyet a megfelelő feltételvizsgálatokkal hasznosítani lehet. Nem iteratív módszer, így a műveletsor pályapontonként csak egyszer zajlik le, tehát arányaiban nagyon gyors. A számítási teljesítmény és az utasításkészlet szükséglete csökken, mivel a módszer nem alkalmaz mátrixműveleteket.

Hátránya a megoldásnak, hogy jelen megvalósításban a q_2-q_3-q_4 szakasz L-Z síkbeli redundanciája miatt, a q_4 szöget a felhasználó által rögzített effektor-orientációként kell használni, mert kiszámítása többértelmű a szakasz túlhatározottsága következtében. További hátrány, hogy a deltoid módszerben az arcuscosinus használatánál (a Matlab acos(…); függvénye) a bemeneti paramétert tesztelni kell, és a teszt miatt a munkatér határfelülete csak megközelíthető a TCP által, de nem elérhető, tehát nincs kinyújtott állapot.

6.4 FABRIK alapú ellenőrzött inverz kinematika megoldó algoritmus

6.4.1 Működési elv

A 3.3.3 fejezetben leírtakat és az ott bemutatott három szabadsági fokú síkbeli példát alkalmazom jelen algoritmusban a kinematikai lánc L-Z síkon forgó csuklóin. A jelenlegi megoldás funkcionálisan teljes. Mind az öt csuklókoordinátát kezelem, ezek közül a q_5-ön kívül mindet ábrázolom is a generált ábrán. A generált ábra három dimenziós Descartes-féle (X-Y-Z) térben (a munkatérben) ábrázolja a kinematikai lánc aktuálisan visszafejtett és beállított csuklókoordinátáinak megfelelő konfigurációkat, pálcika-modell formájában. Az algoritmus előzetes (FABRIK végrehajtását engedélyező) és utólagos (FABRIK eredményeit vizsgáló) feltételvizsgálatokat tartalmaz a geometriai kényszereknek, valamint a csukló- és munkatér határparamétereinek megfelelően. Ezek a vizsgálatok elszórt if-else-end szerkezetekkel

történnek, és adott hibaállapotot reprezentáló „OK flagek”, tehát engedélyező jelzések (változók) értékét állítják be. Az 5.2 fejezet folyamatábrája szerinti fő vizsgálatok, ezeket az engedélyező jeleket hozzák ömlesztetten logikai „ÉS” kapcsolatba, tehát mindegyik igaz volta feltétele az előzetes vizsgálat esetén a FABRIK, az utólagos vizsgálat esetén a pozícionálást szimuláló diagram generálás végrehajtásának. A Matlab kód szerkezete az 5.1 fejezetben kidolgozott elvet és az 5.2 fejezetben tervezett folyamatábrát követi. A program iteratív részét, tehát magát a FABRIK eljárást egy for-end ciklus tartalmazza, amelyet feltételesen megtörünk egy a tolerancia kielégülését vizsgáló if-else-end szerkezet igaz ágába tett break; utasítással. A toleranciavizsgálat az L-Z síkon értelmezett hibavektor normájának és a tolerancia vektor normájának összehasonlításán alapul. A tolerancia vektor a következő:

$$error_{max} = \begin{bmatrix} L_{error_{max}} \\ Z_{error_{max}} \end{bmatrix} = \begin{bmatrix} 0{,}01\ mm \\ 0{,}01\ mm \end{bmatrix} \qquad (6.1)$$

A hibavektor maximális normája az L-Z síkon tehát:

$$\|error_{max}\| = \sqrt{0{,}01^2 + 0{,}01^2} = \frac{\sqrt{2}}{100} mm \qquad (6.2)$$

A prototípus robotkar mechanikai kialakításának becsült pontatlanságai és a hajtáselemek potenciális pontatlanságai, mint például a szíjak megnyúlása okozta hiba, vagy a léptetőmotorok vibrációja nem teszik indokolttá, hogy ilyen pontos legyen a közelítés a valóságban, mert az ilyen szűk tolerancia a prototípus robotkar estén nem teljesíthető, bizonytalan. A Matlab szimulációban azért döntöttem e szűk tolerancia alkalmazása mellett, hogy felmérhető legyen, mire képes a FABRIK módszer, mennyi iterációval teljesíti ezt a szűk toleranciát, mert ha erre képes olyan gyorsan, mint tapasztaltam és a fejezet hátralévő részében bemutatom, akkor egy megengedőbb toleranciával még előnyösebb lesz a közelítési sebesség, persze nem drasztikusan, mint az bemutatásra is került a 3.3.3 fejezetben, de feltétlenül javulni fog.

Az algoritmusban választhatóvá tettem, hogy a q_4 szög visszafejtése automatikus legyen-e, vagy figyelembe vegye a végrehajtás a felhasználó által definiált q_4 értréket.

Az s egyedi alakú, hatdimenziós világkoordináta vektorban a negyedik koordináta egy módválasztó változó, amely, ha „1” értékű, akkor q_4 beállítása automatikus, ha „0”, akkor a program figyelembe veszi az ötödik koordinátát, amely maga a q_4 felhasználói effektor orientáció Az algoritmusbeli közelítő ciklusban if-else-end szerkezetekkel különítem el a „forward” és „backward” fázisok q4-et is visszafejtő, valamint a q4-et nem visszafejtő változatait az s vektor negyedik koordinátájának értékétől függően.

Az algoritmusban alkalmazott bemeneti világkoordináták és kimeneti csuklókoordináták vektorai az alábbiak:

$$s = \begin{bmatrix} X_{TCP} \\ Y_{TCP} \\ Z_{TCP} \\ auto_{q4} \\ q_{4\_TCP} \\ q_{5\_TCP} \end{bmatrix} \qquad (6.3)$$

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} \qquad (6.4)$$

A FABRIK kezdeti csuklópozícióit a következő vektorok írják le:

Törzs: $[0\ 0]^T$

Váll:

$$p_{1init} = \begin{bmatrix} L_{01} \\ Z_{01} \end{bmatrix} \qquad (6.5)$$

Könyök:

$$p_{2init} = p_{1init} + \begin{bmatrix} R_{12}\cos 60° \\ R_{12}\sin 60° \end{bmatrix} \qquad (6.6)$$

Csukló:

$$p_{3init} = p_{2init} + \begin{bmatrix} R_{23} \cos 45° \\ R_{23} \sin 45° \end{bmatrix} \quad (6.7)$$

TCP:

$$p_{4init} = p_{3init} + \begin{bmatrix} R_{35} \cos 30° \\ R_{35} \sin 30° \end{bmatrix} \quad (6.8)$$

A FABRIK alapú inverz kinematika megoldó algoritmus Matlab kódja a szakdolgozat 4. mellékletében található.

6.4.2 Teszt esetek

6.5. táblázat - Bemenet

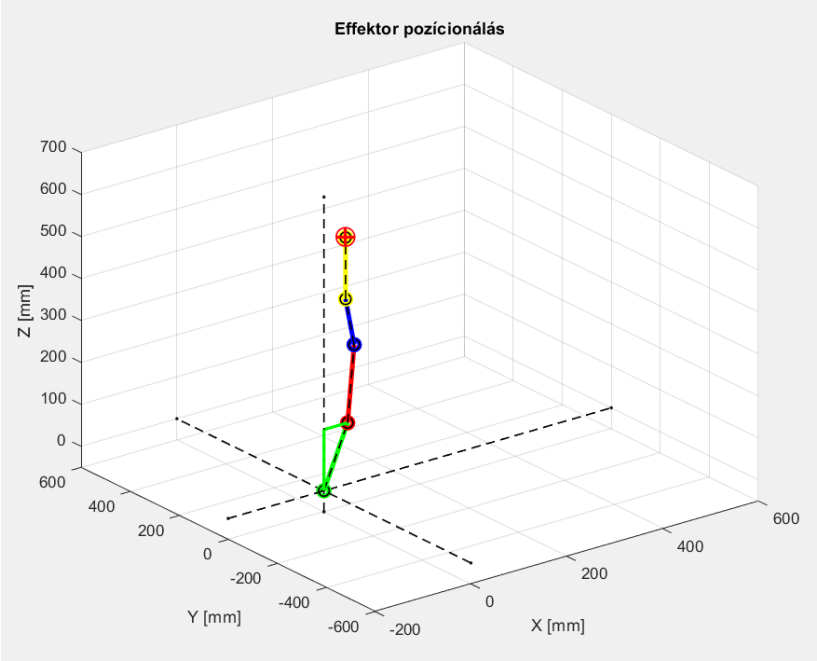
A bemeneti s világkoordináta vektor tartalma					
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	$auto_{q4}$	q_4 [°]	q_5 [°]
45	0	590	0	90	90

Kimeneti konzol üzenetek:

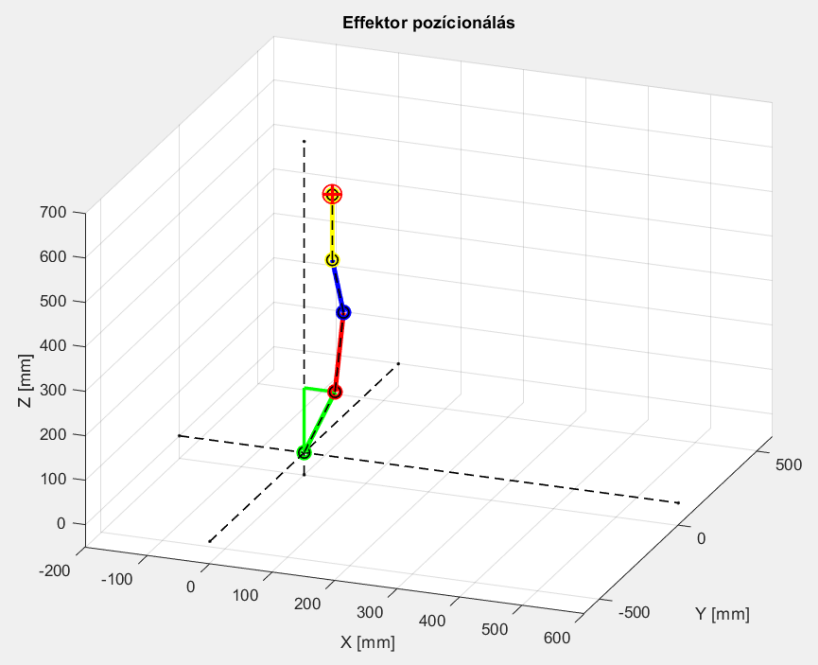
```
Command Window
>> FABRIK_alg
A q4 szöget felhasználó definiálja.
Törzs (q1) szög[°]:                0.0000
Váll (q2) szög[°]:                 85.7045
Könyök (q3) szög[°]:               98.8764
Csukló bólintás (q4) szög[°]:      90.0000
Csukló csavarás (q5) szög[°]:      90.0000
TCP pozíció X hibája [mm]:         0.2328
TCP pozíció Y hibája [mm]:         0.0000
TCP pozíció Z hibája [mm]:         -1.4905

fx >>
```

43. ábra - Konzol üzenet

Generált diagram:



