



MÁRTON LŐRINC – FEHÉR ÁRON

ROBOTIRÁNYÍTÁSOK

LABORATÓRIUMI ÚTMUTATÓ

MÁRTON LŐRINC – FEHÉR ÁRON

ROBOTIRÁNYÍTÁSOK
LABORATÓRIUMI ÚTMUTATÓ



SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
VILLAMOSMÉRNÖKI TANSZÉK

MÁRTON LŐRINC – FEHÉR ÁRON

***ROBOTIRÁNYÍTÁSOK
LABORATÓRIUMI ÚTMUTATÓ***

| Scientia Kiadó |
| Kolozsvár · 2024 |



BETHLEN GÁBOR
Alap

Felelős kiadó:

Sorbán Angella

Lektor:

Gincsainé Szádeczky-Kardoss Emese (Budapest)

Borítóterv:

Tipotéka Kft.

Kiadói koordinátor:

Szabó Beáta

A szakmai felelősséget teljes mértékben a szerzők vállalják.

Első magyar nyelvű kiadás: 2024

©Scientia, 2024

Minden jog fenntartva, beleértve a sokszorosítás, a nyilvános előadás, a rádió- és televízióadás, valamint a fordítás jogát, az egyes fejezeteket illetően is.

Descrierea CIP a Bibliotecii Naționale a României

MÁRTON, LŐRINC

Robotirányítások: laboratóriumi útmutató / Márton Lőrinc, Fehér

Áron. – Cluj-Napoca: Scientia, 2024

Conține bibliografie.

ISBN 978-606-975-089-6

I. Fehér, Áron

004

62

TARTALOM

1. Bevezető	11
2. Robotokban alkalmazott beavatkozók – Egyenáramú motor	35
3. Szögelfordulás- és szögsebességmérés robotcsuklóban – Inkrementális adó	47
4. Szögsebesség-szabályozás	56
5. Szögelfordulás-szabályozás	63
6. Pályakövetést megvalósító szögelfordulás-szabályozás	73
7. Robotmanipulátorok direkt geometriai feladata	84
8. Robotmanipulátorok inverz geometriai feladata	93
9. Robotmanipulátorok kinematikája	106
10. Robotok programozása	117
11. Pályatervezés robotmanipulátoroknak	131
12. Robotmanipulátorok PID szabályozása	142
13. Kiszámított nyomatékok módszere	156
Függelék	167
Szakirodalom	169

Rezumat	171
Abstract	173
A szerzőkről	175

CUPRINS

1. Introducere – Noțiuni de bază de robotică	11
2. Elemente de execuție aplicate în roboți – Motorul de curent continuu	35
3. Măsurarea deplasării și vitezei unghiulare în articulațiile roboților – Traductorul incremental de rotație	47
4. Reglarea vitezei unghiulare	56
5. Reglarea poziției unghiulare	63
6. Reglarea poziției unghiulare pentru urmărirea traiectoriei	73
7. Problema de geometrie directă a manipuletoarelor robotice	84
8. Problema de geometrie inversă a manipuletoarelor robotice	93
9. Cinematica manipuletoarelor robotice	106
10. Programarea roboților	117
11. Programarea traiectoriei roboților	131
12. Controlul PID al roboților	142
13. Metoda momentului calculat	156

Anexe	167
Bibliografie	169
Rezumat	171
Abstract	173
Despre autori	175

CONTENTS

1. Introduction – Basic notions in robotics	11
2. Robot actuators – Direct current motor	35
3. Angular position and angular velocity measurement in robot joints – Incremental encoder	47
4. Angular velocity control	56
5. Angular position control	63
6. Angular position control for trajectory tracking control	73
7. Direct geometric problem of robot manipulators	84
8. Inverse geometric problem of robot manipulators	93
9. Kinematics of robot manipulators	106
10. Robot programming	117
11. Path planning for robot manipulator	131
12. PID robot control	142
13. Computed Torque Method	156
Annexes	167
References	169

Rezumat	171
Abstract	173
About the authors	175

BEVEZETŐ

1.1. A fejezet célja

A fejezet célja a robotirányítások során gyakran használt robotikai alapfogalmak bevezetése, a robotirányítási rendszerek általános bemutatása, a robotgeometriai feladatok megoldásánál alkalmazott geometriai fogalmak bemutatása.

1.2. Robotikai alapfogalmak

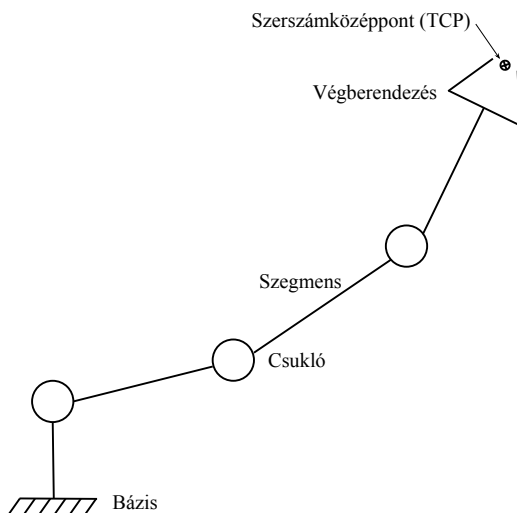
A *robot* egy irányított mechanikai rendszer, amely az alábbi tulajdonságokkal rendelkezik: előre megadott síkbeli vagy térbeli pályán képes mozogni, a pálya mentén előírt feladatot képes elvégezni, újraprogramozható (a feladat megváltoztatható).

Számos robotikai feladat esetében egy céltárgy térbeli *pozícióját* és *orientációját* kívánjuk módosítani (mozgatás). Megkülönböztetünk manipulálási és szállítási feladatokat. Manipulálás esetén a megtett út hossza összemérhető a robot méretével. Szállítási feladatok esetén a megtett út jelentősen nagyobb a robot méreténél. A manipulálási feladatokat tipikusan a környezetéhez rögzített bázisú robotmanipulátorokkal végezzük el, a szállítási feladatokhoz mobilis robotokat alkalmazunk.

A *robotmanipulátorok* felépítésében két részt különíthetünk el, lásd 1.1. ábra:

- *Végberendezés (end effector)*: a céltárgy mozgatására vagy módosítására szolgáló eszköz. A végberendezés a robotmanipulátor azon része, ahol kapcsolatba kerülhet a robot a környezetével. Lehet például egy megfogó vagy megmunkáló eszköz.
- *Artikulált mechanikai struktúra*: a végberendezés térbeli pozícióját és orientációját módosítja előírt sebességgel (és gyorsulással) a robot által elvégzendő feladatnak megfelelően.

A *robot bázisa (base)* az a platform, ahova az artikulált mechanikai struktúra alapja rögzítve van a környezetéhez. A robotirányításhoz szükséges számításoknál a bázist egy térbeli pontnak tekintjük.



1.1. ábra. Robotmanipulátor felépítése

Az *artikulált mechanizmus* szegmensek és robotcsuklók sorozatából áll.

A *szegmenseket (link)* merev testeknek tekintjük, amelyeket csuklók kötnek össze. A szegmensek mérete, alakja, száma nagymértékben meghatározzák a robot munkaterét.

A *csuklók (joint)* képezik a robotmanipulátor mozgó részeit, megvalósítják a szegmensek mozgását. Több csuklótípust különböztetünk meg. A primitív csuklócsalád csak egy tengely mentén/körül képes mozogni. Az összetett csuklók több tengely körül képes elmozdulni (például botkormány).

A primitív csuklókat is két osztályba sorolhatjuk, lásd 1.2. ábra:

1. A *rotációs csuklók* elfordulást valósítanak meg. Legelterjedtebb típusai:

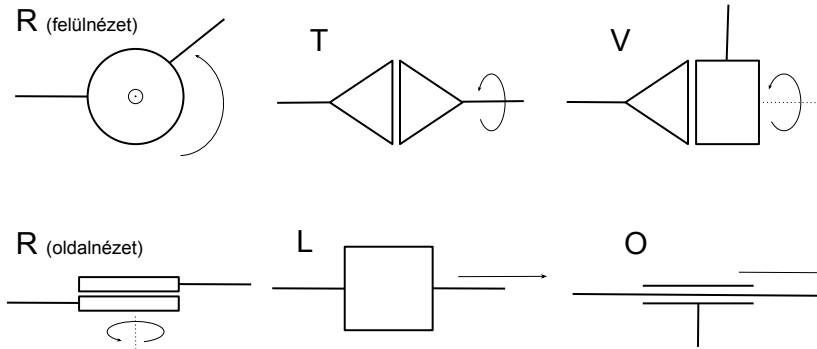
- *R. Forgó csuklók (Rotating joint)*: a forgási tengely merőleges az előző szegmensre. A csukló előtti és utáni szegmens sok esetben párhuzamos.

- *T. Csavarodó csuklók (Twisting joint)*: a forgási tengely egybeesik az előző szegmenssel. A csukló előtti és utáni szegmensek általában egymás meghosszabbításában vannak.
- *V. Keringő csuklók (Revolving joint)*: a forgási tengely egybeesik az előző szegmenssel. A csukló előtti és utáni szegmensek sok esetben egymásra merőlegesek.

2. A *transzlációs (prizmatikus) csuklók* elmozdulást valósítanak meg.

Legelterjedtebb típusai:

- *L. Lineáris csuklók*: az elmozdulási tengely párhuzamos az előző szegmenssel. A csukló előtti és utáni szegmensek párhuzamosak vagy egymás meghosszabbításában vannak.
- *O. Ortogonális csuklók*: az elmozdulási tengely merőleges az előző szegmensre. A csukló előtti és utáni szegmensek egymásra merőlegesek.



1.2. ábra. Robotcsukló típusok

Csuklótér (Joint space): az a tér, amelyben leírjuk a robot csuklóinak összes lehetséges pozícióját (konfigurációját). A robotcsuklók adott pozícióját $\mathbf{q} \in \mathbb{R}^n$ vektorban tároljuk. A $\mathbf{q}$ elemei a *csuklótér*. A vektor tartalmazhat szögeket (rotációs csuklók esetén) és távolságokat (transzlációs csuklók esetén), amelyeket egy adott referenciapozícióhoz képest mérünk. Egy rögzített $\mathbf{q}$ érték a robot egy konfigurációját adja.

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \\ \dots \\ q_n \end{pmatrix} \quad (1.1)$$

q_i az i -edik csuklópozíció.

A csuklóváltozók terének dimenziója (n) a robot szabadságfoka (DOF – *Degrees of Freedom*). Amennyiben a robotmanipulátor csak primitív csuklókat tartalmaz, a csuklók száma megegyezik a robot szabadságfokával ($n = DOF$). Összetett csuklókat tartalmazó robotmanipulátorok esetén a szabadságfok nagyobb a csuklók számánál.

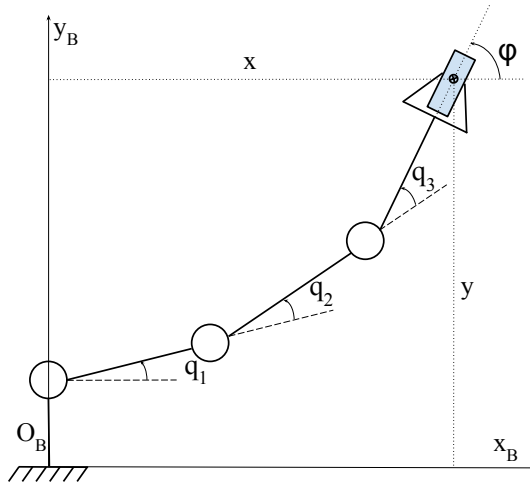
Munkatér (Task space): Az a tér, amelyben leírjuk a végberendezés összes lehetséges pozícióját ($\mathbf{x} \in \mathbb{R}^m$). A végberendezés térbeli pozíciójának megadásához kiválasztjuk a végberendezés egy jellegzetes pontját, a *szerszámközéppontot* (*TCP – Tool Center Point*). Például manipulálási feladatok esetében ez lehet az a pont, ahol a megfogó összeshár a céltárgyon. A szerszámközéppont térbeli pozícióját három koordinátával jellemezhetjük. A végberendezés orientációját szintén három paraméterrel (szöggel) jellemezhetjük. Megmondhatjuk például, hogy egy derékszögű referencia koordináta-rendszer mind a három tengelyéhez képest a céltárgyhoz rögzített koordináta-rendszer mennyire van elforgatva. Ennek megfelelően térben a munkatér maximális dimenziója hat ($m \leq 6$), három pozíció koordinátát és három szöget tartalmazhat. A végberendezés konfigurációját (térbeli pozícióját és orientációját) leíró vektor:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ Rx \\ Ry \\ Rz \end{pmatrix} \quad (1.2)$$

x, y, z a három pozíciókoordináta, Rx, Ry, Rz az orientációt definiáló három szög.

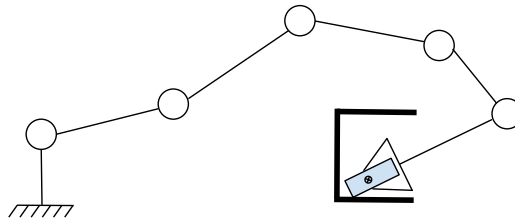
A munkatér dimenziója kisebb vagy egyenlő a robot szabadságfokával, a munkatérben egy merev szegmensekkel rendelkező robot nem tud több tengely mentén/körül irányított mozgást generálni, mint ahány tengely mentén a csuklók el tudnak mozdulni.

Példa: Az 1.3. ábrán látható robot az $x_B O_B y_B$ síkban mozog. A robot három rotációs csuklóval rendelkezik, amelyek erre a síkra merőleges tengelyek körül képesek elfordulni. Ennek megfelelően a robot szabadságfoka 3, a robot konfigurációja 3 szöget tartalmaz: $\mathbf{q} = (q_1 \ q_2 \ q_3)^\top$. A robot végberendezésének síkbeli pozícióját két koordinátával jellemezhetjük. A végberendezés orientációját (a végberendezés síkbeli elfordulását) az utolsó szegmens és az $O_B x_B$ tengely által bezárt szöggel jellemezhetjük. Ennek megfelelően a munkatér dimenziója is 3: $\mathbf{x} = (x \ y \ \varphi)^\top$.



1.3. ábra. Csuklókoordináták és világkoordináták síkban

Abban az esetben, amikor a munkatérben akadályra számíthatunk, és a robot végberendezése be kell hatoljon az akadályok mögé a feladat elvégzéséhez, célszerű olyan robotokat alkalmazni, amelyeknek a szabadságfoka nagyobb, mint a munkatér dimenziója. Ezeket a robotokat *redundáns robotoknak* nevezzük. Redundáns robotok esetében $DOF > m$, vagyis a csuklózóterének a dimenziója nagyobb, mint a munkatér dimenziója. Az 1.4. ábrán egy síkban (a munkatér dimenziója 3) dolgozó redundáns robotot látunk, amelynek a szabadságfoka 5.

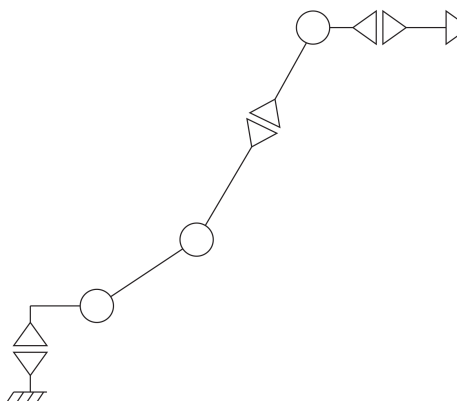


1.4. ábra. Redundáns robotarchitektúra síkban

Robotarchitektúrák

A soros robotmanipulátorokban rotációs és translációs csuklók sorozata követi egymást. A robot szabadságfokát, felépítését, mechanikai méretezését mindig a robotok által elvégzendő feladatosztályok szabják meg.

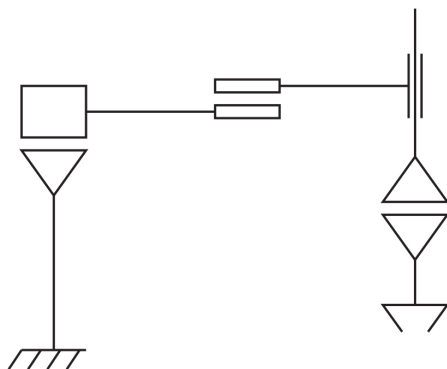
A legelterjedtebb iparban alkalmazott architektúra nagyrészt a klasszikus *PUMA* (*Programable Universal Machine for Assembly / Programable Universal Manipulation Arm*) robot architektúráját követi. Ennek megfelelően az első három szabadságfokhoz tartozó szegmensek tipikusan nagyobb méretűek a minél távolabbi térbeli pozíció eléréséhez. Az utolsó három szabadságfok felelős a végberendezés forgatásáért, a végberendezés orientációjának beállításáért. Ilyen típusú architektúráknál a robot első három szabadságfokához tartozó részét *robotkarnak* (*robot arm*), az orientációt megvalósító három szabadságfokhoz tartozó részt *kézcsuklónak* (*robot wrist*) nevezzük. A legelterjedtebb robotmanipulátorokban mind a hat szabadságfokot rotációs csuklókkal valósítjuk meg, lásd 1.5. ábra.



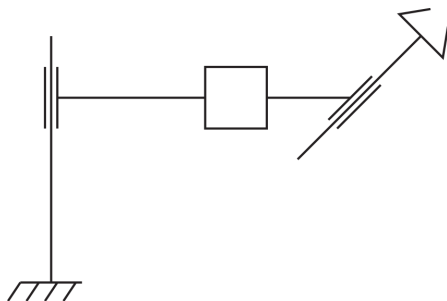
1.5. ábra. Elterjedt ipari robotarchitektúra

A *SCARA* (*Selective Compliance Assembly Robot Arm*) közismert ipari robotarchitektúra, amelyet vízszintes felületeken végzendő feladatokhoz alkalmazhatunk. A vízszintes felület fölött két rotációs csuklóval biztosítja a pozicionálást. A felületre merőlegesen elfordulást és elmozdulást is biztosít, lásd 1.6. ábra.

Hasonló feladatokra alkalmazhatunk Descartes-féle robotokat, amelyek mindhárom térbeli tengely mentén az elmozdulást translációs csuklókkal oldják meg, lásd 1.7. ábra.



1.6. ábra. SCARA robotarchitektúra



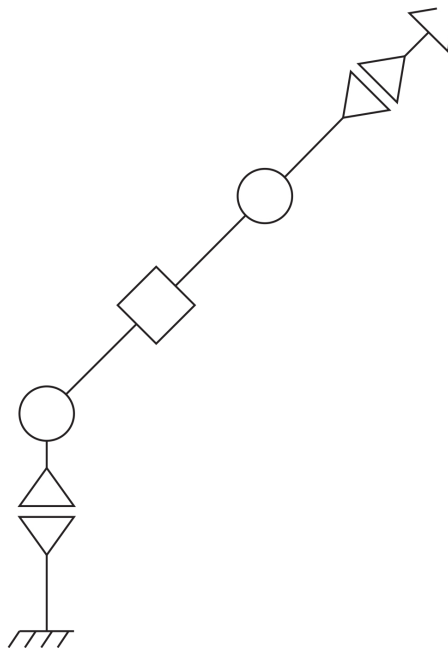
1.7. ábra. Descartes-féle robotarchitektúra

Az első ipari robot (*Unimate robot*) architektúrája az 1.8. ábrán látható. A pozicionálás része a gömbkoordináták elvét követi, két rotációs csuklóval, egymásra merőleges tengelyekkel és egy transzlációs csuklóval (kinyúlás).

Ipari robotok néhány alkalmazási területe

A robotirányítás szempontjából fontos osztályozási kritérium, hogy a robot a feladat elvégzése közben milyen módon kerül kapcsolatba a környezetével.

Kontaktus hiánya vagy minimális mértékű kontaktus: a végberendezés tömege, a végberendezésre ható erők nem vagy elhanyagolható mértékben változnak. Ilyenek például a felügyeleti, tesztelési feladatok (a robot végberendezése pl. egy kamera), hegesztési feladatok, ragasztócsík felhelyezése.



1.8. ábra. *Unimate-szerű robotarchitektúra*

Anyagmozgatás: a munka során a robot végberendezésének tömege, inerciája szakaszosan, jelentős mértékben változhat. Ilyen feladat például céltárgyak áthelyezése futószalagról palettákra (palettázás), ipari folyamatok kiszolgálása (céltárgyak behelyezése berendezésekbe, kivétele berendezésekből).

Megmunkálás: a feladat során a robot végberendezésére jelentős környezeti erők/nyomatékok hatnak. Ilyen feladatok például fúrás, csiszolás, vágás, csavarozás.

A robotok jellemzői

Az ipari robotok jellemzői szabványokban vannak rögzítve. Ezen jellemzők alapján a felhasználó meg tudja választani az alkalmazásához szükséges robottípust. Néhány fontos, az irányításhoz kapcsolódó jellemző:

1. *Valós munkatér (workspace):* Az a tér, amelyben a robot végberendezése mozoghat. A munkatér mérete függ a robot szabadságfokától, a robot

alakjától, méretétől, lásd például 1.9. ábra. Ugyanakkor függ a csuklózváltozók határértékeitől; valós robotoknál a csuklózváltozók egy minimum és maximum érték között változhatnak ($q_i \in [q_{imin} \dots q_{imax}]$).

A munkatérhez köthető a *maximális kinyúlás* (*maximum reach*) fogalma is, ami a robot bázisától a legnagyobb távolság, ameddig a végberendezés még elér.

2. *Hasznos tömeg* (*payload*): a robot végberendezése által mozgatott céltárgy vagy szerszám maximális megengedett tömege.

3. *Felbontás* (*resolution*): a legkisebb távolság vagy szög, amit a robot végberendezése (vagy egy csukló) képes megtenni.

4. *Pozicionálási pontosság* (*position accuracy*): a robot több kiindulópontból megközelít egy előírt célpozíciót. A mozgás megvalósításához az összes csukló kell mozogjon. A pozicionálási pontosságot úgy definiáljuk, mint az átlagos megérkezési pozíció és a célpozíció közötti távolság.

5. *Pozíció ismételhetősége* (*position repeatability*): a robot többször elvégez egy előírt mozgást állandó környezeti tényezők mellett. A mozgás megvalósításához az összes csukló kell mozogjon. A pozíció ismételhetősége azon legkisebb kör sugara, amely bezárja az összes megérkezési pozíciót.

Ipari robotok esetében az ismételhetőség és pontosság tipikusan 0.1–0.01 mm közötti tartományban van.

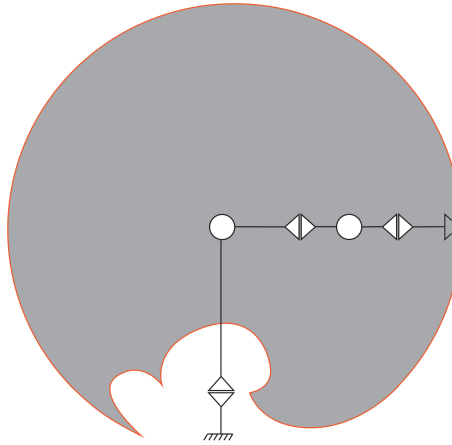
Az 1.10. ábrán ‘ $\otimes$ ’ a célpozíciót, a ‘ $\bullet$ ’ a különböző megérkezési pozíciókat jelölik. Az I. alábrán rossz pontosságot, de jó ismételhetőséget láthatunk. A II. alábrán a pontosság jó (viszonylag egyenletes szórás a célpozíció körül), de az ismételhetőség értéke alacsony.

Bizonyos robotok esetében megadhatjuk a pontosságot és az ismételhetőséget lineáris pálya mentén történő mozgás esetében is (*linear path accuracy*, *linear path repeatability*).

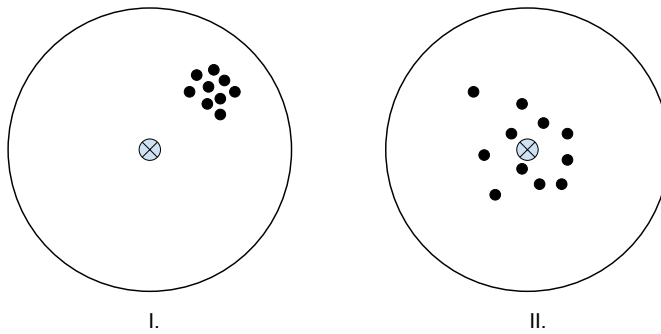
Ugyancsak fontos jellemző még a csuklók által elérhető maximális sebesség.

A robotirányítási rendszer

A robotirányítási rendszer magába foglalja a robotot, a robot irányítását végrehajtó szoftvert (lásd 1.11. ábra), a számítástechnikai eszközt, amin az irányítási szoftver fut (irányító számítógép), valamint az irányító számítógépet és a robotot összekapcsoló hardvert. Ez utóbbi felelős a robot beavatkozáinak meghajtásáért, valamint a robot érzékelői jeleinek előfeldolgozásáért.



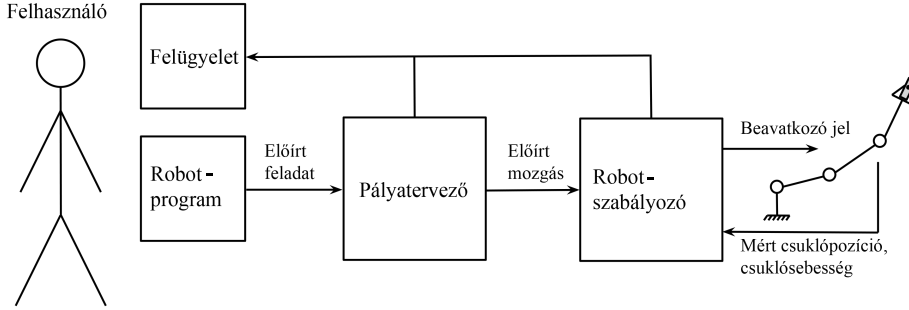
1.9. ábra. 3D robot valós munkaterének egy szimmetria síkjával képzett metszete



1.10. ábra. Pontosság vs. ismételhetőség

A robotot irányító szoftver tipikusan tartalmaz egy kommunikációs interfészt a felhasználó felé. A felhasználó legtöbbször egy robotprogramozási nyelv (RPL – Robot Programming Language) segítségével írja le a robotnak a feladatot. A robotirányító szoftver értelmezi a leírt utasításokat és meghatározza a robot előírt mozgását (*pályatervezés*). Az előírt mozgás és a mért csuklópozíciók, csuklósebességek alapján ugyancsak a robotirányító szoftver határozza meg a robot beavatkozóinak a vezérlőjeleit (*robotszabályozás*).

Az elvégzendő feladatnak megfelelő robotprogram megírása tipikusan off-line módon történik, vagyis a feladat-végrehajtás előtt. Az alacsony szintű irányítás on-line módon fut, vagyis a feladat-végrehajtás közben.



1.11. ábra. Robotirányítási rendszer

Pozíció és orientáció jellemzése térben

Egy test térbeli konfigurációjának (pozíció és orientáció) leírására koordinátákat és szögeket alkalmazunk, amelyeket derékszögű koordináta-rendszerekben (keretekben) adunk meg.

A koordináta-rendszerek definiálásához a jobbkéz-szabályt alkalmazzuk: a jobb kezünk első három ujjá a hüvelykujjtól kezdődően jelenti az x , y illetve a z tengelyeket.

Egy tengely körüli pozitív forgásirányt szintén a jobbkéz-szabály segítségével határozzuk meg: ha a jobb hüvelykujj jelöli az irányított tengelyt, amely körül a forgás történik, a behajlított többi ujj iránya a pozitív forgásirányt adja. Tehát a pozitív forgásirány az óramutató járásával ellentétes.

Pozíció jellemzése térben

Egy térbeli pont P pozíciója egy, a K koordináta-rendszer origójából kiinduló háromkomponensű vektorral jellemezhető.

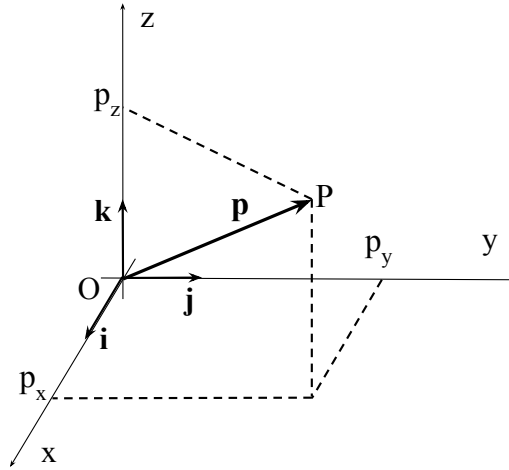
Legyenek a koordináta-rendszer x , y és z tengelyeihez rendelt egységvektorok:

$$\mathbf{i} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{j} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (1.3)$$

A P ponthoz tartozó vektor, lásd 1.12. ábra:

$$\mathbf{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = p_x \mathbf{i} + p_y \mathbf{j} + p_z \mathbf{k} \quad (1.4)$$

p_x, p_y, p_z skalárok, a pont *koordinátái*.



1.12. ábra. Pozíció jellemzése térben

Orientáció jellemzése síkban. Elemi forgatások

Először a síkbeli forgatásokat tárgyaljuk.

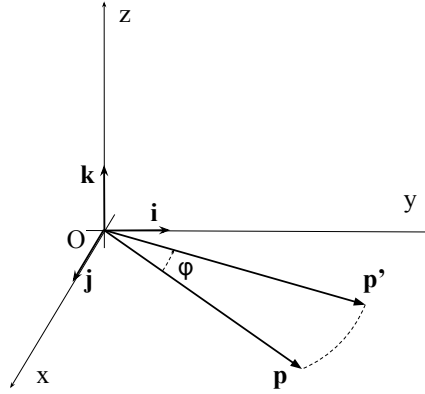
Elemi forgatás a z tengely körül: Legyen egy, a koordináta-rendszer origójából kiinduló vektor az xOy síkban: $\mathbf{p} = (p_x \ p_y \ 0)^\top$. A vektor kiindulópontja rögzített az origóhoz, hossza nem változik.

$$\mathbf{p} = p_x \mathbf{i} + p_y \mathbf{j} + 0 \mathbf{k} \quad (1.5)$$

Forgassuk el a vektort φ szöggel, lásd 1.13. ábra.

Jelölje $\mathbf{p}' = (p'_x \ p'_y \ 0)^\top$ az elforgatott vektort.

$$\mathbf{p}' = p'_x \mathbf{i} + p'_y \mathbf{j} + 0 \mathbf{k} \quad (1.6)$$

1.13. ábra. Forgatás z tengely körül

Keressük az elforgatott vektor $\mathbf{p}'$ és az eredeti vektor $\mathbf{p}$ koordinátái közötti összefüggést.

A feladat megoldásához felvesszünk egy K' koordináta-rendszert, ez először egybeesik az eredeti koordináta-rendszerrel. A forgatásánál rögzítve van a vektorhoz, és vele forog a z tengely körül.

Vegyük észre, hogy a forgatás után a K' koordináta-rendszerben a $\mathbf{p}'$ vektor koordinátái ugyanazok maradnak, mint a K koordináta-rendszerben a $\mathbf{p}$ koordinátái, lásd 1.14. ábra. Tehát:

$$\mathbf{p}' = p_x \mathbf{i}' + p_y \mathbf{j}' + 0 \mathbf{k}' \quad (1.7)$$

A fenti egyenlet felírható az alábbi formában:

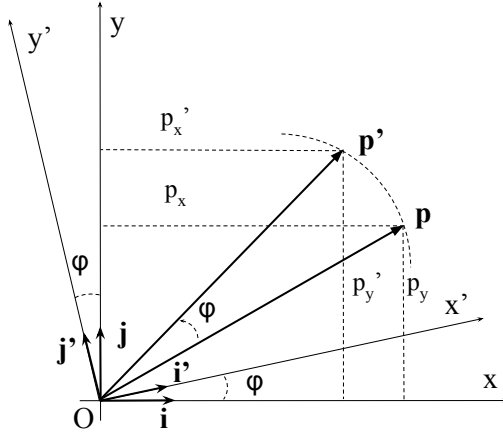
$$\mathbf{p}' = R_z \mathbf{p} \quad (1.8)$$

ahol az R_z 3×3 dimenziójú valós elemű mátrix oszlopai megegyeznek az $\mathbf{i}'$, $\mathbf{j}'$, $\mathbf{k}'$ vektorok koordinátaival az eredeti K koordináta-rendszerben felírva:

$$R_z = (\mathbf{i}' \ \mathbf{j}' \ \mathbf{k}') \quad (1.9)$$

Figyelembe véve, hogy az egységvektorok hossza 1, az 1.14. ábra alapján látszik, hogy:

$$\mathbf{i}' = \begin{pmatrix} \cos(\varphi) \\ \sin(\varphi) \\ 0 \end{pmatrix} \quad \mathbf{j}' = \begin{pmatrix} -\sin(\varphi) \\ \cos(\varphi) \\ 0 \end{pmatrix} \quad \mathbf{k}' = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (1.10)$$



1.14. ábra. Síkbeli forgatás

Tehát

$$R_z = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1.11)$$

Az R_z mátrixot *rotációs mátrixnak* nevezzük, amely a z körüli forgatást írja le.

Példa: legyen a $\mathbf{p}$ vektor: $\mathbf{p} = (1 \ 1 \ 0)^\top$. Forgassuk el $\pi/2$ szöggel. A kapott $\mathbf{p}'$ elforgatott vektor:

$$\mathbf{p}' = \begin{pmatrix} \cos(\pi/2) & -\sin(\pi/2) & 0 \\ \sin(\pi/2) & \cos(\pi/2) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \quad (1.12)$$

Elemi forgatás az x tengely körül: legyen egy, a koordináta-rendszer origójából kiinduló vektor, amelynek az yOz síkban forog. A vektor kiindulópontja rögzített az origóhoz, hossza nem változik. Forgassuk el a vektort φ szöggel. Ugyanolyan megfontolásokkal, mint a z tengely körüli forgatás esetén, az x tengely körüli forgatáshoz rendelt rotációs mátrix:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{pmatrix} \quad (1.13)$$

Elemi forgatás az y tengely körül: Legyen egy, a koordináta-rendszer origójából kiinduló vektor, amelynek az xOz síkban forog. A vektor kiindulópontja rögzített az origóhoz, hossza nem változik. Forgassuk el a vektort φ szöggel. Ugyanolyan megfontolásokkal, mint a z tengely körüli forgatás esetén, az y tengely körüli forgatáshoz rendelt rotációs mátrix:

$$R_y = \begin{pmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{pmatrix} \quad (1.14)$$

A rotációs mátrix: a rotációs mátrix elemei általában az alábbi tulajdonságokkal rendelkeznek:

- Az oszlopokat alkotó vektorok hossza 1.
- Az oszlopokat alkotó vektorok egymásra merőlegesek.

A fenti tulajdonságok miatt igazak az alábbi kijelentések:

- $\det(R) = 1$
- A rotációs mátrix ortogonális mátrix, vagyis: $R^{-1} = R^\top$.
- Két rotációs mátrix szorzata szintén rotációs mátrix: $R = R_1 R_2$.
- A rotációs mátrixok szorzása általában nem kommutatív művelet $R_1 R_2 \neq R_2 R_1$.

Orientáció jellemzése térben

Az orientációt térben Euler-szögekkel jellemezhetjük, amely mindig egy szöghármas. A három szöggel három síkbeli elemi forgatást definiálhatunk. A térbeli elfogatást három szukcesszív elemi elforgatással valósítjuk meg. Belátható, hogy három szöggel egy térbeli tárgy tetszőleges elforgatása leírható.

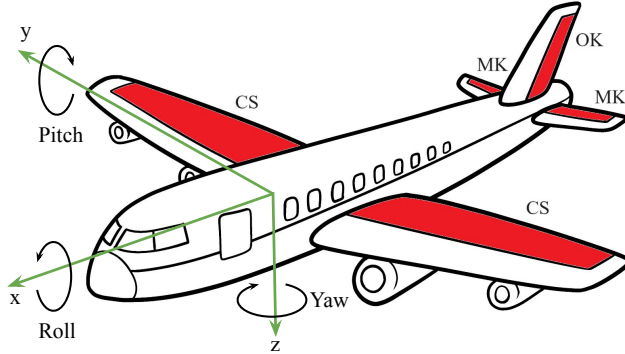
XYZ Euler-szögek: RPY szögek

A Roll – Pitch – Yaw (billenés – bólintás – elfordulás) szögek a térbeli orientáció jellemzésére alkalmas szöghármas. A szukcesszív elemi forgatások sorozata ebben az esetben: forgatás x körül, forgatás y körül, majd forgatás z tengely körül.

A robotika mellett az RPY szögeket alkalmazzák például repülőgép-irányításban is. Az irányváltások esetén a repülőgép kormányfelületei ezt a három szöget módosítják, lásd 1.15. ábra:

- A Roll (billenés, x körüli elfordulás) a csűrőlapok (Cs) ellentétes irányú kimozdításával történik.

- A Pitch (bólintás, y körüli elfordulás) a magassági kormányfelületek (MK) egyirányú kimozdításával történik.
- A Yaw (elfordulás, z körül) az oldal kormányfelület (OK) kimozdításával történik.



1.15. ábra. RPY szögek

Jelölje a z körüli forgatást φ , az y körüli forgatást θ , valamint az x körüli forgatást ψ . Az RPY szögeknek megfelelő rotációs mátrix:

$$R_{RPY} = R_z R_y R_x = \quad (1.15)$$

$$\begin{pmatrix} c_\varphi & -s_\varphi & 0 \\ s_\varphi & c_\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{pmatrix} =$$

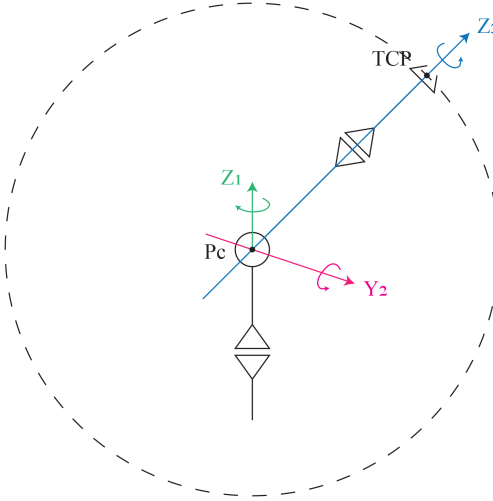
$$\begin{pmatrix} c_\varphi c_\theta & c_\varphi s_\theta s_\psi - s_\varphi c_\psi & c_\varphi s_\theta c_\psi + s_\varphi s_\psi \\ s_\varphi c_\theta & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & s_\varphi s_\theta c_\psi - c_\varphi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{pmatrix},$$

ahol $c_\varphi = \cos(\varphi)$, $s_\varphi = \sin(\varphi)$, $c_\theta = \cos(\theta)$, $s_\theta = \sin(\theta)$, $c_\psi = \cos(\psi)$, $s_\psi = \sin(\psi)$.

ZYZ Euler-szögek: valódi Euler-szögek

ZYZ típusú valódi Euler-szögek esetében az első forgatás z tengely körül történik φ szöggel, az y körüli forgatást θ , az utolsó forgatás szintén a z tengely körül történik ψ szöggel.

Számos ipari robot utolsó három szabadságfoka az Euler-szögek formalizmusa szerint van kialakítva (Euler-csukló, gömbcsukló), lásd 1.16. ábra. Az első és a harmadik csuklók forgástengelyei megegyeznek a szegmensek hossztengelyével (ezek a z tengelyek), a középső forgástengely pedig merőleges a szegmensek által meghatározott síkra (y tengely). Ugyanakkor mindhárom forgástengely egy ponton megy át (P_c).



1.16. ábra. Euler-csukló (gömbcsukló)

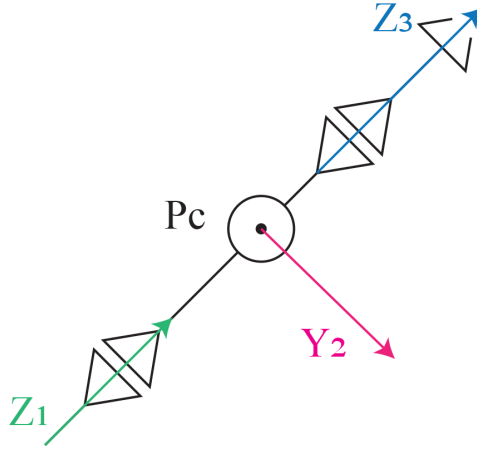
Ezzel a kialakítással, az R_{z1} , R_{y2} elemi forgatásokkal, a végberendezés egy olyan gömb pontjait tudja elérni ideális esetben, amelynek a középpontja P_C , a sugara pedig a $P_C - TCP$ távolság, lásd 1.16. ábra. A végberendezés a gömb sugárirányába fog mutatni. Az utolsó, R_{z3} forgatással figyelembe tudjuk venni a céltárgy alakját, és ennek megfelelően fordíthatjuk el a végberendezést.

A ZYZ Euler-szögeknek megfelelő rotációs mátrix:

$$R_{ZYZ} = R_z R_y R_z = \quad (1.16)$$

$$\begin{pmatrix} c_\varphi & -s_\varphi & 0 \\ s_\varphi & c_\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{pmatrix} \begin{pmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} c_\varphi c_\theta c_\psi - s_\varphi s_\psi & -c_\varphi c_\theta s_\psi - s_\varphi c_\psi & c_\varphi s_\theta \\ s_\varphi c_\theta c_\psi + c_\varphi s_\psi & -s_\varphi c_\theta s_\psi + c_\varphi c_\psi & s_\varphi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{pmatrix},$$



1.17. ábra. Szinguláris konfiguráció

ahol $c_\varphi = \cos(\varphi)$, $s_\varphi = \sin(\varphi)$, $c_\theta = \cos(\theta)$, $s_\theta = \sin(\theta)$, $c_\psi = \cos(\psi)$, $s_\psi = \sin(\psi)$

Inverz orientációs feladat térben: a feladathoz adott egy rotációs mátrixsal definiált elvárt forgatás:

$$R_d = \begin{pmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ i_z & j_z & k_z \end{pmatrix} \quad (1.17)$$

Keressük a (1.17) mátrixsal definiált elforgatásnak megfelelő Euler-szögeket (φ , θ , ψ).

A megoldáshoz egyenlővé tesszük az (1.17) és (1.16) mátrixokat:

$$R_{ZYZ} = R_d \quad (1.18)$$

Látszik, hogy a θ szöget az (1.18) mátrixegyenlet (3,3) elemeinek megfeleltetéséből származó egyenletből tudjuk meghatározni:

$$\cos(\theta) = k_z \quad (1.19)$$

Szoftveres implementálásnál törekszünk az `atan2` függvény alkalmazására (mivel az `atan` függvény értékkészlete a $[-\pi/2, \pi/2]$ intervallum, az

atan2 értékkészlete pedig $(-\pi, \pi]$ (négykvadránsos), tehát az egyenlet megoldása az árkusztangens függvény segítségével:

$$\theta = \pm \text{atan2} \left(\sqrt{1 - k_z^2}, k_z \right) \quad (1.20)$$

Látszik, hogy két megoldás van, általában θ és $\theta + \pi$. Ha $\theta \neq 0$, az (1,3), (2,3) valamint a (3,1), (3,2) mátrixelemek megfeleltetéséből a φ , ψ szögek is számíthatóak:

$$\frac{\sin(\varphi)\sin(\theta)}{\cos(\varphi)\sin(\theta)} = \frac{k_y}{k_x} \implies \varphi = \text{atan2}(k_y, k_x) \quad (1.21)$$

$$\frac{\sin(\psi)\sin(\theta)}{\cos(\psi)\sin(\theta)} = -\frac{j_z}{i_z} \implies \psi = \text{atan2}(j_z, -i_z) \quad (1.22)$$

Példa: legyen az alábbi rotációs mátrix:

$$R_d = \begin{pmatrix} -0.1268 & -0.9268 & 0.3536 \\ 0.7803 & 0.1268 & 0.6124 \\ -0.6124 & 0.3536 & 0.7071 \end{pmatrix} \quad (1.23)$$

Az (1.21), (1.20), (1.22) összefüggéseket alkalmazva, a mátrixnak megfelelő Euler-szögek: $\varphi = \pi/3$, $\theta = \pi/4$, $\psi = \pi/6$.

Szinguláris konfiguráció: vegyük észre, hogy abban az esetben, amikor $\theta = 0$ vagy $\theta = \pi$, az (1.21) és (1.22) egyenletekben az osztás nem végezhető el, mivel a számláló is és a nevező is szorozva van zéróval.

