UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,

TÎRGU MUREȘ

SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

REGLAREA ȘI IMPLEMENTAREA PRACTICĂ A UNUI ACROBOT

PROIECT DE DIPLOMĂ

Coordonator științific:

Prof.dr.ing. Dávid László

Absolvent:

Lukács Szabolcs

2023

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA **Viza facultății:**
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Specializarea: **Automatică și informatică aplicată**

LUCRARE DE DIPLOMĂ

Coordonator științific:	Candidat: **Nume student**
ș.l. dr. ing. Nume cadru didactic	Anul absolvirii: **2023**

a) Tema lucrării de licență:
CONTROLUL AUTOMAT AL SERELOR

b) Problemele principale tratate:
- Studiu bibliografic privind sistemele de reglare.
- Realizarea unei aplicații pentru simularea procesului studiat.
- Clasificarea metodelor de reglare

c) Desene obligatorii:
- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat.

d) Softuri obligatorii:
-Aplicație

e) Bibliografia recomandată:
- Márton Lőrinc, Irányítsátechnika, Scientia, 2009
- Dávid László, Tehnici de optimizare: metode numerice de calcul în tehnica reglării optimale, Editura Universității Petru Maior, Tg. Mureș, 2000,

f) Termene obligatorii de consultații: săptămânal
g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Primit tema la data de: 31.03.2022
Termen de predare: 27.06.2023

Semnătura Director Departament | Semnătura coordonatorului

Semnătura responsabilului programului de studiu | Semnătura candidatului

Declarație

Subsemnatul Lukács Szabolcs, absolvent(ă) al/a specializării automatică şi informatică aplicată, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu Mureș

Data: 25.06.2023

Absolvent

Semnătura……

REGLAREA ȘI IMPLEMENTAREA PRACTICĂ A UNUI ACROBOT

Extras

Tema tezei mele este implementarea unui sistem de control dinamic neliniar subactuat, și anume un "ACROBOT", care poate fi folosit ca exercițiu de laborator. Controlul sistemelor subactuate necesită un tip de control diferit față de cel al sistemelor convenționale. Scopul meu este de a construi un mecanism experimental pentru a demonstra un astfel de control.

Sistemul dinamic studiat este "acrobotul", care este un pendul dublu inversat, dar care are un dispozitiv de intervenție doar în a doua articulație, deci are două grade de libertate, dar poate interveni doar într-unul dintre gradele de libertate, adică este subactuat.

Scopul principal al controlului este de a muta acrobotul din punctul de echilibru inferior, stabil, în punctul de echilibru superior, instabil, și de a-l stabiliza acolo cu ajutorul unui algoritm de control DLQR. Prin modelarea și simularea precisă a sistemului real, am obținut traiectoria prescrisă a celui de-al doilea segment. Pentru a pune în aplicare acest lucru on-line, adică în timp real, am utilizat o rețea neuronală pentru a mă asigura că unghiul prescris este obținut pe baza stărilor în timp real.

În implementare am folosit un servomotor pentru a deplasa al doilea segment și un encoder incremental pentru a măsura unghiurile și vitezele unghiulare. Prelucrarea semnalului și calcularea semnalului de acțiune se realizează cu ajutorul unor plăci de dezvoltare arduino conectate în rețea.

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM

MAROSVÁSÁRHELYI KAR

AUTOMATIKA ÉS ALKALMAZOTT INFORMATIKA SZAK

ACROBOT SZABÁLYOZÁSA ÉS GYAKORLATI MEGVALÓSÍTÁSA

DIPLOMADOLGOZAT

Témavezető:

Dr. Dávid László, egyetemi tanár

Hallgató:

Lukács Szabolcs

2023

Kivonat

A dolgozatom témája egy olyan nemlineáris, alulaktuált dinamikus rendszer szabályozásának megvalósítása, nevezetesen egy „ACROBOT”, amelyet később laborgyakorlatként is lehet majd használni. Az alulaktuált rendszerek szabályozása másfajta szabályozást igényel, mint a hagyományos rendszereké. Célom egy olyan kísérleti eszköz elkészítése, amin be lehet mutatni egy ilyen szabályozást.

Az tanulmányozott dinamikus rendszer az „acrobot”, ami egy inverz dupla inga, de csak a 2. csuklóban van beavatkozó eszköz, tehát két szabadságfokkal rendelkezik, de csak az egyik szabadságfokban lehet beavatkozni, vagyis alulaktuált.

A szabályozás fő célja eljuttatni az acrobotot az alsó, stabil egyensúlyi pontjáról, a felső, instabil pontjára, és ott stabilizálni felhasználva egy DLQR szabályozási algoritmust. A valós rendszert pontosan modellezve, és szimulálva megkaptam a második szegmens előírt pályáját. Ahhoz, hogy ezt on-line, azaz valós időben lehessen implementálni, egy neuronálót használtam fel, amely biztosítja, hogy a valós idejű állapotok alapján kapjuk meg az előírt szöget.

A megvalósítás során egy szervomotort használok a második szegmens mozgatására, és egy inkrementális enkódert, a szögek és szögsebességek mérésére. A jelek feldolgozása és a beavatkozó jel kiszámítása hálózatba csatolt arduino fejlesztőlapokkal történik.

Kulcsszavak: Acrobot, Alulaktuált rendszer, LQR

Abstract

The topic of my thesis is the implementation of a nonlinear underactuated dynamic control system, namely an "ACROBOT", which can be used as a laboratory exercise. The control of underactuated systems requires a different kind of control than that of conventional systems. My goal is to build an experimental device to demonstrate such a control.

The dynamic system under study is the "acrobot", which is an inverted double pendulum, but it has an intervention device only in the 2nd joint, so it has two degrees of freedom, but it can only intervene in one degree of freedom, i.e., it is underactuated.

The main purpose of the control is to move the acrobot from its lower, stable equilibrium point to its upper, unstable point and stabilize it there using a DLQR control algorithm. By accurately modeling and simulating the real system, I obtained the prescribed trajectory of the second segment. In order to implement this on-line, in real time, I used a neural network to ensure that the prescribed angle is obtained based on the real-time states.

In the implementation, I use a servo motor to move the second segment and an incremental encoder to measure the angles and angular velocities. The signal processing and the calculation of the action signal is done with networked arduino development boards.

Keywords: Acrobot, Underactuated Systems, LQR

Tartalomjegyzék

1 Ábrajegyzék

2 Bevezető

Az acrobot egy nemlineáris alulaktuált rendszer. Akkor nevezünk egy rendszert alulaktuáltnak, ha szabadságfokainak a száma nagyobb, mint a rendszer független beavatkozóinak száma. Ez azt jelenti, hogy a rendszernek van néhány "passzív" foka, amelyet nem lehet közvetlenül irányítani az elérhető aktuátorokkal, így nagyon fontos szerepet kap ezen rendszerek szabályozásában a rendszer belső dinamikája.[1] Az alulaktuált rendszerek fontossága abban rejlik, hogy a való életben is számos rendszer alulaktuált. Az alulaktuált rendszerek egyre gyakrabban találhatóak meg a modern robotikában, ilyenek például a sétáló robotok, segway-ek, bipedal robotok, űrbéli robotok, flexibilis robotok, a legtöbb emberi mozgást utánozni próbáló robot.[2] Az egyik legegyszerűbb ilyen rendszer, ezen a területen már alappéldának számító acrobot, ami az emberi járás egyik legegyszerűbb modellje.

Az acrobot esetében 2 szabadságfokról beszélhetünk, és egy beavatkozóról. Ha a beavatkozó az első csuklóban helyezkedik el, akkor pendubotnak nevezzük, ha a másodikban, akkor acrobot. Ha egyik csuklóban sincs beavatkozó, abban az esetben dupla inverz ingáról beszélünk.

Az acrobot esetében a klasszikus szabályozás a robot stabilizálását jelenti a fenti egyensúlyi pontja körül. Általában ezt két részre lehet osztani, az ú.n. “Swing-up” és “Balancing” fázisra[3]. A swing-up fázisban a cél a második szegmens eljuttatása az equilibrium pont környezetébe, a stabilizáló fázis feladata ott tartani és az instabil pontja körül stabilizálni a rendszert.

A dolgozatom célja egy egyszerű acrobot fizikai megvalósítása és annak szabályozása, amely később használható oktatási célra, másfajta szabályozások tanulmányozására. Az acrobot dinamikája eléggé komplex ahhoz, hogy sokféle szabályozási technikát tanulmányozni lehessen rajta.

3 Irodalmi áttekintés

Az acrobottal és annak szabályozási nehézségeivel leginkább a ’90-es évek elején kezdtek el foglalkozni. A cél az alulaktuált rendszerek tanulmányozása volt, és az egyik ilyen rendszer az acrobot. Az egyik ilyen cikk, amelyet a későbbiekben sokan idéztek, az Mark.W.Spong [2] cikke, amiben bemutat egy működő szabályozást, az ú.n. ‘swing-up and balance’. Itt az energia pumpálás elvét használja, majd egy LQR szabályozást a linearizált tartományon belüli stabil szabályozásra. A robotot eljuttatja a felső instabil pontjára, majd ott történik egy váltás a két szabályzó között, ami már képes ott stabilan tartani a szerkezetet. Ez volt az egyik legelső és legegyszerűbben leimplementálható szabályozása egy ilyen rendszernek. Egy másik fontos irodalom volt Russ Tedrake MIT kurzus könyve[4], amelyben az acrobot mozgási egyenletei voltak részletezve. Az acrobot mozgásának és szabályozásának a tanulmányozása még a napjainkban is aktív. Manapság másfajta szabályzások megvalósításával foglalkoznak, például optimális pályakövetés, minimum idejű, modell prediktív szabályozások, robusztusabb szabályozások tervezése. Továbbá tanulmányozzák a gépi tanulás eredményeit is, amelyben egy visszacsatoláson keresztül egy valós rendszert is tanítani lehet az egyre pontosabb szabályozás elérésére.[5]

3.1 Gyakorlati megvalósítások

Egy acrobot fizikai megvalósítása a legtöbb esteben nehézkes. Két fő részre oszthatók: „Directly driven acrobot” (DDA) és „Remotely driven acrobot” (RDA).[6]

3.1.1 RDA (Remotely Driven Acrobot)

A második szegmens meghajtása egy vezérműszíjon keresztül van meghajtva, így elérve, hogy az aktuátor nem mozog együtt az egész rendszerrel, hanem egy fix ponthoz rögzítve tudja vezérelni a második szegmenst. Előnye, hogy a motor súlya nem terheli a rendszert, így a szegmensek könnyebb anyagból is készíthetők, és nagyobb nyomaték is kifejthető, az áttételnek köszönhetően. Hátránya a fizikai megvalósításban a bonyolultabb felépítés, az áttételek megoldása és stabilan tartása a robot teljes mozgási tartományán belül. A gyakorlatban ez az elterjedtebb modell az acrobot bemutatására.

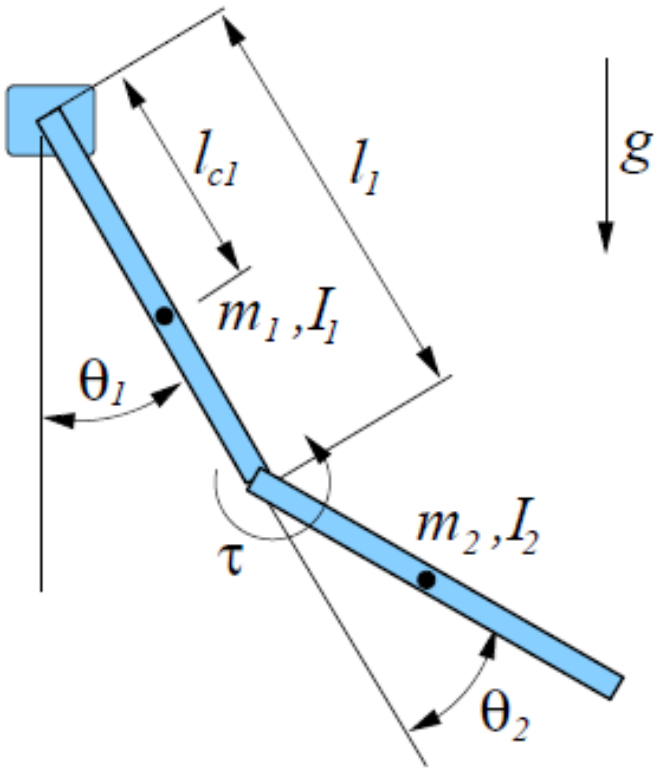
3.1.2 DDA (Directly Driven Acrobot)

A directly driven acrobot esetében közvetlenül van meghajtva a második szegmens. Ebben az esetben a motor, vagy a beavatkozó közvetlenül hajtja meg a második szegmenst. Kevésbé elterjedt, mivel itt korlátozottak a rendszer paraméterei. Ha túl nagy a motor tömege, akkor a második kar végén lévő súlyt is növelni kell, hogy fel lehessen tornásztatni az inverz állapotába. Ha viszont nagyobb az alsó súly, az általában nehezebb, drágább motorhoz vezet, mivel nagyobb nyomaték lesz szükséges ennek mozgatására. Előnye viszont, hogy egyszerűbb fizikai felépítéssel is kivitelezhető.

A fenti előnyöket és hátrányokat figyelembe véve a DDA mellett döntöttem, mivel manapság már elérhetőek kellően könnyű és viszonylag nagy maximális kifejthető nyomatékkal rendelkező szervomotorok. Így kevésbé komplex mechanikai kivitelezésben is megvalósítható a rendszer.

4 Az acrobot dinamikája

Az acrobot egy két szabadságfokkal rendelkező, úgynevezett dupla, kétszeresen csatolt inga. A felfüggesztéshez közelebbi csuklót sokszor nevezik vállnak, a távolabbit, ahol az aktuátor jelen van, könyöknek.[4] Ez a (4.1. ábra)-n τ-val jelölve látható a beavatkozó nyomaték, illetve az 1. táblázat.-ban az ábrához tartozó jelmagyarázat, benne a modellezés során is felhasznált összes paraméterrel. A szabályozási nehézséget az adja, hogy figyelembe kell venni az állapotfüggő kapcsolatot az aktív és a passzív szabadságfokok közt.



4.1. ábra. Az acrobot modellje [4]

A szegmensek tehetetlenségi nyomatéka:[7]

$$I = \frac{1}{12} m l^2 \qquad (1)$$

τ	nyomaték
l_1	Első szegmens hossza
l_2	Második szegmens hossza
lc_1	Első szegmens tömegközéppontjának hossza
lc_2	Második tömegközéppontjának hossza
m_1	Első szegmens tömege
m_2	Második szegmens tömege
g	Gravitációs állandó
θ_1	Első szegmens relatív elfordulása
θ_2	Második szegmens relatív elfordulása
I_1	Első szegmens inerciája
I_2	Második szegmens inerciája

1. táblázat. *Jelmagyarázat*

4.1 Mozgásegyenletek

A mozgásegyenletek a könnyebb átláthatóság kedvéért a Lagrange mechanika alapján vannak levezetve.[4]

x_1 x_2 jelöli az abszolút helyét az első és a második szegmens tömegközéppontjának.

$$x_1 = \begin{pmatrix} lc_1 * \sin(q_1) \\ -lc_1 * \cos(q_1) \end{pmatrix} \qquad x_2 = x_1 + \begin{pmatrix} l_2 * \sin(q_1 + q_2) \\ -l_2 * \cos(q_1 + q_2) \end{pmatrix}$$

(2)

Így az acrobot mozgási energiája P, ami két részből tevődik össze.

$$P_1 = \frac{1}{2} * I_1 * \dot{q_1^2}$$

(3)

$$P_2 = \frac{1}{2} * (m_2 * l_1^2 + I_2 + 2 * m_2 * l_1 * lc_2 * \cos(q_2)) * \dot{q_1^2} + \frac{1}{2} * I_2 * \dot{q_1^2}$$

$$+ (I_2 + m_2 * l_1 * lc_2 * \cos(q_2)) * \dot{q_1^2} * \dot{q_2^2}$$

(4)

Mozgási energia

$$P = P_1 + P_2$$

(5)

Potenciális energia

$$K = -m_1 * g * lc_1 * \cos(q_1) - m_2 * g * (l_1 * c_1 + l_2 * \cos(q_1 + q_2)$$

(6)

A Lagrange függvény

$$L = P - K$$

(7)

Kiszámítva a parciális deriváltakat, [8] majd behelyettesítve az Euler-Lagrange-egyenletbe(8),[9]

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \left(\frac{\partial L}{\partial q}\right) = \tau$$

(8)

megkapjuk a mozgási egyenleteket. Ezeket rendszerezve kapjuk az acrobot mozgását leíró mátrixokat.[4]

$$M(q) = \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + 2m_2 l_1 lc_2 \cos(q2) & I_2 + m_2 l_1 lc_2 \cos(q_2) \\ I_2 + m_2 l_1 lc_2 \cos(q_2) & I_2 \end{bmatrix}$$

(9)

$$C(q, \dot{q}) = \begin{bmatrix} -2m_2 l_1 lc_2 \sin(q2)\, \dot{q}_2 & -m_2 l_1 lc_2 \sin(q_2)\dot{q}_2 \\ m_2 l_1 lc_2 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix}$$

(10)

$$G(q) = \begin{bmatrix} m_1 g lc_1 \sin(q_1) - m_2 g(l_1 \sin(q_1) + lc_2 \sin(q_1 + q_2)) \\ m_2 g lc_2 \sin(q_1 + q_2) \end{bmatrix}$$

(11)

Ahol:

M - tehetetlenségi mátrix

C - Centrifugális és a Coriolis erők vektora

G - a gravitációs erő hatása

Összeségében a következőképpen írható fel az acrobot dinamikus modellje:

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = Bu$$

(12)

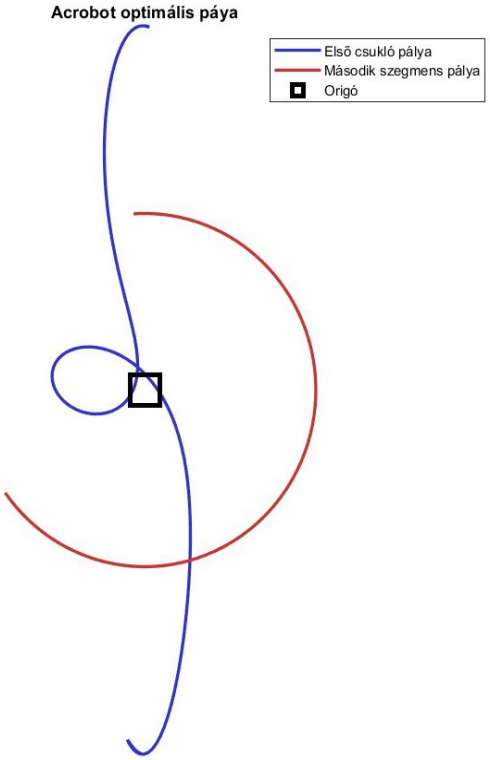
Ahol az acrobot esetében

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad (13)$$

Csak az egyik bemenet aktív, mivel csak a második csuklóban van a beavatkozó.