44. ábra - Kimeneti diagram

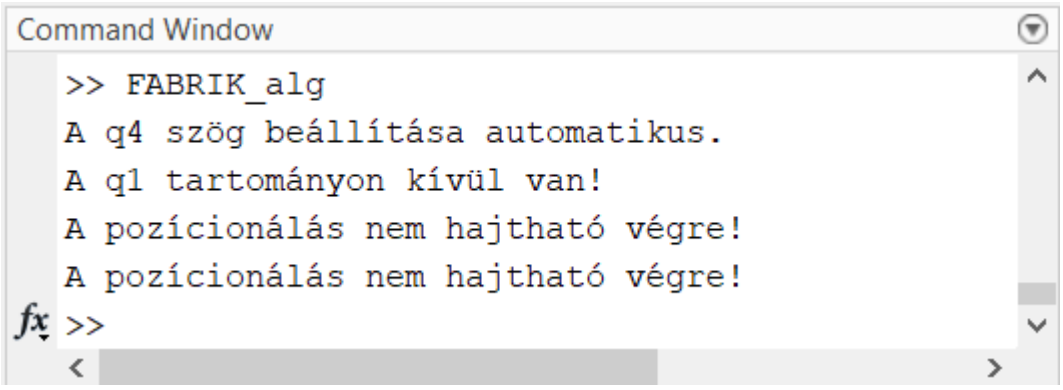


45. ábra - Kimeneti diagram

6.6. táblázat - Bemenet

A bemeneti s világkoordináta vektor tartalma					
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	$auto_{q4}$	q_4 [°]	q_5 [°]
-155	60	360	1	90	90

Kimeneti konzol üzenetek:



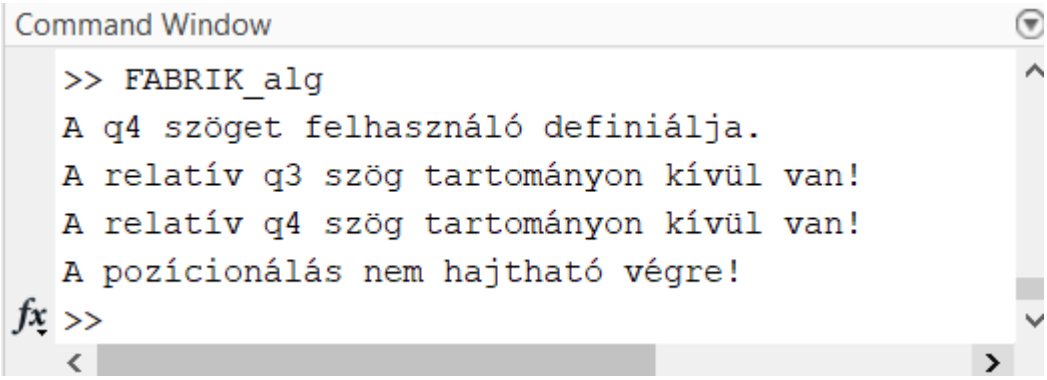
46. ábra - Konzol üzenet

Generált diagram: NINCS.

6.7. táblázat - Bemenet

A bemeneti s világkoordináta vektor tartalma					
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	$auto_{q4}$	q_4 [°]	q_5 [°]
155	80	350	0	90	90

Kimeneti konzol üzenetek:



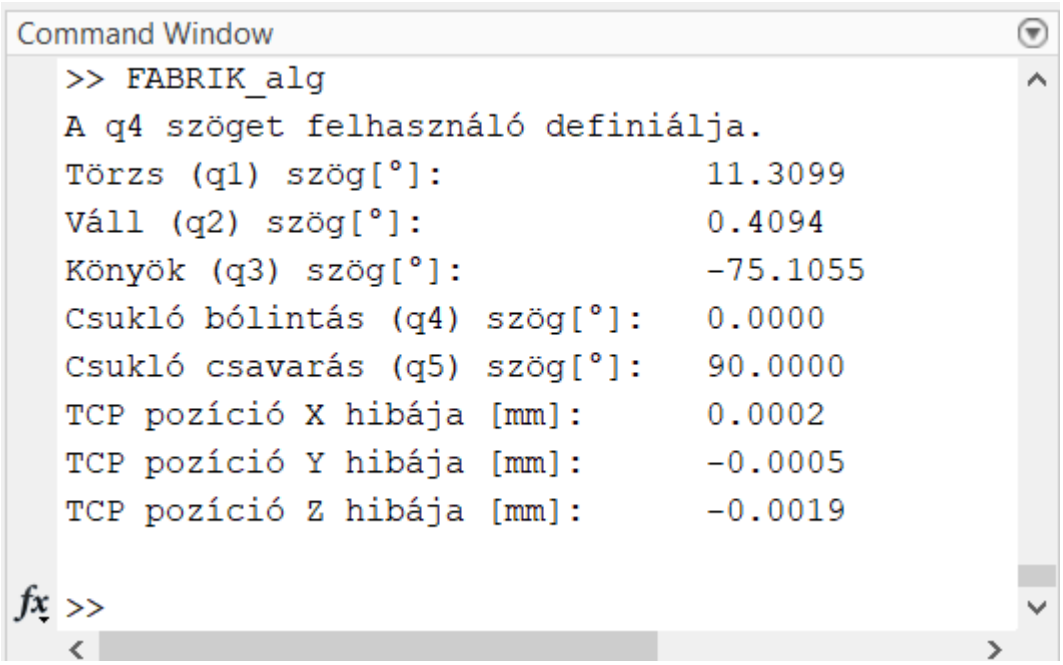
47. ábra - Konzol üzenet

Generált diagram: NINCS.

6.8. táblázat - Bemenet

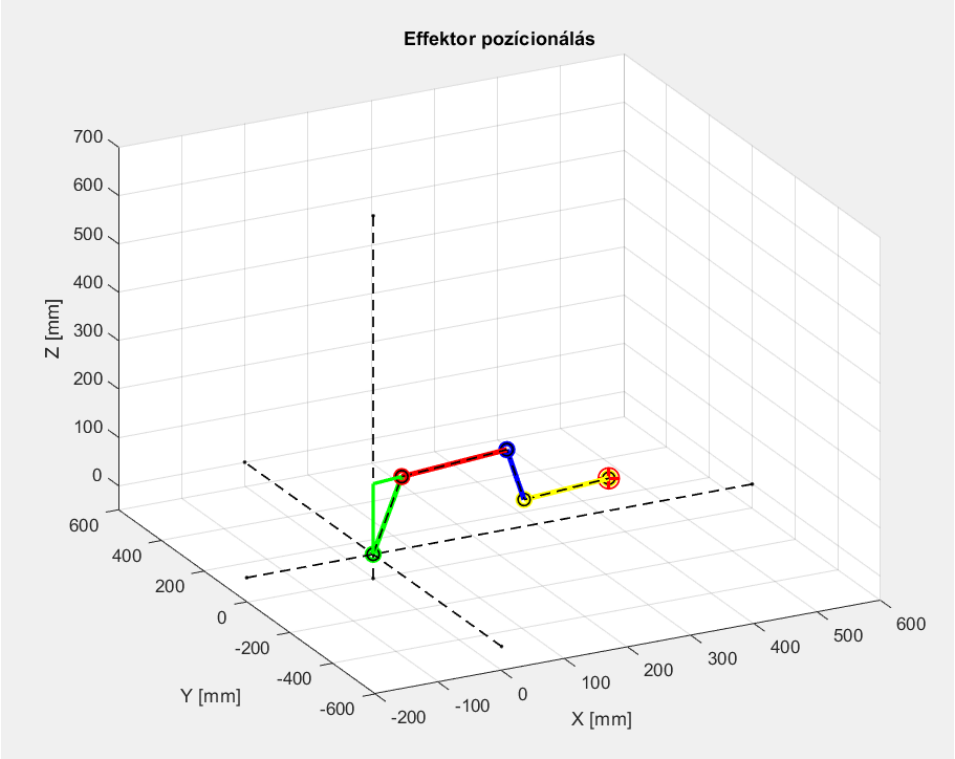
A bemeneti s világkoordináta vektor tartalma					
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	$auto_{q4}$	q_4 [°]	q_5 [°]
400	80	35	0	0	90

Kimeneti konzol üzenetek:

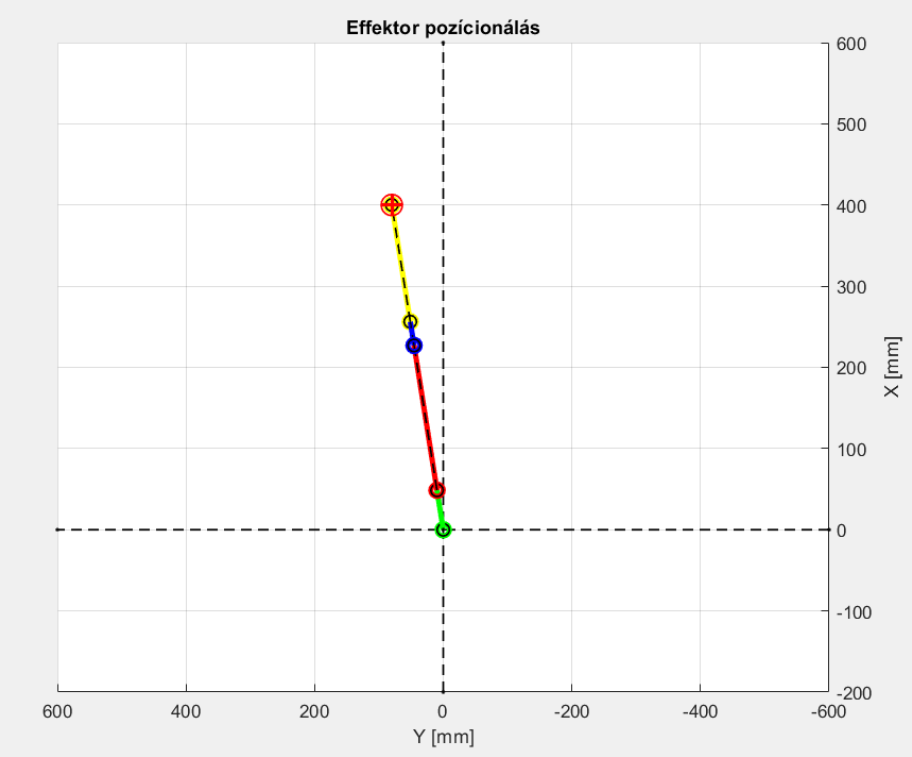


***48**. ábra - Konzol üzenet*

Generált diagram:



49. ábra - Kimeneti diagram

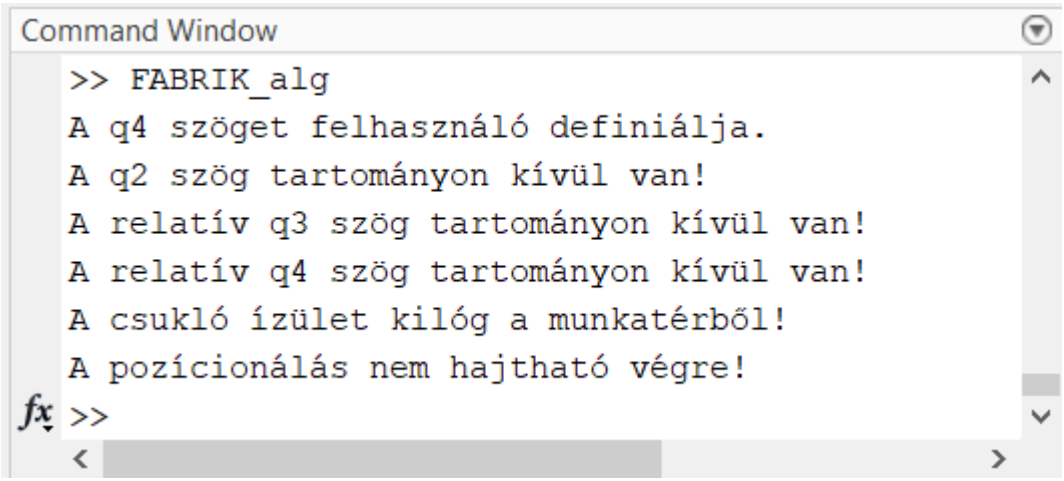


50. ábra - Kimeneti diagram

6.9. táblázat - Bemenet

A bemeneti s világkoordináta vektor tartalma					
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	$auto_{q4}$	q_4 [°]	q_5 [°]
200	-180	25	0	0	90

Kimeneti konzol üzenetek:



51. ábra - Konzol üzenet

Generált diagram: NINCS.

