Primjena Hopfieldovih neuronskih mreža u rješavanju optimizacijskih problema

Kosanović, Boris

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:815353

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Boris Kosanović

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Prof. dr. sc. Josip Kasać

Student:

Boris Kosanović

Zagreb, 2017.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru dr. sc. Josipu Kasaću na korisnim savjetima i pružanoj podršci tijekom pisanja ovog rada

Boris Kosanović

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **BORIS KOSANOVIĆ** Mat. br.: 0035190473

Naslov rada na hrvatskom jeziku: **PRIMJENA HOPFIELDOVIH NEURONSKIH MREŽA U RJEŠAVANJU OPTIMIZACIJSKIH PROBLEMA**

Naslov rada na engleskom jeziku: **APPLICATION OF HOPFIELD NEURAL NETWORKS TO SOLUTION OF OPTIMIZATION PROBLEMS**

Opis zadatka:

Hopfieldove neuronske mreže predstavljaju klasu dinamičkih neuronskih mreža sa širokim poljem primjena u rješavanju raznih optimizacijskih problema poput linearnog i kvadratičnog programiranja, cjelobrojnog programiranja, problema trgovačkog putnika i drugih kombinatornih optimizacijskih problema. Također, mnogi algebarski problemi poput invertiranja matrica i rješavanja linearnih matričnih jednadžbi mogu se formulirati kao optimizacijski problemi i riješiti primjenom Hopfieldove neuronske mreže.

U zadatku je potrebno:

- Implementirati Hopfieldovu neuronsku mrežu za rješavanje problema invertiranja matrica i rješavanja Lyapunovljeve matrične jednadžbe.
- Implementirati Hopfieldovu neuronsku mrežu za rješavanje problema kvadratičnog programiranja s linearnim ograničenjima.
- Primijeniti Hopfieldovu neuronsku mrežu za rješavanja Lyapunovljeve matrične jednadžbe na sintezu linearnog regulatora.
- Primijeniti Hopfieldovu neuronsku mrežu za kvadratično programiranje na problem optimalnog upravljanja linearnim sustavima.

Zadatak zadan:
30. studenog 2016.

Zadatak zadao:

Kasać Josip
Prof. dr. sc. Josip Kasać

Rok predaje rada:
1. rok: 24. veljače 2017.
2. rok (izvanredni): 28. lipnja 2017.
3. rok: 22. rujna 2017.

Predviđeni datumi obrane:
1. rok: 27.2. - 03.03. 2017.
2. rok (izvanredni): 30. 06. 2017.
3. rok: 25.9. - 29. 09. 2017.

v.d. predsjednika Povjerenstva:

Izv. prof. dr. sc. Branko Bauer

SADRŽAJ

SADRŽAJ I
POPIS SLIKA III
POPIS TABLICA IV
POPIS OZNAKA V
SAŽETAK VII
SUMMARY VIII
1. UVOD 1
2. NEURONSKE MREŽE 3
2.1. Biološki neuron 3
2.2. Umjetni neuron 4
2.3. Sličnosti i razlike mozga i računala 6
3. HOPFIELDOVE NEURONSKE MREŽE 8
3.1. Hopfield-ov umjetni neuron 8
3.2. Funkcija energije 10
3.3. Gradijentna metoda 11
4. LINEARNI REGULACIJSKI SUSTAV 13
4.1. Linearni multivarijabilni sustavi 13
4.2. Transformacija varijabli stanja linearnih sustava 14
4.3. Lyapunovljeva analiza stabilnosti 15
4.4. Sinteza regulatora metodom podešavanja polova 15
5. ANALIZA I SINTEZA LINEARNIH SUSTAVA PRIMJENOM HOPFIELDOVIH NEURONSKIH MREŽA 17
5.1. Rješavanje problema sustava linearnih jednadžbi 19
5.1.1. Algoritam učenja mreže za rješavanje sustava linearnih jednadžbi 19
5.1.2. Primjer sustava linearnih jednadžbi 20
5.2. Rješavanje problema inverza matrice 22
5.2.1. Algoritam učenja mreže za invertiranje matrice 22
5.2.2. Primjer invertiranja matrice 23
5.3. Rješavanje Lyapunovljeve matrične jednadžbe 25
5.3.1. Algoritam učenja mreže za rješavanje Lyapunovljeve matrične jednadžbe 25
5.3.2. Primjer rješavanja Lyapunovljeve jednadžbe pomoću neuronskih mreža 27
5.4. Rješavanje matrične jednadžbe metode podešavanja polova 29
5.4.1. Algoritam učenja mreže za problem rješavanja sinteze regulatora metodom podešavanja polova 29
5.4.2. Primjer rješavanja sinteze regulatora metodom podešavanja polova pomoću neuronskih mreža 30
5.4.2.1. Primjer 1 31
5.4.2.2. Primjer 2 34
5.5. Rješavanje problema kvadratičnog programiranja s linearnim ograničenjima 37

5.5.1. Algoritam učenja mreže za rješavanje problema kvadratičnog programiranja s linearnim ograničenjima 37
5.5.2. Primjeri rješavanja problema kvadratičnog programiranja s linearnim ograničenjima pomoću neuronskih mreža 38
5.5.2.1. Rješavanje problema linearnog programiranja s linearnim ograničenjima . 38
5.5.2.2. Rješavanje problema kvadratičnog programiranja s linearnim ograničenjima 40

6. REGULACIJA INVERZNOG NJIHALA PRIMJENOM HOPFIELDOVIH NEURONSKIH MREŽA 42
6.1. Dinamički model inverznog njihala 42
6.2. Linearizacija nelinearnog modela inverznog njihala 43
6.3. Sinteza linearnog regulatora 44
6.4. Simulacijski rezultati inverznog njihala 44
6.4.1. Inverzno njihalo bez upravljanja 45
6.4.2. Inverzno njihalo s linearnim regulatorom 46
6.4.3. Inverzno njihalo sa „swing-up“ regulatorom 50

7. ZAKLJUČAK 54

LITERATURA 55

PRILOZI 56

POPIS SLIKA

Slika 1. Pojednostavljeni prikaz umjetne neuronske mreže [6] 3
Slika 2. Pojednostavljena struktura biološkog neurona [7] 4
Slika 3. Struktura umjetnog neurona 5
Slika 4. Hopfieldov model osnovnog dinamičkog neurona [2] 8
Slika 5. Funkcionalna struktura Hopfieldovog neurona [2] 10
Slika 6. Pogreška mreže kod rješavanja sustava linearnih jednadžbi 21
Slika 7. Konvergencija rješenja sustava linearnih jednadžbi 21
Slika 8. Pogreška mreže kod invertiranja matrice 24
Slika 9. Konvergencija članova invertirane matrice 24
Slika 10. Pogreška rješenja 27
Slika 11. Razlika između egzaktnog i iterativnog rješenja 28
Slika 12. Konvergencija rješenja Lyapunovljeve jednadžbe 28
Slika 13. Pogreška matrice P 32
Slika 14. Konvergencija članova matrice P 32
Slika 15. Pogreška inverza matrice P 33
Slika 16. Pogreška inverza matrice P (uvećano) 33
Slika 17. Pogreška matrice P 35
Slika 18. Konvergencija članova matrice P 35
Slika 19. Pogreška inverza matrice P 36
Slika 20. Pogreška inverza matrice P (uvećano) 36
Slika 21. Konvergencija rješenja 39
Slika 22. Konvergencija rješenja 41
Slika 23. Sustav kolica i njihala [8] 42
Slika 24. Provjera nelinearnog modela 45
Slika 25. Odziv nelinearnog i lineariziranog modela za $\theta(0)=0,1$ 46
Slika 26. Upravljačke varijable lineariziranog i nelinearnog modela za $\theta(0)=0,6$ 46
Slika 27. Odziv nelinearnog i lineariziranog modela za $\theta(0)=0,6$ 47
Slika 28. Upravljačke varijable lineariziranog i nelinearnog modela za $\theta(0)=0,6$ 47
Slika 29. Odziv nelinearnog i lineariziranog modela za $\theta(0)=0,96$ 48
Slika 30. Upravljačke varijable lineariziranog i nelinearnog modela za $\theta(0)=0,96$ 48
Slika 31. Odziv nelinearnog i lineariziranog modela za $\theta(0)=0,98$ 49
Slika 32. Upravljačke varijable lineariziranog i nelinearnog modela za $\theta(0)=0,98$ 49
Slika 33. Upravljanje nelinearnog modela pomoću 1. verzije „swing-up“ regulatora 50
Slika 34. Upravljačka varijabla 1. verzije „swing-up“ regulatora 51
Slika 35. Upravljanje nelinearnog modela pomoću 2. verzije „swing-up“ regulatora 51
Slika 36. Upravljačka varijabla 2. verzije „swing-up“ regulatora 52
Slika 37. Upravljanje nelinearnog modela pomoću „swing-up“ regulatora zasnovanog na energiji 52
Slika 38. Upravljačka varijabla „swing-up“ regulatora zasnovanog na energiji 53

POPIS TABLICA

Tablica 1. Razlike između digitalnog računala i ljudskog mozga [4] 6
Tablica 2. Usporedba karakteristika paradigmi [4] 7

POPIS OZNAKA

Oznaka	Jedinica	Opis
$\mathbf{A}$	-	Matrica koeficijenata sustava
$\mathbf{B}$	-	Matrica ulaza sustava
$\mathbf{C}$	-	Matrica izlaza sustava
C_j	F	Kapacitet j-tog neurona
c	-	Koeficijent viskoznog trenja njihala
$\mathbf{D}$	-	Matrica prijenosa sustava
E	-	Funkcija energije
$\mathbf{e}$	-	Pogreška sustava
F	N	Sila
G_{ji}	1/Ω	Provodljivost i-tog ulaza j-tog neurona
$\mathbf{g}_{je}$	-	Vektor odstupanja od ograničenja zadanih linearnom jednadžbom
$\mathbf{g}_{ne}$	-	Vektor odstupanja od ograničenja zadanih linearnom jednadžbom
g	m/s^2	Ubrzanje sile teže
I	m^4	Moment inercije
$\mathbf{K}$	-	Matrica pojačanja
$\mathbf{k}$	-	Kaznena funkcija
k	-	Koeficijent viskoznog trenja kolica
L	m	Duljina njihala
M	kg	Masa kolica
m	kg	Masa njihala
R_{ji}	Ω	Otpor i-tog ulaza j-tog neurona
t	s	Vrijeme
$\mathbf{u}$	-	Vektor upravljanja
u_j	V	Stanje j-tog neurona
v_j	V	Izlaz j-tog neurona
$\mathbf{W}$	-	Matrica sinaptičkih težina
w_{ji}	-	Sinaptičke težine i-tog ulaza j-tog neurona
$\mathbf{x}$	-	Vektor stanja
$\mathbf{x}_0$	-	Vektor početnih uvjeta
x	m	Položaj
$\dot{x}$	m/s	Brzina
$\ddot{x}$	m/s^2	Ubrzanje
x_j	-	Izlaz j-tog neurona
$\mathbf{y}$	-	Vektor stanja
α_j	-	Koeficijent prigušenja
$\mathbf{\Lambda}$	-	Matrica željenih svojstvenih vrijednosti

θ	rad	Kut zakreta
$\dot{\theta}$	rad/s	Kutna brzina
$\ddot{\theta}$	rad/s^2	Kutno ubrzanje
κ	-	Kazneni faktor
$\boldsymbol{\mu}$	-	Matica koeficijenata učenja
μ_{ji}	-	Koeficijent učenja i-tog ulaza j-tog neurona
τ_j	s	Integracijska vremenska konstanta

SAŽETAK

Hopfieldove neuronske mreže predstavljaju klasu dinamičkih neuronskih mreža sa širokim poljem primjena u rješavanju raznih optimizacijskih problema. Također mnogi algebarski problemi, mogu se formulirati kao optimizacijski problemi i riješiti primjenom Hopfieldove neuronske mreže. U ovom radu riješeni su problemi kvadratičnog programiranja, invertiranja matrica i rješavanja linearnih matričnih jednadžbi, te su primijenjeni kod sinteze linearnog regulatora inverznog njihala. Prilikom učenja Hopfieldove neuronske mreže korišten je gradijentni algoritam. Upravljački sustav inverznog njihala testiran je nizom simulacija s različitim kutevima otklona od inverznog položaja. Rezultati simulacija pokazuju da Hopfieldova neuronska mreža daje zadovoljavajuća rješenja u području gdje je sustav moguće upravljati linearnim regulatorom. Za implementaciju Hopfieldovih neuronskih mreža i usporedbu dobivenih rezultata s egzaktnim rješenjima korišten je programski paket Matlab.

Ključne riječi: Hopfieldova neuronska mreža, optimizacija, gradijentni algoritam

SUMMARY

Hopfield neural networks are a class of dynamic neural networks with a wide field of application in solving various optimization problems. Also many algebraic problems can be formulated as an optimization problem and solved using the Hopfield neural network. This paper deals with problems of quadratic programming, inverting a matrix and solving linear matrix equations, which are then applied to the synthesis of linear regulators inverted pendulum. Gradient algorithm is used during learning phase of Hopfield neural network. The control system of inverted pendulum is tested by a series of simulations with different angles of deflection of the inverted position. Simulation results show that Hopfield neural network gives satisfactory solutions in the area where the system can be operated with linear regulator. Matlab is used for the implementation of Hopfield neural networks and comparison of the results with the exact solutions.

Key words: Hopfield neural network, optimization, gradient algorithm

1. UVOD

Neuronske mreže ne predstavljaju novi koncept. Početak neuronskih mreža se veže za 1943. godinu kada su Warren McCulloch i Walter Pitts predstavili jednostavan model umjetnog neurona. Takav model neurona i danas se koristi kao osnovni blok za izgradnju umjetnih neuronskih mreža. Tek sredinom 80-ih kada je predstavljen „Backpropagation" algoritam, raste znanstveni i komercijalni interes za neuronske mreže.

Neuronske mreže imaju drugačiji pristup rješavanju problema od konvencionalnog rješavanja problema računalom. Konvencionalni pristup rješavanju problema je putem algoritama gdje računalo izvršava komplet instrukcija te tako dolazi do rješenja problema. Ako nam svaki od koraka u rješavanju problema nije poznat računalo neće doći do rješenja zadanog problema. Ta činjenica nas ograničava na mogućnost rješavanja problema koje razumijemo i znamo riješiti. Rješavanje problema koje ne razumijemo ili ne znamo kako ih opisati pomoću algoritama (prepoznavanje lica, prepoznavanje rukopisa) nije moguće riješiti konvencionalnim metodama nego se rješavaju neuronskim mrežama.

U sklopu ovog rada korištena je Hopfieldova neuronska mreža za rješavanje raznih jednostavnih problema koji su vodili rješavanju problema inverznog njihala.

Problem inverznog njihala je zadan u prostoru stanja te se pomoću Hopfieldove neuronske mreže dobiva pojačanje regulatora koji se koristi prilikom regulacije te se promatra ponašanje inverznog njihala za razne kuteve i regulatore.

Pregled po poglavljima:

U drugom poglavlju je predstavljena umjetna neuronska mreža. Prvo je opisan biološki neuron kako bi ga mogli usporediti s umjetnim neuronom koji je osnovni gradivni element umjetne neuronske mreže. Na kraju su iznesene sličnosti i razlike između računala i mozga.

U trećem poglavlju su opisani umjetni neuroni za Hopfieldovu neuronsku mrežu. Nakon toga je opisana funkcija energije i njena važnost kod Hopfieldovih m neuronskih mreža koje rade na principu minimizacije funkcije energije. Na kraju je prikazan gradijentni algoritam koji se koristi kod učenja mreže.

U četvrtom poglavlju su dani alati za sintezu regulatora. Prvo je prikazana transformacija varijabli stanja linearnih sustava kako bi se pojednostavile dinamičke jednadžbe. Zatim je opisana metoda Lyapunovljeve analize stabilnosti i na kraju je opisana sinteza regulatora metodom podešavanja polova.

U petom poglavlju je opisana metoda učenja Hopfieldove neuronske mreže općenito. Nakon toga su izvedeni algoritmi za učenje mreže kako bi rješavale iduće probleme:

- sustava linearnih jednadžbi
- inverz matrice
- Lyapunovljeva matrična jednadžba
- matrična jednadžba metode podešavanja polova
- linearno programiranje s linearnim ograničenjima
- kvadratično programiranje s linearnim ograničenjima

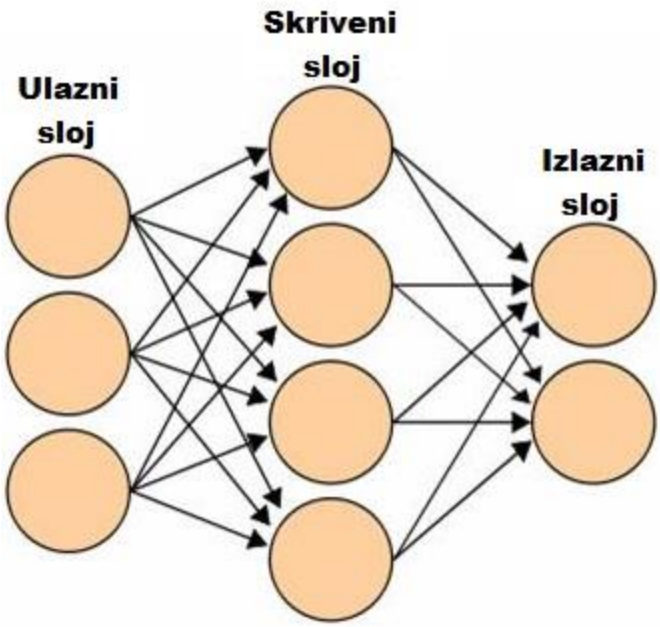
te je prikazan primjer učenja mreže za svaki od tih problema.

U zadnjem poglavlju su opisani matematički modeli nelinearnog i linearizirano inverznog njihala. Nakon toga je prikazana sinteza regulatora za primjer inverznog njihala. Na kraju su prikazani rezultati simulacije inverznog njihala s različitim upravljanjima i ulaznim parametrima:

- bez upravljanja
- linearni regulator s različitim kutevima otklona
- veliki kut otklona sa „swing-up“ regulatorima
- veliki kut otklona sa „swing-up“ regulatorom koji se zasniva na energiji

2. NEURONSKE MREŽE

Umjetna neuronska mreža je zbir umjetnih neurona koji su međusobno povezani i interaktivni kroz operacije obrade signala. Mreža može imati jedan ili više ulaza, jedan ili više izlaza, a između se nalaze jedan ili više skrivenih slojeva. Pojedinačni neuroni su, kao i slojevi, međusobno spojeni vezama kroz koje idu signali.



Slika 1. Pojednostavljeni prikaz umjetne neuronske mreže [6]

Neuronske mreže obrađuju informacije na sličan način kao ljudski mozak. Mreža je sastavljena od mnogo međusobno vezanih elemenata (neurona) koji rade paralelno kako bi riješili specifičan problem. Mreža uči iz primjera. Kako bi mreža ispravno radila primjeri za učenje moraju biti pažljivo odabrani. Problem je što ne postoji način za provjeru ispravnosti naučene mreže ako ne dođe do pogreške.

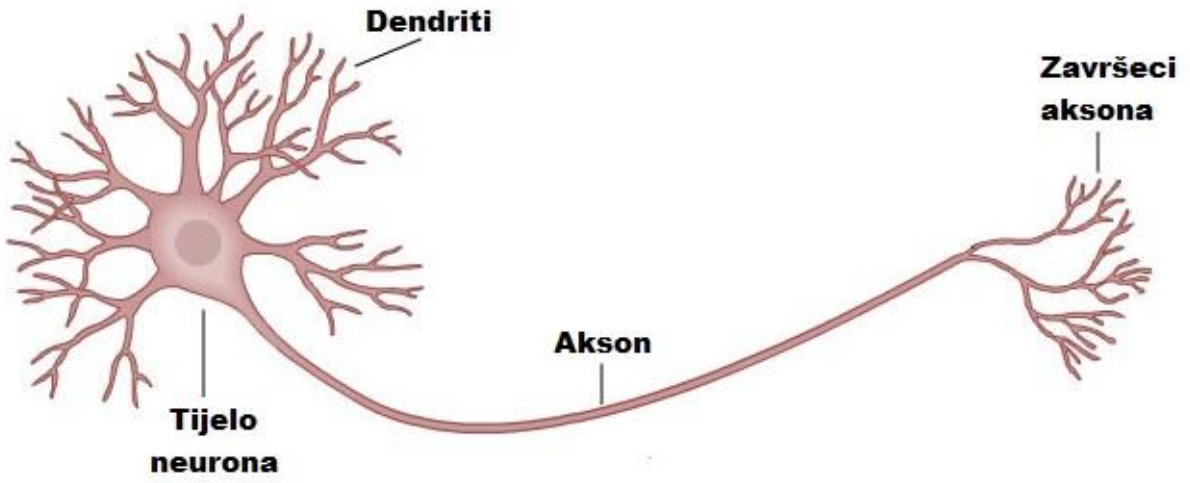
2.1. Biološki neuron

Kako bi mogli uspoređivati sličnosti između umjetnog neurona i biološkog neurona prvo se moramo upoznati sa strukturom i funkcijama biološkog neurona. Osnovni dijelovi neurona su tijelo neurona, akson i mnoštvo dendrita koji okružuju tijelo neurona.

Tijelo neurona je okrugli središnji dio stanice neurona u kojem se obavljaju gotovo sve logičke funkcije neurona. U tijelu neurona se nalazi sve potrebne genetske i metaboličke funkcije koje održavaju neuron na životu.

Glavna funkcija aksona je prenositi živčane impulse s tijela jednog neuron na druge neurone ili izvršne organe. Akson je živčano vlakno koji je s jedne strane vezan za tijelo neurona, a se druge strane se grana. Na početku aksona se nalazi aksonski brežuljak gdje se signali pretvaraju u sekvencu živčanih impulsa. Ti impulsi putuju aksonom do drugog kraja aksona koji je više ili manje razgranat. Krajevi ovih grana završavaju malim zadebljanjima koja su najčešće u dodiruju s dendritima, a rjeđe s tijelom drugog neurona.

Dendriti su jako razgranata struktura živčanih vlakana koji povezuju neuron s okolnim neuronima. Ta živčana vlakna su nepravilnih oblika i povezana su na tijelo neurona. Svaki neuron ima između 10^3 i 10^4 dendrita. Dendriti prihvaćaju signale od okolnih neurona preko sinapsi.



Slika 2. Pojednostavljena struktura biološkog neurona [7]

Sinapsa je mali razmak koji se nalazi između završetka prethodnog neurona i dendrita ili tijela drugog neurona. Svaki neuron putem aksona formira mnogo sinaptičke veze s okolnim neuronima. Impulsi neurona putuju kroz akson do sinapsi gdje se signali različitog intenziteta šalju kroz dendrite ili direktno na tijelo drugih neurona. Svi signali koje neuron šalje mogu biti smirujući ili uzbudni. Neuron šalje impuls kroz svoj akson ako je uzbuda prešla prag osjetljivosti neurona odnosno ako je iznos uzbude veći od smirujućeg utjecaja za neki kritični iznos.

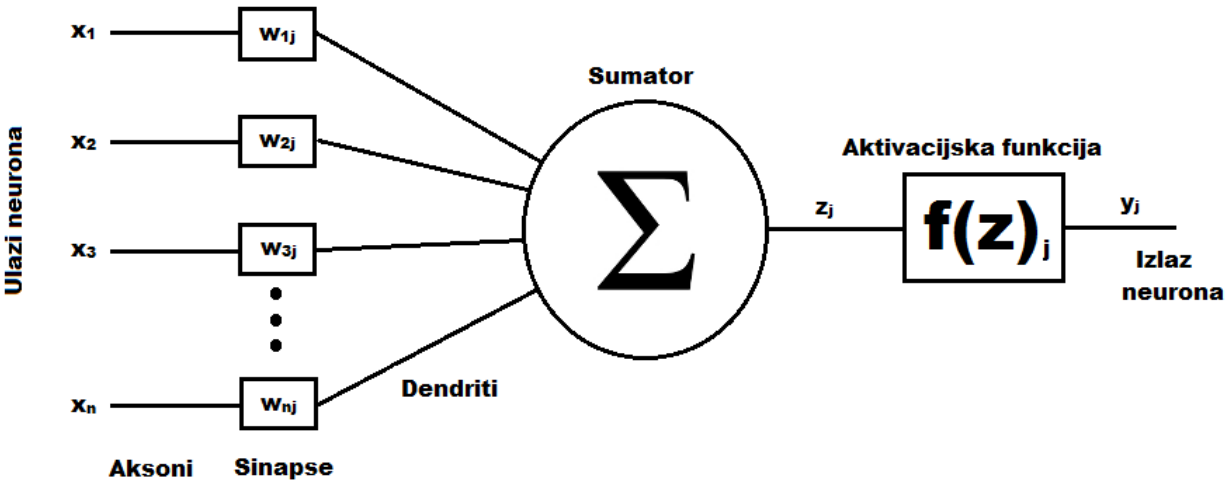
2.2. Umjetni neuron

Umjetni neuroni su gradivni elementi svake umjetne neuronske mreže, oni pokušavaju simulirati strukturu i funkcije bioloških neurona. Međutim, modeli umjetnih neurona nisu ograničeni biološkim neuronima te su samo djelomično zasnovani na biološkom neuronu. [2]

Neki od razloga su:

- Ponašanje živčanog sustava je složeno te ga ne razumijemo u potpunosti.
- Samo dio ponašanja stvarnog neurona je neophodno za mogućnost obrade podataka, a drugi dio ponašanja stvara beznačajne nuspojave.
- Tehnički gledano, implementacija potpuno funkcionalnog stvarnog neurona je vjerojatno nemoguća i neefikasna.
- Umjetne neuronske mreže konstruirane su prema problemu, njihova arhitektura i značajke ovise o problemu koji rješavaju.

Tijelo biološkog neurona zamijenjeno je sumatorom, ulogu dendrita preuzimaju ulazi u sumator. Prag osjetljivosti kod umjetnog neurona je prikazan aktivacijskom funkcijom. Sinaptičke veze biološkog neurona su zamijenjene težinskim faktorima preko kojih umjetni neuron ostvaruje veze sa svojom okolinom.