4.2 Az optimális pálya

Az acrobot optimális pályája alatt, ebben az esetben azt értjük, amikor a rendszer a legkisebb nyomaték kifejtésével tud eljutni az alsó egyensúlyi pontjáról a felső, instabil pontjára. Az 5.2.-ben részletezett szabályozás nem optimális pályát követ, hanem a folyamatos energia pumpálás módszerével juttatja el az acrobotot a felső pontjára. Bebizonyítható, hogy nem ez az optimális pálya, viszont ez a módszer valósítható meg a legmegbízhatóbban, és a valós időben.

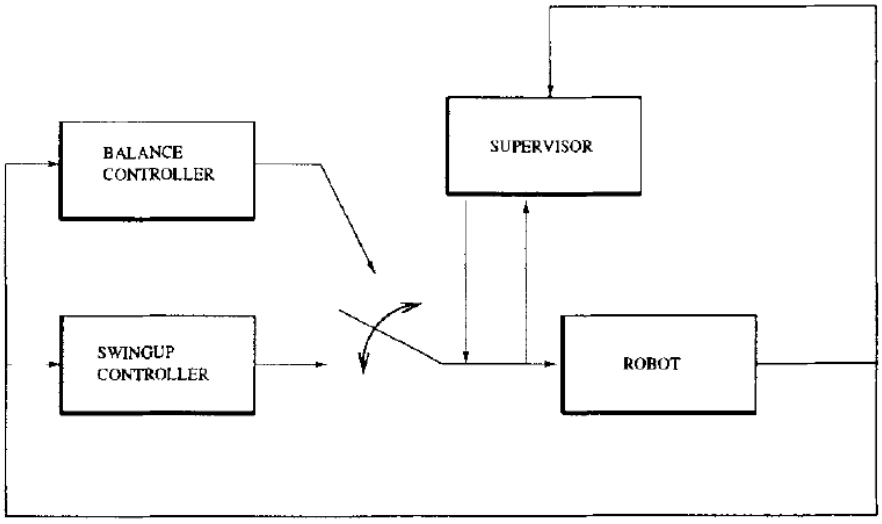


4.2. ábra. Az Acrobot optimális pályája

Az acrobot optimális pályájának kiszámításához az optimTraj[10] nevű Matlabban futtatható programot használtam. A program által kiszámított pálya a (4.2. ábra.)-n látható. A könnyebb átláthatóság érdekében az első és a második szegmens végpontja által leírt pálya látható. A számításokban a valós rendszer paramétereit használtam. Megfigyelhető, hogy ebben az esetben a súlyt elég volt egyszer, viszont nagy nyomatékkal belendíteni, és az már elegendő volt ahhoz, hogy a teljes szerkezetet eljuttassa a felső egyensúlyi állapotába. A programba be vannak vezetve a 4.1. fejezetben levezetett mozgási egyenletek, ebben az esetben a költségfüggvény a nyomaték négyzete. Az algoritmus a differenciálegyenletek megoldására a negyedrendű Runge-Kutta módszer alkalmazza.

5 Szabályzó

A rendszer nemlineáris jellegéből adódóan a szabályzási feladatot nem lehet megoldani csak egy lineáris szabályzóval. Ennek megoldására többféle módszer is létezik, például olyan szabályzási algoritmusok, amelyek lefedik a teljes, globális mozgási tartományt, ilyenek például a neuronhálóval megoldott szabályozások, részleges linearizálást alkalmazó szabályzások vagy két egymástól különböző szabályzó hibrid alkalmazása.



5.1. ábra. A szabályzás tömbvázlata[11, p. 12]

Az elv az, hogy az acrobotot eljuttatjuk az elvárt egyensúlyi pontja közelébe, úgy, hogy megfelelő állapotok szerint érje el azt. Azután egy másik kontroller veszi át az irányítást, amely ott tartja, stabilizálja az egyensúlyi pontja körül. Amint az 5.1. ábrán látható a rendszer állapotát egy egység folyamatosan figyeli, és ezek függvényében kapcsol át a két különböző szabályzó között. Ideális esetben csak egyszer kell átváltani a stabilizálást végző szabályzóra, viszont amennyiben ez zajok miatt nem lett volna sikeres, akkor a kapcsoló visszavált és folytatja tovább a rendszer feltornáztatását.

5.1 Lineáris szabályzó

Ahhoz, hogy alkalmazni tudjuk az LQR szabályzót, linearizálni kell a rendszert a felső egyensúlyi állapota körül (UEP). Az UEP közvetlen környezetében sikeresen fog tudni működni a lineáris szabályzó, mivel az eltérések a nemlineáris és a lineáris rendszer között minimálisak.[2]

UEP pont:

$$q_1 = pi, \qquad q_2 = 0, \qquad \dot{q_1} = 0, \qquad \dot{q}_2 2 = 0$$

(14)

5.1.1 Linearizált modell

A linearizált modellt a következő alakban tudjuk felírni.

$$\dot{x} = A * x + B * u$$

(15)

Ahol az A mátrix a rendszermátrix és B mátrix a bemeneti mátrix.

Linearizálva a rendszert a [12] szerint és behelyettesítve a valós paramétereket:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 28.0744 & -34.4596 & 0 & 0 \\ -31.7716 & 95.9008 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ -168.3284 \\ 407.3343 \end{bmatrix}$$

(16)

A C 4x4 egységmátrix és D 4x1 méretű null-mátrix.

5.1.2 Diszkretizálás

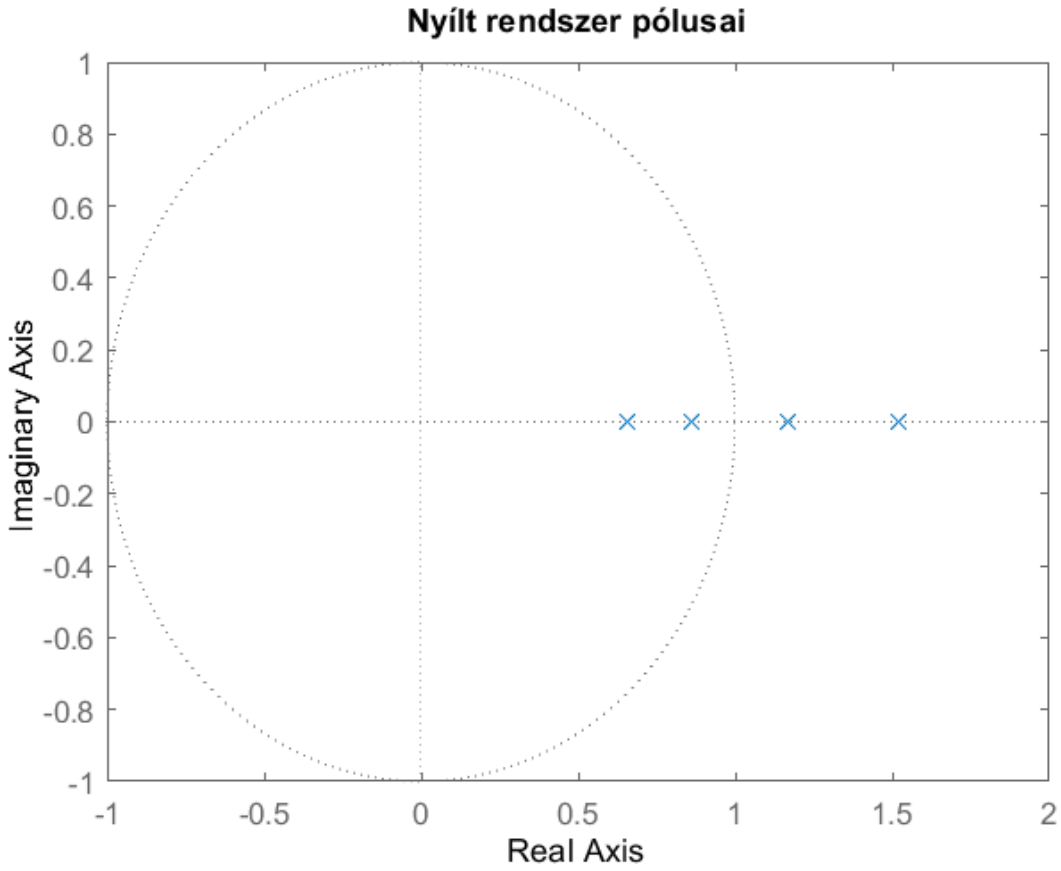
Mivel a valóságban diszkrét értékekkel kell megvalósítani a rendszer szabályozását, ezért diszkretizáltam a rendszert. A diszkrét rendszerre való áttérést a nulladrendű tartó módszerrel végeztem. Mintavételezési idő T_s = 20 ms.

A kapott diszkrét rendszer:

$$A = \begin{bmatrix} 1.02 & -0.02 & 0.04 & 0 \\ -0.02 & 1.078 & 0 & 0.04 \\ 1.14 & -1.42 & 1.02 & -0.02 \\ -1.31 & 3.94 & -0.02 & 1.07 \end{bmatrix} \qquad B = \begin{bmatrix} -0.13 \\ 0.33 \\ -6.93 \\ 16.77 \end{bmatrix} \tag{17}$$

5.1.3 Szabályozhatóság vizsgálata

A nyílt rendszer nem stabil, ami várható volt, mert az instabil egyensúlyi állapota körül linearizáltuk. A rendszer pólusai közül kettő is az egységsugarú körön kívül van (5.2. ábra).



5.2. ábra. A nyílt rendszer instabil pólusai

A rendszer teljesen irányítható, ha az irányíthatósági mátrixának rangja egyenlő a rendszermátrix n-el, tudva, hogy A_{nxn} méretű.[13, p. 145]

A rendszer rangja 4, ugyan úgy, mint az irányíthatósági mátrixé tehát irányítható a rendszer.

5.1.4 DLQR szabályzó

A munkapont körül linearizált diszkrét rendszerre[12] lehet alkalmazni a discrete linear quadratic regulator algoritmust. Ehhez meg kell oldani a diszkrét Ricatti egyenletet[14]. A diszkrét Ricatti egyenlet megoldását (P mátrix) a diszkrét Schur módszert alkalmazva kaptam meg.

Ezután a K erősítésvektor megkaphatjuk

$$K = ((R + B'PB)B'PA)^{-1} \tag{18}$$

Ahol Q 4x4 méretű egységmátrix, R = 100 választva.

$$K = [-9.21 \quad -3.02 \quad -2.36 \quad -0.91] \tag{19}$$

A szabályzójel számítása:

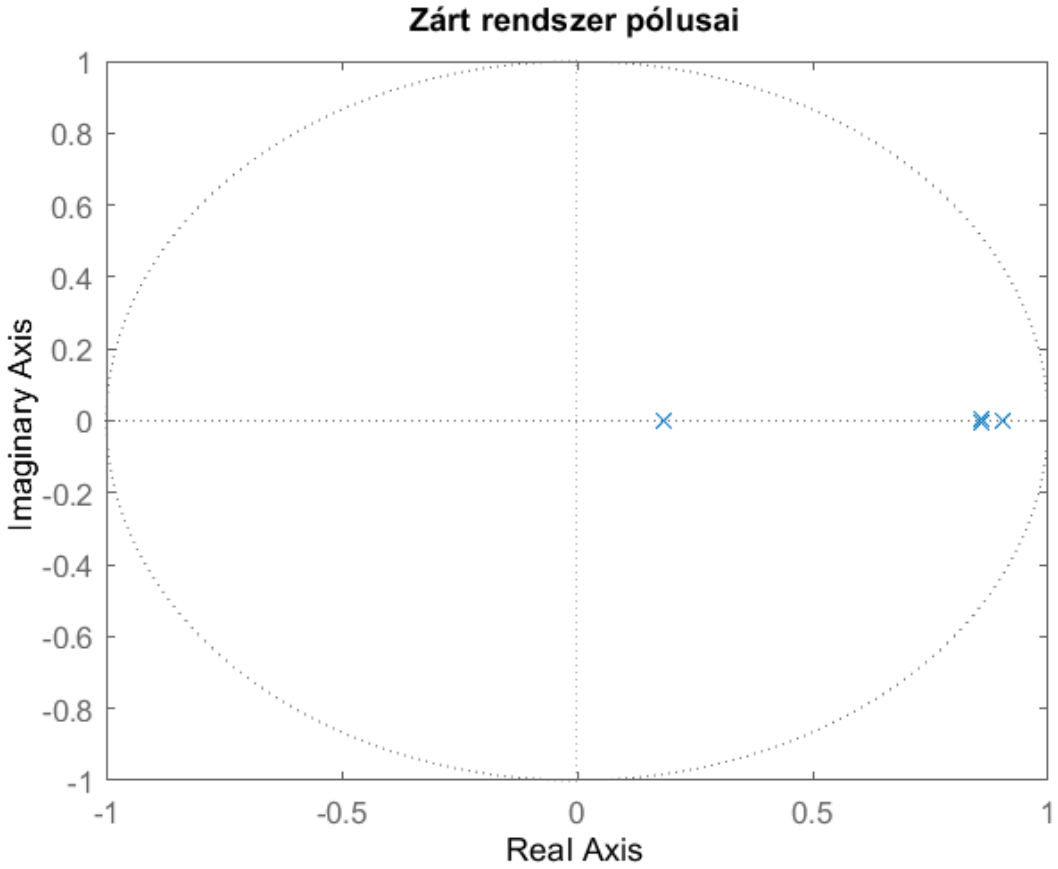
$$u = -K * x \tag{20}$$

Ahol x állapotokat jelöli.

$$x = [q_1 \quad q_2 \quad \dot{q}_1 \quad \dot{q}_2] \tag{21}$$

A zárt szabályzási kört megkaphatjuk, ha az A rendszermátrixból kivonjuk a K erősítésvektor által súlyozott bementi B mátrixot.

A szabályozott rendszer sajátértékei, vagyis pólusai az egység sugarú körön belül vannak, tehát látható, hogy a rendszer stabil (5.3. ábra).

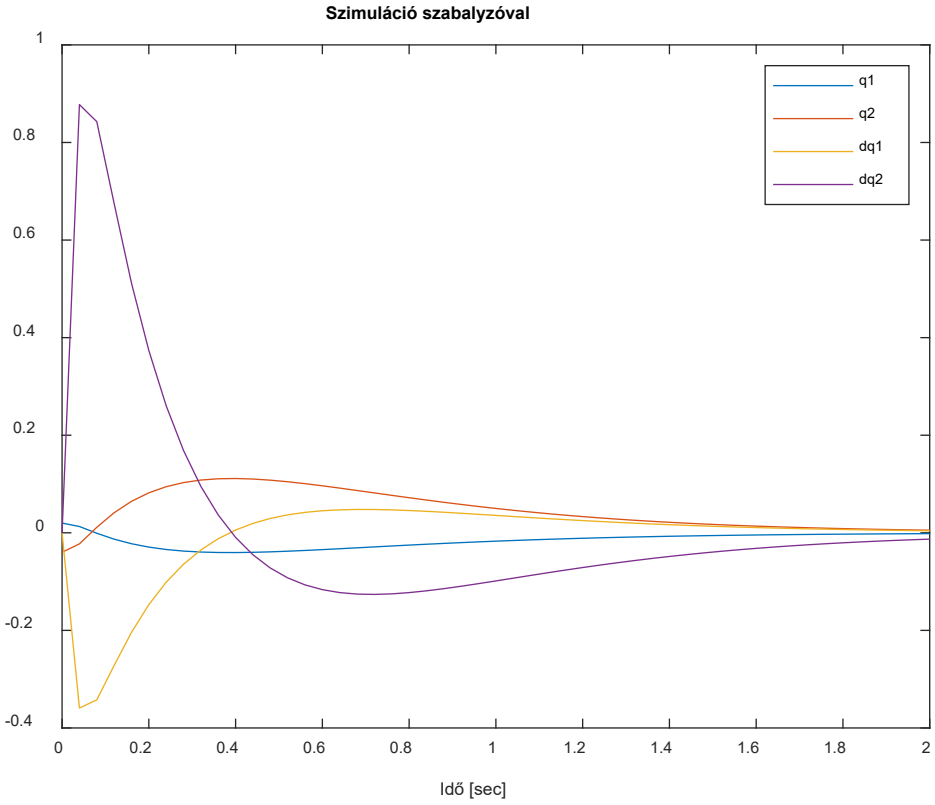


5.3. ábra. A zárt rendszer stabil pólusai

Szimulálva a rendszert látható, hogy az állapotokat a 0-ba vezeti a rendszer. (5.4. ábra.)