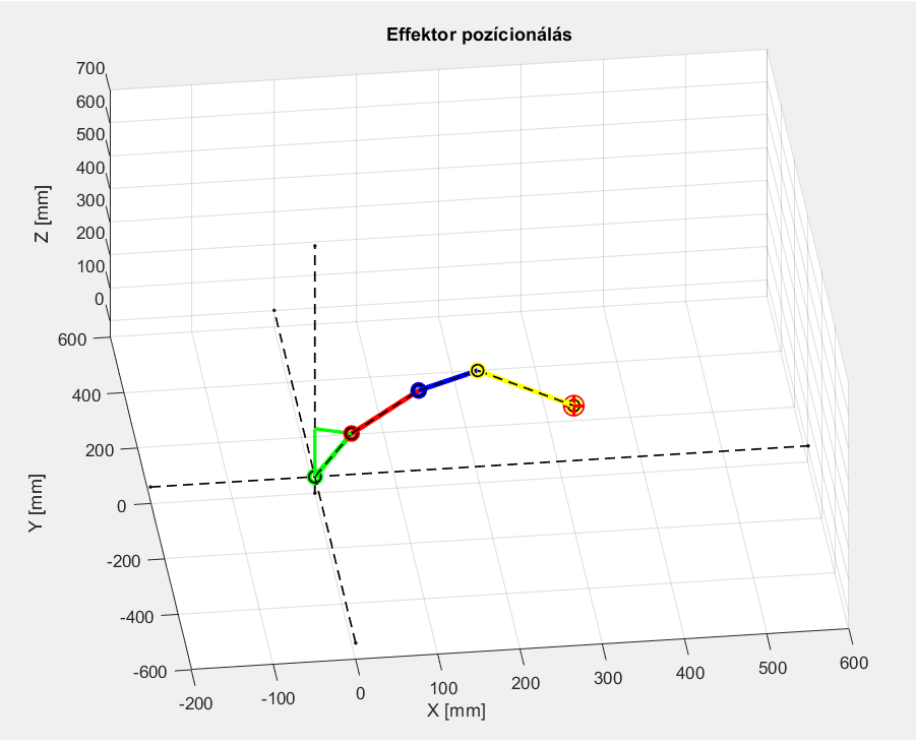
A bemeneti s világkoordináta vektor tartalma					
X_{TCP} [mm]	Y_{TCP} [mm]	Z_{TCP} [mm]	$auto_{q4}$	q_4 [°]	q_5 [°]
300	-180	320	1	0	90

Kimeneti konzol üzenetek:

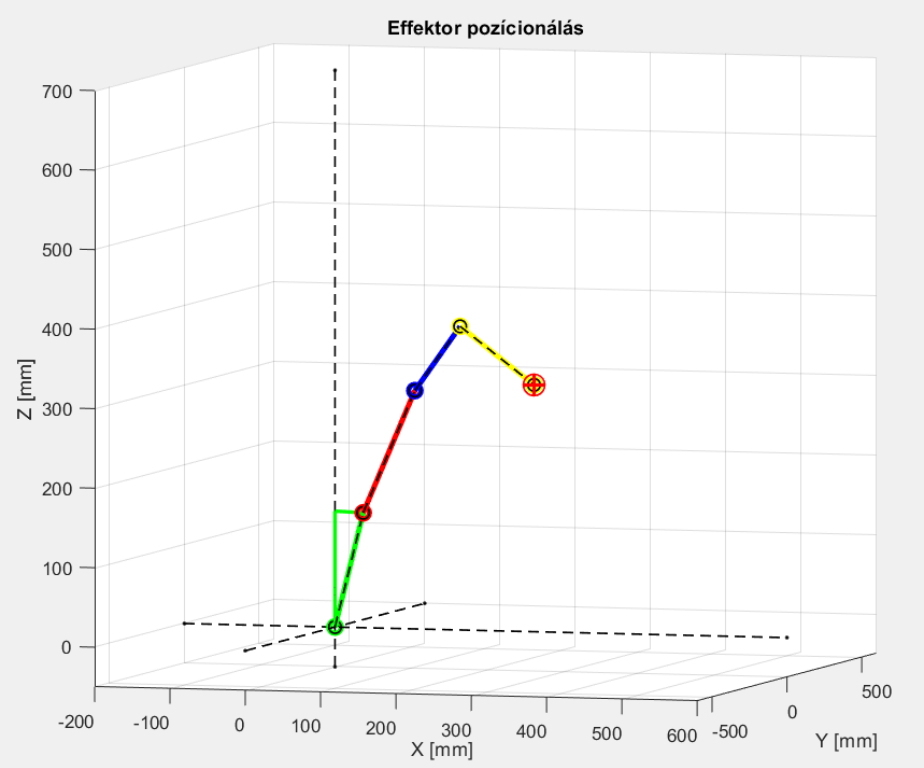
```
Command Window
>> FABRIK_alg
A q4 szög beállítása automatikus.
Törzs (q1) szög[°]:              -30.9638
Váll (q2) szög[°]:               60.0793
Könyök (q3) szög[°]:             46.6470
Csukló bólintás (q4) szög[°]:    -27.5855
Csukló csavarás (q5) szög[°]:    90.0000
TCP pozíció X hibája [mm]:       0.0010
TCP pozíció Y hibája [mm]:       0.0005
TCP pozíció Z hibája [mm]:       -0.0006

fx >>
```

52. ábra - Konzol üzenet

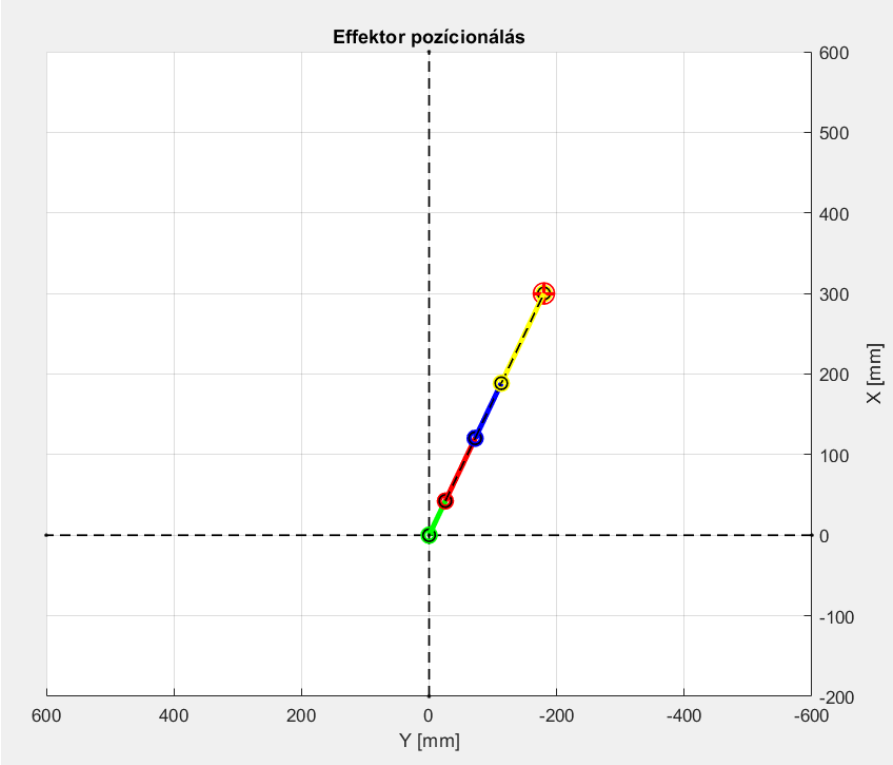
Generált diagram:



53. ábra - Kimeneti diagram



54. ábra - Kimeneti diagram



55. ábra - Kimeneti diagram

6.4.3 Eredmények kiértékelése

Amikor írtam a kódot, az iterációs limitet először tíz iterációra állítottam be, amit a ciklus nem ért el sok esetben, viszont, volt olyan eset, amikor a tizenkettedik iterációban tudott csak teljesülni a tolerancia. Erre való tekintettel kiterjesztettem húsz iterációra a limitet.

Az öt szabadsági fokú inverz kinematika megoldó algoritmustól azt vártam el, hogy gyors, pontos és egyszerű legyen, a működés megbízhatósága és a megismételhetőség szempontjából stabil közelítést és eredményeket szolgáltasson, valamint, hogy a megoldást geometriai szinten, szögként szolgáltassa. A FABRIK alapú inverz kinematika megoldó algoritmus, amit terveztem és fejlesztettem, szimulált környezetben az elvárásaimnak megfelelően működik.

Összefoglalás

A szakdolgozatom célkitűzése a prototípus csuklókaros robot inverz kinematika megoldó algoritmusának szoftveres megvalósítása volt. Az első fejezetben összegyűjtöttem az ipari robotika legfontosabb alapfogalmait, hogy érthetővé váljon a szakdolgozatom többi fejezete. A fejezet utolsó szakaszában felvezettem az inverz kinematikával kapcsolatban az alapvető problémát, melyre a további fejezetek megoldást hivatottak keresni és találni. A második fejezetben a direkt kinematika bevett megoldási módszerét, a Denavit-Hartenberg féle koordináta transzformációt mutattam be. A harmadik fejezetben az inverz kinematika megoldási módszereit vizsgáltam, mint az analitikus módszer, a klasszikus numerikus módszerek és a heurisztikus keresési stratégiák. A negyedik fejezetben bemutattam a prototípus csuklókaros robot terveit. Az ötödik fejezetben, minden addigi ismeret birtokában megoldást dolgoztam ki. Végül a hatodik fejezetben sor került a tervezett inverz kinematika megoldó algoritmus szoftveres megvalósítására. Jelen szakdolgozatban kidolgozott, megtervezett, fejlesztett és szimulált, Forward and Backward Reaching Inverse Kinematics módszer alapján működő inverz kinematika megoldó algoritmust alkalmazhatónak minősítem a 32 bites ARM környezetben megvalósítandó öt szabadsági fokú, csuklókaros robot manipulátor mozgásirányító szoftverrendszerének rendszerelemeként. A szakdolgozat témájának továbbfejlesztési lehetősége az említett mozgásirányító rendszer szoftveres megvalósítása és a prototípus robot üzembe állítása. Ennek részletei és folyamatai véleményem szerint, több szakdolgozat témájául is szolgálhatnak.