A $\theta = 0$ vagy $\theta = \pi$ esetben a Z_1 és Z_3 tengelyek egybeesnek, a φ és ψ szögek változtatásának ugyanaz a hatása, az Euler-csukló „szabadságfokot veszít”, lásd 1.17. ábra. Ezt a konfigurációt ($\theta = 0$ vagy $\theta = \pi$; $\varphi, \psi \in [-\pi, \pi]$) hívjuk szinguláris konfigurációnak.

Szinguláris konfiguráció ($\cos(\theta) = 1$) esetén az inverz orientációs feladatnál a φ és ψ megoldásához az (1.18) mátrixegyenletben figyelembe vehetjük például az R_{ZYX} mátrix (1,1) és (1,2) elemeit:

$$\frac{\sin(\varphi)\cos(\psi) + \cos(\varphi)\sin(\psi)}{\cos(\varphi)\cos(\psi) - \sin(\varphi)\sin(\psi)} = \frac{i_y}{i_x} \implies \text{tg}(\varphi + \psi) = \frac{i_y}{i_x} \quad (1.24)$$

Szinguláris konfigurációban nem számítható ki φ és ψ külön-külön, csak az összegük. Ebben az esetben robotikai feladatoknál egy lehetséges számítási eljárás az, hogy valamelyik szöget a kettő közül (például ψ) az előző értéken tartjuk ($\psi_{\text{előző}}$) és a másikat ehhez képest számítjuk az (1.24) összefüggés segítségével.

$$\psi = \psi_{\text{előző}} \quad (1.25)$$

$$\varphi = \text{atan2}(i_y, i_x) - \psi_{\text{előző}} \quad (1.26)$$

Pozíció és orientáció együttes jellemzése térben

Legyen adva egy bázis koordináta-rendszer térben (K_O) és egy térbeli tárgy (merev test), amelyhez egy koordináta-rendszert rögzítünk (K_E). Ahhoz, hogy a térbeli tárgy konfigurációját leírjuk a bázis koordináta-rendszerhez képest, szükségünk van:

- Egy pozíció vektorra ($\mathbf{p}_{OE}$), amely a bázis koordináta-rendszerben megadja a testhez rögzített koordináta-rendszer (K_E) origójának vektorát.
- Egy rotációs mátrixra (R_{OE}), amely megadja, hogy a koordináta-rendszerek tengelyei „mennyire vannak elforgatva” egymáshoz képest.

Célszerű lenne a pozíció információt (háromelemű vektor) és az orientáció információt (3×3 -as mátrix) egy struktúrában tárolni. Ennek meghatározásához fogalmazzuk meg a következő feladatot: legyen egy (V) térbeli pont, amelynek ismerjük a pozíció vektorát K_O -ban ($\mathbf{v}_O$), lásd 1.18. ábra. Határozzuk meg a V koordinátáit a K_E -ben.

Jelöljük K_E -ben a pozíció vektort $\mathbf{v}_E$ -vel. Amennyiben a két koordináta-rendszer tengelyei nem lennének elforgatva egymáshoz képest ($R_{OE} = I$), a pozíció vektor egyszerűen lenne számítható vektorösszeadással a következő összefüggésből: $\mathbf{v}_O = \mathbf{p}_{OE} + \mathbf{v}_E$.

Amennyiben a két koordináta-rendszer között elforgatás is van, a vektorok összeadásánál figyelembe kell vennünk, hogy a $\mathbf{v}_E$ koordinátái el vannak forgatva a K_O -hoz képest, a forgatást R_{OE} definiálja. Tehát $\mathbf{v}_E$ az alábbi összefüggésből számítható:

$$\mathbf{v}_O = \mathbf{p}_{OE} + R_{OE} \mathbf{v}_E \quad (1.27)$$

Az fenti egyenlet felírható az alábbi kompakt formában:

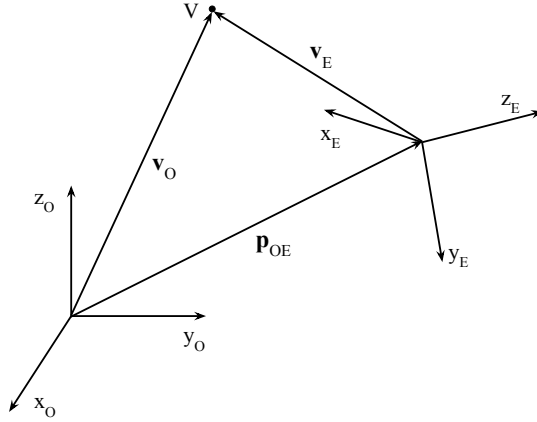
$$\begin{pmatrix} \mathbf{v}_O \\ 1 \end{pmatrix} = \begin{pmatrix} R_{OE} & \mathbf{p}_{OE} \\ \mathbf{0}^\top & 1 \end{pmatrix} \begin{pmatrix} \mathbf{v}_E \\ 1 \end{pmatrix}, \quad (1.28)$$

ahol $\mathbf{0}^\top = (0 \ 0 \ 0)$.

A $\mathbf{v}_h = (x \ y \ z \ 1)^\top$ vektort (normalizált) homogén koordinátavektornak nevezzük. Legyen $\mathbf{v}_{h\alpha} = \alpha \mathbf{v}_h$, $\alpha \neq 0$. Vegyük észre, hogy a $\mathbf{v}_{h\alpha}$ homogén koordinátavektor alapján az eredeti pont koordinátái bármikor visszaszámíthatóak, a vektort el kell osztani az utolsó elemével.

A 4×4 dimenziójú

$$T_{OE} = \begin{pmatrix} R_{OE} & \mathbf{p}_{OE} \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (1.29)$$



1.18. ábra. Koordinátatranszformáció térben

mátrixot a *homogén tranzformáció mátrixának* nevezzük. A mátrix kompakt módon adja meg bármely két koordináta-rendszer térbeli relatív helyzetét.

Amennyiben csak eltolás van (nincs forgatás), a homogén tranzformáció mátrixának alakja:

$$T_{OE} = \begin{pmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (1.30)$$

ahol p_x, p_y, p_z a $\mathbf{p}_{OE}$ elemei.

Amennyiben csak forgatás van (nincs eltolás), a homogén tranzformáció mátrixa:

$$T_{OE} = \begin{pmatrix} i_{Ex} & j_{Ex} & k_{Ex} & 0 \\ i_{Ey} & j_{Ey} & k_{Ey} & 0 \\ i_{Ez} & j_{Ez} & k_{Ez} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (1.31)$$

ahol i_E, j_E, k_E az R_{OE} oszlopai.

A homogén tranzformáció mátrixának tulajdonságai:

- T_{OE} invertálható (mivel a rotációs mátrix invertálható), az inverze is homogén tranzformáció mátrix:

$$T_{OE}^{-1} = \begin{pmatrix} R_{OE}^\top & -R_{OE}^\top \mathbf{p}_{OE} \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (1.32)$$

- Két homogén transzformáció mátrix szorzata is homogén transzformáció mátrix:

$$T_{01}T_{12} = \begin{pmatrix} R_{01}R_{12} & R_{01}\mathbf{p}_{12} + \mathbf{p}_{01} \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (1.33)$$

$T_{02} = T_{01}T_{12}$ a 0-s koordináta-rendszer és a 2-es koordináta-rendszer közötti transzformációt definiálja.

Faktorizáció: bármely transzformáció összetehető egy rotációból és egy translációból:

$$T = \begin{pmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{pmatrix} = \begin{pmatrix} I & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{pmatrix} \begin{pmatrix} R & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{pmatrix}$$

Bázis-kamera-céltárgy transzformáció: Legyen egy robotmanipulátor, amelynek a munkaterében található céltárgy pozícióját és (rotációs mátrixszal definiált) orientációját egy képfelismerő eszköz (kamera) segítségével határozzuk meg. Legyen a robot bázisához rendelt koordináta-rendszer K_B . A kamera térbeli helyzetét a bázishoz képest az R_{BK} rotációs mátrix és $\mathbf{p}_{BK}$ pozícióvektor definiálják. Legyen a céltárgy felismert orientációja és pozíciója a K_G koordináta-rendszerben R_{KG} , illetve $\mathbf{p}_{KG}$. Határozzuk meg a céltárgy pozícióját és orientációját K_B -ben.

A kamera-céltárgy ($K_K - K_G$ koordináta-rendszerek), bázis-kamera ($K_B - K_K$ koordináta-rendszerek) közötti homogén transzformáció mátrixa:

$$T_{KG} = \begin{pmatrix} R_{KG} & \mathbf{p}_{KG} \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad T_{BK} = \begin{pmatrix} R_{BK} & \mathbf{p}_{BK} \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (1.34)$$

A homogén transzformáció a bázis és a céltárgy között:

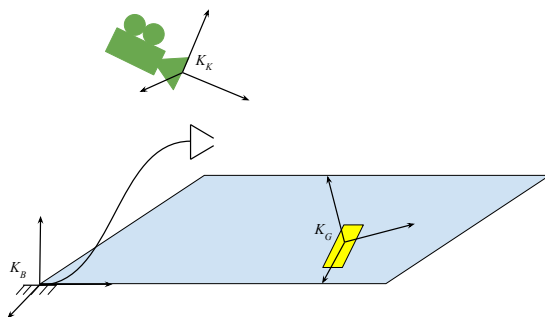
$$T_{BG} = T_{BK}T_{KG} \quad (1.35)$$

Példa: legyen a kamera térbeli helyzete a bázishoz képest: $\mathbf{p}_{BK} = (2 \ 2 \ 3)^\top$, lásd 1.20. ábra. A kamera lefele néz a z tengely mentén, vagyis:

$$R_{BK} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad (1.36)$$

Legyen a kamera által felismert céltárgy pozíció $\mathbf{p}_{KG} = (1 \ 1 \ 3)^\top$ és rotációs mátrix:

$$R_{KG} = \begin{pmatrix} \cos(\pi/4) & -\sin(\pi/4) & 0 \\ \sin(\pi/4) & \cos(\pi/4) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1.37)$$

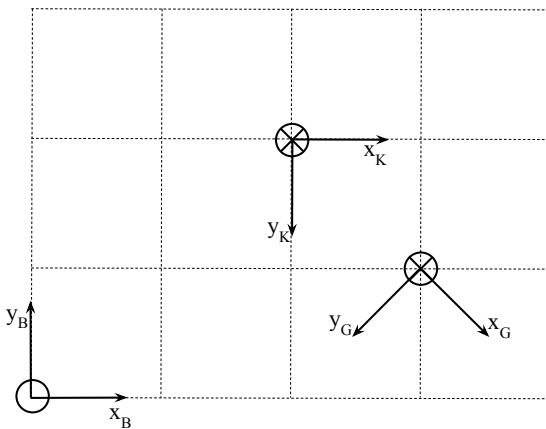


1.19. ábra. Kamera-bázis transzformáció

(z tengely körüli forgatás $\pi/4$ szöggel).

Az (1.35) egyenletet felhasználva, kapjuk:

$$T_{BG} = \begin{pmatrix} \cos(\pi/4) & -\sin(\pi/4) & 0 & 3 \\ -\sin(\pi/4) & -\cos(\pi/4) & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1.38)$$



1.20. ábra. Kamera-bázis transzformáció példa – felülnézet

Kérdések és feladatok

1. Az 1.10. ábra alapján készítsen olyan ábrát, amelyben a pontosság és az ismételhetőség is gyenge, valamint egy olyan ábrát, amelyben a pontosság és az ismételhetőség is jó.
2. Igazolja, hogy az R_x , R_y mátrixokat az (1.13) és az (1.14) összefüggések definiálják.
3. Oldja meg az inverz orientációs feladatot RPY szögek esetében.

ROBOTOKBAN ALKALMAZOTT BEAVATKOZÓK – EGYENÁRAMÚ MOTOR

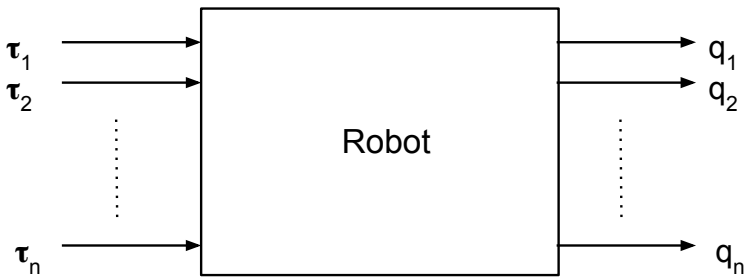
2.1. A gyakorlat célja

A gyakorlat célja a robothajtásokban gyakran használt beavatkozó, az egyenáramú motor vezérléséhez szükséges ismeretek felelevenítése, az egyenáramú motor számítógépes vezérlésének megvalósítása.

2.2. Elméleti bevezető

A robot mint irányított mechanikai rendszer

A robotkar egy többszabadságfokú mechanikai rendszer, amely bemenetének a csuklókra ható erőket vagy nyomatékokat tekinthetjük, kimenetnek pedig a csuklókonfigurációk (pozíciók vagy szögek csuklótípustól függően). Ahhoz, hogy a robotkar előírt mozgást hajtson végre, a csuklókra ható nyomatékokat egy irányítási algoritmus számolja ki, amelyet általában számítógépen valósítunk meg.



2.1. ábra. Robot mint irányított mechanikai rendszer

A 2.1. ábrán egy n primitív csuklót tartalmazó (tehát n szabadságfokú) robot tömbrajza látható. Mindegyik csuklót egy beavatkozó hajt meg. A beavatkozók által kifejtett nyomatékot τ_i , a csuklókonfigurációt pedig q_i jelöli.

Az alábbiakban a csuklókonfigurációt szögpozíciónak ($\alpha = q_i$) tekintjük. Jelöljük az i -ik csukló szögsebességét:

$$\omega = \dot{q}_i \quad (2.1)$$

Az egyenáramú motor vezérlésorientált modellezése

Az egyenáramú motor esetén a kifejtett nyomaték τ_m a nominális működési tartományban a motorárammal (i) tekinthető arányosnak.

$$\tau_m = c_\tau i, \quad (2.2)$$

ahol c_τ a nyomatékállandó.

A motor bemenetének tipikusan a motorra adott feszültséget tekintjük (u). Az összefüggést a motoráram és a motor bemeneti feszültsége között a nominális működési tartományban az alábbi egyenlet adja (lásd [4] 9. fejezet):

$$L \frac{di}{dt} + iR + c_\omega \omega = u \quad (2.3)$$

ahol c_ω a sebességállandó inverze, L, R a rotor induktivitása, illetve ellenállása.

Látszik, hogy állandósult állapotban ($\frac{di}{dt} \approx 0$) a motoráram az alábbi módon számítható:

$$i = \frac{1}{R}u - \frac{c_\omega}{R}\omega \quad (2.4)$$

Tehát állandósult állapotban a bemeneti feszültség és az áram (illetve nyomaték) között is lineáris összefüggés van.

A motor mechanikai viselkedését megadó egyenlet

$$J \frac{d\omega}{dt} = \tau_m - \tau_l, \quad (2.5)$$

ahol J a rotor és terhelés inerciája és τ_l a motor terhelése által kifejtett nyomaték, beleértve a súrlódást is.

Állandósult állapotban ($\frac{d\omega}{dt} \approx 0$) az áram arányos a terheléssel $c_\tau i = \tau_m = \tau_l$. Tehát, a (2.4) összefüggés alapján, a sebesség lineárisan függ a bemeneti feszültségtől:

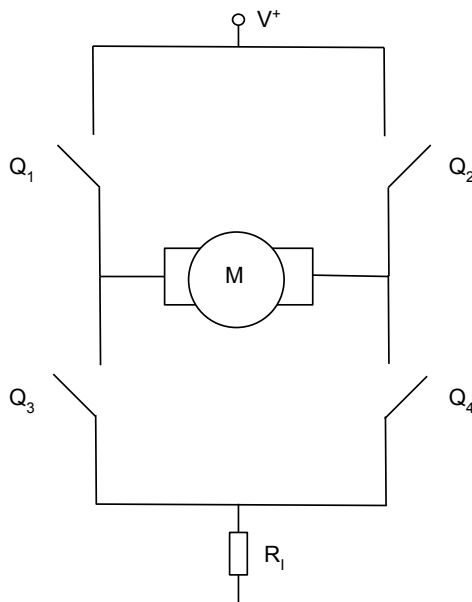
$$\omega = \frac{1}{c_\omega}u - \frac{R}{c_\omega c_\tau}\tau_l \quad (2.6)$$

A fenti összefüggésnek megfelelően, motorvezérlés esetén a motorra adott feszültséget (u) beavatkozó jelnek, a terhelő nyomatékot (τ_l) zavarhatásnak tekinthetjük. A szabályozott kimenet pozíciószabályozás esetén a szögelfordulás: $\alpha = \int_{t_0}^t \omega dt$, $\alpha(t_0) = \alpha_0$.

Az egyenáramú motor vezérlése

Sebességirány-vezérlés

A motor (M) vezérlésére a H-híd alapú bekötést alkalmazhatjuk. A híd elvi rajza a 2.2. ábrán látható. A kapcsolók kialakításánál alkalmazhatunk például megfelelően méretezett tranzisztorokat kiegészítve védődiódákkal (lásd például [2]).



2.2. ábra. *H híd*

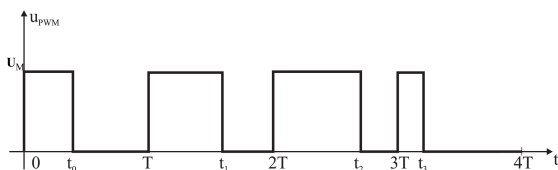
A motor forgásirányát a motoron átfolyó áram irányával módosíthatjuk. Ha az áram iránya megváltozik, a mozgás iránya is megváltozik. A hídban vagy a Q1 és Q4 kapcsolók vannak nyitott állásban és a Q2 és Q3 zárva, vagy a Q2 és Q3 kapcsolók vannak nyitva és a Q1 és Q4 zárva. Ennek megfelelően változtatható a forgás iránya.

Az R_I árammérő-ellenállást vagy árammérő szenzort jelöl. Látszik, hogy amennyiben R_I mérőellenállás, a rajta eső feszültség arányos lesz a motoráram abszolút értékével, forgásiránytól függetlenül.

Sebességérték-vezérlés

A sebességérték-vezérlést a bemeneti feszültség módosításával érjük el, amely tipikusan impulzusszélesség modulált (PWM – Pulse Width Modulation) jel, lásd 2.3. ábra. A jel periódusa (T) konstans, adott a jel maximális értéke (U_M).

A periódusonkénti kitöltési tényezőt (t_i impulzusszélességet) változtathatjuk a vezérlésnél.



2.3. ábra. PWM jel

A periódust tipikusan kis értéknek választjuk (a laborstand esetében 1/20000 s). Minél kisebb a motor elektromos időállandója, T értéke annál kisebb kell legyen. Ha a PWM jel periódusa elégségesen kicsi a motor elektromos időállandójához (R/L) képest, az aluláteresztő szűrőhatás miatt, konstans kitöltés esetén a motor sebessége egyenletes lesz, nem fog változni a PWM jel periódusa alatt. A mintavételezési periódus csökkentésével a zajok is csökkennek, viszont a gyors tranziens jelenségek miatt megnő a rendszer áramigénye.

Határozzuk meg a PWM jel átlagértékét ($\tilde{u}_{PWM}$) egy T periódus alatt. Feltételezzük, hogy a PWM jel szélessége (kitöltése) $0 < t_i < T$.

$$\begin{aligned}\tilde{u}_{PWM} &= \frac{1}{T} \int_0^T u_{PWM} d\tau = \frac{1}{T} \left(\int_0^{t_i} U_M d\tau + \int_{t_i}^T 0 d\tau \right) \\ &= \frac{1}{T} U_M \int_0^{t_i} d\tau = \frac{U_M}{T} t_i \quad (2.7)\end{aligned}$$

Tehát a PWM jel átlagértéke egyenesen arányos a kitöltéssel. A motorvezérlésnél a PWM jel magas állapotának idejét (t_i) fogjuk beavatkozó jelként alkalmazni.

Az egyidejű sebességérték-vezérlés és sebességirány-vezérléshez a H hidat kombináljuk a PWM vezérléssel: a H híd megfelelő kapcsoló párijait (Q1, Q4, valamint Q2, Q3) PWM jellel vezéreljük.

Az áttétel hatása a vezérlésre

Általában a motor illesztése a robotszegmenshez egy áttételen keresztül történik. Az áttétel a robotkarokban a sebességet tipikusan leosztja, az átvitt nyomatékot pedig felszorozza.

Két fogaskerékből álló egyszerű áttétel a 2.4. ábrán látható. Amennyiben a fogaskerek átmérője D a motoroldalon és D_R a robotszegmens oldalon, az áttétel értéke:

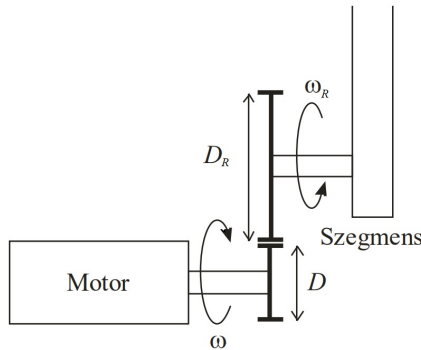
$$N = \frac{D_R}{D} \quad (2.8)$$

Ha a motoroldalon a szögsebesség ω , akkor a robot szegmens oldalon a szögsebesség:

$$\omega_R = \frac{\omega}{N} \quad (2.9)$$

A nyomaték a szegmens oldalon felszorozódik a motoroldalhoz képest. Ugyanakkor figyelembe kell venni az áttételben és a motor felfüggesztésénél fellépő súrlódási hatásokat is. Ha az áttételben fellépő súrlódás miatti veszteségnek megfelelő nyomaték τ_f , akkor a motor oldali nyomaték és a robotszegmens oldali nyomaték közötti összefüggés:

$$\tau_R = N\tau_m - \tau_f \quad (2.10)$$



2.4. ábra. Két fogaskeréssel magvalósított áttétel

2.3. A gyakorlat menete

A *gyakorlat célja*: adatbegyűjtő kártya és számítógépes program felhasználásával vezéreljünk meg egy Maxon A-max 32 24V/15W egyenáramú motort.

1. Ismerkedjünk meg a motorvezérléshez szükséges hardver eszközökkel és az adatbegyűjtő kártya funkcióival.
2. Konfiguráljuk az adatbegyűjtő kártyát a gyártó által biztosított meghajtók segítségével.
3. Egy szöveges dobozba beírt feszültségérték alapján vezéreljük meg a DC motort, figyelembe véve az előjelet és a maximális kiadható feszültség szintet.

A motor analóg vagy modulált digitális teljesítmény jelet igényel, amelyet a számítógép kimenete nem képes előállítani. E célból létrehoztunk egy motorvezérlő áramkört, melynek feladata a kis teljesítményű vezér jelet felerősíteni. A vezérlő áramkört egy National Instruments USB-6001 adatgyűjtő kártya segítségével illesztjük a számítógéphez (Lásd Függelék első ábrája).

Hardver

Motorvezérlő áramkör

A hardver vezérlőáramkörét a Függelék második ábrája szemlélteti. Az ábra jobb alsó sarkán látható az LMD18200 3A/55V-os Texas Instruments által gyártott motorvezérlő H-híd. A vezérlő áramkör zárt állapotú ellenállása $0.33\ \Omega$, kapcsolási ideje $100\ ns$ alatti, a vezér impulzus minimális szélessége $1\ \mu s$. A vezérlő áramkör ugyanakkor egy minimális kapcsolási késleltetést is biztosít a rövidzár elkerülésének érdekében.

Az integrált áramkör képes a motoráramát mérni. A mérési linearitás $\pm 6\%$, ha $1\ A$ és $3\ A$ áramfelvétellel dolgozunk.

A H híd áramkörnek három bemenete van:

- **DIR**: a motor forgásának forgás irányát adja meg.
- **BRK**: a motor bemenetét rövidre zárjuk, létrehozva az aktív fékezést.
- **PWM**: az impulzusszélesség modulált vezérlőjel.

Továbbá, a lapon található egy dsPIC30F2010 16 b-es DSC integrált áramkör, melynek segítségével egy analóg referencia jelből előállítjuk a vezérléshez szükséges impulzusszélesség modulált jelet. A DSC egy engedélyező bemenetet is tartalmaz, melyet ha nullára kötünk, a modulált jel mindig nulla marad. A DSC használható a motor hardverközeli vezérlésére is.

Az áramkör további része az érzékeléshez kapcsolódik, ezért a következő gyakorlat alatt fogjuk tárgyalni.

Számítógép és vezérlő interfész

A motor szabályozását számítógépen fogjuk elvégezni, ezért szükségünk van egy interfészre a vezérlőegység és a számítógép között. Erre a célra a National Instruments által gyártott USB-6001 adatbegyűjtő kártyát alkalmazzuk. A kártya (2.5. ábra) képességei:

- 8 db egyvégpontos vagy 4 db differenciális 14 b felbontással és maximum 20 kps mintavételezési frekvenciával rendelkező analóg bemenet
- 2 db 14 b felbontással (9.1mV pontosság) és maximum 5 kps frissítési frekvenciával rendelkező analóg kimenet
- 13 db digitális ki-, bemenet
- egy 32 b-es programozható számláló
- USB 2.0 FullSpeed számítógépes interfész

A kártyát programozhatjuk C/C++, C#/ .NET, VB/ .NET, LabView, LabWindows/CVI, MeasurementStudio és Python környezetekben.

Szoftver

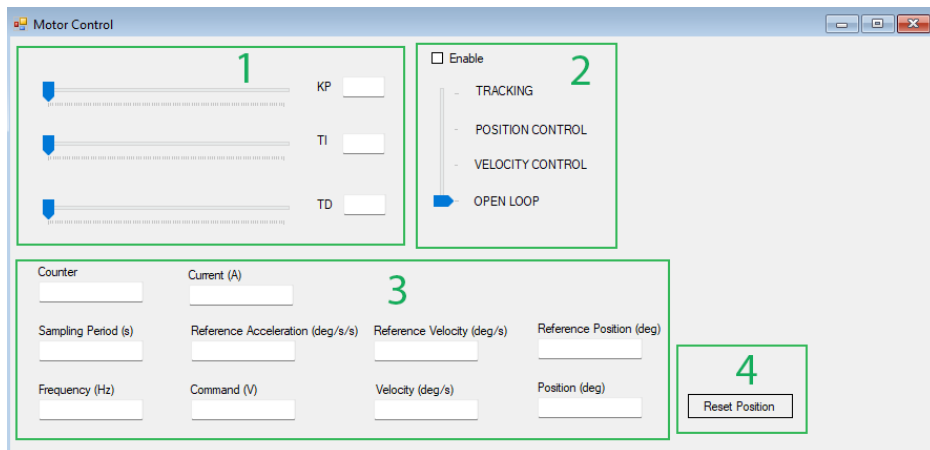
A motorvezérlő alkalmazás tegyen eleget az alábbi feltételeknek:

- A programot C# környezetben fejlesztjük, a .NET alapú kezelőfelület rendelkezésre áll (ez már adott).
- A program indításakor inicializálja a szükséges kártyatulajdonságokat és grafikus ablakelemeket.
- *CheckBox* segítségével engedélyezze a motor működését.
- Ciklikusan kérdezze le egy *EditBox*-ból a vezérlési feszültséget és küldje ki az adatbegyűjtő kártyára (lekezelve az előjelet mint forgásirányt).
- A program bezárásakor a motort állítsa le.



2.5. ábra. *National Instruments USB-6001 adatbegyűjtő kártya. Minden jog fenntartva a National Instruments Corporation számára.*

Az alkalmazás grafikus része és pár segéd metódus már a rendelkezésünkre áll: A felületen elkülönített részekben találjuk a szabályozó paramé-



2.6. ábra. Vezérlő alkalmazás grafikus kezelőfelülete

tereinek beállításáért felelős csúszkákat (1), az üzemmód kiválasztásáért felelős csúszkát és a szabályozást engedélyező és letiltó gombot (2). A szöveges dobozok segítségével vizualizáljuk a folyamaton mért és számolt értékeket (3). A *Reset Position* gomb fogja biztosítani a relatív pozíció nullázását (4).

A vezérlési feladatnál a *Command (V)* szöveges dobozt és az *Enable* gombot kell használni. A szabályozási feladatoknál a *Command (V)* szöveges doboz kimenetként szerepel. A mérési eredmények megjelenítésére használjuk a többi szöveges dobozt (kivéve a referenciákat). A sebesség szabályozási feladatnál a KP és TI csúszkáról olvassuk be a szabályozó paramétereket, a referenciát a *Reference Velocity (deg/s)* szöveges dobozból olvassuk be. A pozíciószabályozási feladatnál a KP és TD csúszkáról olvassuk be a szabályozó paramétereket, a referenciát a *Reference Position (deg)* szöveges dobozból olvassuk be. A relatív pozíció nullázást a *Reset Position* gomb segítségével végezhetjük. Pályakövetési feladat esetén a KP csúszkáról olvassuk be a szabályozó paraméterét, és a referencia szöveges dobozok mind az előállított referenciaértéket fogják megjeleníteni.

Az adatbegyűjtő kártya sikeres eléréséhez szükséges a *NationalInstruments.Common.dll* és *NationalInstruments.DAQmx.dll* használata, mely biztosítja a kártyakezelő függvényeket, az adat dekódolását és enkódolását. *Megjegyzés:* a kártya felismeréséhez installálni kell a gyártó által szolgáltatott szoftvercsomagot: NI-DAQmx (<https://www.ni.com/en/support/downloads/drivers/download.ni-daq-mx.html#484356>). Ezenfelül alkalmazni fogjuk az *MmTimer.dll* állományt, mely folyamatok időzítésére nyújt egy külön szálon futó megoldást.

A driverek használatát a referencia ablakban adjuk meg, a használatukat a *using* kulcsszóval jelezzük:

```
using PreciseTimer;
using NationalInstruments.DAQmx;
```

A .NET-ben használt metódusok:

- *..._Form_Load*: azon metódus, mely a grafikus felület betöltésekor hívódik meg egyszer.
- *..._Timer_Tick*: időzítő metódus, akkor hívódik meg, amikor az időzítő elszámol a megadott periódusig. Lehet egyszeri vagy ciklikus lefutású.
- *..._Form_Closed*: a grafikus kezelőfelület bezárása után közvetlen meghívódó metódus.

A motor vezérléséhez három vezérlőjel logikáját kell leprogramozzuk:

- *Enable*: 1 aktív kimenet a 0-s digitális port 0-s bitje, a motor mozgásának engedélyezéséért felel.
- *Direction*: kimenet a 0-s digitális port 1-es bitje, a motor forgásirányát adja meg (ahol 0 a trigonometriai iránynak felel meg).
- *CommandVoltage*: 0-s analóg kimenet [0, 10]V intervallumban, a motor vezérlő feszültsége.

A kártyának minden egyes funkcionalitása egy különálló feladatként fogható fel, külön task-ok hajtják végre:

```
NationalInstruments.DAQmx.Task AOutTask =
    new NationalInstruments.DAQmx.Task();
NationalInstruments.DAQmx.Task DOutTask =
    new NationalInstruments.DAQmx.Task();

AnalogSingleChannelWriter setAOut;
DigitalSingleChannelWriter setDOut;
```

Az analóg feszültség csatornát az alábbi paranccsal tudjuk beállítani:

```

AOutTask.AOChannels.CreateVoltageChannel(
    string PhysicalChannelName,
    string AssignedName,
    double MinimumValue,
    double MaximumValue,
    AOVoltageUnits Units);

```

A *PhysicalChannelName* az eszközön található analóg kimenetet jelöli (pl. „DevX/aoY”). Az *AssignedName* a kimenet egyedi neve (pl. „AOut”). A *MinimumValue* a mért jel intervallumának minimuma (pl. „0”). A *MaximumValue* a mért jel intervallumának maximuma (pl. „5.0”). A *Units* a kiadandó jel mértékegysége (pl. „AOVoltageUnits.Volts”).

Ahhoz, hogy a *setAOut* osztályon keresztül jelet tudjunk kiadni, példányosítani kell:

```
setAOut = new AnalogSingleChannelWriter(AOutTask.Stream);
```

Ezután egy analóg minta kiküldése egyszerűen az alábbi metódussal végezhető el:

```
setAOut.WriteSingleSample(bool AutoStart, double AOutVal);
```

ahol *AutoStart* megmondja, ha a pufferre tett értéket automatikusan ki akarjuk tenni a kimentere, vagy külön paranccsal oldjuk meg, illetve *AOutVal* a kiadandó analóg érték.

A digitális kimenetet és a kimenet író metódusát az analóg kimenethez hasonlóan állíthatjuk be. Előbb konfiguráljuk a csatornát:

```

DOutTask.DOChannels.CreateChannel(
    string Lines,
    string AssignedName,
    ChannelLineGrouping Grouping);

```

A *Lines* az eszközön található digitális kimenetet jelöli (pl. „DevX/portY”). A *Grouping* a kimeneti bitek csoportosítását adja meg (pl. „ChannelLineGrouping.OneChannelForAllLines”). Egy digitális minta kiküldése egyszerűen az alábbi metódussal végezhető el:

```
setDOut.WriteSingleSamplePort(bool AutoStart, uint data);
```

ahol *data* a teljes digitális vonalra kiadott érték.

Az időzítő modult *mmTimer sTimer* alakban hozzuk létre, és az alábbi négy paramétert kell beállítani:

- *.Mode*: az időzítő viselkedését állítja be. *TimerMode.Periodic* és *TimerMode.Oneshot* választható.
- *.Resolution*: a szoftveres számláló felbontása milliszekundumban.
- *.Period*: az időzítő periódusa milliszekundumban. Ha ez az idő letelt, az időzítő egy új eseményt generál.
- *.Tick*: ide adjuk meg az esemény lekezelésére használt metódust, pl.

```
sTimer.Tick += new EventHandler(mmTimer_Tick);
...
private void mmTimer_Tick(object sender, EventArgs e)
{ ... }
```

Az időzítő függvényben engedélyezzük a motor forgatását, ha a grafikus felületen az *Enable_checkBox* elem ki van jelölve. A grafikus felületen található feladat kiválasztó csúszkáról („*Select_Lab_trackBar*”) olvassuk le az értéket, felhasználva a már létrehozott *getTrackBarVal* metódust. Ha az első állásban van a csúszka, olvassuk be a „*Command_textBox*” dobozban levő értéket, lekezelve a numerikus/alfanumerikus inkompatibilitási problémákat. A beolvasáshoz alkalmazzuk a már létrehozott *getTextboxVal* metódust. Kezeljük le az előjelet mint irány bitet, majd küldjük ki a digitális értékeket és a beolvasott vezérlő jel abszolút értékét mint analóg jelet.

Nagyon fontos, hogy az alkalmazás bezárásakor nullázzuk az analóg kimenetet és tiltsuk le a motor forgását.

2.4. Kérdések és feladatok

1. Milyen modernebb integrált áramkörök léteznek a DC motor H-hidas vezérlésére?
2. Hogyan kellene módosítani a programot ahhoz, hogy a vezérlőfeszültséget egy csúszkáról tudjuk kiküldeni?
3. Hogyan lehet megvezérelni a DC motort abban az esetben, ha csupán digitális kimeneteket használhatunk (szoftveres PWM megvalósítás)?

SZÖGELFORDULÁS- ÉS SZÖGSEBESSÉGMÉRÉS ROBOTCSUKLÓKBAN – INKREMENTÁLIS ADÓ

3.1. A gyakorlat célja

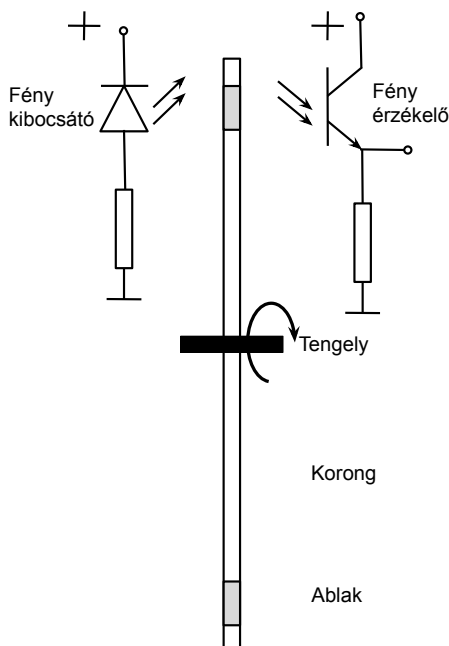
A gyakorlat célja a robotcsuklókban gyakran használt szögelfordulás-érzékelő, az inkrementális adó (enkóder, inkrementális forgójeladó) működési elvének felelevenítése, szögelfordulás- és szögsebességmérés szoftveres megvalósítása.

3.2. Elméleti bevezető

Az inkrementális adó a rotációs robotcsuklók szögelfordulásának mérésére alkalmazható érzékelő. Tipikusan a beavatkozó motor rotorjához rögzítjük, ugyanakkor a mechanikai terhelés és a beavatkozó motor között legtöbb esetben áttétel van, ezért számos esetben érdemes szögelfordulás-érzékelőt a terhelés oldalon is alkalmazni.

Az inkrementális adó mérési elve fényimpulzus számoláson alapszik. A fényimpulzusok az érzékelő tengelyére rögzített, ablakokat tartalmazó korong és egy fotóemitter-fotódetektor pár segítségével jönnek létre, lásd [3.1. ábra](#). A fényimpulzusokat a fotóérzékelő elektromos impulzusokká alakítja. Egy hasonló megoldáshoz jutunk, ha az inkrementális adót több miniatűr vagy egy síkpolarizált mágnessel látjuk el, és a fotoérzékelőket Hall-érzékelőkre cseréljük le. Az érzékelő tengelyét csatoljuk a beavatkozó motor tengelyére.

Az inkrementális adók relatív szögelfordulás- és szögsebességmérésre alkalmas szenzorok. Egységnyi idő alatt a tengelyük elfordulási szögével arányos számú impulzust szolgáltatnak. Ha a csukló sebessége megváltozik, az



3.1. ábra. Inkrementális adó működése

impulzussorozat frekvenciája nő meg. Így az inkrementális adóval megvalósított pozíciómérés impulzusszámlálásra, a sebességmérés frekvenciamérésre vezethető vissza.

Az inkrementális adó fontos paramétere a felbontás (N); ezt a korongon levő ablakok száma adja. Tipikus felbontás értékek $N = 1000 \dots 10000$. A felbontást az elvárt pozíció- vagy sebességszabályozási pontosság alapján lehet megválasztani.

Relatív elmozdulás mérés: A szögelfordulás-mérés az impulzusok számolásán alapszik. Feltételezzük, hogy $\Delta t_k = t_k - t_{k-1}$ mérési időintervallum alatt n_k darab impulzust számoltunk. t_{k-1} a mérés kezdeti időpillanata, t_k időpillanatban számítjuk az elfordulást. Mivel N impulzus 2π elfordulásnak felel meg, az elfordulás mértékét ($\Delta\alpha_k$) radiánban az alábbi módon számíthatjuk:

$$\Delta\alpha_k = |\alpha(t_k) - \alpha(t_{k-1})| = \frac{2\pi n_k}{N} \text{ (rad)} \quad (3.1)$$

Látszik, hogy a szögelfordulás-mérés pontossága $2\pi/N$ (rad).

Az inkrementális adó mindig az előző méréshez viszonyított relatív elfordulásértéket ad. Ahhoz, hogy $\alpha(t_k)$ abszolút szögpozíció értékét megkapjuk, szükséges ismerni $\alpha(t_{k-1})$ kiinduló szögpozíciót és az elfordulás irányát.

Írány meghatározása: a (3.1) összefüggés segítségével csak a szögelfordulás mértéke számítható, az elmozdulás iránya nem. Ahhoz, hogy a forgás irányát meg tudjuk határozni, az inkrementális adók esetén két sor ablak van a korongon, egymáshoz képest egy negyed fázissal (90 fokkal) eltolva (lásd 3.3. ábra). Mindkettő egymástól független fényérzékelővel van felszerelve. Ennek megfelelően két impulzussorozatot (A, B) kapunk.

Példa (lásd 3.2. ábra): az irány meghatározásához azt használhatjuk ki, hogy amikor az inkrementális adó korongja CW (*ClockWise*) irányba forog, a B csatorna pozitív frontján az A csatorna értéke 1. Ha az inkrementális adó korongja CCW (*CounterClockWise*) irányba forog, a B csatorna pozitív frontján az A csatorna értéke 0.

Az irányinformációt az A és B digitális impulzussorozatok alapján egy D tárolóval kaphatjuk meg, lásd 3.2. ábra. Ha a CLK bemeneten pozitív front van, a D tároló a bemenetet kiírja a Q kimenetre és ezt az értéket Q megtartja a következő CLK pozitív frontig. Amennyiben az A impulzussorozatot a D-tároló D (*Data*) bemenetére, a B impulzus sorozatot a D-tároló CLK (órajel bemenetére) csatoljuk, akkor D tároló Q kimenetéről megkapjuk az irányinformációt.

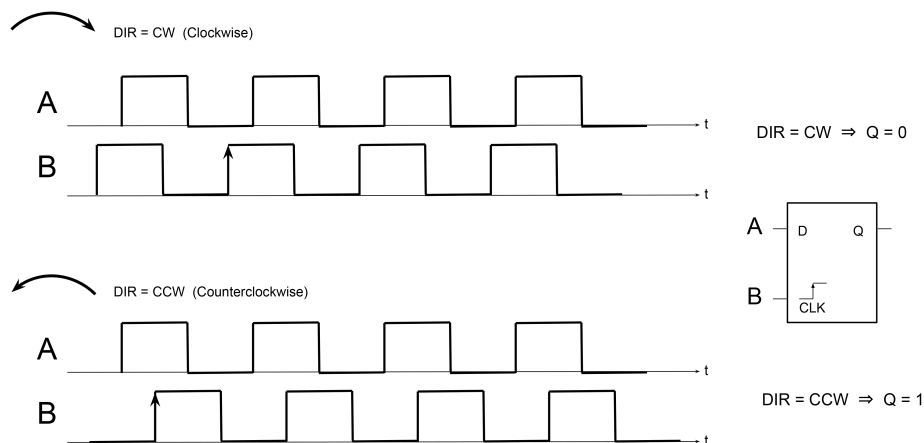
Az A, B csatornák bekötése fordítva is történhet.

Legyen a szögelfordulás irányához rendelt előjel változó:

$$dir = \begin{cases} 1, & \text{ha } Q = 1 \\ -1, & \text{ha } Q = 0 \end{cases} \quad (3.2)$$

Nullázás: ahhoz, hogy inkrementális adóval abszolút szögelfordulást tudjunk meghatározni, az A és B csatornákon érkező impulzussorozatokból kapott információ nem elégséges. Szükség van egy extra jelre, ami megadja a robotcsukló (általában a felhasználó által meghatározott) kezdeti referencia pozícióját (α_0). Amikor a robotcsukló pozíciója megegyezik a kezdeti pozícióval, egy extra szenzor ezt digitális jellel jelzi az inkrementális adó jeleit feldolgozó rendszernek, ez pedig elvégzi az $\alpha(t_0) = \alpha_0$ megfeleltetést. Ezek után a szögelfordulást ehhez a kezdőpozícióhoz képest számoljuk.

Sok inkrementális adó rendelkezik egy Z (zéró, index) kimenettel is, amely az érzékelő tengelyének egy teljes elfordulása során egyszer szolgáltat impulzusjelet egy referencia pozícióban. Sok esetben ezt is használhatjuk nullázáshoz.



3.2. ábra. Irány meghatározása inkrementális adóval

A szögsebességet úgy számíthatjuk, hogy a relatív szögelfordulást osztjuk a Δt_k mérési időintervallummal.

$$\alpha(t_k) = \alpha(t_{k-1}) + dir \cdot \frac{2\pi n_k}{N}, \quad \alpha(t_0) = \alpha_0 \quad (rad) \quad (3.3)$$

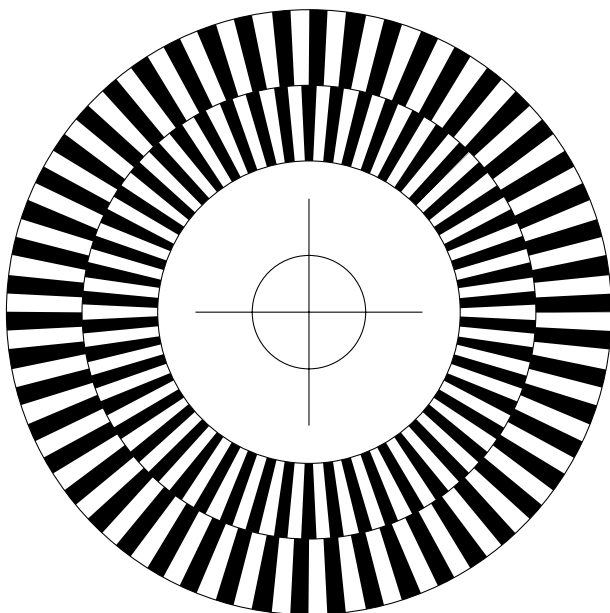
$$\omega(t_k) = dir \cdot \frac{2\pi n_k}{N \Delta t_k} = dir \cdot \frac{2\pi f_k}{N} \quad (rad/sec), \quad (3.4)$$

ahol f_k az impulzussorozat frekvenciája a mérés t_k időpillanatában. Az impulzusszámláláshoz akár az A, akár a B csatornától érkező jeleket alkalmazhatjuk.