A szimulációhoz használt kezdőállapot radiánban:

$$x_0 = [0.02 \quad -0.04 \quad 0 \quad 0] \tag{22}$$



5.4. ábra. A rendszer állapotai szimuláció közben

5.2 Energia alapú szabályozás

Az energia alapú szabályozás elve, hogy a rendszerbe energiát pumpálunk, azzal, hogy a második szegmenst az első szegmens szögelfordulása és gyorsulása alapján kilengessük. Lényegében be kell lengetni a rendszert, addig amíg eléri a felső egyensúlyi pontját. Ezzel a módszerrel folyamatosan mozgási energiát viszünk be a rendszerbe.[3] A feladat, hogy a kezdeti [0 0 0 0] állapotából eljuttassuk a [pi 0 0 0] állapotba.

Az alapötlet az, hogy a belengetjük a második szegmenst egy rögzített ±α szög között, akkor amikor az fázisban van, úgy, hogy az amplitúdója az első szegmens kilengésének minden lengéssel növekedjen. A legegyszerűbb stratégia akkor lengetni a szegmenst, amikor a $\dot{q}_1$ null-átmenetben van.

Ennek megvalósítására ilyen függvény például a szaturációs függvény[3]. Ennek egy kifinomultabb változata az arcus tangens függvény.

$$q_2^d = (2\alpha/\pi)\tan^{-1}(\dot{q_1})$$

(23)

Ezt vezérlőjelként kiadva k_p és k_d erősítési tényezőkkel végezzük, ahol a k_p az ideális pálya és az aktuális pálya közti eltérést erősíti, k_d pedig a gyorsulást.[3]

$$u = k_p\left(q_2^d - q_2\right) - k_d\dot{q_2}$$

(24)

Ahhoz, hogy a belengetés időben is hatékony legyen, és a felső inverz pontot is közel nulla sebességgel érje el a következő értékeket alkalmaztam a szabályzáshoz:

$$\alpha = 1\,[rad], \quad k_p = 450 \quad k_d = 0.3$$

(25)

5.3 Switch logic

A két szabályzó között csak a megfelelő állapotban szabad váltani. Fontos, hogy az UEP pontot lehetőleg minimális szöggyorsulás mellett érje el. A feltételt, ahol a szabályzó átválthat a legtöbb esetben empirikus módon határozzák meg. A következő feltételt vettem alapul.[15, p. 15]

$$|q_1| + |q_2| + 0.1|\dot{q_1}| + 0.1|\dot{q_2}| < \xi$$

(26)

Sokszor azonban az átváltás pillanatában a lineáris szabályzó kezdeti beavatkozó jele nagyon nagy lehet, amit a fizikai rendszerben már nem lehet megvalósítani. Ennek elkerülésére én figyelembe vettem a lineáris szabályzó jelét is. Ahhoz, hogy a rendszer sikeresen tudjon váltani a két szabályzó között a DLQR szabályzó jele is egy határon belülre kell kerüljön. A feltétel a váltásra így a következőképpen alakult

$$|q_1| + |q_2| + 0.1|\dot{q_1}| + 0.1|\dot{q_2}| + |u| < \xi$$

(27)

Ahol az u a lineáris szabályzó által kiszámított szabályzójel. Az ξ értéke, amelynél a szabályzó sikeresen átválthat 1.6.

Fontos, hogy az átváltás után megtörténhet, hogy a fenti feltétel újra nem teljesül kilengések miatt, ügyelni kell arra, hogy ne váltsunk vissza a feltornáztatáshoz használt szabályzóra. Különbséget kell tenni a sikertele szabályozás és az ideiglenes kilengések között úgy, hogyha sikertelen, akkor térjen vissza az acrobot feltornásztatásához. Ehhez egy felsőbb limitet választottam ξ = 15. Ezt meghaladva a váltó logika sikertelennek veszi a lineáris tartományban a szabályzást és visszavált az első fázisban használt szabályzóra.

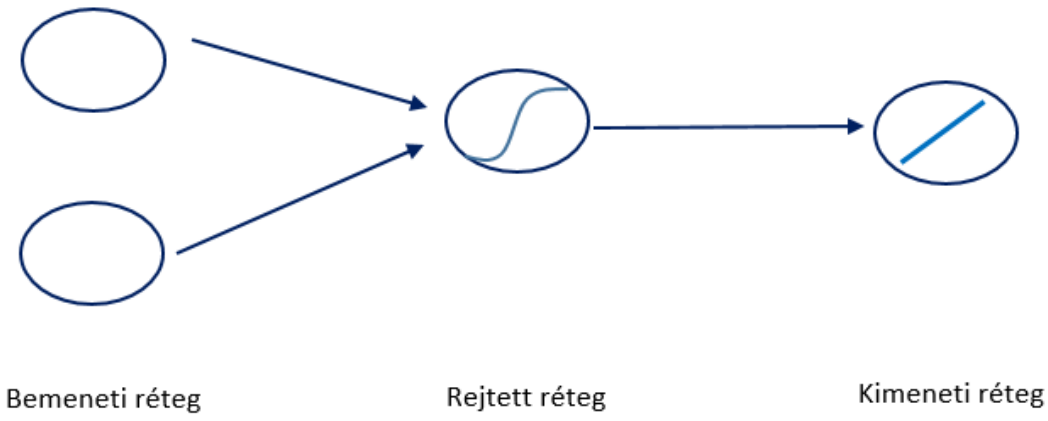
6 Neuronháló

Mivel a szabályozást valós környezetben is megkell valósítani, így törekedtem az egyszerű kivitelezésre, olyan számításokra, amelyeket egy mikroprocesszoron is el lehet végezni. A szervomotornak egy előírt szöget kell megadni és nem nyomatékot, tehát a szabályzó jel kiszámítása után (ami egy erő, nyomatékban kifejezve) ahhoz, hogy megkapjam az előírt szöget, valós időben kéne szimulálni a rendszert, kiintegrálni a mozgási egyenleteket. A szimuláció alapján kigenerált szabályzó jelet kiadva a valós rendszernek csak off-line irányítást kapnánk, mivel nincs visszacsatolás, nem vesszük figyelembe az állapotokat. Ahhoz, hogy on-line szabályozás legyen, és elkerüljem a nagy számítási igényt az előírt szög kiszámítására, ezért alkalmaztam egy MLP típusú neuronhálót. A neuronhálót a Simulinkben elvégzett szimuláció alapján tanítottam, a mért q_1 és $\dot{q}_1$ állapotok szerint tudja kiszámítani a második szegmens előírt szögét. A szervo szabályozás erre az előírt szögre kell vezérelje a motort. Két neuronhálót alkalmaztam külön a két szabályzóra. A neuronháló előnye, hogy a betanítás után nagyon kevés számítási időt igénylő számításokkal is elérhető egy olyan szabályozás, ami a rendszer állapotai függvényében számolja ki az előírt szöget. Viszonylag nagy pontossággal sikerült megtanulja az összefüggéseket, még úgy is, hogy csak a passzív csukló állapotai vannak mérve a valós rendszeren, tehát a betanítás is csak ezek alapján történt.

6.1 Egyrétegű előrecsatolt neuronháló felépítése

A neuronháló 3 fő részre osztható, az első a bemeneti réteg, amely továbbítja az adatokat a háló többi részéhez. A második a rejtett réteg, amely egy neuront tartalmaz, és a harmadik a kimeneti réteg. Két bemeneti neuron van, mivel két állapotot mérünk, a szögelfordulást és a szögsebességet.

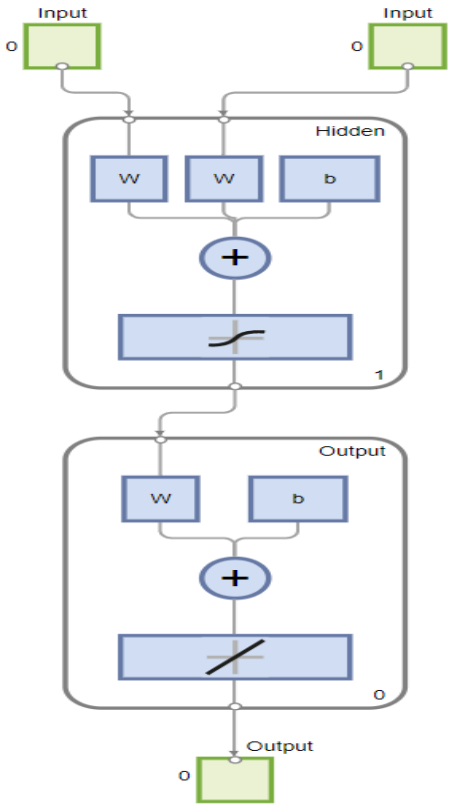
Egy kimeneti neuron található, mivel egy vezérlőjelet kell kiszámítsunk. A neuronháló bemenetén az adatokat előfeldolgozni kell, így az eltolt, és skálázott értékek kerülnek a rejtett rétegre.



6.1. ábra. Egy rétegű, egy neuront tartalmazó neuronháló felépítése, az aktivációs függvényeket feltüntetve

6.2 Megvalósítás

A megvalósítás a Matlab feedforwardnet[16] beépített függvénnyel és az ahhoz kapcsolódó funkciókkal történt. A tanításhoz a Levenberg-Marquardt algoritmus használtam, a hibaszámolás az átlagos négyzetes eltérés szerint történik. A rejtett réteg aktivációs függvénye tansig, a kimeneti rétegé purelin aktivációs függvény.

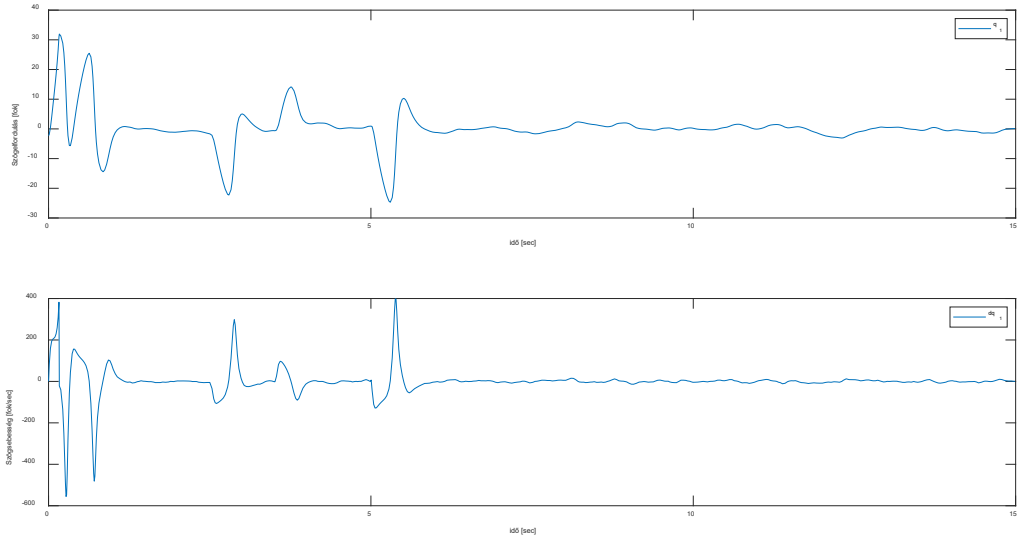


6.2. ábra. Matlabban megvalósított neuronháló tömbvázlata

6.2.1 DLQR

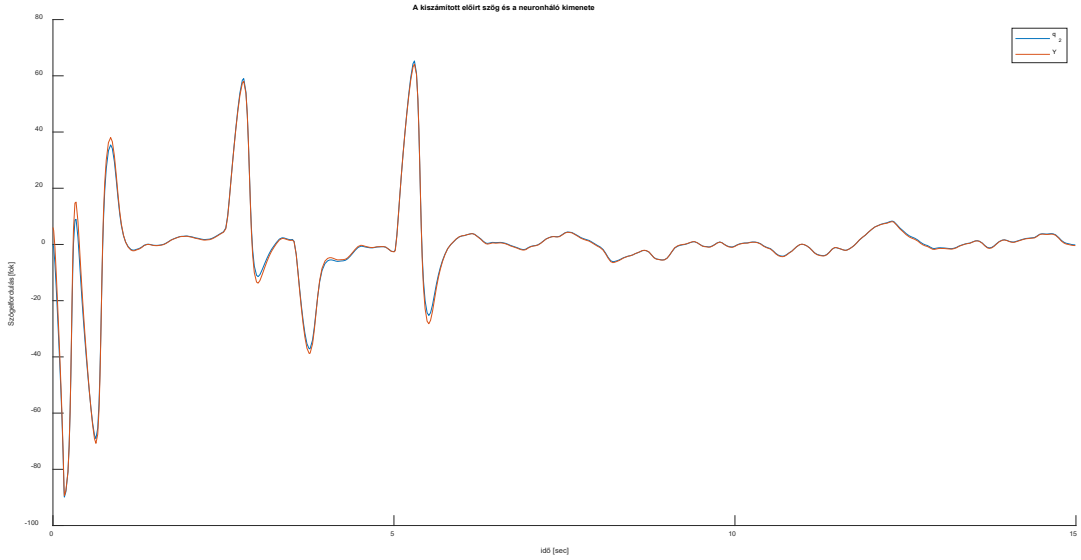
A 6. fejezetben szimuláltam le a szabályozást. A kezdőállapotokat a lineáris tartomány szélére érdemes választani, amelyekre még sikeres a lineáris szabályzó, hogy a tesztjelek, amikre tanítom a neuronhálót, minél nagyobb tartományt fedjenek le. A rendszerre egy konstans fehér zajt és 3 előre meghatározott pontba egy nagyobb zajt is adtam, amelyek célja az volt, hogy valósághűbb legyen a szimulált szabályozás és minél aktívabb legyen a szabályzó, így több hasznos adatot generálva a tanítóhalmazhoz.

Az (6.3. ábra.) ábrán látható a DLQR szabályzót alkalmazva az első szegmens időbeli szögelfordulása és szögsebessége. Mivel a DLQR szabályozás a (a 4.1. ábra szerinti koordináta rendszerben) π radián értékre kell eljuttassa az első állapotot, ezért a neuronhálónak is így adtam meg az adatokat, π radiánnal eltolva. A könnyebb átláthatóság kedvéért ezeket az értékeket ábrázoláskor a továbbiakban átalakítottam fok, illetve fok/másodperc mértékegységekké.



6.3. ábra. Tanítóhalmaz (DLQR szabályozás fehér zajjal)

A (6.4. ábra) ábrán látható a tanítás eredménye. Látható, hogy sikeresen megtudta tanulni az összefüggést az első szegmens állapotai és az előírt szöge között. A legnagyobb eltérések 3-5 ° közöttiek, amelyek csak a gyors, intenzív beavatkozásoknál jelennek meg, viszonylag kevés ideig.

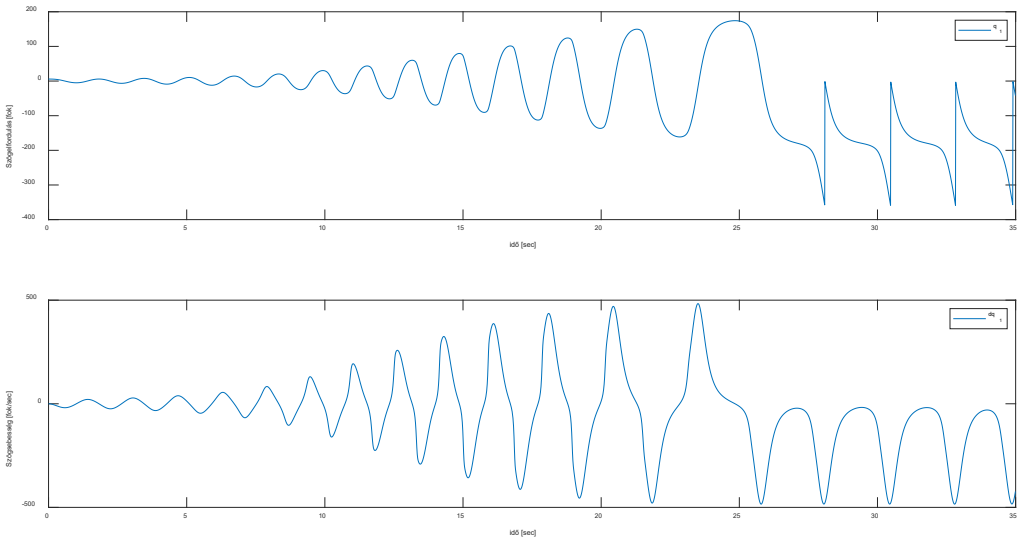


6.4. ábra. A neuronháló kimenete összehasonlítva a kiszámított előírt szöggel

6.2.2 Swing up

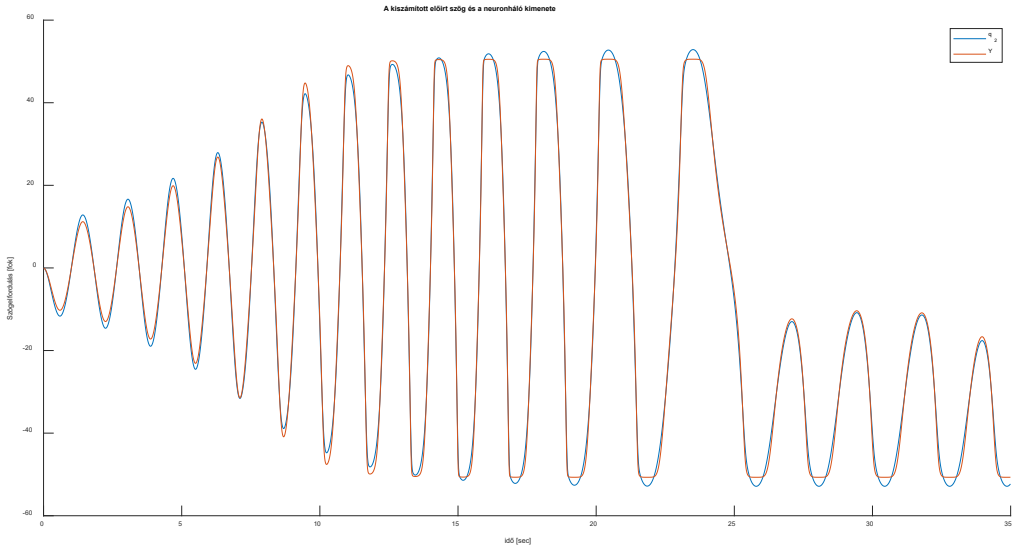
A swing-up, vagyis feltornáztató fázisra is ugyan azt a neuronháló felépítést és tanítási algoritmusokat alkalmaztam. Leszimulálva egy sikeres feltornásztatást, és nem átkapcsolva a DLQR szabályzóra (5.1.4) megkaptam egy az 6.5. ábra alsó felében látható q2 pályát időben. Ezt a pályát követve a valós rendszer is fel kell tudja tornáztatni magát az UEP pontjának közelébe. Az 6.5. ábra első grafikonján látható, hogy a szabályzó sikeresen eléri a 180°-ot 25 másodperc után, viszont ott nem tudja megtartani magát, ilyenkor kell majd alkalmazni, az ezen a tartományon pontosabb DLQR szabályzót.