Slika 3. Struktura umjetnog neurona

Težinski faktor može biti pozitivan ili negativan broj, a kod suvremenih umjetnih neuronskih mreža i neka funkcija (promjenjivi težinski faktor). Kada je težinski faktor jednak nuli onda odgovarajuća veza s okolinom neurona ne postoji, te se ne ucrtava u shemi neuronske mreže. Težinski faktori imaju funkciju sinapsi kod umjetnog neurona, oni povezuju izlaze drugih neurona (aksone) s ulazima sumatora (dendritima). Izlaz iz sumatora se povezuje na ulaz aktivacijske funkcije koja na izlazu daje izlaz neurona. Aktivacijske funkcije se dijele na linearne i nelinearne. Kod linearnih, izlaz sumatora se množi s nekim faktorom i tako se dobiva izlaz neurona. Nelinearne aktivacijske funkcije mogu poprimiti razne oblike. [1]

Najčešći oblici nelinearnih aktivacijskih funkcija:

- Funkcije praga osjetljivosti
- Sigmoidalne funkcije
- Hiperbolične funkcije
- Harmoničke funkcije

2.3. Sličnosti i razlike mozga i računala

Što se tiče sličnosti između računala i mozga, nisu mnogobrojne ali postoje. Mozak i računalo koriste električne signale za rad, imaju veliki broj jednostavnih elemenata i izvode funkcije koje se općenito mogu nazvati računarskim funkcijama.

Tablica 1. Razlike između digitalnog računala i ljudskog mozga [4]

atribut	mozak	računalo
Tip elemenata za procesiranje	neuron	bistabil
Brzina prijenosa	2 ms ciklus	ns ciklus
Broj procesora	Oko 10^{11}	10 ili manje
Broj veza među procesorima	10^3-10^4	10 ili manje
Način rada	serijski, paralelno	serijski
Signali	analogni	digitalni
Informacije	ispravne i neispravne	ispravne
Pogreške	nefatalne	fatalne
Redundancija	stotine novih stanica	eventualno rezervni sustav

Kako se može vidjeti u tablici 1, računalo u usporedbi s mozgom ima puno veću brzinu prijenosa. Međutim brzina „računanja“ mozga je neusporedivo veća od one klasičnog računala, zahvaljujući ogromnom broju paralelnih jedinica za računanje. Trenutna arhitektura računala je pogodna za odvijanje procesa u serijskim sekvencama (von Neumannovo računalo), gdje se sljedeća sekvenca izvrši kad prethodna završi. Ta arhitektura značajno usporava proces računanja. Provode se mnoga istraživanja na području arhitekture računala koja bi omogućila učinkovitu implementaciju neuronske mreže. Sljedeća bitna razlika je stupanj pogreške. Računalo ne pravi pogrešku sve dok su ulazi, softver i hardver ispravni. Mozak uglavnom daje

bolje zaključke i aproksimacije i za nepotpune ulazne podatke. To ponekad može biti loše, ali ta činjenica nam omogućava da steknemo nova znanja.

Ipak, možemo koristiti konvencionalna računala pri implementaciji neuronske mreže, odbacujući pri tome rješavanje problema putem algoritama. Tada se nalazimo u hibridnom području gdje koristimo sekvencijski stroj koji imitira neuronsku mrežu kao visoko-paralelnu arhitekturu. Takav je sustav fizički von Neumannovo računalo, ali se na razini obrade podataka odriče simboličke paradigme u korist paradigme neuronskih mreža. Bitne razlike između paradigmi dane su u tablici 2. [4]

Tablica 2. Usporedba karakteristika paradigmi [4]

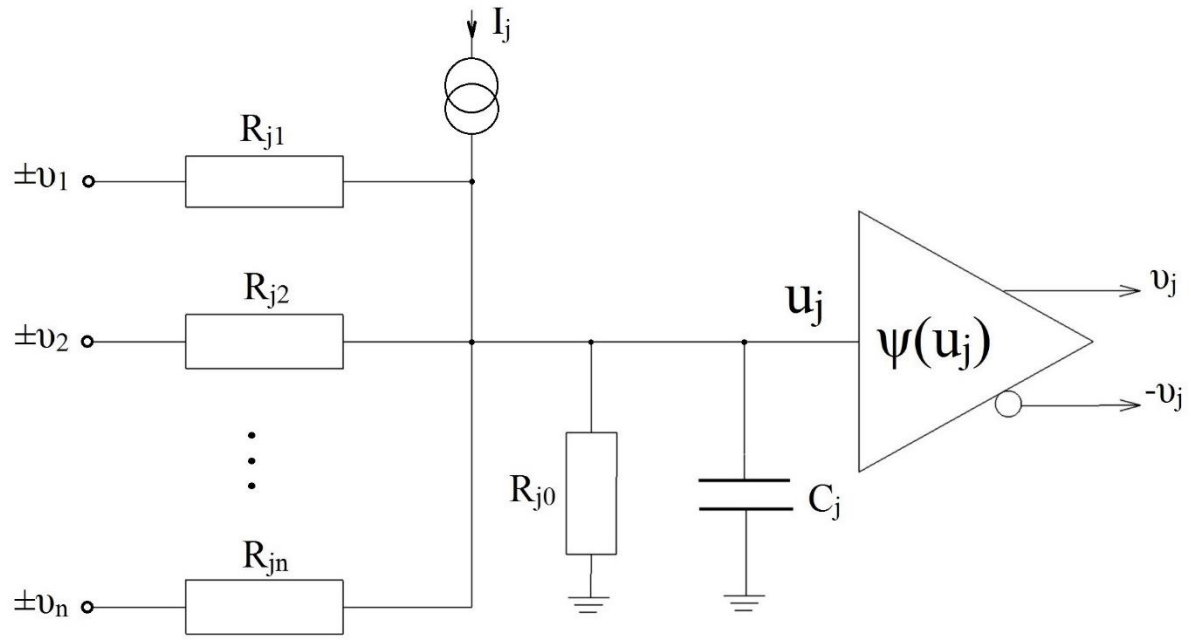
von Neumann	neuronska mreža
Računalu se unaprijed detaljno mora opisati algoritam u točnom slijedu koraka	Neuronska mreža uči samostalno ili s učiteljem
Podaci moraju biti precizni – nejasni ili neizraziti podaci ne obrađuju se adekvatno	Podatci ne moraju biti precizni (gotovo uvijek su neprecizni)
Arhitektura je osjetljiva – kod uništenja nekoliko memorijskih ćelija računalo ne funkcionira	Obrada i rezultat ne mora puno ovisiti o pojedinačnom elementu mreže
Postoji eksplicitna veza između semantičkih objekata (varijabli, brojeva, zapisa u bazi...) i sklopovlja računala preko pokazivača na memoriju	Pohranjeno znanje je implicitno, ali ga je teško interpretirati

3. HOPFIELDOVE NEURONSKE MREŽE

Hopfieldove neuronske mreže postale su popularne 1982. godine. Hopfieldove neuronske mreže su jedne od najčešće korištenih mreža za rješavanje problema optimizacije ili problema matematičkog programiranja. Velika prednost Hopfieldove neuronske mreže pred ostalim mrežama je mogućnost implementacije na elektroničkim krugovima za „on-line“ rješavanje paralelno distribuiranih procesa.

3.1. Hopfield-ov umjetni neuron

Hopfield-ov model umjetnog neurona je vjerojatno najpopularniji model dinamičkog umjetnog neurona



Slika 4. Hopfieldov model osnovnog dinamičkog neurona [2]

Na slici 4. je prikazan neuron realiziran pomoću analogne tehnike. Neuron se sastoji od kondenzatora C_j, otpornika R_{ji} i nelinearnog pojačala sa sigmoidalnom prijenosnom funkcijom. Pojačalo ima dva izlaza koji daju simetrične signale. Sinaptičke veze se ostvaruju preko otpornika R_{ji}. Kod pozitivne sinaptičke veze otpornik R_{ji} se spaja na pozitivan izlaz pojačala ,a ako je potrebna negativna sinaptička veza otpornik se spaja na negativan izlaz. Struja I_j predstavlja bias.

Primjenom Ohmovog zakona i Kirchhoffovog zakona za struju dobivamo sljedeće diferencijalne jednadžbe koje opisuju neuron:

$$C_j \frac{\mathrm{d}u_j}{\mathrm{d}t} = -\frac{u_j}{R_j} + \sum_{i=1}^{n} \frac{v_i}{R_{ji}} + I_j \tag{3.1}$$

gdje je

$$v_j = \psi(u_j) \qquad (j = 1,2,3, \ldots, n)$$

i ψ je sigmoidalna aktivacijska funkcija,

$$\frac{1}{R_j} = \frac{1}{R_{j0}} + \sum_{i=1}^{n} \frac{1}{R_{ji}} = G_{j0} + \sum_{i=1}^{n} G_{ji} \tag{3.2}$$

gdje G_{ji} predstavlja provodljivost (i=0,1,2,...,n). Ove jednadžbe mogu se zapisati u općenitom obliku.

$$\tau_j \frac{\mathrm{d}u_j}{\mathrm{d}t} = -\propto_j u_j + \left(\sum_{i=1}^{n} w_{ji} x_i + \theta_j \right) \tag{3.3}$$

$$x_j = \psi(u_j) \tag{3.4}$$

gdje

$x_i = v_i$ (i = 1,2,...,n) su ulazni signali (naponi, potencijali)

u_j je unutarnji signal kojeg zovemo unutrašnji potencijal

$\tau_j = r_j C_j$ je integracijska vremenska konstanta

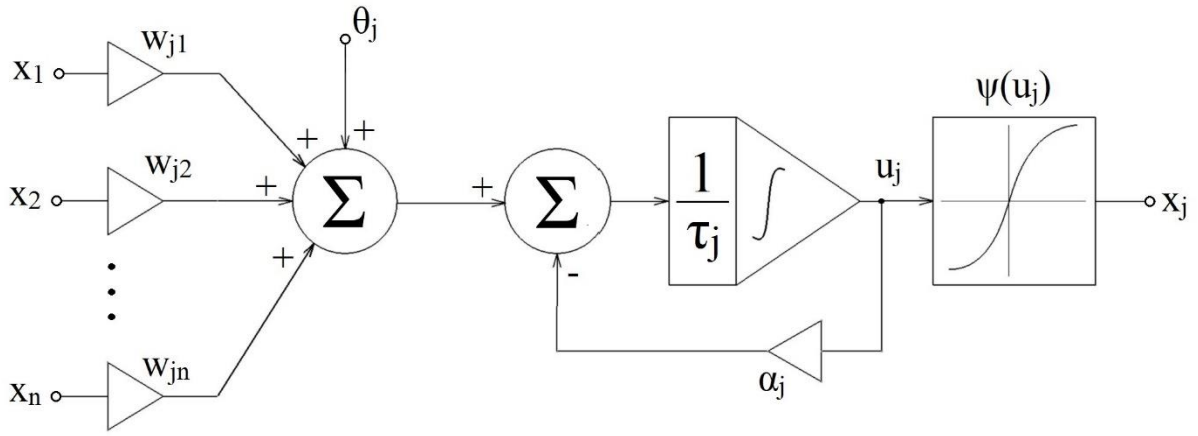
r_j jedinični otpor

$\alpha_j = r_j / R_j$ je koeficijent prigušenja, faktor koji unutarnji signal u_j postavlja u nulu ako su ulazi u nuli

$w_{ji} = \pm\, r_j / R_{ji} = \pm\, r_j / G_{ji}$ sinaptičke težine (imaju pozitivan predznak ako je R_{ji} spojen na $+x_i$ i negativan ako je R_{ji} spojen na $-x_i$)

$\theta_j = r_j I_j$ linearni pomak signala

Ovaj model neurona je dobar ako neuronsku mrežu implementiramo u analognoj tehnici. Kako bi neuronsku mrežu implementirali na računalu ovu elektroničku shemu moramo pretvoriti u funkcionalnu strukturu.



Slika 5. Funkcionalna struktura Hopfieldovog neurona [2]

Funkcionalni blok dijagram, koji je prikazan na slici 5 i odgovara jednadžbama (3.3) i (3.4) sastoji se od sumatora sinaptičkih težina (w_{ij}), prigušenog integratora i nelinearnog pojačala sa sigmoidalnom aktivacijskom funkcijom. Pojačalo korišteno u Hopfieldovom modelu neurona (sa simetričnim izlazima $\pm x_j$) je opisano monotonom, diferencijabilnom funkcijom

$$\psi(u_j) = \frac{1}{1 + e^{\gamma u_j}} \quad \text{ili} \quad \psi(u_j) = \tanh(\gamma u_j) \tag{3.5}$$

gdje je $\gamma > 0$ određuje nagib ili stupanj porasta koji je često promjenjiv te se može mijenjati tijekom računalnog procesa. Pretpostavlja se da je dinamika pojačala zanemariva. Dinamika neuron se određuje pomoću kondenzatora C_j i otpornika R_{ji}. Dinamika se mijenja podešavanjem tih parametara. Sinaptičke težine se određuju pomoću ulazne provodljivosti $G_{ji} = 1/R_{ji}$ povezane na izlaze ($+v_j$ ili $-v_j$) j-tog pojačala. [2]

3.2. Funkcija energije

Ruski matematičar A.M. Lyapunov utemeljio je moćnu teoriju stabilnosti i tehniku za njeno testiranje. Jedan od najčešćih načina proučavanja konvergencije dinamičkih sustava opisanih sustavom diferencijalnih jednadžbi je pronaći Lyapunovljevu funkciju, također zvanu i funkcija energije pošto je Lyapunovljeva funkcija zasnovana na konceptu energije i odnosu pohranjene energije na stabilnost sustava.

Funkcija energije $E = (\mathbf{x},\mathbf{W},\boldsymbol{\theta})$ je definirana u prostoru stanja, nerastuća je duž trajektorija te je ograničena s donje strane. Funkcija energije će se zvati Lyapunovljeva funkcija ako je padajuća na svim nekonstantnim trajektorijama. Lyapunovljev teorem kaže: ako za dani sustav diferencijalnih jednadžbi postoji funkcija energije E, zvana Lyapunovljeva funkcija koja zadovoljava idući izraz:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \sum_{j=1}^{n} \frac{\partial E}{\partial x_j} \frac{\mathrm{d}x_j}{\mathrm{d}t} + \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial E}{\partial w_{ij}} \frac{\mathrm{d}w_{ij}}{\mathrm{d}t} \leq 0 \tag{3.6}$$

sa

$$\frac{\mathrm{d}E}{\mathrm{d}t} = 0 \quad \text{samo za} \quad \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = 0 \quad \text{i} \quad \frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t} = 0$$

tada je sustav stabilan, to jest $x_j(t)$ i $w_{ij}(t)$ konvergiraju kada $t \to \infty$.

Kako je funkcija energije ograničena monotono padajuća funkcija vremena koja konvergira do konstante i njena derivacija po vremenu konvergira u 0. Derivacija energije po vremenu dE/dt je strogo manja od 0 osim u ravnoteži gdje nestaje. [2]

3.3. Gradijentna metoda

Jedna od najvažnijih metoda za optimizaciju bez ograničenja se zasniva na tako zvanoj gradijentnoj metodi spusta koja se odnosi na gradijentnu metodu. Sve gradijentne metode spusta za optimizaciju bez ograničenja su bazirane na standardnim metodama koje su znane kao metoda najbržeg spusta i Newtonova metoda. Te metode pretvaraju problem minimizacije u sustav diferencijalnih jednadžbi prvog reda

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\sum_{i=1}^{n} \mu_{ji} \frac{\partial E}{\partial x_i} \tag{3.7}$$

s početnim uvjetom $x_j(0) = x_j^{(0)}$, koji se može zapisati u kompaktnom matričnom obliku

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = -\boldsymbol{\mu}(\mathbf{x}, t) \nabla_x E(\mathbf{x}), \tag{3.8}$$

gdje je

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \left[\frac{\mathrm{d}x_1}{\mathrm{d}t}, \frac{\mathrm{d}x_2}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}x_n}{\mathrm{d}t}\right]^{\mathrm{T}}, \tag{3.9}$$

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}} \tag{3.10}$$

i $\boldsymbol{\mu}(\mathbf{x},t)$ je simetrična pozitivno definitna matrica koja se često naziva i matrica učenja kod koje su članovi ovisni o vremenu i vektoru $\mathbf{x}(t)$. Kako bi pronašli vektor $\mathbf{x}^{\cdot}$ koji minimizira funkciju $E(\mathbf{x})$ moramo riješiti ili simulirati sustav običnih diferencijalnih jednadžbi s početnim uvjetima. To znači da se minimum funkcije energije možemo odrediti prateći krivulju rješenja gradijentnog sustava sa

$$\mathbf{x}^{\cdot} = \lim_{t \to \infty} \mathbf{x}(t) \tag{3.11}$$

Međutim, treba se primijetiti da nas zanima samo točka ravnoteže, ne cijela trajektorija $\mathbf{x}(t)$. Kako bi pokazali da je sustav diferencijalnih jednadžbi stabilan moramo odrediti derivaciju funkcije energije

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \sum_{j=1}^{n} \frac{\partial E}{\partial x_j} \frac{\mathrm{d}x_j}{\mathrm{d}t} = -[\nabla_x E(\mathbf{x})]^{\mathrm{T}} \boldsymbol{\mu}(\mathbf{x}, t) \nabla_x E(\mathbf{x}) \leq 0 \tag{3.12}$$

pod uvjetom da je matrica $\boldsymbol{\mu}$ simetrična i pozitivno definitna. Jednadžba (3.12) garantira da se vrijednost funkcije energije $E(\mathbf{x})$ smanjuje u vremenu i konvergira u stabilni lokalni minimum kako vrijeme teži u beskonačnost. Brzina konvergencije u minimum određena je matricom $\boldsymbol{\mu}$. U najjednostavnijoj metodi poznatoj kao metoda najbržeg spusta matrica $\boldsymbol{\mu}(\mathbf{x},t)$ je reducirana na jediničnu matricu pomnoženu s pozitivnom konstantom μ_0, u tom slučaju sustav je pojednostavljen

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\mu_0 \frac{\partial E(\mathbf{x})}{\partial x_j}, \qquad x_j(0) = x_j^{(0)} \quad (j = 1,2, \ldots, n) \tag{3.13}$$

gdje je pozitivni koeficijent μ_0 parametar učenja. Može se primijetiti da su vektori $\mathrm{d}\mathbf{x}/\mathrm{d}t$ i $\nabla_x E(\mathbf{x})$ suprotni vektori. Procjena vektora $\mathbf{x}(t)$ u vremenu će rezultirati minimizacijom funkcije energije $E(\mathbf{x})$ kako prolazi vrijeme. Trajektorija $\mathbf{x}(t)$ je kreće duž putanje koja ima najveću stopu pada. Vremenski diskretna verzija metode najbržeg spusta se može zapisati

$$x_j[(k+1)\tau] = x_j(k\tau) - \eta^{(k)} \frac{\partial E(\mathbf{x})}{dx_j} \Bigg|_{x_j = x_j^{(k)}} \tag{3.14}$$

također se može zapisati i u kompaktnijem vektorskom obliku

$$\mathbf{x}^{(\mathrm{k}+1)} = \mathbf{x}^{(\mathrm{k})} - \eta^{(\mathrm{k})} \nabla_k E(\mathbf{x}^{(\mathrm{k})}) \tag{3.15}$$

gdje je $0 \leq \eta^{(k)} \leq \eta_{max}$. Parametar $\eta^{(k)}$ se odnosi na koeficijent učenja. Kod vremenski diskretnog algoritma najbržeg spusta kontrolni parametar $\eta^{(k)}$ je pozitivan te ne smije prelaziti neku maksimalnu vrijednost $\eta^{(k)} < \eta_{max}$ kako bi se osigurala stabilnost algoritma. Ta S druge strane, za algoritam u kontinuiranom vremenu parametar učenja mora biti pozitivan i proizvoljno velik bez utjecaja na stabilnost sustava. To je jedna od važnijih prednosti algoritma u kontinuiranom vremenu. [2]

4. LINEARNI REGULACIJSKI SUSTAV

4.1. Linearni multivarijabilni sustavi

Linearni vremenski-varijabilni kontinuirani dinamički sustavi mogu se prikazati sljedećim jednadžbama stanja:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t), \qquad \mathbf{x}(t_0) = \mathbf{x}_{0,} \tag{4.1}$$

$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t), \tag{4.2}$$

gdje je

$\mathbf{x}(t) \in \mathbb{R}^n$ – vektor stanja,

$\mathbf{u}(t) \in \mathbb{R}^n$ – vektor upravljanja,

$\mathbf{y}(t) \in \mathbb{R}^n$ – vektor stanja,

Nadalje,

$\mathbf{A}(t) \in \mathbb{R}^{n\,x\,n}$ je vremenski promjenjiva matrica koeficijenta sustava,

$\mathbf{B}(t) \in \mathbb{R}^{n\,x\,m}$ je vremenski promjenjiva matrica ulaza sustava,

$\mathbf{C}(t) \in \mathbb{R}^{p\,x\,n}$ je vremenski promjenjiva matrica izlaza sustava, a

$\mathbf{D}(t) \in \mathbb{R}^{p\,x\,m}$ je vremenski promjenjiva matrica prijenosa sustava.

Jednadžba (4.1) naziva se jednadžba stanja, dok se jednadžba (4.2) naziva jednadžba izlaza. Obe jednadžbe zajedno nazivaju se dinamičkim jednadžbama stanja i izlaza.

Kada su matrice **A**, **B**, **C** i **D** vremenski neovisne, tada imamo vremenski-invarijantni linearni kontinuirani sustav, prikazana sljedećim jednadžbama stanja

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(t_0) = \mathbf{x}_{0,} \tag{4.3}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \tag{4.4}$$

U slučaju kada je $\mathbf{u}(t) = 0$, govorimo o autonomnom linearnom sustavu. [3]

4.2. Transformacija varijabli stanja linearnih sustava

Izbor varijabli stanja dinamičkih sustava nije jednoznačan. Direktni izbor fizikalnih varijabli stanja(pozicija, brzina, struja, napon,...) ne mora biti najpodesniji izbor sa stanovišta analize dinamičkih sustava. Odgovarajućom transformacijom varijabli stanja moguće je bitno pojednostaviti dinamičke jednadžbe sustava i time bitno olakšati samu analizu i sintezu.

Linearna transformacija varijabli stanja ima sljedeći oblik

$$\mathbf{x}(t) = \mathbf{P}\mathbf{z}(t), \tag{4.5}$$

gdje je $\mathbf{P}$ konstantna matrica transformacije, dok je $\mathbf{z}(t)$ novi, tzv. kanonski vektor stanja. Matrica transformacije $\mathbf{P}$ mora biti nesingularna da bi bila moguća obrnuta transformacija $\mathbf{z}(t)=\mathbf{P}^{-1}\mathbf{x}(t)$.

S obzirom na to da vrijedi

$$\dot{\mathbf{x}}(t) = \mathbf{P}\dot{\mathbf{z}}(t), \tag{4.6}$$

uvrštavanjem transformacije (4.5) u jednadžbe stanja (4.3) i (4.4), dobivamo

$$\dot{\mathbf{z}}(t) = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}\mathbf{z}(t) + \mathbf{P}^{-1}\mathbf{B}\mathbf{u}(t), \qquad \mathbf{z}(t_0) = \mathbf{P}^{-1}\mathbf{x}_{0,} \tag{4.7}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{P}\mathbf{z}(t) + \mathbf{D}\mathbf{u}(t). \tag{4.8}$$

Uvedemo li sljedeću notaciju

$$\hat{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}, \quad \hat{\mathbf{B}} = \mathbf{P}^{-1}\mathbf{B}, \quad \hat{\mathbf{C}} = \mathbf{C}\mathbf{P}, \quad \hat{\mathbf{D}} = \mathbf{D} \tag{4.9}$$

Jednadžbe stanja (4.7) i (4.8) postaju

$$\dot{\mathbf{z}}(t) = \hat{\mathbf{A}}\mathbf{z}(t) + \hat{\mathbf{B}}\mathbf{u}(t), \qquad \mathbf{z}(t_0) = \mathbf{z}_{0,} \tag{4.10}$$

$$\mathbf{y}(t) = \hat{\mathbf{C}}\mathbf{z}(t) + \hat{\mathbf{D}}\mathbf{u}(t). \tag{4.11}$$

Varijabla stanja $\mathbf{z}(t)$ je kanonska varijabla ako transformirana matrica $\hat{\mathbf{A}}$ ima dijagonalni oblik

$$\hat{\mathbf{A}} = \mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\} = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix}, \tag{4.12}$$

gdje su $\lambda_1,\ldots,\lambda_n$ međusobno različite svojstvene vrijednosti matrice $\mathbf{A}$.

Modalna matrica predstavlja matricu transformacije varijabli stanja koja sustav nekanonske forme prevodi u kanonsku (dijagonalnu) formu (4.12).

Modalnu matricu dobivamo rješenjem matrične jednadžbe

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{\Lambda}, \tag{4.13}$$

po matrici transformacije $\mathbf{P}$, gdje je $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \lambda_2,\ldots,\lambda_n\}$. Pomnožimo li prethodnu jednadžbu s matricom $\mathbf{P}$ s desne strane dobivamo $\mathbf{A}\mathbf{P} = \mathbf{P}\mathbf{\Lambda}$. [3]

4.3. Lyapunovljeva analiza stabilnosti

Razmotrimo stabilnost linearnog vremenski-invarijantnog linearnog sustava

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}, \tag{4.14}$$

primjenom Lyapunovljeve metode. Razmotrimo kvadratičnu Lyapunovljevu funkciju

$$\mathrm{V} = \mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x}, \tag{4.15}$$

gdje je **P** simetrična pozitivno definitna matrica. Da bi sustav bio stabilan, derivacija Lyapunovljeve funkcije mora biti negativno definitna, odnosno

$$\dot{\mathrm{V}} = -\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}, \tag{4.16}$$

gdje je **Q** neka simetrična pozitivno-definitna matrica.
Deriviranjem Lyapunovljeve funkcije po vremenu dobivamo

$$\dot{\mathrm{V}} = \dot{\mathbf{x}}^{\mathrm{T}}\mathbf{P}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{P}\dot{\mathbf{x}} = -\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}. \tag{4.17}$$

Uvrstimo li (4.13) u (4.16), dobivamo

$$\dot{\mathrm{V}} = \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\left(\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A}\right)\mathbf{x} = -\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}, \tag{4.18}$$

iz čega proizlazi

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q}. \tag{4.19}$$

Matrična jednadžba (4.19) naziva se Lyapunovljeva matrična jednadžba.