Abszolút enkóder: ez a szögelfordulás-érzékelő az inkrementális adók azon hiányosságát kívánja kiküszöbölni, hogy az csak relatív pozícióinformáció mérésére alkalmas. Az abszolút enkóder egy teljes fordulaton abszolút szögelfordulást képes mérni. A korongja bináris vagy Gray-kódolásban tartalmazza a szögelfordulás információt, lásd 3.4. ábra.

3.3. A gyakorlat menete

A gyakorlat célja: adatbegyűjtő kártya és számítógépes program felhasználásával határozzuk meg az egyenáramú motor forgási sebességét és relatív szögelfordulását.



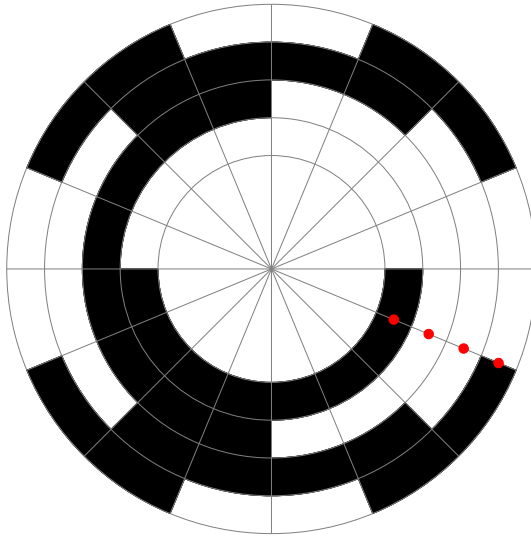
3.3. ábra. *Inkrementális adó korongja*

1. Konfiguráljuk az adatbegyűjtő kártya számláló és digitális bemenetét az inkrementális adóból jövő impulzusok és irány méréséhez.
2. Mérjük meg a beérkező impulzus frekvenciáját, majd számoljuk ki a motor oldali szögsebességet.
3. Egy numerikus integrálási módszer segítségével határozzuk meg a relatív szögelfordulást.

A gyakorlat során az előző gyakorlaton előkészített projekt továbbfejlesztésén fogunk dolgozni. Első lépésként konfiguráljuk az NI USB-6001 kártya számláló bemenetét az inkrementális adó által létrehozott jelek periodikus beolvasására. Második lépésként a beolvasott értékek alapján kiszámoljuk a motor sebességét, illetve a relatív elmozdulását.

Impulzusok beolvasása

Az adatbegyűjtő kártya rendelkezik egy 32 b-es számláló bemenettel. A számláló regiszternek tudunk kezdő értéket adni, meg tudjuk mondani



3.4. ábra. 4 bites abszolút enkóder korongja

a számlálás irányát, illetve a számlálás érzékenységet (felfutó vagy lefutó élek).

```
NationalInstruments.DAQmx.Task CInTask =
    new NationalInstruments.DAQmx.Task ();
CounterSingleChannelReader getCIn;
CInTask.CIChannels.CreateCountEdgesChannel(
    string PhysicalChannelName,
    string AssignedName,
    CICountEdgesActiveEdge edge,
    long initialCount,
    CICountEdgesCountDirection countDirection);
getCIn = new CounterSingleChannelReader(CInTask.Stream);
CInTask.Start();
```

A *PhysicalChannelName* az eszközön található számláló bemenetet jelöli (pl. „DevX/ctrY”). Az *AssignedName* a számláló egyedi neve (pl. „CIn”). Az *edge* a számláló érzékenysége („CICountEdgesActiveEdge.Rising” vagy „CICountEdgesActiveEdge.Falling”). Az *initialCount* a számláló kezdeti értéke (pl. „0”). A *countDirection* a számlálási irány („CICountEdgesCountDirection.Up”, vagy „CICountEdgesCountDirection.Down”). Állítsuk be a kártyát a 0-s számlálóra, felfutó élekre, 0-tól felfele számolva. Az eddigi

kártya feladatokról eltérően a számlálót el kell indítani. Erre való a *CIn-Task.Start()* metódus.

Az impulzusok beolvasását periodikusan végezzük az időzítő esemény metódus elején, mielőtt az elvégzendő feladatra kerülne sor. A számláló regiszter beolvasását az alábbi metódus oldja meg:

```
getCIn.ReadSingleSampleUInt32();
```

Amint a neve is mondja, ezen metódus egy 32 b-es pozitív egész számot fog visszatéríteni. Mivel mi a frekvenciára vagyunk kíváncsiak, két beolvasás között eltelt időnek kell a reciprok értékét számolni (figyelembe véve az esetleges túlsordulást). Fontos, hogy az előző minta tárolását perszisztens vagy globális változóban végezzük. Az alábbi kód egy lehetséges megoldást szemléltet az eltelt idő mérésére:

```
Counter_In_k = getCIn.ReadSingleSampleUInt32();
if (Counter_In_k_1 <= Counter_In_k)
    Delta_Count = Counter_In_k - Counter_In_k_1;
else
    Delta_Count =
        Counter_In_k + (UInt32.MaxValue - Counter_In_k_1);
Counter_In_k_1 = Counter_In_k;
```

Az eltelt időt visszaszámolhatjuk, ismerve az időzítő periódusát és a ciklusszámot, illetve meg is mérhetjük:

```
Delta_t = 0;
DateTime t_k = DateTime.Now;
if (t_k_1 != DateTime.MinValue)\{
    TimeSpan period = t_k - t_k_1;
    Delta_t = period.TotalSeconds;
\}
t_k_1 = t_k;
```

A *Delta_Count* és periódus ismeretében a frekvencia megadható a kettő hányadosaként.

A sebesség méréséhez szükségünk van a forgás irányára, melyet az 1-es digitális port 0-s bitjéről olvasunk be. Az irány jel generálásáért felelős áramkört a Függelék második ábráján szereplő kapcsolási rajz IC3A D tárolója képezi. Az alábbi kódrészlet segít az előjel (forgásirány) meghatározásához:

```
DInTask.DIChannels.CreateChannel(
    "Dev0/port1",
    "DIn",
```

```

ChannelLineGrouping.OneChannelForAllLines);
getDIn = new DigitalSingleChannelReader(DInTask.Stream);

Int32 Dir_Encoder =
    getDIn.ReadSingleSamplePortByte() & 0x01;

```

Sebesség mérése

Az inkrementális adó felbontása $N = 500$, a sebességet a (3.4)-es összefüggéssel számoljuk ki, figyelembe véve az irány bitet (1-es bit jelenti a pozitív irányt).

Pozíció mérése

Folytonos esetben, ha adott a pillanatnyi sebesség, a pozíciót integrálással számoljuk vissza:

$$\alpha(t) = \int_0^t \omega(\eta) d\eta, \quad (3.5)$$

A programunkban viszont diszkrét értékekkel dolgozunk, ezért a húrtrapéz megközelítést fogjuk alkalmazni:

$$\alpha[k] = \sum_{l=1}^k \frac{\omega[l] + \omega[l-1]}{2} \Delta t \quad (3.6)$$

$$\alpha[k] = \alpha[k-1] + \frac{\Delta t}{2} (\omega[k] + \omega[k-1]) \quad (3.7)$$

A mért és számolt értékeket a *setTextboxVal(string name, string value)* metódussal írhatjuk ki a grafikus felületre. A különböző szövegdobozok nevét a grafikus felületről olvashatjuk le szerkesztő üzemmódban. Mivel a mintavételezés 40 ms-körüli, a gyors változásokat nehezen tudjuk követni a képernyőn. A tiszta kiírás érdekében csak minden 10-ik mintát írjuk ki.

A pozíciószabályozási algoritmusok ismételhető tesztelésének érdekében vezessük be a relatív pozíció nullázását. Mivel a labor stand nem tartalmaz végállaskapcsolót, az abszolút nulla pozíciót nem tudjuk megkeresni, viszont az aktuális pozíciót elnevezhetjük nullának. Erre fogjuk használni a *Reset_Position_button* gombot, melynek lenyomásakor nullázzuk a pozíció aktuális és múlt minta értékeit ($\alpha[k] = \alpha[k-1] = 0$).

3.4. Kérdések és feladatok

1. Oldjuk meg a motor áramának mérését, tudva, hogy a vezérlő áram transzformátora $1\text{ A} : 377\text{ }\mu\text{A}$ aránnyal dolgozik, és a vezérlő integrált áramkör CS kimenete az adatbegyűjtő kártya 0-s analóg bemenetére van kötve (Lásd a Függelék első ábrája).
2. Mérjük meg a mintavételezési idő átlagértékét és szórását.
3. Mekkora a sebesség és a pozíció számolás pontossága?

SZÖGSEBESSÉG-SZABÁLYOZÁS

4.1. A gyakorlat célja

A gyakorlat célja egyenáramú motorral megvalósított hajtás sebesség-szabályozása konstans előírt érték esetén, az egyenáramú motor szabályozás-orientált modelljének felelevenítése, a sebességszabályozás tervezése és szoftveres megvalósítása.

4.2. Elméleti bevezető

4.2.1. Egyenáramú motor modellezése sebességszabályozáshoz

Az egyenáramú motor, mint elektromechanikus rendszer, dinamikus viselkedése a mechanika és a villamosságtan törvényeiből származó differenciálegyenletekkel írható le. A mechanikai és villamos egyenletek csatoltak, mivel a nominális működési tartományban a motor által kifejtett nyomaték (τ_m) arányos az i rotorárammal, illetve a rotorba visszaindukált feszültség arányos a szögsebességgel (ω).

Szögsebesség-szabályozás esetén a dinamikus modellből kiolvasható rotorra kapcsolt feszültség (beavatkozó jel – u) és a motor tengelyére ható külső terhelés (ismeretlen zavar – τ_l) hatása a szögsebességre (szabályozott jellemző – ω):

$$\begin{cases} L \frac{di}{dt} + Ri + c_\omega \omega = u \\ J \frac{d\omega}{dt} = c_\tau i + \tau_l \end{cases} \quad (4.1)$$

ahol L a rotor induktivitása, R a rotor ellenállása, c_ω a sebességállandó inverze, J a motortengely és az ismert terhelés inerciája, c_τ pedig a nyomatékállandó.

Feltételezzük, hogy a terhelés magába foglalja a rotor felfüggesztésében, valamint az áttételekben fellépő ismeretlen súrlódási hatásokat is.

Feltételezve, hogy a rotor elektromos időállandója elhanyagolható a szabályozott motor domináns időállandójához képest ($L/R \approx 0$), az áram-értéket ($i = u/R - c_\omega \omega/R$) behelyettesíthetjük a mechanikai egyenletbe:

$$J \frac{d\omega}{dt} + \frac{c_\tau c_\omega}{R} \omega = \frac{c_\tau}{R} u + \tau_l \quad (4.2)$$

Az egyenlet a szögsebesség dinamikáját írja le a bemeneti feszültség és terhelés függvényében, szabályozótervezéshez alkalmazható.

4.2.2. A PI (proporcionális-integráló) szabályozó

PI szabályozás esetén a beavatkozó jelet (u) a szabályozási hiba (e) és annak integráljának súlyozott összegeként számítjuk. Az integráló csatorna a múltbeli hiba összegzett hatását mutatja.

Folytonos idejű PI-szabályozó esetében a beavatkozó jelet az alábbi formában kapjuk:

$$u(t) = K_P \cdot \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau \right), \quad K_P, T_i > 0. \quad (4.3)$$

A $K_P > 0$ paraméter a proporcionális erősítés, $T_i > 0$ az integrálási idő.

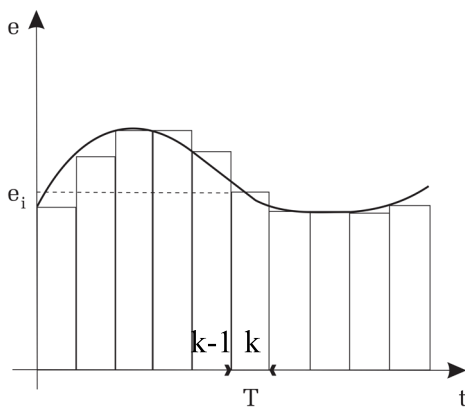
Az átviteli függvényt az alábbi formában kapjuk:

$$\begin{aligned} U(s) &= K_P \cdot \left(E(s) + \frac{1}{T_i s} E(s) \right) \\ H_{PI}(s) &= \frac{U(s)}{E(s)} = K_P \cdot \left(1 + \frac{1}{T_i s} \right). \end{aligned} \quad (4.4)$$

A PI szabályozó egy zérust tartalmaz, valamint az integráló hatás miatt a pólusa a zéróban van.

A szabályozó mintavételes megvalósításánál az integráló tag megközelítésére a téglalap módszert lehet alkalmazni. Legyen T a mintavételi periódus. Az $e(t)$ időben felvett hibafüggvény alatti területet T szélességű téglalapokkal közelítjük meg (lásd 4.1. ábra). Az integrál megközelítőleg egyenlő a téglalapok területének összegével a k -dik mintavételig:

$$\int_0^{kT} e(t) dt \approx \sum_{i=0}^k T e[i]. \quad (4.5)$$



4.1. ábra. Az integrál megközelítése téglalapokkal

Felhasználva az (4.5) megközelítést, a beavatkozójel számítása a k -dik mintavételben:

$$u[k] = K_P \left(e[k] + \frac{T}{T_i} \sum_{i=0}^k e[i] \right). \quad (4.6)$$

Egyszerűbb megvalósítási forma kapható, ha a beavatkozójel számítása rekurzívan történik. Felírva a beavatkozó jelet a k és $k-1$ mintavételben, majd egymásból kivonva:

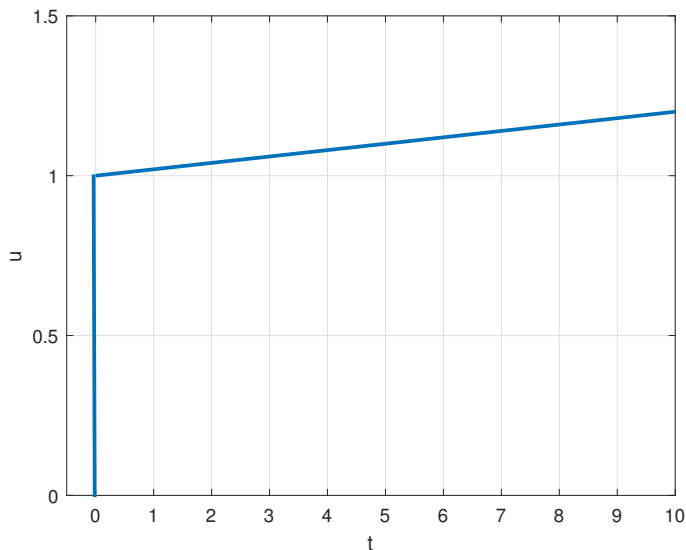
$$\begin{aligned} u[k] &= K_P \left(e[k] + \frac{T}{T_i} \sum_{i=0}^k e[i] \right) \\ u[k-1] &= K_P \left(e[k-1] + \frac{T}{T_i} \sum_{i=0}^{k-1} e[i] \right) \\ u[k] - u[k-1] &= K_P \cdot \left(e[k] - e[k-1] + \frac{T}{T_i} e[k] \right). \end{aligned} \quad (4.7)$$

Így a beavatkozójel a k -dik mintavételben:

$$u[k] = u[k-1] + K_P \left(e[k] - e[k-1] + \frac{T}{T_i} e[k] \right) \quad k = 1, 2, \dots, \quad (4.8)$$

ahol $u[0] = e[0] = 0$. Látszik, hogy a beavatkozójel értéke mindaddig változni fog, amíg a szabályozási hiba nem válik nullává.

Egységugrás hibabemenet esetén a beavatkozó jel a K_P értékre ugrik, majd lineárisan nőni fog, lásd 4.2. ábra. Minél kisebb a T_i paraméter értéke, a növekedés meredeksége annál nagyobb.



4.2. ábra. Az ideális PI-szabályozó egységugrásra adott válasza ($K_P = 1$, $T_i = 50$)

4.2.3. A szögsebesség-szabályozó tervezése és vizsgálata

A szabályozási célok:

- Állandósult állapotban a szögsebesség tartson az előírt értékhez ($\lim_{t \rightarrow \infty} (\omega_r - \omega(t)) = 0$) konstans terhelő nyomaték (τ_l) jelenlétében is.
- Tranziens állapotban a szabályozási idő legyen állítható szabályozó paraméterek segítségével, a szabályozott válasz legyen túllövésmentes.

Modell szabályozótervezéshez: zéró kezdeti sebességállapotot feltételezve alkalmazhatjuk a (4.2) egyenletre a Laplace-transzformáltat:

$$T_F s \Omega(s) + \Omega(s) = K_F (U(s) + D(s)), \quad (4.9)$$

ahol:

$$T_F = \frac{JR}{c_\tau c_\omega} \quad (4.10)$$

$$K_F = \frac{1}{c_\omega} \quad (4.11)$$

$$D(s) = \frac{R}{c_\tau c_\omega} \mathcal{T}_l(s) \quad (4.12)$$

Terhelés hiányában ($D(s) = 0$) felírhatjuk az irányított folyamat átviteli függvényét:

$$H_F(s) = \frac{\Omega(s)}{U(s)} = \frac{K_F}{T_F s + 1} \quad (4.13)$$

Szabályzótervezés: figyelembe véve, hogy a folyamatnak csak egy stabil pólusa van, illetve azt a tényt, hogy az integrátort tartalmazó szabályozók képesek kompenzálni a konstans terhelés hatását, a szabályozási feladathoz PI szabályozót alkalmazunk.

Legyen az előírt (referencia) szögsebesség ω_r és a szabályozási hiba

$$E(s) = \Omega_r(s) - \Omega(s). \quad (4.14)$$

Alkalmazva a pólus-zérus kiejtés módszerét, válasszuk az integrálási időt: $T_i = T_F$. Az így választott szabályozóparaméterrel a nyílt és a zárt rendszer modellje:

$$H_N(s) = \frac{\Omega(s)}{E(s)} = \frac{K_P K_F}{T_i s} \quad (4.15)$$

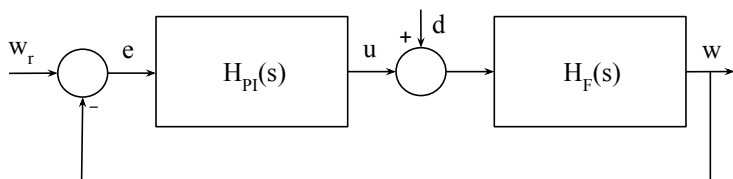
$$H_o(s) = \frac{\Omega(s)}{\Omega_r(s)} = \frac{1}{\frac{T_i}{K_P K_F} s + 1} \quad (4.16)$$

A zárt rendszer erősítése 1, a pólusa stabil, tehát állandósult állapotban biztosítja az alapjelkövetést. A zárt rendszer időállandója a K_P szabályozóparaméterrel módosítható.

A szabályozás elemzése zavar jelenlétében: az állandósult állapotbeli hiba számításához feltételezzük, hogy a terhelés (zavar) konstans, egység-ugrással modellezhető ($D(s) = 1/s$), valamint $\Omega_r(s) = 0$. Az állandósult állapotbeli hiba, lásd 4.3. ábra:

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s E(s) \\ &= \lim_{s \rightarrow 0} s \frac{H_F(s)}{1 + H_N(s)} D(s) = \lim_{s \rightarrow 0} \frac{\frac{K_F}{T_F s + 1}}{1 + \frac{K_P K_F}{T_i s}} = 0. \end{aligned} \quad (4.17)$$

Tehát konstans terhelés mellett az állandósult állapotbeli hiba zéró marad. Mivel a rendszer lineáris, a zavarelnyomás állandósult állapotban akkor is biztosított, ha $\Omega_r(s) \neq 0$.



4.3. ábra. A szabályozási hurok tömbvázlata

4.3. A gyakorlat menete

A *gyakorlat célja*: készítsünk egy szabályozási algoritmust a DC motor szögsebességének előírt konstans referencia tartására.

1. Egy szövegdobozból olvassuk be a referencia szögsebességet.
2. A PI szabályozó paramétereit olvassuk be a megfelelő csúszkáról.
3. Implementáljuk a PI szabályozó algoritmust, felhasználva a referencia és mért sebesség értékeket, illetve a beolvasott paramétereket.

Az elméleti alapok szerint szükségünk lesz két perszisztens változóra: előző minta szabályozási hibája e_{k-1} és az előző minta szabályozási jele u_{k-1} (lásd (4.8). egyenlet). Inicializáljuk ezen változókat nullával.

Mivel a szabályozási feladat periodikusan végzendő, a programot az időzítő által kezelt módszerben kell megírunk. Ugyanakkor arra is kell figyelni, hogy csak abban az esetben küldjünk ki szabályozó jelet, ha az engedélyezés be van kapcsolva.

```

if (OPEN_LOOP)
    // Lab work 1
    if (VELOCITY_CONTROL)
        // Lab work 3
    if (POSITION_CONTROL)
        // Lab work 4
  
```

```
if (TRACKING_CONTROL)
    // Lab work 5
```

A labortevékenységet kiválasztó csúszka lekezelése után a sebességszabályozással foglalkozunk. Beolvassuk a referenciát a neki megfelelő nevű szövegdobozból, majd eltároljuk lebegőpontos számként.

A proporcionális erősítést és az integrálási időt a *KP* és *TI* csúszkákról olvassuk be. Ezek az értékek 0-100, illetve 1-100 közötti egész számok, melyeket skálázni kell. Osszuk el az integrálási időt és a proporcionális erősítést 1000-rel.

Számoljuk ki az aktuális szabályozási hibát $e_k = ref - v_k$, a mért sebesség és a referencia alapján. Számoljuk ki a szabályozási jelet a (4.8)-as összefüggés alapján. A kimenet védelme miatt korlátozzuk le a szabályozó jelet $[-10, 10]V$ intervallumra. Az aktuális szabályozó jelet és hibát mentjük el a múltbeli változókba: $e_{k-1} = e_k$, $u_{k-1} = u_k$. Végül írassuk ki a szabályozó jelet a neki megfelelő szövegdobozba.

A szabályozójel kiküldését lásd a 2.3-as alfejezetben.

Tesztelés terhelés jelenlétében: miután meggyőződünk a szabályozó algoritmus működéséről, vizsgáljuk meg, hogyan viselkedik a rendszer, ha módosítjuk a terhelő nyomatékot. A szabályozott egyenáramú motor tengelyére az áttétel után helyeztünk egy másodlagos egyenáramú motort (lásd a Függelék 1-es ábráját). Ha a motor bemenetét rövidre zárjuk, akkor a motor aktív fékezést biztosít (Lenz törvénye magyarázza ezt a jelenséget). Vizsgáljuk meg a szabályozási pontosságot aktív fékezés mellett.

4.4. Kérdések és feladatok

1. Mekkora maximális terhelő nyomatékot tudunk kompenzálni, ha ismert a motor maximális szögsebessége és a maximális tápfeszültség (használjuk a (4.2)-es egyenletet)?
2. Hogyan néz ki a diszkrét PI szabályozó húrtrapéz megközelítéssel?
3. Bővítsük ki a programot úgy, hogy kezeljük le az integrátor wind-up jelenségét (lásd [4]).

SZÖGELFORDULÁS-SZABÁLYOZÁS

5.1. A gyakorlat célja

A gyakorlat célja egyenáramú motorral megvalósított szögelfordulás szabályozása konstans előírt érték esetén, a pozíciószabályozás modellalapú tervezése és szoftveres megvalósítása.

5.2. Elméleti bevezető

5.2.1. Egyenáramú motor modellezése szögelfordulás-szabályozáshoz

Az egyenáramú motor pozíciószabályozása esetén a modellezésénél a bemeneti feszültség és a terhelés hatását vizsgáljuk a szögelfordulásra. Az alkalmazott matematikai modellben elhanyagoljuk a motor elektromos idő-állandóját:

$$\begin{cases} J_m \frac{d\omega}{dt} + A_m \omega = \underbrace{K_m u}_{\tau_m} - \tau_l \\ \frac{d\alpha}{dt} = \omega \end{cases} \quad (5.1)$$

A modellben u a rotorra adott bemeneti feszültséget (beavatkozó jel), τ_l a terhelést, τ_m a motor által kifejtett nyomatékot, ω a szögsebességet és α a szögelfordulást (szabályozott jellemző) jelölik. A paraméterek: J_m a motor rotorjának inerciája, K_m a bemenet erősítési tényezője, $A_m \omega$ a visszaindult feszültség által generált fékezőnyomaték $A_m > 0$ együtthatóval.

A zavarhatást a külső terhelés jelenti, amit az inerciális terhelés és sűrűlódási hatás által kifejtett nyomaték összegzett hatásaként modellezünk.

$$\tau_l = J_l \frac{d\omega}{dt} + \tau_f \quad (5.2)$$

ahol τ_f a sűrűlódás által kifejtett fékezőnyomaték, J_l a terhelés inerciája

A súrlódási hatás modellezésénél a viszkózus (sebességgel arányos csillapítás) és száraz (Coulomb) súrlódási komponenseket vesszük figyelembe:

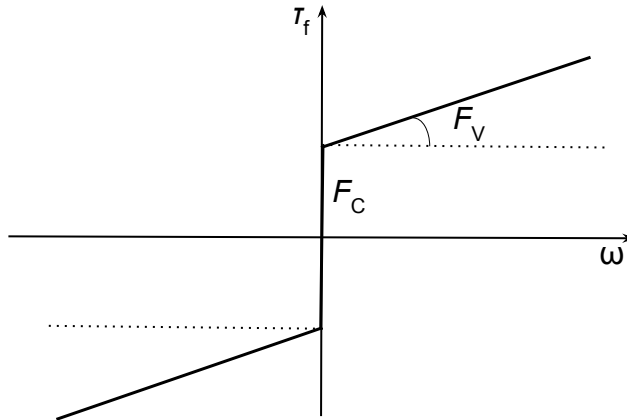
$$\tau_f(\omega, \tau_m) = \begin{cases} F_C \text{sign}(\omega) + F_V \omega, & \text{amikor } |\tau_m| > F_C \text{ vagy } |\omega| > 0. \\ \tau_m, & \text{amikor } |\tau_m| < F_C \text{ és } \omega = 0 \end{cases} \quad (5.3)$$

Paraméterek: $F_V > 0$ viszkózus súrlódási együttható (csillapítás), $F_C > 0$ a Coulomb súrlódási (száraz súrlódási) együttható.

$$\text{sign}(\omega) = \begin{cases} 1, & \text{amikor } \omega > 0 \\ 0, & \text{amikor } \omega = 0 \\ -1, & \text{amikor } \omega < 0. \end{cases} \quad (5.4)$$

A súrlódási modell azt mutatja, hogy abban az esetben, amikor a motor által kifejtett nyomaték értéke kisebb, mint a Coulomb súrlódási együttható, a mozgás nem indul el (*tapadás*). Amikor τ_m eléri az F_C együttható értékét, a mozgás elindul, és a mozgást akadályozó súrlódási erő arányosan nő a sebességgel.

A súrlódási modellt a sebesség függvényében az 5.1. ábra mutatja.



5.1. ábra. Coulomb és viszkózus súrlódás

A tapadásnak zéró sebesség környékén van domináns hatása. Pozíciószabályozás esetén, konstans előírt pozíciót feltételezve, a megálláskor a motor alacsony sebességtartományba kerül, tehát a tapadás befolyásolja a szabályozás minőségét állandósult állapotban.

Figyelembe véve az inerciális terhelést és a súrlódási modellt, az irányított rendszer modellje:

$$J \frac{d^2 \alpha}{dt^2} + A \frac{d\alpha}{dt} = K_m u - \tau_C, \quad (5.5)$$

ahol $J = J_m + J_l$, $A = A_m + F_V$. τ_C a Coulomb súrlódás által generált fékezőnyomatékot jelöli ((5.3) modell úgy, hogy $F_V = 0$).

5.2.2. A PD (proporcionális-deriváló) szabályozó

PD szabályozás esetén a beavatkozó jelet (u) a szabályozási hiba (e) és annak deriváltjának súlyozott összegeként számítjuk. A deriváló csatorna a hiba változásának irányát, valamint nagyságát mutatja.

Folytonos idejű ideális PD szabályozó esetében a beavatkozó jelet az alábbi formában kapjuk:

$$u(t) = K_P \cdot \left(e(t) + T_D \frac{de}{dt} \right), \quad K_P, T_D > 0. \quad (5.6)$$

A $K_P > 0$ paraméter a proporcinális erősítés, $T_D > 0$ a deriválási idő.

Az átviteli függvényt az alábbi formában kapjuk:

$$\begin{aligned} U(s) &= K_P \cdot (E(s) + T_D s E(s)) \\ H_{PD}(s) &= \frac{U(s)}{E(s)} = K_P \cdot (1 + T_D s). \end{aligned} \quad (5.7)$$

Az ideális PD szabályozó egy zérust tartalmaz.

A szabályozó mintavételes megvalósításánál a deriváló tag megközelítésére véges differenciákon alapuló approximációt alkalmazunk:

$$\frac{de}{dt} \approx \frac{e[k] - e[k-1]}{T}. \quad (5.8)$$

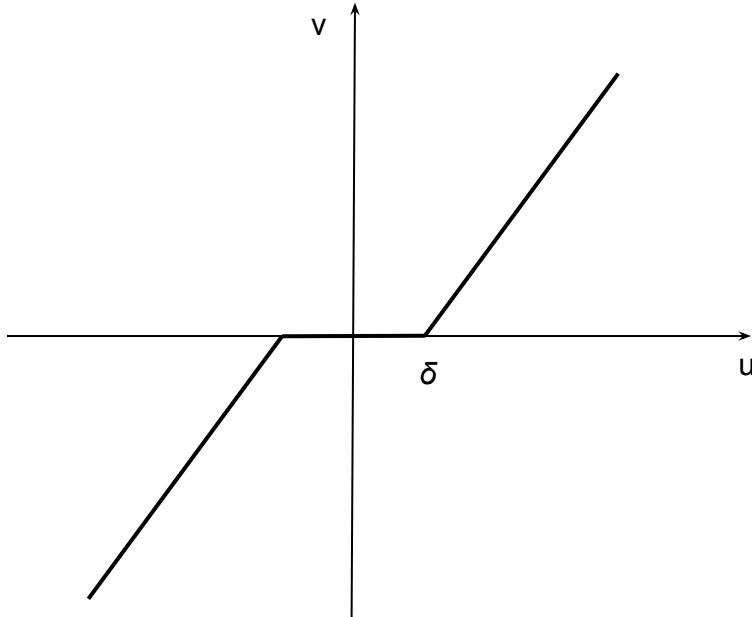
ahol T a mintavételi periódus.

Felhasználva az (5.8) megközelítést, a beavatkozójel számítása a k -dik mintavételben:

$$u[k] = K_P \left(e[k] + T_D \frac{e[k] - e[k-1]}{T} \right). \quad (5.9)$$

5.2.3. A holtsáv és kompenzálása

A holtsáv (Dead Zone) nemlinearitás statikus átvitelét (u bemenet, v kimenet) az 5.2-es ábra mutatja. A kimenet addig nem változik, amíg a bemenet abszolút értéke nem éri el a holtsáv szélesség paraméterét (δ). Ha u abszolút értéke nagyobb mint δ , a kimenet a bemenettel arányosan nő.

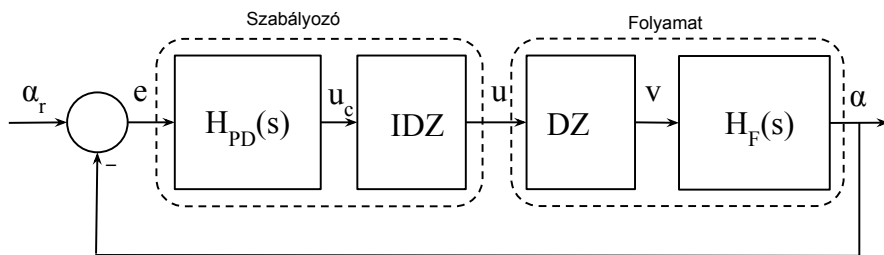


5.2. ábra. Holtsáv

A holtsáv modellje:

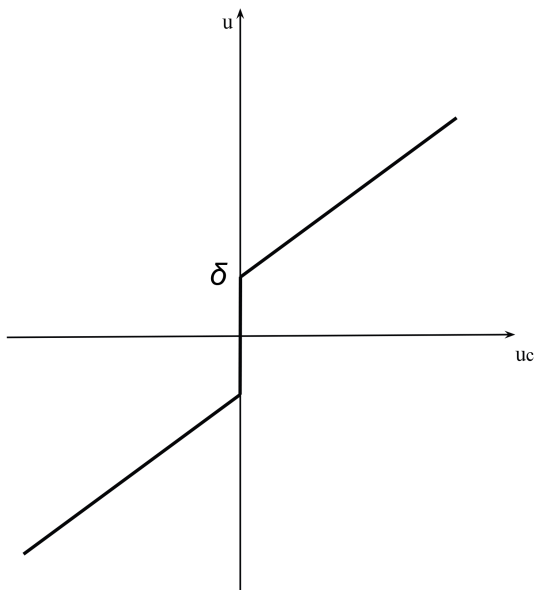
$$v = DZ(u) = \begin{cases} u - \delta, & \text{amikor } u > \delta \\ 0, & \text{amikor } |u| \leq \delta \\ u + \delta, & \text{amikor } u < -\delta \end{cases} \quad (5.10)$$

Mechanikai irányítási rendszerekben a holtsáv hatékony modell a tapadási súrlódási jelenség leírására. A tapadás miatt a mozgás csak akkor indul el, amikor a bemeneti erő/nyomaték elérte az F_C értéket, lásd az (5.3)-as modellt. A mechanikai rendszert tapadási súrlódással úgy modellezhetjük, mint egy statikus holtsávmodell, ahol $\delta = F_C$, sorban a mechanikai rendszer lineáris részével (lásd az 5.3. ábrán a *Folyamatot*).



5.3. ábra. Pozíciósabályozás holtávkompenzációval

A holtáv kompenzálására (hatásának kiegyenlítésére) a szabályozó kimenetét kell módosítanunk a holtáv inverzének modelljével:



5.4. ábra. A holtáv inverze

Az inverz holtáv modell:

$$u = IDZ(u_c) = \begin{cases} u_c + \delta, & \text{amikor } u_c > 0 \\ 0, & \text{amikor } u_c = 0 \\ u_c - \delta, & \text{amikor } u_c < 0 \end{cases} \quad (5.11)$$

Vegyük észre, hogy a holsáv inverze sorban a holsávval egységnyi átvitelnek felel meg, a holsáv inverze képes kompenzálni a holsáv hatását.

$$v = DZ(u) = DZ(IDZ(u_c)) = u_c \quad (5.12)$$

Tehát a mechanikai szabályozási rendszerekben a holsáv jelenségét úgy kompenzálhatjuk, hogy a holsáv nélküli rendszerre tervezett szabályozó kimenetét módosítjuk az $IDZ(\cdot)$ függvénnyel, lásd 5.3. ábra.

5.2.4. A szögpozíció szabályozó tervezése és vizsgálata

A szabályozási célok:

- Állandósult állapotban a szögelfordulás értéke tartson a konstans előírt értékhez ($\lim_{t \rightarrow \infty} (\alpha_r - \alpha(t)) = 0$) tapadó súrlódás jelenlétében is.
- Transziens állapotban a szabályozási idő legyen állítható szabályozó paraméterek segítségével, a szabályozott válasz legyen túllövésmentes.

Modell szabályozótervezéshez: Zéró kezdeti állapotot feltételezve és elhanyagolva a τ_C tagot alkalmazhatjuk az (5.5)-ös motormodellre a Laplace transzformáltat:

$$T_F s^2 \mathcal{A}(s) + s \mathcal{A}(s) = K_F U(s), \quad (5.13)$$

ahol:

$$T_F = J/A \quad (5.14)$$

$$K_F = K_u/A, \quad (5.15)$$

és $\mathcal{A}(s)$ az $\alpha(t)$ szögelfordulás Laplace transzformáltját jelöli.

Tehát az irányított folyamat lineáris részének átviteli függvénye:

$$H_F(s) = \frac{\mathcal{A}(s)}{U(s)} = \frac{K_F}{s(T_F s + 1)} \quad (5.16)$$

Szabályzótervezés: figyelembe véve, hogy H_F -nek csak egy stabil pólus van, és tartalmaz integrátort, a szabályozási feladathoz PD szabályozót alkalmazunk.

Legyen az előírt (referencia) szögelfordulás α_r és a szabályozási hiba

$$e(s) = \mathcal{A}_r(s) - \mathcal{A}(s) \quad (5.17)$$

Alkalmazva a pólus-zérus kiejtés módszerét, válasszuk az deriválási időt:

$$T_D = T_F. \quad (5.18)$$

Az így választott szabályozóparaméterrel a nyílt és a zárt rendszer:

$$H_N(s) = \frac{\mathcal{A}(s)}{E(s)} = H_{PD}(s)H_F(s) = \frac{K_P K_F}{s} \quad (5.19)$$

$$H_o(s) = \frac{\mathcal{A}(s)}{\mathcal{A}_r(s)} = \frac{1}{\frac{1}{K_P K_F} s + 1} \quad (5.20)$$

A zárt rendszer erősítése 1, a pólusa stabil, tehát állandósult állapotban biztosítja az alapjelkövetést. A zárt rendszer időállandója a K_P szabályozóparaméterrel módosítható.

A PD szabályozó kimenetét az (5.9) összefüggéssel számíthatjuk.

A PD által kiszámított beavatkozó jelet szükséges még a holsáv inverzével módosítani a tapadási súrlódás kompenzálásához, lásd (5.11) összefüggés.

A szabályozás szimulációs vizsgálata paraméterbizonytalanság jelenlétében: számos alkalmazás esetében a T_F paraméter nem ismert pontosan, mivel függ a terhelés inerciájától és a viszkózus súrlódási paramétértől. Ezért az (5.18) pólus-zérus kiejtést nem tudjuk pontosan elvégezni, és a nyílt rendszer modellje:

$$H_N(s) = H_{PD}(s)H_F(s) = \frac{K_P K_F (T_D s + 1)}{s(T_F s + 1)} \quad (5.21)$$

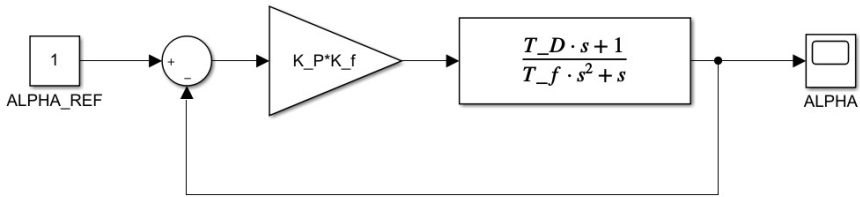
A szimulációk segítségével megvizsgálható, hogy mi történik abban az esetben, hogyha T_D nem kompenzálja tökéletesen T_F -et. A szimulációs modell az 5.5. ábrán látható ($K_P = 1$, $K_F = 1$, $T_F = 1$).

Abban az esetben, amikor T_D nagyobb, mint a folyamat időállandója (T_F), lassabb választ kapunk.

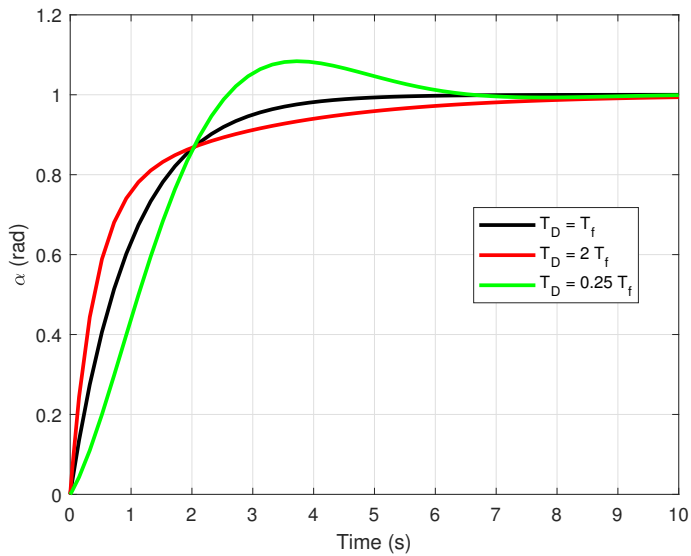
Abban az esetben, amikor T_D kisebb, mint a folyamat időállandója (T_F), túllövés jelenik meg a szabályozott válaszban, lásd 5.6. ábra.

5.3. A gyakorlat menete

A gyakorlat célja: készítsünk egy szabályozási algoritmust a DC motor szögpozíciójának előírt konstans referencia tartására.



5.5. ábra. Pozíciószabályozási hurok modellje



5.6. ábra. T_D paraméter hangolásának hatása

1. Mérjük meg a holtidőt a motor felnyitott vezérlésével.
2. Egy szöveges dobozból olvassuk be a referencia pozíciót.
3. A PD szabályozó paramétereit olvassuk be a megfelelő csúszkákról.
4. Implementáljuk a PD szabályozó algoritmust holtidőkompenzálással, felhasználva a referencia és mért sebesség értékeket, illetve a beolvasott paramétereket.

Holtsáv mérése: induljunk ki az (5.1)-es egyenletből. A célunk, hogy megkapjuk azt a maximális feszültséget, amivel betáplálva a motorunkat még nem mozdul ki. Ha nincs mozgás, α konstans, azaz $\dot{\alpha} = \ddot{\alpha} = 0$, ahonnan $\delta := u = \frac{\tau_c}{K_u}$. Azaz, a tápfeszültség mérésével nem a sűrűlódási nyomatékot mérjük, hanem az azzal arányos értéket, amit fel tudunk használni a holsáv kompenzálására (lásd az (5.11)-es összefüggést). Figyelni kell arra, hogy ha egy magas feszültségszintről indulunk, és csökkentjük a feszültséget, a motor forgása miatt téves eredményt kapunk. Javasolt 0V-ról indulni felfele, ellenkező esetben szükséges a motort megállítani minden mérés előtt.

Adatok előkészítése: a pozíció szabályozáshoz az előírt pozíció értékét a *Position_Ref_textBox* szöveges dobozból olvassuk be $^\circ/s$ mértékegységben a *getTextBoxVal* metódus segítségével. A proporcionális erősítést és a deriválási időt a *KP_trackBar* és *TD_trackBar* csúszkákról olvassuk le a *getTrackBarVal* metódus segítségével. A beolvasott értékek 0 és 100 között változhatnak egységnyi lépéssel. Ezen értékeket skálázzuk a megfelelő dinamika biztosítása érdekében: a *KP_trackBar* és *TD_trackBar* értékét osszuk el 1000-rel.

PD szabályozó: a pozíciószabályozó algoritmus implementálásához leg hamarabb a szabályozási hibát számoljuk ki:

$$e[k] = \alpha_{ref} - \alpha[k]$$

A PD szabályozót az (5.9)-es összefüggés alapján programozzuk le, ahol T a mért mintavételi periódus.

Holtsávkompenzálás: a holsáv kompenzálásához felhasználjuk a felnyitott rendszeren végzett mérésünk eredményét (δ):

```

if(u_k > 0){
    u_k = u_k + delta;
}
if(u_k < 0){
    u_k = u_k - delta;
}

```

A szabályozójel korlátozását és kiküldését az eddigi programnak már meg kell oldania.

5.4. Kérdések és feladatok

1. Hasonlítsuk össze a szabályozási hibát holtávkompenzálás nélkül és kompenzálással.
2. Cseréljük le a holtávkompenzálást integráló taggal, PID szabályozót alkalmazva.
3. Módosítsuk a programot úgy, hogy konstans referencia helyett lineárisan változó referenciát kövessen. Hogyan változik a szabályozási hiba?