Mivel a valós rendszeren is a [-2π, 2π] tartományban dolgozik a szabályzó, így a tanítóhalmazban is ezen a tartományon belülre korlátoztam a szögeket. Ilyen abban az esetben fordulhatott elő, ha akár többször is körbe fordult az acrobot. Ez látható a 28. másodperc után.



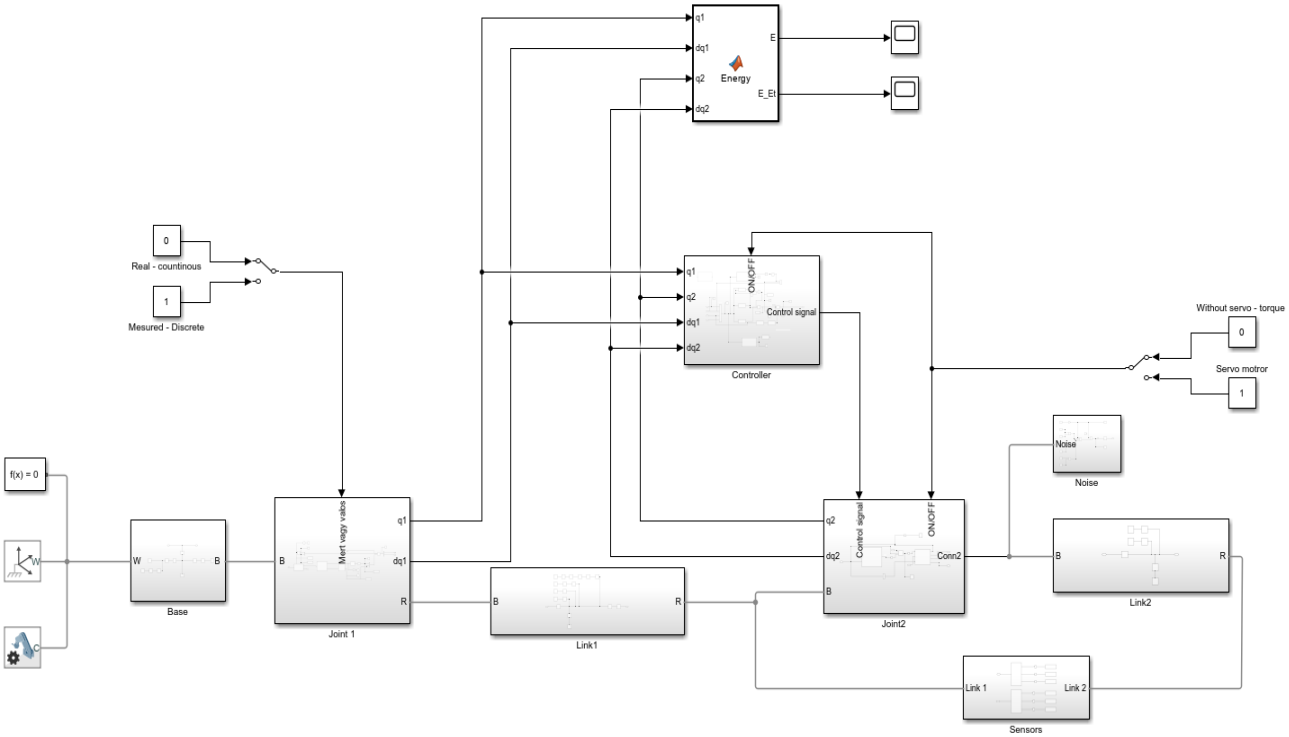
6.5. ábra. Tanítóhalmaz a feltornáztatáshoz (Swing up szabályozás)

A tanitás eredménye látható a (5.6. ábra) ábrán. A legnagyobb különbség az előírt q_2 szögpálya és a neuronháló által kiszámított között, 2°, tehát a tanulás itt is sikeresnek bizonyult.



6.6. ábra. A neuronháló kimenete és a kiszámított q_2 szög összehasonlítása

7 Modellezés, szimuláció



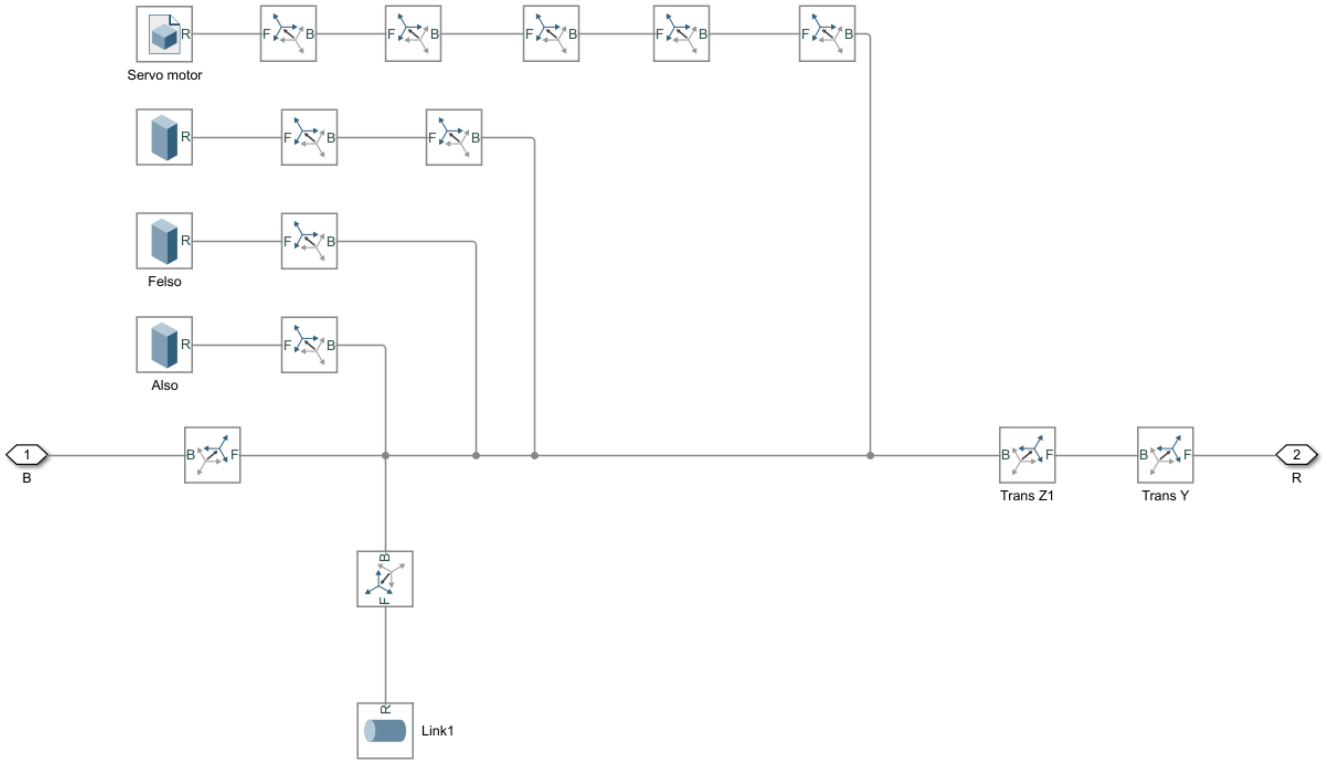
7.1. ábra. Az acrobot és a szabályzás Simulink modellje

A modellezésre a Simulink-hez tartozó Simscape[17] csomagot használtam, simulink elemekkel egybekötve. A simscape egy különálló szoftvercsomag a Simulinken belül, amely a fizikai elemek modellezésére szolgál. Ennek segítségével hoztam létre az acrobot 3D-s modelljét, megadva a pontos paramétereket. Ezek a következő alrendszerek: Base, Joint1, Link1, Joint2, Link2, amelyek a (7.1. ábra) ábrán láthatóak. A Sensors nevű alrendszerben mérem külön a két szegmens egyes paramétereit, mint például az össztömeg vagy az inercia. A Noise alrendszerben találhatóak a zaj generáláshoz szükséges elemek, amelyek a második szegmensre hatnak. Látható, hogy a Controller tömb megkapja a rendszer állapotait, majd a vezérlőjelet kiszámítva hat a második, aktív csuklóra. A jobb oldalon látható kapcsolóval lehet választani, hogy a második csuklót direktben nyomatékkal vezéreljük, vagy beiktassuk a szervomotort, azáltal megvezérelve a második csuklót. A baloldalon található kapcsolóval lehet állítani, hogy a szabályzó a Simulink által mért szögeket és szöggyorsulásokat használja, vagy a diszkért, forgójeladó (incremental rotary encoder) által kiszámított értéket. Ezzel a megvalósítással lehet tanulmányozni, hogy a forgójeladó pontossága, hogyan hat a számításokra, itt megtalálható egy dekóder is, ami az enkóder által szolgáltatott jelekből kiszámolja a szögelfordulást és a szögsebességet. Fent látható a rendszer energiáját kiszámító Simulink tömb, amely a [15, p. 3]szerint számolja ki a rendszer pillanatnyi energiáját. A második kimenetén található a cél energia (0 mozgási energia és potenciális energia a felső invertált helyzetében) és az aktuális energiája közti eltérés. A szimuláció az ode23t[18] módszert használja numerikus integrálásra, amely egy időlépéses, adaptív integráló algoritmus, amely ebben az esetben hatékonyabbnak bizonyult, mint az ode45. A paraméterek kiszámítását, a szimuláció elkezdését, majd az eredmények megjelenítését Main (4.függelék) program fogja össze.

7.1 A modell felépítése

7.1.1 Fizikai elemek

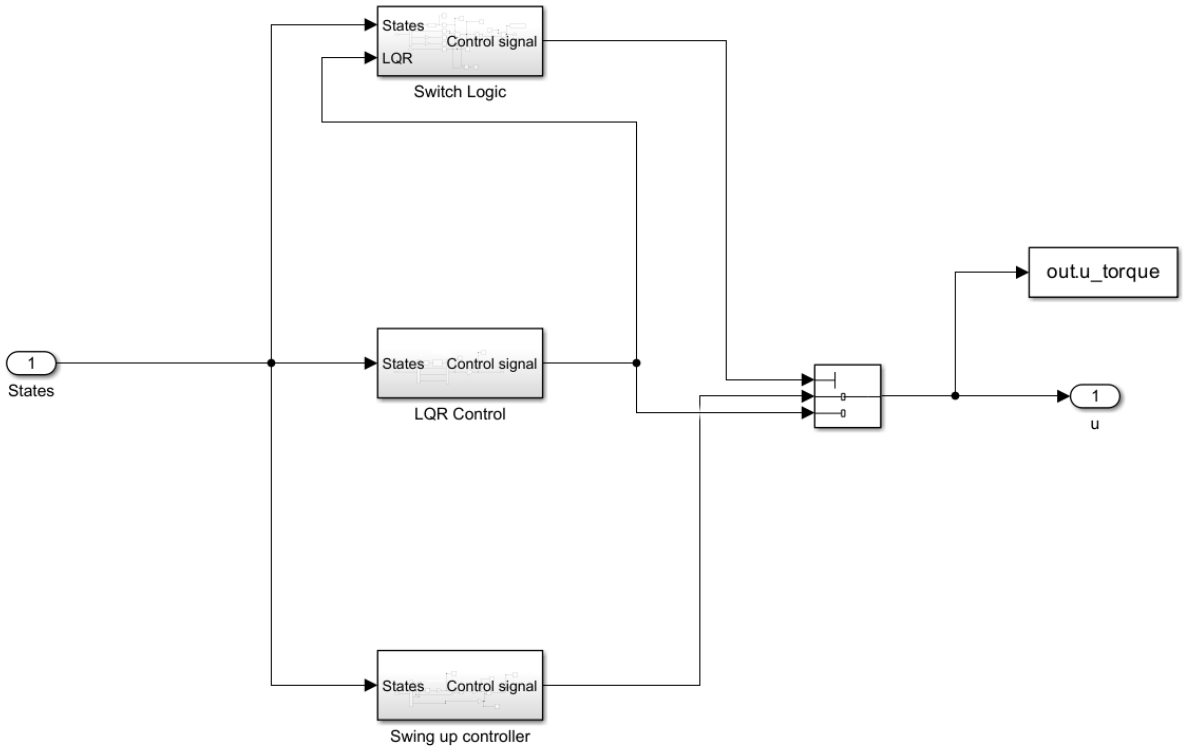
A Simscape csomag által könnyen meglehet adni a fizikai elemek tulajdonságait, mint a méretük, tömegük. Könnyedén és alrendszerekre bontva le lehet modellezni az egész robotot ezáltal.



7.2. ábra. A második szegmens Simscape modellje

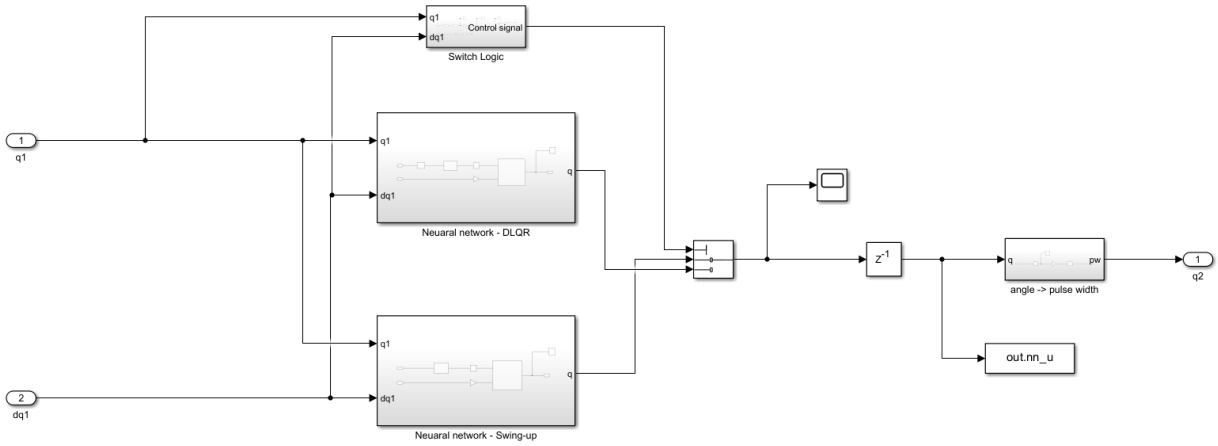
A (7.2. ábra) ábrán látható a második szegmens fizikai modellje. Megtalálható a rúd, a végén lévő összekötő elemek, és a szervomotor 3D-s modellje is.[19]

7.1.2 Szabályzó



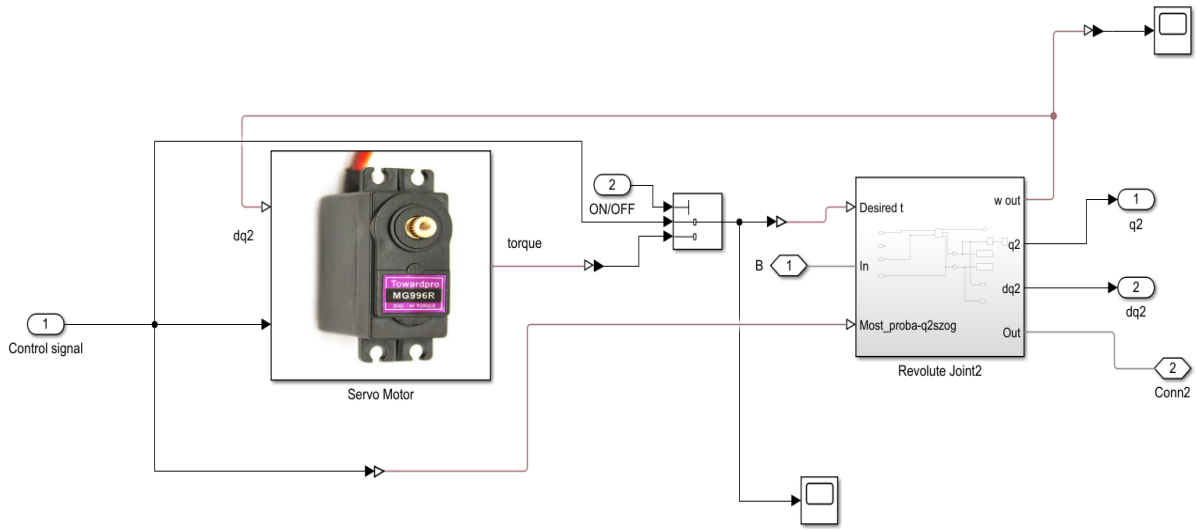
7.3. ábra. A szabályzó tömbvázlata

A szimulációban megtalálható mind a DLQR és energia alapú szabályzó (7.3. ábra), mind ennek a neuronhálóval való implementálása (7.4. ábra). A neuronhálók Simulink modelljét, vagyis az adatok feldolgozását, majd a súlyokkal való szorzását a tanítás után a gensim beépített függvénnyel generáltam ki.