Summary

The objective of my thesis was the software implementation of the inverse kinematics solver algorithm of the prototype articulated robot. In the first chapter, I have collected the most important basic concepts of industrial robotics to make the rest of my thesis understandable. In the last section of the chapter I have outlined the basic problem of inverse kinematics, which the following chapters will seek and find a solution to. In the second chapter, I introduced an established solution method for direct kinematics, the Denavit-Hartenberg coordinate transformation. In the third chapter, solution methods for inverse kinematics such as the analytical method, classical numerical methods and heuristic search strategies are discussed. In the fourth chapter, I presented the design of the prototype articulated robot. In the fifth chapter, I elaborated a solution based on all the knowledge gained so far. Finally, in chapter six, the software implementation of the designed inverse kinematics solver algorithm was presented. The inverse kinematics solver algorithm based on the Forward and Backward Reaching Inverse Kinematics method, which was designed, developed and simulated in this thesis, is considered applicable as a system component of the motion control software system of the five degrees of freedom articulated robot manipulator to be implemented in a 32-bit ARM environment. The future research suggestion of my thesis are the software implementation of the said motion control system and the commissioning of the prototype robot. In my opinion, the details and processes of this could be the subject of several theses.

Irodalomjegyzék

[1] Magyar Szabványügyi Testület, MSZ EN ISO 10218-1:2011 - Robotok és robotszerkezetek. Ipari robotok biztonsági követelményei. 1. rész: Robotok (ISO 10218-1:2011), Budapest: Magyar Szabványügyi Testület, 2011.

[2] G. Mester, Robotika, Budapest: Typotex Kiadó, 2011.

[3] B. Kulcsár, Robottechnika I., Budapest: Typotex Kiadó, 2012.

[4] A. Zelei, „Mechanizmusok (BME GEMM NGME) 20200423 Direkt és inverz kinematika," Budapesti Műszaki és Gazdaságtudományi Egyetem, 23. 04. 2020. [Online]. Available: https://www.youtube.com/watch?v=RMgqGIYGA7A. [Hozzáférés dátuma: 21. 09. 2022].

[5] A. Sodemann, „Robotics 1 U1 (Kinematics) S3 (Rotation Matrices) P1 (Rotation Matrices)," 11. 06. 2017. [Online]. Available: https://www.youtube.com/watch?v=lVjFhNv2N8o. [Hozzáférés dátuma: 03. 10. 2023].

[6] B. Kulcsár, Robottechnika II., Budapest: Typotex Kiadó, 2012.

[7] A. Sodemann, „1 1 5 Lecture Video 1 of 1 Homogeneous Transformation Matrix Example and Coordinate Transformation," 10. 10. 2015. [Online]. Available: https://www.youtube.com/watch?v=tAu8-gkxAcE&t=710s. [Hozzáférés dátuma: 04. 10. 2023].

[8] A. Sodemann, „Robotics 1 U1 (Kinematics) S6 (Inverse Kinematics) P1 (Inverse Kinematics)," 14. 06. 2017. [Online]. Available: https://www.youtube.com/watch?v=3RHg6OJLYhM. [Hozzáférés dátuma: 10. 10. 2023].

[9] I. N. Bronstejn, G. Musiol, H. Mühlig és K. A. Szemengyajev, Matematikai kézikönyv, Budapest: Typotex Kiadó, 2009.

[10] M. Andalibi, „Numerical Inverse Kinematics Using the FABRIK Method," 12. 07. 2021. [Online]. Available: https://www.youtube.com/watch?v=nWbScy4joS0&t=642s. [Hozzáférés dátuma: 21. 10. 2023].

[11] M. Andalibi, „Numerical Inverse Kinematics Using the CCD Method,” 12. 07. 2021. [Online]. Available: https://www.youtube.com/watch?v=sWm77hrOZH4&t=2313s. [Hozzáférés dátuma: 20. 10. 2023].

[12] R. Freud, Lineáris Algebra, Budapest: ELTE Eötvös Kiadó, 2014.

[13] M. Andalibi, „Numerical Inverse Kinematics Using the Newton Raphson Method,” 12. 07. 2021. [Online]. Available: https://www.youtube.com/watch?v=_fVgZASO0Cw&t=3s. [Hozzáférés dátuma: 15. 10. 2023.].

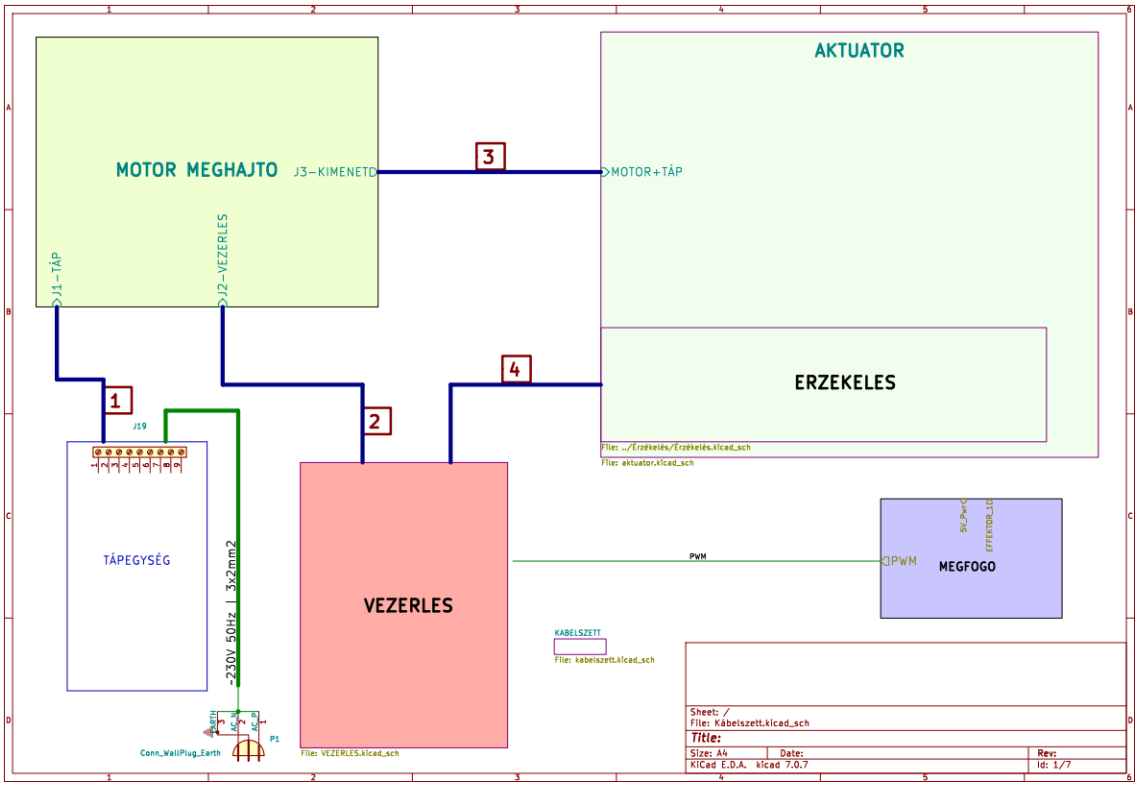
[14] A. Aristidou és J. Lasenby, „FABRIK: A fast, iterative solver for the Inverse Kinematics problem,” 15. 05. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1524070311000178. [Hozzáférés dátuma: 29. 11. 2023].

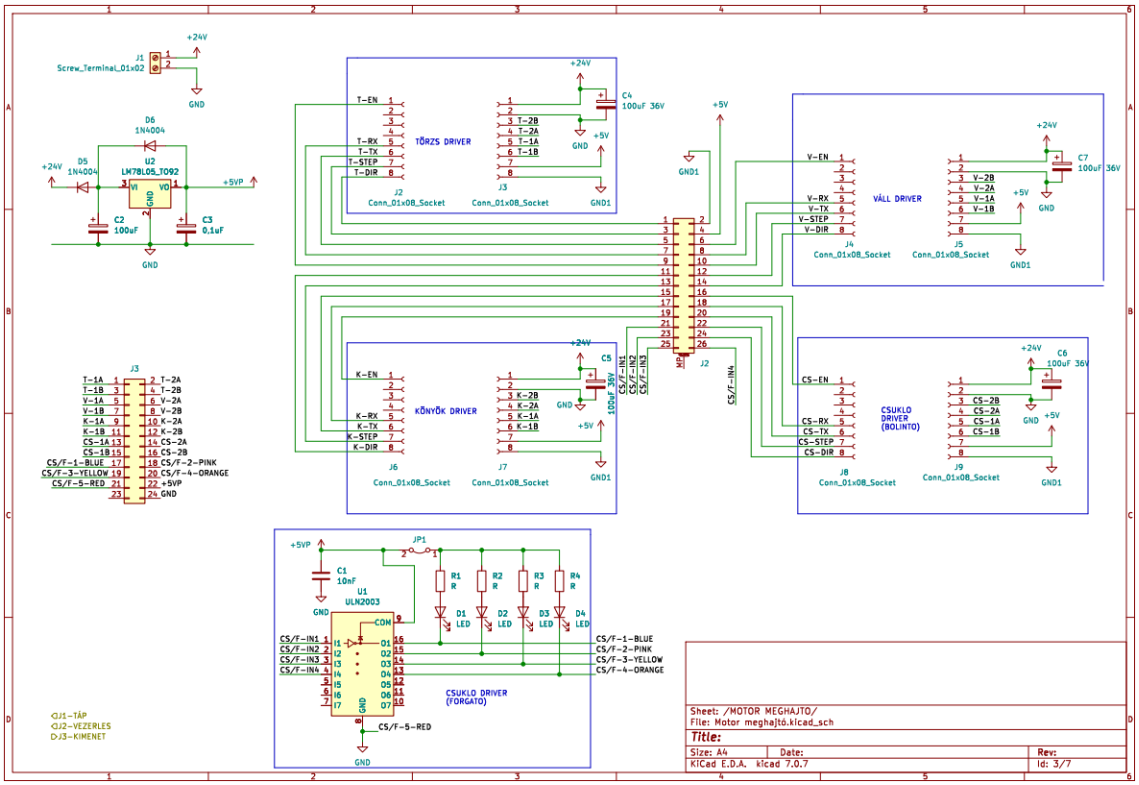
[15] Budapesti Műszaki és Gazdaságtudományi Egyetem, „4.1. Informált (heurisztikus) keresési stratégiák,” 2009. [Online]. Available: http://project.mit.bme.hu/mi_almanach/books/aima/ch04s01. [Hozzáférés dátuma: 01. 12. 2023].