PÁLYAKÖVETÉST MEGVALÓSÍTÓ SZÖGELFORDULÁS-SZABÁLYOZÁS

6.1. A gyakorlat célja

A gyakorlat szögelfordulás-szabályozás megvalósításával foglalkozik időben változó előírt értékek esetén. A gyakorlat célja a pozíció, sebesség és gyorsuláspályák tervezése és implementálása, a pályakövetést megvalósító szabályozás tervezése és szoftveres megvalósítása. Feltételezzük, hogy az alkalmazott beavatkozó egyenáramú motor.

6.2. Elméleti bevezető

6.2.1. Minimális idejű polinomiális pályatervezés

A pályatervezés célja előírt pozíció, sebesség és gyorsulás meghatározása a szabályozás minden időpontjában.

A tervezéshez szükséges bemeneti adatok:

- kiindulópozíció: x_0
- célpozíció: x_E
- abszolút maximális sebesség: $v_M > 0$
- abszolút maximális gyorsulás: $a_M > 0$

Az elérhető maximális sebesség a terhelés oldalon az alkalmazott beavatkozó maximális sebességétől, valamint az alkalmazott áttételi aránytól függ.

Az elérhető maximális gyorsulás a beavatkozó motor és áttétel által kifejezhető maximális erő/nyomaték függvényében határozható meg.

A tervezés eredménye az előírt pozíció ($x = x(t)$), sebesség ($v = v(t)$) és gyorsulás ($a = a(t)$) időfüggvényei, amely segítségével a mozgás minden t időpontjában a pozíció, sebesség, gyorsulás meghatározhatóak.

Jelölje t_0 az indulás időpontját $x(t_0) = x_0$ (t_0 ismert adat a pályatervezéshez) és t_E a megérkezés időpontját $x(t_E) = x_E$.

A számítások egyszerűsítéséhez bevezetünk két feltételezést:

- $x_0 < x_E$
- $v(t_0) = 0, v(t_E) = 0$

Ugyanakkor a leírt módszer könnyen általánosítható, a feltételezések feloldhatóak.

Az előírt pozíciót ($x = x(t)$), sebességet ($v = v(t)$) és gyorsulást ($a = a(t)$) nullad-, első-, illetve másodfokú polinomok segítségével fogjuk leírni az idő függvényében. A pályatervezésnél figyelembe kell venni, hogy:

$$\dot{x}(t) = v(t), \quad (6.1)$$

$$\dot{v}(t) = a(t). \quad (6.2)$$

Az alkalmazott tervezési eljárás biztosítja, hogy adott v_M és a_M mellett a pálya időben optimális legyen.

A pályát az alábbi elv alapján tervezzük: a beavatkozót gyorsítjuk maximális gyorsulással, amíg elérjük a maximális sebességet. Ezután tartjuk a maximális sebességet nulla gyorsulás mellett. A pálya végén ugyanannyi időt, mint amennyit gyorsítottunk, lassítunk negatív maximális gyorsulással.

A 6.1. ábra egy ilyen pályát mutat az alábbi paraméterekkel: $t_0 = 0s$, $x_0 = 0m$, $x_E = 5m$, $v_M = 1m/s$, $a_M = 0.5m/s^2$.

Tehát a pálya három szakaszra bontható fel: gyorsuló (1), konstans sebességű (2) és lassuló szakaszra (3). Ahhoz, hogy a pozíciópálya a szakaszok találkozási pontjában (1-2, 2-3) folytonos legyen, a találkozási pontokban a sebességértékek (pozíció deriváltja) egyenlőek kell legyenek.

A *gyorsuló szakaszon* (1) a gyorsulás konstans, tehát a sebesség elsőfokú, a pozíció másodfokú polinommal írható le. Legyen általánosan a polinomok alakja:

$$x_1(t) = c_{x1}(t - t_0)^2 + b_{x1}(t - t_0) + a_{x1} \quad (6.3)$$

$$v_1(t) = b_{v1}(t - t_0) + a_{v1} \quad (6.4)$$

$$a_1(t) = a_{a1} \quad (6.5)$$

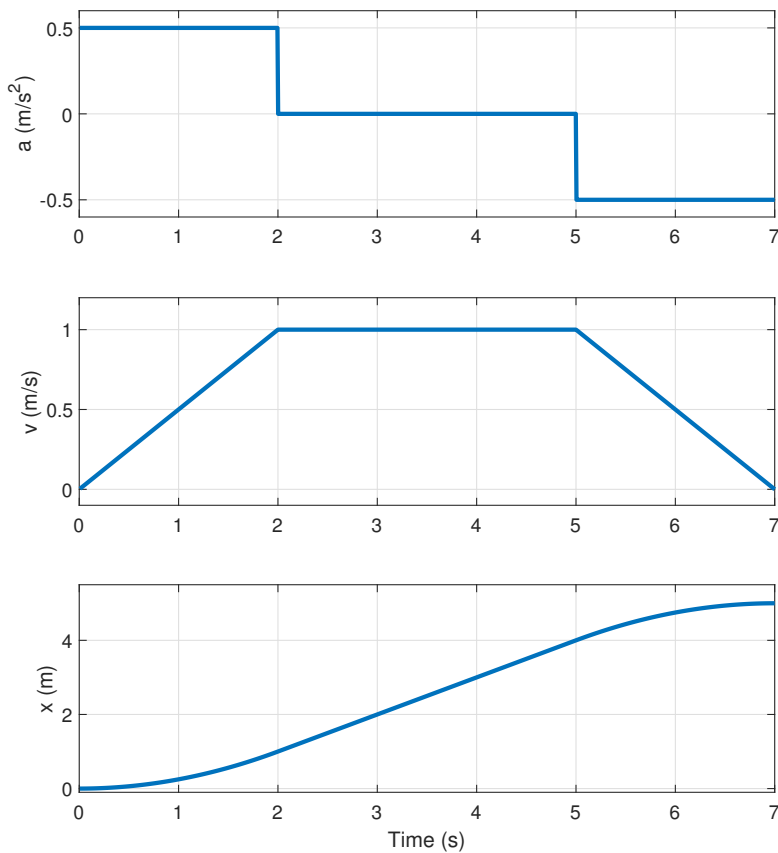
Határozzuk meg a polinomok együtthatóit.

Ebben a szakaszban a gyorsulás konstans maximális, tehát $a_{a1} = a_M$.

Figyelembe véve, hogy $\dot{v}_1(t) = a_1(t)$, következik $b_{v1} = a_{a1} = a_M$.

Mivel a sebesség a t_0 pillanatban 0, következik $a_{v1} = 0$.

Figyelembe véve, hogy $\dot{x}_1(t) = v_1(t)$, következik $c_{x1} = b_{v1}/2 = a_M/2$, valamint $b_{x1} = a_{v1} = 0$.



6.1. ábra. *Gyorsulás, sebesség és pozíció profilok*

Mivel a pozíció a t_0 pillanatban x_0 , következnek, hogy $a_{x1} = x_0$.

Az ismert együttthatójú sebesség- és pozíciópolinomokból meghatározható a gyorsuló szakasz lejártának időpontja (t_1), valamint pozíciója (x_1). Mivel $v(t_1) = v_M$, következnek:

$$t_1 = \frac{v_M}{a_M} + t_0 \quad (6.6)$$

$$x_1 = x(t_1) = \frac{v_M^2}{2a_M} + x_0 \quad (6.7)$$

A *konstans sebesség szakaszon* (2) a gyorsulás zéró, így a sebesség- és pozíciópolinomok általánosan:

$$x_2(t) = b_{x2}(t - t_1) + a_{x2} \quad (6.8)$$

$$v_2(t) = a_{v2} \quad (6.9)$$

$$a_2(t) = 0 \quad (6.10)$$

Ebben a szakaszban a sebesség maximális, $a_{v2} = v_M$.

Figyelembe véve, hogy $\dot{x}_2(t) = v_2(t)$, következik $b_{x2} = a_{v2} = v_M$.

A $t = t_1$ pillanatban a pozíció $x_2(t_1) = x_1$, tehát $a_{x2} = x_1 = v_M^2/(2a_M) + x_0$.

Figyelembe véve, hogy a gyorsulási és lassulási szakaszok időtartama és hossza egyenlőek, meghatározható a konstans sebesség szakasz befejezésének időpillanata (t_2) és pozíciója (x_2):

$$x_E - x_2 = x_1 - x_0 \quad (6.11)$$

$$v_M = \frac{x_2 - x_1}{t_2 - t_1} \quad (6.12)$$

Innen következik, hogy

$$x_2 = x_E - x_1 + x_0 \quad (6.13)$$

$$t_2 = \frac{x_2 - x_1}{v_M} + t_1 \quad (6.14)$$

A *lassuló szakaszon* (3) a gyorsulás konstans negatív. Legyen általánosan a polinomok alakja:

$$x_3(t) = c_{x3}(t - t_2)^2 + b_{x3}(t - t_2) + a_{x3} \quad (6.15)$$

$$v_3(t) = b_{v3}(t - t_2) + a_{v3} \quad (6.16)$$

$$a_3(t) = a_{a3} \quad (6.17)$$

Határozzuk meg a polinomok együtthatóit.

Ebben a szakaszban a gyorsulás konstans maximális, de negatív előjellel, tehát $a_{a3} = -a_M$.

Figyelembe véve, hogy $\dot{v}_3(t) = a_3(t)$, következik $b_{v3} = a_{a3} = -a_M$.

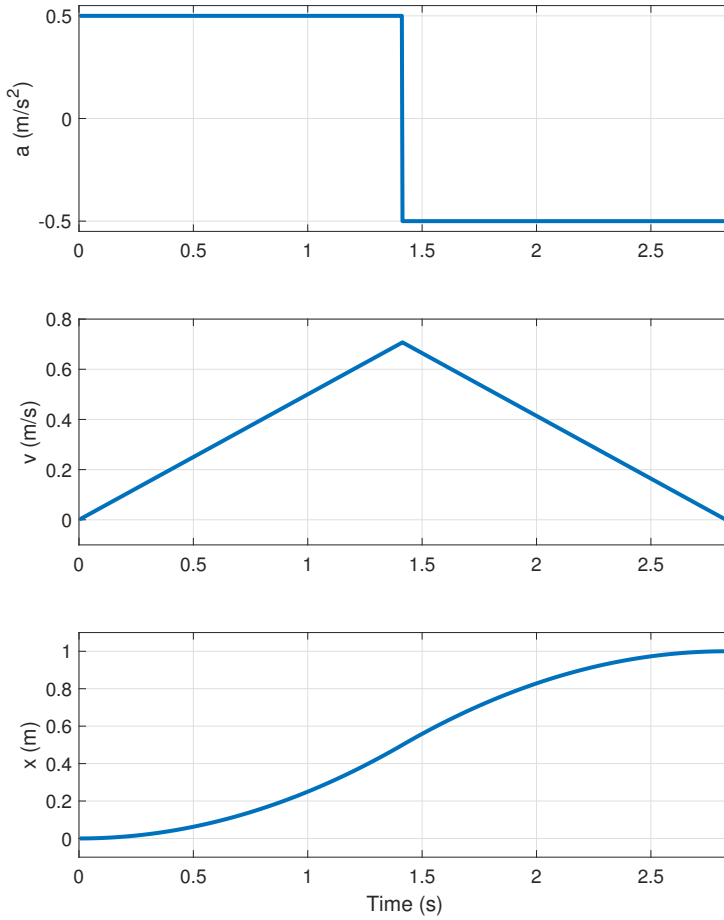
Mivel a sebesség a t_2 pillanatban v_M , következik $a_{v3} = v_M$.

Figyelembe véve, hogy $\dot{x}_3(t) = v_3(t)$, következik $c_{x3} = b_{v3}/2 = -a_M/2$, valamint $b_{x3} = a_{v3} = v_M$.

Mivel a pozíció a t_2 pillanatban x_2 , következik, hogy $a_{x3} = x_2$.

A pálya megérkezési pillanata (t_E) az alábbi módon számítható:

$$t_E = t_0 + 2(t_1 - t_0) + \frac{x_2 - x_1}{v_M} \quad (6.18)$$



6.2. ábra. Gyorsulás, sebesség és pozíció profilok (rövid pálya)

Kétszakaszos pálya

Abban az esetben, amikor a kiindulópozíció és a célpozíció közötti távolság „kicsi”, előfordulhat, hogy a sebesség a gyorsuló szakasz végéig nem éri el a v_M értéket. Ebben az esetben a pálya csak egy gyorsuló és egy lassuló szakaszt fog tartalmazni, a maximális sebesség pedig kisebb lesz, mint v_M .

A 6.2. ábrán ilyen pályára látunk példát az alábbi paraméterekkel: $t_0 = 0s$, $x_0 = 0m$, $x_E = 1m$, $v_M = 1m/s$, $a_M = 0.5m/s^2$.

Annak a feltétele, hogy a pálya csak kétszakaszos legyen:

$$x_E - x_0 < 2(x_1 - x_0), \quad (6.19)$$

ahol x_1 -et a (6.7)-es összefüggés alapján számítjuk.

A kétszakaszos pályatervezéshez ugyanazokat a bemeneti adatokat és feltételezéseket alkalmazzuk, mint általános esetben.

Jelölje v_m az elérhető maximális sebességet:

$$v_m < v_M \quad (6.20)$$

Az értéke meghatározásához először kiszámítjuk a gyorsulási szakasz idejét.

A *gyorsulási szakaszt* (1) definiáló polinomiális pályák alakja ugyanaz, mint az általános esetben, csak a pályaszakasz hossza és ideje (x_1 , t_1) változik:

$$x_1(t) = \frac{a_M}{2}(t - t_0)^2 + x_0 \quad (6.21)$$

$$v_1(t) = a_M(t - t_0) \quad (6.22)$$

$$a_1(t) = a_M \quad (6.23)$$

Mivel a gyorsulási szakasz hossza a teljes pályahossz fele, a (6.21) és (6.22) alapján következik:

$$x_1 = x_1(t_1) = (x_E - x_0)/2 \quad (6.24)$$

$$t_1 = \sqrt{\frac{2(x_1 - x_0)}{a_M}} + t_0 \quad (6.25)$$

$$v_m = a_M(t_1 - t_0) \quad (6.26)$$

A *lassulási szakaszt* (3) definiáló polinomiális pályák alakja:

$$x_3(t) = -\frac{a_M}{2}(t - t_1)^2 + v_m(t - t_1) + x_1 \quad (6.27)$$

$$v_3(t) = -a_M(t - t_1) + v_m \quad (6.28)$$

$$a_3(t) = -a_M \quad (6.29)$$

A pálya megérkezési pillanata: $t_E = 2(t_1 - t_0) + t_0$.

6.2.2. A pályakövetést megvalósító szabályozó tervezése

A szabályozási célok:

- Legyen egy időben folytonos pozíciópálya $\alpha_r(t)$. A szögelfordulás értéke tartson az előírt értékhez ($\lim_{t \rightarrow \infty} (\alpha_r(t) - \alpha(t)) = 0$).
- Tranziens állapotban a szabályozási idő legyen állítható szabályozó paraméter segítségével, a szabályozott válasz legyen túllövésmentes.

A pályakövetést megvalósító szabályozó tervezéshez az alábbi folyamat-modellt alkalmazzuk:

$$J \frac{d^2 \alpha}{dt^2} + A \frac{d\alpha}{dt} = K_u u, \quad (6.30)$$

ahol α a szögelfordulás (szabályozott kimenet), u a rotorra adott feszültség (beavatkozó jel). Paraméterek: J a motor és terhelés inerciája, K_u bemeneti erősítés, A a csillapítási tényező. A csillapítási tag a rotorban visszaindukált feszültség, valamint a viszkózus súrlódási komponens miatt jelenik meg.

Holtsáv kompenzálása: A tapadó súrlódást a szabályozó tervezésére alkalmazott modell nem tartalmazza. Ugyanakkor a tapadó súrlódás jelentősen befolyásolja a pozíciószabályozás pontosságát. Ennek a hatását a holtsávmodell alapú eljárással lehet kompenzálni. A módszer leírásához lásd az 5.2.1. és 5.2.3. alfejezeteket.

A szabályozáshoz feltételezzük, hogy adott az előírt szögpozíció, szögsebesség és szöggyorsulás pálya ($\alpha_r(t)$, $\dot{\alpha}_r(t)$, $\ddot{\alpha}_r(t)$), illetve mért a motor szögelfordulása (α) és szögsebessége ($\omega = \dot{\alpha}$).

Megjegyzés: ebben az alfejezetben az előírt szögelfordulás jelöléséhez α_r -et használunk. Ez a pályatervezéssel foglalkozó alfejezetben $x(t)$ -nek felel meg ($\alpha_r(t) := x(t)$). Ennek megfelelően: ($\dot{\alpha}_r(t) := \dot{x}(t) = v(t)$, $\ddot{\alpha}_r(t) := \ddot{x}(t) = a(t)$).

A szabályozó tervezéséhez képezzük az összetett sebesség és pozícióhibát:

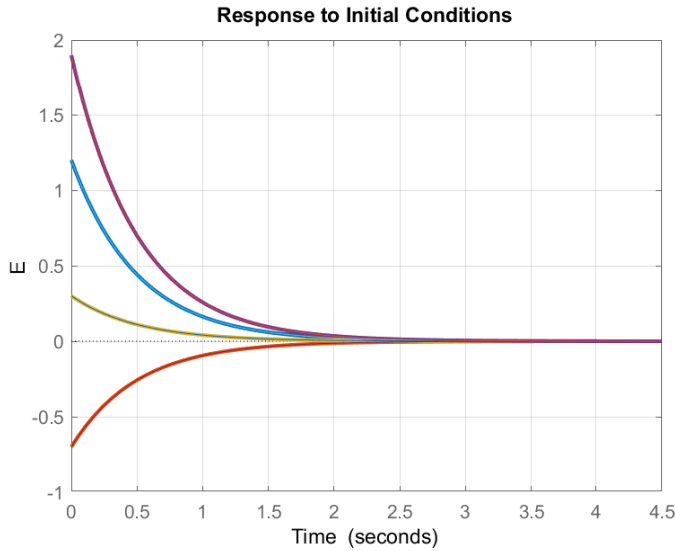
$$E = (\dot{\alpha}_r - \dot{\alpha}) + \lambda(\alpha_r - \alpha), \quad (6.31)$$

ahol $\lambda > 1$ szabályozóparaméter. A λ mértékegysége $1/s$, tehát az összetett hiba tulajdonképpen sebességhiba. λ értékét tipikusan 1-nél nagyobbak választjuk azért, hogy a pozícióhiba nagyobb súllyal jelenjen meg az összetett hibában, mint a sebességhiba.

Az u beavatkozó jelet úgy tervezzük, hogy az összetett hiba szabályozás közben kielégítse az alábbi differenciálegyenletet:

$$\dot{E} + K_P E = 0, \quad E(t = 0) = E_0 \quad (6.32)$$

Látszik, hogy a (6.32)-es egyenlet karakterisztikus polinomja $s + K_P = 0$, tehát $K_P > 0$ esetén (6.32) elsőfokú lineáris differenciálegyenlet $E(t)$ megoldása olyan időfüggvény, amely 0-hoz konvergál bármilyen véges E_0 kezdőállapotból, lásd 6.3. ábra. A konvergencia sebessége annál nagyobb, minél nagyobb K_P .



6.3. ábra. A (6.32)-es egyenlet trajektóriái $K_p = 1$ esetén, különböző E_0 kezdeti értékekre

Az irányított folyamat (6.30) modellje felírható az alábbi alakban is:

$$\ddot{\alpha} = -\frac{A}{J}\dot{\alpha} + \frac{K_u}{J}u \quad (6.33)$$

Behelyettesítve a $\ddot{\alpha}$ szöggyorsulást az összetett hiba deriváltjába, kapjuk:

$$\dot{E} = (\ddot{\alpha}_r - \ddot{\alpha}) + \lambda(\dot{\alpha}_r - \dot{\alpha}) = \ddot{\alpha}_r + \frac{A}{J}\dot{\alpha} - \frac{K_u}{J}u + \lambda(\dot{\alpha}_r - \dot{\alpha}) \quad (6.34)$$

Válasszuk a beavatkozó jelet:

$$u = \frac{J}{K_u} \left(\ddot{\alpha}_r + \frac{A}{J} \dot{\alpha} + \lambda(\dot{\alpha}_r - \dot{\alpha}) + K_P E \right) \quad (6.35)$$

Behelyettesítve a (6.35) beavatkozó jelet a (6.34) összefüggésbe látszik, hogy a szabályozott rendszerben az összetett hiba dinamikája ekvivalens a (6.32) összefüggéssel:

$$\dot{E} = -K_P E \quad (6.36)$$

Tehát $K_P > 0$ esetén a beavatkozó jel garantálja, hogy az összetett pozíció és sebességhiba nullába konvergáljon.

6.3. A gyakorlat menete

A *gyakorlat célja*: implementáljuk az elméleti részben bemutatott polinomiális pályakövető szabályozót adott $x_E > x_0$ kezdeti adatokkal, ismerve a laborstandon levő DC motor paramétereit.

1. Katalógusadatok alapján számoljuk ki a folyamat paramétereit.
2. A maximális gyorsulás, sebesség és végpozíció ismerete alapján döntünk el, hogy három- vagy kétszakaszos pályára van szükség.
3. Generáljuk ki a pályát, felhasználva a mintavételezési periódus nagyságát.
4. Implementáljuk a szabályozó algoritmust holtávkompenzálással.

A *folyamatparaméterek*:

$$\begin{cases} R = 7.13[\Omega] \\ c_1 = \frac{1}{26.16}[Vs/rad] \\ c_2 = 38.2E - 3[Nm/A] \\ J = 1.89E - 4[kgm^2] \end{cases} \Rightarrow$$

$$K_m = \frac{K_u}{J} = 2835[V^{-1}s^{-2}]$$

$$K_\omega = \frac{A}{J} = 10.83[radV^{-2}s^{-3}]$$

Mivel pozíciószabályozást végzünk, a pozícióhibára nagyobb hangsúlyt fektetünk, mint a sebességhibára ($\lambda > 1$ a (6.31) összefüggésben). Robotikai alkalmazásokban tipikusan $\lambda = 10$ értéket használjuk.

Válasszuk a maximális gyorsulást $a_{max} = 180^\circ/s^2$, a maximális sebességet $v_{max} = 180^\circ/s$ és a cél pozíciót $x_E = 720^\circ$ értékkel. A paraméterek ismeretében számoljuk ki a polinomiális pálya specifikus pontjait: $t_1, t_2, t_f, t_E, x_1, x_2$ felhasználva a 6.2.1. részben bemutatott összefüggéseket. Döntsük el, hogy három- vagy kétszakaszos pályára van szükség a (6.19)-es összefüggés segítségével.

A pályatervezés és pályakövetés akkor kell megvalósuljon, amikor a *Select_Lab_trackBar* csúszka *Tracking* állapotba kerül, és a szabályozás engedélyezve van. Ekkor legyen egy perszisztens időváltozó $time = 0$, mely segítségével a pályareferenciát ($a_{ref}, v_{ref}, p_{ref}$) megadjuk:

```
time = time + T;
if(three_sections == true){
    if(time < t_1){
        // Gyorsulás szakasz
    }
    else if(time < t_2){
        // Állandó sebesség szakasz
    }
    else if(time < t_f){
        // Lassulás szakasz
    }
    else{
        // Vége
        a_ref = 0;
        v_ref = 0;
        p_ref = x_E;
    }
}
else{
    if(time < t_1){
        // Gyorsulás szakasz
    }
    else if(time < t_2){
        // Lassulás szakasz
    }
    else{
        // Vége
        a_ref = 0;
```

```

    v_ref = 0;
    p_ref = x_E;
}

```

A beavatkozó jel kiszámításánál a K_P paramétert a $KP_trackBar$ csúszkáról olvassuk le. Kiszámoljuk az összetett hibát, a szabályozó jelet, és hozzáteesszük a holtávkompenzálást:

```

E = (v_ref - vel_k) + lambda * (p_ref - pos_k)
u_k = (a_ref + lambda * (v_ref - vel_k)
      + K_w * vel_k + K_P * E) / K_m;
if(u_k > 0)
    u_k += delta;
if(u_k < 0)
    u_k -= delta;

```

A program futtatása során vizsgáljuk meg a pályakövetési pontosságot különböző K_P értékekre.

6.4. Kérdések és feladatok

1. Számítsuk ki az előírt pályát (pozíció, sebesség és gyorsulás polinomok) a 6.2.1. fejezetben megadott kezdeti adatokkal úgy, hogy $x_E < x_0$.
2. Számítsuk ki az előírt pályát (pozíció, sebesség és gyorsulás polinomok) a 6.2.1. fejezetben megadott kezdeti adatokkal úgy, hogy $0 < v(t_0) < v_M$.
3. Hogyan módosíthatjuk a szabályozási algoritmust abban az esetben, amikor a holtávot nem ismerjük, és növelni szeretnénk a szabályozási pontosságot?

ROBOTMANIPULÁTOROK DIREKT GEOMETRIAI FELADATA

7.1. A gyakorlat célja

A fejezet célja a robotok direkt geometriai feladatának vizsgálata és implementálása általános esetre. Példaként egy ipari robot direkt geometriai feladatát oldjuk meg az első három szabadságfokra.

7.2. Elméleti bevezető

7.2.1 A direkt geometriai feladat általános megfogalmazása

A geometriai modell a robotok térbeli elhelyezkedését írja le egy adott időpillanatban, nem tárgyalja a robot mozgását.

A modell fontos alkalmazása, a robot végberendezése térbeli pozíciójának és orientációjának meghatározása, amennyiben ismerjük a robot csuklóinak a pozícióit, illetve a robot architektúráját (direkt geometriai feladat).

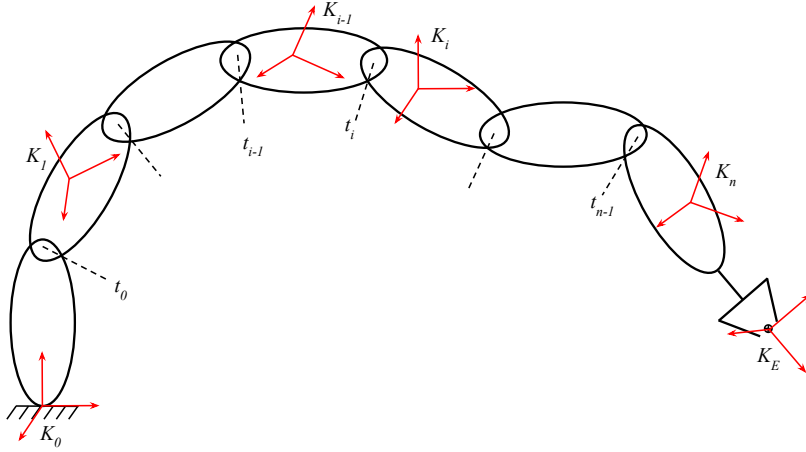
A végberendezésének térbeli pozícióját a szerszámközéppont (TCP) koordinátájával jellemezzük. A koordinátákat a robot bázisához rendelt koordináta-rendszerhez (K_0) képest adjuk meg. A végberendezéshez ugyancsak rendelünk egy koordináta-rendszert (K_E), aminek a origója a TCP. A végberendezés orientációját a K_0 és K_E közti rotációs mátrixszal definiáljuk.

Legyen egy robotmanipulátor (merev, nyílt láncú, elágazás nélküli mechanizmus), amelyben a szomszédos szegmenseket primitív csuklók (rotációs vagy translációs) kapcsolják össze.

A robotban jelölje n a csuklók számát ($n = DOF$). Ennek megfelelően a robotban $n + 1$ szegmens van, lásd 7.1. ábra. Az első szegmens rögzített a bázishoz.

Rendeljünk minden szegmenshez egy K_i koordináta-rendszert ($i = 0 \dots n$), lásd 7.1. ábra. A legelső, 0-ik szegmenshez rendelt koordináta-rendszer a K_0 bázis koordináta-rendszer.

Jelölje t_i a csuklók tengelyeit ($i = 0 \dots n - 1$). Az i -ik csukló csatolja az $i - 1$ -ik és az i -ik szegmenseket. Rotációs csukló esetén az elfordulás t_i körül történik, translációs csukló esetén az elmozdulás t_i mentén történik.



7.1. ábra. Robotmanipulátor elvi rajza a direkt geometriai feladathoz

Legyen a K_{i-1} és a K_i koordináta-rendszerek közötti homogén transzformáció mátrixa $T_{i-1,i}$.

A homogén transzformáció a K_0 és K_E kerek között:

$$T_{0,E} = T_{0,1} \cdot T_{1,2} \cdot \dots \cdot T_{i-1,i} \cdot \dots \cdot T_{n-1,n} \cdot T_{n,E} \quad (7.1)$$

A $T_{0,E}$ mátrix tartalmazza a TCP koordinátáit K_0 -ban, valamint az orientációt megadó rotációs mátrixot a K_0 és K_E koordináta-rendszerek között.

7.2.1 A Denavit–Hartenberg-konvenció

A Denavit–Hartenberg-konvenció eljárást ad a K_i koordináta-rendszerek hozzárendeléséhez a szegmensekhez, valamint az egymást követő koordináta-rendszerek közötti homogén transzformációs mátrixok ($T_{i-1,i}$) számításához is a csuklótípusok, illetve a robot geometriai paramétereinek függvényében.

Koordináta-rendszerek hozzárendelése a szegmensekhez:

A K_i koordináta-rendszert mindig a K_{i-1} -ik koordináta-rendszerhez képest definiáljuk az alábbi módon:

- A K_i koordináta-rendszer z_i tengelye megegyezik a t_i csuklóteneggellyel.
- Az x_i tengelyt úgy vesszük fel, hogy merőleges legyen z_{i-1} -re (a K_i -ben x_i merőleges z_i -re is).
- A K_i origója ott van, ahol x_i tengely metszi a t_i csuklóteneggelyt.

A koordináta-rendszerek y_i tengelyeinek iránya a jobbkéz-szabályból automatikusan következik.

A K_0 bázis koordináta-rendszert úgy vesszük fel, hogy a z_0 tengely egybeessen az első csukló tengelyével, t_0 -val. Az origót és az x_0 tengelyt a felhasználó adja meg, a konvenció nem írja elő ezek elhelyezését.

A *homogén transzformáció mátrixának* ($T_{i-1,i}$) *meghatározása*: a fenti konvenció szerinti koordináta-rendszer hozzárendelésekkel a homogén transzformáció mátrixa bármely K_{i-1} és K_i között 4 jellemzőtől függ.

A geometriai jellemzők meghatározásához vegyük fel az alábbi segédegyeneseket:

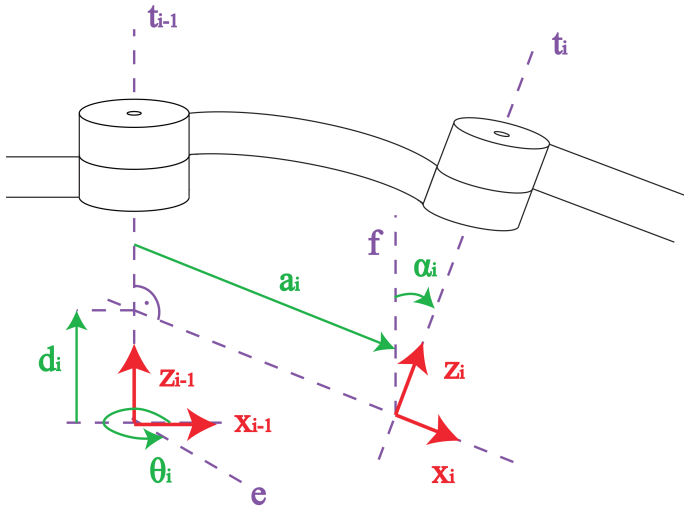
- Legyen az e egyenes, amely párhuzamos x_i -vel és áthalad K_{i-1} origóján.
- Legyen az f egyenes, amely párhuzamos z_{i-1} -el és áthalad K_i origóján.

A *Denavit–Hartenberg-jellemzők*, lásd 7.2. ábra:

- θ_i (*csuklószög*) – az a szög, amellyel a z_{i-1} körül el kell forgatni az x_{i-1} tengelyt, hogy az e egyenesbe essen.
- d_i (*csukló ofszet*) – az a távolság, amellyel el kell tolni az e egyenest z_{i-1} mentén, hogy tengelybe kerüljön x_i -vel.
- a_i (*szegmens hossz*) – az a távolság, amellyel a z_{i-1} és x_i tengely metszéspontját el kell tolni x_i mentén, hogy az a K_i origójába érkezzon.
- α_i (*szegmens csavarodás*) – az a szög, amellyel az f egyenest el kell forgatni x_i körül, hogy a z_i tengelybe essen.

Kivételes eset – t_{i-1} és t_i párhuzamosak:

- A Denavit–Hartenberg-konvenció alapján a K_i origóját bárhol felvehetjük a t_i -n. Ebben esetben úgy érdemes megválasztani az origó helyét, hogy a homogén transzformáció mátrixának minél egyszerűbb alakja legyen. Választhatunk például úgy, hogy a d_i paraméter 0 legyen.
- Az α_i paraméter 0.



7.2. ábra. Denavit–Hartenberg-paraméterek

A Denavit–Hartenberg-jellemzőkkel definiáltunk egy forgatás-eltolás-forgatás-eltolás szekvenciát, amivel eljuthatunk a K_{i-1} koordináta-rendszerből a K_i koordináta-rendszerbe:

$$T_{i-1,i} = Rot(z_{i-1}, \theta_i) Tran(z_{i-1}, d_i) Tran(x_i, a_i) Rot(x_i, \alpha_i) \quad (7.2)$$

$$\begin{aligned}
 &= \begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (7.3)
 \end{aligned}$$

ahol $\cos(\theta_i) = c_{\theta_i}$, $\sin(\theta_i) = s_{\theta_i}$, $\cos(\alpha_i) = c_{\alpha_i}$, $\sin(\alpha_i) = s_{\alpha_i}$.

Csuklóváltozók: a négy Denavit–Hartenberg-jellemző közül mindig egyiknek változhat az értéke a robot mozgása közben, ha primitív csuklókat feltételezünk. Rotációs csuklók esetében a változó θ_i (csukló szög), prizmatikus (transzlációs) csukló esetében pedig a_i (szegmens megnyúlás). Változó paraméterek a robot csuklóváltozói, amiket q_i -vel jelölünk. A teljes robotra

7.1. táblázat. *Robot geometriai leírása*

T	$\underline{\theta}$	$\underline{d}$	$\underline{a}$	$\underline{\alpha}$
$0 \rightarrow 1$	θ_1	d_1	a_1	α_1
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$i - 1 \rightarrow i$	θ_i	d_i	a_i	α_i
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$n - 1 \rightarrow n$	θ_n	d_n	a_n	α_n
$n \rightarrow E$	θ_E	d_E	a_E	α_E

a csuklóváltozók vektora:

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \\ \dots \\ q_n \end{pmatrix}, \quad (7.4)$$

ahol

$$q_i = \begin{cases} \theta_i, & \text{ha } i \text{ rotációs csukló,} \\ d_i, & \text{ha } i \text{ transzlációs csukló.} \end{cases} \quad (7.5)$$

A *robotok geometriai leírása* egy táblázat, amely tartalmazza egy robot összes szegmenséhez tartozó Denavit–Hartenberg-paramétert, lásd 7.1. táblázat. Megjelöljük a táblázatban azt is, hogy melyik paraméter a csuklóváltozó.

A legutolsó, $n \rightarrow E$ transzformáció leírja a végberendezés térbeli helyzetét az utolsó szegmenshez képest. Általában csak konstans elemeket tartalmaz, amelyek függenek a végberendezés geometriájától.

A *direkt geometriai feladat pontos meghatározása és megoldása*: adott egy merev, nyílt láncú, elágazás nélküli robotmanipulátor, amelynek ismerjük a nem változó Denavit–Hartenberg-paramétereit. Adott $\mathbf{q}$ csuklóváltozók vektora. Határozzuk meg a $T_{0,E}$ homogén transzformációs mátrixot.

A feladat megoldásához a (7.1) összefüggést kell alkalmazni, ahol (7.2) definiálja a $T_{i-1,i}$ transzformációt.

7.2. táblázat. *Robot geometriai leírása*

T	$\underline{\theta}$	$\underline{d}$	$\underline{a}$	$\underline{\alpha}$
$0 \rightarrow 1$	θ_1^*	D_1	A_1	$\pi/2$
$1 \rightarrow 2$	θ_2^*	0	A_2	0
$2 \rightarrow 3$	θ_3^*	0	A_3	0

7.3. A gyakorlat menete

A *gyakorlat célja*: az elméleti részben bemutatott Denavit–Hartenberg-konvenciót felhasználva készítsünk egy Python szkriptet, mely egy KUKA robot első három szabadságfokára megoldja a direkt geometriai feladatot.

1. A Python *NumPy* könyvtára segítségével hozzuk létre az elemi műveletekhez szükséges homogén transzformációs mátrixokat minden tengelyhez.
2. Készítsünk egy metódust, amely kiszámolja a Denavit–Hartenberg-táblázat alapján a csuklókhoz rendelt koordináta-rendszerek homogén transzformációs mátrixait.
3. Írjunk egy-egy metódust, amely visszaadja az utolsó szegmens végéhez rendelt koordináta-rendszer origójának térbeli koordinátáit.

Példa: az ábrán az ipari robotok első három szabadságfokánál a gyakorlatban elterjedt architektúra látható. A robot Denavit–Hartenberg-koordinátáit a 7.3. ábra, a Denavit–Hartenberg-paramétereit a 7.2. táblázat tartalmazza. A táblázatban a *-gal jelölt elemek a csuklóváltozók.

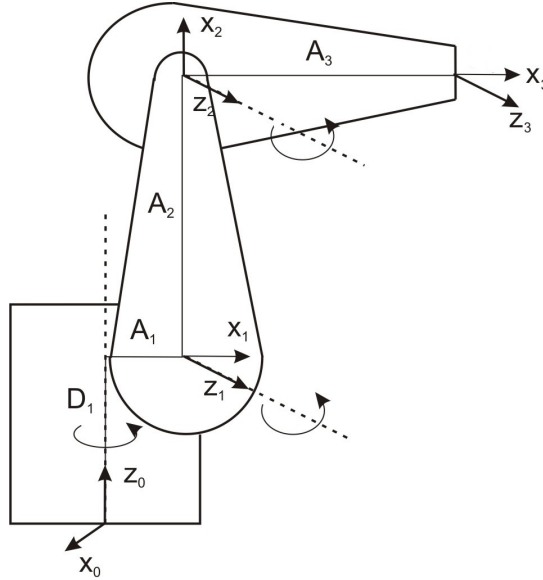
A táblázat alapján a $0 \rightarrow 1$ transzformációknak megfelelő Denavit–Hartenberg-mátrix:

$$T_{0,1} = Rot(z_0, \theta_1) Tran(z_0, d_1) Tran(x_1, a_1) Rot(x_1, \alpha_1)$$

$$T_{0,1} = \begin{pmatrix} c_{\theta_1} & 0 & -s_{\theta_1} & a_1 c_{\theta_1} \\ s_{\theta_1} & 0 & c_{\theta_1} & a_1 s_{\theta_1} \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

az $1 \rightarrow 2$ sornak megfelelő Denavit–Hartenberg-mátrix pedig:

$$T_{1,2} = Rot(z_1, \theta_2) Tran(z_1, d_2) Tran(x_2, a_2) Rot(x_2, \alpha_2)$$



7.3. ábra. Ipari robot első három csuklója ($\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = \pi/2$ csuklóváltozókra)

$$T_{1,2} = \begin{pmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & a_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & a_2 s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

ahol $\cos(\theta_1) = c_{\theta_1}$, $\sin(\theta_1) = s_{\theta_1}$, $\cos(\theta_2) = c_{\theta_2}$, $\sin(\theta_2) = s_{\theta_2}$.

A programot a `robot_geo.py` szkriptben készítjük el. A `robot_model` osztály tartalmazza a robot struktúráját: `_link_len` és `_joint_angle` a robot-hoz tartozó szegmens hossz és csuklóváltozó privát tömbök. Az `__init__` konstruktor segítségével adjunk értékeket a fenti tulajdonságoknak (a csuklóváltozók alapértelmezett értéke legyen 0).

Python osztályokban azon metódusokat, melyek nem használják fel az osztály paramétereit és egyéb metódusait (`self` tulajdonságon keresztül), statikusnak deklarálunk (`@staticmethod`).

Írjunk egy-egy statikus metódust a fok és radián mértékegységek közötti átalakításra (`_deg2rad`, `_rad2deg`).

Készítsük el az alap transzformációkért felelős statikus metódusokat. Az X tengely mentén való eltolás mátrixa:

$$Tran(x, a) = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A neki megfelelő metódus:

```
@staticmethod
def _translate_x(a: float) -> np.ndarray[float]:
    T = np.eye(4)
    T[0, 3] = a
    return T
```

Hasonlóan egészítsük ki az Y és Z tengely mentén való eltolás mátrixait számoló metódusokat.

Az X tengely körüli forgatás mátrixa:

$$Rot(x, \theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A neki megfelelő metódus:

```
@staticmethod
def _rotate_x(theta: float) -> np.ndarray[float]:
    T = np.eye(4)
    T[1, 1] = np.cos(theta)
    T[1, 2] = -np.sin(theta)
    T[2, 1] = np.sin(theta)
    T[2, 2] = np.cos(theta)
    return T
```

Vegyük észre, hogy a tömbök indexelése 0-tól kezdődik. Hasonlóan készítsük el az Y és Z tengelyek körüli forgatás mátrixait.

A *direct_kinematics* metódusban implementáljuk le a direkt geometriai feladatot. Az alapl műveleteket a *self* tulajdonságon keresztül érhetjük el, míg a mátrix szorzásra használhatjuk a *@* vagy a *numpy.matmul* metódust:

```
def direct_kinematics(self) -> np.ndarray[float]:
    tcp_pose_vector = np.array([0, 0, 0, 1])
```



```

Rz_1 = self._rotate_z(self._joint_angle[0])
...
return Rz_1 @ Tz_1 @ Tx_1 @ Rx_1 @ Rz_2 @ Tz_2 @ Tx_2
        @ Rx_2 @ Rz_3 @ Tz_3 @ Tx_3 @ Rx_3
        @ tcp_pose_vector

```

Próbáljuk ki az algoritmust $d_1 = 10$, $a_1 = 5$, $a_2 = 20$, $a_3 = 20$ cm hosszúságú szegmensekkel és $\theta_1 = 0^\circ$, $\theta_2 = -45^\circ$, $\theta_3 = -45^\circ$ csukló változókkal.

Megjegyzés: a Numpy trigonometrikus függvények radiánnal dolgoznak, ezért át kell alakítani az értékeket.

Az eredmény helyes megvalósítás esetén: $x = 19.142$, $y = 0$, $z = 44.142$ cm.

Kérdések és feladatok

1. Bővítsük ki a szkriptet egy olyan metódussal, amely az utolsó szegmens orientációját adja vissza Euler szögekben.
2. Mutassuk meg, hogy a homogén transzformációs mátrixok szorzással alkotott csoportja asszociatív.
3. Mutassuk meg, hogy a csupán rotációt tartalmazó homogén transzformációs mátrixok szorzással alkotott csoportja nem kommutatív.

ROBOTMANIPULÁTOROK INVERZ GEOMETRIAI FELADATA

8.1. A gyakorlat célja

A fejezet célja a robotok inverz geometriai feladatának vizsgálata és numerikus implementálása. Példaként egy ipari robot inverz geometriai feladatát oldjuk meg az első három szabadságfokra, összehasonlítva az analitikus eredményt a numerikus megközelítéssel.

8.2. Elméleti bevezető

8.2.1. Analitikus módszer

Inverz geometriai feladat esetében adott a robot végberendezésének pozíciója, illetve orientációja, keressük a

$$\mathbf{q} = (q_1 \ q_2 \ \dots \ q_n)^\top \quad (8.1)$$

csuklóváltozókat. Ugyancsak ismertnek tekintjük a robot konstans Denavit–Hartenberg-paramétereit.