Stabilnost linearnih sustava određujemo primjenom Lyapunovljeve matrične jednadžbe (4.19) na sljedeći način: prvo se izabere neka pozitivno definitna matrica **Q** zatim se riješi Lyapunovljeva jednadžba (4.19) po matrici **P** i na kraju se provjeri da li je matrica **P** pozitivno definitna (primjenom Sylvesterovog teorema ili određivanjem svojstvenih vrijednosti koje moraju biti pozitivne). [3]

4.4. Sinteza regulatora metodom podešavanja polova

Ako je kompletni vektor stanja mjerljiv, tada je moguće zatvoriti direktnu regulacijsku petlju po vektoru stanja $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) + \mathbf{w}(t)$ gdje je **K** matrica pojačanja, a $\mathbf{w}(t)$ je referentni vektor vođenja. Nakon uvrštavanja zakona upravljanja $\mathbf{u}(t)$ u jednadžbe (4.3) i (4.4) problem sinteze regulatora podešavanjem polova sustava možemo formulirati na sljedeći način: za unaprijed zadane polove sustava s povratnom vezom λ_1, λ_2,...,λ_n koji su identični svojstvenim vrijednostima matrice (**A - BK**), treba odrediti matricu pojačanja **K**, uz pretpostavku da je sustav potpuno kontrolabilan dobivamo

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{\mathrm{r}}\mathbf{x}(t) + \mathbf{B}\mathbf{w}(t). \tag{4.20}$$

gdje je $\mathbf{A}_r = \mathbf{A} - \mathbf{BK}$.

Primijenimo li sada modalnu transformaciju $\mathbf{x}(t) = \mathbf{Pz}(t)$, jednadžba (4.20) postaje

$$\dot{\mathbf{z}}(t) = \mathbf{\Lambda z}(t) + \hat{\mathbf{B}}\mathbf{w}(t), \tag{4.21}$$

gdje je $\mathbf{\Lambda} = \mathbf{P}^{-1}\mathbf{A}_r\,\mathbf{P} = \text{diag}\{\lambda_1, \lambda_2,...,\lambda_n\}$, dok je $\hat{\mathbf{B}} = \mathbf{P}^{-1}\mathbf{B}$. S obzirom na to da operacija sličnosti matrica ne mijenja svojstvene vrijednosti , slijedi da su svojstvene vrijednosti matrice $\mathbf{A}_r$ (ili polovi) jednaki svojstvenim vrijednostima (dijagonalnim elementima) matrice $\mathbf{\Lambda}$.

Na osnovu izraza $\mathbf{A}_r = \mathbf{A} - \mathbf{BK}$ i $\mathbf{\Lambda} = \mathbf{P}^{-1}\mathbf{A}_r\,\mathbf{P}$ dobivamo

$$\mathbf{A} - \mathbf{BK} = \mathbf{P\Lambda P}^{-1}. \tag{4.22}$$

Nakon množenja s desne strane matricom $\mathbf{P}$ izraz (4.22) postaje

$$\mathbf{AP} - \mathbf{BKP} = \mathbf{P\Lambda}\,. \tag{4.23}$$

Uvedemo li sada oznaku $\hat{\mathbf{K}} = \mathbf{KP}$, te nakon prebacivanja pojedinih članova, izraz (4.23) postaje

$$\mathbf{AP} - \mathbf{P\Lambda} = \mathbf{B}\hat{\mathbf{K}}, \tag{4.24}$$

dok je

$$\mathbf{K} = \hat{\mathbf{K}}\mathbf{P}^{-1}. \tag{4.25}$$

Izraz (4.24) predstavlja linearnu matričnu jednadžbu po nepoznatoj matrici $\mathbf{P}$ dok su matrice $\mathbf{A}$, $\mathbf{B}$, $\mathbf{\Lambda}$ i $\hat{\mathbf{K}}$ unaprijed zadane. Dijagonalna matrica $\mathbf{\Lambda}$ sadrži željene svojstvene vrijednosti zatvorenog regulacijskog kruga, a matrica $\hat{\mathbf{K}}$ je proizvoljna konstantna matrica. Na kraju, matricu pojačanja $\mathbf{K}$ regulatora stanja dobivamo na osnovu jednadžbe (4.25). [3]

5. ANALIZA I SINTEZA LINEARNIH SUSTAVA PRIMJENOM HOPFIELDOVIH NEURONSKIH MREŽA

Stabilna stanja Hopfieldove neuronske mreže su određena iz sustava diferencijalnih jednadžbi (3.3) tako da se derivacija unutarnjeg potencijala izjednači s nulom ($du_j/dt = 0$).

$$-\propto_j u_j + \sum_{i=1}^{n} w_{ji}\, \psi(u_j) + \theta_j = 0 \qquad (j = 1,2,\dots,n) \tag{5.1}$$

Hopfield je pokazao da je dovoljan uvjet stabilnosti mreže, simetričnost sinaptičkih težina, $w_{ji}=w_{ij}$ i nule na dijagonali $w_{jj} = 0$.

Pokazano je da su stabilna stanja mreže u lokalnom minimumu funkcije energije

$$E(\mathbf{x}) = -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji}\, x_i x_j - \sum_{j=1}^{n} x_j\, \theta_j + \sum_{j=1}^{n}\frac{\alpha_j}{\gamma_j}\int_0^{x_j} \psi_j^{-1}(x)\mathrm{d}x \tag{5.2}$$

koja se može zapisati u kompaktnijem matričnom zapisu

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{W}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + \sum_{j=1}^{n}\frac{\alpha_j}{\gamma_j}\int_0^{x_j} \psi_j^{-1}(x)\mathrm{d}x \tag{5.3}$$

gdje je

$$\mathbf{x} = [x_1, x_2, \dots, x_n]^{\mathrm{T}},$$

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_n]^{\mathrm{T}}, \qquad \alpha_j = \frac{r_j}{R_j}$$

i $\psi_j^{-1}(x_j)$ je inverz aktivacijske funkcije. Prva dva člana funkcije energije odgovaraju kvadratičnoj funkciji čijim rješavanjem dobivamo rješenje željenog problema. Zadnji član funkcije energije se obično ne koristi kod projektiranja neuronske mreže, te njegova vrijednost ovisi o obliku nelinearne aktivacijske funkcije ψ_j. Za veliko pozitivno pojačanje γ_j, gdje je nagib od $\psi_j(u_j)$ približno beskonačan te se aktivacijska funkcija može zapisati kao signum funkcija

$$\psi(u_j) = \mathrm{sign}(u_j) = \begin{cases} -1 \text{ ako } u_j < 0 \\ +1 \text{ ako } u_j > 0 \end{cases} \tag{5.4}$$

taj član se može zanemariti pošto je njegova vrijednost vrlo mala. U tom slučaju se funkcija energije može zapisati u sljedećem obliku

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{W}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}. \tag{5.5}$$

Točke ravnoteže se nalaze u lokalnom minimumu funkcije energije (5.5) koja opisuje mrežu. Hopfieldova neuronska mreže može se smatrati procesom minimizacije funkcije energije.

Važno svojstvo Hopfieldove neuronske mreže je zagarantirana konvergencija u stabilno stanje pod uvjetom da je matrica **W** simetrična te da na dijagonali ima nule ($w_{jj} = 0$).

Lyapunovljeva teorija stabilnosti zahtjeva da funkcija energije bude monotono padajuća u vremenu. Uzimajući u obzir vremensku derivaciju funkcije energije (5.2)

$$\begin{aligned}\frac{\mathrm{d}E}{\mathrm{d}t} &= \sum_{j=1}^{n}\frac{\partial E}{\partial x_j}\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\sum_{j=1}^{n}\frac{\mathrm{d}x_j}{\mathrm{d}t}\left(\sum_{i=1}^{n} w_{ji}x_i + \theta_j - \alpha_j u_j\right) = -\sum_{j=1}^{n}\tau_j\frac{\mathrm{d}x_j}{\mathrm{d}t}\frac{\mathrm{d}u_j}{\mathrm{d}t} \\ &= -\sum_{j=1}^{n}\tau_j\frac{\mathrm{d}u_j}{\mathrm{d}x_j}\left(\frac{\mathrm{d}x_j}{\mathrm{d}t}\right)^2 = -\sum_{j=1}^{n}\tau_j\left[\psi_j^{-1}(x_j)\right]'\left(\frac{\mathrm{d}x_j}{\mathrm{d}t}\right)^2\end{aligned} \tag{5.6}$$

Kako je vremenska konstanta $\tau_j = r_j C_j$ pozitivna za svaki j i nelinearni inverz funkcije $\psi_j^{-1}(x_j)$ je monotono rastuća ($[\psi_j^{-1}(x_j)]' = \mathrm{d}u_j / \mathrm{d}x_j > 0$) tako da možemo napisati

$$\frac{\mathrm{d}E}{\mathrm{d}t} \leq 0 \tag{5.7}$$

$\mathrm{d}E/\mathrm{d}t = 0$ podrazumijeva $\mathrm{d}x_j/\mathrm{d}t = 0$ za svaki j. Ako uzmemo u obzir tu činjenicu i činjenicu da je energija $E(\mathbf{x})$ ograničena od ispod možemo zaključiti da mreža konvergira u stabilno stanje koje je lokalni minimum funkcije energije $E(\mathbf{x})$. Treba uzeti u obzir da se funkcija energije računa kao prvi integral diferencijalne dinamičke jednadžbe (3.3). Drugim riječima, Hopfieldov model je gradijentni sustav opisan matričnom diferencijalnom jednadžbom

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -\boldsymbol{\mu}\nabla_x E(\mathbf{x}), \tag{5.8}$$

gdje je

$$\mathbf{u} = [u_1, u_2, \ldots, u_n]^{\mathrm{T}},$$

$$\boldsymbol{\mu} = \mathrm{diag}(\tau_1^{-1}, \tau_2^{-1}, \ldots \tau_n^{-1}),$$

$$\nabla_x E(\mathbf{x}) = \left[\frac{\partial E}{\partial x_1}, \frac{\partial E}{\partial x_2}, \ldots \frac{\partial E}{\partial x_n},\right]^{\mathrm{T}}.$$

Funkcija energije je neprocjenjiv alat zato što nam omogućava da komplicirane dinamičke mreže pretvorimo u oblik optimizacije. [2]

5.1. Rješavanje problema sustava linearnih jednadžbi

5.1.1. Algoritam učenja mreže za rješavanje sustava linearnih jednadžbi

Najjednostavniji problem sustava linearnih jednadžbi zapisanog u matričnom obliku je onaj gdje je nepoznanica vektor.

$$\mathbf{Ax} = \mathbf{b} \tag{5.9}$$

Pogreška ovog sustava jednadžbi se može jednostavno zapisati kao

$$\mathbf{e} = \mathbf{Ax} - \mathbf{b} \tag{5.10}$$

Kako bi ovaj sustav linearnih jednadžbi riješili pomoću Hopfieldove neuronske mreže moramo definirati pozitivno definitnu funkciju cilja (kvadrat pogreške)

$$E = \frac{1}{2}\|\mathbf{e}\|^2 = \frac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{e} \tag{5.11}$$

uvrštavanjem pogreške (5.10) u funkcija cilja (5.11) dobije se

$$E = \frac{1}{2}(\mathbf{Ax} - \mathbf{b})^{\mathrm{T}}(\mathbf{Ax} - \mathbf{b}) \tag{5.12}$$

Da bi sredili izraz moramo transponirati prvu zagradu te izmnožiti zagrade

$$E = \frac{1}{2}\left(\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} - \mathbf{b}^{\mathrm{T}}\right)(\mathbf{Ax} - \mathbf{b}) = \frac{1}{2}\left(\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Ax} - \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{b} - \mathbf{b}^{\mathrm{T}}\mathbf{Ax} + \mathbf{b}^{\mathrm{T}}\mathbf{b}\right) \tag{5.13}$$

$$\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{b} = \mathbf{y}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{b}^{\mathrm{T}}\mathbf{Ax} = \mathbf{b}^{\mathrm{T}}\mathbf{y}$$

$$\mathbf{b}^{\mathrm{T}}\mathbf{y} = \mathbf{y}^{\mathrm{T}}\mathbf{b}$$

gdje je

$$\mathbf{y} = \mathbf{Ax}$$

tako da možemo zbrojiti srednja dva člana te dobivamo funkciju cilja

$$E = \frac{1}{2}\left(\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Ax} - 2\mathbf{b}^{\mathrm{T}}\mathbf{Ax} + \mathbf{b}^{\mathrm{T}}\mathbf{b}\right) \tag{5.14}$$

kako bi mogli primijeniti pravilo učenja u kontinuiranom obliku (gradijentni algoritam)

$$\dot{\mathbf{x}} = -\mu\left(\frac{\partial E}{\partial \mathbf{x}}\right)^{\mathrm{T}} \tag{5.15}$$

moramo pronaći derivaciju funkcije cilja

kada primijenimo pravila za derivaciju matrica

$$\frac{\partial(\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x})}{\partial\mathbf{x}} = 2\mathbf{x}^{\mathrm{T}}\mathbf{A}$$
$$\frac{\partial(\mathbf{A}\mathbf{x})}{\partial\mathbf{x}} = \mathbf{A}$$

dobijemo

$$\left(\frac{\partial E}{\partial\mathbf{x}}\right)^{\mathrm{T}} = \left(\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A} - \mathbf{b}^{\mathrm{T}}\mathbf{A}\right)^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \mathbf{A}^{\mathrm{T}}\mathbf{b} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{5.16}$$

uvrštavanjem derivacije (5.16) u gradijentni algoritam (5.15) dobivamo algoritam za učenje mreže

$$\dot{\mathbf{x}} = -\mu\,\mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{5.17}$$

5.1.2. Primjer sustava linearnih jednadžbi

Treba riješiti sustav linearnih jednadžbi s 3 nepoznanice

$$4x_1 + 2x_2 + x_3 = 13$$
$$2x_1 - 3x_2 + 2x_3 = 7$$
$$3x_1 + 2x_2 - x_3 = 5$$

koja u matričnom zapisu ima sljedeći oblik

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

gdje su

$$\mathbf{A} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & -3 & 2 \\ 3 & 2 & -1 \end{bmatrix},$$
$$\mathbf{x} = [x_1 \quad x_2 \quad x_3]^{\mathrm{T}},$$
$$\mathbf{b} = [13 \quad 7 \quad 5]^{\mathrm{T}}.$$

Kako bi mogli provjeriti rezultate dobivene Hopfieldovom neuronskom mrežom, prvo ćemo izračunati $\mathbf{x}$ jednostavnim množenjem cijele jednadžbe sa $\mathbf{A}^{-1}$ s lijeve strane

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.$$

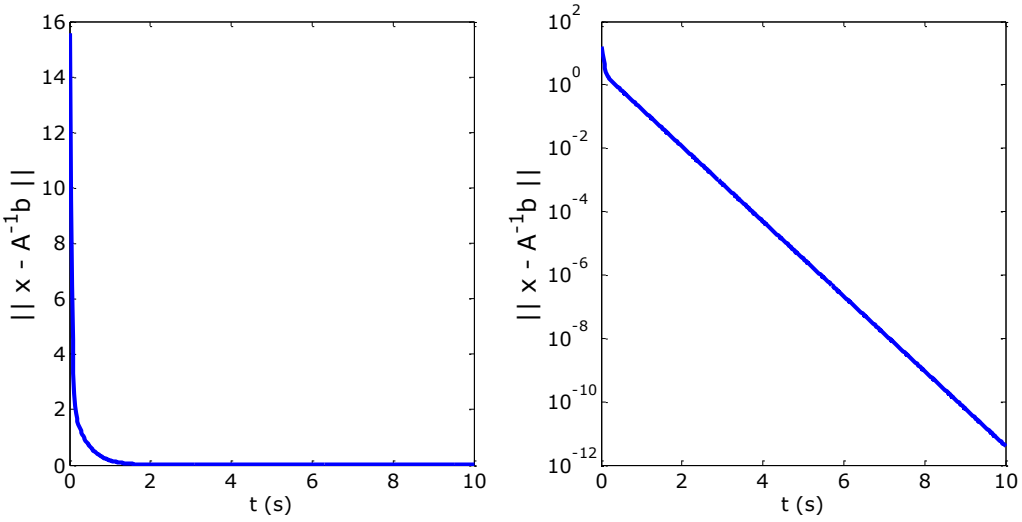
Matlab kod za rješavanje ovog problema nalazi se u prilogu. Rješenje ove jednadžbe dobiveno pomoću Hopfieldove neuronske mreže je:

$$\mathbf{x} = [2 \quad 1 \quad 3]^{\mathrm{T}}$$

Razlika između egzaktnog rješenja i rješenja dobivenog Hopfieldovom neuronskom mrežom je:

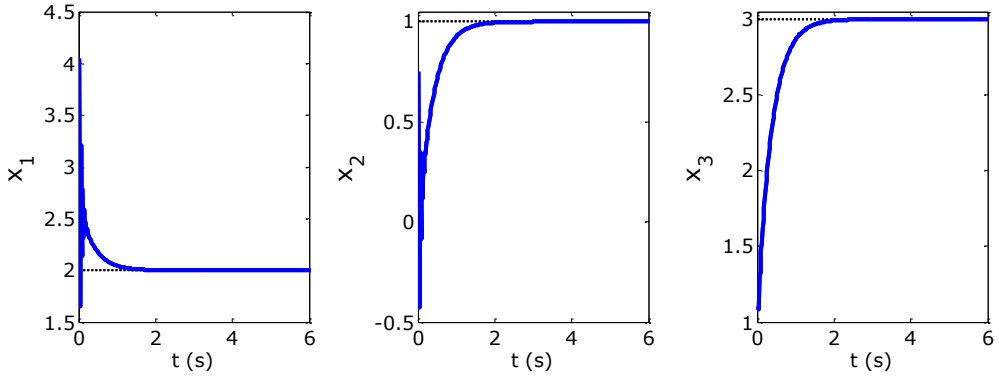
$$\mathbf{x} - \mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} 1{,}0434 \cdot 10^{-12} \\ -1{,}696 \cdot 10^{-12} \\ -3{,}114 \cdot 10^{-12} \end{bmatrix}$$

Pogreške i konvergencija rješenja dobivenih Matlabom prikazana su na sljedećim dijagramima:



Slika 6. Pogreška mreže kod rješavanja sustava linearnih jednadžbi

Na slici 6 prikazana je pogreška mreže u normalnom mjerilu na lijevom dijagramu i u logaritamskom mjerilu na desnom dijagramu. Iz ovih dijagrama se može vidjeti da pogreška monotono opada u vremenu.



Slika 7. Konvergencija rješenja sustava linearnih jednadžbi

Na slici 7 se može vidjeti da vektor **x** konvergira u stabilna stanja koja su ujedno i rješenja linearne jednadžbe.

5.2. Rješavanje problema inverza matrice

5.2.1. Algoritam učenja mreže za invertiranje matrice

Problem inverza matrice može se zapisati u matričnom obliku

$$\mathbf{AX} = \mathbf{B} \tag{5.18}$$

gdje je matrica **B** jedinična matrica. Izraz za gradijentni algoritam će se izvesti pomoću izraza (5.18), a zatim će se matrica **B** zamijeniti jediničnom matricom. Pogreška ovog sustava jednadžbi može se jednostavno zapisati kao

$$\mathbf{e} = \mathbf{AX} - \mathbf{B} \tag{5.19}$$

Kako bi ovaj riješili problem inverza matrice pomoću Hopfieldove neuronske mreže moramo definirati pozitivno definitnu funkciju cilja (kvadrat Frobeniusove norme)

$$E = \frac{1}{2}\|\mathbf{e}\|_F^2 = \frac{1}{2}\mathrm{Tr}\left(\mathbf{e}\mathbf{e}^{\mathrm{T}}\right) \tag{5.20}$$

te uvrštavanjem pogreške (5.19) u funkcija cilja (5.20) i njenim sređivanjem dobijemo

$$\begin{aligned} E &= \frac{1}{2}\mathrm{Tr}\{(\mathbf{AX}-\mathbf{B})(\mathbf{AX}-\mathbf{B})^{\mathrm{T}}\} = \frac{1}{2}\mathrm{Tr}\left\{(\mathbf{AX}-\mathbf{B})\left(\mathbf{X}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}-\mathbf{B}^{\mathrm{T}}\right)\right\} = \\ &= \frac{1}{2}\mathrm{Tr}\{\mathbf{AXX}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} - \mathbf{BX}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} - \mathbf{AXB}^{\mathrm{T}} + \mathbf{BB}^{\mathrm{T}}\} \end{aligned} \tag{5.21}$$

Trag zbroja može se zapisati kao zbroj tragova

$$E = \frac{1}{2}\mathrm{Tr}\{\mathbf{AXX}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\} - \frac{1}{2}\mathrm{Tr}\{\mathbf{BX}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\} - \frac{1}{2}\mathrm{Tr}\{\mathbf{AXB}^{\mathrm{T}}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{BB}^{\mathrm{T}}\} \tag{5.22}$$

Trag matrice i trag transponirane matrice je jednak (Tr (**A**) = Tr ($\mathbf{A}^{\mathrm{T}}$))

$$\mathrm{Tr}\{\mathbf{BX}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\} = \mathrm{Tr}\{\mathbf{B}(\mathbf{AX})^{\mathrm{T}}\} = \mathrm{Tr}\{(\mathbf{AX})\mathbf{B}^{\mathrm{T}}\} = \mathrm{Tr}\{\mathbf{AXB}^{\mathrm{T}}\} \tag{5.23}$$

tako da srednja dva člana u jednadžbi (5.22) možemo zbrojiti te dobijemo funkciju cilja

$$E = \frac{1}{2}\mathrm{Tr}\{\mathbf{AXX}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\} - \mathrm{Tr}\{\mathbf{AXB}^{\mathrm{T}}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{BB}^{\mathrm{T}}\} \tag{5.24}$$

kako bi mogli primijeniti pravilo učenja u kontinuiranom obliku (gradijentni algoritam)

$$\dot{\mathbf{X}} = -\mu\left(\frac{\partial E}{\partial \mathbf{X}}\right)^{\mathrm{T}} \tag{5.25}$$

moramo pronaći derivaciju funkcije cilja. Kada primijenimo pravila za derivaciju tragova matrica

$$\frac{\partial}{\partial \mathbf{X}}\mathrm{Tr}\left(\mathbf{AXBX}^{\mathrm{T}}\mathbf{C}\right) = \mathbf{A}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\mathbf{XB}^{\mathrm{T}} + \mathbf{CAXB}$$

$$\frac{\partial}{\partial \mathbf{X}}\mathrm{Tr}(\mathbf{AXB}) = \mathbf{A}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}$$

dobijemo

$$\left(\frac{\partial E}{\partial \mathbf{X}}\right)^{\mathrm{T}} = \frac{1}{2}\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{X} + \mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{X}\right) - \mathbf{A}^{\mathrm{T}}\mathbf{B} = \mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{X} - \mathbf{A}^{\mathrm{T}}\mathbf{B} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{X} - \mathbf{B}) \tag{5.26}$$

Uvrštavanjem dobivene derivacije u gradijentni algoritam dobivamo algoritam za učenje mreže

$$\dot{\mathbf{X}} = -\mu\, \mathbf{A}^{T}(\mathbf{A}\mathbf{X} - \mathbf{B}) \tag{5.27}$$

Kako bi ovaj algoritam mogli iskoristiti za rješavanje problema inverza matrice, matricu **B** moramo zamijeniti matricom **I** te dobijemo sljedeći izraz za učenje mreže

$$\dot{\mathbf{X}} = -\mu\, \mathbf{A}^{T}(\mathbf{A}\mathbf{X} - \mathbf{I}) \tag{5.28}$$

5.2.2. Primjer invertiranja matrice

U ovom primjeru će se tražiti inverz od matrice **A**. Ovaj problem se u matričnom obliku može zapisati kao

$$\mathbf{A}\mathbf{X} = \mathbf{I}$$

gdje je

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 6 \\ 2 & 2 & -4 \\ -2 & 1 & 9 \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix},$$

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

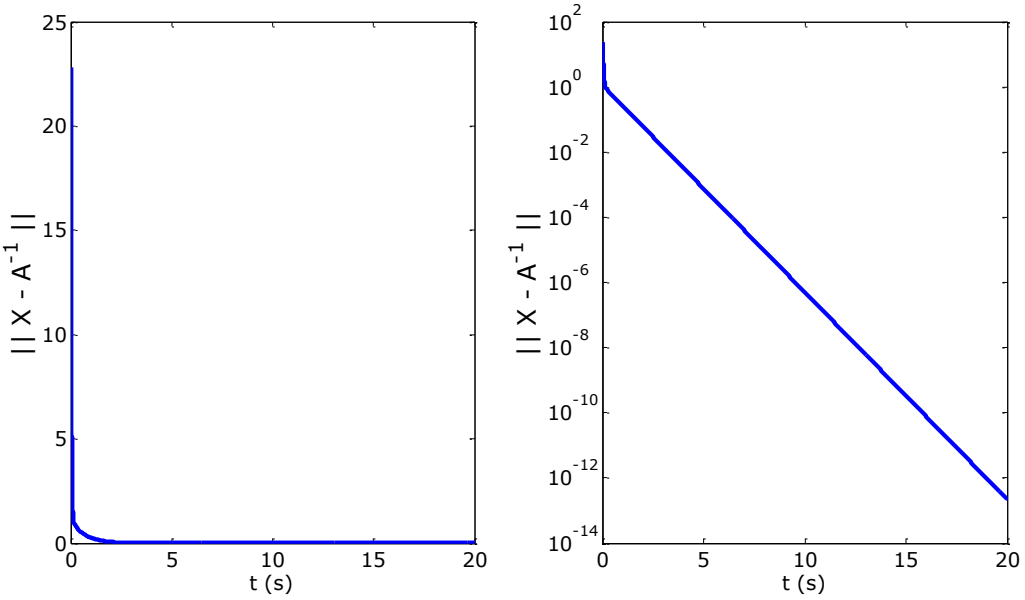
Matrica **A** je matrica čiji inverz tražimo, a **X** je inverz te matrice. Njihovim umnoškom se treba dobiti jedinična matrica. Matlab kod za rješavanje ovog problema nalazi se u prilogu. Rješenje ove jednadžbe dobiveno pomoću Hopfieldove neuronske mreže je:

$$\mathbf{X} = \begin{bmatrix} 1{,}2222 & -1{,}6667 & -1{,}5556 \\ -0{,}5556 & 1{,}6667 & 0.8889 \\ 0{,}3333 & -0{,}5 & -0.3333 \end{bmatrix},$$

Razlika između egzaktnog rješenja i rješenja dobivenog Hopfieldovom neuronskom mrežom:

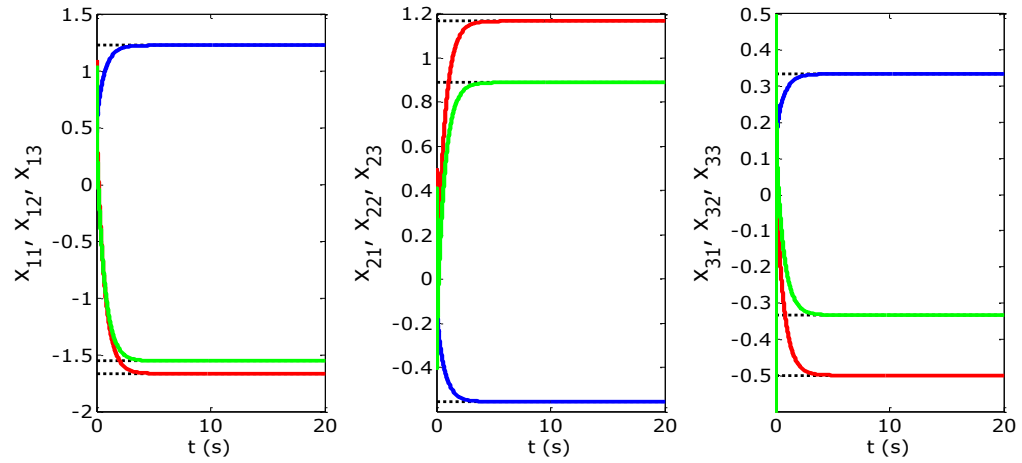
$$\mathbf{X} - \mathrm{inv}(\mathbf{A}) = \begin{bmatrix} -1{,}3434 \cdot 10^{-13} & 4{,}2588 \cdot 10^{-13} & 3{,}7992 \cdot 10^{-13} \\ 8{,}1268 \cdot 10^{-14} & -2{,}5668 \cdot 10^{-13} & -2{,}2915 \cdot 10^{-13} \\ -3{,}5472 \cdot 10^{-14} & 1{,}1247 \cdot 10^{-13} & 1{,}0003 \cdot 10^{-13} \end{bmatrix}.$$

Pogreške i konvergencija rješenja dobivenih Matlabom prikazana su na sljedećim dijagramima:



Slika 8. Pogreška mreže kod invertiranja matrice

Na slici 8 prikazana je pogreška mreže u normalnom mjerilu na lijevom dijagramu i u logaritamskom mjerilu na desnom dijagramu. Iz ovih dijagrama se može vidjeti da pogreška monoton opada u vremenu.