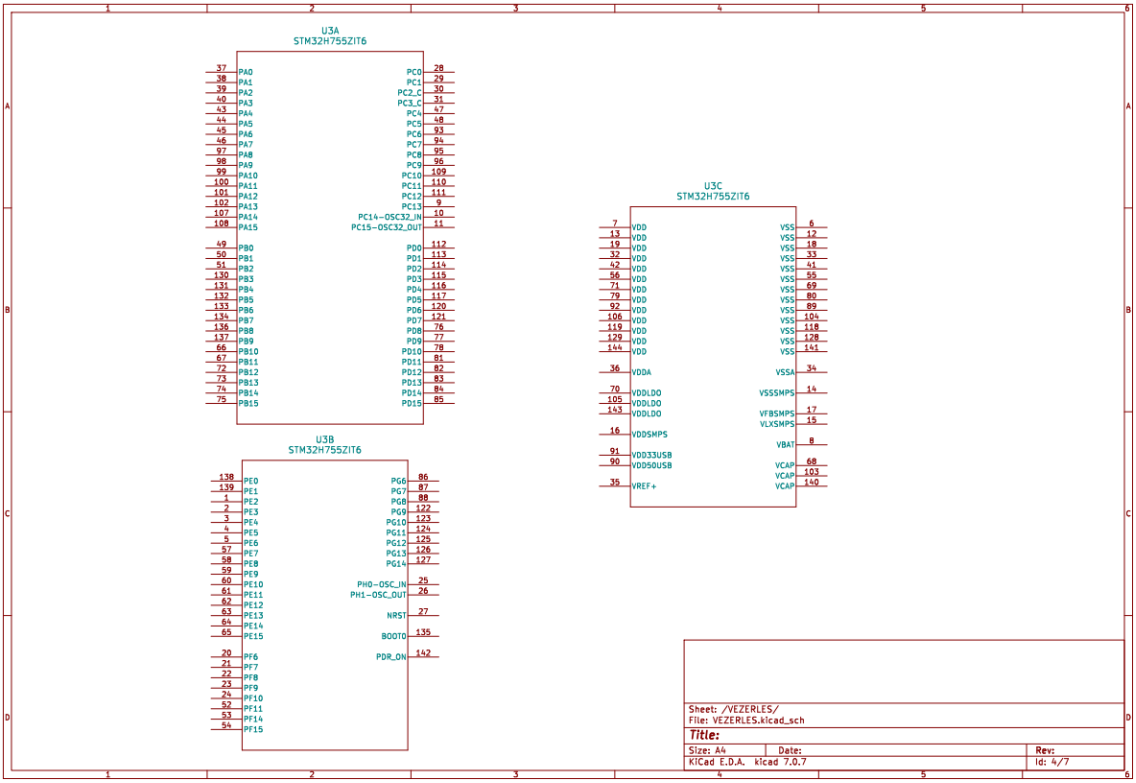
[16] F. Zsenák, Általános géptan, Győr: Széchenyi István Egyetem, 2007.

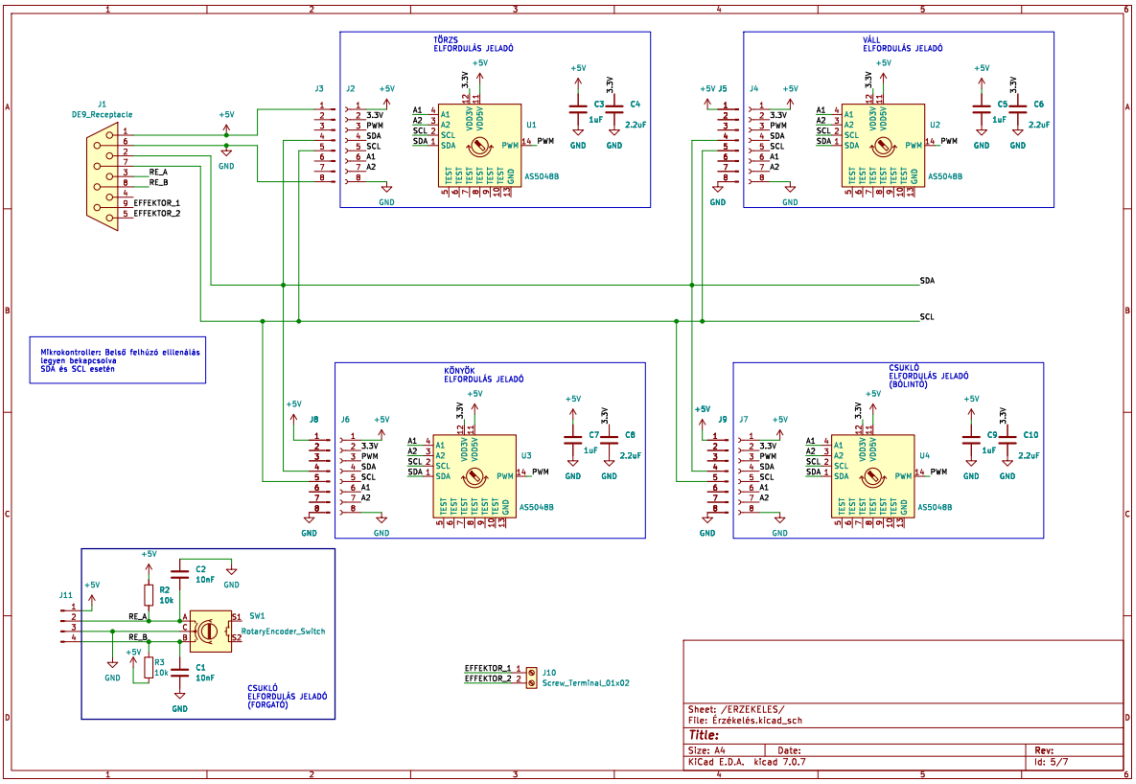
Mellékletek

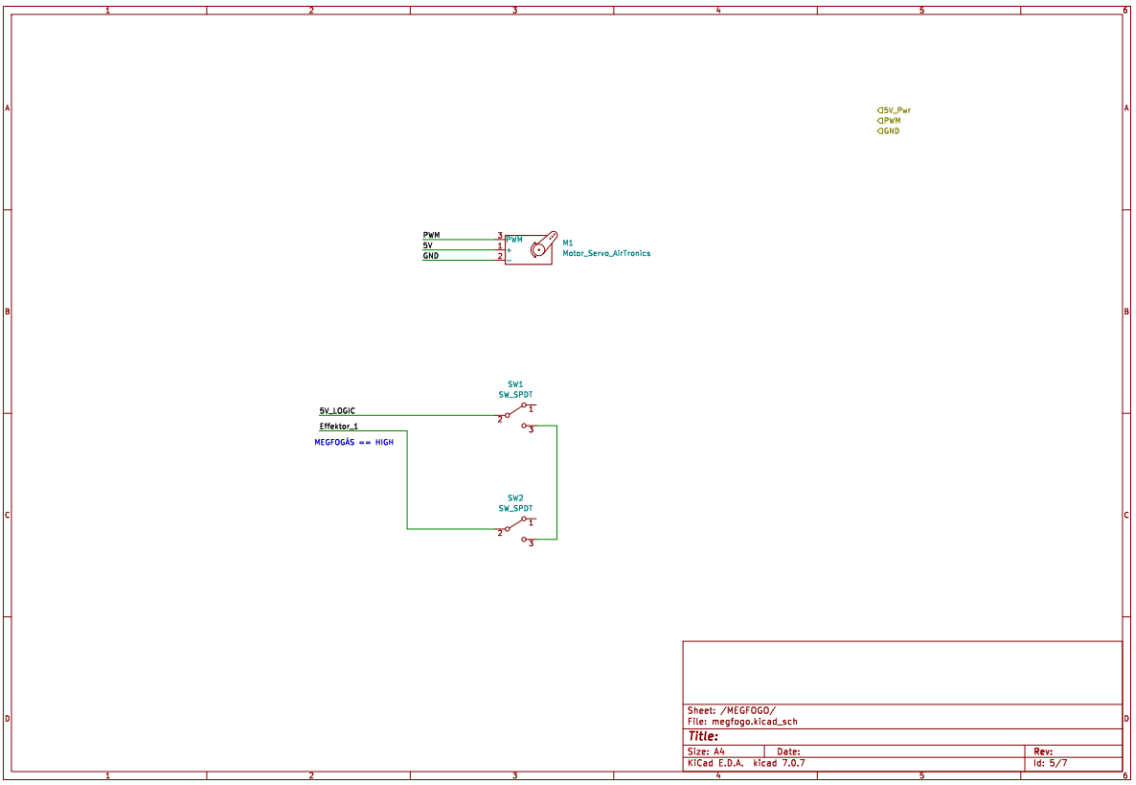
1. Melléklet – Prototípus robot kapcsolási rajzai