A végberendezés előírt pozícióját a szerszámközepont (TCP) koordinátaival jellemezzük:

$$\mathbf{p}_d = (x_d \ y_d \ z_d)^\top \quad (8.2)$$

A

végberendezés előírt orientációját a végberendezéshez rendelt koordináta-rendszer elforgatásával jellemezzük a bázis koordináta-rendszerhez képest. Legyen az előírt elforgatáshoz tartozó rotációs mátrix:

$$\mathbf{R}_d = (\mathbf{i}_d \ \mathbf{j}_d \ \mathbf{k}_d) \quad (8.3)$$

Az előírt pozíció és orientáció függvényében megfogalmazhatjuk az előírt homogén transzformáció mátrixát a bázis koordináta-rendszer és a végberendezéshez rendelt koordináta-rendszer között:

$$T_d = \begin{pmatrix} R_d & \mathbf{p}_d \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (8.4)$$

Az inverz geometriai feladat *általános megoldásához* először felírjuk a Denavit–Hartenberg-konvenció alapján a homogén transzformáció mátrixát a csuklózváltozók függvényében. A mátrixban a csuklózváltozókat ismeretlennek tekintjük:

$$T_{0,n}(\mathbf{q}) = T_{0,1}(q_1) \cdot T_{1,2}(q_2) \cdot \dots \cdot T_{i-1,i}(q_i) \cdot \dots \cdot T_{n-1,n}(q_n) \quad (8.5)$$

Egyenlővé tesszük a (8.4) és (8.5) mátrixokat:

$$T_{0,n}(\mathbf{q}) = T_d \quad (8.6)$$

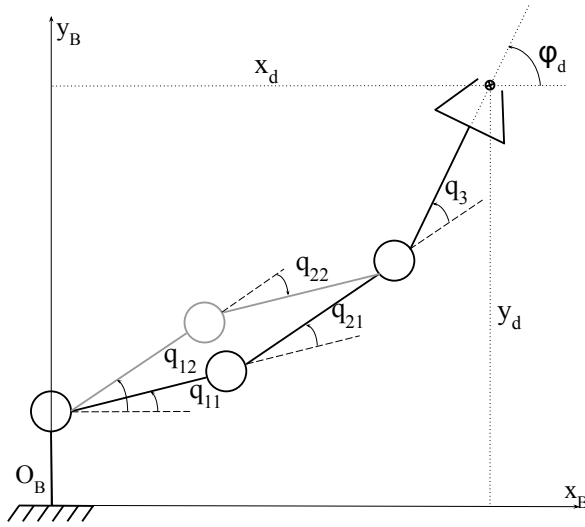
A feladat megoldásához a két mátrix megfelelő elemeit kell megfeleltetni. A mátrixegyenletből (legalább) n olyan egyenletet emelünk ki, amelyek függnek a q_i csuklózváltozóktól. Az egyenletrendszerben meg kell jelenjen az összes csuklózváltozó ($i = 1 \dots n$). Az egyenlet megoldása szolgáltatja a keresett $\mathbf{q}$ csuklókoordinátákat.

Hat szabadságfokú robot esetében általában olyan hat egyenletből álló egyenletrendszert kell megoldani, amelyek trigonometrikus kifejezéseket tartalmazhatnak. Problémát jelent, hogy általában a kapott egyenletrendszer nemlineáris, explicit megoldás nehezen kapható. Ilyenkor numerikus módszereket alkalmazhatunk a nemlineáris egyenletrendszer megoldására.

Azt is figyelembe kell venni, hogy egy adott inverz geometriai feladatnak több megoldása létezhet. Például a 8.1. ábrán bemutatott síkban mozgó robot esetén $\mathbf{q}_1 = (q_{11} \ q_{21} \ q_3)^\top$ és $\mathbf{q}_2 = (q_{12} \ q_{22} \ q_3)^\top$ is megoldása az $(x_d \ y_d \ \varphi_d)^\top$ által definiált inverz geometriai feladatnak, ahol φ_d szög a Z tengely körüli forgatás rotációs mátrixát határozza meg. Ilyen esetben a megoldás kiválasztásánál figyelembe vehetjük például a munkatérben elhelyezkedő akadályokat. Cél, hogy a robot mozgás közben minél távolabb legyen az akadályoktól az ütközések elkerülése miatt.

8.2.2. Inverz geometriai feladat megoldása dekompozícióval

Adott egy hat szabadságfokú ipari robot ($\mathbf{q} = (q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6)^\top$). Általában az első három csuklózváltozóval ($q_1, \ q_2, \ q_3$) a robot végberendezésének térbeli helyzetét állíthatják be (robotkar), a negyedik, ötödik, hatodik



8.1. ábra. *Inverz geometria síkban mozgó robotnál*

csuklóváltó (q_4 , q_5 , q_6) a végberendezés orientációjának beállításáért felelős (kézcscukló).

Oldjuk meg az inverz geometriai feladatot egy olyan robotra, amelynek az utolsó három csukló forgástengelye egy ponton megy át (Euler-robotcsukló, gömbcsukló). Legyen ez a pont az Euler-robotcsukló közép-pontja (P_C).

Az inverz geometriai feladat megoldásánál az alábbi lépéseket követjük:

1. Az előírt TCP pozíció alapján meghatározzuk P_C térbeli pozícióját.
2. Az ismert P_C térbeli pozíció alapján meghatározzuk az első három csuklóváltót (q_1 , q_2 , q_3).
3. Az ismert első három csuklóváltó és az előírt orientáció alapján meghatározzuk a maradék három csuklóváltót (q_4 , q_5 , q_6).

1. P_C pozíciója: az utolsó három koordináta-rendszer esetén a K_4 és K_5 koordináta-rendszereknek ugyanaz az origójuk, a K_6 el van tolva a d_6 csukló ofszet paraméterrel, lásd 8.2. ábra. Tehát a P_C és TCP közötti távolságot a d_6 adja.

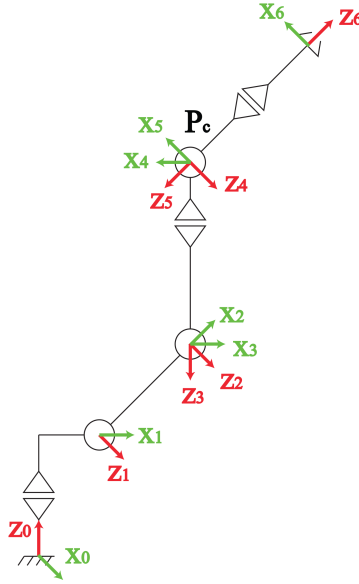
Feltételezzük, hogy a z_6 tengely (utolsó csuklótengely) megegyezik a végberendezés z tengelyével. Így z_6 egybe kell essen az R_d mátrix $\mathbf{k}_d$ vektorával (a végberendezés előírt z tengelyének iránya). Ennek megfelelően P_C

pozícióvektora a bázis koordináta-rendszerben:

$$\mathbf{p}_c = \mathbf{p}_d - d_6 \mathbf{k}_d \quad (8.7)$$

Példa:

$$\begin{aligned} \mathbf{p}_d &= (3 \ 3 \ 0)^\top \\ d_6 &= 0.5 \\ \mathbf{k}_d &= (1 \ 0 \ 0)^\top \\ \Rightarrow \mathbf{p}_c &= (3 \ 3 \ 0)^\top - 0.5(1 \ 0 \ 0)^\top = (2.5 \ 3 \ 0)^\top \end{aligned}$$



8.2. ábra. Koordináta-rendszerek a Denavit–Hartenber-konvenció alapján

2. *Előírt pozíció biztosítása*: Legyenek az 1. lépésben meghatározott P_c pont koordinátái:

$$\mathbf{p}_c = (x_{dc} \ y_{dc} \ z_{dc})^\top \quad (8.8)$$

Ennek függvényében határozzuk meg az első három csuklózváltozót.

Általános esetben a megoldáshoz az alábbi egyenletből indulhatunk ki (feltételezve, hogy a K_4 -es koordináta-rendszer origója P_c):

$$\mathbf{p}_{03}(q_1, q_2, q_3) = \mathbf{p}_c, \quad (8.9)$$

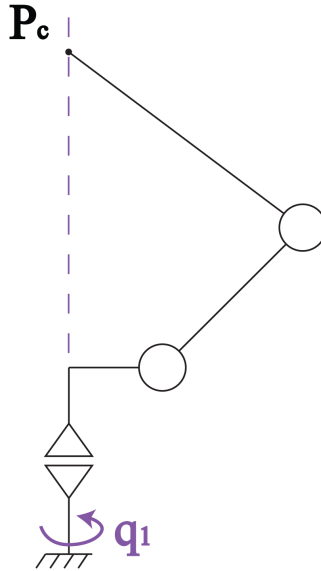
ahol $\mathbf{p}_{03}$ a $T_{03} = T_{01}(q_1)T_{12}(q_2)T_{23}(q_3)$ homogén transzformáció mátrixban a pozícióvektor.

Számos robotarchitektúra esetén, lásd például 8.2. ábra, direkt trigonometriai számításokkal a q_1 , q_2 , q_3 csuklóváltozók meghatározhatóak.

Az első csuklóváltozó az alábbi egyenletből határozható meg, lásd 8.4. ábra:

$$q_1 = \text{atan2}(y_{dc}, x_{dc}) \quad (8.10)$$

Ugyancsak a 8.4. ábrából látszik, hogy $q_1 = \text{atan2}(y_{dc}, x_{dc}) + \pi$ is eleget tesz az adott koordináta-feltételeknek.



8.3. ábra. Robotkar szinguláris konfigurációban

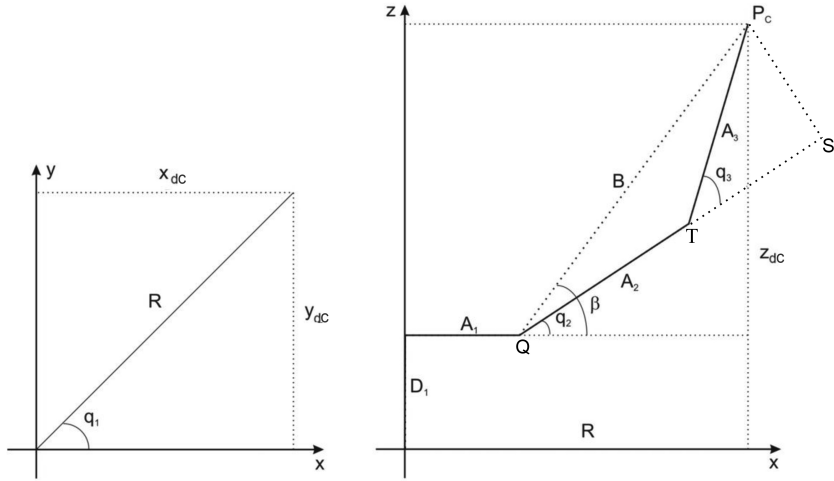
Abban az esetben, amikor $x_{dc} = y_{dc} = 0$, a q_1 szög nem számítható. Ez egy szinguláris konfiguráció, bármilyen $q_1 \in [-\pi, \pi]$ értékre elérhető az $y_{dc} = x_{dc} = 0$ konfiguráció, lásd 8.3. ábra.

Ugyancsak meghatározható a robot vízszintes kinyúlása:

$$R = \sqrt{x_{dc}^2 + y_{dc}^2} \quad (8.11)$$

A q_2 , q_3 számításához z_{dc} koordinátát is figyelembe kell venni a 8.4. ábra alapján.

$$B = \sqrt{(R - A_1)^2 + (z_{dc} - D_1)^2} \quad (8.12)$$



8.4. ábra. Inverz geometria számítása a robot bázis koordináta-rendszerében

$$\beta = \text{atan2}(z_{dc} - D_1, R - A_1) \quad (8.13)$$

A QRP_c háromszögben alkalmazhatjuk a koszinusz tételt a $\pi - q_3$ szögre. Felhasználva, hogy $\cos(\pi - \alpha) = -\cos(\alpha)$, következik:

$$\cos(q_3) = \frac{B^2 - A_2^2 - A_3^2}{\underbrace{2A_2A_3}_{C_q}} \quad (8.14)$$

Innen számíthatjuk a q_3 csuklózóváltozót:

$$q_3 = \text{atan2}\left(\pm C_q, \sqrt{1 - C_q^2}\right) \quad (8.15)$$

Látszik, hogy az egyenletnek két megoldása van, hasonlóan a 8.1. ábra első két csuklójához.

A q_2 csuklózóváltozót a β szög ismertében számíthatjuk a QSP_c háromszögben:

$$q_2 = \beta - \text{atan2}(A_2 + A_3\cos(q_3), A_3\sin(q_3)) \quad (8.16)$$

A q_2 -re is két megoldást kapunk a (8.15) alapján kapott két q_3 megoldásnak megfelelően.

3. *Előírt orientáció biztosítása*: írjuk fel a robot direkt geometriai feladatát az alábbi formában:

$$T_{03}(q_1, q_2, q_3) \cdot T_{36}(q_4, q_5, q_6) = T_d \quad (8.17)$$

Mivel q_1 -t, q_2 -t, q_3 -t már meghatároztuk a második lépésben, az egyenletet az alábbi alakra hozzuk:

$$T_{36}(q_4, q_5, q_6) = T_{03}^{-1}(q_1, q_2, q_3)T_d, \quad (8.18)$$

ahol

$$T_{03} = \begin{pmatrix} R_{03}(q_1, q_2, q_3) & \mathbf{p}_{03}(q_1, q_2, q_3) \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (8.19)$$

$$T_{36} = \begin{pmatrix} R_{36}(q_4, q_5, q_6) & \mathbf{p}_{36}(q_4, q_5, q_6) \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (8.20)$$

A q_4, q_5, q_6 csuklóparaméterek a (8.19) egyenletből határozhatóak meg, felhasználva a rotációs mátrixokat:

$$R_{36}(q_4, q_5, q_6) = R_{13}^\top(q_1, q_2, q_3)R_d \quad (8.21)$$

Az $R_{13}(q_1, q_2, q_3)$ tagja az egyenletnek ismert, a q_1, q_2, q_3 csuklókoordinátákat a 2. lépésben már meghatároztuk. A mátrixegyenletben az elemek összehasonlításával számíthatók az ismeretlen csuklózváltozók, hasonló módon, mint az Euler-szögek esetében.

8.2.3. Inverz geometriai feladat numerikus megoldása Jacobi-mátrixszal

Tekintsünk egy robotkart $\mathbf{q}$ pillanatnyi csuklózváltozókkal (az egyszerűség kedvéért csupán rotációs csuklókkal). A robot végberendezésének helyzetét megkapjuk a robot geometriáját leíró egyenletek segítségével:

$$\mathbf{p}(\mathbf{q}) = \mathbf{x},$$

ahol $\mathbf{p}(\mathbf{q})$ a robot T_{03} homogén transzformációs mátrixában a pozíció vektor.

A cél azon $\mathbf{q}^*$ csuklózváltozók meghatározása, amelyekre a robot végberendezése az előírt ($\mathbf{x}^*$)

$$\mathbf{p}(\mathbf{q}^*) = \mathbf{x}^*$$

referencia helyzetbe kerül.

Képezzük a robot geometria Taylor-sorát a $\mathbf{q}^*$ csuklózváltozók mentén:

$$\mathbf{p}(\mathbf{q}) \Big|_{\mathbf{q}=\mathbf{q}^*} = \mathbf{p}(\mathbf{q}^*) + \frac{\partial \mathbf{p}}{\partial \mathbf{q}} \Big|_{\mathbf{q}=\mathbf{q}^*} (\mathbf{q} - \mathbf{q}^*) + \underbrace{\frac{\partial^2 \mathbf{p}}{\partial^2 \mathbf{q}} \Big|_{\mathbf{q}=\mathbf{q}^*} (\mathbf{q} - \mathbf{q}^*)^2 + \dots}_{O(\mathbf{q})}$$

A magasabb rendű parciális deriváltakat elhanyagolhatjuk $O(\mathbf{q}) \approx 0$.

A robot geometria parciális deriváltja a csuklózváltozók szerint maga a Jakobi-mátrix. Legyen $\Delta \mathbf{q} := \mathbf{q}^* - \mathbf{q}$ (elemi csukló elmozdulás). Ekkor a fenti sorba fejtetten kifejezés felírható a

$$\mathbf{p}(\mathbf{q}) - \mathbf{x}^* = -J \Delta \mathbf{q},$$

ahol $J = \frac{\partial \mathbf{p}}{\partial \mathbf{q}}$ a robot lineáris sebességkomponenseit meghatározó Jacobi-mátrixa. Amennyiben J négyzetes mátrix ($\dim \mathbf{p} = \dim \mathbf{q}$) és $\det(J) \neq 0$ (azaz a tér dimenziója megegyezik a szabadságfokkal, és $\mathbf{q}^*$ nem szinguláris konfiguráció) a csuklóállapot lépést kifejezhetjük, mint

$$\Delta \mathbf{q} = -J^{-1}(\mathbf{p}(\mathbf{q}) - \mathbf{x}^*).$$

Példa: amennyiben $\mathbf{p} = (x \ y \ z)^\top$ és $\mathbf{q} = (q_1 \ q_2 \ q_3)^\top$

$$J = \begin{pmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} \end{pmatrix}$$

Ha J nem négyzetes (vagy szinguláris), de lineárisan független oszlopokat tartalmaz, akkor a Moore–Penrose-féle pseudo-inverz mátrixot alkalmazhatjuk:

$$\Delta \mathbf{q} = -J^\dagger (\mathbf{p}(\mathbf{q}) - \mathbf{x}^*),$$

ahol $J^\dagger = (J^\top J)^{-1} J^\top$.

Kaptunk egy összefüggést, mely megad egy elemi csuklózváltozó elmozdulást, amely közelíti a végberendezést az előírt helyzethez.

A numerikus módszer megvalósítása: a közelség méréséhez használjunk egy $\|\cdot\|$ vektor normát mint metrikát és egy N maximális iteráció számot. Az algoritmusunk célja, hogy az eredményt ($\mathbf{q}^*$) legalább ϵ pontossággal megkapjuk, vagyis $\|\mathbf{p}(\mathbf{q}^*) - \mathbf{x}^*\| \leq \epsilon$ az algoritmus kilépési feltétele.

Ekkor a Jakobi inverz-alapú iteratív algoritmust az inverz geometriai feladat megoldásához a következő lépésekkel fogalmazhatjuk meg:

0. Adott az aktuális csukláváltozó $\mathbf{q}_0$, az elvárt $\mathbf{x}^*$ végberendezés helyzet, N maximális iteráció és maximális iteráció hiba ϵ .
1. Legyen $\mathbf{q} = \mathbf{q}_0$, és $idx = 1$.
2. Adjuk meg a robotkar végberendezésének helyzetét a direkt geometriai egyenletek segítségével $\mathbf{x} = \mathbf{p}(\mathbf{q})$.
3. Adjuk meg a robotkar Jakobi-mátrixát $J(\mathbf{q})$.
4. Ha szinguláris konfigurációt kapunk, vége, máskülönben számoljuk ki a $\Delta\mathbf{q}$ elemi csukló lépést a végső pozíció irányába.
5. Frissítsük a csukláváltozót $\mathbf{q} := \mathbf{q} + \Delta\mathbf{q}$.
6. Számoljuk ki az eredmény közelségét $d = \|\mathbf{x} - \mathbf{p}(\mathbf{q})\|$.
7. Ha $d \geq \epsilon$, akkor $idx := idx + 1$, máskülönben vége.
8. Ha $idx \leq N$ ugorjunk a 2-ik lépéshez, máskülönben vége.

Ha a 7-es pontból léptünk ki, akkor megtaláltuk az eredményt ($\mathbf{q}^*$) legfeljebb ϵ pontossággal. Ha a 8-as pontból léptünk ki, akkor nem értük el a kívánt pontosságot, az előírt iterációs szám alatt.

8.2.4. Jakobi-mátrix számítása

A J mátrix analitikus számításához legyen a bázis vektor $\mathbf{b} = (0 \ 0 \ 0 \ 1)^\top$, ekkor három szabadságfok esetén a végberendezés pozícióját megkaphatjuk a

$$\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = T_{03}\mathbf{b}$$

összefüggéssel, ahol $T_{0,3} = T_{0,1}T_{1,2}T_{2,3}$. A Jakobi-mátrix fellelhető mint

$$J = \frac{d\mathbf{x}}{d\mathbf{q}} = \frac{dT_{0,3}\mathbf{b}}{d\mathbf{q}} = \underbrace{\frac{dT_{0,1}}{d\mathbf{q}}T_{1,2}T_{2,3}\mathbf{b}}_{J_0} + \underbrace{T_{0,1}\frac{dT_{1,2}}{d\mathbf{q}}T_{2,3}\mathbf{b}}_{J_1} + \underbrace{T_{0,1}T_{1,2}\frac{dT_{2,3}}{d\mathbf{q}}\mathbf{b}}_{J_2}.$$

A $\frac{dT_{i,j}}{d\mathbf{q}}$ tenzorok számolását le tudjuk egyszerűsíteni. Emlékezzünk vissza a robotunk Denavit–Hartenberg-táblázatára. Minden sorban egyetlen csukló-változó szerepelt a Z-tengely rotációjánál, azaz a változók $q_i = \theta_i$, és

$$\begin{aligned} T_{0,1} &= Rot(z_0, q_1) Tran(z_0, d_1) Tran(x_1, a_1) Rot(x_1, \alpha_1) \\ T_{1,2} &= Rot(z_1, q_2) Tran(z_1, d_2) Tran(x_2, a_2) Rot(x_2, \alpha_2) \\ T_{2,3} &= Rot(z_2, q_3) Tran(z_2, d_3) Tran(x_3, a_3) Rot(x_3, \alpha_3) \end{aligned}$$

Ezért a Jacobi-mátrix oszlopait fel tudjuk írni, mint

$$\begin{aligned} J_0 &= \frac{dRot(z_0, q_1)}{dq_1} Tran(z_0, d_1) Tran(x_1, a_1) Rot(x_1, \alpha_1) T_{1,2} T_{2,3} \mathbf{b} \\ J_1 &= T_{0,1} \frac{dRot(z_1, q_2)}{dq_2} Tran(z_1, d_2) Tran(x_2, a_2) Rot(x_2, \alpha_2) T_{2,3} \mathbf{b} \\ J_2 &= T_{0,1} T_{1,2} \frac{dRot(z_2, q_3)}{dq_3} Tran(z_2, d_3) Tran(x_3, a_3) Rot(x_3, \alpha_3) \mathbf{b} \end{aligned}$$

Példa a számolások során felmerülő részeredményre:

$$\frac{dRot(z_0, q_1)}{dq_1} = \begin{pmatrix} -\sin(q_1) & -\cos(q_1) & 0 & 0 \\ \cos(q_1) & -\sin(q_1) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

A KUKA robot első három szabadságfoka esetén (lásd 7.2.1. alfejezet) a pozícióvektor felírható a csuklóváltozók szerint:

$$\mathbf{p}(\mathbf{q}) = \begin{pmatrix} [a_1 + a_2 \cos(q_2) + a_3 \cos(q_2 + q_3)] \cos(q_1) \\ [a_1 + a_2 \cos(q_2) + a_3 \cos(q_2 + q_3)] \sin(q_1) \\ a_2 \sin(q_2) - a_3 \sin(q_2 + q_3) + d_1 \end{pmatrix},$$

A Jacobi-mátrix:

$$J(\mathbf{q}) = \begin{pmatrix} -[a_1 + a_2 c_{q_2} + a_3 c_{q_{23}}] s_{q_1} & -[a_2 s_{q_2} + a_3 s_{q_{23}}] c_{q_1} & -a_3 s_{q_{23}} c_{q_1} \\ [a_1 + a_2 c_{q_2} + a_3 c_{q_{23}}] c_{q_1} & -[a_2 s_{q_2} + a_3 s_{q_{23}}] s_{q_1} & -a_3 s_{q_{23}} s_{q_1} \\ 0 & a_2 c_{q_2} - a_3 c_{q_{23}} & -a_3 c_{q_{23}} \end{pmatrix},$$

ahol $c_{q_1} = \cos(q_1)$, $s_{q_1} = \sin(q_1)$, $c_{q_2} = \cos(q_2)$, $c_{q_{23}} = \cos(q_2 + q_3)$, $s_{q_2} = \sin(q_2)$, $s_{q_{23}} = \sin(q_2 + q_3)$.

A Jacobi-mátrix determinánsa pedig:

$$j(\mathbf{q}) = \det J(\mathbf{q}) = a_2 a_3 [a_1 + a_2 \cos(q_2) + a_3 \cos(q_2 + q_3)] \sin(q_3).$$

8.3. A gyakorlat menete

A *gyakorlat célja*: implementáljuk le a KUKA robot első három szabadságfokára az inverz geometriai feladat numerikus megoldását Jakobi-inverzszel, hasonlítsuk össze a kapott eredményt az analitikus eredménnyel. A megoldás lépései:

1. Írjunk egy metódust, mely módosítja a robotkar osztály csuklózváltozóit.
2. Általánosítsuk a direkt geometria feladat metódus bemenetét.
3. Írjunk egy metódust a Z-tengely mentén való elforgatási homogén transzformációs mátrix deriváltjának kiszámolására.
4. Számoljuk ki a Jakobi-mátrixot.
5. Írjuk meg az iteratív algoritmust.
6. Hasonlítsuk össze az iteratív eredményeket az analitikus eredményekkel.

Jelenleg a robotkar osztály csuklózváltozó tulajdonságait csak példányosításakor tudjuk megadni, ami a direkt geometriánál rendben volt, viszont nem szeretnénk több példányt létrehozni minden egyes inverz geometria tesztre.

```
def set_joints(self, joint1: float, joint2: float,
               joint3: float, unit='deg') -> None:
```

A csuklózváltozókat megadhatjuk radiánban vagy fokban (kezeljük le a metódusban), az alapértelmezett mértékegység legyen fok.

Láttuk az algoritmus általános leírásában (2. és 6. lépések), hogy szükséges többször megoldani a direkt geometriai feladatot, más-más csuklózváltozó értékekre. Ezért hozzunk létre egy új metódust, amely nem az osztály tulajdonságából számol végberendezés-helyzetet, hanem a paramétereiből:

```
def _direct_kinematics_arb_angle(self,
                                  joints: np.ndarray[ float ])
    -> np.ndarray[ float ]:
```

Az eljárás megegyezik a direkt geometriai feladat esetében alkalmazott szoftveres megvalósítással.

Az alábbi metódussal számoljuk ki a Z-tengely mentén való elforgatási homogén transzformációs mátrixszának deriváltját.

```
def _drotate_z(phiz_rad: float)
    -> np.ndarray[ float ]:
```

Az összefüggéseket analitikus úton megkaphatjuk skaláris deriválttal.

Írjuk meg a Jakobi-mátrix metódust a fent szemléltetett módszer alapján:

```
def _jacobi_mat(self, joints: np.ndarray[ float ])
    -> np.ndarray[ float ]:
```

Az iteratív algoritmust leíró metódus fejléce:

```
def inverse_kinematics_numeric(self,
    reference_tcp: np.ndarray[ float ],
    max_iter: int = 1000,
    tolerance: float = 1e-6,
    bprint: bool = False)
    -> [np.ndarray[ float ], int ]:
```

Itt *reference_tcp* a végberendezés elvárt pozícióvektora ($\mathbf{x}^*$, 3 elemű valós vektor), *max_iter* a maximális iterációs szám adott alapértelmezett értékkel, *tolerance* a metrika kiszámításához használt tűréshatár, *bprint* pedig azt mondja meg, hogy akarunk üzeneteket íratni vagy sem.

A szkript elején állítsunk egy hiba változót nagy számra (pl. *error* = 100). Az iterációt vigyük 1-től a megadott maximális iterációs számig. A ciklusban kövessük a fent megadott lépéseket.

Mivel a robot architektúra adott, nem kell leellenőrizni, hogy a Jakobi-mátrix négyzetes vagy sem. Az inverz létét viszont le kell ellenőrizni (*np.linalg.det()* a determináns számoló metódus). Ha a determináns nulla és *bprint* igaz, írjuk ki, hogy null a determináns. Térítsük vissza a kiindulási csuklóállapotokat és egy 0 értéket.

Ha létezik az inverz, a *np.linalg.inv()* metódussal számoljuk ki. Egy *iJx* mátrix-vektor szorzást az *iJ.dot(x)* metódussal végzünk el. A normát megkapjuk, mint *np.linalg.norm()*, ha nem mondjuk meg a norma típusát, az alapértelmezett 2-es normát használja, azaz a Descartes-féle távolságot.

Ha a hiba nem megfelelően kicsi, de az iteráció lejárt és *bprint* aktív, írjuk ki, hogy az adott számú lépés alatt mekkora hibával oldottuk meg a feladatot. Térítsük vissza az eredő csuklózváltozókat és egy 0 értéket.

Ha a hiba kritérium teljesült és *bprint* aktív, írassuk ki az elért hibát. Térítsük vissza az eredő csuklózváltozókat és 1 értéket.

A numerikus módszer túrése legyen $1E - 11$ cm. Teszteljük le az algoritmusunkat 0° , 45° , 45° kezdeti feltétellel, és keressük meg a 0, 10, 10 cm végberendezés helyzetnek megfelelő csuklóváltozókat. Helyes működés esetén a visszatérési érték 90° , $\pm 82.819^\circ$, $\pm 165.638^\circ$.

Kérdések és feladatok

1. Milyen robotkarok esetében különbözik a munkatér dimenziója a szabadságfoktól?
2. Egy 2DOF RR robot esetén milyen csuklóváltozók esetén lesz $\det(J) = 0$?
3. Vezessük le a Jacobi-mátrix számolását a 8.2.1. fejezetben bemutatott eljárással.

ROBOTMANIPULÁTOROK KINEMATIKÁJA

9.1. A gyakorlat célja

A robotmanipulátorok kinematikai alapfogalmainak áttekintése, az inverz kinematikai feladat implementálása, a robot szinguláris konfigurációinak feltérképezése.

9.2. Elméleti bevezető

A robotok kinematikai modellezése során a robot sebességét vizsgáljuk. Összefüggést keresünk a robot csuklósebességei és a végberendezés sebességkomponensei között.

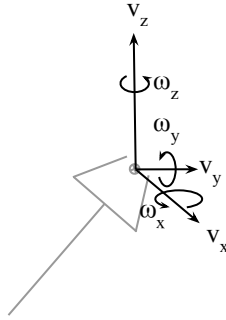
A végberendezés lineáris sebességvektorának komponenseit a bázis koordináta-rendszerben egy háromelemű vektorban adjuk meg:

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \quad (9.1)$$

A szögsebesség meghatározásához a robot csuklótengelyei körül történő forgó mozgást feltételezünk. A szögsebesség vektor egyenlő a szögváltozás (skalár) és a csuklótengely bázis koordináta-rendszerben felírt egységvektora (három komponensű vektor) szorzatával. Például ha a csuklószög θ , és a forgás egy koordináta-rendszer z tengelye körül történik, aminek az egységvektora $\mathbf{k} = (0 \ 0 \ 1)^\top$, akkor a szögsebesség $\omega = \dot{\theta}\mathbf{k}$.

A végberendezés szögsebesség komponenseit szintén a robot bázis koordináta-rendszerében írjuk fel:

$$\omega = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}, \quad (9.2)$$



9.1. ábra. Sebesség és szögsebesség komponensek

ahol $\omega_x, \omega_y, \omega_z$ a szögsebesség komponensek a bázis koordináta-rendszerben.

Legyen a robot csuklóváltozóinak a vektora $\mathbf{q}$. A csuklósebességek mértékének vektora

$$\dot{\mathbf{q}} = (\dot{q}_1 \ \dot{q}_2 \ \dots \ \dot{q}_n)^\top, \quad (9.3)$$

ahol n a robot szabadságfoka.

Az összefüggést a végberendezés sebességvektor komponensei és a csuklósebességek között mátrix formában keressük:

$$\begin{pmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{pmatrix} = J \dot{\mathbf{q}} \quad (9.4)$$

J a robot *Jacobi-mátrixa*, amelynek mérete $6 \times n$.

Látszik, hogy a Jacobi-mátrix i -ik oszlopa a végberendezés sebesség és szögsebesség vektorainak komponenseit adja abban az esetben, amikor $\dot{q}_i = 1$ és az összes többi csuklósebesség 0.

Direkt sebességkinematikai feladat: adott a robot csuklósebességeinek vektora: $\dot{\mathbf{q}}$. Keressük a $\dot{\mathbf{q}}$ csuklósebességeknek megfelelő végberendezés sebességvektort $(\mathbf{v}^\top \ \boldsymbol{\omega}^\top)^\top$.

Amennyiben ismert a robot Jacobi-mátrixa, a (9.4) összefüggéssel a végberendezés sebességének komponensei meghatározhatóak.

9.2.1 A Jacobi-mátrix számítása

A Jacobi-mátrix felírásához külön tárgyaljuk a lineáris sebességeket és csuklósebességeket.

$$\begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix} = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \dot{\mathbf{q}} \quad (9.5)$$

ahol J_v , J_ω mátrixok méretei $3 \times n$.

J_v meghatározása: a Jacobi-mátrix lineáris sebességeket leíró komponense meghatározásához felírjuk a robot T_{0n} homogén transzformációs mátrixát a $\mathbf{q}$ csuklózváltozók függvényében:

$$T_{0n} = \begin{pmatrix} R_{0n}(\mathbf{q}) & \mathbf{p}_{0n}(\mathbf{q}) \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (9.6)$$

A $\mathbf{v}$ végberendezés sebességvektor komponensei a bázis-koordináta-rendszerben a végberendezés térbeli helyzetét leíró $\mathbf{p}_{0n}$ vektor koordinátáinak az időbeli deriváltjai:

$$\mathbf{v} = \dot{\mathbf{p}}_{0n}(\mathbf{q}) = \begin{pmatrix} \dot{p}_x(\mathbf{q}) \\ \dot{p}_y(\mathbf{q}) \\ \dot{p}_z(\mathbf{q}) \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \cdots & \frac{\partial p_x}{\partial q_n} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \cdots & \frac{\partial p_y}{\partial q_n} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \cdots & \frac{\partial p_z}{\partial q_n} \end{pmatrix}}_{J_v} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{pmatrix} \quad (9.7)$$

Tehát a Jacobi-mátrix lineáris sebességeket leíró komponense elemei a $\mathbf{p}_{0n}$ pozícióvektor elemeinek parciális deriváltjai a q_i csuklózváltozók szerint, $i = 1, 2, \dots, n$ (lásd 8.2.1. alfejezet).

J_ω meghatározása: a Jacobi-mátrix szögsebességeket leíró komponense esetében meg kell különböztetnünk a translációs csuklókat és a rotációs csuklókat. A translációs csuklók nem generálnak szögsebesség-komponenst a tengelyük mentén. Vezessük be az alábbi logikai függvényt:

$$\rho_i = \begin{cases} 0, & \text{ha } i \text{ translációs csukló} \\ 1, & \text{ha } i \text{ rotációs csukló} \end{cases} \quad (9.8)$$

A Jacobi-mátrix szögsebességeket leíró komponense meghatározásához felírjuk az i -ik csukló robot T_{0i} homogén transzformációs mátrixát:

$$T_{0i} = \begin{pmatrix} R_{0i}(q) & \mathbf{p}_{0i}(\mathbf{q}) \\ \mathbf{0}^\top & 1 \end{pmatrix}, \quad (9.9)$$

ahol

$$R_{0i} = \begin{pmatrix} \mathbf{i}_{0i} & \mathbf{j}_{0i} & \mathbf{k}_{0i} \end{pmatrix} \quad (9.10)$$

az i -ik szegmenshez rendelt koordináta-rendszer elforgatását jellemző rotációs mátrix a bázis koordináta-rendszerhez képest.

A Denavit–Hartenberg-konvenció alapján az i -ik csukló elfordulása (elmozdulása) a $\mathbf{k}_{0i}$ egységvektor által definiált tengely körül (mentén) történik. Tehát a rotációs csuklók szögsebesség-komponensei a bázis koordináta-rendszerben $\dot{q}_i \mathbf{k}_{0i}$.

A végberendezés szögsebességéhez minden rotációs csukló szögsebessége hozzájárul. A végberendezés szögsebesség-komponensei a bázis koordináta-rendszerben az alábbi vektor összegzéssel számíthatóak:

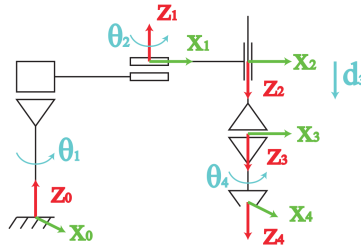
$$\boldsymbol{\omega} = \sum_{i=1}^n \rho_i \dot{q}_i \mathbf{k}_{0i} \quad (9.11)$$

ahol $\dot{q}_i = \dot{\theta}_i$ a Denavit–Hartenberg-konvenció szerint.

Tehát a Jacobi-mátrix szögsebességeket leíró komponense:

$$J_{\omega} = (\rho_1 \mathbf{k}_{01} \quad \rho_2 \mathbf{k}_{02} \quad \dots \quad \rho_n \mathbf{k}_{0n}) \quad (9.12)$$

Példa – SCARA robot Jacobi-mátrixa: a 9.2. ábrán látható robot Denavit–Hartenberg-paramétereit a 9.1. táblázat tartalmazza.



9.2. ábra. SCARA robot csukló tengelyei

9.1. táblázat. SCARA robot Denavit–Hartenberg-paraméterei

T	$\underline{\theta}$	$\underline{d}$	$\underline{a}$	$\underline{\alpha}$
$0 \rightarrow 1$	θ_1^*	d_1	a_1	0
$1 \rightarrow 2$	θ_2^*	0	a_2	π
$2 \rightarrow 3$	0	d_3^*	0	0
$3 \rightarrow 4$	θ_4^*	d_4	0	0

A Denavit–Hartenberg-paramétereknek megfelelő homogén transzformáció mátrixa a bázistól a végberendezésig:

$$T_{04} = \begin{pmatrix} c_{\theta_{12}}c_{\theta_4} + s_{\theta_{12}}s_{\theta_4} & -c_{\theta_{12}}s_{\theta_4} + s_{\theta_{12}}c_{\theta_4} & 0 & \underbrace{a_1c_{\theta_1} + a_2c_{\theta_{12}}}_{p_x} \\ s_{\theta_{12}}c_{\theta_4} - c_{\theta_{12}}s_{\theta_4} & -s_{\theta_{12}}s_{\theta_4} - c_{\theta_{12}}c_{\theta_4} & 0 & \underbrace{a_1s_{\theta_1} + a_2s_{\theta_{12}}}_{p_y} \\ 0 & 0 & -1 & \underbrace{d_1 - d_3 - d_4}_{p_z} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (9.13)$$

ahol $c_{\theta_1} = \cos(\theta_1)$, $c_{\theta_{12}} = \cos(\theta_1 + \theta_2)$, $c_{\theta_4} = \cos(\theta_4)$, $s_{\theta_1} = \sin(\theta_1)$, $s_{\theta_{12}} = \sin(\theta_1 + \theta_2)$, $s_{\theta_4} = \sin(\theta_4)$.

A transzformációs mátrixból kiemelve a pozícióvektort, meghatározható a Jacobi-mátrix lineáris sebességkomponensekhez tartozó része:

$$J_v = \begin{pmatrix} \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} & \frac{\partial p_x}{\partial d_3} & \frac{\partial p_x}{\partial \theta_4} \\ \frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2} & \frac{\partial p_y}{\partial d_3} & \frac{\partial p_y}{\partial \theta_4} \\ \frac{\partial p_z}{\partial \theta_1} & \frac{\partial p_z}{\partial \theta_2} & \frac{\partial p_z}{\partial d_3} & \frac{\partial p_z}{\partial \theta_4} \end{pmatrix} = \begin{pmatrix} -a_1s_{\theta_1} - a_2s_{\theta_{12}} & -a_2s_{\theta_{12}} & 0 & 0 \\ a_1c_{\theta_1} + a_2c_{\theta_{12}} & a_2c_{\theta_{12}} & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad (9.14)$$

A Jacobi-mátrix szögsebességekhez tartozó komponensénél figyelembe lehet venni, hogy a SCARA robot minden rotációs csuklója a bázis z tengelye körül forog, tehát J_ω -nak az x és y tengelyekhez tartozó sorainak összes eleme zéró.

Az első két oszlop esetén a forgástengelyek párhuzamosak a z tengellyel, tehát ezek az oszlopok $(0 \ 0 \ 1)^\top$. A harmadik csukló translációs, tehát a harmadik oszlop $(0 \ 0 \ 0)^\top$. A negyedik csuklónál a z tengely irányítása megfordul, tehát a hozzátartozó egységvektor a bázisban felírva

$(0 \ 0 \ -1)^\top$. Ennek megfelelően:

$$J_\omega = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{pmatrix} \quad (9.15)$$

A robot Jacobi-mátrixát a lineáris sebességeket leíró komponens, illetve a szögsebességeket leíró komponens összetevéséből kapjuk:

$$J = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \quad (9.16)$$

A robot Jacobi-mátrixának elemei általában függenek a csuklózóváltozóktól, tehát mozgás közben a Jacobi-mátrix elemei változnak. A végberendezés sebességkomponenseit az alábbi összefüggés alapján számíthatjuk:

$$\begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix} = J(\mathbf{q})\dot{\mathbf{q}} \quad (9.17)$$

A $J(\mathbf{q})$ elemei mozgás közben változnak, mivel $\mathbf{q}$ változik, így a végberendezés sebességkomponensei nem föltétlenül konstansok abban az esetben sem, amikor a csuklósebességek konstansok, *konstans csuklósebességhez általában nem tartozik konstans végberendezés sebesség.*

Inverz sebességkinematikai feladat: adott a robot végberendezésének előírt sebességvektora, $(\mathbf{v}^\top \ \omega^\top)^\top$. Keressük a csuklósebességek vektorát ($\dot{\mathbf{q}}$), amelyek képesek biztosítani az előírt végberendezés sebességet.

Ha a robot hat szabadságfokú, a Jacobi-mátrix négyzetes lesz, dimenziója 6×6 . Amennyiben a Jacobi-mátrix invertálható, az inverz kinematikai feladat megoldása:

$$\dot{\mathbf{q}} = J^{-1} \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix} \quad (9.18)$$

9.2.1 Szinguláris konfigurációk

A Jacobi-mátrix elemei a robot csuklópozícióitól függenek ($J = J(\mathbf{q})$). Mozgás közben a robot csuklópozíciói változnak, tehát a Jacob-mátrix bizonyos elemei nem konstansok, előfordulhat, hogy a Jacobi-mátrix determinánsa bizonyos csuklózóváltozó értékekre nem nulla, más csuklózóváltozó értékekre nulla lesz.

Azokat a konfigurációkat, amelyekben a Jacobi-mátrix determinánsa nulla, *szinguláris konfigurációnak* ($\mathbf{q}_S$) nevezzük:

$$\det(J(\mathbf{q}_S)) = 0 \quad (9.19)$$

Példa – SCARA robot mozgása xOy -ban: a SCARA robot (9.14) Jacobi-mátrixából emeljük ki azokat az elemeket, amelyek a lineáris sebességkomponenseket írják le x_0Oy_0 síkban:

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \underbrace{\begin{pmatrix} -a_1 s_{\theta_1} - a_2 s_{\theta_{12}} & -a_2 s_{\theta_{12}} \\ a_1 c_{\theta_1} + a_2 c_{\theta_{12}} & a_2 c_{\theta_{12}} \end{pmatrix}}_J \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} \quad (9.20)$$

ahol $\theta_1 = q_1$ és $\theta_2 = q_2$.

A J mátrix determinánsa:

$$\det(J) = a_1 a_2 s_{\theta_2} \quad (9.21)$$

Az inverz kinematikai feladat megoldásához szükséges Jacobi-mátrix inverze:

$$J = \frac{1}{a_1 a_2 s_{\theta_2}} \begin{pmatrix} a_2 c_{\theta_{12}} & a_2 s_{\theta_{12}} \\ -a_1 c_{\theta_1} - a_2 c_{\theta_{12}} & -a_1 s_{\theta_1} - a_2 s_{\theta_{12}} \end{pmatrix} \quad (9.22)$$

A (9.21) összefüggésből látszik, hogy, amennyiben $\theta_2 = 0$ vagy $\theta_2 = \pi$, akkor az inverz Jacobi-mátrix nem határozható meg, az inverz kinematikai feladat nem megoldható. A SCARA robot esetében látszik, hogy ez olyan eseteknek felel meg, amikor a robotkar elérte a munkaterének határát $\theta_2 = 0$, vagy a második szegmens 180° -ban vissza van hajolva.