7.4. ábra. A neuronhálóval megvalósított szabályzás

7.1.3 Szervomotor



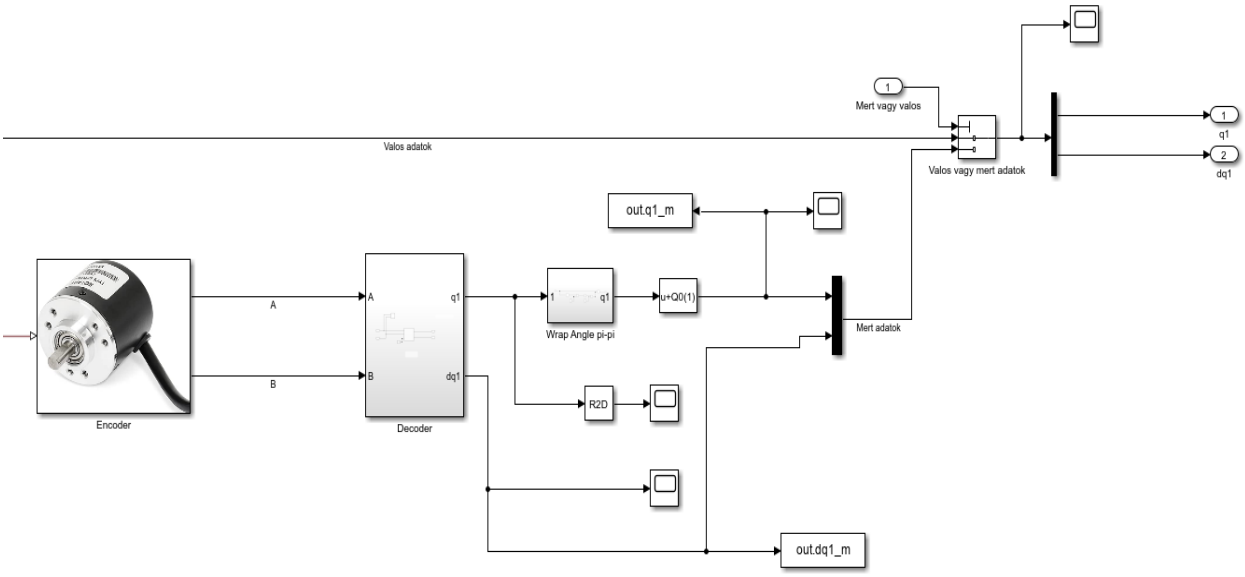
7.5. ábra. A szervomotor és a második csukló Simscape modellje

A Servo Motor alrendszerben található a szervomotor Simscape modellje, a valóságban is használt szervomotor katalógus adatai szerint. Itt látható az ON/OFF kapcsoló szerepe is, amely vagy direktben

a vezérlő jellel hat a csuklóra, ami nyomatékban van kifejezve, vagy a szervomotor által szolgáltatott nyomaték mozdítja el a második szegmenst. Ebben az esetben a vezérlőjel egy előírt szög kell legyen, amit majd a szervomotor követ.

7.1.4 Forgóérzékelő

Az első csuklóban található a forgóérzékelő, vagyis az incremental rotary encoder.

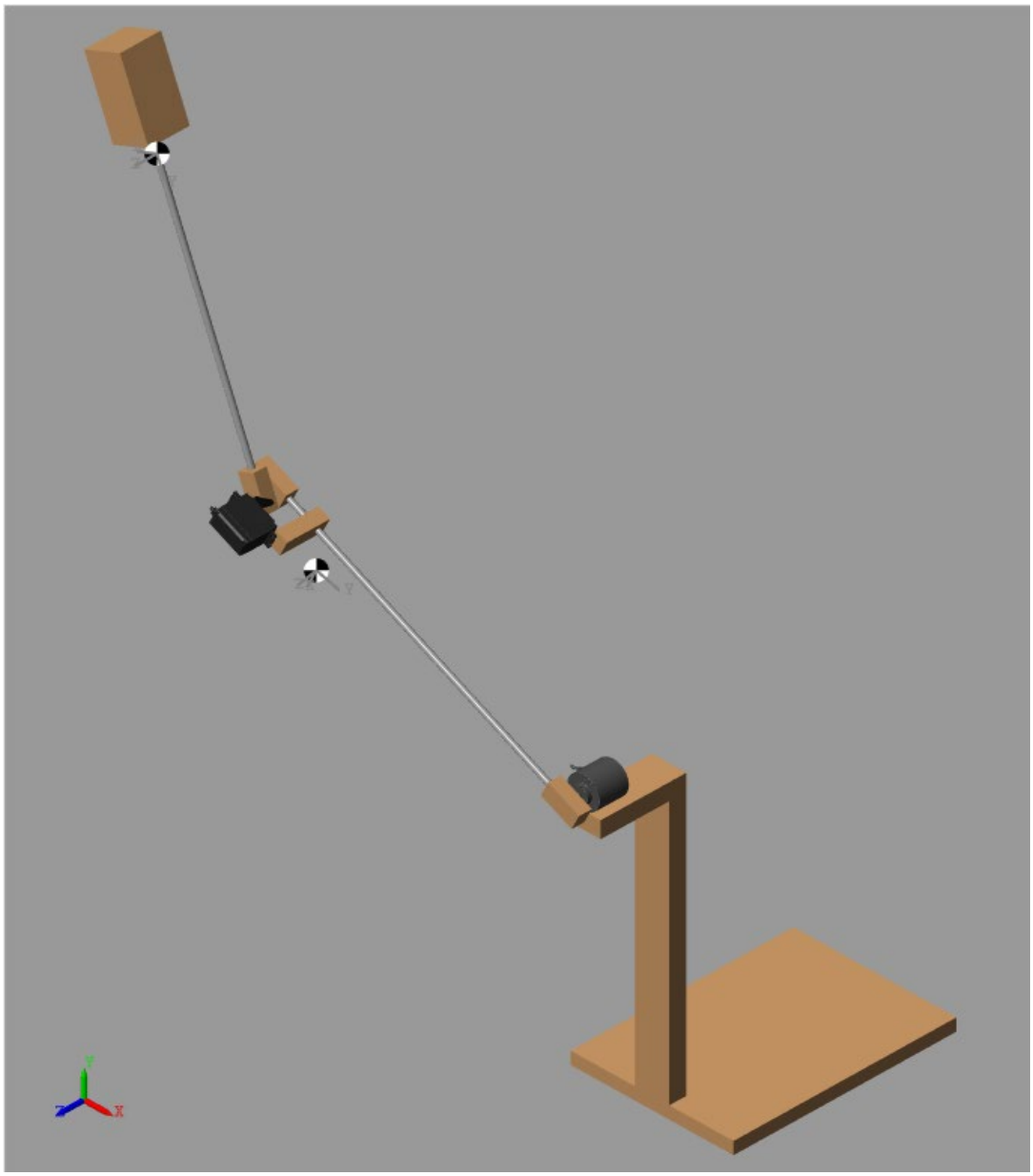


7.6. ábra. Az enkóder és az első csukló Simscape modellje

Itt is megtalálható a forgóérzékelő Simscape modellje[20], illetve egy dekóder is, ami az A és B négyszög jeleket dolgozza fel és számolja ki a diszkrét szöget és szögsebességet. Látható továbbá itt is a második kapcsoló, amivel válthatunk a Simulink által generált adatok és a forgóérzékelő és dekóder által szolgáltatott adatok között.

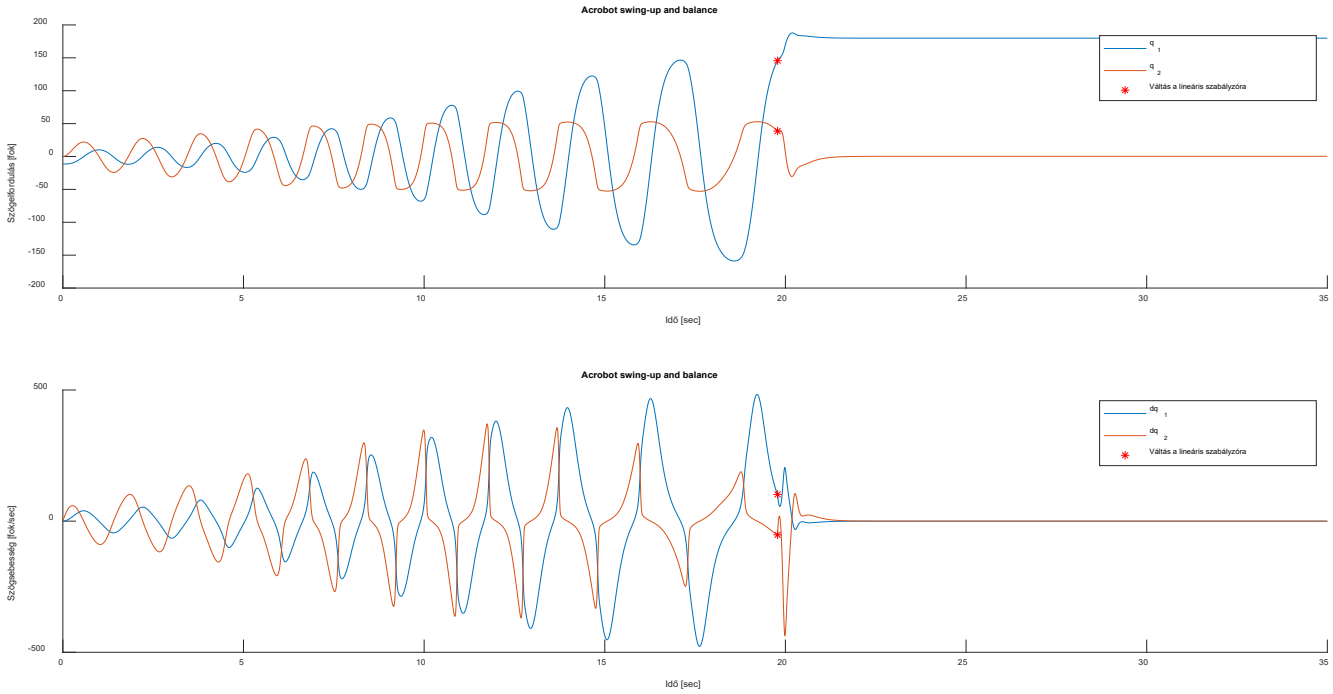
7.2 Eredmények

A valós rendszer paraméterei szerint elvégeztem a szimulációt, a kezdőállapot: $X_0 = [0.1\ 0\ 0\ 0]$. Ha a rendszer kedőállapotát a [0 0 0 0] -ban határozzuk meg, akkor nem fog onnan kimozdulni, tehát érdemes megadni egy minimális 0-tól különböző kezdőállapotot. Ezzel a kezdeti feltornáztató szakasz időbeli hossza is rövidül. A fejlesztőkörnyezetben nyomon lehetett követni a rendszer viselkedését (7.7. ábra).



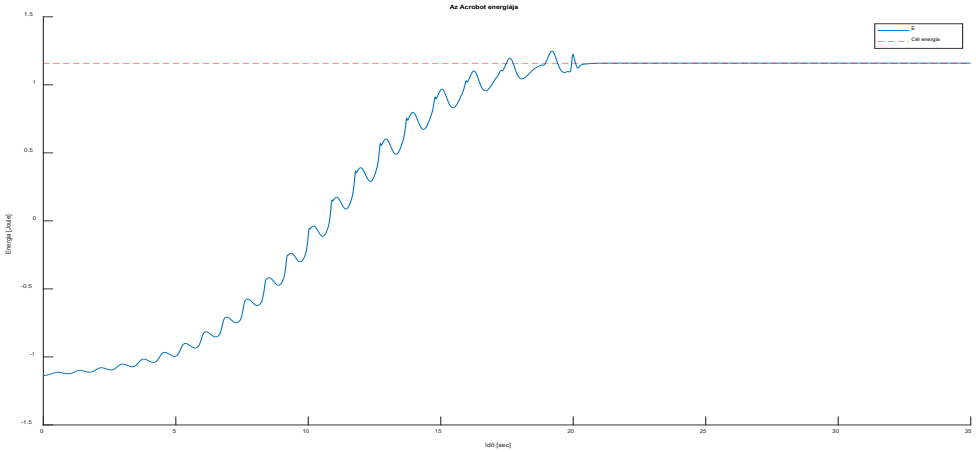
7.7. ábra. Az acrobot 3D-s modellje szimuláció közben

A (7.8. ábra) ábrán láthatunk egy sikeres feltornászást és stabilizálást az UEP pont körül, csillaggal jelölve a váltás időpontját a feltornászást végző szabályzóról a stabilizálást végző lineáris szabályzóra.

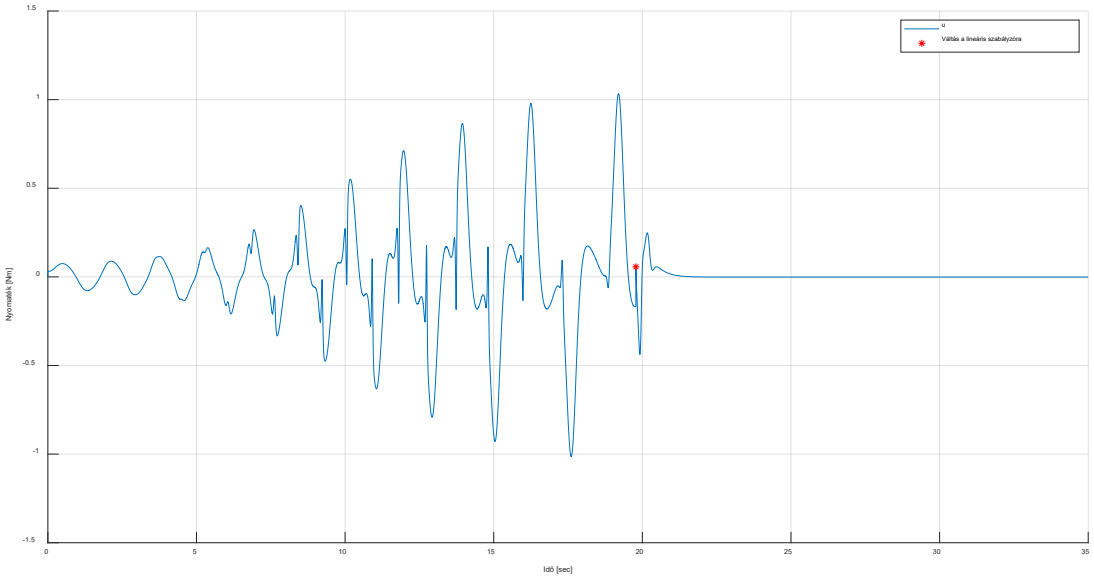


7.8. ábra. Az acrobot szögeinek és szögsebességeinek változása időben egy sikeres szabályzás alatt

A (7.9. ábra) ábrán láthatjuk a rendszer összenergiáját a szabályozás ideje alatt. A célenergia értéke 1.16 J, vagyis csakis a rendszer potenciális energiája az inverz helyzetében.