Slika 9. Konvergencija članova invertirane matrice

Na slici 9 se vidi da vrijednosti matrice **X** konvergiraju u stabilna stanja koja su ujedno i članovi inverzne matrice koja je zadana u primjeru.

5.3. Rješavanje Lyapunovljeve matrične jednadžbe

5.3.1. Algoritam učenja mreže za rješavanje Lyapunovljeve matrične jednadžbe

Lyapunovljeve jednadžbe se pojavljuju u mnogim granama upravljačke teorije, kao što su analiza stabilnosti i optimalna kontrola. Ova jednadžba i srodne jednadžbe su dobile ima prema Ruskom matematičaru Aleksandr Lyapunov.

Lyapunovljeva jednadžba u matričnom obliku

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q} \tag{5.29}$$

gdje su matrice **A**, **P**, **Q** kvadratične matrice, te su **P** i **Q** simetrične pozitivno definitne matrice. Matrica **P** je rješenje ove jednadžbe. Problem je što se do tog rješenja ne dolazi jednostavnim matematičkim računom. Kod rješavanja ove jednadžbe Hopfieldovom neuronskom mrežom moramo pronaći funkciju cilja. Funkcija cilja se može zapisati kao kvadrat Frobeniusove norme pogreške

$$E = \frac{1}{2}\|\mathbf{e}\|_F^2 = \frac{1}{2}\mathrm{Tr}\{\mathbf{e}\mathbf{e}^{\mathrm{T}}\} \tag{5.30}$$

gdje je

$$\mathbf{e} = \mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q}. \tag{5.31}$$

Uvrštavanjem pogreške (5.31) u funkciju cilja (5.30) dobivamo

$$E = \frac{1}{2}\mathrm{Tr}\left\{\left(\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q}\right)\left(\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q}\right)^{\mathrm{T}}\right\} \tag{5.32}$$

$$E = \frac{1}{2}\mathrm{Tr}\left\{\left(\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q}\right)\left(\mathbf{P}^{\mathrm{T}}\mathbf{A} + \mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} + \mathbf{Q}^{\mathrm{T}}\right)\right\}$$

Nakon što izmnožimo zagrade

$$\begin{aligned} E = \frac{1}{2}\mathrm{Tr}\{&\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{A} + \mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} + \mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{Q} + \mathbf{P}\mathbf{A}\mathbf{P}^{\mathrm{T}}\mathbf{A} + \mathbf{P}\mathbf{A}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} \\ &+\mathbf{P}\mathbf{A}\mathbf{Q}^{\mathrm{T}} + \mathbf{Q}\mathbf{P}^{\mathrm{T}}\mathbf{A} + \mathbf{Q}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} + \mathbf{Q}\mathbf{Q}^{\mathrm{T}}\} \end{aligned} \tag{5.33}$$

i trag zbroja razdvojimo na zbroj tragova

$$\begin{aligned} E = &\frac{1}{2}\mathrm{Tr}\{\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{A}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{P}\mathbf{A}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\} \\ &+\frac{1}{2}\mathrm{Tr}\{\mathbf{P}\mathbf{A}\mathbf{P}^{\mathrm{T}}\mathbf{A}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{Q}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{Q}\mathbf{P}^{\mathrm{T}}\mathbf{A}\} \\ &+\frac{1}{2}\mathrm{Tr}\{\mathbf{P}\mathbf{A}\mathbf{Q}^{\mathrm{T}}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{Q}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{Q}\mathbf{Q}^{\mathrm{T}}\} \end{aligned} \tag{5.34}$$

kako bi mogli primijeniti pravilo učenja u kontinuiranom obliku (gradijentni algoritam)

$$\dot{\mathbf{X}} = -\mu\frac{\partial E}{\partial \mathbf{P}} \tag{5.35}$$

moramo pronaći derivaciju funkcije cilja.

Kada primijenimo pravila za derivaciju tragova matrica

$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{A}\right)=\mathbf{A}\mathbf{A}^{\mathrm{T}}\mathbf{P}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}(\mathbf{P}\mathbf{A}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}})=\mathbf{P}\mathbf{A}\mathbf{A}^{\mathrm{T}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\right)=\frac{1}{2}\mathbf{A}\mathbf{P}\mathbf{A}+\frac{1}{2}\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A}^{\mathrm{T}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{P}\mathbf{A}\mathbf{P}^{\mathrm{T}}\mathbf{A}\right)=\frac{1}{2}\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A}^{\mathrm{T}}+\frac{1}{2}\mathbf{A}\mathbf{P}\mathbf{A}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{Q}\right)=\frac{1}{2}\mathbf{A}\mathbf{Q}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{Q}\mathbf{P}^{\mathrm{T}}\mathbf{A}\right)=\frac{1}{2}\mathbf{A}\mathbf{Q}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{P}\mathbf{A}\mathbf{Q}^{\mathrm{T}}\right)=\frac{1}{2}\mathbf{Q}\mathbf{A}^{\mathrm{T}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}(\mathbf{Q}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}})=\frac{1}{2}\mathbf{Q}\mathbf{A}^{\mathrm{T}}$$

dobijemo

$$\frac{\partial E}{\partial \mathbf{P}}=\mathbf{A}\mathbf{A}^{\mathrm{T}}\mathbf{P}+\mathbf{P}\mathbf{A}\mathbf{A}^{\mathrm{T}}+\mathbf{A}\mathbf{P}\mathbf{A}+\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A}^{\mathrm{T}}+\mathbf{A}\mathbf{Q}+\mathbf{Q}\mathbf{A}^{\mathrm{T}} \tag{5.36}$$

kada izlučimo matrice A i A^T te grupiramo varijable dobijemo sljedeći izraz

$$\frac{\partial E}{\partial \mathbf{P}}=\mathbf{A}(\mathbf{A}^{\mathrm{T}}\mathbf{P}+\mathbf{P}\mathbf{A}+\mathbf{Q})+(\mathbf{A}^{\mathrm{T}}\mathbf{P}+\mathbf{P}\mathbf{A}+\mathbf{Q})\mathbf{A}^{\mathrm{T}} \tag{5.37}$$

Kada uvrstimo derivaciju funkcije cilja (5.37) u jednadžbu gradijentnog algoritma (5.35) dobijemo algoritam za učenje neuronske mreže

$$\dot{\mathbf{X}}=-\mu\,(\mathbf{A}\,(\mathbf{A}^{\mathrm{T}}\mathbf{P}+\mathbf{P}\mathbf{A}+\mathbf{Q})+(\mathbf{A}^{\mathrm{T}}\mathbf{P}+\mathbf{P}\mathbf{A}+\mathbf{Q})\mathbf{A}^{\mathrm{T}}) \tag{5.38}$$

5.3.2. Primjer rješavanja Lyapunovljeve jednadžbe pomoću neuronskih mreža

Zadan je problem Lyapunovljeve jednadžbe

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q}$$

gdje su matrice **A** i **Q** zadane i traži se matrica **P**

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -3 & -3 \end{bmatrix},$$
$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}.$$

Kako bi mogli provjeriti dobiveno rješenje, rješenje jednadžbe može se dobiti preko Matlabove funkcije lyap (**A'**, **Q**).

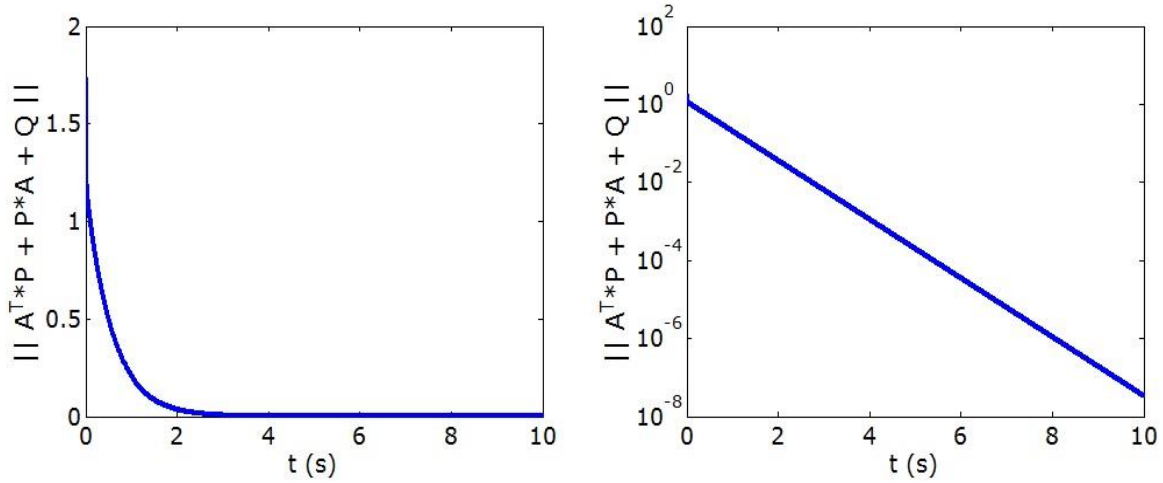
Matlab kod za rješavanje ovog problema nalazi se u prilogu. Rješenje ove jednadžbe dobiveno pomoću Hopfieldove neuronske mreže je

$$\mathbf{P} = \begin{bmatrix} 2{,}3125 & 1{,}9375 & 0{,}5 \\ 1{,}9375 & 3{,}25 & 0{,}8125 \\ 0{,}5 & 0{,}8125 & 0{,}4375 \end{bmatrix}.$$

Razlika između egzaktnog rješenja i rješenja dobivenog Hopfieldovom neuronskom mrežom:

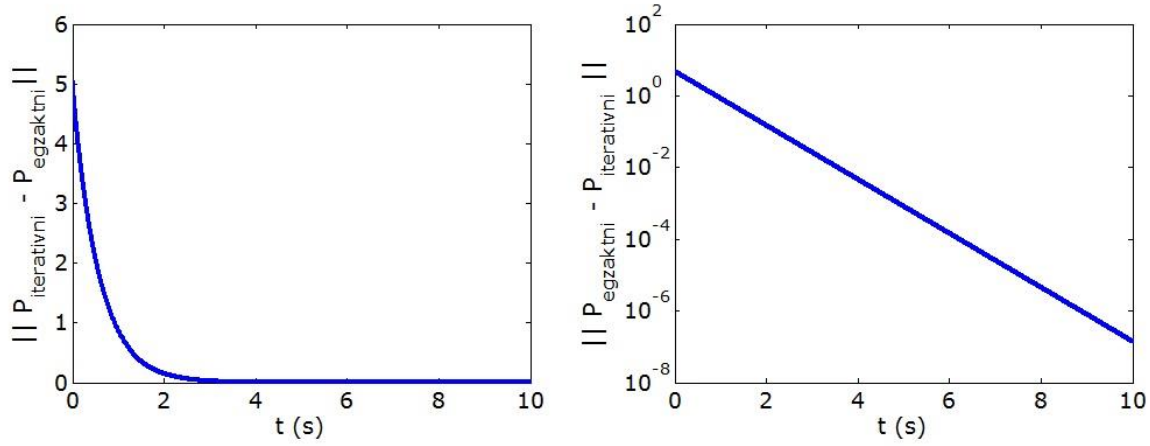
$$\mathbf{P} - \mathrm{lyap}(\mathbf{A}', \mathbf{Q}) = \begin{bmatrix} -7{,}219 \cdot 10^{-8} & -6{,}494 \cdot 10^{-8} & -1{,}587 \cdot 10^{-8} \\ -6{,}494 \cdot 10^{-8} & -7{,}915 \cdot 10^{-8} & -2{,}251 \cdot 10^{-8} \\ -1{,}587 \cdot 10^{-8} & -2{,}251 \cdot 10^{-8} & -8{,}396 \cdot 10^{-9} \end{bmatrix}.$$

Pogreške i konvergencija rješenja dobivenih Matlabom prikazana su na sljedećim dijagramima:



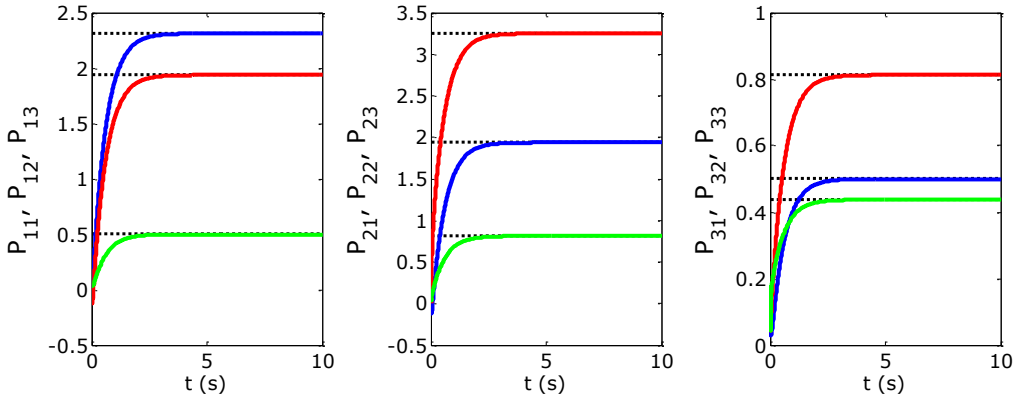
Slika 10. Pogreška rješenja

Na slici 10 možemo vidjeti Frobeniusovu normu pogreške u normalnom i logaritamskom mjerilu te na dijagramu možemo primijetiti da ta pogreška monotono opada s vremenom.



Slika 11. Razlika između egzaktnog i iterativnog rješenja

Na slici 11 možemo vidjeti Frobeniusovu normu razlike između egzaktnog i iterativno dobivenog rješenja u normalnom i logaritamskom mjerilu te na dijagramu možemo primijetiti da ta pogreška monotono opada s vremenom.



Slika 12. Konvergencija rješenja Lyapunovljeve jednadžbe

Na slici 12 se vidi da vrijednosti matrice **P** konvergiraju u stabilna stanja koja su ujedno i rješenja Lyapunovljeve jednadžbe.

5.4. Rješavanje matrične jednadžbe metode podešavanja polova

5.4.1. Algoritam učenja mreže za problem rješavanja sinteze regulatora metodom podešavanja polova

Kako bi neuronsku mrežu iskoristili za rješavanje problema sinteze regulator, prvo moramo pronaći funkciju cilja. Kako bi dobili funkciju cilja potrebna nam je pogreška sustava zadanog jednadžbom (4.24)

$$\mathbf{AP} - \mathbf{P\Lambda} = \mathbf{B}\hat{\mathbf{K}}. \tag{5.39}$$

Pogreška sustava dobije se prebacivanjem umnoška matrica $\mathbf{B}\hat{\mathbf{K}}$ na drugu stranu

$$\mathbf{e} = \mathbf{AP} - \mathbf{P\Lambda} - \mathbf{B}\hat{\mathbf{K}} \tag{5.40}$$

funkcija cilja se može zapisati kao kvadrat Frobeniusove norma pogreške

$$E = \frac{1}{2}\|\mathbf{e}\|_F^2 = \frac{1}{2}\mathrm{Tr}\{\mathbf{e}\mathbf{e}^{\mathrm{T}}\}, \tag{5.41}$$

uvrštavanjem jednadžbe (5.40) u jednadžbu (5.41) dobivamo izraz za funkciju cilja

$$\begin{aligned} E &= \frac{1}{2}\mathrm{Tr}\left\{\left(\mathbf{AP} - \mathbf{P\Lambda} - \mathbf{B}\hat{\mathbf{K}}\right)\left(\mathbf{AP} - \mathbf{P\Lambda} - \mathbf{B}\hat{\mathbf{K}}\right)^{\mathrm{T}}\right\} = \\ &= \frac{1}{2}\mathrm{Tr}\left\{\left(\mathbf{AP} - \mathbf{P\Lambda} - \mathbf{B}\hat{\mathbf{K}}\right)\left(\mathbf{P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} - \mathbf{\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} - \hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\right)\right\}, \end{aligned} \tag{5.42}$$

nakon što izmnožimo zagrade dobijemo

$$\begin{aligned} E = \frac{1}{2}\mathrm{Tr}\{&\mathbf{APP}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} - \mathbf{AP\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} - \mathbf{AP}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}} - \mathbf{P\Lambda P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} + \mathbf{P\Lambda\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} \\ &+\mathbf{P\Lambda}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}} - \mathbf{B}\hat{\mathbf{K}}\mathbf{P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} + \mathbf{B}\hat{\mathbf{K}}\mathbf{\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} + \mathbf{B}\hat{\mathbf{K}}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\}. \end{aligned} \tag{5.43}$$

Trag zbroja se rastavi na zbroj tragova

$$\begin{aligned} E = &\frac{1}{2}\mathrm{Tr}\{\mathbf{APP}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\} - \frac{1}{2}\mathrm{Tr}\{\mathbf{AP\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\} - \frac{1}{2}\mathrm{Tr}\left\{\mathbf{AP}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\right\} \\ &-\frac{1}{2}\mathrm{Tr}\{\mathbf{P\Lambda P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\} + \frac{1}{2}\mathrm{Tr}\{\mathbf{P\Lambda\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\} + \frac{1}{2}\mathrm{Tr}\left\{\mathbf{P\Lambda}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\right\} \\ &-\frac{1}{2}\mathrm{Tr}\left\{\mathbf{B}\hat{\mathbf{K}}\mathbf{P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\right\} + \frac{1}{2}\mathrm{Tr}\left\{\mathbf{B}\hat{\mathbf{K}}\mathbf{\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\right\} + \frac{1}{2}\mathrm{Tr}\left\{\mathbf{B}\hat{\mathbf{K}}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\right\}, \end{aligned} \tag{5.44}$$

kako bi mogli primijeniti pravilo učenja u kontinuiranom obliku (gradijentni algoritam)

$$\dot{\mathbf{X}} = -\mu\frac{\partial E}{\partial \mathbf{P}} \tag{5.45}$$

moramo pronaći derivaciju funkcije cilja.

Kada primijenimo pravila za derivaciju tragova matrica

$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{A}\mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\right)=\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{P}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}(\mathbf{A}\mathbf{P}\boldsymbol{\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}})=\frac{1}{2}\mathbf{A}^{\mathrm{T}}\mathbf{P}\boldsymbol{\Lambda}+\frac{1}{2}\mathbf{A}\mathbf{P}\boldsymbol{\Lambda}^{\mathrm{T}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{A}\mathbf{P}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\right)=\frac{1}{2}\mathbf{A}^{\mathrm{T}}\mathbf{B}\hat{\mathbf{K}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\right)=\frac{1}{2}\mathbf{A}^{\mathrm{T}}\mathbf{P}\boldsymbol{\Lambda}+\frac{1}{2}\mathbf{A}\mathbf{P}\boldsymbol{\Lambda}^{\mathrm{T}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\right)=\frac{1}{2}\mathbf{P}\left(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}+\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}\right)=\mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{P}\boldsymbol{\Lambda}\hat{\mathbf{K}}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\right)=\frac{1}{2}\mathbf{B}\hat{\mathbf{K}}\boldsymbol{\Lambda}^{\mathrm{T}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{B}\hat{\mathbf{K}}\mathbf{P}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\right)=\frac{1}{2}\mathbf{A}^{\mathrm{T}}\mathbf{B}\hat{\mathbf{K}}$$
$$\frac{\partial}{\partial \mathbf{P}}\frac{1}{2}\mathrm{Tr}(\mathbf{B}\hat{\mathbf{K}}\boldsymbol{\Lambda}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}})=\frac{1}{2}\mathbf{B}\hat{\mathbf{K}}\boldsymbol{\Lambda}^{\mathrm{T}}$$

dobijemo

$$\frac{\partial E}{\partial \mathbf{P}}=\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{P}-\mathbf{A}^{\mathrm{T}}\mathbf{P}\boldsymbol{\Lambda}-\mathbf{A}\mathbf{P}\boldsymbol{\Lambda}^{\mathrm{T}}-\mathbf{A}^{\mathrm{T}}\mathbf{B}\hat{\mathbf{K}}+\mathbf{B}\hat{\mathbf{K}}\boldsymbol{\Lambda}^{\mathrm{T}}+\mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}, \tag{5.46}$$

nakon što izlučimo $\mathbf{A}^{\mathrm{T}}$ i $\boldsymbol{\Lambda}^{\mathrm{T}}$ i uvrstimo u jednadžbu (5.45) dobijemo algoritam za učenje mreže

$$\dot{\mathbf{X}}=-\mu[\mathbf{A}^{\mathrm{T}}\left(\mathbf{A}\mathbf{P}-\mathbf{P}\boldsymbol{\Lambda}-\mathbf{B}\hat{\mathbf{K}}\right)-(\mathbf{A}\mathbf{P}-\mathbf{P}\boldsymbol{\Lambda}-\mathbf{B}\hat{\mathbf{K}})\boldsymbol{\Lambda}^{\mathrm{T}}] \tag{5.47}$$

5.4.2. Primjer rješavanja sinteze regulatora metodom podešavanja polova pomoću neuronskih mreža

Zadan je problem sinteze regulatora

$$\mathbf{A}\mathbf{P}-\mathbf{P}\boldsymbol{\Lambda}=\mathbf{B}\hat{\mathbf{K}},$$

gdje su zadane matrice $\mathbf{A}$, $\boldsymbol{\Lambda}$, $\mathbf{B}$ i $\hat{\mathbf{K}}$ te se traži matrica $\mathbf{P}$, a pojačanje regulatora se dobije iz formule (4.24)

$$\mathbf{K}=\hat{\mathbf{K}}\mathbf{P}^{-1}.$$

Kako bi mogli provjeriti dobiveno pojačanje, pojačanje se može dobiti preko Matlab-ove funkcije *place*(**A**, **B**, **pol**), gdje je **pol** vektor u kojem su polovi regulatora. Polovi regulatora se nalaze na dijagonali matrice $\boldsymbol{\Lambda}$.

