Универзитет у Београду

Машински факултет

Мастер академске студије

Индустрија 4.0

РОБОТИКА И ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА

П Р О Ј Е К А Т

Оцена пројектног задатка:	Предметни наставници:	проф. др Зоран Миљковић доц. др Никола Славковић доц. др Милица Петровић
	Предметни сарадници:	Александар Јокић, маст.инж.маш. Лазар Ђокић, маст.инж.маш.
	Група: 1	

Потпис наставника:

РБ	Презиме и име:	Бр.инд.	Потпис:
1.	Немања Јанев	4004/20	
2.	Стефан Ковач	4003/20	
3.	Катарина Пантовић	4005/20	
4.	Јелена Цвијан	4010/20	

Школска година: 2020/2021.

Резиме

У делу пројекта 1.1.1 одређен је непознат фрејм помоћу скице координатних система и трансформационог графа. Затим је у делу 1.1.2 израчуната матрица трансформације за конкретне позиције координатних система са циљем уклапања два дела у склоп.

Методом директног кинематичког проблема у делу 1.2 добијена је зависност спољашњих координата од унутрашњих, а затим је израчунат нулти и задати положај.

За исту конфигурацију су у делу 1.3, применом инверзног кинематичког проблема, одређене унутрашње координате у зависности од спољашњих.

Симулирано је транслаторно кретање робота дуж трајекторије са почетном оријентацијом од 45° у делу 2.1.1, затим је робот обучен за кретање по трајекторији у облику слова А са грешком обучавања од 3,26E-09 у делу 2.1.2, да би потом био обучен и за кретање по трајекторији која је задата функцијом са грешком обучавања од 2.31E-05.

У задатку 3 извршена је обука мреже за класификацију бројева, са најмањом оствареном грешком класификације од 0.0517.

У 4. области циљ је био коришћење вештачке неуронске мреже за функционалну апроксимацију. Грешка добијена за другу функцију је 1.8944e-06, док је за прву добијена грешка испод 2.4546e-5.

Списак слика

Списак табеле

Садржај

Увод

Управљање роботом представља могућност индиректног извршења унапред задатих наредби. Да би постојала могућност управљања роботом или манипулатором, потребно је дефинисати координатне системе и бити упознат са конвенцијом описивања односа више координатних система. У делу пројекта који се тиче роботике описани су покретни и непокретни координатни системи и приказан њихов однос на примеру. Манипулатор се до одређених позиција доводи преко ротационих и транслаторних збглобова те је потребно знати везу између величина које описују промене у зглобовима (унутрашње координате) и позиција које врх манипулатора може да заузме (спољашње координате). Метод помоћу ког је могуће одредити спољашње координате у односу на унутрашње назива се директним кинематичким проблемом и он је описан на примеру сферне кофингурације манипулатора. Такође је приказан метод инверзног кименатичког проблема у коме се на основу задатих спољашњих координата одређују унутрашње што се користи у случају када је потребно да манипулатор стигне у унапред задату позицију. Напоменућемо да су сви модели робота, као и њихови цртежи рађени у окружењу SOLID-WORKS софтверског пакета.

Са индустријализацијом и напредовањем технологије јавља се потреба за аутоматизацијом тешких, досадних и прљавих послова али је потребно да они имају дозу аутономности при непредвиђеним околностима и у непознатом окружењу. Вештачке неуронске мреже успешно решавају проблеме у реалном времену и тиме се њихова аутономност повећава због чега су изузетно значајне у све захтевнијим технолошким процесима у индустрији. У овом делу пројекта приказане су неке од примена вештачких неуронских мрежа: кретање робота по задатим трајекторијама, класификација улазних података и функционална апрооксимација, као и технике програмирања у програмском језику Matlab. Такође су приказани начини формирања обучавајућих парова и обучавања неуронских мрежа .

ROBOTIKA I VEŠTAČKA INTELIGENCIJA
PRVI SAMOSTALNI RAČUNSKI ZADATAK
GRUPA 1

Zadatak 1.1:

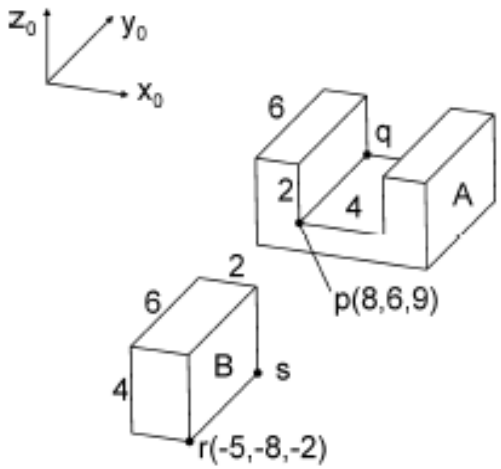
Skicirati raspored četiri frejma ${}^{O}_{A}T$, ${}^{B}_{A}T$, ${}^{B}_{C}T$, ${}^{C}_{O}T$ i odrediti nepoznati frejm ${}^{C}_{O}T$.

$${}^{O}_{A}T = \begin{bmatrix} & R_{x,60} & & 10 \\ & & & -11 \\ & & & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad {}^{B}_{A}T = \begin{bmatrix} & R_{y,45} & & 5 \\ & & & 6 \\ & & & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad {}^{B}_{C}T = \begin{bmatrix} & R_{z,30} & & -4 \\ & & & 5 \\ & & & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Zadatak 1.2:

Naći transformacionu matricu koja rešava operaciju postavljanja (uparivanja) elementa B pri montaži podsklopa A, slika 1, tako da se tačke **p** i **r** i **q** i **s** odnosno pravci **pq** i **rs** poklope.