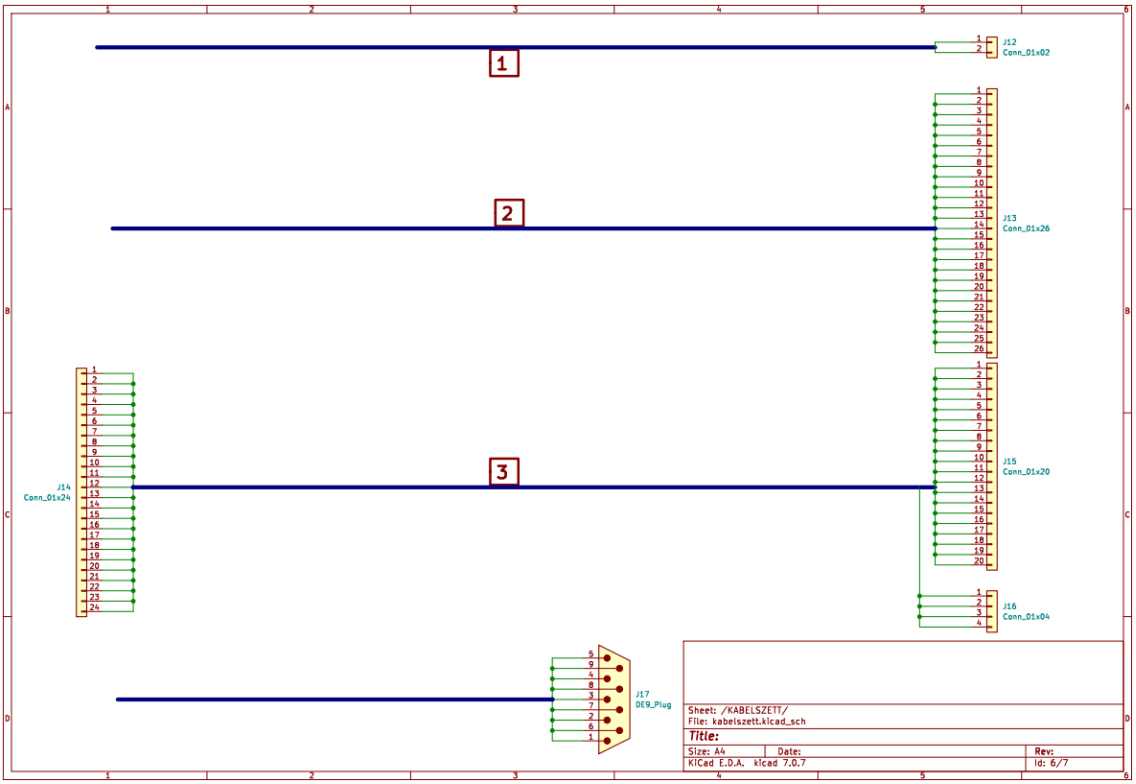


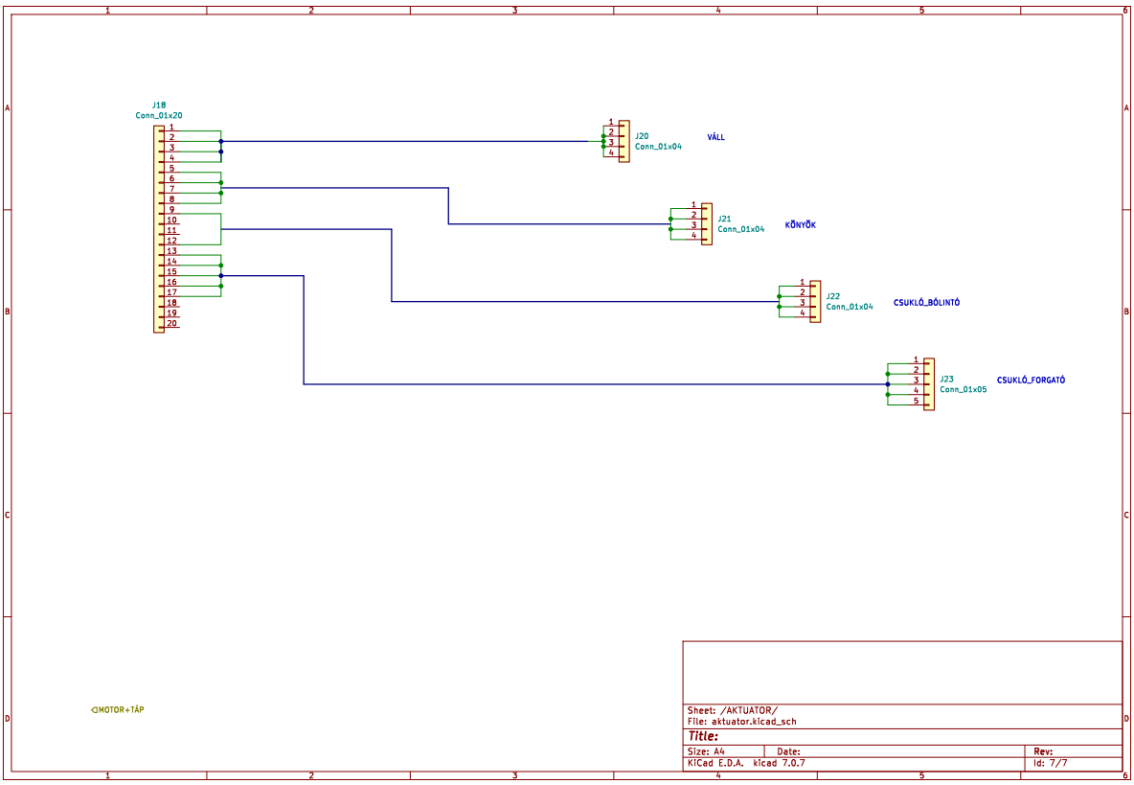



U3A
STM32H755ZIT6
U3C
STM32H755ZIT6
U3B
STM32H755ZIT6
Sheet: /VEZERLES/
File: VEZERLES.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 7.0.7
Id: 4/7


TÖRZS
ELFORDULÁS JELADÓ
VÁLL
ELFORDULÁS JELADÓ
KÖNYÖK
ELFORDULÁS JELADÓ
CSUKLÓ
ELFORDULÁS JELADÓ
(BÓLINTÓ)
CSUKLÓ
ELFORDULÁS JELADÓ
(FORGATÓ)
Mikrokontroller: Belső felhúzó ellenállás
legyen bekapcsolva
SDA és SCL esetén
SDA
SCL
Sheet: /ERZEKELES/
File: Érzékelés.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 7.0.7
Id: 5/7


5V_Pwr
PWM
GND
PWM
5V
GND
PWM
M1
Motor_Servo_AirTronics
SW1
SW_SPDT
5V_LOGIC
Effektor_1
MEGFOGÁS == HIGH
SW2
SW_SPDT
Sheet: /MEGFOGO/
File: megfogo.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 7.0.7
Id: 5/7


1
2
3
J12
Conn_01x02
J13
Conn_01x26
J14
Conn_01x24
J15
Conn_01x20
J16
Conn_01x04
J17
DE9_Plug
Sheet: /KABELSZETT/
File: kabelszett.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 7.0.7
Id: 6/7


J18
Conn_01x20
J20
Conn_01x04
VÁLL
J21
Conn_01x04
KÖNYÖK
J22
Conn_01x04
CSUKLÓ_BÓLINTÓ
J23
Conn_01x05
CSUKLÓ_FORGATÓ
MOTOR+TÁP
Sheet: /AKTUATOR/
File: aktuator.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 7.0.7
Id: 7/7

2. Melléklet – Newton-Raphson IK algoritmus Matlab kódja

```
%Newton-Raphson algoritmus teszt: (1-3) munkatér tartományra
 %Sánta Szabolcs
 %-------------------------------------------------------------

 a3 = 182;                  % Felkar axistávolság [mm]
 a4 = 118;                  % Alkar axistávolság [mm]

% 45° - 45° -> H13 = L13 = (a3+a4)/sqrt(2);

 H13 = 157;          % Z-koordináta (belső) [mm]
 L13 = 211;          % L(sík)-koordináta (belső) [mm]

 q2_0_deg = 90;                   % Initial Guess: Váll csuklóváltozó [deg]
 q3_0_deg = 0;                   % Initial Guess: Könyök csuklóváltozó [deg]

 q2_0 = q2_0_deg*(pi/180);          % Initial Guess: Váll csuklóváltozó [rad]
 q3_0 = q3_0_deg*(pi/180);          % Initial Guess: Könyök csuklóváltozó
[rad]

 x = [q2_0; q3_0];                  % kezdeti csuklóváltozók oszlopvektora

 F1 = a3*sin(q2_0) + a4*sin(q3_0) - H13;        % Trig egyenlet #1
 F2 = a3*cos(q2_0) + a4*cos(q3_0) - L13;        % Trig egyenlet #2

 F = [F1; F2];                  % Függvény oszlopvektor

 %J = [a3*cos(q2), a4*cos(q3); -a3*sin(q2), -a4*sin(q3)];    %Jacobi mátrix

 %-------------------------------------------------------------------------
 %  UPDATE EQUATION:
 %-------------------------------------------------------------------------
 max_iter = 100;                 % iteráció limit
 iit = zeros(1, max_iter+1);     % Iterációs indexek tömbje
 vi = 0;                         % végindex kezdeti érték

 xn = zeros(2, max_iter+1);      % csuklóváltozó-vektorok tömbje
 Fn = zeros(2, max_iter+1);      % függvény vektorok tömbje
 xn(:, 1) = x;                  % első vektor az initial guess
 Fn(:, 1) = F;

 e_max = 0.001;              % epsilon: függvény érték max megengedhető
eltérése nullától
 e = zeros(2, max_iter+1);
 d_max = 0.001;              % delta: max megengedhető zérushely hiba
 d = zeros(2, max_iter+1);

 fi = zeros(2, max_iter+1);

 for i = 1:max_iter

     %  Iterációs indexek tömbje
     iit(1, i) = i;
```

```
    %  Függvény vektor
    Fn(1, i) = a3*sin(xn(1, i)) + a4*sin(xn(2, i)) - H13;
    Fn(2, i) = a3*cos(xn(1, i)) + a4*cos(xn(2, i)) - L13;

    %  Jacobi mátrix: Trig egyenletrendszer szögek szerinti parciális
deriváltjainak mátrixa:
    Jn = [a3*cos(xn(1, i)), a4*cos(xn(2, i));
         -a3*sin(xn(1, i)), -a4*sin(xn(2, i))];
    %disp('Jacobi determináns:')
    jndet = det(Jn);

    %  ÚJ csuklóváltozó vektor
    if jndet == 0                           %   zérus determináns esete ->
pszeudo-inverz
        disp('PINV-Jacobi');                %   legyen nyoma a konzolban, ha
az inverz nem létezik
        xn(:, i+1) = xn(:, i) - pinv(Jn) * Fn(:, i);
    else                                    %   nem-zérus determináns ->
létezik inverz
        %disp('INV-Jacobi');
        xn(:, i+1) = xn(:, i) - inv(Jn) * Fn(:, i);
    end

    %  -180°... qn ...+180° korrekció
    fi(:, i) = mod(xn(:, i+1), 2*pi);

    if fi(:, i) > pi
        xn(:, i+1) = mod(fi(:, i), pi) - pi;          % +180-nál nagyobb
szög áthelyezése:     0 ... -180 tartományba
    else
        if fi(:, i) < (-pi)
            xn(:, i+1) = mod(fi(:, i), pi) + pi;      % -180-nál kisebb
szög áthelyezése:      0 ... +180 tartományba
        end
    end

    %  Tűréshatárok
    e(:, i) = abs(Fn(:, i));
    d(:, i) = abs(xn(:, i+1) - xn(:, i));
    if e(:, i) < e_max
        if d(:, i) < d_max
            vi = i;            % végindex
            break;
        end
    end
end

if vi == 0
    vi = max_iter - 1;
end


x_deg = rad2deg(xn);


q2 = x_deg(1, vi);     % közelített váll szög
```

```
q3 = x_deg(2, vi);     % közelített könyök szög
fprintf("\nVáll: \t\tq2 = %f °\n", q2);
fprintf("\nKönyök: \tq3 = %f °\n", q3);

z1 = 0;                    % váll koordináták
l1 = 0;
z2 = a3*sin(q2*(pi/180));           % könyök koordináták
l2 = a3*cos(q2*(pi/180));
z3 = z2 + a4*sin(q3*(pi/180));      % csukló koordináták
l3 = l2 + a4*cos(q3*(pi/180));

figure('Name','Inverz Kinematikai Diagram', 'Position',[50, 100, 550, 600]);

plot([l1, l2], [z1, z2], 'r-o', 'LineWidth', 2);
hold on;
plot([l2, l3], [z2, z3], 'b-o', 'LineWidth', 2);
hold on;
plot(L13, H13, 'yo', 'LineWidth', 3);
hold on;
plot(L13, H13, 'r.','LineWidth', 5);
hold off;
axis equal;
title('Robotkar pozíció (Váll-Könyök-Csukló)');
xlabel('L: Vízszintes pozíció [mm]');
ylabel('Z: Függőleges pozíció [mm]');

figure('Name','Iterációs Állapotok','position',[600, 100, 850, 600]);

subplot(4,2,1);
plot(iit(1, :), x_deg(1, :), 'r-o');
title('q_2 - Csuklóváltozó (Váll szöge)');
xlabel('iteráció');
ylabel('Szög [°]');

subplot(4,2,3);
plot(iit(1, :), x_deg(2, :), 'b-o');
title('q_3 - Csuklóváltozó (Könyök szöge)');
xlabel('iteráció');
ylabel('Szög [°]');

subplot(4,2,5);
plot(iit(1, :), Fn(1, :));
title('Függőleges pozíciófüggvény zérusérték: F_1(q_2,q_3) ---> 0');
xlabel('iteráció');
ylabel('pozíció [mm]');

subplot(4,2,7);
plot(iit(1, :), Fn(2, :));
title('Vízszintes pozíciófüggvény zérusérték: F_2(q_2,q_3) ---> 0');
xlabel('iteráció');
ylabel('pozíció [mm]');

subplot(4,2,2);
plot(iit(1, :), d(1, :), 'r-o');
title('Iterációnkénti szög-eltérés: delta_q_2');
xlabel('iteráció');
```

```
ylabel('szög-eltérés [rad]');

subplot(4,2,4);
plot(iit(1, :), d(2, :), 'b-o');
title('Iterációnkénti szög-eltérés: delta_q_3');
xlabel('iteráció');
ylabel('szög-eltérés [rad]');

subplot(4,2,6);
plot(iit(1, :), e(1, :));
title('F_1 zérusérték hiba: epsilon_1');
xlabel('iteráció');
ylabel('zérushiba [mm]');

subplot(4,2,8);
plot(iit(1, :), e(2, :));
title('F_2 zérusérték hiba: epsilon_2');
xlabel('iteráció');
ylabel('zérushiba [mm]');

%fprintf("H13 = %f\n", H13);
%fprintf("L13 = %f\n", L13);
```