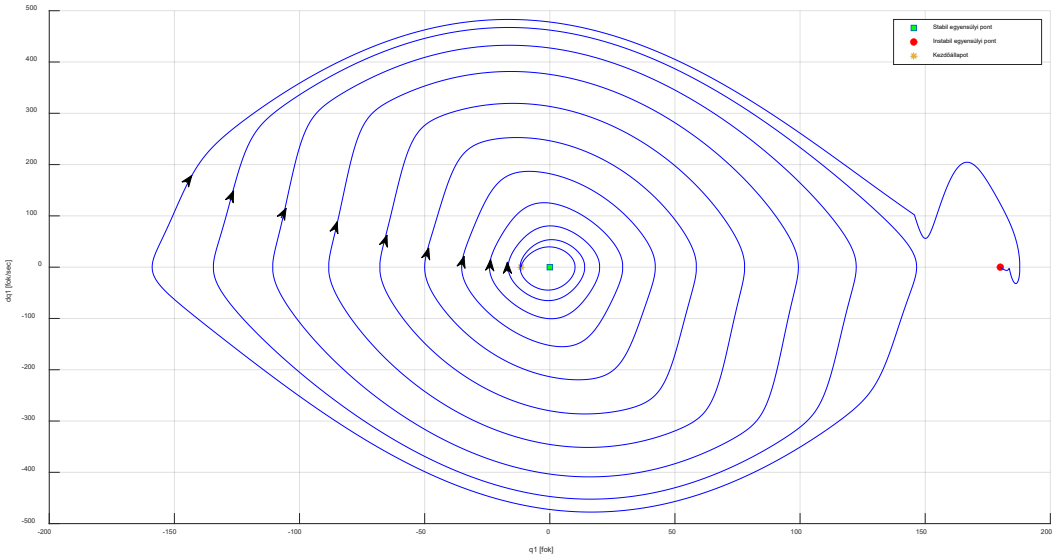


7.9. ábra. A teljes rendszer energiája és a cél energia a szabályozás alatt



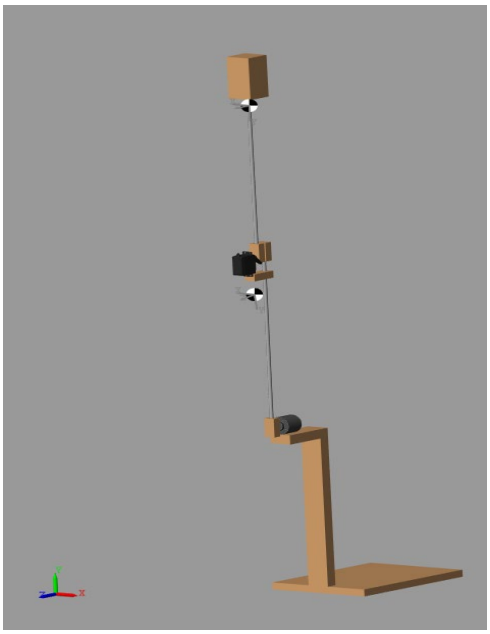
7.10. ábra. A szabályzó jel változása Nm-ben a szabályozás közben

A (7.10. ábra**)** látható a szabályzó jel, vagyis a második csuklóra ható nyomaték. Ezt a fizikai rendszeren nem tudjuk közvetlenül ráadni a rendszerre a szervomotorral, ott az előírt pályát követi a rendszer.



7.11. ábra. Az acrobot fázisportréja, az első szegmens szögelfordulása és szögsebesség függvényében

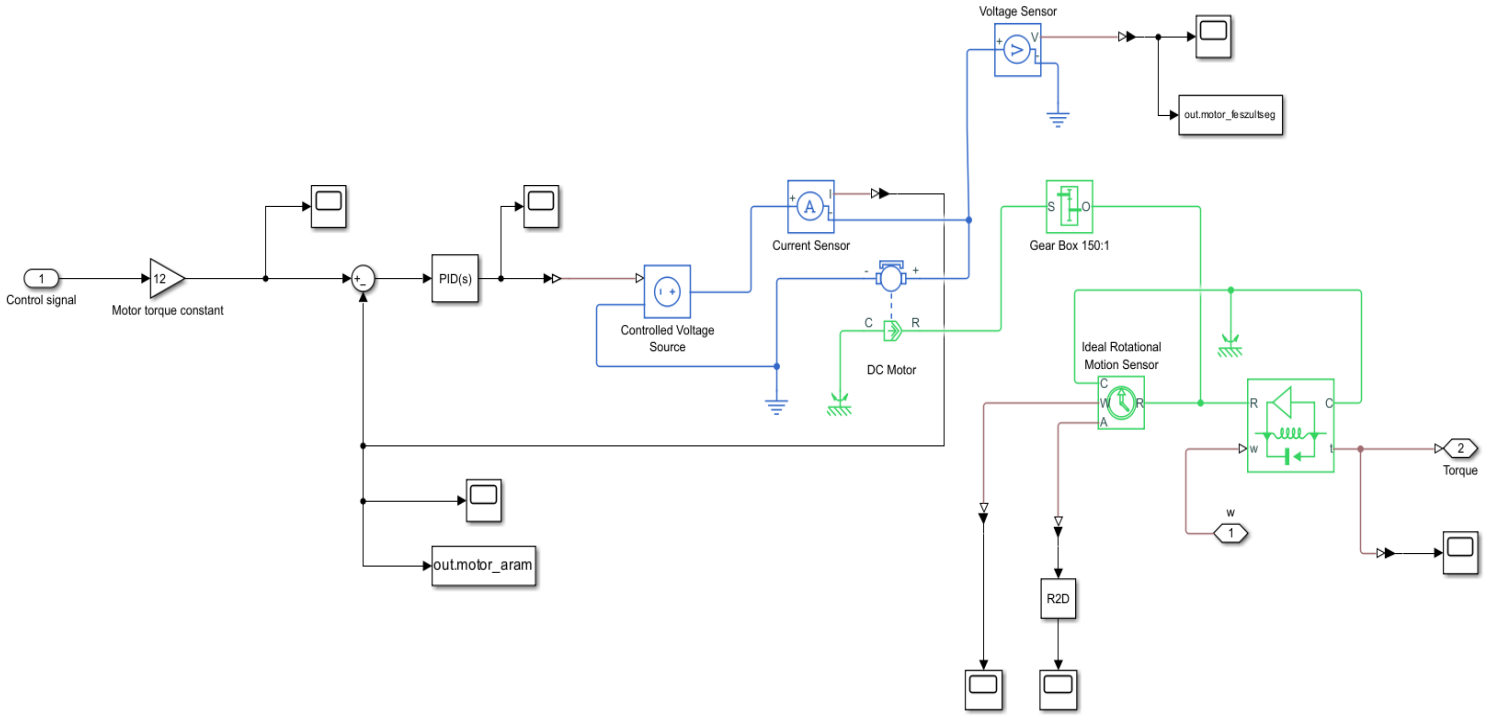
A (7.11. ábra)-n látható a fázisportré egy része a teljes szabályozási szakasz alatt. Az első szegmens szögsebessége látható a szögelfordulás függvényében. Látható, ahogy a 0 közeli kezdeti állapotból eléri az instabil egyensúlyi helyzetét, 180 °-ot, 0 sebességgel.



7.12. ábra. Az acrobot 3D-s modellje a felső inverz pontjában

7.3 Modellezés DC motorral

Ahogy a (bevezetőben vagy hol) említettem a gyakorlati megvalósítás során a szervomotornak előnyei és hátrányai is vannak. Egy másik megoldás lehet az egyenáramú motorok használata egy fogaskerék áttételt használva a kellő nyomaték eléréséhez. Ennek a hátránya a motor ára, mivel egy viszonylag kis méretű motorral kell elérni akkora nyomatékot, ami eltudja fordítani a második szegmenst, a rajta levő súllyal együtt. Másik hátránya a nehezebb gyakorlati szabályozása, mivel nyomatékszabályozást kell végezni, nem a tengely sebességét kell szabályozni, hanem az azon megjelenő nyomatékot. Amint a (7.10. ábra)-n látható a szabályzójel, vagyis a nyomaték időben gyorsan változik, ezt a nyomatékot kell előállítani a motor tengelyén is, lehetőleg úgy, hogy a tranziensek hatása minél kisebb legyen. Ennek tanulmányozására elkészítettem a Simulink környezetben a modell szabályozását a szervomotor helyett egy egyenáramú motort használva (7.13. ábra).



7.13. ábra. Egyenáramú motor szabályozása Simulinkben

A modellezésben a Simscape által megadott általános DC motor modelljét használtam, ehhez csatlakoztatva egy nyomatékváltót 150:1 arányú áttétellel. Egy feszültség-, és árammérővel mérem a motoron sarkain levő feszültséget, illetve a motoron átfolyó áramot. A Simulinkben használt DC

motor kimenetén egy ú.n. “Rotation” mennyiség jelenik meg, ezért szükséges azt nyomatékká alakítani ahhoz, hogy kompatibilis legyen a modell többi részével, továbbá itt is mérem a szögelfordulást és a szögsebességet.

A szabályozás elve az, hogy a motornyomaték arányos a motoron átfolyó áramerősséggel.[21]

$$T = k_T \cdot I$$

(28)

T a motor tengelyén megjelenő nyomaték Nm-ben kifejezve, k_T a motor nyomaték állandója [Nm/A], és I a motoron átfolyó áramerősség [A].

Az arányosságot a motornyomaték állandó határozza meg, ami minden motornál különböző lehet. Ezt meglehet határozni a katalógus adatok, vagy mérések alapján. Ideális esetben a nyomaték szabályozását egy áramszabályzóval lehet megvalósítani, viszont mivel ez a gyakorlatban nehezen kivitelezhető, ezért már a modellben is a feszültséget szabályozom úgy, hogy a mért áram legyen egyenlő a referencia áramerősséggel. A modellben a motornyomaték állandót 0.083 Nm/A -re választottam, tehát ennek az reciprokával való szorzása (12) a nyomatéknak megadja az ennek megfelelő áramerősséget. A szabályozást egy PID szabályzó végzi a *2*.táblázat táblázatban látható paraméterek szerint. A szabályzó bemenetére a referencia jel és a mért áramerősség különbsége van csatolva. Fontos, hogy a szabályzó lekövesse a gyors változásokat a referenciajelben, de pontos is legyen a az UEP pont közelében, ezért felel az integráló tag. A feszültség változtatását egy ideális változtatható feszültségforrás végzi.

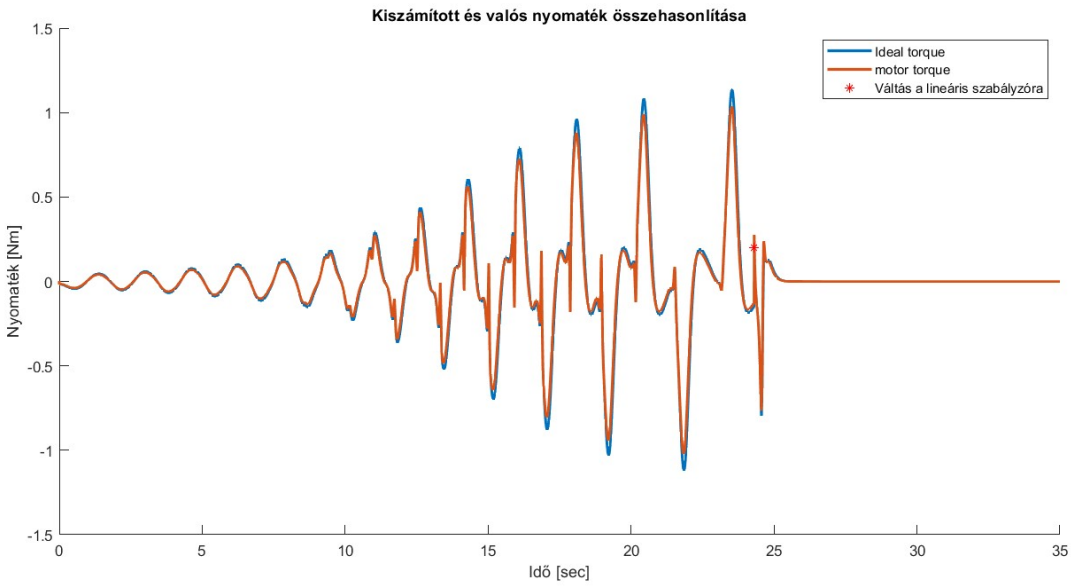
P	**10**
I	**0.1**
D	**0.5**

2.táblázat. *PID szabályzó paraméterei*

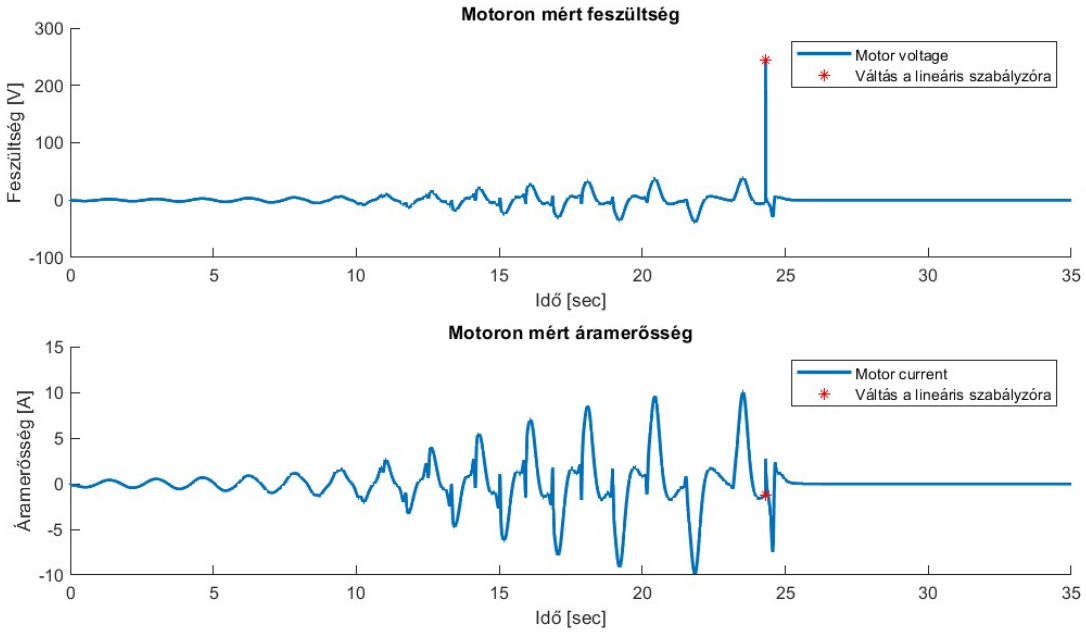
7.3.1 Eredmények

A szimulációban sikeres volt a szabályozás, az acrobot elérte a felső egyensúlyi pontját, majd ott tudta stabilizálni magát. A (7.14. ábra)-n látható a kiszámított ideális nyomaték és a motor tengelyén megjelenő nyomatékok összehasonlítása. Megfigyelhetőek eltérések a két érték között, de ennek ellenére a szabályzó sikeresen tudta szabályozni a rendszert.

A (7.15. ábra)-n látható a motoron mért feszültség és áram külön ábrázolva. Elsősorban látszik, hogy az áram arányos a nyomatékkal, mert a két jel időbeni változása szinte azonos. A feszültségnél látható egy kiugró érték, ami természetesen csak az elméleti modellben jöhet létre. Ez a két szabályzó közötti váltásnál történik, ahol a nyomaték, nagyon rövid idő alatt az ellenkező irányban kell, hogy elmozdítsa a második szegmenst.



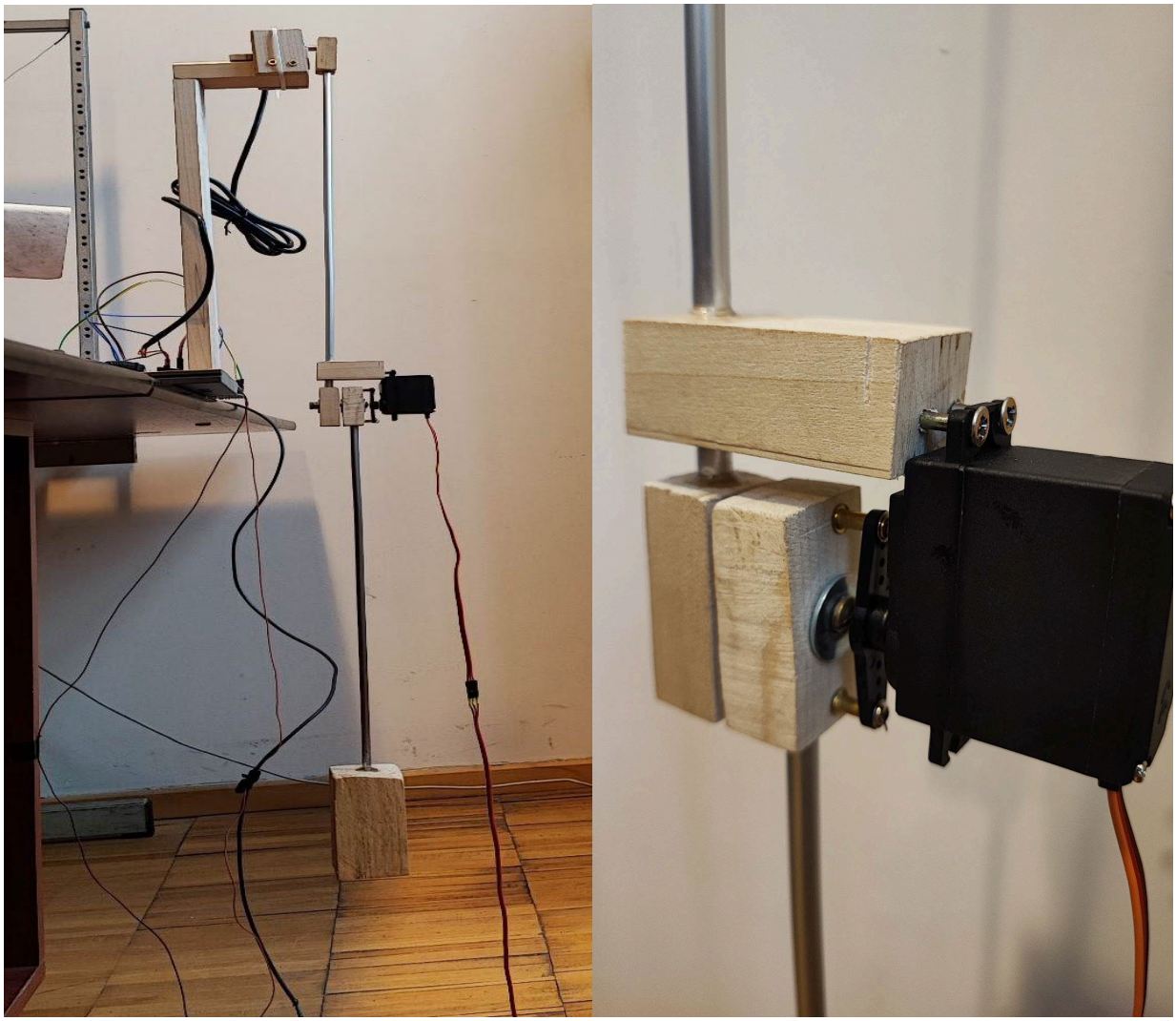
7.14. ábra. Ideális és a motor tengelyén megjelenő nyomatékok összehasonlítása