5.4.2.1. Primjer 1

Zadane su matrice

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 3 & 3 \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\mathbf{\Lambda} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix},$$

$$\widehat{\mathbf{K}} = [1 \quad 1 \quad 1].$$

Matlab kod za rješavanje ovog problema nalazi se u prilogu. Rješenje jednadžbe za podešavanje polova dobiveno pomoću Hopfieldove neuronske mreže je

$$\mathbf{P} = \begin{bmatrix} 0{,}5 & 0{,}0667 & 0{,}0217 \\ -0{,}5 & -0{,}1333 & -0{,}0652 \\ 0{,}5 & 0{,}2667 & 0{,}1957 \end{bmatrix},$$

pojačanje je

$$\mathbf{K} = [7 \quad 14 \quad 9]$$

i polovi koji su dobiveni tako da izračunamo svojstvene vrijednosti matrice dobivene izrazom **A-BK**

$$\mathbf{p} = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}$$

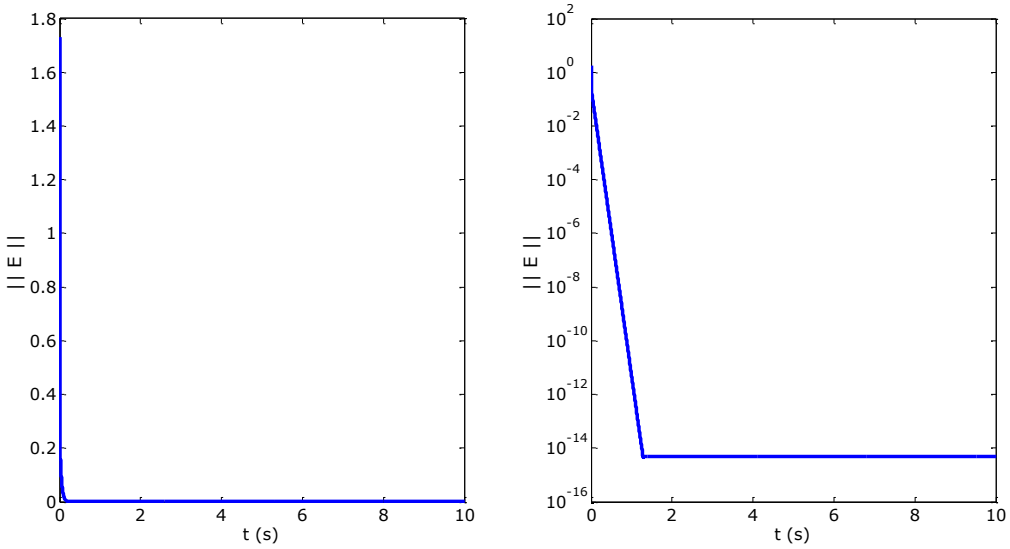
Razlika između egzaktnih rješenja i rješenja dobivenih pomoću Hopfieldove neuronske mreže:

$$\Delta\mathbf{P} = \begin{bmatrix} -0{,}9326 \cdot 10^{-14} & -0{,}0208 \cdot 10^{-14} & -0{,}0014 \cdot 10^{-14} \\ 0{,}9770 \cdot 10^{-14} & 0{,}0416 \cdot 10^{-14} & 0{,}0028 \cdot 10^{-14} \\ -0{,}5329 \cdot 10^{-14} & -0{,}0278 \cdot 10^{-14} & 0 \end{bmatrix},$$

$$\Delta\mathbf{K} = [-0{,}0985 \cdot 10^{-4} \quad -0{,}1221 \cdot 10^{-4} \quad -0{,}0332 \cdot 10^{-4}],$$

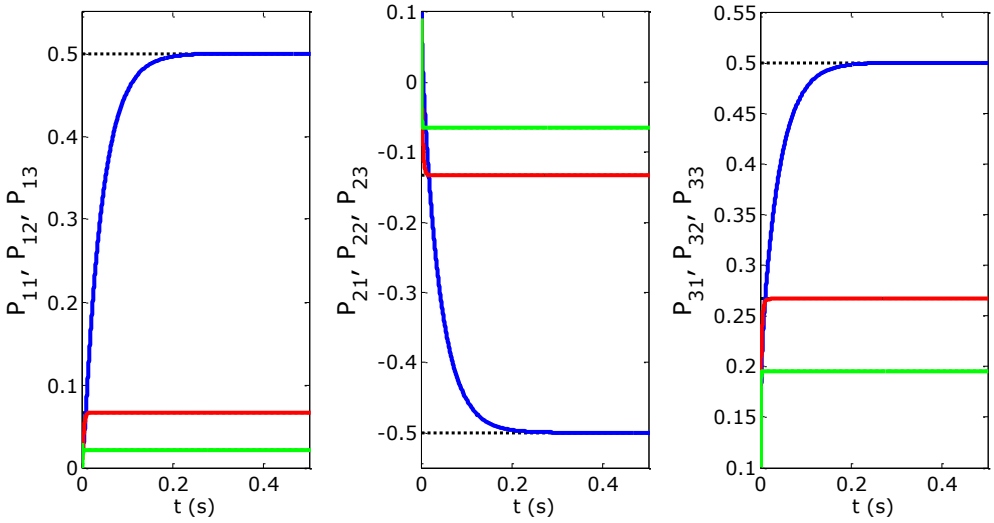
$$\Delta\mathbf{p} = \begin{bmatrix} -0.0486 \cdot 10^{-5} \\ -0.1260 \cdot 10^{-5} \\ -0.1579 \cdot 10^{-5} \end{bmatrix}.$$

Rezultati dobiveni Matlabom prikazana su na sljedećim dijagramima:



Slika 13. Pogreška matrice P

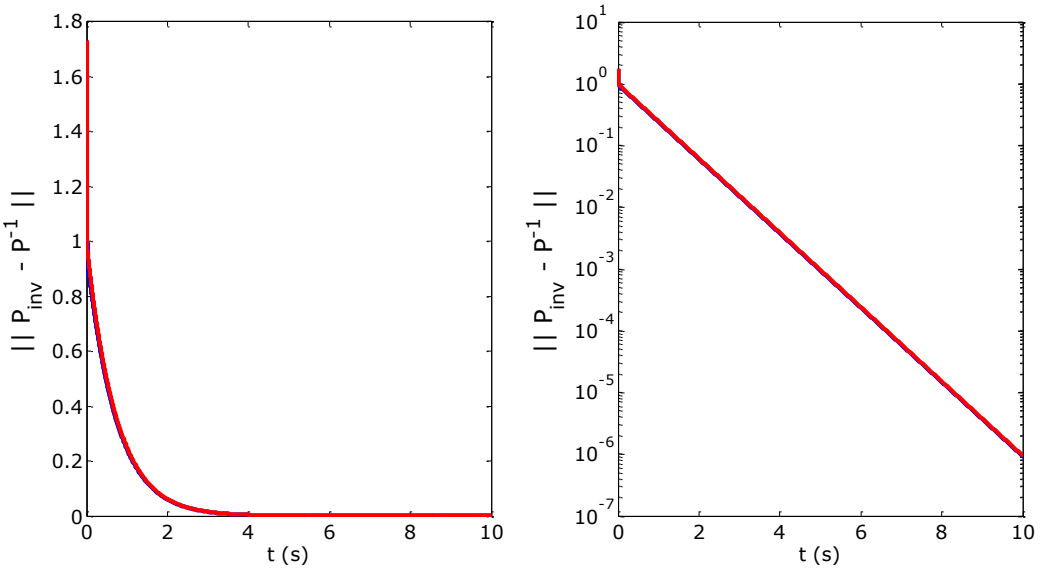
Na slici 13 možemo vidjeti Frobeniusovu normu pogreške u normalnom i logaritamskom mjerilu te na dijagramu možemo primijetiti da ta pogreška monotono opada s vremenom.



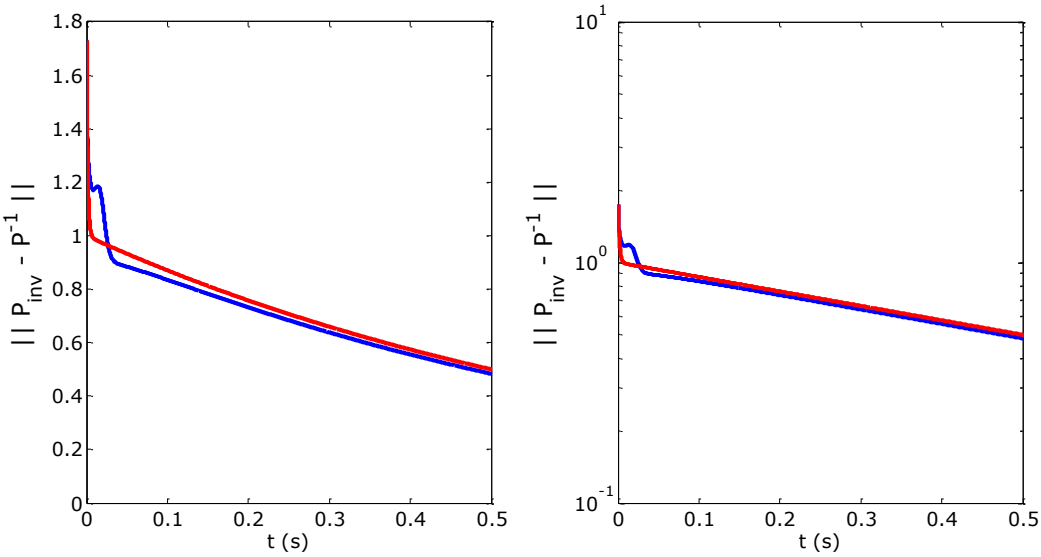
Slika 14. Konvergencija članova matrice P

Na slici 14 se vidi da vrijednosti matrice **P** konvergiraju u stabilna stanja. Kako je za izračun pojačanja potreban inverz matrice **P,** taj inverz je dobiven pomoću Hopfieldove neuronske mreže na dva načina:

- usporedno s matricom **P**
- nakon što je matrica **P** dobivena



Slika 15. Pogreška inverza matrice P



Slika 16. Pogreška inverza matrice P (uvećano)

Na slikama 15 i 16 je prikazana pogreška inverza matrice. Plavom linijom je prikazan slučaj gdje se inverz dobiva usporedno s matricom **P**, a crvena linija pokazuje slučaj gdje je inverz dobiven nakon što je matrica **P** dobivena. Vidljivo je da u oba slučaja pogreška inverza matrice monotono opada u vremenu nakon početnih oscilacija. U ovom primjeru bolje je inverz matrice računati nakon što je dobivena matrica **P**, čisto iz razloga što pogreška matrica **P** dosegne minimum u periodu od 1-2 sekundi, a inverz matrice **P** monotono opada i nakon 10 sekundi.

5.4.2.2. Primjer 2

Zadane su matrice

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 1 \\ 1 & 3 & 3 \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$\mathbf{\Lambda} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix},$$
$$\widehat{\mathbf{K}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Matlab kod za rješavanje ovog problema nalazi se u prilogu. Rješenje jednadžbe za podešavanje polova dobiveno pomoću Hopfieldove neuronske mreže je

$$\mathbf{P} = \begin{bmatrix} 0{,}5 & 0{,}8 & -0{,}5 \\ -0{,}5 & -1{,}6 & 1{,}5 \\ 0{,}5 & 1 & -0{,}5 \end{bmatrix},$$

pojačanje je

$$\mathbf{K} = \begin{bmatrix} -1 & 2 & 5 \\ -1 & 2 & 5 \end{bmatrix}.$$

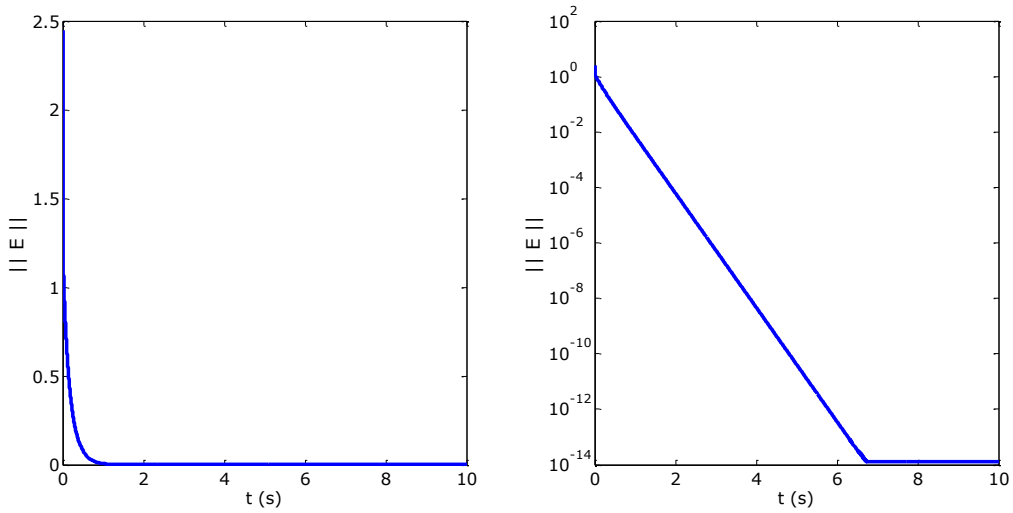
i polovi koji su dobiveni tako da izračunamo svojstvene vrijednosti matrice dobivene izrazom **A-BK**

$$\mathbf{p} = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}$$

Razlika između egzaktnih rješenja i rješenja dobivenih pomoću Hopfieldove neuronske mreže:

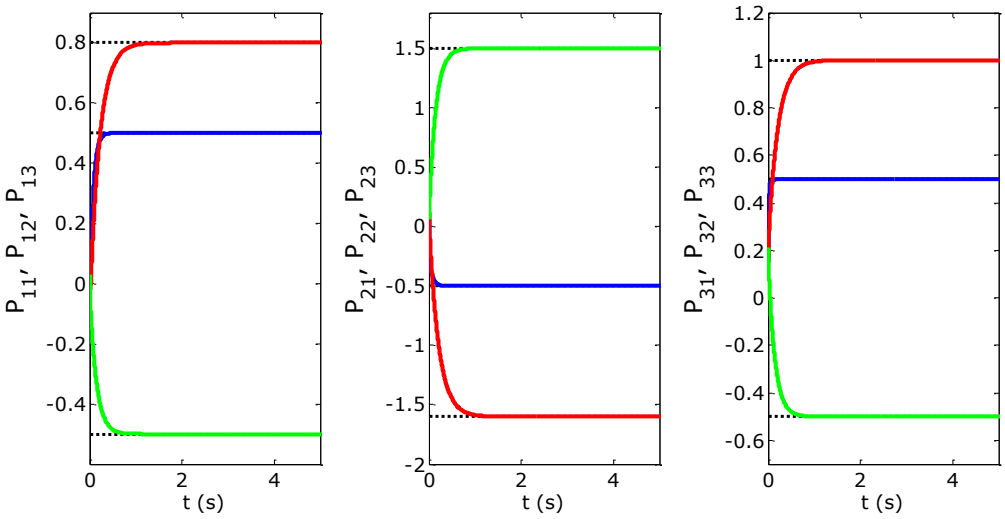
$$\Delta\mathbf{P} = \begin{bmatrix} -0{,}0211 \cdot 10^{-13} & -0{,}121 \cdot 10^{-13} & 0{,}045 \cdot 10^{-13} \\ 0{,}0105 \cdot 10^{-13} & 0{,}2243 \cdot 10^{-13} & -0{,}1377 \cdot 10^{-13} \\ -0{,}0017 \cdot 10^{-13} & -0{,}1077 \cdot 10^{-13} & 0{,}0638 \cdot 10^{-13} \end{bmatrix},$$
$$\Delta\mathbf{p} = \begin{bmatrix} -0.0028 \cdot 10^{-5} \\ -0.1443 \cdot 10^{-5} \\ -0.0265 \cdot 10^{-5} \end{bmatrix}.$$

Rezultati dobiveni Matlabom prikazana su na sljedećim dijagramima:



Slika 17. Pogreška matrice P

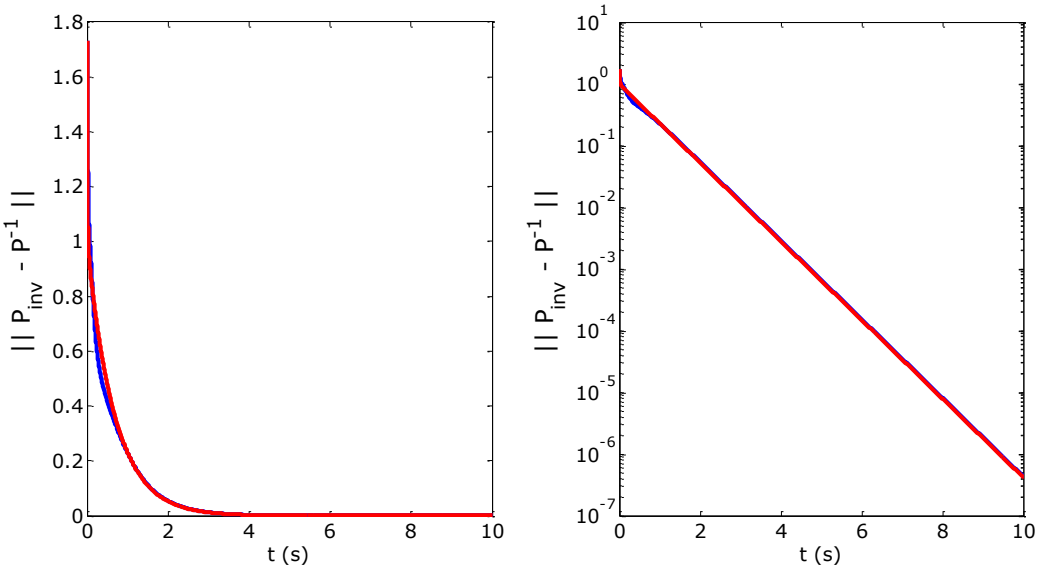
Na slici 17 možemo vidjeti Frobeniusovu normu pogreške u normalnom i logaritamskom mjerilu, te na dijagramu možemo primijetiti da ta pogreška monotono opada s vremenom.



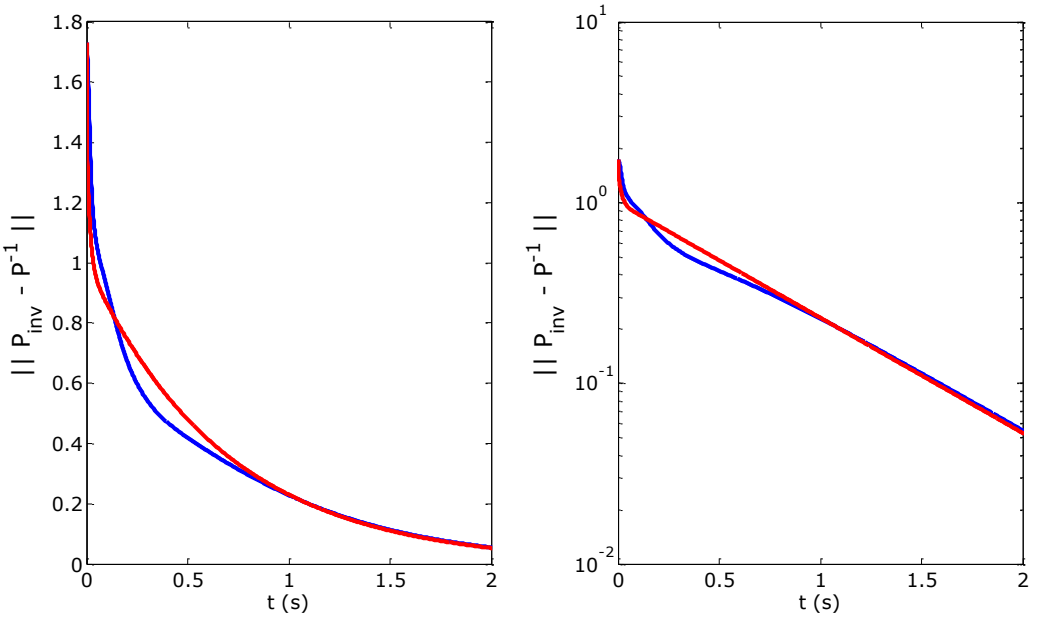
Slika 18. Konvergencija članova matrice P

Na slici 18 se vidi da vrijednosti matrice **P** konvergiraju u stabilna stanja. Kako je za izračun pojačanja potreban inverz matrice **P,** taj inverz je dobiven pomoću Hopfieldove neuronske mreže na dva načina:

- usporedno s matricom **P**
- nakon što je matrica **P** dobivena



Slika 19. Pogreška inverza matrice P



Slika 20. Pogreška inverza matrice P (uvećano)

Na slikama 19 i 20 je prikazana pogreška inverza matrice. Plavom linijom je prikazan slučaj gdje se inverz dobiva usporedno s matricom **P**, a crvena linija pokazuje slučaj gdje je inverz dobiven nakon što je matrica **P** dobivena. Vidljivo je da u oba slučaja pogreška inverza matrice monotono opada u vremenu nakon početnih oscilacija. U ovom primjeru razlika između opadanja pogreške matrice **P** i inverza matrice **P** je manja nego u prvom primjeru. Pogreška matrice P je u minimumu između 6-7 sekundi dok inverz matrice **P** opada i nakon 10 sekundi.

5.5. Rješavanje problema kvadratičnog programiranja s linearnim ograničenjima

5.5.1. Algoritam učenja mreže za rješavanje problema kvadratičnog programiranja s linearnim ograničenjima

Problem kvadratičnog programiranja je optimizacijski problem. Za razliku od prijašnjih primjera kod kvadratičnog programiranja funkcija cilja nam je unaprijed zadana jednadžbom

$$E = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x}\,, \tag{5.48}$$

uz ograničenja

$$\mathbf{A}\mathbf{x} - \mathbf{b} \geq 0\,, \tag{5.49}$$

$$\mathbf{E}\mathbf{x} - \mathbf{d} = 0\,, \tag{5.50}$$

gdje su zadani matrica **Q**, **A**, **E** i vektori **b**, **c**, **d**, a traži se vektor **x**. Kako bi mogli koristiti gradijentni algoritam za učenje mreže potrebno je derivirati funkciju cilja. Kada primijenimo pravila za derivaciju:

$$\frac{\partial\left(\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}\right)}{\partial\mathbf{x}} = 2\mathbf{x}^{\mathrm{T}}\mathbf{Q}\,,$$
$$\frac{\partial(\mathbf{c}^{\mathrm{T}}\mathbf{x})}{\partial\mathbf{x}} = \mathbf{c}^{\mathrm{T}}\,,$$

dobijemo

$$\frac{\partial E}{\partial\mathbf{x}} = \mathbf{x}^{\mathrm{T}}\mathbf{Q} + \mathbf{c}^{\mathrm{T}}\,. \tag{5.51}$$

Tada gradijentni algoritam možemo zapisati

$$\dot{\mathbf{x}} = -\mu\frac{\partial E}{\partial\mathbf{x}} = -\mu\left(\mathbf{x}^{\mathrm{T}}\mathbf{Q} + \mathbf{c}^{\mathrm{T}}\right), \tag{5.52}$$

ovaj oblik gradijentnog algoritma može se koristiti ako nema linearnih ograničenja. Za slučaj kada imamo ograničenja koristimo kaznene funkcije. Kaznena funkcija u matričnom obliku, može se zapisati

$$\mathbf{k}(\mathbf{x}) = \mathbf{A}^{\mathrm{T}}\mathbf{g}_{\mathrm{ne}}(\mathbf{x}) + \mathbf{E}^{\mathrm{T}}\mathbf{g}_{\mathrm{je}}(\mathbf{x})\,, \tag{5.53}$$

gdje su

$$\mathbf{g}_{\mathrm{ne}}(\mathbf{x}) = \min(\mathbf{A}\mathbf{x} - \mathbf{b}\,, \mathbf{0})\,, \tag{5.54}$$
$$\mathbf{g}_{\mathrm{je}}(\mathbf{x}) = \mathbf{E}\mathbf{x} - \mathbf{d}\,.$$

Vektor $\mathbf{g}_{\mathrm{ne}}$ sadrži vrijednosti odstupanja od ograničenja zadanih linearnom nejednadžbom, a vektor $\mathbf{g}_{\mathrm{je}}$ sadrži vrijednosti odstupanja od ograničenja zadanih linearnom jednadžbom. Kako bi mogli rješavati probleme kvadratičnog programiranja s linearnim ograničenjima, u gradijentni algoritam (5.52) se ubaci kaznena funkcija (5.53), te ima oblik

$$\dot{\mathbf{x}} = -\mu \frac{\partial E}{\partial \mathbf{x}} = -\mu \left(\mathbf{x}^{\mathrm{T}} \mathbf{Q} + \mathbf{c}^{\mathrm{T}} + \kappa \, \mathbf{k}^{\mathrm{T}}(\mathbf{x}) \right), \tag{5.55}$$

gdje je κ kazneni faktor.

5.5.2. Primjeri rješavanja problema kvadratičnog programiranja s linearnim ograničenjima pomoću neuronskih mreža

5.5.2.1. Rješavanje problema linearnog programiranja s linearnim ograničenjima

Kako bi algoritam prilagodili rješavanju linearnog programiranja, matrica **Q** je matrica nula. U ovom primjeru traži se minimum iduće funkcije

$$\min f(\mathbf{x}) = -2x_1 - 4x_2 - 3x_3,$$

uz ograničenja,

$$\begin{aligned} &r_1(\mathbf{x}) = -3x_1 - 4x_2 - 2x_3 + 60 \geq 0, \\ &r_2(\mathbf{x}) = -2x_1 - x_2 - 2x_3 + 40 \geq 0, \\ &r_3(\mathbf{x}) = -x_1 - 3x_2 - 2x_3 + 80 \geq 0, \\ &x_1 \geq 0, \qquad x_2 \geq 0, \qquad x_3 \geq 0 \end{aligned}$$

Ovaj problem može se zapisati u matričnom obliku

$$\min f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^{\mathrm{T}} \mathbf{Q} \mathbf{x} + \mathbf{c}^{\mathrm{T}} \mathbf{x},$$

$$\mathbf{A}\mathbf{x} - \mathbf{b} \geq 0,$$

gdje su,

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{c} = \begin{bmatrix} -2 \\ -4 \\ -3 \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} -3 & -4 & -2 \\ -2 & -1 & -2 \\ -1 & -3 & -2 \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} -60 \\ -40 \\ -80 \end{bmatrix},$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

Kako bi mogli provjeriti dobiveno rješenje, rješenje jednadžbe može se dobiti preko Matlabove funkcije quadprog (**Q**, **c**, **-A**, **-b**, **E**, **d**).