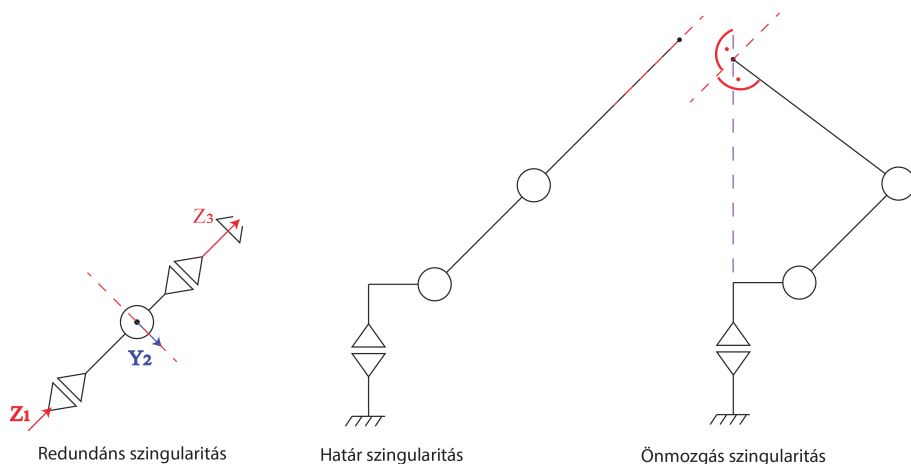
SCARA robot esetén a szinguláris konfigurációk:

$$\mathbf{q}_{S1} = \begin{pmatrix} \theta_1 \\ 0 \end{pmatrix}, \quad \mathbf{q}_{S2} = \begin{pmatrix} \theta_1 \\ \pi \end{pmatrix}, \quad \theta_1 \in [-\pi, \pi]; \quad (9.23)$$

Szinguláris konfigurációban a robot a munkatér bizonyos irányjaiban nem tud sebességet generálni (*degenerált irányok*). Ezekbe az irányokba a robot nem tud instant módon elmozdulni.

Szinguláris típusok: általában megkülönböztetünk kézcsukló szingularitásokat és kar szingularitásokat.

Kézcsukló szingularitás: az Euler-kézcsukló akkor kerül szinguláris konfigurációba, amikor a csukló három forgástengelye közül az első (z_1) és a



9.3. ábra. Robotmanipulátorok szinguláris konfigurációban

harmadik (z_3) tengelye egybe esik, lásd 9.3. ábra. Ekkor, ha a harmadik tengely szögsebessége ugyanaz, mint az elsőé, csak ellentétes irányú, a végberendezés nem mozog.

Mivel ebben a szinguláris konfigurációban az első vagy a harmadik csukló mozgásával ugyanazt a végberendezés mozgást érhetjük el, ezt *redundancia szingularitás*nak is nevezzük.

A degenerált irány megegyezik a második forgástengely irányával.

Általában is, olyan háromtengelyű kézcsuklók esetében, amelyeknél a forgástengelyek egy ponton haladnak át, ha két forgástengely kollineárisá válik, szinguláris konfigurációban kerülünk, a kézcsukló szabadságfokot veszít.

Kar szingularitások: tipikus karsingularitás, amikor a robotkar teljesen ki van nyúlva, elérte a munkatér határát (*határ szingularitás*). Ebben az esetben a degenerált irányt a kinyúló szegmens iránya határozza meg.

Lehetnek kar szingularitások a munkatér belsejében is. A 9.3. ábrán látható szingularitás akkor alakul ki, amikor a robot végpontja az első tengely forgástengelyére esik; ezt *önmozgás szingularitás*nak is nevezzük. Ennél a szingularitásnál a degenerált irány merőleges a második és harmadik szegmens által meghatározott síkra.

9.3. A gyakorlat menete

A gyakorlat célja: a KUKA robot első három szabadságfokát figyelembe véve térképezzük fel a robot munkaterét, majd mentsük el egy *CSV* állományba a munkatér szinguláris pontjainak megfelelő csuklózváltozókat. *Megjegyzés:* felhasználjuk az előző két labor alatt fejlesztett direkt és inverz geometriai metódusokat.

1. A teljes csuklózváltozók terét egy adott *res* felbontással járjuk be.
2. A direkt geometriai feladat segítségével feltérképezzük a csuklózváltozóknak megfelelő végberendezés koordinátákat (munkateret).
3. A Jakobi-mátrix determinánsát felhasználva határozzuk meg a szinguláris konfigurációt.
4. Generáljuk ki azokat a csuklózváltozókat, amelyek a robotmanipulátort végigviszik a $(-40 \ -20 \ -20)^T - (20 \ 40 \ 30)^T$ pontok között feszített egyenesen.

Legyen minden csuklózváltozó $[0, 2\pi]$ intervallumból. Ekkor a különálló ponthalmazt képezni tudjuk a *numpy linspace* metódusával, melynek paraméterei a minimális érték, maximális érték és a felbontás. A csuklózváltozók terét a *numpy meshgrid* metódusával alkotjuk, végül a *numpy vstack* metódusa az ismert alakba hozza a halmazt:

```
q1, q2, q3 = np.meshgrid(q1, q2, q3)
Q = np.vstack((q1.flatten(), q2.flatten(), q3.flatten())).T
```

A munkatér generálása egyszerű direkt geometriai feladat végigiterálva a csuklózváltozó pontokon:

```
for q in Q:
    R.set_joints(q[0], q[1], q[2], unit='rad')
    _, _, v = R.direct_kinematics()
    x.append(v[0])
    y.append(v[1])
    z.append(v[2])
fig = plt.figure(2)
ax = plt.axes(projection='3d')
ax.scatter3D(x, y, z)
plt.show()
```

A szinguláris pontok meghatározásához írjunk egy `get_jacobian(Q)` metódust a robot osztályba, amely az adott robot geometria és csukló változók függvényében visszatéríti a Jacobi-mátrixot. Egy mátrix determinánsát Pythonban a `numpy.linalg.det` metódussal kapjuk meg. Igaz, hogy $\det(J) = 0$ számít szingularitásnak, viszont minél közelebb kerülünk ehhez a ponthoz, annál nehezebb a kinematikai számításokat elvégezni. Ebből kifolyólag minden olyan pontot, amelyre $|\det(J)| < 1E-6$ szinguláris pontnak tekintünk és elmentünk egy listába. Egy `theta_vect` lista tartalmát a

```
np.savetxt('singular.csv', theta_vect, delimiter=',')
```

metódussal mentjük ki állományba.

Pályagenerálás: a pályagenerálási feladat az előző metódusokra épül. Ha adott két térbeli pont $\mathbf{a} = a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}$ és $\mathbf{b} = b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}$, az a vektor, mely az origóból indul, és a végpontja végigjárja az AB szakaszt, felírható az alábbi lineáris interpolációval:

$$\mathbf{r} = \mathbf{a} + \alpha(\mathbf{b} - \mathbf{a}), \quad \alpha \in [0, 1].$$

Azaz, ha mi a megadott két térbeli pont között akarjuk mozgatni a végberendezést, akkor a neki megfelelő térbeli koordináták:

$$\begin{aligned} x &= a_x + \alpha(b_x - a_x) \\ y &= a_y + \alpha(b_y - a_y) \\ z &= a_z + \alpha(b_z - a_z) \end{aligned}$$

A feladatot az alábbi lépésekben végezzük el:

- Generáljuk a végberendezés pályáját a $(-40 \quad -20 \quad -20)^\top - (20 \quad 40 \quad 30)^\top$ pontok között 1000 lépésben.
- Legyen a robot csuklóinak kezdeti állapota az első pontnak megfelelő csuklólévő vektor (használjuk az inverz geometria metódust).
- Határozzuk meg a következő végberendezés pontnak megfelelő csuklólévőket.
- Ha nem értünk szingularitásba, legyen az új csuklóállapot a robot aktuális csuklólévője.
- Ha szingularitásba értünk, jelezzük ezt, és próbálkozzunk a következő ponttal, ameddig elérünk a végpontba.

9.4. Kérdések és feladatok

1. Módosítsuk a Python szkriptet úgy, hogy egy körpályán mozogjon a robot végberendezése.
2. Módosítsuk a Python szkriptet úgy, hogy a szinguláris pontok más színnel jelenjenek meg a munkatér ábrázolásánál.
3. Python alatt képezzük a munkatér szélső pontjai által alkotott burkolófelületet.

ROBOTOK PROGRAMOZÁSA

10.1. A gyakorlat célja

A Universal Robots robotcsalád programozási környezetének megismerése, az *URScript* programozási nyelv fontosabb utasításainak, változótípusainak bemutatása; robotprogram fejlesztése szimulációs környezetben.

10.2. Elméleti bevezető

10.2.1. Robotprogramozási nyelvek

A robotprogramozási nyelvek magas szintű programozási nyelvek, amelyek segítségével leírhatjuk egy robotmanipulátor feladatát. A feladatot jellemzően mozgássorozatként írjuk le, meghatározva az egyes mozgáselemek jellegzetességeit, mint például a sebesség, gyorsulás, a pályaelem formája.

Vezérlő utasítások segítségével lehetőség van feltételes vagy ismétlődő mozgásszekvenciák leírására. A robotprogram jól elválasztható részeit megadhatjuk alrutinokban a klasszikus programozási nyelvekhez hasonlóan.

Komplex feladatok esetében a robotprogramozási nyelvekben lehetőség van kommunikálásra külső eszközökkel. Sok esetben lehetőség van hálózati kommunikációra távoli számítástechnikai eszközökkel. Működési feltételek megállapításához használhatunk digitális és analóg ki- és bemeneteket, például fényzorompó vagy erőmérő bélyeg jelének beolvasására.

Mivel az ipari robotok legtöbb esetben időtálló és megbízható automatizálási eszközök, hosszú időintervallumon keresztül használhatjuk őket. Ennek következményeként a robotprogramozási nyelv kompatibilitást kell mutasson a régi típusú robot szoftverekkel is. Ezért a robotprogramozási nyelvek viszonylag lassan fejlődnek, a nyelvek alapjait a robotgyártók nem, vagy csak részben módosítják.

A robotprogramozási nyelvek szintaxisa sok esetben az Assembly-szerű nyelvekre hasonlít. A címkék használata is számos robotprogramozási nyelvben megmaradt.

Az ipari robotok vezérlője általában tartalmaz egy kézi vezérlőegységet, amin keresztül a robotot programozni is lehet. Komplexebb robotprogramokat azonban érdemes számítógépen fejleszteni, a robotgyártó által javasolt programozási környezetben.

A robotgyártók eltérő programozási nyelveket ajánlanak a robotjaikhoz.

Az *ABB* robotok programozási nyelve a *RAPID*, a robot számítógépről is elérhető a *RobotStudio* környezetből.

A *FANUC* robotokat *TP* (*Teach Pendant*) nyelven lehet programozni, de komplexebb feladatok leírására a *KAREL* nyelvet ajánlják. Ugyanakkor a *KAREL* nem tartalmaz robotmozgató utasításokat.

A *KUKA* robotok programozási nyelve a *KRL*.

Az *UR* robotcsaládot, a többi robothoz hasonlóan, programozhatjuk egyenesen a kézi vezérlőről (*PolyScope*), vagy számítógépen keresztül az *URScript* nyelvet használva, ami Python programozási környezetből is elérhető. Az alábbiakban az *UR* robotcsalád programozását tárgyaljuk.

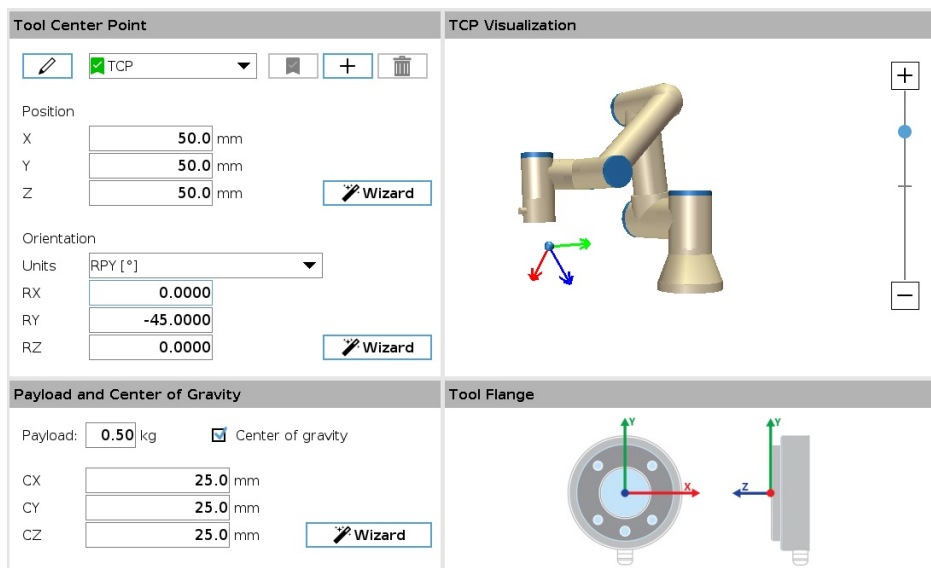
10.2.2. Végberendezés paraméterek beállítása

A robotot a feladata elvégzéséhez végberendezéssel szereljük fel. Ez lehet megfogó, festékszóró, megmunkáló eszköz (például fúrófej) stb. Amennyiben az eszköz megfogó, a robot egy céltárgyat mozgathat. A végberendezés és az esetleges céltárgy együttes mechanikai paramétereit meg kell adni a robotirányítási szoftvernek.

Szerszámközpont (TCP) helye és iránya: a szerelési felülethez (utolsó szegmens pereméhez) rendelt koordináta-rendszerben adjuk meg. A 10.1. ábrán ez a *Tool Flange* alábrán látszik.

Megadjuk a szerszámközpont térbeli koordinátáit, illetve a szerszám irányát, például RPY szögekkel leírva, lásd a *Tool Center Point* alábra. Ezek az információk geometriai számításokhoz szükségesek, például a direkt vagy inverz geometriai feladatok megoldásához.

Szükséges még megadni a szerszám és mozgatott tárgy együttesének tömegét és tömegközéppontjának térbeli helyzetét, ugyancsak a robot szerelési felülethez rendelt koordináta-rendszerben, lásd a *Payload and Center of Gravity* alábra. Ezt az információt például a robotszabályozás használja, mivel az előírt mozgás megvalósításához szükséges a mechanikai paramétereiktől függő erőket, nyomatékokat kompenzálni.



10.1. ábra. Munkadarab beállítására szolgáló interfész (minden jog fenntartva a Universal Robots™ részére)

A feladat közben a robot által mozgatott céltárgy, sőt a szerszámközeppont helye is változhat. Ezért a mechanikai paraméterek a programozási nyelvben is módosíthatóak.

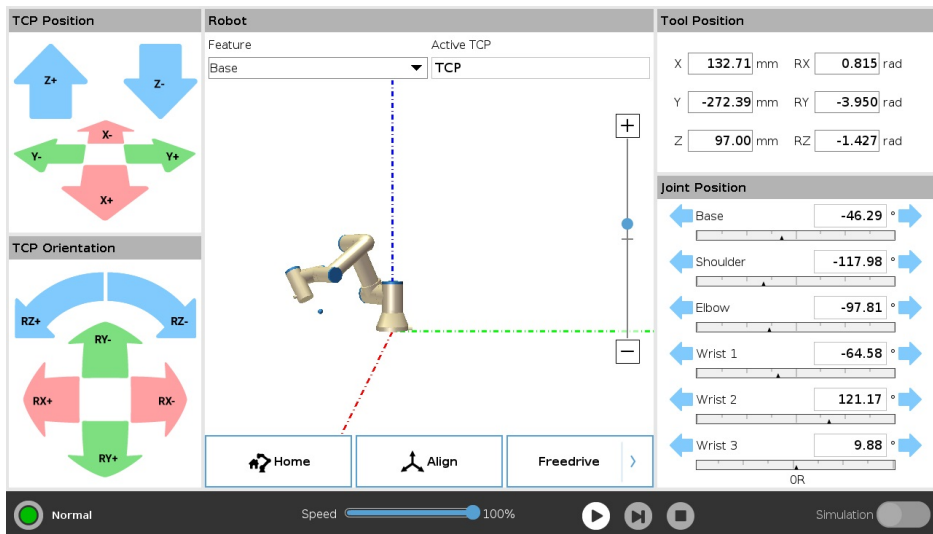
10.2.3. Kézi vezérlés

A kézi vezérlés esetén mozgathatjuk a robotot a különböző csuklótengelyek körül, valamint a végberendezést a bázis koordináta-rendszer tengelyei mentén/körül.

A mozgítás során a robotirányítási szoftver megjeleníti a csuklóváltozók értékeit, valamint a végberendezés pozícióját és orientációját.

A csuklók elnevezései:

- q_1 – bázis (base)
- q_2 – váll (shoulder)
- q_3 – könyök (elbow)
- q_4 – csukló 1 (wrist 1)
- q_5 – csukló 2 (wrist 2)
- q_6 – csukló 3 (wrist 3)



10.2. ábra. Kézi vezérlésre szolgáló interfész (minden jog fenntartva a Universal RobotsTM részére)

A végberendezés kétirányú kézi vezérlése a bázis koordináta-rendszer három tengelye mentén, illetve körül történik (X , Y , Z , RX , RY , RZ). A tengelyek színjelölései:

- x – piros
- y – zöld
- z – kék

10.2.4. Az URScript nyelv néhány eleme

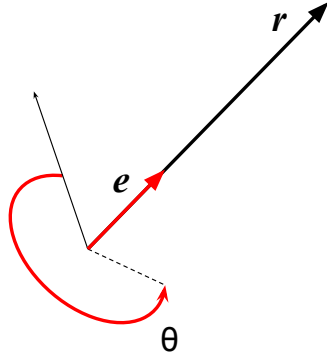
Változók

A programozott robot 6 szabadságfokú, hat rotációs csuklóval rendelkezik. Ennek megfelelően egy csuklókonfigurációt az *URScript* nyelvben 6 elemű vektorral adunk meg:

$$q = [q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6]$$

A vektor komponenseit radiánban adjuk meg.

Egy végberendezés konfigurációt az *URScript* nyelvben *waypoint*-nak nevezünk. A változónak az alábbi formája van:



10.3. ábra. Rotációs vektor

$p[x, y, z, ax, ay, az]$, ahol x, y, z a szerszámközep pont a bázis koordináta-rendszerben, ax, ay, az pedig az orientációt definiálja a bázis koordináta-rendszerhez képest.

Az orientációt *tengely körüli forgatással* definiáljuk. Az orientáció leírásánál először megadunk egy egységnyi hosszúságú vektort a bázis koordináta-rendszerben. Ez lesz a forgástengely:

$$\mathbf{e} = (e_x \ e_y \ e_z)^T, \quad e_x^2 + e_y^2 + e_z^2 = 1. \quad (10.1)$$

Ezenfelül még szükség van egy θ szögre, ami megadja az elforgatás mértékét a forgástengely körül.

Az $\mathbf{e}$, vektorral valamint a θ szöggel bármilyen térbeli forgatást jellemezhetők, lásd 10.3. ábra.

A p konfiguráció vektor rotációt definiáló komponenseit (ax, ay, az) úgy határozzuk meg, hogy az $\mathbf{e}$ vektort megszorozzuk a θ szöggel:

$$(ax \ ay \ az)^T = \theta(e_x \ e_y \ e_z)^T. \quad (10.2)$$

A végberendezés konfigurációban a pozíció komponenseket mm -ben, az orientáció komponenseket radiánban adjuk meg.

Például egy z tengely körüli $\pi/2$ forgatást leíró rotációs vektor:

$$(ax \ ay \ az)^T = (0 \ 0 \ \pi/2)^T. \quad (10.3)$$

A konfiguráció típusú változók mellett a közismert változótípusokat is használhatjuk a programozási nyelvben, mint például `int`, `float`, `string`, `bool`.

Általános műveletek, utasítások

Példák műveletekre:

- $1+2-3$
- $4*5/6$
- $(1+2)*3/(4-5)$
- `"Hello" + ", " + "World!"`

Példák logikai változókra és logikai műveletekre:

- `True or False and (1 == 2)`
- `1 > 2 or 3 != 4 xor 5 < -6`
- `not 42 >= 87 and 87 <= 42)`
- `"Hello" != "World" and "abc" == "abc"`

Példa feltételes utasításra:

```
if a > 2:
    a = a + 1
elseif b < 2:
    b = b * a
else:
    a = a + b
end
```

Példa ciklikus utasításra:

```
l = [1, 2, 3, 4, 5]
i = 0
while i < 5:
    l[i] = l[i] * 2
    i = i + 1
end
```

Függvények használata:

```
def add(a, b):
    return a+b
end
...
result = add(1, 4)
```

A nyelvben be vannak építve a fontosabb matematikai függvények, mint például:

- trigonometriai függvények: például `cos(x)`, `atan2(x,y)`
- más nemlineáris függvények: például `log(x)`, `sqrt(x)`
- kerekítésekre alkalmazott függvények: például `ceil(x)`, `floor(x)`

Késleltetés: `sleep(t)` – t másodperc várakozást szűr be a program futásába.

Robotirányítást támogató függvények és eljárások

`get_actual_joint_positions()`

`get_actual_joint_speeds()`

Hatkomponensű vektort térítenek vissza (q , dq) a csuklók szögpozíció, illetve szögsebesség komponenseivel (radiánban, illetve radián/szekundumban mérve).

`get_joint_torques()`

Hatkomponensű nyomatékvektort térít vissza (τ), amely a csuklókra ható külső nyomatékokat tartalmazza (Nm-ben mérve). A visszatérített nyomatékkomponensek nem tartalmazzák a gravitáció, súrlódás stb. robotra kifejtett hatását.

`get_actual_tcp_pose()`

Hatkomponensű vektort térít vissza (p), amely a végberendezés pozícióját, orientációját tartalmazza. Az orientáció a forgatási vektor alapján van definiálva.

`get_actual_tcp_speed()`

Hatkomponensű vektort térít vissza (dp). Az első három eleme a vektornak a végberendezés lineáris sebesség komponensei a három bázis koordináta-rendszer tengely mentén. A második három elem pedig definiálja a forgástengely vektor sebesség komponenseit. A vektor hossza a szögsebesség a forgástengely körül (radián/szekundumban).

`get_forward_kin(q)`

Hatkomponensű konfiguráció vektort (p) térít vissza, amely a q csuklókoordinátáknak megfelelő végberendezés konfigurációt tartalmazza.

`get_inverse_kin(p)`

Hatkomponensű q vektort térít vissza, amely a p végberendezés konfigurációnak megfelelő csuklókonfigurációt tartalmazza.

`is_within_safety_limits(p)`

Logikai változót térít vissza (*true* vagy *false*), amely megadja, hogy a p végberendezés konfiguráció elérhető-e a robot számára.

`move1(p, a=1.4, v=1.05, t=0, r=0)`

A pályatervezés világkoordinátákban történik, a végberendezés az aktuális konfigurációjából indulva ér el a $p = [x \ y \ z \ rx \ ry \ rz]$ célkonfigurációba.

a – robot végberendezés maximális gyorsulása

v – a robot végberendezés maximális sebessége

t – a mozgás ideje. Amennyiben ez a paraméter nem zero, a robot kiszámolja a sebességet és gyorsulást a paraméterben megadott idő függvényében, nem veszi figyelembe v és a bemeneti paramétereket.

r – a pálya görbület paramétere a pontban (*blend radius*). Ha ez az érték nem zero, a robot nem áll meg a p vektorral definiált pozícióban, hanem elhalad a közelében, a következő mozgató utasításban megadott cél fele.

Példa: A 10.4. ábra a *blend radius* bemeneti paraméter hatását mutatja a robot mozgására. A mozgás két egymás utáni `move1` utasítással van kialakítva.

I. eset: a robot érinti a p_via konfigurációt ($r = 0$).

`move1(p_via, a=1.4, v=1.05, t=0, r=0)`

`move1(p_goal, a=1.4, v=1.05, t=0, r=0)`

II. eset: a robot nem érinti a p_via konfigurációt, az adott görbülettel elhalad mellette ($r = 0.5$).

`move1(p_via, a=1.4, v=1.05, t=0, r=0.5)`

`move1(p_goal, a=1.4, v=1.05, t=0, r=0)`

`movej(q, ddq=1.4, dq=1.05, t=0, r=0)`

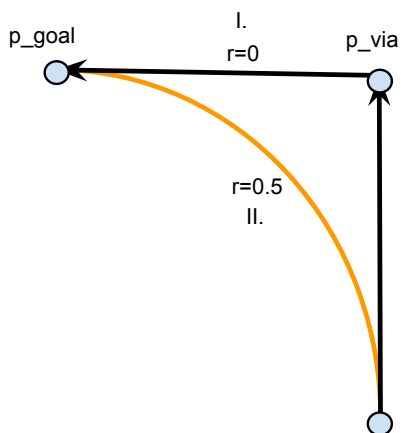
A robot csukló koordinátákban fog mozogni az aktuális konfigurációjából a $q = [q1 \ q2 \ q3 \ q4 \ q5 \ q6]$ cél konfigurációba.

ddq – robot csuklók maximális gyorsulása

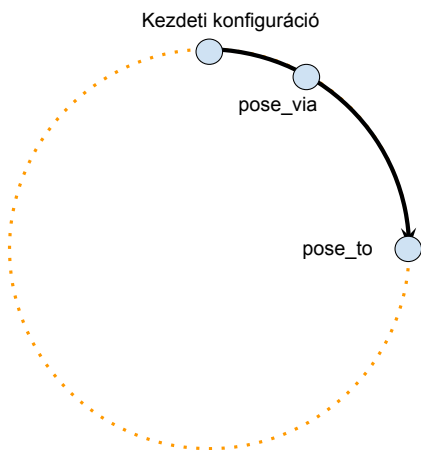
dq – a robot csuklók maximális sebessége

t – a mozgás ideje. Amennyiben ez a paraméter nem zero, a robot kiszámolja a sebességet és gyorsulást, nem veszi figyelembe dq és ddq bemeneti paramétereket.

r – a pálya görbület paramétere a pontban (*blend radius*). Ha ez az érték nem zero, a robot nem áll meg a q vektorral definiált konfigurációban, hanem elhalad a közelében, a következő mozgató utasításban megadott cél irányába.



10.4. ábra. *Elhaladás via pont mellett*



10.5. ábra. *Körmozgás utasítás*

```
movec(pose_via, pose_to, a=1.2, v=0.25, r=0, mode=0)
```

A robot végberendezése egy köríven mozog az aktuális konfigurációból kiindulva a *pose_via*-n keresztül a *pose_to* célpozícióba (lásd 10.5. ábra). A *via* konfiguráció megadása kötelező, mert három pontra van szükség ahhoz, hogy a robotirányítási szoftver a kört meg tudja határozni.

pose_to – célpozíció

pose_via – a kör egy pontja, amin a végberendezés áthalad

a – robot végberendezés maximális gyorsulása

v – a robot végberendezés maximális sebessége

mode – a mozgás közbeni orientációt szabályozó logikai változó

0 – az orientáció a kiinduló konfiguráció és a végpont konfiguráció között változik a megadott orientációs paramétereknek megfelelően

1 – az orientáció konstans, tangenciális a körhöz képest

Az ipari robotok vezérlői általában digitális és analóg be- és kimenetekkel vannak ellátva a robotikai feladathoz szükséges extra jelek beolvasásához, illetve kiküldéséhez. Ezek segítségével vezérelhetjük a robot környezetében található automatizálási eszközöket a robot szoftveren belül (például fényzorompó beolvasása, futószalag indítása vagy pneumatikus henger vezérlése).

Példa: A robotvezérlőn található általános célú digitális ki- és bemenetek lekérdezése, beállítása.

```
get_digital_in(n)
```

```
set_digital_out(n, b)
```

n – a digitális kimenet azonosítója (0:9)

b – a digitális kimenet kívánt állapota (*true* vagy *false*)

10.3. A gyakorlat menete

A gyakorlat célja: az URSim környezetben valósítsunk meg egy Pick-and-Place feladatot az UR3-as robottal, tudva azt, hogy a megfogó szerkezet középpontja 5cm-re van eltolva a robot végpontjától. A megfogó szerkezet tömege 0.2 kg. A nyitását a 0-ás digitális kimenet alacsony logikai állapota, a csukását pedig magas állapota jelenti.

1. A helyes mozgatáshoz és a szabályozáshoz adjuk meg a végberendezés mechanikai paramétereit.

2. Programozzuk le a Pick-and-Place feladatot a megadott beállításokkal.
3. Teszteljük a folyamatot csökkentett, majd maximális sebességgel.

1. *Az URSim beüzemelése:* Az URSim off-line szimulációs környezet a roboton futó alkalmazás tükörképe. A robot Linux (lubuntu) operációs rendszert tartalmaz, ezért Windows környezetben natívan nem tudjuk futtatni az alkalmazást. A Linux emulálására több lehetőség adott:

1. Docker környezetben futtathatunk egy URSim konténert
2. Virtuális gépben betölthetünk egy URSim képet

Egy virtuális gép önálló egységet képez saját kernellel és biztonsági egységgel, habár több tárhelyet igényel, és lassabb a betöltése. A laborfeladatot az **Oracle VirtualBox** segítségével oldjuk meg, viszont egyéb virtuális gép (pl. VMWare Player) is hasonlóan beállítható. Töltsük le az **URSim** képet.

- Indítsuk el a virtuális környezetet.
- Hozzunk létre egy új gépet (New).
- Adjuk meg a gép nevét és típusát (pl URSim, Linux, Ubuntu).
- Adjuk meg a memóriaigényt (768 MB).
- Adjuk meg a letöltött URSim képet (URSim.zip) tartalmazó könyvtárat (C://URSim – laborgépek esetén).
- A Start gombbal indítsuk el a gépet és várjuk meg a lubuntu betöltését.
- Indítsuk el az UR3 szimulációs környezetet.

2. *Pick-and-Place eljárás:*

Toljuk el a végberendezés középpontját z tengely mentén 5 cm-rel, és a tömeget állítsuk be 200 g-ra, a 10.6. ábra szerint.

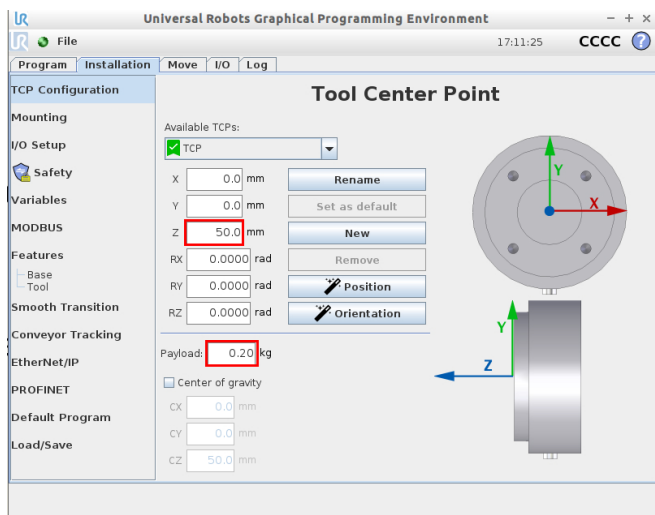
A mozgássorozat folyamatábráját a 10.7-es ábra szemlélteti. A robot kezdetben egy ismeretlen konfigurációban található, ezért első lépésben a robotot a HOME konfigurációba visszük *moveJ* utasítással.

A munkadarab világkoordinátája (W2)

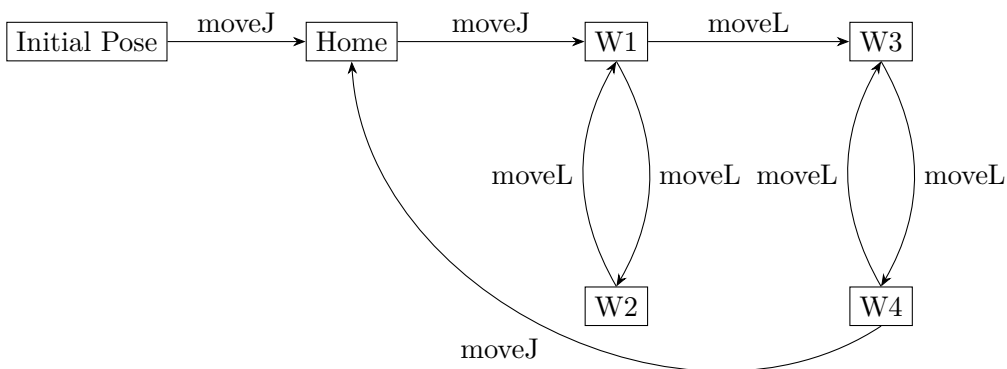
$$[298.6 \text{ mm}, -112.35 \text{ mm}, -400 \text{ mm}],$$

a munkadarab fölötti via pont (W1) világkoordinátája

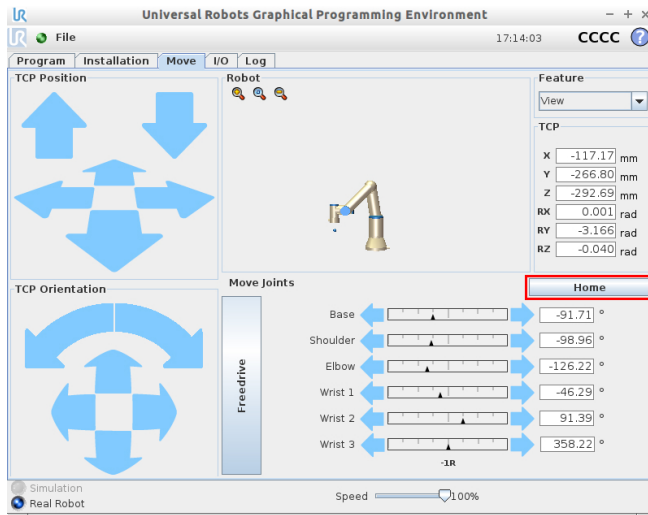
$$[298.6 \text{ mm}, -112.35 \text{ mm}, -136.55 \text{ mm}]$$



10.6. ábra. Végberendezés beállításai (minden jog fenntartva a Universal Robots™ részére)



10.7. ábra. A feladat folyamatábrája



10.8. ábra. Home konfiguráció (minden jog fenntartva a Universal Robots™ részére)

és a neki megfelelő csuklókoordináta

$$[0^\circ, -90^\circ, -90^\circ, -90^\circ, 90^\circ, 0^\circ].$$

A W3-as via pont világkoordinátája

$$[370 \text{ mm}, 69.7 \text{ mm}, -136.55 \text{ mm}].$$

A célpont (W4) világkoordinátája

$$[370 \text{ mm}, 69.7 \text{ mm}, -400 \text{ mm}].$$

A robotkart juttassuk el a munkadarab fölé (W1), csuklómozgást használva, nyissuk ki a megfogót, és várjunk 0.1 s-ot. Lineáris mozgással érjük el a munkadarabot (W2), csukjuk be a megfogót, várjunk 0.1 s-ot, majd vigyük vissza a végberendezést lineárisan a W1-es pontba.

Vigyük át a munkadarabot a W3-as pontba lineáris vagy csuklómozgással. A W3-as pontból közelítsük meg a W4-es pontot lineárisan, nyissuk ki a megfogót, és várjunk 0.1 s-ot. Vigyük vissza a végberendezést a W3-as pontba, csukjuk össze a megfogót, és csuklómozgással juttassuk el a Home állapotba.

Az ablak alján látható sebességcsúszka segítségével tesztelhetjük a programunkat csökkentett sebességgel ($\text{speed} = 20\%$). A robot térbeli mozgását a *Graphics* ablakban láthatjuk.

Amennyiben a mozgásszekvencia helyes, teszteljük a programot maximális sebességgel ($\text{speed} = 100\%$).

10.4. Kérdések és feladatok

1. Hogyan módosítsuk a programunkat, ha a végberendezés tartalmaz végálláskapcsolókat (nyitott állapotban logikai magas, zárt állapotban logikai alacsony állapot), és nem szeretnénk idő alapú várakozást alkalmazni?
2. Módosítsuk a programunkat úgy, hogy a robot az alkatrész áthelyezésekor lekerekített ívben kerüljön át a célpontba.
3. Tegyük be egy feltételes program részletet, mely ellenőrzi, hogy az alkatrész meghalad-e egy előírt tömeget (felhasználva a *zero_ftsensor()* és *get_tcp_force()* utasításokat). Ha meghaladja, álljon le a program futása, és írjon ki egy figyelmeztetést.

PÁLYATERVEZÉS ROBOTMANIPULÁTOROKNAK

11.1. A gyakorlat célja

Robotmanipulátorok pályatervezési feladatának megismerése, pályatervezési feladatok megvalósítása UR roboton.

11.2. Elméleti bevezető

Pályatervezési alapfogalmak robotmanipulátorokhoz

A pályatervezés során a robotok előírt mozgását határozzuk meg a robotikai feladat leírásának megfelelően. Tipikusan a végberendezés mozgását tervezzük.

A mozgás minden pillanatában a végberendezés minden konfigurációjához (térbeli pozíció és orientáció) $\mathbf{p}$ tartozik egy vagy több csuklókonfiguráció ($\mathbf{q}$) is:

$$\mathbf{p} = (x \ y \ z \ RX \ RY \ RZ)^\top \quad (11.1)$$

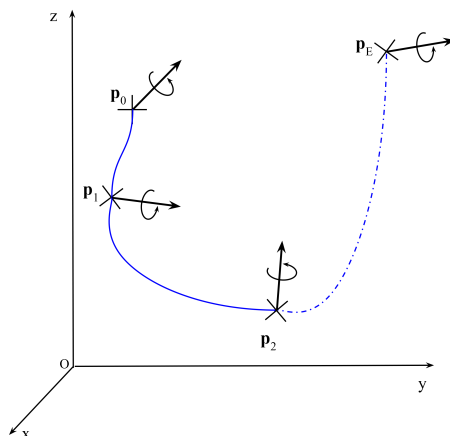
$$\mathbf{p} = \mathbf{p}(\mathbf{q}), \quad \mathbf{q} = (q_1 \ q_2 \ \dots \ q_n)^\top, \quad (11.2)$$

ahol x , y , z a végberendezés térbeli pozíciójának koordinátái a bázis koordináta-rendszerben, RX , RY , RZ az orientációt jellemző szögek, n a robot szabadságfoka, q_i az i -ik csuklópozíció.

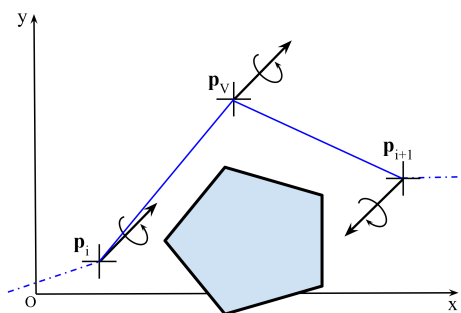
A robotmanipulátorok végberendezése egy konfigurációsorozaton kell végighaladjon a feladat megoldásához. Ezt nevezzük a *robot útjának*, lásd 11.1. ábra.

$$\mathbf{p}_0 \rightarrow \mathbf{p}_1 \rightarrow \mathbf{p}_j \dots \rightarrow \mathbf{p}_E \quad (11.3)$$

Sok esetben a robot az utat ismételten, ciklikusan többször is be kell járja. Ebben az esetben $\mathbf{p}_E = \mathbf{p}_0$. Komplexebb robotirányítási feladatoknál



11.1. ábra. *A robot útja és pályagörbéje*



11.2. ábra. *Áthaladás via pontokban*

az út a ciklusok során változhat a feladat logikájának és a külső feltételeknek megfelelően.

Amennyiben a robot munkaterében akadály van két egymást követő konfiguráció között, az utat kiterjeszthetjük ún. via pontokkal ($\mathbf{p}_v$), amelyek nem a feladat megoldását szolgálják, csak az ütközésmentes mozgás eléréséhez szükségesek, lásd 11.2. ábra. Via pontokban a robot tipikusan nem áll meg, a robot sebessége nem kell zero legyen.

Megmunkálási, anyagmozgatási feladatoknál a robot egy konfigurációban adott időintervallumot eltölt a feladat részének elvégzéséig (például a fúrás ideje megmunkálásnál, megfogó kinyitása anyagmozgatásnál).

Amennyiben a feladatot csak az úthoz tartozó konfigurációkban ($\mathbf{p}_i$) végzi a robot, célszerű, ha két útkonfiguráció között a lehető leggyorsabban végighalad.

Más feladatoknál, mint például festési vagy hegesztési feladatok, fontos a két munka közötti áthaladás sebessége, illetve az összekötő pályaszakasz geometriája (formája) is.

Pályatervezés csuklókoordinátákban és világkoordinátákban

A robot útjára tipikusan egy *pályagörbét* fektetünk, ami meghatározza a robot mozgását az útvonal konfigurációi között, lásd 11.1. ábra, két egymást követő konfiguráció közötti pályagörberész a *pályaelem*.

A pályagörbe tervezésének bemeneti adatai az útvonal konfigurációi, az előírt sebesség és gyorsulás a pálya mentén, a pályaelemek előírt formája.

A tervezés eredménye a robot csuklók előírt pozíciója, sebessége és gyorsulása a mozgás teljes időtartamára:

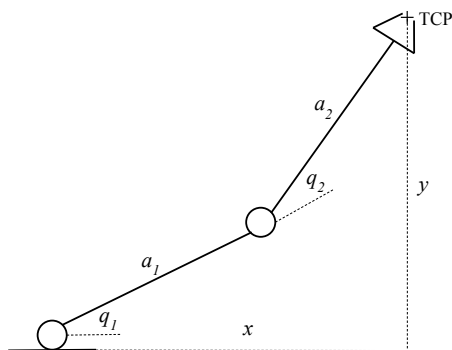
$$\begin{aligned}\mathbf{q}_r &= \mathbf{q}_r(t) \\ \dot{\mathbf{q}}_r &= \dot{\mathbf{q}}_r(t) \\ \ddot{\mathbf{q}}_r &= \ddot{\mathbf{q}}_r(t),\end{aligned}\tag{11.4}$$

ahol $\mathbf{q}_r$ a referencia csuklókonfiguráció.

A legtöbb esetben elvárt, hogy a pálya két útvonal-konfiguráció között sima (legalább kétszer deriválható) függvény legyen. Sok esetben (például via ponton áthaladás) a simaságot megköveteljük az útvonal-konfigurációban is.

Az út konfigurációinak ($\mathbf{p}_i$) megfelelő csuklókoordinátákat ($\mathbf{q}_i$) megkaphatjuk az inverz geometriai feladat megoldásával. Amennyiben az út konfigurációiban az előírt sebesség nem nulla, a megfelelő csuklósebességet az inverz kinematikai feladat megoldásával tudjuk meghatározni.

Pályatervezés csuklókoordinátákban: az útvonal-konfigurációknak megfelelő csuklókonfigurációkat „kötjük össze” pályaelemekkel és határozzuk meg a pálya menti csuklósebességeket és csuklógyorsulásokat.



11.3. ábra. Kétszabadságfokú robot

Pályatervezés világkoordinátákban: az útvonal-konfigurációknak megfelelő végberendezés konfigurációkat „kötjük össze” pályaelemekkel és határozzuk meg a pálya menti végberendezés-sebességeket és végberendezés-gyorsulásokat:

$$\begin{aligned}\mathbf{x}_r &= \mathbf{x}_r(t) \\ \dot{\mathbf{x}}_r &= \dot{\mathbf{x}}_r(t) \\ \ddot{\mathbf{x}}_r &= \ddot{\mathbf{x}}_r(t)\end{aligned}\tag{11.5}$$

Ennek alapján kigeneráljuk a csuklózváltozókat, megoldva az inverz geometriai (IG) és inverz kinematikai feladatokat minden időpillanatban:

$$\begin{aligned}\mathbf{q}_r &= IG(\mathbf{x}_r(t)) \\ \dot{\mathbf{q}}_r &= J^{-1}(\mathbf{q}_r)\dot{\mathbf{x}}_r \\ \ddot{\mathbf{q}}_r &= J^{-1}(\mathbf{q}_r)\left(\ddot{\mathbf{x}}_r(t) - \dot{J}(\mathbf{q}_r, \dot{\mathbf{q}}_r)\dot{\mathbf{q}}_r\right)\end{aligned}\tag{11.6}$$

A $\dot{J}(\mathbf{q}_r)$ mátrix számításánál a $J(\mathbf{q}_r)$ Jacobi-mátrix minden elemét deriváljuk időben. Figyelembe kell venni, hogy $\mathbf{q}_r$ időfüggő, és a deriválásnál kell alkalmazni a láncszabályt.

Példa: a 11.3. ábrán látható robot Jacobi-mátrixa:

$$J(\mathbf{q}_r) = \begin{pmatrix} -a_1 s_{q_1} - a_2 s_{q_{12}} & -a_2 s_{q_{12}} \\ a_1 c_{q_1} + a_2 c_{q_{12}} & a_2 c_{q_{12}} \end{pmatrix},\tag{11.7}$$

ahol $s_{q_1} = \sin(q_1)$, $c_{q_1} = \cos(q_1)$, $s_{q_{12}} = \sin(q_1 + q_2)$, $c_{q_{12}} = \cos(q_1 + q_2)$.