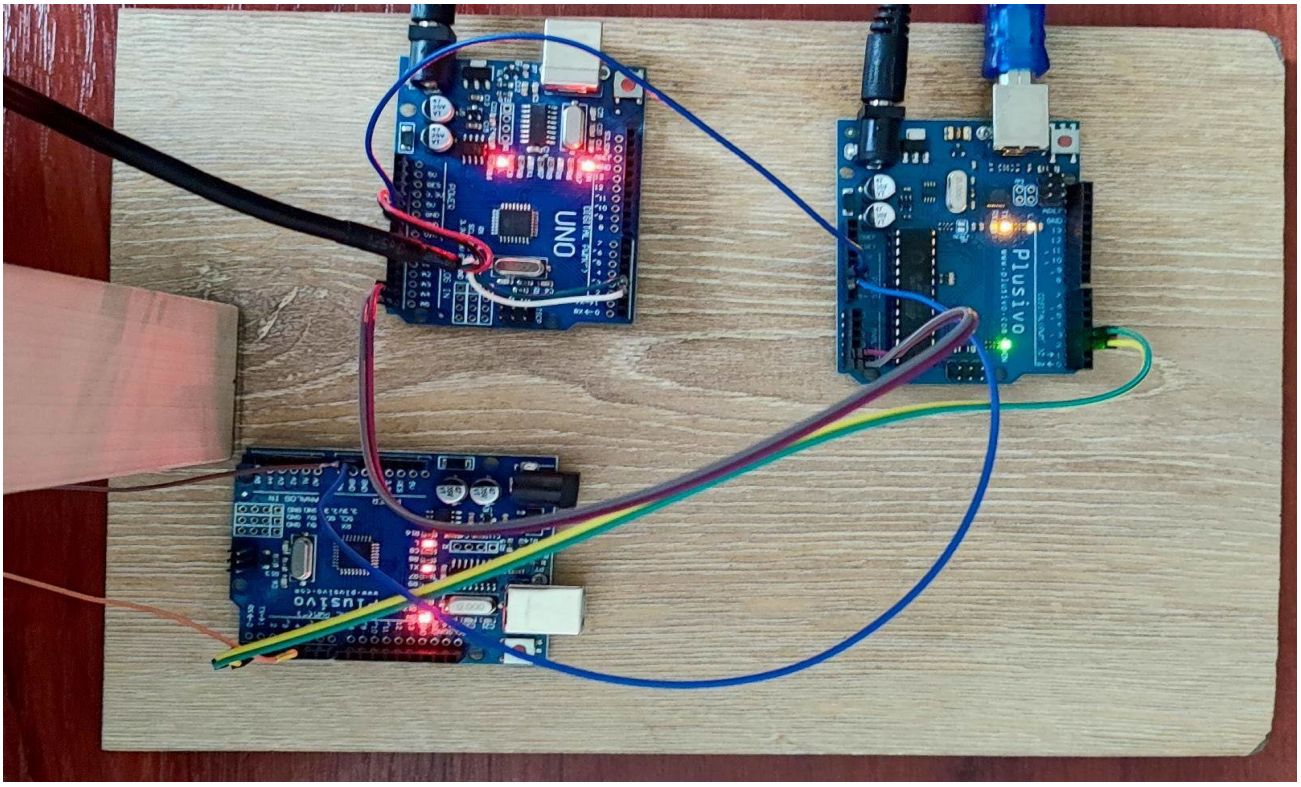
7.15. ábra. Motor áram és feszültség

8 Gyakorlati megvalósítás

A (8.1. ábra) ábrán látható az acrobot gyakorlati megvalósítása. A tartószerkezet alsó felében található a szabályozási algoritmust futtató arduino hálózat (8.2. ábra). A tartószerkezet felső felében található a forgóérzékelő, amelyhez rögzítve van maga az acrobot. A szegmensek alumínium rudakból állnak, amelyeknek a végén találhatóak az összekötő elemek. A második szegmens végén található a súly, a nagyobb tehetetlenség elérése érdekében, aminek a tömege 86 gramm. A szervomotorhoz szükséges vezetékek alól vannak elvezetve, mivel nagy mozgási tartományt fed le működés közben.



8.1. ábra. Az acrobot gyakorlati megvalósítása, jobb oldalt a szervomotor és a második csukló



8.2. ábra. Arduino fejlesztőlapokkal megvalósított szabályozás

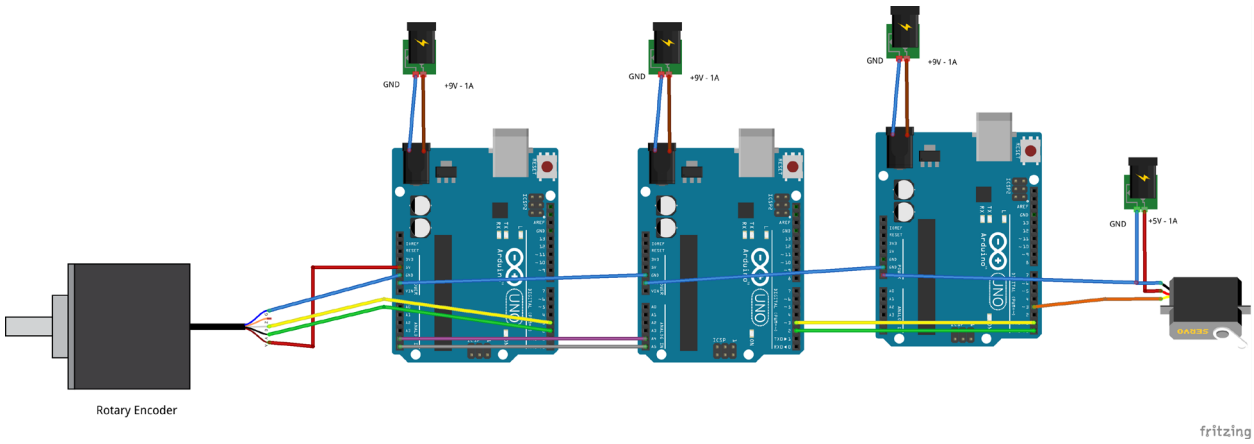
A *3.* táblázatban láthatóak a valós rendszer paraméterei. A szegmensek hosszánál nem csak az alumínium rudak hosszát vettem figyelembe, hanem magának a szegmenseknek a súly középpontját, ugyan úgy ahogy az elméleti acrobot mozgásegyenleteiben is szerepel. A második szegmens végén található egy súly, ami több inerciát ad a rendszernek, így elérve a könnyebb energiapumpálást a rendszerbe, és kompenzálja az első csuklóban található szervomotor tömegét.

Jelölés	Érték	Mértékegység
$\mathbf{l_1}$	0.4	m
$\mathbf{l_2}$	0.425	m
$\mathbf{lc_1}$	0.2885	m
$\mathbf{lc_2}$	0.34	m
$\mathbf{m_1}$	0.1012	kg
$\mathbf{m_2}$	0.1203	kg
$\mathbf{I_1}$	0.0015	$kg*m^2$
$\mathbf{I_2}$	0.0018	$kg*m^2$

3. táblázat. *A valós rendszer paraméterei*

8.1 Szabályozó felépítése

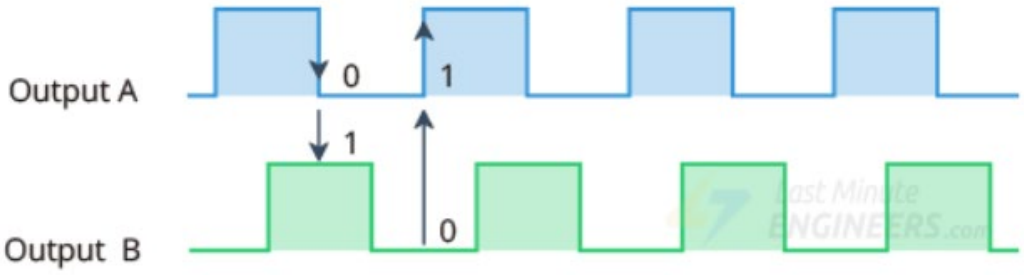
A 8.3. ábra ábrán látható a szabályozás tömbvázlata. Balról az első arduino feladata a forgóérzékelőtől érkező adatok fogadása és azok feldolgozása. Itt kiszámításra kerül a forgóérzékelő által érkező jelekből az első csukló relatív elfordulása a kezdőpozíciójához képest, és ennek a szögsebessége. A második arduino feladata a szabályzó jel kiszámítása a beérkező szög, szögsebesség és a neuronhálók segítségével. Itt a 24. fejezetben részletezett két neuronháló került leimplementálásra, a Matlab programban kiszámolt súlyzók alapján végzem el a mátrix szorzásokat. Mivel kis méretű neuronháló is megtudta tanulni, ezért valós időben gyorsan tud működni a szabályozás. A harmadik arduino feladata a szervomotor irányítása, feladata, hogy az előző egységtől kapott előírt szögre irányítsa a szervomotort. Itt a külön egység alkalmazására legfőképp a Timerek közötti konfliktusok miatt volt szükség.



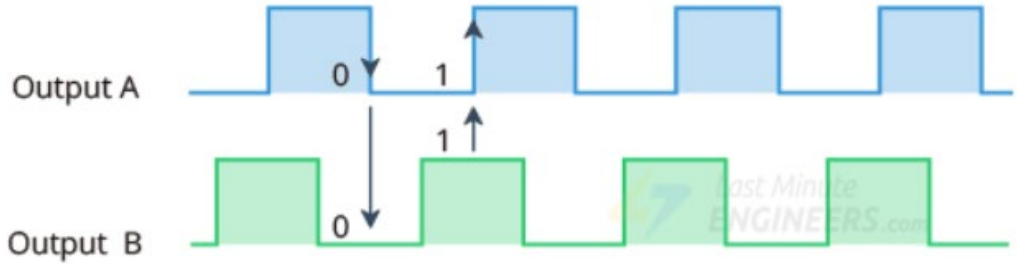
8.3. ábra. A gyakorlati szabályozás tömbvázlata

8.2 Állapotok mérése

A (8.4. ábra) és (8.5. ábra) ábrákon látható a forgóérzékelőtől érkező A és B jel logikai feldolgozása, amelyekből a forgás irányát is meglehet határozni. A forgóérzékelőben 2 tárcsa található, amelyek bizonyos helyeken lyuggatva vannak, így forgás közben egy bizonyos pozícióban vagy átengedik a fényt, vagy nem. Forgás közben ezt egy-egy érzékelő érzékeli, és ennek alapján rendel hozzá egy feszültségszintet, ami lehet 0 vagy 5 Volt. Forgás közben így egy négyszög jelet eredményez. A 2 tárcsa egymáshoz képest relatív el van tolva, ebből lehet meghatározni a forgás irányát.



8.4. ábra. A forgóérzékelő által küldött A és B jelforma, amikor a forgás az óramutató járásával megegyező irányban történik[22]



8.5. ábra. A forgóérzékelő által küldött A és B jelforma, amikor a forgás az óramutató járásával megegyező irányban történik[22]

A szög és szögsebesség mérése egy 600 impulzusos forgóérzékelővel történik. Egy teljes fordulat alatt 600 impulzust ad le, A és B fázisban, melyek 90°-kal vannak eltolva egymáshoz képest. A feldolgozás során a két jel helyzetéből kitudom számítani a forgás irányát. Az A fázis lefutó élére nézem meg a B értékét. (2.függelék) Ha az 1, akkor az óramutató irányába forog, ha 0 akkor azzal ellentétesen. Ezt a logikát alkalmazom a B jelre is, ott az A jel értékét figyelem, a B jel lefutó élére, ezzel megduplázva a felbontást. Mindegyik lefutó él egy megszakítást generál az arduinon, ezzel elérve, hogy mindegyik jelet figyelembe vegye. A processzor minden 20ms-onként kiszámolja a relatív szögelfordulást és a szögsebességet (3.függelék). Mivel ebben az esetben egy fordulat 1200 impulzust tartalmaz, ezért minden lefutó él 0.3°-t jelent. A sebesség mérése a (29) képlettel történik, ahol a mintavételezési idő[t]

20ms.[23] Az n jelöli a beérkezett pulzusok számát, összeadva az A és B vezetéken érkező jeleket. N a pulzusok száma egy teljes kör alatt (1200). Egy 600 impulzusos forgóérzékelőt használtam, és mivel mind a két beérkező négyszög jelet felhasználtam, így 1200 impulzus érkezik be egy fordulat alatt.

$$\omega = \frac{2\pi n}{Nt} \tag{29}$$

8.3 Vezérlőjel számítás

A számításokat külön arduino végzi. Itt került implementálásra a két szabályozásnak megfelelő egyszerű neuronháló és a köztük való váltást eldöntő logika. A neuronháló megvalósítása (1.függelék):

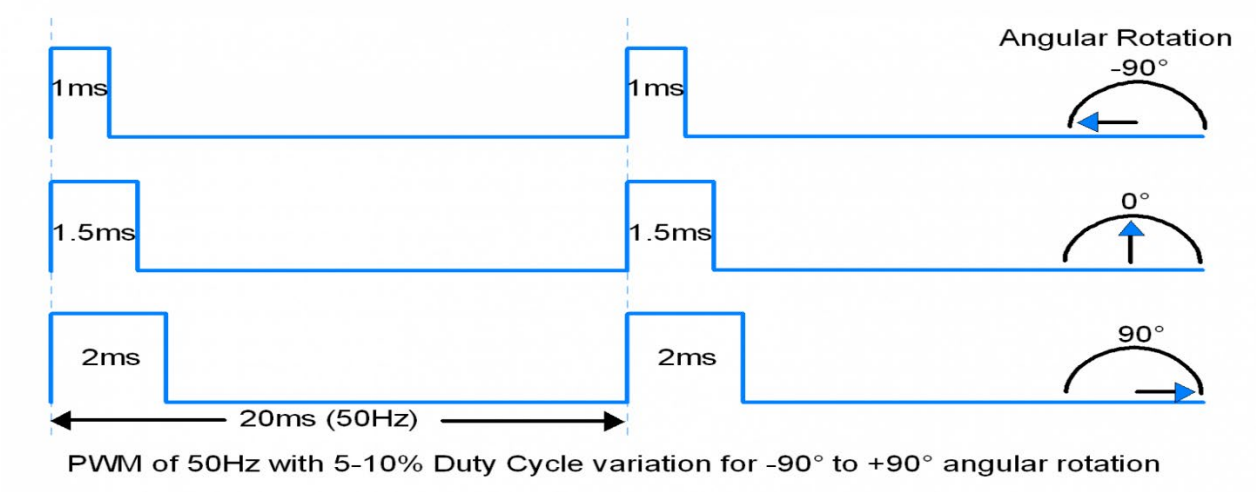
- Preprocess, adatok skálázása
- Inger kiszámítása
- Aktivációs függvény alkalmazása
- Rejtett réteg ingerének kiszámítása
- Purelin aktivációs függvény alkalmazása
- Process output, végeredmény skálázása

A Matlabban kiszámolt súlyzók értékeit használom. A számítások elkezdése előtt szükség van az adatok skálázására, ami egy maximum és minimum érték közé skálázza be a 2 bemenetet, vagyis a szögpozíciót és a szögsebességet. Ezután történik a súlyzókkal való szorzás, majd az aktivációs függvény (szigmoid) alkalmazása. Ezután kiszámításra kerül a rejtett réteg ingere. A következő aktivációs függvény után (purelin), ami egy lineáris aktivációs függvény, történik meg a végeredmény skálázása.

Mivel, a gyakorlati fázisban a 2. csukló állapotai nincsenek mérve, ezért a két szabályozás közötti váltásnál egy egyszerűbb feltételt alkalmazok. Ennek az alapja az, hogy akkor vált át a lineáris szabályzóra, ha az első csukló pozíciója a kezdőpozícióhoz képest 180° és a sebessége is minimális.

8.4 Szervo motor vezérlés

A szervomotor belsejében található egy belső szabályzó áramkör (PID szabályzó). Ennek segítségével képes pontos szögpozícióra beállni. Ennek az irányítása egy pwm jellel történik, amely 50 Hz frekvenciájú. A kitöltési tényező változtatásával lehet vezérelni a motort. A szervomotor 180°-t képes fordulni. Az 1ms-os kitöltési tényező a minimális -90°-k felel meg, 2ms a 90°-k. A megadott kitöltési tényező és a szervomotor szöge egyenesen arányos a két szélsőérték között, tehát ennek alapján bármilyen szöget be tudunk állítani 5 microsecundumos pontossággal. Az arduinon ezt a Servo.h könyvtár segítségével oldottam meg. Itt megadva a kívánt szöget, automatikusan kiszámítja a szögnek megfelelő kitöltési tényezőt, és beállítja az adott timereket. A Timer1-et az 50Hz-es jel létrehozásához, a Timer2-t a kitöltési tényező pontos megadásához használja a rendszer.