Matlab kod za rješavanje ovog problema nalazi se u prilogu. Rješenje ove jednadžbe dobiveno pomoću Hopfieldove neuronske mreže je

$$f(\mathbf{x}) = -76{,}6463\,,$$

u točki

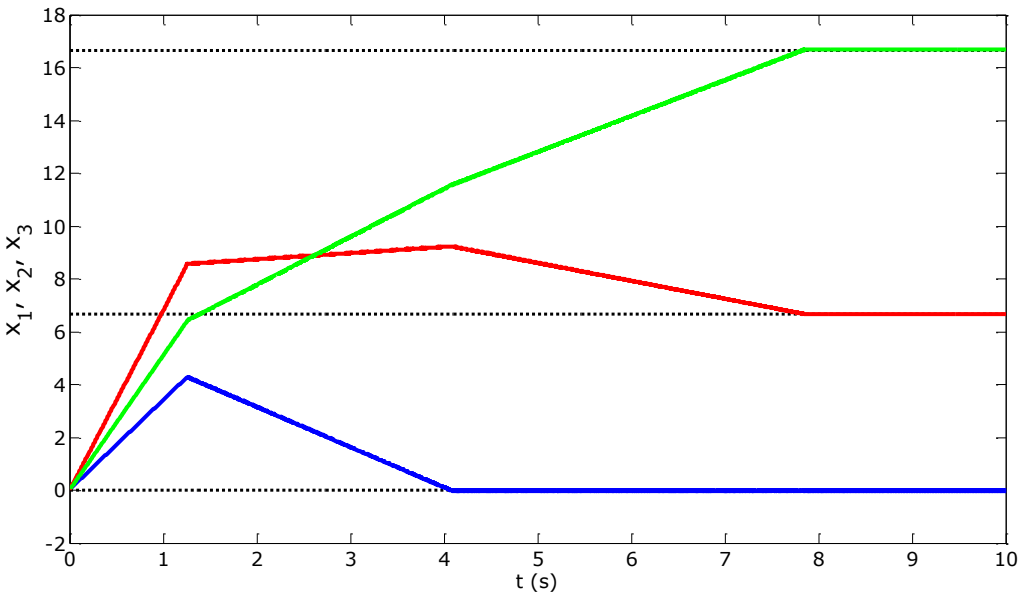
$$\mathbf{x} = \begin{bmatrix} -0{,}0255 \\ 6{,}6633 \\ 16{,}6813 \end{bmatrix},$$

a razlika između egzaktnih rješenja i rješenja dobivenih Hopfieldovom neuronskom mrežom je

$$\Delta f = 0{,}0204\,,$$

$$\Delta\mathbf{x} = \begin{bmatrix} -0{,}0255 \\ -0{,}0034 \\ 0{,}0147 \end{bmatrix}.$$

Na idućem dijagramu je prikazana konvergencija rješenja dobivenih Matlabom:



Slika 21. Konvergencija rješenja

Iz slike 21 vidljivo je da rješenje linearnog programiranja s linearnim ograničenjima konvergira u stabilna stanja.

5.5.2.2. Rješavanje problema kvadratičnog programiranja s linearnim ograničenjima

Problem kvadratičnog programiranja zadan je idućom funkcijom

$$\min f(\mathbf{x}) = \frac{11}{2}x_1^2 + \frac{9}{2}x_2^2 + 6x_3^2 + 2x_4^2 + \frac{5}{2}x_5^2 + 4x_1x_2 - 3x_1x_3 + 4x_1x_4 + x_1x_5 + x_2x_4 - x_3x_4 + 4x_2 + 2x_4 + x_5 ,$$

uz ograničenja

$$r_1(\mathbf{x}) = x_1 - 2x_2 - 3x_3 + 9 \geq 0,$$
$$r_2(\mathbf{x}) = -2x_1 + 2x_4 + 5 \geq 0,$$
$$r_3(\mathbf{x}) = -x_2 + x_3 + 5x_4 \geq 0,$$
$$r_4(\mathbf{x}) = -x_1 + 2x_2 - 2x_4 - 3 = 0,$$
$$r_5(\mathbf{x}) = x_3 + 3x_4 - 5 = 0,$$
$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad x_4 \geq 0, \quad x_5 \geq 0$$

Ovaj problem može se zapisati u matričnom obliku

$$\min f(\mathbf{x}) = \frac{1}{2}\mathbf{x^T Q x} + \mathbf{c^T x},$$
$$\mathbf{Ax} - \mathbf{b} \geq 0,$$
$$\mathbf{Ex} - \mathbf{d} = 0,$$

gdje su,

$$\mathbf{Q} = \begin{bmatrix} 11 & 4 & -3 & 4 & 1 \\ 4 & 9 & 0 & 1 & 0 \\ -3 & 0 & 12 & -3 & 0 \\ 4 & 1 & -3 & 4 & 0 \\ 1 & 0 & 0 & 0 & 5 \end{bmatrix},$$
$$\mathbf{c} = [0 \quad 4 \quad 0 \quad 2 \quad 1]^{\mathrm{T}},$$
$$\mathbf{A} = \begin{bmatrix} 1 & -2 & -3 & 0 & 0 \\ -2 & 0 & 0 & 2 & 0 \\ 0 & -1 & 1 & 5 & 0 \end{bmatrix},$$
$$\mathbf{b} = \begin{bmatrix} -9 \\ -5 \\ 0 \end{bmatrix},$$
$$\mathbf{E} = \begin{bmatrix} -1 & 2 & 0 & -2 & 0 \\ 0 & 0 & 1 & 3 & 0 \end{bmatrix},$$
$$\mathbf{d} = \begin{bmatrix} 3 \\ 5 \end{bmatrix},$$
$$\mathbf{x} = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5]^{\mathrm{T}}.$$

Kako bi mogli provjeriti dobiveno rješenje, rješenje jednadžbe može se dobiti preko Matlabove funkcije quadprog (**Q**, **c**, **-A**, **-b**, **E**, **d**).

Matlab kod za rješavanje ovog problema nalazi se u prilogu. Rješenje ove jednadžbe dobiveno pomoću Hopfieldove neuronske mreže je

$$f(\mathbf{x}) = 58{,}9491\,,$$

u točki

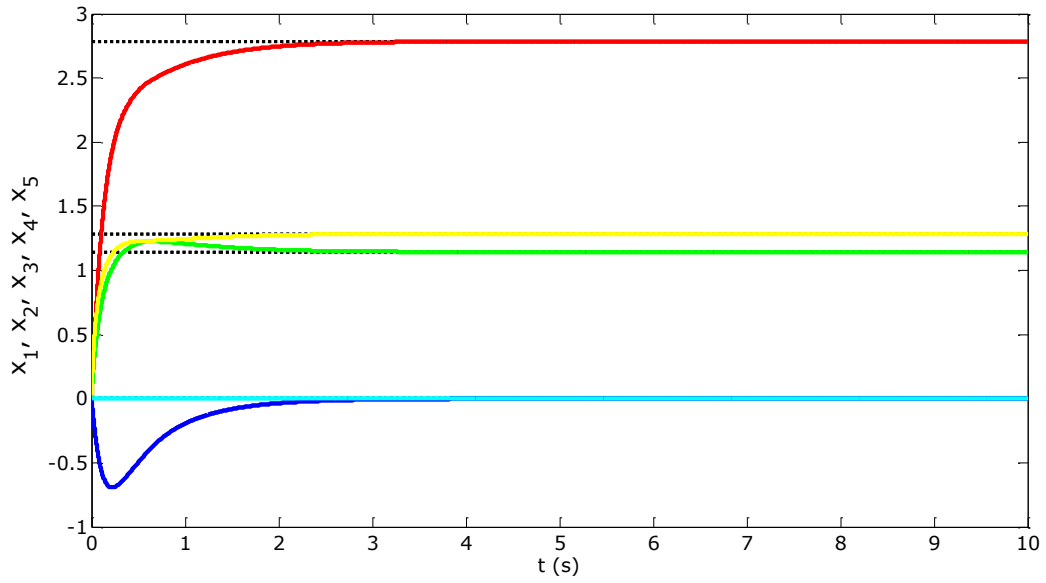
$$\mathbf{x} = \begin{bmatrix} -5{,}6186 \cdot 10^{-3} \\ 2{,}7857 \\ 1{,}1429 \\ 1{,}2857 \\ -2 \cdot 10^{-6} \end{bmatrix},$$

a razlika između egzaktnih rješenja i rješenja dobivenih Hopfieldovom neuronskom mrežom je

$$\Delta f = -0{,}0024\,,$$

$$\Delta\mathbf{x} = \begin{bmatrix} -0{,}5619 \cdot 10^{-4} \\ -0{,}6045 \cdot 10^{-4} \\ 0{,}2227 \cdot 10^{-4} \\ -0{,}1610 \cdot 10^{-4} \\ -0{,}0200 \cdot 10^{-4} \end{bmatrix}.$$

Na idućem dijagramu je prikazana konvergencija rješenja dobivenih Matlabom:



Slika 22. Konvergencija rješenja

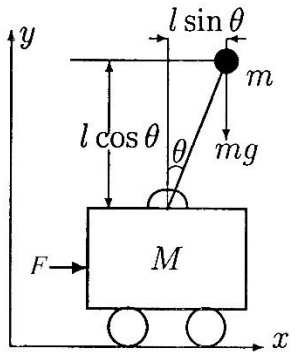
Iz slike 22 vidljivo je da rješenje kvadratičnog programiranja s linearnim ograničenjima konvergira u stabilna stanja.

6. REGULACIJA INVERZNOG NJIHALA PRIMJENOM HOPFIELDOVIH NEURONSKIH MREŽA

Inverzno njihalo je popularni eksperiment koji se koristi u edukacijske svrhe u modernoj teoriji upravljanja. Inverzno njihalo se sastoji od pokretnih kolica i njihala koje je vezano za kolica te se može slobodno rotirati. Cilj inverznog njihala je pokretanjem kolica po horizontalnoj površini dovesti njihalo u uspravni nestabilni položaj. Kako ne možemo upravljati kutnim ubrzanjem, ovo je primjer podupravljanog sustava.

6.1. Dinamički model inverznog njihala

Dinamički model kolica i njihala sastoji se od



Slika 23. Sustav kolica i njihala [8]

Prilikom modeliranja inverznog njihala sa slike 23, uzimamo u obzir standardne pretpostavke:

- štap nema masu
- masa njihala se nalazi u jednoj točki
- nema trenja

gdje je M masa kolica, m masa njihala koncentrirana u točki, θ kut zakreta koji njihalo zatvara s vertikalom i l duljina štapa. Jednadžba kretanja može se dobiti preko drugog Newtonovog zakona. [8]

Sustav se može zapisati kao:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\mathbf{q} + \mathbf{G}(\dot{\mathbf{q}}) = \boldsymbol{\tau} \tag{6.1}$$

gdje su

$$\mathbf{q} = \begin{bmatrix} x \\ \theta \end{bmatrix}, \quad \boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} M+m & ml\cos\theta \\ ml\cos\theta & ml^2 \end{bmatrix},$$

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} 0 & -ml\sin\theta\dot{\theta} \\ 0 & 0 \end{bmatrix},$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} 0 \\ -mgl\sin\theta \end{bmatrix} \quad \text{i} \quad \boldsymbol{\tau} = \begin{bmatrix} F \\ 0 \end{bmatrix}.$$

6.2. Linearizacija nelinearnog modela inverznog njihala

Kako bi mogli koristiti linearni regulator po varijablama stanja moramo najprije linearizirati model inverznog njihala. Nelinearni matematički model kolica i štapa opisan je sljedećim jednadžbama:

$$\ddot{\theta} = \frac{1}{I + L^2 m}\left[Lm(g\sin\theta - \ddot{x}\cos\theta) - c\dot{\theta}\right], \tag{6.2}$$

$$\ddot{x} = \frac{1}{M + m}\left[F - Lm\left(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right) - k\dot{x}\right]. \tag{6.3}$$

Inverzni položaj njihala je nestabilna ravnotežna pozicija (θ,$\dot{\theta}$) = (0,0). U toj točki se nalazi i ishodište prostora stanja. U blizini ravnotežnog položaja su kut θ i kutna brzina $\dot{\theta}$ jako mali. Općenito se za male kuteve θ i kutna brzine $\dot{\theta}$ vrijedi: $\sin(\theta) \approx \theta$, $\cos(\theta) \approx 1$ i $(\dot{\theta})^2\ \theta \approx 0$. Kada te pretpostavke uvrstimo u jednadžbe (6.2) i (6.3) dobivamo linearizirani matematički model oko ravnotežnog položaja:

$$\ddot{\theta} = \frac{1}{I + L^2 m}\left[Lm(g\theta - \ddot{x}) - c\dot{\theta}\right], \tag{6.4}$$

$$\ddot{x} = \frac{1}{M + m}\left[F - Lm\ddot{\theta} - k\dot{x}\right]. \tag{6.5}$$

Kako bi se ove jednadžbe mogle zapisati u prostoru stanja, $\ddot{x}$ i $\ddot{\theta}$ moraju biti prikazane diferencijalnim jednadžbama nižeg reda. Kako bi dobili diferencijalne jednadžbe nižeg reda, moramo zamijeniti $\ddot{x}$ u jednadžbi (6.4) s izrazom iz jednadžbe (6.5),a $\ddot{\theta}$ u jednadžbi (6.5) s izrazom iz jednadžbe (6.4). Tada se taj sustav jednadžbi može zapisati u lineariziranom prostoru stanja. [9]

$$\dot{\mathbf{s}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -kv_2 & \dfrac{-(Lm)^2 g v_2}{I + L^2 m} & \dfrac{Lmcv_2}{I + L^2 m} \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{Lmkv_1}{M + m} & Lmgv_1 & -cv_1 \end{bmatrix} \boldsymbol{s} + \begin{bmatrix} 0 \\ v_2 \\ 0 \\ \dfrac{-Lmv_1}{M + m} \end{bmatrix} F \tag{6.6}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{s} \tag{6.7}$$

gdje je

$$\mathbf{s} = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]^{\mathrm{T}} \tag{6.8}$$

$$\mathbf{y} = [x \quad \theta]^{\mathbf{T}} \tag{6.9}$$

$$v_1 = \frac{M + m}{I(M + m) + L^2 mM} \quad \text{i} \quad v_2 = \frac{I + L^2 m}{I(M + m) + L^2 mM} \tag{6.10}$$

6.3. Sinteza linearnog regulatora

Sustav u prostoru stanja možemo zapisati

$$\dot{\mathbf{s}}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{b}u(t), \tag{6.11}$$

nakon što ponovimo postupak koji je opisan jednadžbama (4.20) – (4.23) i dobijemo linearnu matričnu jednadžbu koja se može zapisati kao

$$\mathbf{AP} - \mathbf{P\Lambda} = \mathbf{b}\hat{\mathbf{K}}, \tag{6.12}$$

gdje je

$$\mathbf{K} = \hat{\mathbf{K}}\mathbf{P}^{-1}. \tag{6.13}$$

Izraz (6.12) predstavlja linearnu matričnu jednadžbu po nepoznatoj matrici **P** dok su matrice **A**, **Λ** i $\hat{\mathbf{K}}$ i vektor **b** unaprijed zadani. Matrica **A** i vektor **b** određeni su karakteristikama sustava, dok dijagonalna matrica **Λ** sadrži željene svojstvene vrijednosti zatvorenog regulacijskog kruga, a matrica $\hat{\mathbf{K}}$ je proizvoljna konstantna matrica. Na kraju, matricu pojačanja **K** regulatora stanja dobivamo na osnovu jednadžbe (6.13).

Matrica **P** kod ovog regulatora dobiva se Hopfieldovom neuronskom mrežom čiji je algoritam za učenje izveden formulama (5.39)-(5.46)

$$\dot{\mathbf{X}} = -\mu[\mathbf{A}^{\mathrm{T}}(\mathbf{AP} - \mathbf{P\Lambda} - \mathbf{B}\hat{\mathbf{K}}) - (\mathbf{AP} - \mathbf{P\Lambda} - \mathbf{B}\hat{\mathbf{K}})\mathbf{\Lambda}^{\mathrm{T}}] \tag{6.14}$$

6.4. Simulacijski rezultati inverznog njihala

Simulirani sustav ima iduće parametre:

M = 1 kg

m = 0,5 kg

L = 1 m

I = 0,1 m^4

k = 0,05

c = 0,01

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0{,}0462 & -3{,}7731 & 0{,}0077 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}0385 & 11{,}3192 & -0{,}0231 \end{bmatrix},$$

$$\mathbf{b} = [0 \quad 0{,}9231 \quad 0 \quad -0{,}7692]^{\mathrm{T}},$$

$$\mathbf{\Lambda} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1{,}2 & 0 & 0 \\ 0 & 0 & -1{,}4 & 0 \\ 0 & 0 & 0 & -1{,}6 \end{bmatrix},$$

Matrica P dobivena Hopfieldovom neuronskom mrežom je

$$\mathbf{P} = \begin{bmatrix} -0{,}6634 & -0{,}4489 & -0{,}3198 & -0{,}2356 \\ 0{,}6632 & 0{,}5386 & 0{,}4478 & 0{,}377 \\ -0{,}0768 & -0{,}0797 & -0{,}0837 & -0{,}0891 \\ 0{,}0768 & 0{,}0957 & 0{,}1172 & 0{,}1426 \end{bmatrix},$$

a pojačanje je

$$\mathbf{K} = [-0{,}3562 \quad -1{,}1808 \quad -28{,}2171 \quad -8{,}087].$$

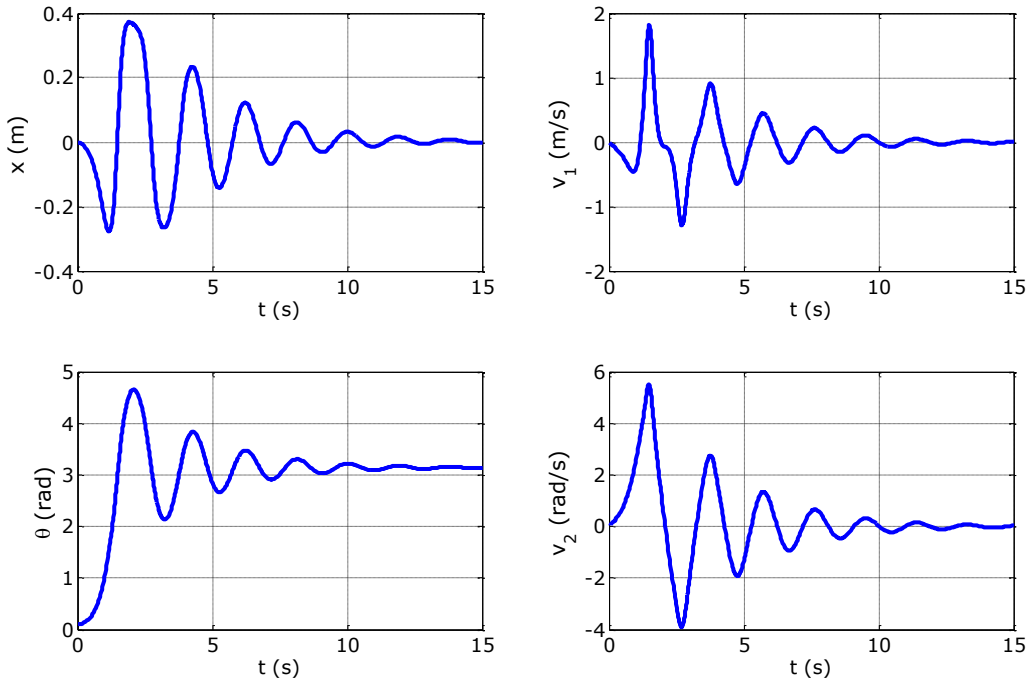
Razlika između egzaktnog pojačanja i pojačanja dobivenog pomoću Hopfieldove neuronske mreže je

$$\Delta\mathbf{K} = \begin{bmatrix} -0{,}0124 \cdot 10^{-9} \\ -0{,}0269 \cdot 10^{-9} \\ -0{,}2050 \cdot 10^{-9} \\ -0{,}0776 \cdot 10^{-9} \end{bmatrix}^{\mathrm{T}}.$$

Dobiveno pojačanje **K** koristit ćemo u idućim primjerima. Matlab kod za dobivena rješenja i dijagrame koji slijede nalazi se u prilogu.

6.4.1. Inverzno njihalo bez upravljanja

U ovom primjeru, njihalo je otklonjeno 0.1 rad od ravnotežnog položaja i nema upravljanja ($\theta(0)$=0,1 rad; u=0)



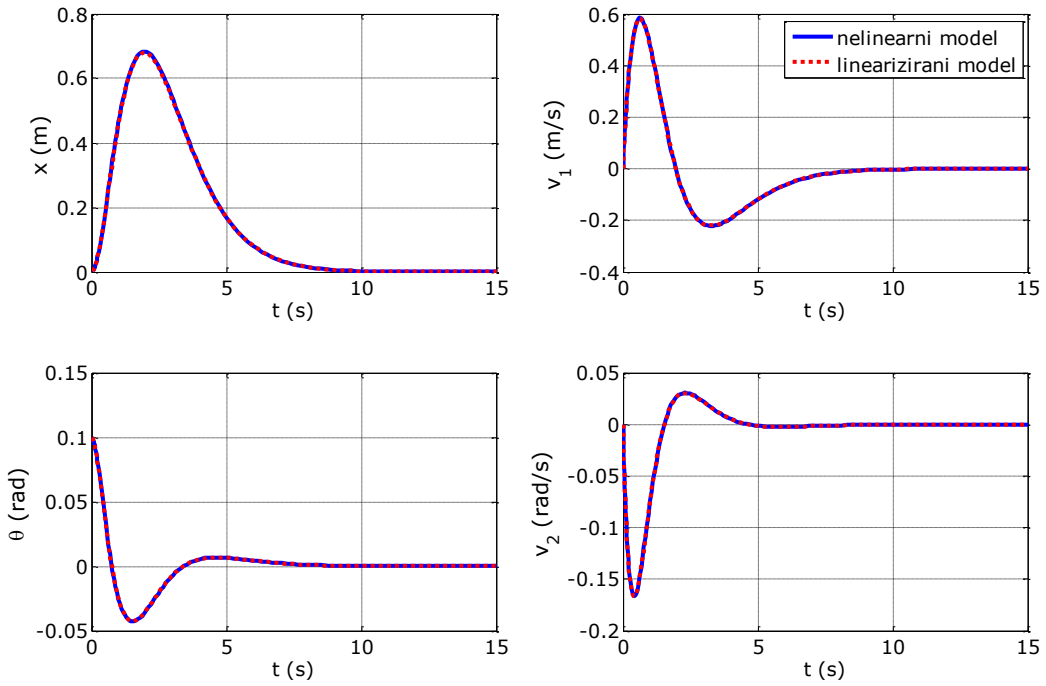
Slika 24. Provjera nelinearnog modela

Sa slike 24 vidljivo je da se njihalo uslijed trenja zaustavi u donjem stabilnom položaju.

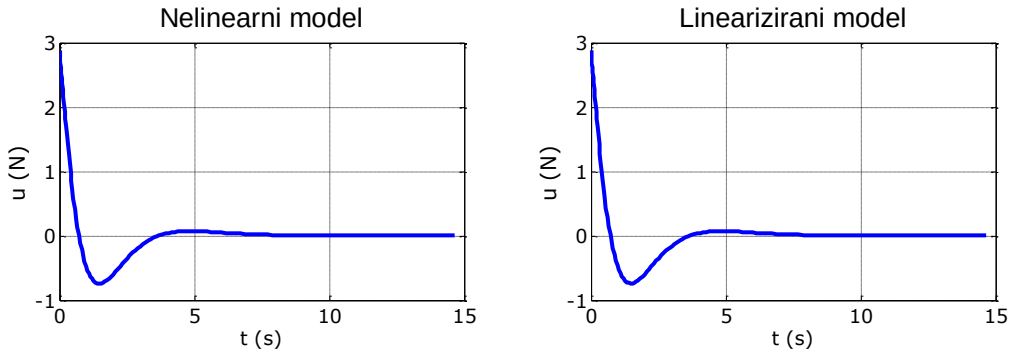
6.4.2. Inverzno njihalo s linearnim regulatorom

Linearni regulator korišten kod inverznog njihala je $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$. Kod inverznog njihala s linearnim regulatorom imamo više slučajeva. Započet ćemo s malim otklonom njihala od ravnotežnog položaja, te ćemo nastaviti s povećanjem otklona njihala.

Otklon njihala od ravnotežnog položaja: $\theta(0)=0{,}1$ rad



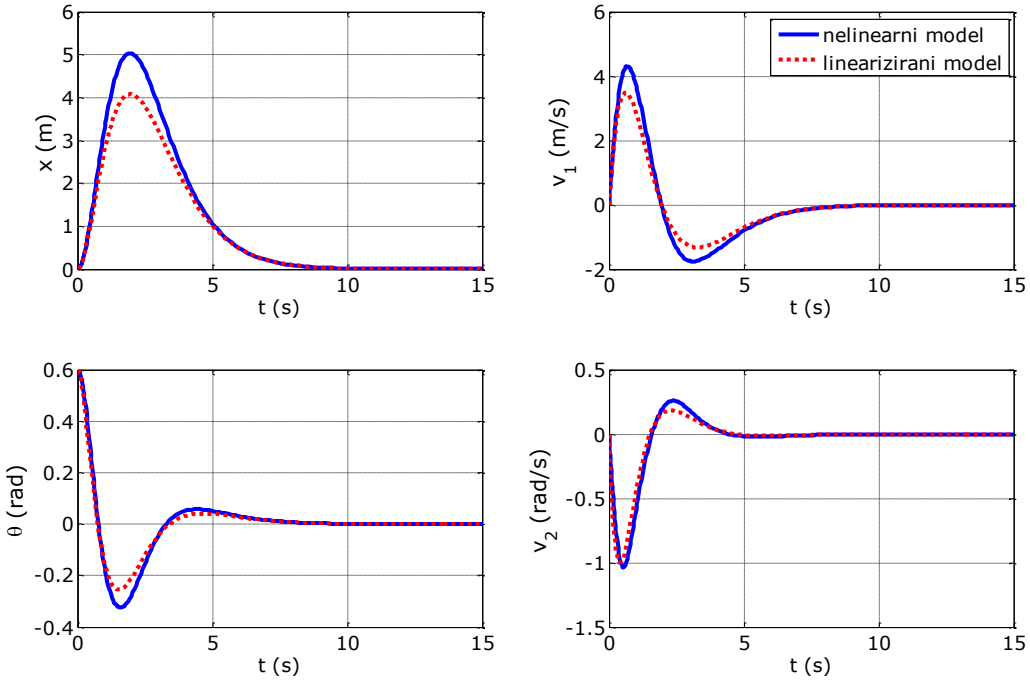
Slika 25. Odziv nelinearnog i linearizirano g modela za $\theta(0)=0{,}1$



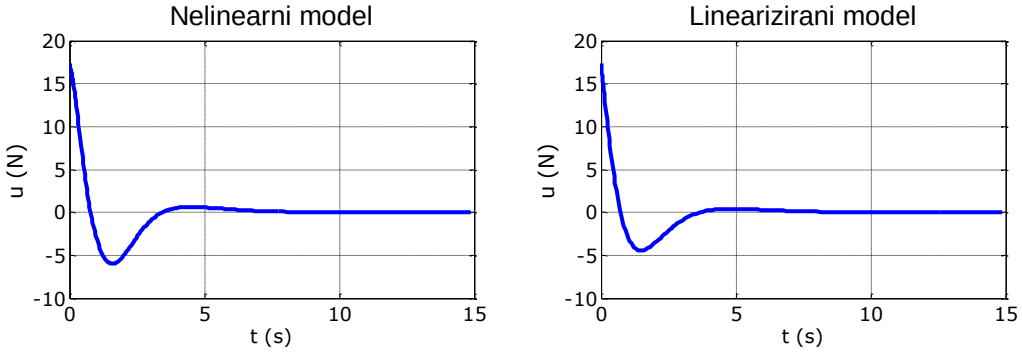
Slika 26. Upravljačke varijable linearizirano g i nelinearnog modela za $\theta(0)=0{,}6$

Sa slika 25 i 26 može se vidjeti da nema razlike između linearizirano g i nelinearnog modela.