A mátrix deriváltja időben:

$$\dot{J}(\mathbf{q}_r, \dot{\mathbf{q}}_r) = \begin{pmatrix} -a_1 c_{q_1} \dot{q}_1 - a_2 c_{q_{12}} (\dot{q}_1 + \dot{q}_2) & -a_2 c_{q_{12}} (\dot{q}_1 + \dot{q}_2) \\ -a_1 s_{q_1} \dot{q}_1 - a_2 s_{q_{12}} (\dot{q}_1 + \dot{q}_2) & -a_2 s_{q_{12}} (\dot{q}_1 + \dot{q}_2) \end{pmatrix} \quad (11.8)$$

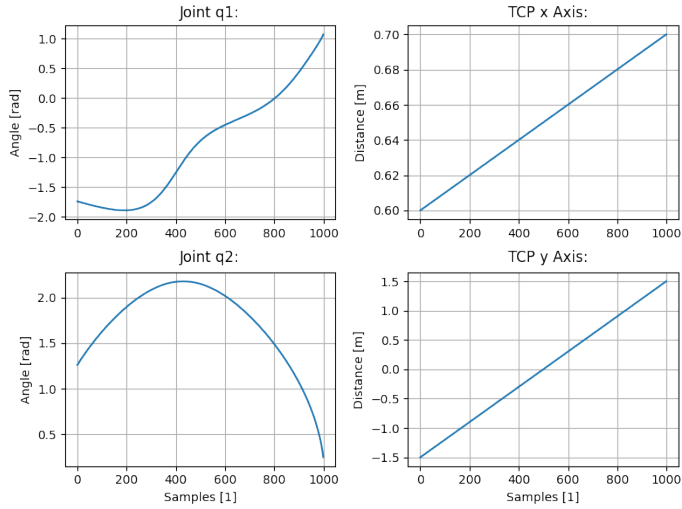
A világkoordinátákban történő pályatervezés esetén a pálya menti csuklósebességek és csuklógyorsulások számítására szükség van a Jacobi-mátrix inverzére. Tehát amikor a robot szinguláris konfigurációban van, vagyis a Jacobi-mátrix determinánsa zéró, nem tudunk pálya menti sebességet és gyorsulást számolni. A szinguláris konfigurációk közvetlen környezetében a Jacobi-mátrix determinánsa nagy értékeket vehet fel, ezért az inverz Jacobi-mátrixon alapuló sebességszámítások szinguláris konfigurációk környezetében valós robot beavatkozóval általában nem megvalósítható csuklósebességet eredményezhet.

A világkoordinátákban végzett pályatervezés számításigényes, ugyanis a pálya minden pontjára meg kell oldani az inverz geometriai és inverz kinematikai feladatokat. A gyakorlatban a pályagörbe pontokat és az ezekhez tartozó sebesség- és gyorsulásértékeket a robotszabályozó mintavételi periódusának megfelelő időbeli felbontással kell kiszámolni.

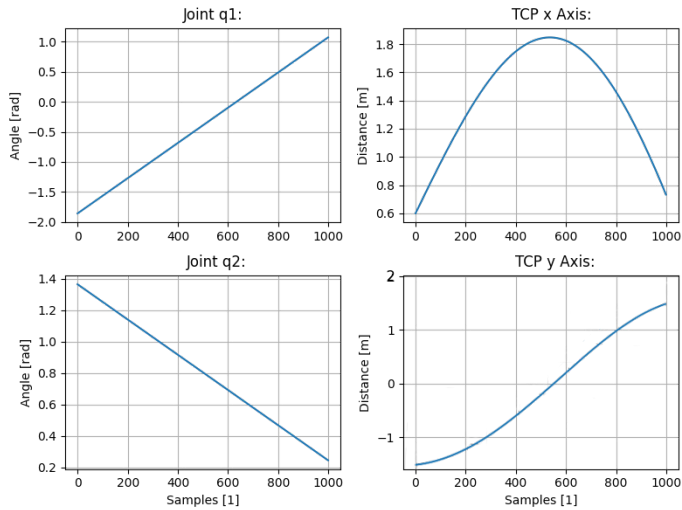
A világkoordinátákban tervezett pálya előnye, hogy két útkonfiguráció között a végberendezéses térbeli mozgását pontosan előírhatjuk (egyenes, kör, polinomokkal vagy trigonometrikus függvénnyel leírható görbe stb.).

Csuklókoordinátákban tervezett pálya esetén két útkonfiguráció között nem tudunk a végberendezésnek előírt mozgást biztosítani. Amennyiben előírjuk a pályát csuklókoordinátákban ($\mathbf{q}_r(t)$), a világkoordinátákban a végberendezés mozgását a direkt geometriai feladat (DG) adja meg ($\mathbf{x}_r(t) = DG(\mathbf{q}_r(t))$). Ugyanígy, a csuklókoordinátákban tervezett sebességpályának megfelelő végberendezés sebességkomponenseket a direkt kinematikai feladat adja ($\dot{\mathbf{x}}_r = J(\mathbf{q}_r)\dot{\mathbf{q}}_r$).

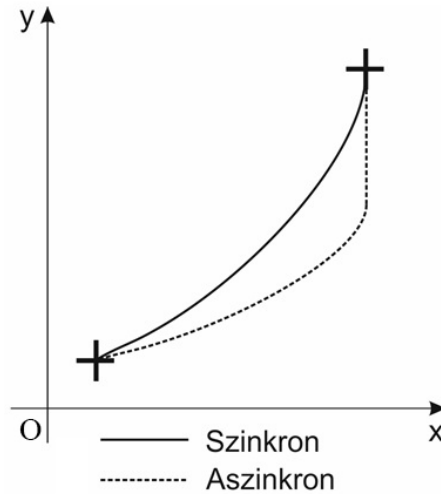
Példa: a 11.4. és 11.5. ábrák a világkoordinátákban és csuklókoordinátákban történő pályagörbéket hasonlítják össze a 11.3. ábrán látható robot esetében. A robotszegmensek hossza: $a_1 = 1\text{ m}$, $a_2 = 1\text{ m}$. A végberendezés kezdeti konfigurációja $x_1 = 0.6\text{ m}$, $y_1 = -1.5\text{ m}$. A megfelelő csuklózváltozók $q_{11} = -1.74\text{ rad}$, $q_{21} = 1.26\text{ rad}$. A végberendezés megérkezési konfigurációja $x_2 = 0.7\text{ m}$, $y_2 = 1.5\text{ m}$. A megfelelő csuklózváltozók $q_{12} = 1.07\text{ rad}$, $q_{22} = 0.25\text{ rad}$. Az ábrákon a végberendezés koordináták (x , y) és a csuklózváltozók (q_1 , q_2) láthatóak. Látszik, hogy ha a pálya csuklókoordinátákban egy egyenes szakasz, a végberendezés pályagörbéje nem lesz egyenes, a formáját a direkt geometria feladat adja meg.



11.4. ábra. *Lineáris pálya világkoordinátákban*



11.5. ábra. *Lineáris pálya csuklókoordinátákban*



11.6. ábra. Szinkron vs. aszinkron pálya

Szinkron és aszinkron pályák: A pályán való haladás során a robot végberendezése több tengely mentén mozog. Előfordulhat, hogy a pályaelem hossza valamelyik tengely mentén rövidebb, mint egy másik tengely mentén, így a mozgás, azonos sebességek mellett, e tengely mentén hamarabb véget ér. Az ilyen pályákat *aszinkron pályáknak* nevezzük.

Ha az a cél, hogy minden tengely mentén a mozgás ugyanabban az időpillanatban járjon le, a tengely menti sebességeket szinkronizálni kell. A szinkronizált sebességű pályákat *szinkron pályáknak* nevezzük.

A 11.6. ábra szinkron és aszinkron pályákat hasonlít össze két tengely esetén világkoordinátákban. Látszik, hogy aszinkron pálya esetén az x tengely mentén hamarabb befejeződik a mozgás.

11.3. A gyakorlat menete

A gyakorlat célja: írjunk robotprogramot, használva az UR3 (a robotot a 11.9. ábra szemlélteti) kezelőfelületét az alábbi két feladat elvégzésére:

1. Körpálya rajzolás adott pontban elhelyezett ceruzával.
2. Alkatrész-összeillesztés ismert kezdeti állapotokkal.

Robotok pályatanítása: robotprogramozás során a végberendezést két célpont közötti mozgásra bírjuk. Ezeket a célpontokat megadhatjuk előzetesen ismert vagy mért világkoordináták/csuklókonfigurációk alkalmazásával, vagy betaníthatjuk a pontokat.

A robotot szabad mozgás (free-drive) üzemmódban kézzel tudjuk mozgatni. Elvisszük a robotot a szükséges célpontba, majd lementjük az aktuális konfigurációnak megfelelő világkoordinátát (lineáris mozgás esetén) vagy csuklókonfigurációt (csuklómozgás esetén).

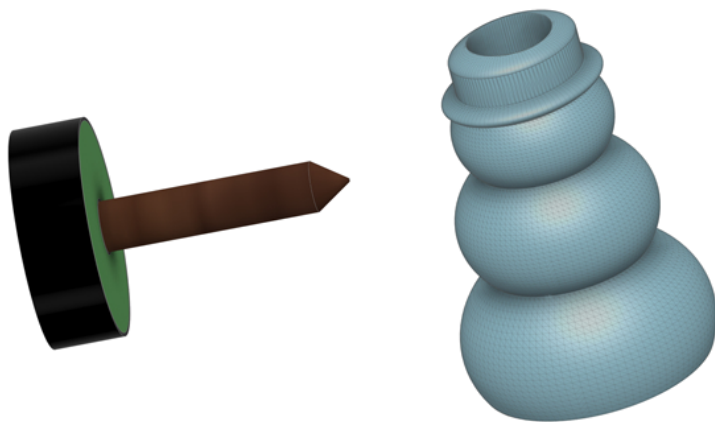
Hogyha a gravitációs beállítás vagy a végberendezés mechanikai paramétere nem tükrözi a valóságot, szabad mozgás alatt a robot elkezdhet magától mozogni (nem működik a gravitációkompenzálás). Ebben az esetben leállítjuk a szabad mozgást, és újrakonfiguráljuk a mechanikai paramétereiket.

Körpálya mozgássorozat: adott egy foglalatba helyezett írószer, lásd 11.7. ábra. A célunk, hogy rajzoljunk az írószerrel egy kört, majd helyezzük vissza a foglalatba.

1. A robot megközelíti a foglalatban elhelyezett ceruzát.
2. A robot megfogja a ceruzát.
3. A robot kiemeli a ceruzát a foglalatból.
4. A robot ráhelyezi a ceruzát a kijelölt kezdeti pontra.
5. A robot 4 ponton keresztül megrajzol egy kört, felhasználva a *moveP* utasítást.
6. A robot megközelíti a ceruza foglalatot.
7. A robot leereszti a ceruzát a foglalatba.
8. A robot elengedi a ceruzát.
9. A robot felemelkedik a ceruzától.
10. A robot visszatér a kezdőpozícióba.

Összeillesztés mozgássorozat: adott egy bemélyedést tartalmazó kocka és egy kulcs alkatrész, lásd 11.8. ábra. A célunk, hogy a kockát mozgatva belehelyezzük a kulcsot.

1. A robot megközelíti a kockát úgy, hogy a végberendezés z tengelye a bemélyedésre mutasson.

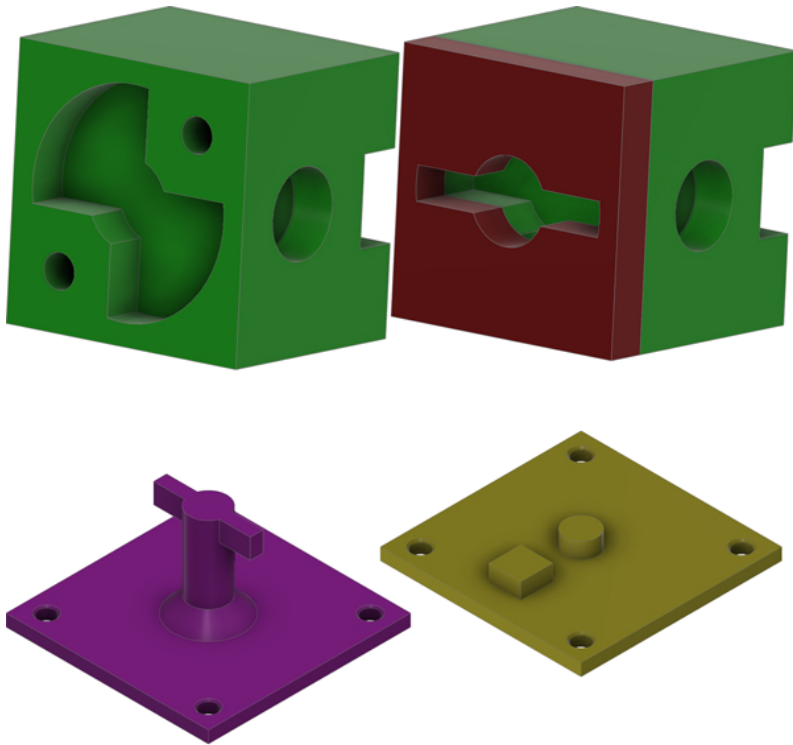


11.7. ábra. *Munkatárgyak a körpálya mozgássorozathoz*

2. A robot megfogja a kockát.
3. A robot megközelíti a kulcsot.
4. A robot ráhelyezi a kockát a kulcsra, korlátozva a kifejtett erőt *force* paranccsal.
5. A robot 90 fokkal elforgatja a kockát.
6. A robot elengedi a kockát.
7. A robot visszatér a kezdőpozícióba.

11.4. Kérdések és feladatok

1. Mit értünk kör alatt? Hogyan nézne ki egy kör, ha a teret 1-es vagy végtelen normából származott metrikával látnuk el?
2. Hogyan tudjuk a kört kirajzolni csupán lineáris mozgássorozattal?
3. A *moveP* utasításban megadhatunk via pontokat és körmozgás-szekvenciákat. Mi a különbség egy csupán körmozgás és egy csupán viapont-sorozat között?



11.8. ábra. Munkatárgyak az összeillesztéshez



11.9. ábra. *A laborfeladatokat végrehajtó UR3 robotkar*

ROBOTMANIPULÁTOROK PID SZABÁLYOZÁSA

12.1. A gyakorlat célja

Robotmanipulátorok PID szabályozásának vizsgálata, PID szabályozás tervezése többszabadságfokú robotkarokon szimulációs környezetben.

12.2. Elméleti bevezető

12.2.1. Robotok szabályozása

A robotok szabályozója a robotirányítási rendszer része, amely a robotcsukló beavatkozó által kifejtendő erőket/nyomatékokat számítja ki az előírt mozgás és a mért csukópozíciók, csuklósebességek függvényében.

Az előírt mozgást általában az előírt csuklókonfigurációval ($\mathbf{q}_r$), csuklósebességgel és csuklógyorsulással adjuk meg, amit a robotirányítási rendszer pályatervezője számít ki:

$$\begin{aligned}\mathbf{q}_r &= \mathbf{q}_r(t) \\ \dot{\mathbf{q}}_r &= \dot{\mathbf{q}}_r(t) \\ \ddot{\mathbf{q}}_r &= \ddot{\mathbf{q}}_r(t)\end{aligned}\tag{12.1}$$

A szabályzó tervezéséhez és vizsgálatához a robot dinamikus modelljéből indulunk ki.

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = \boldsymbol{\tau},\tag{12.2}$$

ahol $M(\mathbf{q})$ a robot tömegmátrixa, $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ a centrifugális és Coriolis erőket leíró vektor, $G(\mathbf{q})$ a gravitáció hatását leíró erő/nyomaték vektor, $\boldsymbol{\tau}$ a beavatkozó által generált erők/nyomatékok vektora.

A modell az alábbi tulajdonságokkal rendelkezik:

- A tömegmátrix szimmetrikus: $M(\mathbf{q}) = M(\mathbf{q})^T, \forall \mathbf{q}$.
- A tömegmátrix pozitív definit: $M(\mathbf{q}) > 0, \forall \mathbf{q}$.

- A $\dot{M}(\mathbf{q}) - 2C(\mathbf{q}, \dot{\mathbf{q}})$ mátrix ferdén szimmetrikus, vagyis:

$$\mathbf{x}^T \left(\dot{M}(\mathbf{q}) - 2C(\mathbf{q}, \dot{\mathbf{q}}) \right) \mathbf{x} = 0, \forall \mathbf{x}. \quad (12.3)$$

A robot *szabad mozgást* végez, ha nincs kapcsolatban a környezetével, a végberendezés nem fejt ki erőhatást a környezetre, illetve a környezet nem befolyásolja a robot mozgását a végberendezésre ható erőkkel/nyomatékokkal. Ez alól kivétel az anyagmozgatás, amikor a robot által mozgatott céltárgyra ható gravitációs erő, valamint a céltárgy inerciális paraméterei megváltoztatják a robot dinamikus viselkedését. Ugyanakkor a robot dinamikus viselkedése ebben az esetben is az (12.2) egyenletben megadott modellel leírható, az egyenlet tagjai tartalmazzák a céltárgy paramétereit.

Szabad mozgás szabályozása esetén kétféle stratégiát különböztetünk meg:

- Ponttól pontig (*PTP – Point to Point*) szabályozás: ebben az esetben az előírt csuklókonfigurációk (pontok) konstansok. A robot előírt konfigurációk sorozatán halad végig. Ennek megfelelően a szabályozó alapjele szakaszosan konstans. Két előírt konfiguráció között a robot pályájának formáját nem írjuk elő. A *PTP* mozgást megvalósító szabályozók implementálási költségei tipikusan alacsonyak (kevés mérés szükséges, kevés információ szükséges a robotmodellről, könnyen megvalósíthatóak számítástechnikai eszközökön).
- Pályakövetés (*Tracking*): pályakövetés esetén előírjuk a csuklókonfigurációt, csuklósebességet és a csuklógyorsulást is a mozgás minden pillanatában. Az előírt csuklókonfiguráció tipikusan egy időben változó sima függvény. A szabályozó megvalósításához szükséges mérni a csuklósebességet is, a pályatervező képes kell legyen az előírt pozíció, sebesség és gyorsulás kigenerálására a mozgás minden pillanatában.

A robotmanipulátorok szabályozóinak zöme pályakövetést valósít meg. A *PTP* mozgást tipikusan csak kis költségű alkalmazásoknál vagy a robot tesztelésénél, üzembe helyezésénél alkalmazzuk.

A robot *kényszermozgást* végez, ha a feladatvégzés közben kapcsolatba kerül a környezetével. A környezet erőt/nyomatékot fejt ki a robotra. Tipikus robotikai feladatok, amelyben erőszabályozást is kell végezni, a megmunkálási feladatok. Ilyen alkalmazásoknál a robotirányítási rendszer képes kell legyen *erőszabályozást* végezni, vagyis végberendezéssel a munkatér bizonyos irányába előírt erőt, nyomatékot kifejezni.

12.2.2. PD+G szabályozás

Ezzel a szabályozási stratégiával *PTP* mozgást valósíthatunk meg. Nem írjuk elő a robot sebességét és gyorsulását, a referencia csuklókonfigurációt konstansnak tekintjük:

$$\mathbf{q}_r \text{ konstans.} \quad (12.4)$$

A szabályozás célja annak biztosítása, hogy:

$$\lim_{t \rightarrow \infty} \underbrace{(\mathbf{q}_r - \mathbf{q}(t))}_{\mathbf{e}(t)} = \mathbf{0} \quad (12.5)$$

ahol $\mathbf{e}(t) = (e_1(t) \ e_2(t) \ \dots \ e_n(t))^T$ a szabályozási hiba vektora, minden csukló szabályozási hibáját tartalmazza.

A beavatkozó jelet PD szabályozó számítja ki, kiterjesztve a gravitációt kompenzáló komponenssel. Az i -ik csuklóra a szabályozó formája:

$$\tau_i = K_{pi}(e_i(t) + T_{di}\dot{e}_i(t)) + G_i(\mathbf{q}) \quad (12.6)$$

ahol $i = 1 \dots n$, n a robot szabadságfoka. $K_{pi} > 0$ a szabályozó proporcionális erősítése, $T_{di} > 0$ a szabályozó deriválási ideje, $G_i(\mathbf{q})$ a robot dinamikus modelljében szereplő $G(\mathbf{q})$ gravitációs vektor i -ik eleme.

A teljes robotra felírva a PD+G szabályozó alakja:

$$\boldsymbol{\tau} = K_p(\mathbf{e}(t) + T_d\dot{\mathbf{e}}(t)) + \mathbf{G}(\mathbf{q}) \quad (12.7)$$

ahol $K_p = \text{diag}(K_{p1} \ K_{p2} \ \dots \ K_{pn})$, $T_d = \text{diag}(T_{d1} \ T_{d2} \ \dots \ T_{dn})$. $\text{diag}(\cdot)$ diagonális mátrixot jelöl.

Példa: a 12.1. ábrán látható az egyszabadságfokú mechanikai rendszer dinamikáját leíró modell:

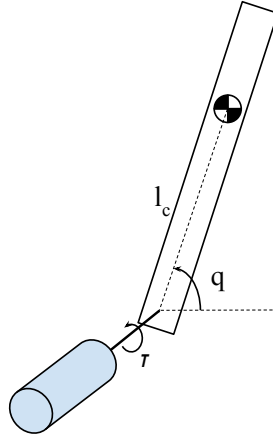
$$J\ddot{q} + mgl_c \cos(q) = \tau \quad (12.8)$$

ahol q a szögelfordulás, τ a csuklóbeavatkozó által kifejtett nyomaték, m a mozgatott szegmens tömege, J a mozgatott szegmens inerciája, l_c a szegmens tömegközéppontjának távolsága a csuklóbeavatkozó tengelyétől.

A PD+G szabályozó alakja a (12.8) modell esetén:

$$e = q_r - q \quad (12.9)$$

$$\tau = K_p(e + T_d\dot{e}) + mgl_c \cos(q) \quad (12.10)$$



12.1. ábra. A robot útja és pályagörbéje

12.2.3. A szabályozás analízise Lyapunov stabilitástétellel

A $PD + G$ szabályozóval irányított robot vizsgálatához a Lyapunov-tételt alkalmazzuk.

Dinamikus rendszerek egysúlyi állapota: legyen egy dinamikus rendszer n dimenziós időfüggő állapotának időbeli változását ($\mathbf{x}(t)$) leíró modell:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad (12.11)$$

ahol $\mathbf{f}(\cdot)$ folytonos.

Az $\mathbf{x}^*$ a rendszer egyensúlyi állapota, ha $\mathbf{x}^* = \mathbf{0}$. Látszik, hogy a rendszer egyensúlyi állapotait úgy kaphatjuk meg, hogy megoldjuk az $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ egyenletet.

A továbbiakban feltételezzük, hogy $\mathbf{x}^* = \mathbf{0}$ egyensúlyi állapota a (12.11) rendszernek.

Stabil egysúlyi állapot: Az $\mathbf{x}^* = \mathbf{0}$ egyensúlyi állapot *aszimptotikusan stabil* a $t = t_0$ pillanatban, ha létezik $r > 0$ véges pozitív érték úgy, hogy ha $\|\mathbf{x}(t_0)\| \leq r$, akkor $\lim_{t \rightarrow \infty} \|\mathbf{x}(t)\| = 0$. $\|\cdot\|$ az euklideszi normát jelöli.

Lyapunov-tétel: a (12.11) rendszer $\mathbf{x}^* = \mathbf{0}$ egyensúlyi állapota aszimptotikusan stabil, ha létezik egy V Lyapunov (energia) függvény az alábbi

tulajdonságokkal:

$$\begin{aligned} V(\mathbf{x}) &> 0, \text{ ha } \mathbf{x} \neq \mathbf{0}, \\ V(\mathbf{x}) &= 0, \text{ ha } \mathbf{x} = \mathbf{0}, \\ \dot{V}(\mathbf{x}) &< 0, \text{ ha } \mathbf{x} \neq \mathbf{0}. \end{aligned} \quad (12.12)$$

A szabályozott robothoz rendelt energiafüggvény: a robot állapotait a csuklókonfigurációk ($\mathbf{q}$) és csuklósebességek ($\dot{\mathbf{q}}$) adják. Legyen a Lyapunov-függvény:

$$V = \frac{1}{2} \dot{\mathbf{q}}^T M(\mathbf{q}) \dot{\mathbf{q}} + \frac{1}{2} \mathbf{e}^T K_p \mathbf{e} \quad (12.13)$$

Az energiafüggvény első tagja a robot mozgási energiája, a második tag a pozíciószabályozás energiája. Mind a két tag pozitív. Látszik, hogy ha a robot a referencia csuklókonfigurációban van ($\mathbf{e} = \mathbf{q}_r - \mathbf{q} = \mathbf{0}$), és a csuklósebességek nullák, akkor a Lyapunov-függvény értéke nulla.

A Lyapunov-függvény deriváltjának számítása:

$$\begin{aligned} \dot{V} &= \frac{1}{2} \ddot{\mathbf{q}}^T M(\mathbf{q}) \dot{\mathbf{q}} + \frac{1}{2} \dot{\mathbf{q}}^T \dot{M}(\mathbf{q}) \dot{\mathbf{q}} + \frac{1}{2} \dot{\mathbf{q}}^T M(\mathbf{q}) \ddot{\mathbf{q}} \\ &\quad + \frac{1}{2} \dot{\mathbf{e}}^T K_p \mathbf{e} + \frac{1}{2} \mathbf{e}^T K_p \dot{\mathbf{e}} \end{aligned} \quad (12.14)$$

Kihasználva, hogy bármely szimmetrikus A mátrix esetén igaz, hogy $\mathbf{x}^T A \mathbf{y} = \mathbf{y}^T A \mathbf{x}$, valamint azt, hogy a referencia csuklókonfiguráció konstans ($\dot{\mathbf{e}} = -\dot{\mathbf{q}}$), következik:

$$\dot{V} = \dot{\mathbf{q}}^T M(\mathbf{q}) \ddot{\mathbf{q}} + \frac{1}{2} \dot{\mathbf{q}}^T \dot{M}(\mathbf{q}) \dot{\mathbf{q}} - \dot{\mathbf{q}}^T K_p \mathbf{e} \quad (12.15)$$

Az $M(\mathbf{q}) \ddot{\mathbf{q}}$ tagot a robot (12.2) dinamikus modelljéből kapjuk, ahol a τ beavatkozó jeleket a (12.7) egyenletben megadott PD+G szabályozó számítja:

$$M(\mathbf{q}) \ddot{\mathbf{q}} = -C(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} - G(\mathbf{q}) + \tau \quad (12.16)$$

$$M(\mathbf{q}) \ddot{\mathbf{q}} = -C(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} + K_p \mathbf{e}(t) - K_p T_d \dot{\mathbf{q}} \quad (12.17)$$

Behelyettesítve a Lyapunov-függvény deriváltjába kapjuk:

$$\dot{V} = \dot{\mathbf{q}}^T (-C(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} + K_p \mathbf{e}(t) - K_p T_d \dot{\mathbf{q}}) + \frac{1}{2} \dot{\mathbf{q}}^T \dot{M}(\mathbf{q}) \dot{\mathbf{q}} - \dot{\mathbf{q}}^T K_p \mathbf{e} \quad (12.18)$$

Kihasználva a (12.3) tulajdonságot, kapjuk:

$$\dot{V} = -\dot{\mathbf{q}}^T K_p T_d \dot{\mathbf{q}} \quad (12.19)$$

Látszik, hogy $\dot{V} < 0$ ha $\dot{\mathbf{q}} \neq 0$, tehát a Lyapunov-függvény mindaddig csökken, amíg a robot mozgásban van. A Lyapunov-tétel értelmében a rendszer aszimptotikusan stabil. Amikor a mozgás megáll $\dot{\mathbf{q}} = \mathbf{0}$, a gyorsulás is nullává válik $\ddot{\mathbf{q}} = \mathbf{0}$. Ebben az esetben a szabályozott robot dinamikus modellje egyensúlyi állapotban:

$$M(\mathbf{q}) \underbrace{\ddot{\mathbf{q}}}_{=0} = -C(\mathbf{q}, \dot{\mathbf{q}}) \underbrace{\dot{\mathbf{q}}}_{=0} + K_p \mathbf{e}(t) - T_d \underbrace{\dot{\mathbf{q}}}_{=0} \quad (12.20)$$

Tehát $K_p \mathbf{e} = \mathbf{0}$, a pozíciósabályozási hiba is konvergál a nullába.

12.2.4. PID+G szabályozás

Nagy pontosságú robotszabályozásoknál figyelembe kell venni a beavatkozó motorok és a robotszegmens közötti áttételekben fellépő súrlódási hatásokat is. A súrlódás által generált fékezőnyomaték a robot dinamikus modelljében az alábbi formában jelenik meg:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = \tau - \tau_f \quad (12.21)$$

ahol τ_f a súrlódás által létrehozott erők/nyomatékok vektora.

A száraz (Coulomb)+viszkózus súrlódási hatást feltételezve az alábbi súrlódási modellt alkalmazzuk:

$$\tau_{fi} = F_{Ci} \text{sign}(q_i) + F_{Vi} \dot{q}_i \quad (12.22)$$

ahol F_{Ci} a száraz (Coulomb) súrlódási együttható, F_{Vi} a viszkózus súrlódási együttható. $\text{sign}(\cdot)$ az előjel függvényt jelöli.

A száraz súrlódási együttható konstans vagy lassan változó terhelésként jelenik meg a rendszerben, az értéke nehezen mérhető. A pontos pozíciószabályozáshoz statikus súrlódás hatását kompenzálni kell. Ehhez olyan szabályozót alkalmazhatunk, amely tartalmaz integráló tagot.

A PID+G robotszabályozó az i -ik csuklóra:

$$\tau_i = K_{pi} \left(e_i(t) + T_{di} \dot{e}_i(t) + \frac{1}{T_{Ii}} \int_0^t e_i(\xi) d\xi \right) + G_i(\mathbf{q}), \quad (12.23)$$

ahol $T_{Ii} > 0$ a szabályozó integrálási ideje.

12.2.4.1. A szabályozás analízise a referencia konfiguráció környezetében

A szabályozott robot vizsgálatához feltételezzük, hogy a robot „közel” van a referencia csuklókonfigurációhoz. Az i -ik szabályozott csukló modellje:

$$M_{ii}(\mathbf{q})\ddot{q}_i + \underbrace{\sum_{j \neq i, j=1}^n M_{ij}(\mathbf{q})\ddot{q}_j + \sum_{j=1}^n C_{ij}(\mathbf{q}, \dot{\mathbf{q}})\dot{q}_j}_{\delta(t)} + G_i(\mathbf{q}) = \quad (12.24)$$

$$\underbrace{K_{pi} \left(e_i(t) + T_{di}\dot{e}_i(t) + \frac{1}{T_{Ii}} \int_0^t e_i(\xi) d\xi \right)}_{\tau_i} + G_i(\mathbf{q}) - F_{Ci} \text{sign}(q_i) - F_{Vi}\dot{q}_i$$

A konstans referencia állapot körüli vizsgálat során a modell azon tagjait, amelyek függenek a többi csukló sebességétől és gyorsulásától ($\delta(t)$), elhanyagoljuk:

$$M_{ii}(\mathbf{q})\ddot{q}_i = K_{pi} \left(e_i(t) + T_{di}\dot{e}_i(t) + \frac{1}{T_{Ii}} \int_0^t e_i(\xi) d\xi \right) - F_{Ci} \text{sign}(q_i) - F_{Vi}\dot{q}_i$$

Figyelembe véve, hogy $\dot{e}_i(t) = -\dot{q}_i(t)$, és alkalmazva az $\mathbf{q} \approx \mathbf{q}_r$ approximációt kapjuk:

$$M_{ii}(\mathbf{q}_r)\ddot{e}_i + (F_{Vi} + K_{pi}T_{di})\dot{e}_i(t) + K_{pi}e_i(t) + \frac{K_{pi}}{T_{Ii}} \int_0^t e_i(\xi) d\xi = F_{Ci} \text{sign}(q_i)$$

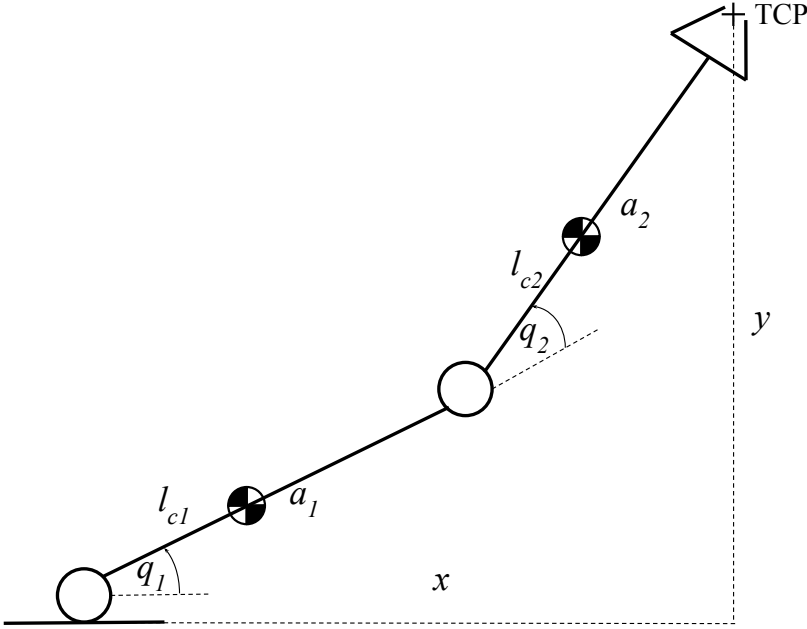
Látszik, hogy konstans Coulomb súrlódási komponenst feltételezve, a statikus súrlódás által létrehozott nyomaték konstans bemeneti zavarként tárgyalható, amelynek hatását a szabályozó integráló komponense kompenzálni képes.

A linearizált szabályozott robotmodell pólusait az alább karakterisztikus egyenlet adja:

$$s^3 + \frac{F_{Vi} + K_{pi}T_{di}}{M_{ii}(\mathbf{q}_r)}s^2 + \frac{K_{pi}}{M_{ii}(\mathbf{q}_r)}s + \frac{K_{pi}}{T_{Ii}M_{ii}(\mathbf{q}_r)} = 0 \quad (12.25)$$

A Ruth–Hurwitz stabilitási kritérium értelmében a PID+G robotszabályozás stabilitásához a referencia konfiguráció környezetében a szabályozó paramétereiket az alábbi összefüggés szerint kell megválasztani:

$$T_{Ii} > \frac{M_{ii}(\mathbf{q}_r)}{F_{Vi} + K_{pi}T_{di}} \quad (12.26)$$



12.2. ábra. RR robot általános merev szegmensekkel

12.3. A gyakorlat menete

A *gyakorlat célja*: kétszabadságfokú RR robotkar modellezése és PTP szabályozása, felhasználva a MATLAB/Simulink Simscape Multibody által biztosított grafikus programozási környezetet.

1. Egy *slx* állományban modellezzük le a robotunkat Simscape Multibody tömbökkel.
2. Lássuk el a rendszert PID szabályozóval, amely konstans referenciát követ.

RR robot dinamikus modellje:

Tekintsük a 12.2. ábrán levő RR robotot. Írjuk fel a robot dinamikus modelljét Euler–Lagrange-egyenletekkel: Legyen a csuklóváltozó vektor $\mathbf{q} = (\theta_1 \quad \theta_2)^\top$ és a csuklók sebesség vektora $\dot{\mathbf{q}} = (\dot{\theta}_1 \quad \dot{\theta}_2)^\top$. Az első tömegközéppont szögsebessége:

$$\omega_{c1} = \dot{q}_1,$$

illetve egyenesvonalú sebesség komponensei a K_1 -es vonatkoztatási rendszerben:

$$\begin{aligned} v_{c_1x} &= 0 \\ v_{c_1y} &= l_{c1}\dot{q}_1 \end{aligned}$$

A második tömegközéppont szögsebessége:

$$\omega_{c_2} = \dot{q}_1 + \dot{q}_2,$$

a hozzátartozó egyenesvonalú sebesség komponensek a K_2 -es vonatkoztatási rendszerben:

$$\begin{aligned} v_{c_2x} &= l_1\dot{q}_1 \sin(q_2) \\ v_{c_2y} &= l_1\dot{q}_1 \cos(q_2) + l_{c_2}(\dot{q}_1 + \dot{q}_2) \end{aligned}$$

A helyzeti energia:

$$\begin{aligned} P_1 &= m_1 g l_{c_1} \sin(q_1) \\ P_2 &= m_2 g [l_1 \sin(q_1) + l_{c_2} \sin(q_1 + q_2)] \end{aligned}$$

A robot mozgási energiája:

$$\begin{aligned} K_1 &= \frac{1}{2}m_1(v_{c_1x}^2 + v_{c_1y}^2) + \frac{1}{2}J_1\omega_{c_1}^2 = 0.5J_1\dot{q}_1^2 + 0.5l_{c_1}^2m_1\dot{q}_1^2 \\ K_2 &= \frac{1}{2}m_2(v_{c_2x}^2 + v_{c_2y}^2) + \frac{1}{2}J_2\omega_{c_2}^2 \\ &= 0.5J_2(\dot{q}_1 + \dot{q}_2)^2 + 0.5m_2(l_1^2 \sin^2(q_1)\dot{q}_1^2 + (l_1 \cos(q_1)\dot{q}_1 + l_{c_2}(\dot{q}_1 + \dot{q}_2))^2) \end{aligned}$$

A Lagrange-operátor:

$$L := L(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{i=1}^2 (K_i - P_i).$$

A robot dinamikus modelljét az

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} = \boldsymbol{\tau}.$$

Euler–Lagrange-egyenlettel kapjuk meg.

Az első csuklóhoz tartozó dinamikus egyenlet:

$$\begin{aligned} & (J_1 + J_2 + l_1^2 m_2 + 2l_1 l c_2 m_2 \cos q_2 + l c_1^2 m_1 + l c_2^2 m_2) \ddot{q}_1 \\ & + (J_2 + l_1 l c_2 m_2 \cos q_2 + l c_2^2 m_2) \ddot{q}_2 \\ & - l_1 l c_2 m_2 \sin q_2 (\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2) + g l_1 m_2 \cos q_1 + g l c_1 m_1 \cos q_1 \\ & + g l c_2 m_2 \cos (q_1 + q_2) = \tau_1 \end{aligned}$$

A második csuklóhoz tartozó dinamikus egyenlet:

$$\begin{aligned} & (J_2 + l c_2 m_2 l_1 \cos q_2 + l c_2^2 m_2) \ddot{q}_1 \\ & + (J_2 + l c_2^2 m_2) \ddot{q}_2 \\ & + l c_2 m_2 l_1 \sin q_2 \dot{q}_1^2 + g l c_2 m_2 \cos (q_1 + q_2) = \tau_2 \end{aligned}$$

Az egyenletet csoportosítjuk:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) + G(\mathbf{q}) = \tau,$$

ahol az egyes komponensek:

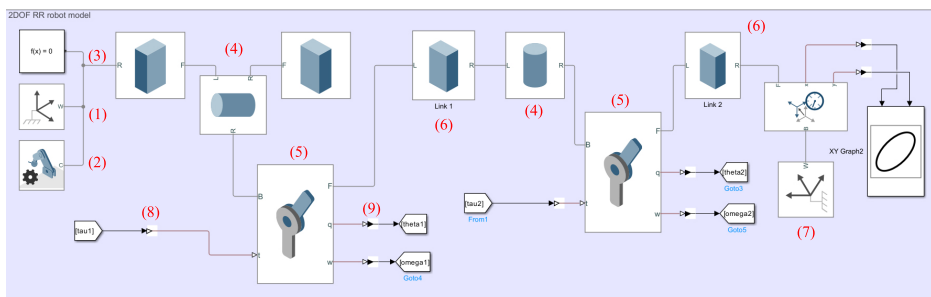
$$\begin{aligned} M(\mathbf{q}) &= \begin{pmatrix} J_1 + J_2 + l_1^2 m_2 + 2l_1 l c_2 m_2 c_{q_2} + l c_1^2 m_1 + l c_2^2 m_2 & J_2 + l_1 l c_2 m_2 c_{q_2} + l c_2^2 m_2 \\ J_2 + l_1 l c_2 m_2 c_{q_2} + l c_2^2 m_2 & J_2 + l c_2^2 m_2 \end{pmatrix} \\ C(\mathbf{q}, \dot{\mathbf{q}}) &= \begin{pmatrix} -l_1 l c_2 m_2 s_{q_2} (\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2) \\ l c_2 m_2 l_1 s_{q_2} \dot{q}_1^2 \end{pmatrix} \\ G(\mathbf{q}) &= \begin{pmatrix} g l_1 m_2 c_{q_1} + g l c_1 m_1 c_{q_1} + g l c_2 m_2 c_{q_{12}} \\ g l c_2 m_2 c_{q_{12}} \end{pmatrix} \end{aligned}$$

Ahol $c_{q_1} = \cos q_1$, $s_{q_1} = \sin q_1$, $c_{q_2} = \cos q_2$, $s_{q_2} = \sin q_2$, $c_{q_{12}} = \cos q_1 + q_2$.

Simscape Multibody alapismeretek és robotmodell:

A Simulink egy MATLAB-alapú grafikus programozási környezet, mely lehetővé teszi multidoméniumos dinamikus rendszerek modellezését, szimulációját és elemzését. A klasszikus Simulink rendszerekben a modellünket egyszerűbb alrendszerek összekapcsolásával építjük fel. Az alrendszerek tartalmazhatnak adatforrást, adatnyelőt vagy módosító tömböt.

A Simscape Multibody egy MATLAB-eszköz, amely többtestű szimulációs környezetet biztosít 3D mechanikai rendszerek, mint például robotok, járművek modellezésére és elemzésére. Emellett a felhasználók hidraulikus,



12.3. ábra. Az *RR robot mechanikai lánc* Simulink alatt

elektromos, pneumatikus és egyéb fizikai rendszereket is integrálhatnak modelljeikbe a Simscape termékcsaládból származó komponensek segítségével.

Feladat: modellezzük a fent bemutatott RR robotot, tudva, hogy a szegmenseket 100 cm x 15 cm x 5 cm merev alumíniumrudakkal közelítjük meg. A csuklókat nyomtatékkal vezéreljük, és mérni szeretnénk a csuklószöget és sebességet.

A robotmodell mechanikai láncát a 12.3. ábra szemlélteti.

A modellünk elemeit egy referencia koordináta-rendszerben építjük fel, ez a világ koordináta-rendszer (*World Frame* (1)). Szükségünk van egy mechanizmus beállítás tömbre (*Mechanism Configuration* (2)), melyben beállíthatjuk a gravitációs gyorsulás irányát és mértéket. Mivel a Simulink végső soron a dinamikus modell megoldását numerikusan közelíti meg, ezért egy numerikus beállítást tartalmazó tömböt (*Solver Configuration* (3)) is csatolnunk kell a rendszerhez.

Az első csukló rögzítését grafikusán két téglatesttel (15 cm x 15 cm x 5 cm) és egy hengerrel (sugár 5 cm, magasság 6 cm) jelöljük (4). Ezek nem játszanak szerepet a szimulációban, csupán a grafikus megjelenítésben.

A rotációs csuklót *Revolute Joint*-ként kapjuk meg. Mivel mi akarjuk szabályozni a csuklót, az Actuation/Torque paramétert állítsuk át Provided by Input-ra.

A Sensing menü alatt engedélyezzük a szögelfordulást (position), és a szögsebességet (velocity) (5).

A második csuklót grafikusán egy hengerrel jelöljük (sugár 5 cm, magasság 6 cm) (6).

Ahhoz, hogy a végberendezés koordinátáit kiolvassuk a modellből, kössünk egy transzformációs érzékelőt a második szegmens után (7). A referencia legyen a bázis koordináta-rendszer, és térítsük vissza a végberendezés pozícióját a világ koordináta-rendszerben.

Mivel a modellünk Simscape tömbökkel készült, és a szabályozást alap Simulink tömbökkel valósítjuk meg, szükséges jelátalakítókat használni a ki- és bemenetek csatolására (Simulink-PS converter (8), PS-Simulink converter (9)).

A rendszer átláthatósága miatt használjunk Goto/From címke párosokat azonos névvel.