8.6. ábra. 50Hz-es PWM jel a szervomotor irányításához, feltüntetve a két szélső és a középső állapotához szükséges kitöltési tényezőt[24]

8.5 Kommunikáció

A szabályozás megvalósításához 3 külön egységet használtam. Ennek a főbb okai: a feladatok elkülönítése, jobb átláthatóság, a Timerek közötti konfliktusok, ütközések elkerülése, könnyebb fejleszthetőség a későbbiekben.

Az első és második arduino között i2C kommunikációs protokollt használok, a Wire.h könytárat használva.[25] Az első, forgóérzékelőtől érkező jeleket feldolgozó egység a Master. 20ms-ként küld el egy union típusú adatot, ami tartalmaz 2 float típusú számot, 8 byte-t küld el összesen.

A Slave-ben minden küldéskor aktiválódik egy megszakítás, ami elmenti a küldött tömb első, és második 4 byte-át külön változókba, amik jelentik a kiszámított aktuális szögelfordulást és szögsebességet.

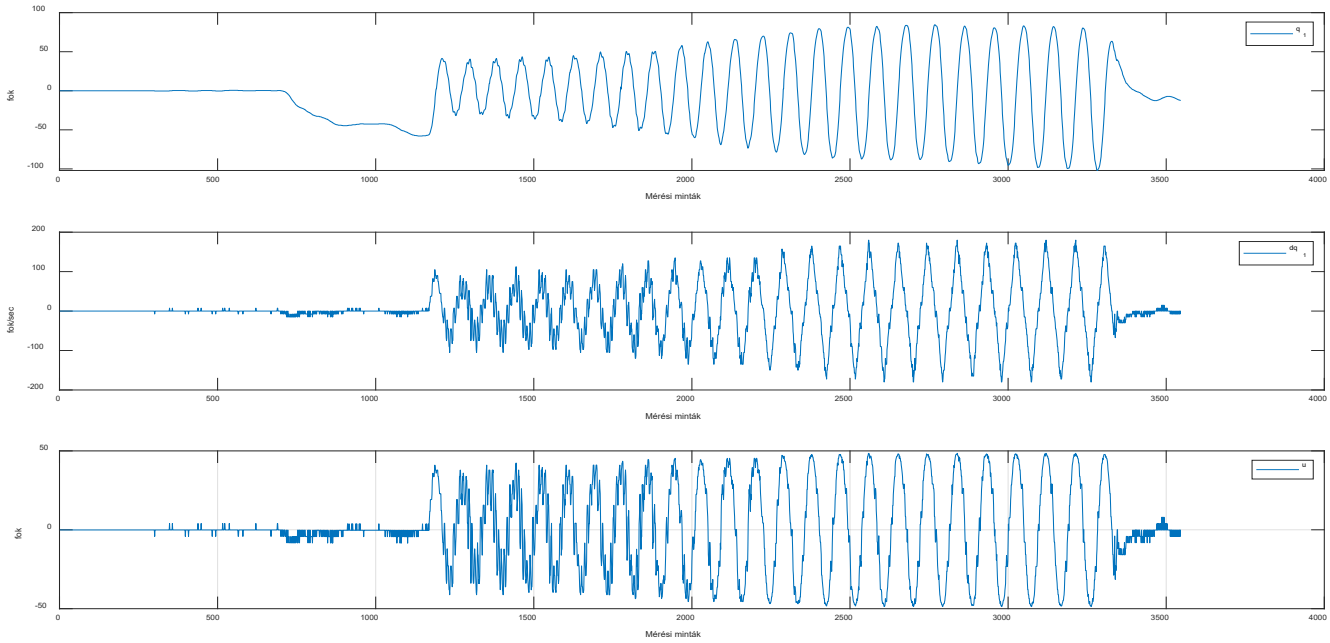
A második és harmadik egység között soros kommunikációt használok. Az arduinon ezeket az RX és TX pinek segítségével lehet megvalósítani. Az eszköz ezek segítségével kommunikál a számítógéppel is, ezért a mérések elvégzése érdekében a számításokat végző arduinon szoftveresen hoztam létre az RX és TX pineknek megfelelő 2,3-as digitális pineket, a SoftwareSerial könyvtárral.[26] A protokoll csomagokat állít össze, amely tartalmaz egy start bitet, az adatot, paritás bitet, stop bitet, majd bitekké darabolva küldi el.

Miután a fogadó fél megkapta a biteket, visszaállítva azokat az UART bufferben tárolja, onnan kiolvasva tárolja el egy változóban. A küldött adat a kiszámított előírt szög float típus, 4 bájton ábrázolva.

9 Gyakorlati eredmények

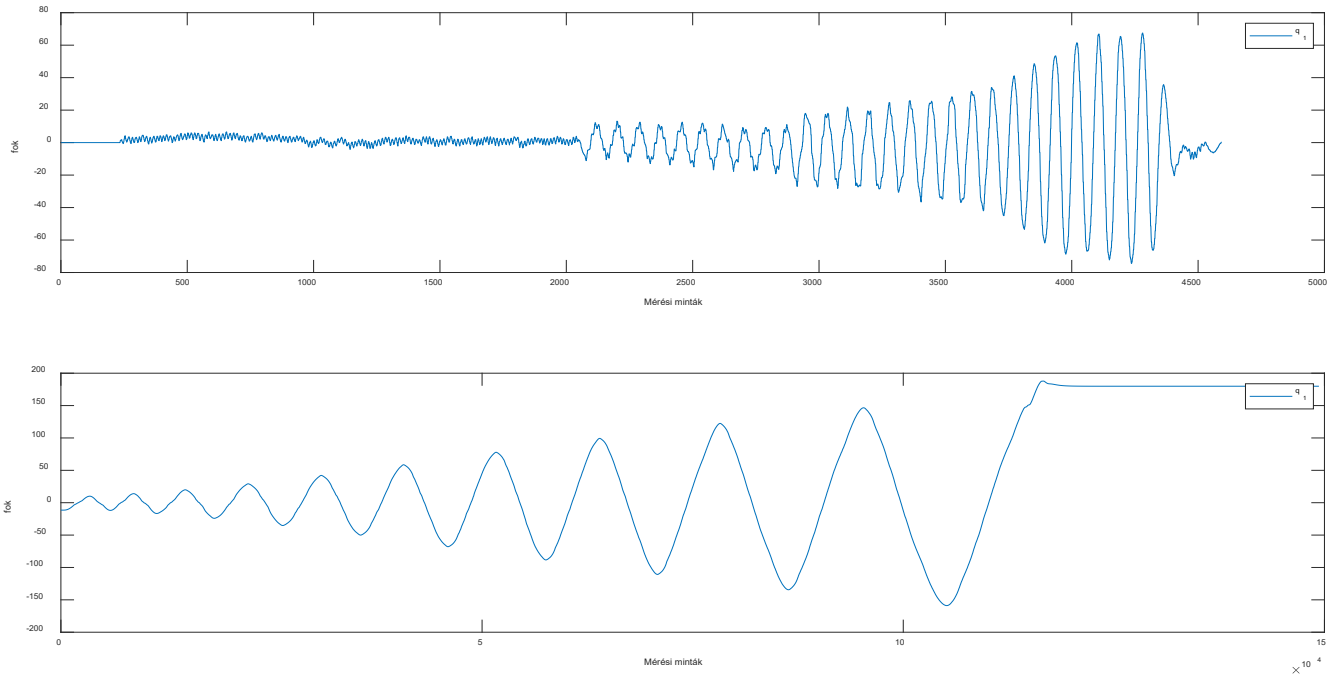
A számításokat végző arduinot a számítógéphez csatlakoztatva tudtam mérni az állapotokat, és a kiszámolt vezérlő jelet. A maximális szög, amit elér a feltornászási fázisban 100.8° (9.1. ábra). A mérést -57° kezdő pozícióból kezdtem, mivel a 0 körüli szögekre, a szervomotor nem tudja időben lekövetni az előírt pályáját, ezért beremeg a rendszer és időben csak később kezdődik el a swing-up fázis.

A rendszer nem tudja elérni az UEP pontját, viszont látható, hogy a lengések amplitúdója fokozatosan nő. Legfőképp a fizikai határai miatt nem tudja elérni az UEP pontját, a továbbiakban szükséges lesz a vezetékek meghosszabbítása, és a tartószerkezet stabilabbá tétele, rögzítése, hogy minimális legyen annak az energia elnyelése.



9.1. ábra. A megvalósított rendszer mérési eredményei szabályozás közben

A (9.2. ábra) ábrán látható összehasonlítva a leszimulált rendszer és a valós mérés. Megfigyelhető, hogy csak több idő után tud kilépni a 0° környékérő a valós rendszer, mivel berezeg.



9.2. ábra. Feltornáztatási fázis összehasonlítása.

Valós rendszer (felső). Szimulált, elméleti modell (alsó)

10 Összegzés, következtetések

A dolgozat célja egy alulaktuált rendszer, az acrobot szabályozása, és annak bemutatása. Látható, hogy e rendszerek szabályozása a valóságban sokkal nehezebb, mint például az acroboté is.

Összességében sikerült megépíteni egy szerkezetet, ami eleget tesz egy alulaktuált rendszer követelményeinek, és jól megközelíti az elméleti acrobot sajátosságait. Elkészült ennek a részletes 3D-s modellje, amin sikerült megvalósítani és bemutatni a ‘swing up and balance’ szabályozást.

A programban lehetőség van a paraméterek változtatására is, tehát felhasználható a későbbi szimulációk során is, amennyiben változtatások történnek a valós rendszeren.

Jövőbeni cél a szerkezet stabilabbá tétele, tovább minimalizálva a másfajta mozgásokból eredő energiaveszteséget. Másik cél egy a 7.3. fejezetben részletezett DC motor beépítése és a szerkezet azzal való szabályozása. Ehhez a továbbiakban szükséges egy nyomatékszabályozást megoldani, ami által mindig a szabályzás által kiszámolt megfelelő nyomaték fog megjelenni a motor tengelyén.

Továbbfejlesztési lehetőségek a valós acroboton még a megfelelő motor kiválasztásában, beszerelésében és annak a szabályozásának a megoldásában vannak.

Ismerve a valós rendszer paramétereit lehetővé vált másfajta szabályozások tanulmányozása, úgy a gyakorlatban, mint ideális környezetben a Matlabon belül. A Matlabban és Simulink környezetekben elkészített program által le lehet modellezni a valóságban meglévő acrobot viselkedését különböző szabályzó jelek hatására. A program és a Simulink modellek is elkülönítve, strukturálva vannak, tehát alkalmas arra, hogy akár a későbbiekben is másfajta szabályozásokat is be lehessen építeni.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani konzulens tanáromnak, dr. Dávid Lászlónak, a mentorálásért és iránymutatásokért a dolgozat elkészítése közben.

11 Irodalomjegyzék

[1] Bodor Bálint, “Alulaktuált robotok szabályozásának eltérő megközelítései TDK dolgozat,” 2015.

[2] Mark W. Spong, “The Swing Up Control Problem For The Acrobot,” 1995.

[3] M. W. Spong, “Swing Up Control of the Acrobot *.”

[4] R. Tedrake, “Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines Course Notes for MIT 6.832,” 2009.

[5] G. Boone, “Efficient Reinforcement Learning: Model-based Acrobot Control,” 1997. [Online]. Available: www.cc.gatech.edu/-gboone

[6] X. Xin and T. Yamasaki, “Energy-based swing-up control for a remotely driven acrobot: Theoretical and experimental results,” *IEEE Transactions on Control Systems Technology*, vol. 20, no. 4, pp. 1048–1056, 2012, doi: 10.1109/TCST.2011.2159220.

[7] R Nave, “Moment of Inertia: Rod.” http://hyperphysics.phy-astr.gsu.edu/hbase/mi2.html#irod (accessed Apr. 17, 2023).

[8] Russ Tedrake, “Multi-Body Dynamics.” http://underactuated.mit.edu/multibody.html#double_pendulum (accessed Apr. 17, 2023).

[9] Márton Lőrinc, “Robotokarok Dinamikus Modellezes Palyatervezes PDG PID.”

[10] Matthew Peter Kelly, “MatthewPeterKelly / OptimTraj.” https://github.com/MatthewPeterKelly/OptimTraj (accessed Jun. 16, 2023).

[11] M. W. Spong, “Underactuated Mechanical Systems.”

[12] X. Xin and M. Kaneda, “A Robust Control Approach to the Swing up Control Problem for the Acrobot,” 2001.

[13] D. N. Kutasi and L. Márton, *Rendszerelmélet laboratóriumi gyakorlatok*. Scientia, 2010.

[14] György Katalin, “Diszkrét LQR állapot utáni szabályzó laborgyakorlat.”

[15] X. Xin and M. Kaneda, “Analysis of the energy-based swing-up control of the Acrobot,” *International Journal of Robust and Nonlinear Control*, vol. 17, no. 16, pp. 1503–1524, Nov. 2007, doi: 10.1002/rnc.1184.

[16] “Matlab feedforwardnet.” https://www.mathworks.com/help/deeplearning/ref/feedforwardnet.html (accessed Apr. 17, 2023).

[17] “Matlab Simscape.” https://www.mathworks.com/products/simscape.html (accessed Apr. 17, 2023).

[18] “Matlab ode23”, Accessed: Apr. 17, 2023. [Online]. Available: https://www.mathworks.com/help/matlab/ref/ode23t.html

[19] “MG996R servo.” https://datasheetspdf.com/pdf/942981/ETC/MG996R/1 (accessed Jun. 19, 2023).

[20] "Optical Rotary Encoder." https://grabcad.com/library/optical-rotary-encoder-1/details?folder_id=13312851 (accessed Apr. 17, 2023).

[21] "Motor torque." https://www.motioncontroltips.com/faq-difference-between-torque-back-emf-motor-constant/ (accessed Jun. 13, 2023).

[22] "How Rotary Encoder Works and Interface It with Arduino." https://lastminuteengineers.com/rotary-encoder-arduino-tutorial/ (accessed Apr. 17, 2023).

[23] "How are encoders used for speed measurement?" https://www.motioncontroltips.com/how-are-encoders-used-for-speed-measurement/ (accessed Apr. 17, 2023).

[24] "Servo Motor." https://www.electronicwings.com/sensors-modules/servo-motor (accessed Apr. 17, 2023).

[25] "How to Setup I2C Communication on the Arduino." https://www.circuitbasics.com/how-to-set-up-i2c-communication-for-arduino/ (accessed Apr. 17, 2023).

[26] "How to Set Up UART Communication on the Arduino." https://www.circuitbasics.com/how-to-set-up-uart-communication-for-arduino/ (accessed Apr. 17, 2023).

12 Függelék

1) Neuronháló implementálása (Arduino/Compute.io)

```
// implementation of the pretrained neural network
// Swing up
float MLP_swing_up(float q1, float dq1)
{
  // preprocess q1
  xq1 = ((q1 - sw_I_xmin) * sw_I_gain) + sw_I_ymin;
  // Input layer
  v = (sw_Iw1 * xq1 + sw_Iw2 * dq1) + sw_b1;
  u = 2 / (1 + exp(-2 * v)) - 1;

  // Hidden layer
  v = (sw_Lw * u) + sw_b2;
  // purelin activation function

  // Process output
  uv = v + 1; // O_min
  uv = uv * sw_O_gain;
  uv = uv + sw_O_xmin;

  return uv;

}
```

2) Enkóder pozíció számolás (Arduino/High_speed_encoder.io)

```
void ai0() {
  // ai0 is activated if DigitalPin nr 2 is going from HIGH to LOW
  // Check pin 3 to determine the direction

  vel_cnt++;
  if (PIND & B00001000)
  {
    counter--;
  }
  else
  {
    counter++;
  }
}
```

3) Enkóder sebesség számolás

```
ISR(TIMER1_COMPA_vect)
{
  vel_cnt = counter - vel_cnt_old;

  ENC_angle = 0.3 * counter; //  0.3 = 360/1200 - degree

  ENC_vel = (2 * 3.14 * vel_cnt) / (1200 * 0.02);  // | 0.02 -> 50Hz

  ENC_vel = ENC_vel * 57.29; // rad/sec to degree/sec

  vel_cnt_old = counter;

  // reset the angle
  if (ENC_angle > 360)
    counter = 0;
  if (ENC_angle < -360)
    counter = 0;

  // Send the value of counter
  converter.q1[0] = ENC_angle;
  converter.q1[1] = ENC_vel;

}
```

4) Matlab szimuláció fő program (Acrobot_Pololu_Simulation/Main.m)

```
%% Betölti a valós rendszer paramétereit
run('real_parameters.m');

%% Linearizálas a munkapont körül
% q1 q2 dq1 dq2
[A,B] = linearizalas(m1,m2,l1,l2,lc1,lc2,J1,J2);

%% LQR számítás
K = LQR_function(A,B);

%% Szimuláció Simulinkben
open_system('Real_Acrobot_Simulation.slx')
Sim_time = 40;
out = sim('Real_Acrobot_Simulation',Sim_time);

% set view convention -> Y up (XY)
% standard views -> Front view

%% Szimulációs eredmények elmentése
run('getParameters.m');

%% Ábrák megjelenítése
run('Visualization_real.m');
```

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ

SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Vizat decan

Conf. dr. ing. Domokos József

Vizat director departament

Ș.l. dr. ing Szabó László Zsolt