3. Melléklet – Analitikus geometriai IK algoritmus Matlab kódja

```
%   Analitikus Geometriai Inverz Kinematika Megoldó
%   Sánta Szabolcs

%-------------------------------------------------------
%   Talp-Váll L-Z síkú offset
Z01 = 145.8;                %   [mm]    függőlegesen
L01 = 49.5;                 %   [mm]    vízszintesen
R01 = sqrt(L01^2 + Z01^2);
jp1 = [L01;Z01];
%   Szegmens hosszok
R12 = 182;                  %   [mm]    Váll-könyök
R23 = 116;                  %   [mm]    Könyök-csukló
R35 = 109.83 + 33.78 + 3;   %   [mm]    Csukló-TCP
R15max = R12+R23+R35;

%   TCP célpozíció
Z_tcp = 312;                %   [mm]    Fel-Le
Y_tcp = 86;                 %   [mm]    Jobbra-Balra
X_tcp = 190;                %   [mm]    Előre-Hátra
L_tcp = sqrt(X_tcp^2 + Y_tcp^2);
R15 = sqrt((L_tcp-L01)^2 + (Z_tcp-Z01)^2);
%   TCP orientáció
Q4_tcp = 48;                %   [°]     Lokális Pitch
Z35 = R35*sin(deg2rad(Q4_tcp));     %   Pitch függőleges vetület
L35 = R35*cos(deg2rad(Q4_tcp));     %   Pitch vízszintes vetület
Q5_tcp = 90;                %   [°]     Lokális Roll

%   Függőleges(L-Z) belső forgássík méretezése
%{
Z13 = Z_tcp - Z01 - Z35;
L13 = L_tcp - L01 - L35;
jp3 = jp1 + [L13;Z13];
R13 = sqrt(L13^2 + Z13^2);
%}

jp3 = [L_tcp-L35; Z_tcp-Z35];
v13 = jp3 - jp1;
R13 = norm(v13);
%jp2u = [0;0];                  %   két lehetséges könyök pozícióvektor init
%jp2d = [0;0];
%-------------------------------------------------------

%-----------------------------------------------------------
%   IK: q1: Törzs;  VÍZSZINTES FORGÁSSÍK    (X-Y)
q1 = rad2deg(atan2(Y_tcp, X_tcp));

if q1 <= 105 && q1 >= -105
    %
else
    disp("hiba: q1 túl nagy")
end

%   IK: q4: Pitch,  q5: Roll
```

```
q4 = Q4_tcp;                            %   Bemenet definiálja
q5 = Q5_tcp;                            %   Bemenet definiálja

%   IK: q2: Váll,   q3: Könyök; FÜGGŐLEGES FORGÁSSÍK    (L-Z)

%   KOSZINUSZTÉTELBŐL:
cosa = (R12^2 + R13^2 - R23^2)/(2*R12*R13);
cosb = (R12^2 + R23^2 - R13^2)/(2*R12*R23);

if cosa <= 1 && cosa >= -1 && cosb <= 1 && cosb >= -1
    alfa = acos(cosa);
    beta = acos(cosb);


    %   1--->2 eltolási skalárok:
    R12CA = R12*cos(alfa);
    R12SA = R12*sin(alfa);

    %   relatív könyök szög, szimmetrikus elbow-up és elbow-down
    %   konfigurációban
    q3r = 180 - rad2deg(beta);

    %   Egységhosszú 1--->3 irányvektor
    %e13 = [jp3(1)-jp1(1); jp3(2)-jp1(2)] / sqrt((jp3(1)-jp1(1))^2 + (jp3(2)-
jp1(2))^2);
    e13 = v13/R13;
    %   Egységhosszú 1--->3 irányvektor elbow-UP normálvektora
    nu = [-e13(2); e13(1)];
    %   Egységhosszú 1--->3 irányvektor elbow-DOWN normálvektora
    nd = [e13(2); -e13(1)];
    %   Deltoid középpont (átlók metszéspontja) 1--->c eltolás
    c = R12CA*e13 + jp1;

    %Könyök lehetséges két pozícióvektora a centrumból eltolva:  (2u<---c---
>2d)
    jp2u = R12SA*nu + c;            %   ELBOW-UP
    jp2d = R12SA*nd + c;            %   ELBOW-DOWN

    u12 = (jp2u-jp1)/norm(jp2u-jp1);
    d12 = (jp2d-jp1)/norm(jp2d-jp1);

    %   Lehetséges (abszolút) váll szögek a két könyök konfiguráció esetére
    q2u = rad2deg(atan2(u12(2),u12(1)));
    q2d = rad2deg(atan2(d12(2),d12(1)));

    %   Könyökpozíciók és csukló pozíció közötti egység-irányvektorok: (2u---
>3<---2d)

    u23 = (jp3-jp2u)/norm(jp3-jp2u);
    d23 = (jp3-jp2d)/norm(jp3-jp2d);
    %   KIFEJTETT:
    %u23 = [jp3(1)-jp2u(1); jp3(2)-jp2u(2)] / sqrt((jp3(1)-jp2u(1))^2 +
(jp3(2)-jp2u(2))^2);
    %d23 = [jp3(1)-jp2d(1); jp3(2)-jp2d(2)] / sqrt((jp3(1)-jp2d(1))^2 +
(jp3(2)-jp2d(2))^2);
```

```
    %   Lehetséges (abszolút) könyök szögek a könyök-csukló irányvektorok
alapján
    q3u = rad2deg(atan2(u23(2),u23(1)));
    q3d = rad2deg(atan2(d23(2),d23(1)));

    disp("q2_UP [°] = ");
    disp(q2u);
    disp("q2_DOWN [°] = ");
    disp(q2d);
    disp("q3_UP [°] = ");
    disp(q3u);
    disp("q3_DOWN [°] = ");
    disp(q3d);

    figure
    plot([-150, 550],[0, 0], 'k-', 'LineWidth', 3);
    hold on;
    plot([0, L01], [Z01, Z01], 'g-', 'LineWidth', 2);
    hold on;
    plot([0, 0], [0, Z01], 'g-', 'LineWidth', 2);
    hold on;
    plot([0, L01], [0, Z01], 'g-', 'LineWidth', 2);
    hold on;
    plot([jp1(1), jp2u(1)], [jp1(2), jp2u(2)], 'r-o', 'LineWidth', 2);
    hold on;
    plot([jp1(1), jp2d(1)], [jp1(2), jp2d(2)], 'r--o', 'LineWidth', 2);
    hold on;
    plot([jp2u(1), jp3(1)], [jp2u(2), jp3(2)], 'b-o', 'LineWidth', 2);
    hold on;
    plot([jp2d(1), jp3(1)], [jp2d(2), jp3(2)], 'b--o', 'LineWidth', 2);
    hold on;
    plot([jp3(1), L_tcp], [jp3(2), Z_tcp], 'y-o', 'LineWidth', 2);
    hold on;
    plot(L_tcp, Z_tcp, 'g-o', 'LineWidth', 2);
    hold on;
    plot(c(1), c(2), 'k-o', 'LineWidth', 2);
    hold off;
    title('Inverz Kinematika L-Z függőleges forgó sík');
    xlabel('L - Vízszintes távolság [mm]');
    ylabel('Z - Függőleges távolság [mm]');
    axis equal;
    grid on;
    ylim([-100,550]);
    xlim([-150,550]);
else
    disp("hiba: célkoordináták nem elérhetőek");
end
```