PID szabályozás: Tekintsünk konstans $(\pi/4 \quad \pi/4)^\top$ referencia csuklószöveget. Alkalmazzunk egy PID szabályozót az 1-es csuklóra és egy PD szabályozót a 2-es csuklóra. Hangolási lépések:

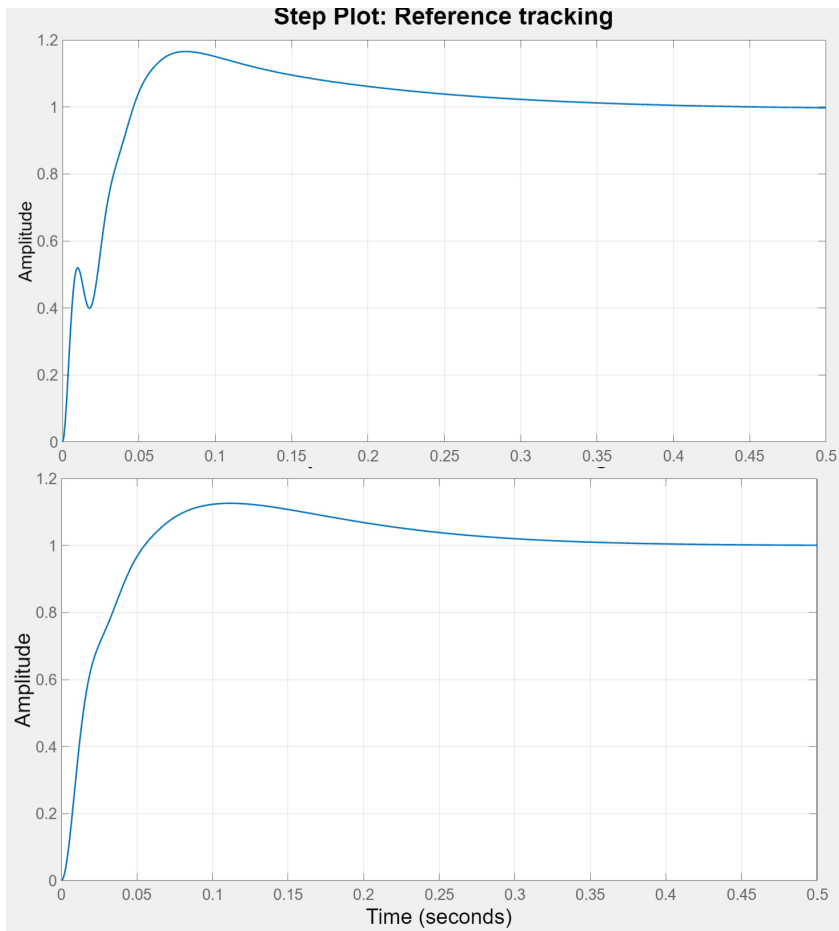
1. Hangoljuk be az 1-es hurkot a MATLAB beépített hangolójával közepes gyorsaságú (0.5 s), robusztus (0.6%) karakterisztikára.
2. Hangoljuk be a 2-es hurkot a MATLAB beépített hangolójával gyors (0.05 s), robusztus (0.6%) karakterisztikára.
3. Végül hangoljuk újra az 1-es hurkot, gyorsítva a válasz időt (0.05 s).

A hangolások után kapott átmeneti függvényeket a 12.4. ábra szemlélteti.

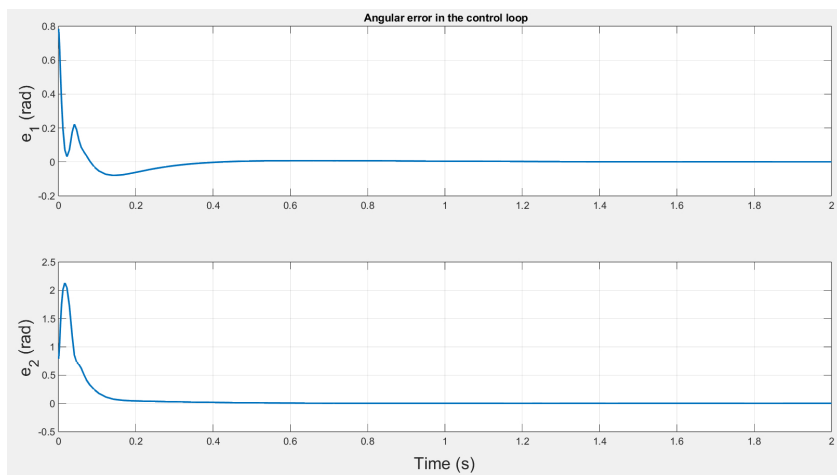
A hangolás után futtassuk le a szimulációt. Helyes modell és hangolás esetén a 12.5. ábrához hasonló viselkedést kapunk.

12.4. Kérdések és feladatok

1. Csak viszkózus súrlódási modellt feltételezve $\tau_{fi} = F_{Vi}\dot{q}_i$, igazoljuk Lyapunov-analízissel, hogy a PD+G szabályozó ugyanúgy biztosítja, hogy a pozíciószabályozási hiba konvergál a nullába, mint abban az esetben, amikor a súrlódás elhanyagolható.
2. Egészítsük ki a robotmodellt viszkózus és Coulomb-féle súrlódással, tudva azt, hogy a Coulomb-féle súrlódási együttható 1.7 Nm , a viszkózus súrlódási együttható 4.2 Nm s/rad .
3. Vizsgáljuk meg a szabályozott csuklók gyorsulás profilját. Hogyan lehetne korlátozni a PID/PD szabályozókon keresztül a csuklók gyorsulását? Futtassuk újra a szimulációt.



12.4. ábra. A hangolt PID (felső ábra) és PD (alsó ábra) szabályozókkal ellátott zárt rendszer átmeneti függvényei



12.5. ábra. *A szabályozási hiba ábrázolva az idő függvényében*

KISZÁMÍTOTT NYOMATÉKOK MÓDSZERE

13.1. A gyakorlat célja

Kiszámított nyomatékok módszerének bemutatása, robotmanipulátorok pályakövetést megvalósító irányítása szimulációs környezetben a kiszámított nyomatékok módszerét alkalmazva.

13.2. Elméleti bevezető

13.2.1. Kiszámított nyomatékok módszere csuklókoordinátákban

Kiszámított nyomatékok módszere (Computed Torque Method – *CTM*) egy szabályozási eljárás, amely a robotmanipulátorok pályakövetést megvalósító irányítására alkalmazható. A robot végberendezése előírt térbeli mozgást kell végrehajtson. Az előírt végberendezés mozgásnak megfelelően a pályatervező kiszámítja a csuklókonfigurációkat és az ennek megfelelő csuklósebességeket, csuklógyorsulásokat.

Az algoritmus feltételezi, hogy a robot dinamikus modellje pontosan ismert. A dinamikus modell tagjai, $M(\mathbf{q})$ – tömegmátrix, $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ – centrifugális és Coriolis-erők vektora, $G(\mathbf{q})$ – a gravitációs hatás vektora, a csuklókonfigurációktól, csuklósebességektől függenek, tehát a számításukhoz ezeket mérni kell. A dinamikus modell paraméterei – szegmensek tömege (m_i), inerciája (I_i), a tömegközéppont helye (l_{ci}), a robot mechanikai tervezése közben meghatározhatók.

A szabályozási feladat: Legyen a robot előírt csuklópozíció pályagörbéje $\mathbf{q}_r = \mathbf{q}_r(t)$ sima, legalább kétszer deriválható függvény. Számítsuk ki a robot beavatkozóji által kifejtett nyomatékokat/erőket, amelyek biztosítják, hogy $\lim_{t \rightarrow \infty} (\mathbf{q}_r(t) - \mathbf{q}(t)) = \mathbf{0}$.

Az algoritmus bemenetei:

- Az előírt csuklókonfiguráció, csuklósebesség, csuklógyorsulás:

$$\mathbf{q}_r = \mathbf{q}_r(t) \quad (13.1)$$

$$\dot{\mathbf{q}}_r = \dot{\mathbf{q}}_r(t)$$

$$\ddot{\mathbf{q}}_r = \ddot{\mathbf{q}}_r(t)$$

- Mért a csuklópozíciók ($\mathbf{q}$) és csuklósebességek ($\dot{\mathbf{q}}$) vektora.
- Ismertek a dinamikus modell tagjai: $M(\mathbf{q}) > 0$, $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$, $G(\mathbf{q})$.

Az *algoritmus kimenete* a robot beavatkozói által kifejtendő nyomaték (τ).

A szabályzó tervezéséhez és vizsgálatához a robot dinamikus modelljéből indulunk ki.

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = \tau, \quad (13.2)$$

ahol $M(\mathbf{q}) > 0$. A robot i -ik csuklójának mozgása függ a többi csukló állapotától (konfiguráció, sebesség) is.

A *tervezés első lépése*: kompenzáljuk a robotra ható erő, inerciák hatását. Ehhez a robotcsuklók beavatkozói által kifejtett nyomatékokat/erőket az alábbi módon határozzuk meg:

$$\tau = M(\mathbf{q})\tau_l + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}), \quad (13.3)$$

ahol τ_l a beavatkozójel lineáris része, amit a tervezés második részében határozzunk meg.

Alkalmazva ezt a beavatkozást a robotra, a dinamikus modell alapján kapjuk:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = M(\mathbf{q})\tau_l + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}), \quad (13.4)$$

$$\ddot{\mathbf{q}} = \tau_l \quad (13.5)$$

Látszik, hogy a (13.3) beavatkozó jel kompenzálja a nemlineáris erőhatásokat, a robotmodellt szétcsatolja, a robot i -ik csuklójának mozgása már nem függ a többi csukló mozgásától.

A *tervezés második lépése*: a τ_l beavatkozó jelet úgy határozzuk meg, hogy a robot kövesse a pályát, vagyis az $\mathbf{e}(t) = \mathbf{q}_r(t) - \mathbf{q}(t)$ pályakövetési hiba nullába konvergáljon.

Az első lépésben meghatározott beavatkozó jellel a robot i -ik csuklójának dinamikáját az alábbi egyenlet adja:

$$\ddot{q}_i = \tau_{li} \quad (13.6)$$

Válasszuk a beavatkozó jelet az alábbi formában:

$$\tau_{li} = \ddot{q}_{ri} + K_{Vi}(\dot{q}_{ri} - \dot{q}_i) + K_{Pi}(q_{ri} - q_i), \quad (13.7)$$

ahol $K_{Pi} > 0$ a proporcionális erősítése (merevség paraméter), $K_{Vi} > 0$ a szabályozó deriváló tagjának erősítése (csillapítás paraméter).

Ezzel a beavatkozó jellel a pályakövetési hiba dinamikáját az alábbi egyenlet adja:

$$\ddot{q}_i = \ddot{q}_{ri} + K_{Vi}(\dot{q}_{ri} - \dot{q}_i) + K_{Pi}(\underbrace{q_{ri} - q_i}_{e_i(t)}) \quad (13.8)$$

$$\ddot{e}_i + K_{Vi}\dot{e}_i + K_{Pi}e_i = 0 \quad (13.9)$$

A dinamikát leíró lineáris differenciálegyenlet karakterisztikus polinomja:

$$s^2 + K_{Vi}s + K_{Pi} = 0 \quad (13.10)$$

Latszik, hogy ha $K_{Pi}, K_{Vi} > 0$, a (13.9) egyenlet stabil, vagyis bármilyen $e_i(0) = e_{i0}$ kezdőállapotból a hiba nullába konvergál, $\lim_{t \rightarrow \infty} e_i(t) = 0$. Tehát a beavatkozó megoldja a szabályozási feladatot minden csuklóra.

Robotirányítási feladatoknál elvárt, hogy a csuklópozíciónak ne legyenek lengései az előírt pálya körül. Ezért a karakterisztikus polinom paramétereit úgy választjuk meg, hogy ne legyenek komplex gyökei. Ennek a feltétele:

$$K_{Vi}^2 > 4K_{Pi} \quad (13.11)$$

A kiszámított nyomatékok módszere – összefoglaló:

A szabályozás minden ciklusában:

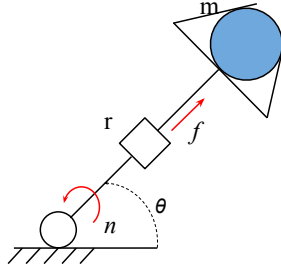
- Adott – $\mathbf{q}_r, \dot{\mathbf{q}}_r, \ddot{\mathbf{q}}_r$
- Mért – $\mathbf{q}, \dot{\mathbf{q}}$
- Kiszámít – $\tau_l = \ddot{\mathbf{q}}_r + K_V(\dot{\mathbf{q}}_r - \dot{\mathbf{q}}) + K_P(\mathbf{q}_r - \mathbf{q})$
- Kiszámít – $\tau = M(\mathbf{q})\tau_l + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q})$
- Visszatérít – τ ,

ahol $K_P = \text{diag}(K_{P1} \dots K_{Pn})$, $K_V = \text{diag}(K_{V1} \dots K_{Vn})$

Példa: legyen a 13.1. ábrán látható robotmanipulátor.

A robot csuklókonfigurációja:

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} \theta \\ r \end{pmatrix} \quad (13.12)$$



13.1. ábra. Kétszabadságfokú R-T (Rotáció – Transzláció) manipulátor

A beavatkozó jelek vektora:

$$\tau = \begin{pmatrix} n \\ f \end{pmatrix} \quad (13.13)$$

A robotszegmens tömegét elhanyagoljuk a mozgatott céltárgy tömege mellett. A robot dinamikus modellje:

$$\begin{pmatrix} mr^2 & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{r} \end{pmatrix} + \begin{pmatrix} 2mr\dot{r}\dot{\theta} \\ -mr\dot{\theta}^2 \end{pmatrix} + \begin{pmatrix} mgr \cos(\theta) \\ mg \sin(\theta) \end{pmatrix} = \begin{pmatrix} n \\ f \end{pmatrix}, \quad (13.14)$$

ahol m a céltárgy tömege, g a gravitációs gyorsulás értéke. Látszik, hogy a gravitáció, centrifugális és Coriolis-erők miatt a robotmodell dinamikájában keresztthatások vannak. Például a szögelfordulás dinamikája függ a transzlációs csukló kinyúlásától és sebességétől.

A kiszámított nyomatékok módszere alapján a beavatkozó jelek:

$$\begin{pmatrix} n \\ f \end{pmatrix} = \begin{pmatrix} mr^2 & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \ddot{q}_{r1} + K_{V1}(\dot{q}_{r1} - \dot{q}_1) + K_{P1}(q_{r1} - q_1) \\ \ddot{q}_{r2} + K_{V2}(\dot{q}_{r2} - \dot{q}_2) + K_{P2}(q_{r2} - q_2) \end{pmatrix} + \begin{pmatrix} 2mr\dot{r}\dot{\theta} \\ -mr\dot{\theta}^2 \end{pmatrix} + \begin{pmatrix} mgr \cos(\theta) \\ mg \sin(\theta) \end{pmatrix} \quad (13.15)$$

13.2.2 A szabályozási algoritmus kibővítése integrátor taggal

Nagy pontosságú robotszabályozásoknál a pályakövetést megvalósító szabályozásnál is figyelembe kell venni a beavatkozó motorok és a robotszegmens közötti áttételekben fellépő súrlódási hatásokat. A nulla állandósult

állapotbeli hiba eléréséhez a szabályozó kell kompenzálja a száraz (Coulomb) súrlódási hatást.

Az algoritmust annyiban változtatjuk meg, hogy a szabályozó lineáris komponensét (τ_{li}) kibővítjük integrátor taggal:

$$\tau_{li} = \ddot{q}_{ri} + K_{Vi}(\dot{q}_{ri} - \dot{q}_i) + K_{Pi}(q_{ri} - q_i) + K_{Ii} \int_0^t (q_{ri}(\xi) - q_i(\xi)) d\xi, \quad (13.16)$$

ahol $K_{Pi} > 0$, $K_{Vi} > 0$, $K_{Ii} > 0$ az integrátor tag erősítése.

Ezzel a beavatkozó jellel a pályakövetési hiba dinamikáját az alábbi egyenlet adja:

$$\ddot{e}_i + K_{Vi}\dot{e}_i + K_{Pi}e_i + K_{Ii}e_i = 0 \quad (13.17)$$

A hiba dinamikát leíró differenciálegyenlet karakterisztikus polinomja:

$$s^3 + K_{Vi}s^2 + K_{Pi}s + K_{Ii} = 0 \quad (13.18)$$

A *lineáris szabályozó paramétereinek tervezése*: a pályakövetési hiba dinamikája stabil, lengésmentes kell legyen. Ugyanakkor szeretnénk előírni a szabályozási időt is.

Válasszunk egy referencia karakterisztikus polinomot úgy, hogy csak valós negatív gyökei vannak:

$$(T_{r1}s + 1)(T_{r2}s + 1)(T_{r3}s + 1) = 0 \quad (13.19)$$

$$T_{r1}T_{r2}T_{r3}s^3 + (T_{r1}T_{r2} + T_{r2}T_{r3} + T_{r3}T_{r1})s^2 + (T_{r1} + T_{r2} + T_{r3})s + 1 = 0$$

Az időállandók pozitívak $T_{r1}, T_{r2}, T_{r3} > 0$.

A szabályozási időt az előírt időállandók fogják meghatározni.

Ahhoz, hogy a pályakövetési hiba dinamikáját a referencia karakterisztikus polinom adja meg, a szabályozó paramétereiket az alábbi módon kell megválasztani:

$$K_{Pi} = \frac{T_{r1} + T_{r2} + T_{r3}}{T_{r1}T_{r2}T_{r3}} \quad (13.20)$$

$$K_{Vi} = \frac{T_{r1}T_{r2} + T_{r2}T_{r3} + T_{r3}T_{r1}}{T_{r1}T_{r2}T_{r3}} \quad (13.21)$$

$$K_{Ii} = \frac{1}{T_{r1}T_{r2}T_{r3}} \quad (13.22)$$

13.2.3. Kiszámított nyomatékok módszere világkoordinátákban

Ebben az esetben a szabályozó alapjele a végberendezés előírt pályája lesz:

$$\mathbf{x}_r = \mathbf{x}_r(t) \quad (13.23)$$

$$\dot{\mathbf{x}}_r = \dot{\mathbf{x}}_r(t)$$

$$\ddot{\mathbf{x}}_r = \ddot{\mathbf{x}}_r(t)$$

$\mathbf{x}_r \in R^m$ a végberendezés előírt konfigurációját jelöli (pozíció koordináták és az orientációt megadó szögek), $m \leq 6$ a munkatér dimenziója.

A *szabályozási feladat*: legyen a robot előírt végberendezés konfiguráció pályagörbéje $\mathbf{x}_r = \mathbf{x}_r(t)$ sima, legalább kétszer deriválható függvény. Számítsuk ki a robot beavatkozási által kifejtett nyomatékokat/erőket, amelyek biztosítják, hogy $\lim_{t \rightarrow \infty} (\mathbf{x}_r(t) - \mathbf{x}(t)) = \mathbf{0}$.

A szabályozó tervezéséhez a robot kinematikai és a (13.2) dinamikus modelljéből indulunk ki.

A kinematikai modell alapján felírjuk a végberendezés gyorsulását:

$$\begin{aligned} \dot{\mathbf{x}} &= J(\mathbf{q})\dot{\mathbf{q}} \\ \ddot{\mathbf{x}} &= J(\mathbf{q})\ddot{\mathbf{q}} + \dot{J}(\mathbf{q})\dot{\mathbf{q}}, \end{aligned} \quad (13.24)$$

ahol $J(\mathbf{q})$ a robot Jacobi-mátrixa.

A robot dinamikus modellje alapján:

$$\ddot{\mathbf{x}} = J(\mathbf{q})M^{-1}(\mathbf{q})(\tau - C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - G(\mathbf{q})) + \dot{J}(\mathbf{q})\dot{\mathbf{q}} \quad (13.25)$$

A beavatkozó jelet úgy választjuk, hogy az kompenzálja a nemlinearitások hatását a végberendezés gyorsulását leíró modellben, linearizálja a robotirányítási rendszert:

$$\tau = M(\mathbf{q})J^{-1}(\mathbf{q})\left(\tau_{xl} - \dot{J}(\mathbf{q})\dot{\mathbf{q}}\right) + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) \quad (13.26)$$

Látszik, hogy ez a beavatkozó jel csak akkor megvalósítható, ha $\det(J(\mathbf{q})) \neq 0$, vagyis a robot nincs szinguláris konfigurációban. Ezért ennél a szabályozási módszernél kiemelten fontos, hogy az előírt pályát úgy tervezzük meg, hogy az ne kerüljön a szinguláris konfigurációk közelébe.

A (13.26) beavatkozással a végberendezés gyorsulását:

$$\ddot{\mathbf{x}} = \tau_{xl} \quad (13.27)$$

A beavatkozó jel τ_{xl} lineáris komponensét úgy választhatjuk meg, hogy az biztosítsa a végberendezés pályakövetési hibájának konvergálását a nulla:

$$\tau_{xl} = \ddot{\mathbf{x}}_r + K_V(\dot{\mathbf{x}}_r - \dot{\mathbf{x}}) + K_P(\underbrace{\mathbf{x}_r - \mathbf{x}}_{\mathbf{e}_{xl}}), \quad (13.28)$$

ahol $K_V = \text{diag}(K_{Vi})$, $K_P = \text{diag}(K_{Pi})$ diagonális mátrixok szigorúan pozitív diagonális elemekkel ($K_{Vi}, K_{Pi} > 0$).

Ezzel a beavatkozással a pályakövetési hiba dinamikáját az alábbi egyenletrendszer adja:

$$\ddot{\mathbf{e}}_{xl} + K_V \dot{\mathbf{e}}_{xl} + K_P \mathbf{e}_{xl} = \mathbf{0} \quad (13.29)$$

A stabil, túllövésmentes válaszhoz K_P és K_V diagonális mátrixok elemeit úgy határozzuk meg, hogy az alábbi karakterisztikus polinomok gyökei valósak és szigorúan negatívak legyenek:

$$s^2 + K_{Vi}s + K_{Pi} = 0 \quad (13.30)$$

Ezzel a választással a τ és τ_{xl} által megadott irányítási algoritmus garantálja, hogy a végberendezés konfigurációjának pályakövetési hibája túllövésmentesen nullához tartson.

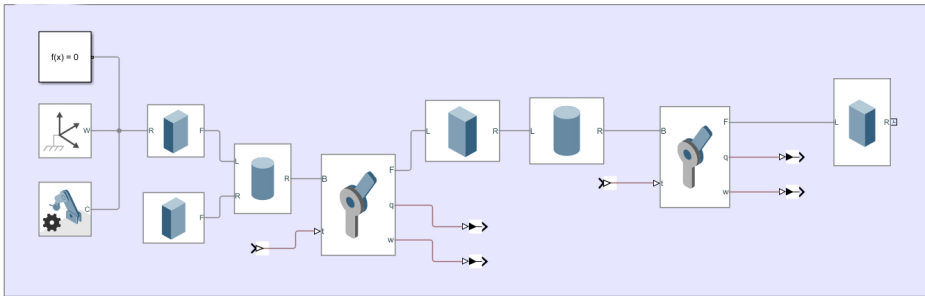
13.3. A gyakorlat menete

A *gyakorlat célja*: felhasználva az előző gyakorlat során készített robot modellt, valósítsuk meg a körpályakövetést a kiszámított nyomatékok módszerét alkalmazva.

1. A robotmodellből hozzunk létre egy merev test gráfot.
2. A generált gráfot felhasználva hozzuk létre a csuklózható terében értelmezett pályakövető szabályozót.
3. Generáljuk ki az körpályát.
4. Az inverz kinematikai összefüggésekkel adjuk meg a csukló referencia pályát, csuklókoordinátákban.
5. Hasonlítsuk össze a robot végberendezésének viselkedését a direkt kinematikából kapott eredményekkel.

Láttuk, hogy a kiszámított nyomatékok módszeréhez szükséges ismerni a robot tömegmátrixát, a sebesség szorzat nyomatékvektort (centrifugális és Coriolis-hatások miatt létrejött nyomaték), illetve a gravitációs nyomaték vektort. Ezeket a robot geometriájából analitikus módon megkaphatjuk, vagy felhasználhatjuk a Simulink által biztosított robotikai modellezési eszközöket.

Előkészítés: az előző órán készített robotmodellt másoljuk át egy külön állományba. A modell csupán a mechanikai részt és a ki-/bemeneteket kell tartalmazza.

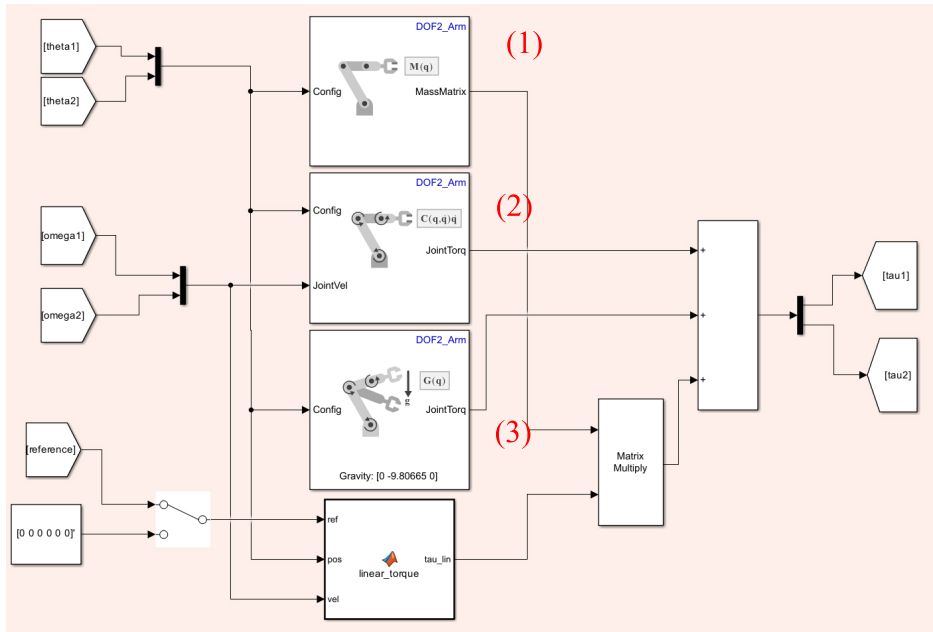


13.2. ábra. A robot mechanikai modellje

Ezek után adjunk meg a prompterben a diszkrétizálási időt ($T_s = 0.001$), mely a modell elemzéséhez fog kelleni, illetve hívjuk meg a `[DOF2_Arm, ArmInfo] = importrobot('rr_2dof_arm.slx');` parancsot. Az első paraméter tartalmazza a merev test gráfot, amiből a rendszer ki tudja olvasni a különböző dinamikus paramétereket.

Linearizálás visszacsatolással: az elméleti részben szereplő (13.3) összefüggést kell összeállítani a (13.3)-as ábrán látható tömbökkel. Az $M(\mathbf{q})$ tömegmátrixot a *Robotics System Toolbox / Joint Space Mass Matrix* tömb számolja ki a csuklózások függvényében (1). A $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ a sebesség szorzat nyomatékvektort a *Robotics System Toolbox / Velocity Product Torque* tömb számolja ki a csuklózások és csuklósebességek függvényében (2). A $G(\mathbf{q})$ gravitációs nyomaték vektort a *Robotics System Toolbox / Gravity Torque* tömb számolja ki a csuklózások függvényében (3).

Pályakövetés biztosítása: a szabályozás utolsó lépése a lineáris nyomaték kiszámolása aszimptotikus konvergenciához. Adott referencia profil és mért csuklózások, illetve csuklósebességek ismeretében a (13.7) összefüggést egy egyszerű *MATLAB Function* tömbbel valósítjuk meg, ahol a proporcionális



13.3. ábra. Kiszámított nyomatékok módszere

erősítés mátrix legyen $Kp = 10I_2$, a deriválási erősítés mátrix pedig $Kd = 4.5I_2$.

Körpályagenerálás: legyen az elvárt referenciapálya egy $(1\text{ m}, 1\text{ m})$ középpontú, 0.5 m sugarú kör. Az x és y koordinátákat idő függvényében parametrikusan megadhatjuk egy *MATLAB Function* tömbbel, ahol a bemenet a szimulációs környezet időjele:

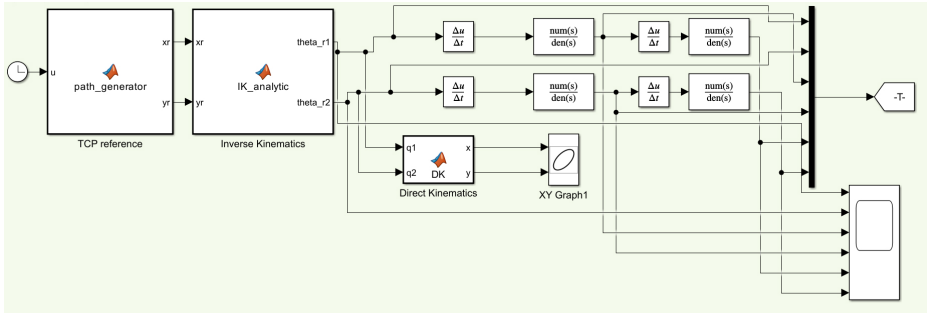
```
function [xr, yr] = path_generator(u)
    xr = 1 + 0.5*sin((2*pi/25)*u+pi/2);
    yr = 1 + 0.5*cos((2*pi/25)*u+pi/2);
```

A világhkoordinátákban megadott referenciát át kell alakítani csuklózálóterébe, felhasználva az analitikus inverz geometriai összefüggéseket. Az összefüggéseket egy *MATLAB Function* tömbben valósítjuk meg:

```
function [theta_r1, theta_r2] = IK_analytic(xr, yr)
    l1 = 1; l2 = 1;
    theta_r2 = acos((xr^2+yr^2-l1^2-l2^2)/(2*l1*l2));
    s_theta2 = sin(theta_r2); c_theta2 = cos(theta_r2);
    theta_r1 = atan2(yr, xr)
```

```
– atan2(12*s_theta2,(11+12*c_theta2));
```

Az így kapott eredmény a referenciaszöveget tartalmazza. A sebesség és gyorsulás számolásához deriválást használunk. Az ugrások csökkentése miatt minden deriválási művelet után alkalmazunk egy elsőrendű aluláteresztő szűrőt $T = 0.01\text{ s}$ időállandóval.

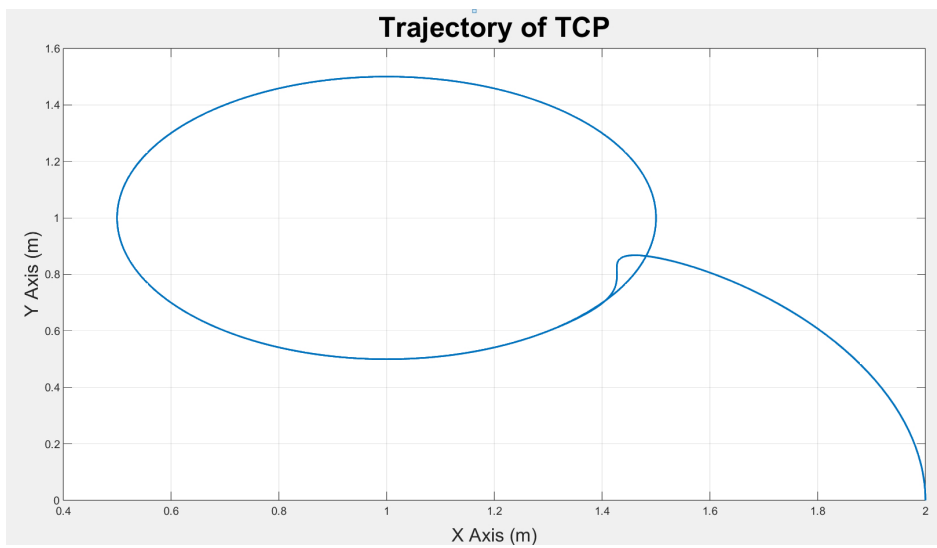


13.4. ábra. Pályagenerálás inverz kinematikai módszerrel

Helyes szabályozás és pályagenerálás esetén a robot végberendezése leírja az előírt körpályát. Az eredmény a 13.5-ös ábrához hasonló kell legyen.

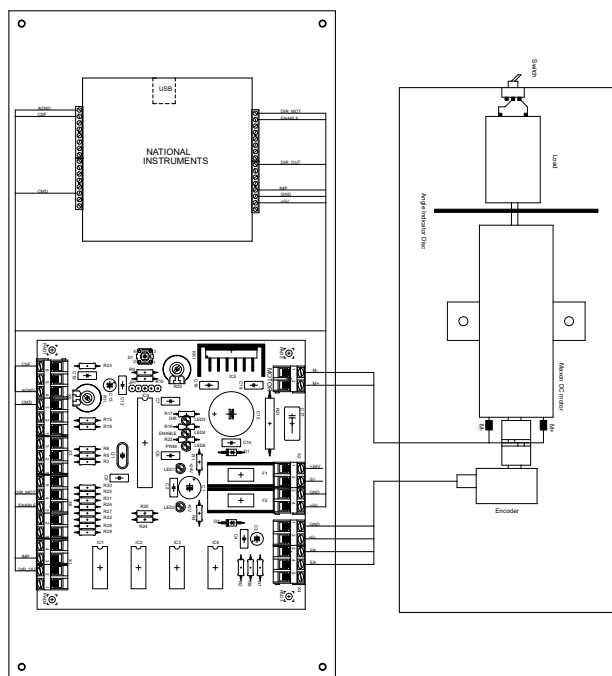
13.4. Kérdések és feladatok

1. Egészítsük ki a robotmodellt viszkózus és Coulomb-féle súrlódással, tudva azt, hogy a Coulomb-féle súrlódási együttható 1.7 Nm , a viszkózus súrlódási együttható 4.2 Nms/rad . Hogyan viselkedik a szabályozott rendszer? Mi történik, ha integrátort teszünk a szabályozóba?
2. Módosítsuk a szabályozót világ koordináta-rendszerben megadott referenciához, inverz kinematikai egyenletek nélkül.
3. Hasonlítsuk össze a kiszámított nyomatékok módszerével szabályozott rendszer és a PID alapú szabályozott rendszer tranziens jellemzőit a létrehozott pálya követésével.

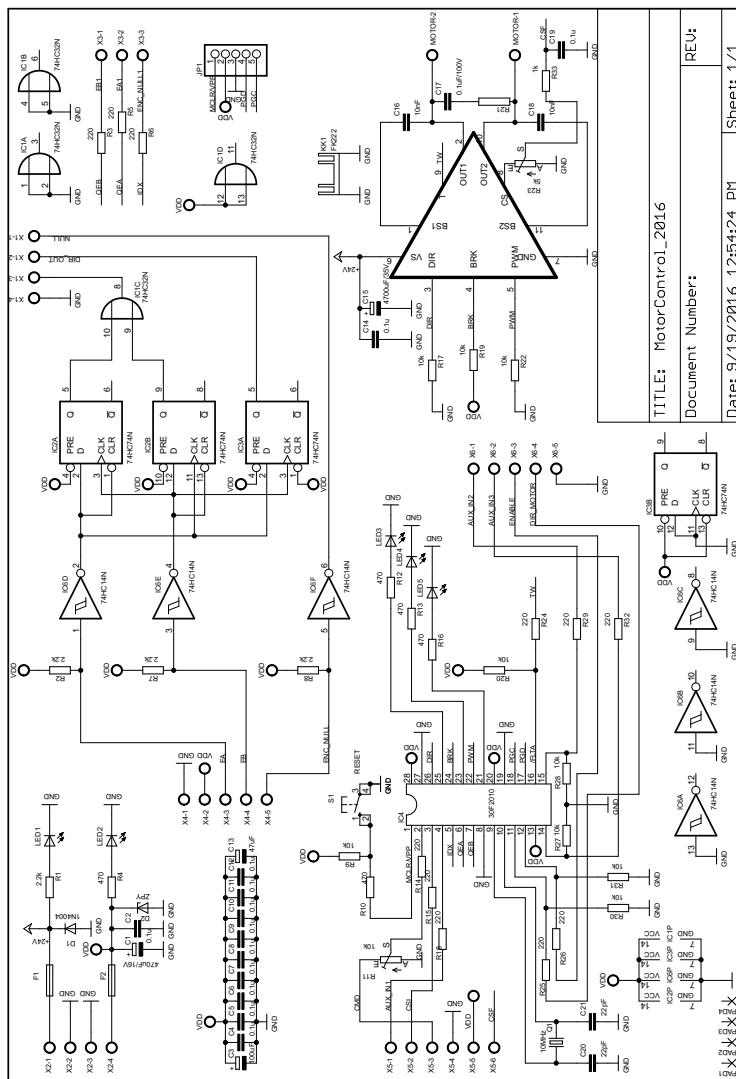


13.5. ábra. *RR robot pályakövetési feladata kiszámított nyomatékok módszerével*

FÜGGELÉK



F1. ábra. *Motorvezérlési stand moduláris bekötési rajza*



F2. ábra. *A vezérlő áramkör kapcsolási rajza*

SZAKIRODALOM

MÁRTON László Ferenc

2006 *Jelek és rendszerek*. Kolozsvár, Scientia.

Texas Instruments Incorporated

2011, rev. 2021 *Current recirculation and Decay Modes*

MÁRTON Lőrinc

2009 *Irányítástechnika*. Kolozsvár, Scientia.

FEHÉR Áron, MÁRTON Lőrinc

2019 *Irányítástechnika. Laboratóriumi útmutató*. Kolozsvár, Scientia.

KISS István, CSELKÓ Richárd

2012 *Villamos gépek és hajtások laboratórium I.* tankonyvtar.hu.

MESTER Gyula

2011 *Robotika*. Budapest, Typotex.

LANTOS Béla

2002 *Robotok irányítása*. Budapest, Akadémiai Kiadó.

KULCSÁR Béla

2012 *Robottechnika I.* tankonyvtar.hu.

KULCSÁR Béla

2012 *Robottechnika II.* tankonyvtar.hu.

Bruno SICILIANO, Oussama KHATIB

2016 *Springer Handbook of Robotics*. Berlin Heidelberg, Springer.

LANTOS Béla, MÁRTON Lőrinc

2011 *Nonlinear Control of Vehicles and Robots*. London, Springer.

Frank L. LEWIS, Darren M. DAWSON, Chaouki T. ABDALLAH

2004 *Robot Manipulator Control: Theory and Practice*. New York, Marcel Dekker.

Mark W. SPONG, Seth HUTCHINSON, and M. VIDYASAGAR

2020 *Robot Modeling and Control*. New York, John Wiley and Sons Ltd.

Peter CORKE

2017 *Robotics, Vision and Control*. Berlin Heidelberg, Springer.

REZUMAT

Robotică – Ghid de laborator

Scopul acestui ghid de laborator este de a oferi o introducere în procedurile și metodele de control, proiectare a traiectoriei și programare utilizate în sistemul de control al roboților.

Capitolul introductiv al ghidului prezintă noțiunile de bază de robotică, necesare pentru proiectarea și implementarea sistemului de control al roboților: descrierea și caracteristicile manipuletoarelor, caracterizarea poziției și orientării spațiale.

Primele cinci exerciții de laborator discută elementele de execuție și senzorii roboților. Sunt prezentate controlul motorului de curent continuu (DC), măsurarea vitezei unghiulare și a poziției unghiulare cu ajutorul unui traductor incremental de rotație, reglarea poziției unghiulare și a vitezei unghiulare a unui motor DC folosind algoritmi de reglare clasici (PI, PD). Cel de-al cincilea exercițiu de laborator se ocupă de proiectarea traiectoriei polinomiale în timp minim și problema de control de urmărire a traiectoriei.

Exercițiile șase și șapte discută implementarea problemelor de geometrie directă și inversă a roboților: determinarea configurației spațiale a efectorului final în funcție de pozițiile articulațiilor, și calculul poziției articulațiilor pe baza configurației efectorului final.

Al optulea exercițiu de laborator tratează implementarea software a problemei de cinematică inversă a manipuletoarelor robotice (calculul vitezei efectorului final) și problema configurațiilor singulare ale robotului.

Exercițiile nouă și zece discută cele mai importante instrucțiuni ale limbajului de programare a roboților, planificarea sarcinii robotului și proiectarea traiectoriei, prin utilizarea mediului de programare și simulare a manipuletoarelor Universal Robots™.

Ultimele două exerciții de laborator tratează proiectarea și implementarea algoritmilor de reglare al roboților într-un mediu de simulare. Discută separat urmărirea unei configurații constante folosind regulatoare de tip PID, precum și metoda momentului calculat pentru urmărirea traiectoriei.

ABSTRACT

Robotics – Laboratory Guide

The purpose of this laboratory guide is to provide an introduction to the control, trajectory planning, and programming procedures and methods used in robot control systems.

The introductory chapter of this guide recalls the basic notions of robotics that are necessary for robot control design and implementation: the description and characteristics of robot manipulators and the characterization of spatial position and orientation.

The first five laboratory exercises discuss the robot actuators, sensors, and the software implementation control tasks of one-degree-of-freedom mechanical systems: the control of the direct current (DC) motor, measurement of angular speed and angular position using an incremental encoder, and angular position and speed control implemented with a DC motor and an incremental encoder using classic control algorithms (PI, PD). The fifth laboratory exercise deals with the task of minimum-time polynomial trajectory planning and tracking control problems.

The sixth and seventh robot exercises discuss the software implementation of direct and inverse geometry problems of robots: the determination of the spatial configuration of the end effector in function of the joint variables and the computation of the joint variables based on the configuration of the end effector.

The eighth laboratory exercise deals with the software implementation of the inverse kinematics task of robot manipulators (computation of the end-effector speed) and discusses the singular robot configurations.

The ninth and tenth lab exercises discuss the application of the most important instructions of the robot programming language, the planning of the robot's task, and the trajectory planning, by using the programming and simulation environment of Universal RobotsTM manipulators.

The last two lab exercises deal with the design and implementation of robot control algorithms in a simulation environment. We discuss separately the constant setpoint configuration tracking using PID type controllers, as well as the computed torque method for trajectory tracking.

A SZERZŐKRŐL

Márton Lőrinc (www.ms.sapientia.ro/~martonl) középiskolai tanulmányait a marosvásárhelyi Bolyai Farkas Elméleti Líceumban végezte 1990–1994 között, matematika–fizika szakon. A marosvásárhelyi Petru Maior Egyetem automatizálás és ipari informatika szakán 1999-ben szerzett egyetemi oklevelet, majd 2000-ben mesteri oklevelet. 2000–2003 között doktorandusz a Budapesti Műszaki és Gazdaságtudományi Egyetemen, 2006-ban doktori címet szerez. 2003-tól a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhelyi Karának oktatója, 2019-től egyetemi tanár. 2007–2010 között posztdoktori kutatói munkáját a Magyar Tudományos Akadémia (MTA) Bolyai János kutatási ösztöndíjának támogatásával végzi. 2010–2012 között a Német Űrkutatási Hivatal (DLR) Robotika és Mechatronika Intézetének kutatója az Alexander von Humboldt Alapítvány támogatásával. 2016-ban három hónapig vendégkutató a Ruhr Egyetemen, Németországban. 2016-tól éves rendszerességgel vendégkutatói tevékenységet végez a budapesti Számítástechnikai és Automatikai Kutatóintézetben (SZTAKI), illetve a Pannon Egyetemen. 2017-től doktorátusvezető a Pannon Egyetem Informatikai Tudományok Doktori Iskolájában. Főbb kutatási területei az irányításelmélet, hálózatok, robotirányítások.

Fehér Áron (www.ms.sapientia.ro/~fehera) középiskolai tanulmányait a marosvásárhelyi Bolyai Farkas Elméleti Líceumban végezte 2007–2011 között, természettudomány szakon. A Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhelyi Karán szerzett mérnöki oklevelet automatika és alkalmazott informatika szakon 2015-ben, majd mesteri oklevelet számítógépes irányítási rendszerek szakon 2017-ben. 2017–2021 között doktorandusz a veszprémi Pannon Egyetem Informatikai Tudományok Doktori Iskolájában, 2022-ben doktori címmel szerez. 2023-tól adjunktus a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhelyi Karán. Főbb kutatási területei a késleltetett rendszerek tranziens viselkedésének vizsgálata, analóg és digitális jelfeldolgozási algoritmusok, biológiai jelfeldolgozás.

Scientia Kiadó

400112 Kolozsvár (Cluj-Napoca)
Mátyás király (Matei Corvin) u. 4. sz.
Tel./fax: +40-364-401454
E-mail: scientia@kpi.sapientia.ro
www.scientiakiado.ro

Korrektúra

Szabó Beáta

Műszaki szerkesztés

Fehér Áron

Tipográfia

Könczey Elemér

Készült az F&F International nyomdában

Igazgató: Ambrus Enikő

A laboratóriumi útmutató bevezeti a hallgatókat a robotmanipulátorok irányítási rendszerében alkalmazott pályatervezési, szabályozási és programozási eljárásokba, módszerekbe, ugyanakkor betekintést nyújt az UR robotok magas szintű programozásába is.

A gyakorlatok segítenek a robotirányítási rendszerekben használt geometriai és kinematikai számítások szoftveres megvalósításának elsajátításában, a robotszabályozók tervezésében és számítógépes implementációjában, valamint a robotokban alkalmazott főbb érzékelők és beavatkozók használatában.

ISBN 978-606-975-089-6



9 786069 750896