Otklon njihala od ravnotežnog položaja: $\theta(0)=0{,}6$ rad



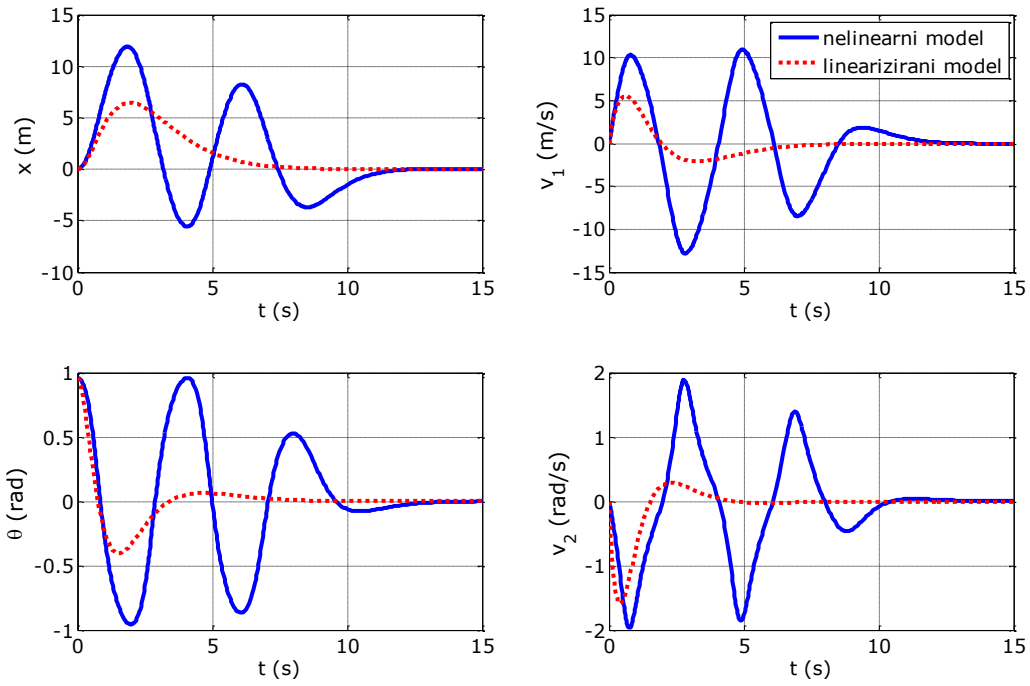
Slika 27. Odziv nelinearnog i linearizirano g modela za $\theta(0)=0{,}6$



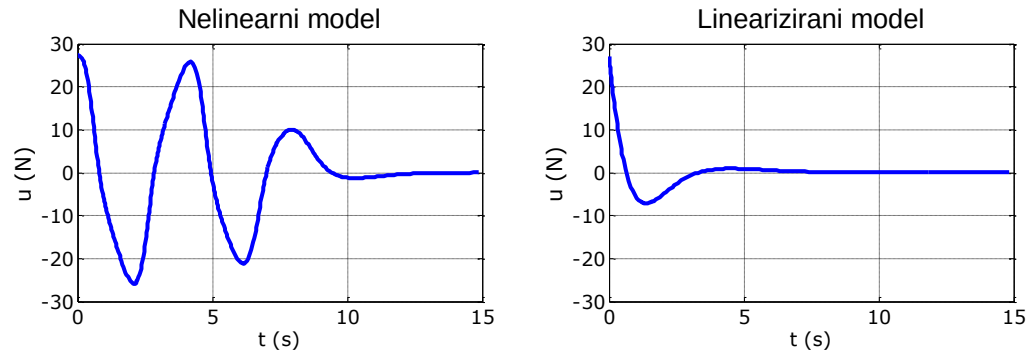
Slika 28. Upravljačke varijable lineariziranog i nelinearnog modela za $\theta(0)=0{,}6$

Iz slika 27 i 28 se vidi da povećanjem kuta otklona njihala nelinearni i linearizirani model počinju odstupati. Nelinearni model može se i dalje upravljati pomoću linearnog regulatora.

Otklon njihala od ravnotežnog položaja: $\theta(0)=0{,}96$ rad



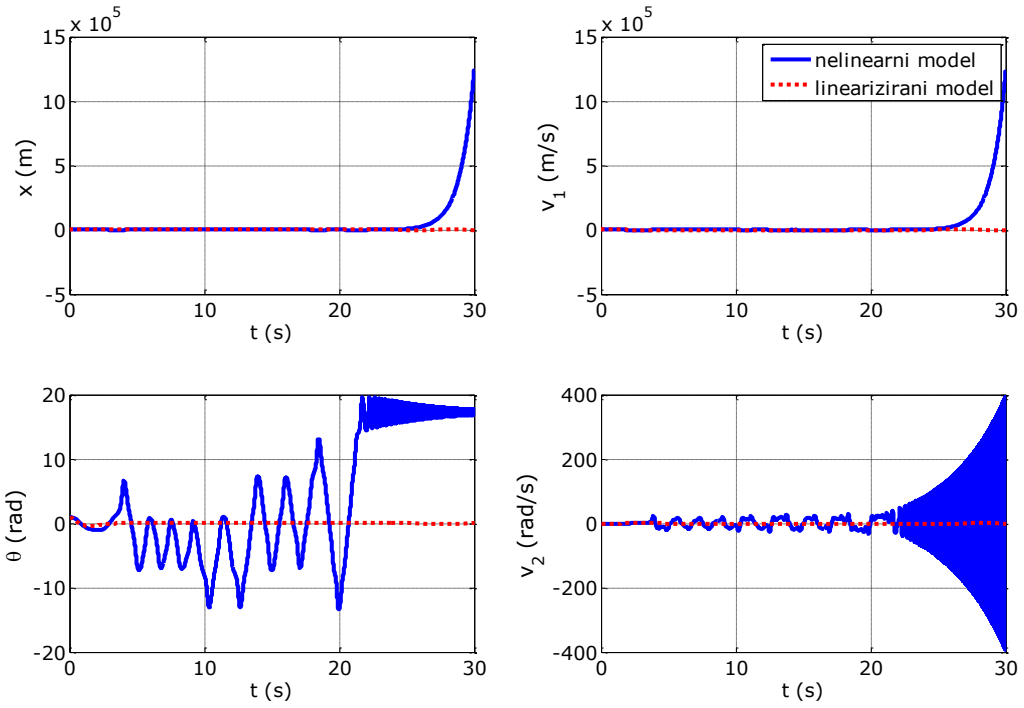
Slika 29. Odziv nelinearnog i linearizirano modela za $\theta(0)=0{,}96$



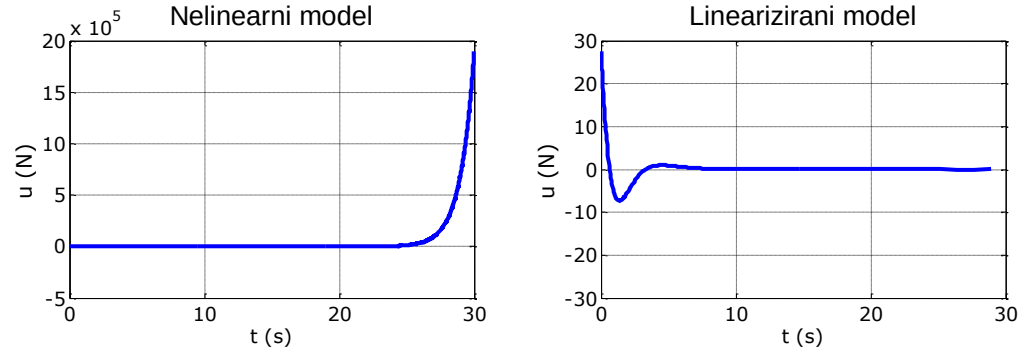
Slika 30. Upravljačke varijable lineariziranog i nelinearnog modela za $\theta(0)=0{,}96$

Iz slika 29 i 30 je vidljivo da se nelinearni model ne poklapa s linearizirani modelom, više oscilira oko ravnotežnog položaja i potrebno mu je više vremena da se stabilizira.

Otklon njihala od ravnotežnog položaja: $\theta(0)=0{,}98$ rad



Slika 31. Odziv nelinearnog i linearizirranog modela za $\theta(0)=0{,}98$



Slika 32. Upravljačke varijable linearizirranog i nelinearnog modela za $\theta(0)=0{,}98$

Iz slika 31 i 32 je vidljivo da upravljanje nelinearnog modela linearnim regulatorom nije moguće nakon 0.96 rad. Linearizirani model se izvodio za slučajeve u blizini ravnotežnog položaja te su se uvele pretpostavke za male kuteve i male brzine. Čim se odmaknemo od ravnotežnog položaja te pretpostavke više ne vrijede, te ne možemo koristiti linearizirani model, a ne možemo ni upravljati nelinearni model s linearnim regulatorom. Za upravljanje inverznim njihalom kod većih kuteva koriste se tako zvani „swing-up“ regulatori.

6.4.3. Inverzno njihalo sa „swing-up“ regulatorom

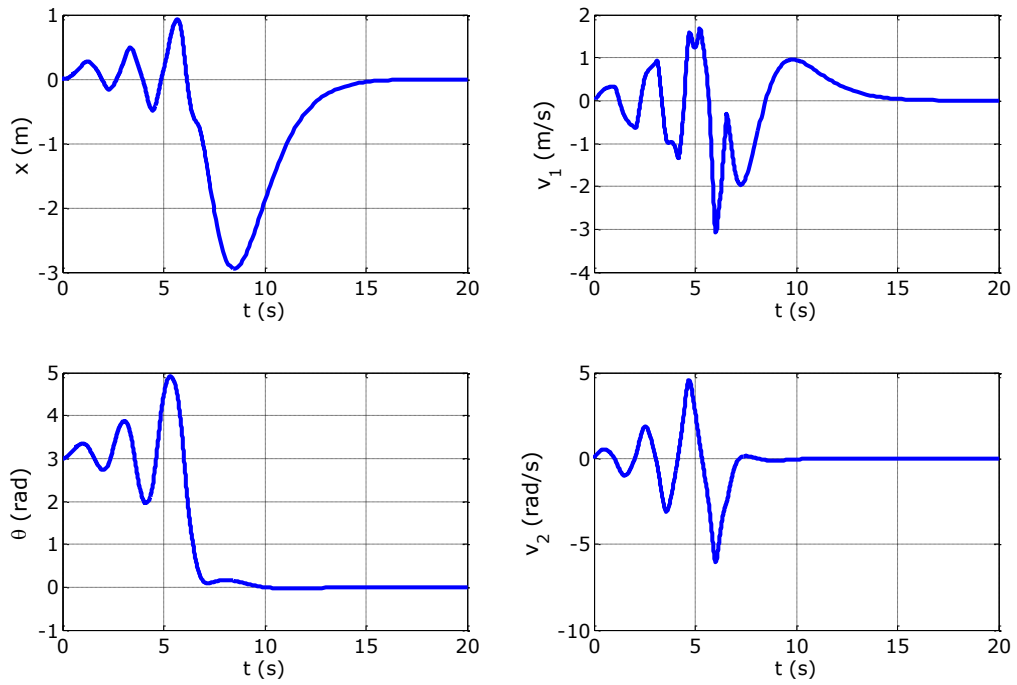
U primjerima s linearnim regulatorom ($\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$) mogli smo upravljati sve slučajeve koji se nalaze u blizi nestabilne ravnotežne točke. Kako smo odmicali od te točke nelinearni regulator je sve teže postizao stabilnost, do točke kada je sustav bio neupravljiv s linearnim regulatorom.

Kako bi mogli upravljati s inverznim njihalom gdje je početni uvjet izvan granica upravljivosti, s linearnim regulatorom, moramo koristiti tako zvane „swing-up“ regulatore. U idućim primjerima je prikazana regulacija s dvije verzije „swing-up“ regulatora i verzijom „swing-up“ regulatora koji se zasniva na energiji. U svim primjerima su korišteni „swing-up“ regulatori do trenutka kada je inverzno njihalo u položaju kada se može upravljati s linearnim regulatorom.

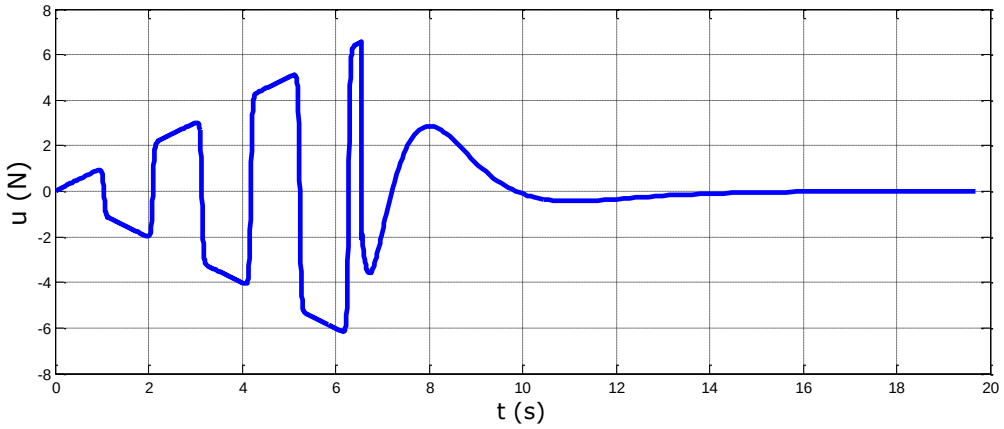
Otklon njihala od ravnotežnog položaja: $\theta(0)=3$ rad

Upravljačka varijabla 1. verzije „swing-up“ regulatora prikazana je idućom jednadžbom:

$$u(t)=t\cdot\tanh(10\cdot\sin(3\cdot t))$$



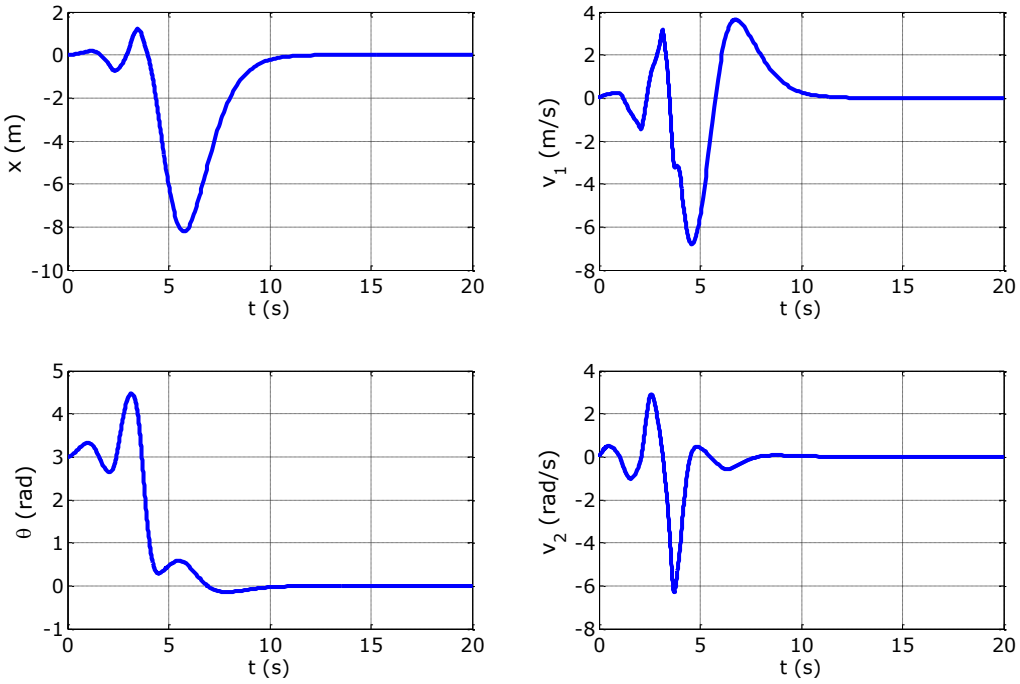
Slika 33. Upravljanje nelinearnog modela pomoću 1. verzije „swing-up“ regulatora



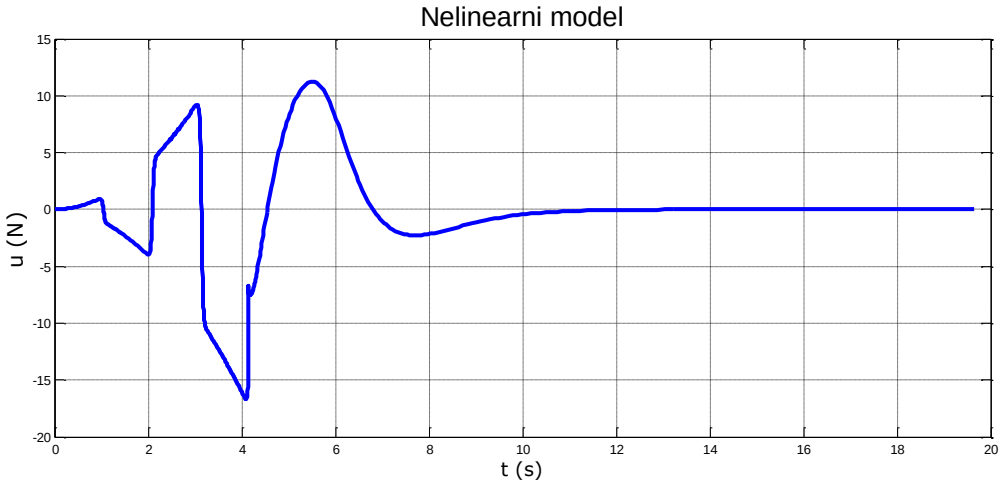
Slika 34. Upravljačka varijabla 1. verzije „swing-up“ regulatora

Upravljačka varijabla 2. verzije „swing-up“ regulatora prikazana je idućom jednadžbom:

$$u(t)=t^2\cdot\tanh(10\cdot\sin(3\cdot t))$$



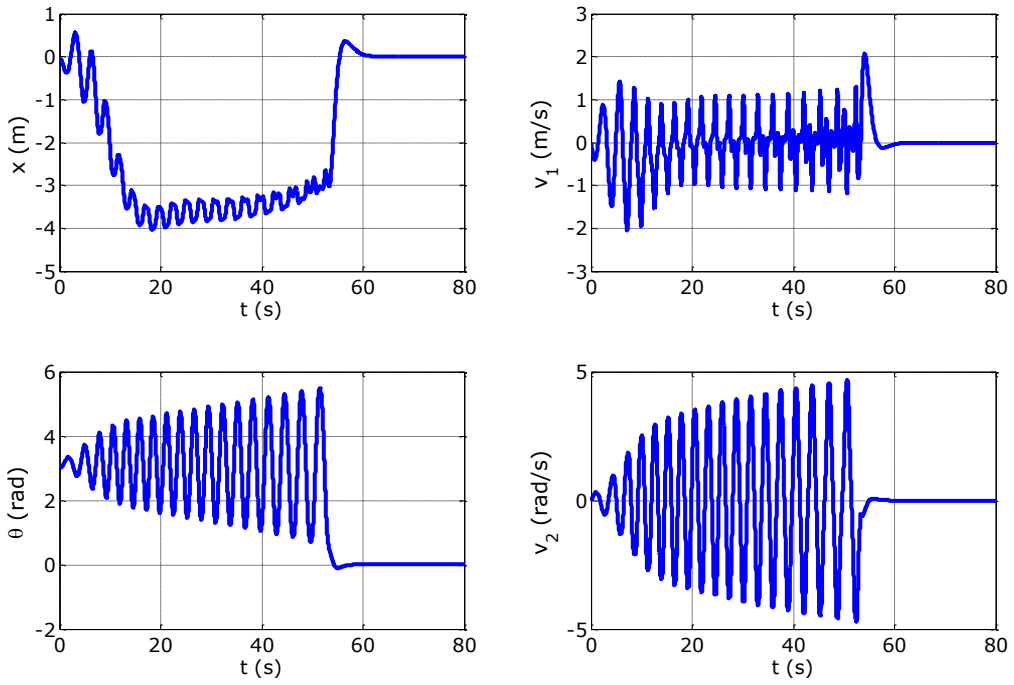
Slika 35. Upravljanje nelinearnog modela pomoću 2. verzije „swing-up“ regulatora



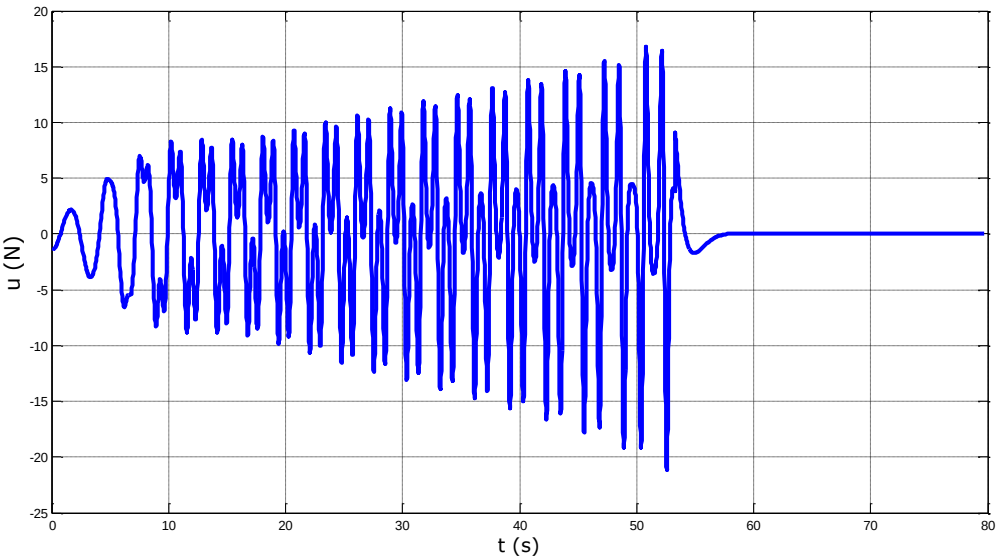
Slika 36. Upravljačka varijabla 2. verzije „swing-up“ regulatora

Upravljačka varijabla „swing-up“ regulatora zasnovanog na energiji prikazana je idućom jednadžbom:

$$u(t) = \frac{k_v \sin\theta\left(g\cos\theta - \dot{\theta}^2\right) - (1+\sin^2\theta)(k_x x + k_{dx}\dot{x})}{k_v + (1+\sin^2\theta)k_E E}$$



Slika 37. Upravljanje nelinearnog modela pomoću „swing-up“ regulatora zasnovanog na energiji



Slika 38. Upravljačka varijabla „swing-up“ regulatora zasnovanog na energiji

Iz slika 33-38 može se vidjeti da 1. i 2. verzija „swing-up“ regulatora postižu nestabilni ravnotežni položaj inverznog njihala između 10 i 15 sekundi dok „swing-up“ regulator koji se zasniva na energiji dolazi u ravnotežni položaj nakon 60 sekundi.

7. ZAKLJUČAK

U ovom radu je korištena je Hopfieldova neuronska mreža kod rješavanja raznih matematičkih problema kao što su: inverz matrice, Lyapunovljeve matrične jednadžbe, sinteza regulatora metodom podešavanja polova. Za svaki od tih problema moguće je dobiti funkciju energije čijom se minimizacijom dobije rješenje sustava. Minimizacija funkcije energije se odvija pomoću gradijentnog algoritma. Na taj se način matematički problemi pretvore u optimizacijske probleme.

U sklopu rada izvedeni su algoritmi za rješavanje tih matematičkih problema te njihova primjena kod sinteze regulatora inverznog njihala. Zatim su simulirani linearizirani i nelinearni model inverznog njihala pri različitim početnim uvjetima, koristeći linearni regulator čije je pojačanje dobiveno Hopfieldovom neuronskom mrežom. Linearni regulator nije imao problema prilikom upravljanja inverznim njihalom u blizini inverznog položaja. Ako se odmaknemo dovoljno od ravnotežnog položaja, linearni regulator ne daje zadovoljavajuće rezultate. Korištenjem tako zvanih „swing-up“ regulatora moguće je upravljati inverznim njihalom bez obzira na njegov početni položaj.

LITERATURA

[1] Novaković, B., Majetić, D., Široki, M.: Umjetne neuronske mreže, Fakultet strojarstva i brodogradnje, Zagreb, 1998.

[2] Cichocki, A., Unbehauen, R.: Neural Networks for Optimization and Signal Processing, Wiley, New York, NY, USA, 1993.

[3] Kasać, J.: Opća teorija sustava, Fakultet strojarstva i brodogradnje, Zagreb, 2007.

[4] Bašić, B. D., Čupić, M., Šnajder, J.: Umjetne neuronske mreže, Fakultet elektrotehnike i računarstva, Zagreb, 2008.

[5] Imperial College London: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/ds12/article1.html (05.02.2017.)

[6] Wikipedija: https://hr.wikipedia.org/wiki/Umjetna_neuronska_mre%C5%BEa (05.02.2017.)

[7] PennState Eberly College of Science: https://online.science.psu.edu/bisc004_activewd001/node/1907 (05.02.2017.)

[8] Lozano, R., Fantoni, I., Block, D. J.: Stabilization of the inverted pendulum around its homoclinic orbit, System & Control Letters, 40 (3), 197-204, 2000.

[9] Bugeja, M.: Non-linear swing-up and stabilizing control of an Inverted Pendulum system, EUROCON 2003. Computer as a Tool., Ljubljana, Slovenija, 22-24 Sept. 2003, pp. 437-441.