4. Melléklet – FABRIK alapú ellenőrzött IK algoritmus Matlab kódja

```
%   FORWARD AND BACKWARD REACHING INVERSE KINEMATICS
%   5 DoF
%   Sánta Szabolcs
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%   INIT

R35 = 109.83 + 33.78 + 3;
leng = [182, 116, R35];        %   szegmens hosszok
q = zeros(5,1);
qinit = [deg2rad(60), deg2rad(45), deg2rad(30)];
Z01 = 145.8;                    %   [mm]    függőlegesen
L01 = 49.5;                     %   [mm]    vízszintesen
R01 = sqrt(L01^2 + Z01^2);      %   [mm]    0-1 offset hossz

%   kezdeti csuklópozíciók oszlopvektorai
p1init = [L01; Z01];
p2init = [p1init(1)+leng(1)*cos(qinit(1)); p1init(2)+leng(1)*sin(qinit(1))];
p3init = [p2init(1)+leng(2)*cos(qinit(2)); p2init(2)+leng(2)*sin(qinit(2))];
p4init = [p3init(1)+leng(3)*cos(qinit(3)); p3init(2)+leng(3)*sin(qinit(3))];

%   egység irányvektorok forward init
e21 = [0;0];
e32 = [0;0];
e43 = [0;0];
%   egység irányvektorok backward init
e12 = [0;0];
e23 = [0;0];
e34 = [0;0];

% maximális ismétlési szám
maxiter = 20;
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%   Felhasználói világkoordináták

%s = [Xtcp; Ytcp; Ztcp; autoQ4 flag; q4; q5];
s = [300; -180; 320; 1; 0; 90];

if s(4) > 0
    s(4) = 1;       %   q4 beállítása automatikus
    fprintf('A q4 szög beállítása automatikus.\n');
else
    s(4) = 0;       %   q4 beállítása a felhasználó által rögzített
    fprintf('A q4 szöget felhasználó definiálja.\n');
end
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%   Inverz Kinematika
```

```
% cél(l,z) oszlopvektor l-z síkon

c = [sqrt(s(1)^2 + s(2)^2); s(3)];

if s(6) <= 90 && s(6) >= -90
    q(5) = s(6);                    % Roll szög
    OKq5 = 1;                   % q5 szög OK flag
else
    disp("A q5 szög tartományon kívül van!");
    OKq5 = 0;
end

if s(4) == 0
    if s(5) <= 105 && s(5) >= -90           %abszolút-q4 ellenőrzés
        q(4) = s(5);    % felhasználó által rögzített Pitch szög
        w = [c(1)-leng(3)*cos(deg2rad(q(4))); c(2)-
leng(3)*sin(deg2rad(q(4)))];
        OKq4abs = 1;                % q4 abszolút szög OK flag
    else
        disp("A felhasználói q4 abszolút szög tartományon kívül van!");
        OKq4abs = 0;
    end
end
% közelítési hiba vektor    [mm]
error = [0;0];
errormax = [0.01; 0.01];
% átmeneti pozíció tömb init
p1temp = p1init;
p2temp = p2init;
p3temp = p3init;
p4temp = p4init;
% forward pozícióvektorok init
p1fwd = [0;0];
p2fwd = [0;0];
p3fwd = [0;0];
p4fwd = [0;0];
% backward pozícióvektorok init
p1bwd = [0;0];
p2bwd = [0;0];
p3bwd = [0;0];
p4bwd = [0;0];

%   céltávolság ellenőrzés
if s(4) == 1
    if norm(c-p1init) <= (leng(1)+leng(2)+leng(3))
        OKtav = 1;                          % céltávolság OK flag
    else
        disp("A cél túl messze van!");
        OKtav = 0;
    end
else
    if s(4) == 0
        if norm(w-p1init) <= (leng(1)+leng(2))
            OKtav = 1;
        else
            disp("A cél túl messze van!");
```

```
            OKtav = 0;
        end
    end
end



%   q1 ellenőrzése
q(1) = rad2deg(atan2(s(2),s(1)));
if q(1) < 105 && q(1) > -105
    OKq1 = 1;                   % q1 szög OK flag
else
    disp("A q1 tartományon kívül van!");
    OKq1 = 0;
end

%   FABRIK  ------------------------------------------------------

if OKq1==1 && OKtav==1 && OKq5==1 && (s(4)==1 || OKq4abs==1)    %OK flagek
ellenőrzése
    for iter = 1:maxiter

        %   FORWARD CIKLUS
        if s(4) == 1
            p4fwd = c;
            e43 = (p3temp - p4fwd) / norm(p3temp - p4fwd);
            p3fwd = p4fwd + leng(3) .* e43;
            e32 = (p2temp - p3fwd) / norm(p2temp - p3fwd);
            p2fwd = p3fwd + leng(2) .* e32;
            e21 = (p1temp - p2fwd) / norm(p1temp - p2fwd);
            p1fwd = p2fwd + leng(1) .* e21;
        else    %s(4) == 0
            p3fwd = w;
            e32 = (p2temp - p3fwd) / norm(p2temp - p3fwd);
            p2fwd = p3fwd + leng(2) .* e32;
            e21 = (p1temp - p2fwd) / norm(p1temp - p2fwd);
            p1fwd = p2fwd + leng(1) .* e21;
        end


        %   BACKWARD CIKLUS
        if s(4) == 1
            p1bwd = p1init;
            e12 = (p2fwd - p1bwd) / norm(p2fwd - p1bwd);
            p2bwd = p1bwd + leng(1) .* e12;
            e23 = (p3fwd - p2bwd) / norm(p3fwd - p2bwd);
            p3bwd = p2bwd + leng(2) .* e23;
            e34 = (p4fwd - p3bwd) / norm(p4fwd - p3bwd);
            p4bwd = p3bwd + leng(3) .* e34;
        else    %s(4)==0
            p1bwd = p1init;
            e12 = (p2fwd - p1bwd) / norm(p2fwd - p1bwd);
            p2bwd = p1bwd + leng(1) .* e12;
            e23 = (p3fwd - p2bwd) / norm(p3fwd - p2bwd);
            p3bwd = p2bwd + leng(2) .* e23;
        end
```

```
        %   Átmeneti pozíciók és csuklószögek
        p1temp = p1bwd;
        p2temp = p2bwd;
        p3temp = p3bwd;

        q(2) = rad2deg(atan2(e12(2), e12(1)));
        q(3) = rad2deg(atan2(e23(2), e23(1)));

        if s(4) == 1
            p4temp = p4bwd;
            q(4) = rad2deg(atan2(e34(2), e34(1)));
        end

        %   Közelítési hibavektor
        if s(4) == 1
            error = p4temp - c;     %   hibavektor
            if norm(error) < norm(errormax)
                break;
            end
        else    %s(4)==0
            error = p3temp - w;     %   hibavektor
            if norm(error) < norm(errormax)
                break;
            end
        end

    end
else
    disp("A pozícionálás nem hajtható végre!");
end

if s(4) == 0
    p4temp = [p3temp(1)+leng(3)*cos(deg2rad(q(4)));
p3temp(2)+leng(3)*sin(deg2rad(q(4)))];
end

%   CSUKLÓTÉR ELLENŐRZÉS    ---------------------------------------------

%   abszolút q2 ellenőrzése
if q(2) >= 0 && q(2) <= 105
    OKq2 = 1;                   %   q2 OK flag
else
    OKq2 = 0;
    disp("A q2 szög tartományon kívül van!");
end

q3r = q(3) - q(2);      % Relatív q3 szög
q4r = q(4) - q(3);      % Relatív q4 szög

%   Relatív q3 ellenőrzése
if q3r <= 105 && q3r >= -105
    OKq3r = 1;              %   q3r OK flag
else
    OKq3r = 0;
    disp("A relatív q3 szög tartományon kívül van!");
```

```
end

%   Relatív q4 ellenőrzése és a szegmens-keresztezés kizárása
if q4r <= 105 && q4r >= -105 && abs(q4r) <= (180-abs(q3r))
    OKq4r = 1;              %   q4r OK flag
else
    OKq4r = 0;
    disp("A relatív q4 szög tartományon kívül van!");
end
%-------------------------------------------------------------------------

%   MUNKATÉR ELLENŐRZÉS    ---------------------------------------------

fi13 = rad2deg(atan2(p3temp(2)-p1temp(2), p3temp(1)-p1temp(1)));     % váll-
csukló irányszög
fi14 = rad2deg(atan2(p3temp(2)-p1temp(2), p3temp(1)-p1temp(1)));     % váll-
TCP irányszög

if fi13 <= 105 && fi13 >= 0
    OKts3 = 1;              % csuklópozíció a munkatérben OK flag
else        % fi13 < 0
    if fi13 < 0 && p3temp(1) >= (L01+leng(1)) && p3temp(1) <=
(L01+leng(1)+leng(2)+leng(3)) && p3temp(2) >= 10
        OKts3 = 1;
    else
        OKts3 = 0;
        disp("A csukló ízület kilóg a munkatérből!");
    end
end

if fi14 <= 105 && fi14 >= 0
    OKts4 = 1;              % szerszámpozíció a munkatérben OK flag
else        % fi14 < 0
    if fi14 < 0 && p4temp(1) >= (L01+leng(1)) && p4temp(1) <=
(L01+leng(1)+leng(2)+leng(3)) && p4temp(2) >= 10
        OKts4 = 1;
    else
        OKts4 = 0;
        disp("A szerszám kilóg a munkatérből!");
    end
end
%-------------------------------------------------------------------------

%   EFFEKTOR POZíCIONÁLÁS   ---------------------------------------------
%   OK flagek ellenőrzése:
if OKtav==1 && OKq1==1 && OKq2==1 && OKq3r==1 && OKq4r==1 && (s(4)==1 ||
OKq4abs==1) && OKq5==1 && OKts3==1 && OKts4==1
    p1xyz = [p1temp(1)*cos(deg2rad(q(1))), p1temp(1)*sin(deg2rad(q(1))),
p1temp(2)];
    p2xyz = [p2temp(1)*cos(deg2rad(q(1))), p2temp(1)*sin(deg2rad(q(1))),
p2temp(2)];
    p3xyz = [p3temp(1)*cos(deg2rad(q(1))), p3temp(1)*sin(deg2rad(q(1))),
p3temp(2)];
    p4xyz = [p4temp(1)*cos(deg2rad(q(1))), p4temp(1)*sin(deg2rad(q(1))),
p4temp(2)];
    X01 = L01*cos(deg2rad(q(1)));
```

```
    Y01 = L01*sin(deg2rad(q(1)));

    figure('Name','Munkatér', 'Position',[50, 100, 750, 600]);
    axis([-200 600 -600 600 -50 700]);
    grid on;
    xlabel('X [mm]');
    ylabel('Y [mm]');
    zlabel('Z [mm]');
    view(3);
    title("Effektor pozícionálás");
    hold on;
    plot3([-200, 600],[0, 0], [0, 0], 'k--.', 'LineWidth', 1);
    hold on;
    plot3([0, 0], [-600, 600], [0, 0], 'k--.', 'LineWidth', 1);
    hold on;
    plot3([0, 0], [0, 0], [-50, 700], 'k--.', 'LineWidth', 1);
    hold on;
    plot3([0, 0], [0, 0], [0, Z01], 'g-', 'LineWidth', 2);
    hold on;
    plot3([0, X01], [0, Y01], [Z01, Z01], 'g-', 'LineWidth', 2);
    hold on;
    plot3([0, p1xyz(1)],[0, p1xyz(2)], [0, p1xyz(3)], 'g-o', 'MarkerSize', 7,
'LineWidth', 3);
    hold on;
    plot3([0, p1xyz(1)],[0, p1xyz(2)], [0, p1xyz(3)], 'k--o', 'MarkerSize',
7, 'LineWidth', 1);
    hold on;
    plot3([p1xyz(1), p2xyz(1)],[p1xyz(2), p2xyz(2)], [p1xyz(3), p2xyz(3)],
'r-o', 'MarkerSize', 7, 'LineWidth', 3);
    hold on;
    plot3([p2xyz(1), p3xyz(1)],[p2xyz(2), p3xyz(2)], [p2xyz(3), p3xyz(3)],
'b-o', 'MarkerSize', 7, 'LineWidth', 3);
    hold on;
    plot3([p3xyz(1), p4xyz(1)],[p3xyz(2), p4xyz(2)], [p3xyz(3), p4xyz(3)],
'y-o', 'MarkerSize', 7, 'LineWidth', 3);
    hold on;
    plot3([p1xyz(1), p2xyz(1)],[p1xyz(2), p2xyz(2)], [p1xyz(3), p2xyz(3)],
'k--o', 'MarkerSize', 7, 'LineWidth', 1);
    hold on;
    plot3([p2xyz(1), p3xyz(1)],[p2xyz(2), p3xyz(2)], [p2xyz(3), p3xyz(3)],
'k--o', 'MarkerSize', 7, 'LineWidth', 1);
    hold on;
    plot3([p3xyz(1), p4xyz(1)],[p3xyz(2), p4xyz(2)], [p3xyz(3), p4xyz(3)],
'k--o', 'MarkerSize', 7, 'LineWidth', 1);
    hold on;
    plot3(s(1), s(2), s(3), 'ro', "MarkerSize", 12, 'LineWidth', 1);
    hold on;
    plot3(s(1), s(2), s(3), 'r+', "MarkerSize", 12, 'LineWidth', 2);
    hold off;

    q1 = q(1);
    q2 = q(2);
    q3 = q(3);
    q4 = q(4);
    q5 = q(5);
```

```
    %   Csuklókoordináták
    fprintf('Törzs (q1) szög[°]:\t\t\t\t%.4f\n', q1);
    fprintf('Váll (q2) szög[°]:\t\t\t\t%.4f\n', q2);
    fprintf('Könyök (q3) szög[°]:\t\t\t%.4f\n', q3);
    fprintf('Csukló bólintás (q4) szög[°]:\t%.4f\n', q4);
    fprintf('Csukló csavarás (q5) szög[°]:\t%.4f\n', q5);
    %-------------------------------------------------------------------
    %   TCP pozícionálás hibája [mm]
    errorx = error(1)*cos(q1);
    errory = error(1)*sin(q1);
    fprintf('TCP pozíció X hibája [mm]:\t\t%.4f\n', errorx);
    fprintf('TCP pozíció Y hibája [mm]:\t\t%.4f\n', errory);
    fprintf('TCP pozíció Z hibája [mm]:\t\t%.4f\n\n', error(2));
    %-------------------------------------------------------------------
else
    disp("A pozícionálás nem hajtható végre!");
end
```