PRILOZI

I. CD-R disc

II. Matlab kodovi:

Rješavanje sustava linearnih jednadžbi:

```
% Numeričko rješenje sustava linearnih jednadžbi: Ax=b
clear
clc

N=500;      % broj iteracija gradijentnog algoritma
T=10;       % vrijeme ucenja kod kontinuirane verzije
h=T/N;      % period sempliranja kod kontinuirane verzije
eta=0.05;   % learning-rate

%**** numerika:
A = [4 2 1; 2 -3 2; 3 2 -1]
b=[13; 7; 5];

stX=A\b     % A\b = inv(A)*b
X=zeros(3,1);
Xegz=inv(A)*b;

for i=1:N,
    t(i)=h*i;
    E = A*X - b;
    X = X - eta*(A'*E);
    pl(i)= norm(E,'fro');
    Xx(i,:,:) = X;
end

X       % ispis riješenja sustava jednadžbi
del_X=X-Xegz

%------------------------------
oo=ones(size(t));
st1X=stX(1,:);
WW1=kron(oo',st1X);
st2X=stX(2,:);
WW2=kron(oo',st2X);
st3X=stX(3,:);
WW3=kron(oo',st3X);
%------------------------------
% norme pogreške u normalnom mjerilu
hFig=figure(1); set(hFig, 'Position', [10 50 1200 750])
subplot(121), plot(t,pl, 'linewidth',3),
ylabel('|| x - A^{-1}b
||','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
```

```
set(gca,'fontsize',14,'fontname','Verdana');
% norme pogreške u logaritamskom mjerilu
subplot(122), semilogy(t,pl, 'linewidth',3)
ylabel('|| x - A^{-1}b
||','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');

% x1(t)
hFig=figure(2); set(hFig, 'Position', [10 50 1200 750])
subplot(131), plot(t, WW1, 'k:', 'linewidth',2), hold on,
subplot(131), plot(t, Xx(:,1,1), 'b', 'linewidth',3),
ylabel('x_{1}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
axis([0 6 1.5 4.5]);
set(gca,'fontsize',14,'fontname','Verdana');
% x2(t)
subplot(132), plot(t, WW2, 'k:', 'linewidth',2), hold on,
subplot(132), plot(t, Xx(:,2,1), 'b', 'linewidth',3),
ylabel('x_{2}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
axis([0 6 -0.5 1.05]);
set(gca,'fontsize',14,'fontname','Verdana');
% x3(t)
subplot(133), plot(t, WW3, 'k:', 'linewidth',2), hold on,
subplot(133), plot(t, Xx(:,3,1), 'b', 'linewidth',3),
ylabel('x_{3}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
axis([0 6 1 3.05]);
set(gca,'fontsize',14,'fontname','Verdana');
```

Invertiranje matrice:

```
% Numeričko rješenje inverzne matrice: AX=I
clear
clc

N=20000;      % broj iteracija gradijentnog algoritma
T=20;         % vrijeme ucenja kod kontinuirane verzije
h=T/N;        % period sempliranja kod kontinuirane verzije
eta=0.014;    % learning-rate
mu =7;        % mu = eta/h

%**** numerika:
A = [1 4 6; 2 2 -4; -2 1 9]
invA=inv(A)
I=eye(3);
X=ones(3,3);
```

```
for i=1:N,
    t(i)=h*i;
    E = A*X - I;
    X = X - eta*(A'*E);
    pl(i)= norm(E,'fro');
    Xx(i,:,:) = X;
end
X       % invertirana matrica
del_inv = X-invA    % razlika između egzaktne i inverza
                        matrice dobivene mrežom

%-----------------------------
oo=ones(size(t));
st1A=invA(1,:);
WW1=kron(oo',st1A);
st2A=invA(2,:);
WW2=kron(oo',st2A);
st3A=invA(3,:);
WW3=kron(oo',st3A);
%-----------------------------

% norme pogreške u normalnom mjerilu
hFig=figure(1); set(hFig, 'Position', [10 50 1200 750])
subplot(121), plot(t,pl, 'linewidth',3),
ylabel('|| X - A^{-1} ||','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
% norme pogreške u logaritamskom mjerilu
subplot(122), semilogy(t,pl, 'linewidth',3)
ylabel('|| X - A^{-1} ||','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');

% x11,x12,x13
hFig=figure(2); set(hFig, 'Position', [10 50 1200 750])
subplot(131), plot(t, WW1, 'k:', 'linewidth',2.5), hold on,
subplot(131), plot(t, Xx(:,1,1), 'b', t, Xx(:,1,2), 'r', t,
Xx(:,1,3), 'g', 'linewidth',3),
ylabel('x_{11}, x_{12},
x_{13}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
axis([0 20 -2 1.5])
set(gca,'fontsize',14,'fontname','Verdana');
% x21,x22,x23
subplot(132), plot(t, WW2, 'k:', 'linewidth',2.5), hold on,
subplot(132), plot(t, Xx(:,2,1), 'b', t, Xx(:,2,2), 'r', t,
Xx(:,2,3), 'g', 'linewidth',3),
ylabel('x_{21}, x_{22},
x_{23}','FontSize',18,'FontName','Verdana'),
```

```
xlabel('t (s)','FontSize',14,'FontName','Verdana');
axis([0 20 -0.6 1.2])
set(gca,'fontsize',14,'fontname','Verdana');
% x31,x32,x33
subplot(133), plot(t, WW3, 'k:', 'linewidth',2.5), hold on,
subplot(133), plot(t, Xx(:,3,1), 'b', t, Xx(:,3,2), 'r', t,
Xx(:,3,3), 'g', 'linewidth',3),
ylabel('x_{31}, x_{32},
x_{33}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
axis([0 20 -0.6 0.5])
set(gca,'fontsize',14,'fontname','Verdana');
```

Rješavanje Lyapunovljeve jednadžbe:

```
% Numeričko rješenje Lyapunovljeve jednadzbe:A'*P + P*A + Q =0
clear all
close all
clc

N=10000;          % broj iteracija gradijentnog algoritma
T=10;             % vrijeme ucenja kod kontinuirane verzije
h=T/N;            % period uzorkovanja kod kontinuirane verzije
eta=0.03;         % learning-rate

A = [0 1 0; 0 0 1; -1 -3 -3];
Q = [1 0 0; 0 1 0; 0 0 1];              % zadani Q>0

P2=lyap(A',Q)         % P_{egzaktni}
P=zeros(3,3);         % inicijalni P (moze biti rand)

for i=1:N,
    t(i)=h*i;

    E = A'*P + P*A + Q;             % error
    P = P - eta*(A*E + E*A');       % gradijentni algoritam
    Pp(i,:,:) = P;

    nE(i)= norm(E,'fro');           % Frobeniusova norma
    dP(i) = norm(P-P2,'fro');
end

P                     % ispis rješenja Lyapunovljeve jednadžbe
eigP=eig(P)           % direktna provjera pozitivne definitnosti
%
```

```
%-----------------------------
oo=ones(size(t));
st1A=P2(1,:);
WW1=kron(oo',st1A);
st2A=P2(2,:);
WW2=kron(oo',st2A);
st3A=P2(3,:);
WW3=kron(oo',st3A);
%-----------------------------

hFig=figure(2); set(hFig, 'Position', [10 50 1200 750])
subplot(131), plot(t, WW1, 'k:', 'linewidth',2.5), hold on,
subplot(131), plot(t, Pp(:,1,1), 'b', t, Pp(:,1,2), 'r', t,
Pp(:,1,3), 'g', 'linewidth',3),
ylabel('P_{11}, P_{12},
P_{13}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(132), plot(t, WW2, 'k:', 'linewidth',2.5), hold on,
subplot(132), plot(t, Pp(:,2,1), 'b', t, Pp(:,2,2), 'r', t,
Pp(:,2,3), 'g', 'linewidth',3),
ylabel('P_{21}, P_{22},
P_{23}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(133), plot(t, WW3, 'k:', 'linewidth',2.5), hold on,
subplot(133), plot(t, Pp(:,3,1), 'b', t, Pp(:,3,2), 'r', t,
Pp(:,3,3), 'g', 'linewidth',3),
ylabel('P_{31}, P_{32},
P_{33}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%%%

hFig=figure(1); set(hFig, 'Position', [10 50 1400 900])
subplot(221), plot(t,nE, 'linewidth',3),
ylabel('|| A^T*P + P*A + Q
||','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(222), semilogy(t,nE, 'linewidth',3)
ylabel('|| A^T*P + P*A + Q
||','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(223), plot(t,dP, 'linewidth',3)
```

```
ylabel('|| P_{iterativni} -
P_{egzaktni}||','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(224), semilogy(t,dP, 'linewidth',3)
ylabel('|| P_{egzaktni} - P_{iterativni}
||','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
```

Rješavanje jednadžbe za podešavanje polova:

```
% Numeričko rješenje matricne jednadzbe podesavanja polova:
% A*P - P*L = B*Kk
% K=Kk*inv(P)
clear all
close all
clc

% primjer 1 (m=1):
% A = [0 1 0; 0 0 1; 1 3 3];
% B = [0; 0; 1];
% etainv=2.3;
% N=80000;          % broj iteracija gradijentnog algoritma

% % primjer 2 (m=2):
A = [0 1 0; 0 -2 1; 1 3 3];
B = [0 0; 1 0; 0 1];
etainv=0.2;
N=10000;          % broj iteracija gradijentnog algoritma

T=10;             % vrijeme ucenja kod kontinuirane verzije
h=T/N;            % period sempliranja kod kontinuirane verzije
eta=0.03;         % learning-rate

Kk = ones(size(B))'      % Proizvoljno zadana matrica K_kapa
Q = B*Kk;                % zadani Q>0
pol=[-1 -2 -3]           % vektor zeljenih polova sustava
L = diag(pol);           % matrica s zeljenim polovima na
dijagonali
eigA=eig(A)              % direktna provjera stabilnosti
otvorenog sustava
P=zeros(size(A));        % inicijalni P (moze biti rand)
P2=lyap(A,-L,-B*Kk)      % P dobiven matlabom
Pinv=zeros(size(P));
I=eye(size(P));
Pinv2=Pinv;
for i=1:N,
    t(i)=h*i;
```

```
    E = A*P - P*L - B*Kk;        % error1
    P = P - eta*(A'*E - E*L');   % gradijentni algoritam
    Pp(i,:,:) = P;
    Einv = P*Pinv - I;
    Pinv = Pinv - etainv*(P'*Einv);
    pl(i)= norm(Einv,'fro');
    nE(i)= norm(E,'fro');        % Frobeniusova norma
end

for i=1:N,
    Einv2 = P*Pinv2 - I;
    Pinv2 = Pinv2 - etainv*(P'*Einv2);
    pl2(i)= norm(Einv2,'fro');
end

P
eigP=eig(P)          % direktna provjera pozitivne definitnosti
K=Kk*Pinv          % matrica pojacanja
Ar = A - B*K;        % Matrica A zatvorenog kruga
eAr = eig(Ar)
polPog=eig(A-B*K)
Kplace = place(A,B,pol) % Matlabova funkcija
error_K = K - Kplace    % usporedba Hopfield i Matlab (error_K
= 0 -> samo za sustave s jednim ulazom, m=1)
error_p = pol - eAr'    % usporedba zeljenih polova i polova
zatvorenog kruga (error_p = 0 -> uvijek)
delP=P-P2
%------------------------------
oo=ones(size(t));
st1P=P2(1,:);
WW1=kron(oo',st1P);
st2P=P2(2,:);
WW2=kron(oo',st2P);
st3P=P2(3,:);
WW3=kron(oo',st3P);
%------------------------------

hFig=figure(1); set(hFig, 'Position', [10 50 1400 900])
subplot(121), plot(t,nE, 'linewidth',3),
ylabel('|| E ||','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(122), semilogy(t,nE, 'linewidth',3)
ylabel('|| E ||','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');
%
hFig=figure(2); set(hFig, 'Position', [10 50 1200 750])
subplot(131), plot(t, WW1, 'k:', 'linewidth',2.5), hold on,
```

```
subplot(131), plot(t, Pp(:,1,1), 'b', t, Pp(:,1,2), 'r', t,
Pp(:,1,3), 'g', 'linewidth',3),
ylabel('P_{11}, P_{12},
P_{13}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%axis([0 0.5 0 0.55])
axis([0 5 -0.6 0.9])
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(132), plot(t, WW2, 'k:', 'linewidth',2.5), hold on,
subplot(132), plot(t, Pp(:,2,1), 'b', t, Pp(:,2,2), 'r', t,
Pp(:,2,3), 'g', 'linewidth',3),
ylabel('P_{21}, P_{22},
P_{23}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%axis([0 0.5 -0.55 0.1])
axis([0 5 -2 1.8])
set(gca,'fontsize',14,'fontname','Verdana');
%
subplot(133), plot(t, WW3, 'k:', 'linewidth',2.5), hold on,
subplot(133), plot(t, Pp(:,3,1), 'b', t, Pp(:,3,2), 'r', t,
Pp(:,3,3), 'g', 'linewidth',3),
ylabel('P_{31}, P_{32},
P_{33}','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%axis([0 0.5 0.1 0.55])
axis([0 5 -0.7 1.2])
set(gca,'fontsize',14,'fontname','Verdana');

% norme pogreške u normalnom mjerilu
hFig=figure(3); set(hFig, 'Position', [10 50 1200 750])
subplot(121), plot(t,pl, 'linewidth',3), hold on
subplot(121), plot(t,pl2, 'r','linewidth',3),
ylabel('|| P_{inv} - P^{-1}
||','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%axis([0 0.5 0 1.8])
axis([0 2 0 1.8])
set(gca,'fontsize',14,'fontname','Verdana');
% norme pogreške u logaritamskom mjerilu
subplot(122), semilogy(t,pl, 'linewidth',3), hold on
subplot(122), semilogy(t,pl2, 'r','linewidth',3)
ylabel('|| P_{inv} - P^{-1}
||','FontSize',18,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%axis([0 0.5 0.1 10])
axis([0 2 0.01 10])
set(gca,'fontsize',14,'fontname','Verdana');
```

Rješavanje problema linearnog i kvadratičnog programiranja s linearnim ograničenjima:

```
% Rjesavanje problema kvadratičnog programiranja
% Potrebno je minimizirati funkciju
%   f(x)=1/2*(x'*Q*x)+c'*x
% uz ograničenja:
%   A*x-b>=0
%   E*x-d=0

clc
clear all

N=10000;
T=10;
h=T/N;

A=[];
b=[];
E=[];
d=[];
xGornjiLimit=[];
xDonjiLimit=[];

% 1. primjer
kapa=80;
eta=0.0017;
Q=[0 0 0;0 0 0;0 0 0];
c=[-2; -4;-3];
A=[-3 -4 -2;-2 -1 -2;-1 -3 -2];
b=[-60;-40;-80];
xDonjiLimit=[0 0 0]';
x=zeros(3,1);

% 2.primjer
kapa=500000;
eta=0.000000005;
Q=[11 4 -3 4 1;4 9 0 1 0;-3 0 12 -3 0;4 1 -3 4 0;1 0 0 0 5];
c=[0;4;0;2;1]
A=[1 -2 -3 0 0;-2 0 0 2 0;0 -1 1 5 0];
b=[-9;-5;0];
E=[-1 2 0 -2 0;0 0 1 3 0];
d=[3;5];
xDonjiLimit=[0 0 0 0 0]';
x=zeros(5,1);

% egzaktno rjesenje
stX=quadprog(Q,c,-A,-b,E,d,xDonjiLimit,xGornjiLimit)
n=size(x,1);
Qsv=eig(Q)
```

```
for i=1:N
    ogr_limD=zeros(size(x,1),1);
    ogr_limG=zeros(size(x,1),1);
    ogr_jed=zeros(size(x,1),1);
    ogr_nejed=zeros(size(x,1),1);
    t(i)=h*i;

    % ograničenja zadana jednadžbom
    if(size(E)~=0)
        Z_jed=E*x-d;
        ogr_jed=E'*Z_jed;
    end

    % ograničenja zadana nejednadžbom
    if(size(A)~=0)
        Z_nejed=(A*x-b);
        Z_nejed=min(Z_nejed,0);
        ogr_nejed=A'*Z_nejed;
    end

    % donja granica x-a
    if(size(xDonjiLimit)~=0)
        ogr_limD=x-xDonjiLimit;
        ogr_limD=min(ogr_limD,0);
    end

    % gornja granica x-a
    if(size(xGornjiLimit)~=0)
        ogr_limG=x-xGornjiLimit;
        ogr_limG=max(ogr_limG,0);
    end
   dEx=x'*Q+c'+(ogr_nejed'+ogr_jed'+ogr_limD'+ogr_limG')*kapa;
    x=x-eta*dEx';
    Ex=x-stX;
    nE(i)=norm(Ex,'fro');
    X(i,:)=x;
end
x
%-----------------------------
oo=ones(size(t));
st1A=stX';
WW1=kron(oo',st1A);
%-----------------------------
hFig=figure(1); set(hFig, 'Position', [10 50 1400 900])
semilogy(t,nE, 'linewidth',3)
ylabel('|| E ||','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
set(gca,'fontsize',14,'fontname','Verdana');

hFig=figure(2); set(hFig, 'Position', [10 50 1200 750])
```

```
if(n==3)
    plot(t, WW1, 'k:', 'linewidth',2.5), hold on,
    plot(t, X(:,1), 'b', t, X(:,2), 'r', t, X(:,3), 'g',
'linewidth',3),
    ylabel('x_{1}, x_{2},
x_{3}','FontSize',18,'FontName','Verdana'),
    xlabel('t (s)','FontSize',14,'FontName','Verdana');
    set(gca,'fontsize',14,'fontname','Verdana');
end

if(n==5)
    plot(t, WW1, 'k:', 'linewidth',2.5), hold on,
    plot(t, X(:,1), 'b', t, X(:,2), 'r', t, X(:,3), 'g', t,
X(:,4), 'y', t, X(:,5),'c', 'linewidth',3),
    ylabel('x_{1}, x_{2}, x_{3}, x_{4}, x_{5}
','FontSize',18,'FontName','Verdana'),
    xlabel('t (s)','FontSize',14,'FontName','Verdana');
    set(gca,'fontsize',14,'fontname','Verdana');
end

f=1/2*(x'*Q*x)+c'*x
fegz=1/2*(stX'*Q*stX)+c'*stX
```

Rješavanje problema inverznog njihala:

```
clear all
clc
close all

global n
global A B K
global m1 m2 I2 L2 grav D1 D2

n=4;

T=80;

x30 = 3;      % Kutno odstupanje njihala u t=0; (Lin: x30 =
0.69; rho = 1); (x30 = 0.89;)
%x30 = 3.0;    % Pocetni kut za Swing-up algoritam

X0 = [0 0 x30 0 0];     % pocetni uvijeti

m1=1;          % masa kolica
m2=0.5;        % masa njihala u centru gravitacije
L2=1;          % udaljenost izmedu centra gravitacije njihala i
njegovog lezista
I2 = 0.1;      % moment inercije njihala u centru gravitacije
grav=9.81;     % ubrzanje sile teže
D1 = 0.05;      % koeficjent viskoznog trenja kolica
D2 = 0.01;      % koeficjent viskoznog trenja njihala

%---Linearizirani model
% vektor stanja je x=[q1; v1; q2; v2]
m11 = m1 + m2;
m22 = I2 + m2*L2^2;
m33 = m2*L2;
naz = I2*m11 + m1*m2*L2^2;
w1 = m11/naz;
w2 = m22/naz;

a12 = -D1*w2;
a13 = -m33^2*grav*w2/m22;
a14 = m33*D2*w2/m22;
a42 = m33*D1*w1/m11;
a43 = m33*grav*w1;
a44 = -D2*w1;
b2 = w2;
b4 = -m33*w1/m11;

A=[0 1 0 0; 0 a12 a13 a14; 0 0 0 1;0 a42 a43 a44]
B=[0; b2; 0; b4]

%-------------------------------------------------------------
```

```
N=10000;
h=T/N;          % period sempliranja kod kontinuirane verzije
eta=0.01;       % learning-rate
Kk = ones(size(B))'     % Proizvoljno zadana matrica K_kapa
Q = B*Kk;               % zadani Q>0
pol=[-1 -1.2 -1.4 -1.6]      % vektor zeljenih polova sustava
L = diag(pol);     % matrica s zeljenim polovima na dijagonali
eigA=eig(A)  % direktna provjera stabilnosti otvorenog sustava
P=zeros(size(A));       % inicijalni P (moze biti rand)
P2=lyap(A,-L,-B*Kk)     % P dobiven matlabom
egzInvP=inv(P2)

for i=1:N,
    t(i)=h*i;
    E = A*P - P*L - B*Kk;        % error1
    P = P - eta*(A'*E - E*L');  % gradijentni algoritam
    Pp(i,:,:) = P;
    nE(i)= norm(E,'fro');        % Frobeniusova norma
end
%-----Pole-placement regulator-----------------------------%
p=1*[-1 -1.2 -1.4 -1.6];  % zeljeni polovi zatvorenog kruga
K=place(A,B,p)          % linearni regulator nominalnog sustava
%--------------------------------------------------------%
KegzinvP=Kk*inv(P)

delK=KegzinvP-K

Ar = A-B*K;
eigAr = eig(Ar);   % polovi zatvorenog kruga (moraju biti isti
kao "p")

for i=1:1
    if(i==1)
        K=KegzinvP;
    end

%--Integracija nelinearnog modela---------------------------%
options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t,y] = ode15s('InverzPend02',[0 T],X0,options);
U1=diff(y(:,5))./diff(t);   % upravljacka varijabla
%--------------------------------------------------------%

%--Integracija lineariziranog modela------------------------%
options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t2,y2] = ode15s('InverzPend01',[0 T],X0,options);
U2=diff(y2(:,5))./diff(t2);   % upravljacka varijabla
%--------------------------------------------------------%


%========================================================
```

```
% title('Usporedba lineariziranog i nelinearnog modela')
hFig=figure(1+i*2); set(hFig, 'Position', [10 50 1200 750])

%
%subplot(2,2,1), plot(t,y(:,1), 'b', t2,y2(:,1), 'r:',
'linewidth',3), grid,
subplot(2,2,1), plot(t,y(:,1), 'linewidth',3), grid,
    set(gca,'fontsize',14,'fontname','Verdana');
ylabel('x (m)','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%
%subplot(2,2,2), plot(t,y(:,2), 'b', t2,y2(:,2), 'r:',
'linewidth',3), grid
subplot(2,2,2), plot(t,y(:,2), 'b', 'linewidth',3), grid
set(gca,'fontsize',14,'fontname','Verdana');
ylabel('v_1 (m/s)','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');

%subplot(2,2,3), plot(t,y(:,3), 'b', t2,y2(:,3), 'r:',
'linewidth',3), grid
subplot(2,2,3), plot(t,y(:,3), 'b', 'linewidth',3), grid
set(gca,'fontsize',14,'fontname','Verdana');
ylabel('\theta (rad)','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%
%
%subplot(2,2,4), plot(t,y(:,4), 'b', t2,y2(:,4), 'r:',
'linewidth',3), grid
subplot(2,2,4), plot(t,y(:,4), 'b', 'linewidth',3), grid
set(gca,'fontsize',14,'fontname','Verdana');
ylabel('v_2 (rad/s)','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
%
%============================================================

hFig=figure(2+i*2); set(hFig, 'Position', [10 50 1200 750])
subplot(2,2,1), plot(t(1:end-1),U1, 'b', 'linewidth',3), grid,
title('Nelinearni model','FontSize',20)
set(gca,'fontsize',14,'fontname','Verdana');
ylabel('u (N)','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');

subplot(2,2,2), plot(t2(1:end-1),U2, 'b', 'linewidth',3),
grid, title('Linearizirani model','FontSize',20)
axis([0 T -30 30])
set(gca,'fontsize',14,'fontname','Verdana');
ylabel('u (N)','FontSize',16,'FontName','Verdana'),
xlabel('t (s)','FontSize',14,'FontName','Verdana');
```

Integracija lineariziranog modela:

```
function dy = InverzPend01(t,y)
dy = zeros(5,1);
global A B K n
x=y(1:n);   % vektor stanja
% Regulator:
u = -K*x;
%-------------------------------------------------------------%
dy = A*x + B*u;     % linearizirani model
dy(5) = u;
%-------------------------------------------------------------%
```

Integracija nelinearnog modela:

```
function dy = InverzPend02(t,y)
global n
dy = zeros(n+1,1);
global A B K n
global m1 m2 I2 L2 grav D1 D2
%-- Varijable --%
q1 = y(1);      % pozicija kolica
v1 = y(2);      % brzina kolica
q2 = y(3);      % kut zakreta njihala
v2 = y(4);      % kutna brzina njihala

%--Inertia matrix:
m11 = m1 + m2;
m22 = I2 + m2*L2^2;
m12 = m2*L2*cos(q2);
M = [m11 m12; m12 m22];

%--Coriolis vector:
c1 = -m2*L2*v2^2*sin(q2);
c2 = 0;
Cv = [c1; c2];

%--Gravity vector:
g1 = 0;
g2 = -m2*grav*L2*sin(q2);
G = [g1; g2];

%--(Viskozno trenje):
R=[D1*v1; D2*v2];

%--(Control Vector):
x=y(1:n);

% -------LinearniRegulator:-------
 u = -K*x;        % linearni zakon upravljanja
 u = 0;           % bez upravljanja
% --------------------------------
```

```
%--- Swing Up: ver1_a ---------------------------
u = 1*t*tanh(10*sin(3*t));
if (abs(q2)<0.7)
    u = -K*x;
end
%--------------------------------

%--- Swing Up: ver1_b ---------------------------
q1d = 1*t*t*tanh(10*sin(3*t));
u = q1d;
% u = -10*(q1-q1d);
if (abs(q2<0.7))
    u = -K*x;
end
%--------------------------------

%--- Energy-based Swing Up --- na temelju clanka [4] --------
k_D=50;
k_V=20;
k_P=5;
Bb = [1; 0];
qq = [q1; q2];
E = 0.5*qq'*M*qq + m2*grav*L2*(cos(q2)-1);
u = ((k_D*sin(q2)*(grav*cos(q2)-v2^2)-
(1+sin(q2)^2)*(k_P*q1+k_V*v1))/(k_D+(1+sin(q2)^2)*E));
if (abs(q2<0.5))
    u = -K*x;
end
%--------------------------------

%----Dinamika inverznog njihala---------------%
F = -inv(M)*(Cv + G + R);
U = [u; 0];
W = inv(M)*U;

dy(1) = v1;
dy(2) = F(1) + W(1);
dy(3) = v2;
dy(4) = F(2) + W(2);
dy(5) = u;
```