Zadatkom obuhvatiti:
1. pridruživanje koordinatnih sistema objektima,
2. određivanje transformacionih matrica za opis svakog koordinatnog sistema u odnosu na referentni k.s. $ox_0y_0z_0$,
3. skicirati raspored frejmova i transformacioni graf i
4. sračunati transformacionu matricu $({}^{A}_{B}T)$.



Slika 1. *Postavljanje elementa B pri montaži podsklopa A*

1. Роботика

1.1.1 Одређивање непознатог фрејма

У првом делу ове тачке пројектног задатка потребно је скицирати дате фрејмове и преко њих одредити непознати фрејм ${}^{C}_{0}\mathrm{T}$. Једначинама (1) (2) (3) представљени су фрејмови у облику у ком су задатком задати.

$$ {}^{O}_{A}\mathrm{T} = \begin{bmatrix} & R_x, 60 & & 10 \\ & & & -11 \\ & & & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{B}_{A}\mathrm{T} = \begin{bmatrix} & R_y, 45 & & 5 \\ & & & 6 \\ & & & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

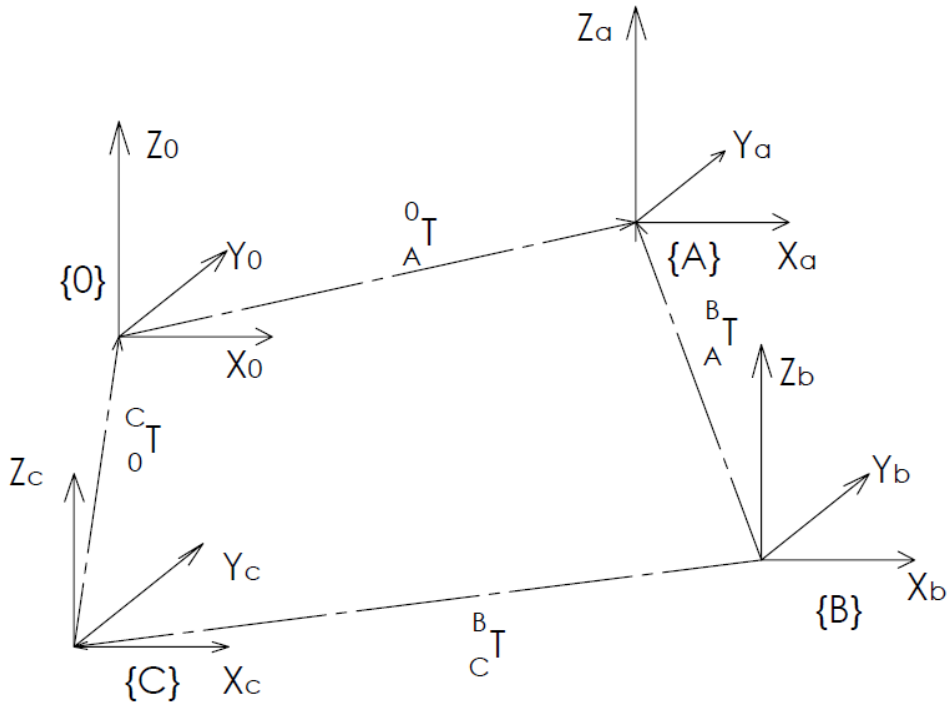
$$ {}^{B}_{C}\mathrm{T} = \begin{bmatrix} & R_z, 30 & & -4 \\ & & & 5 \\ & & & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$ {}^{C}_{0}\mathrm{T} = ? $$

Затим се могу скицирати што је претстављено на Слици 1. Уз помоћ скице може се формирати једначина (4) и њеном трансформацијом долази се до једначине непознатог фрејма (5) .

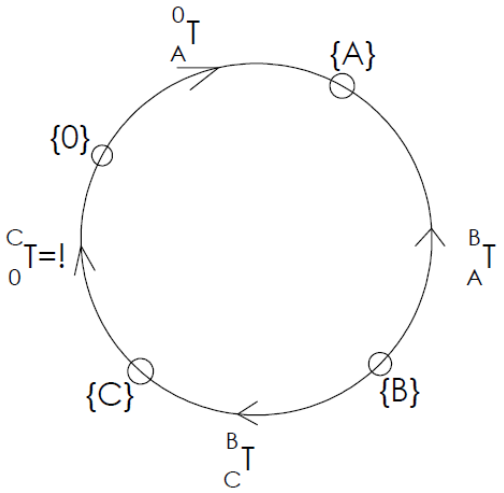
$$ {}^{B}_{C}\mathrm{T} \cdot {}^{C}_{0}\mathrm{T} \cdot {}^{0}_{A}\mathrm{T} = {}^{B}_{A}\mathrm{T} \tag{4} $$

$$ {}^{C}_{0}\mathrm{T} = {}^{B}_{C}\mathrm{T}^{-1} \cdot {}^{B}_{A}\mathrm{T} \cdot {}^{0}_{A}\mathrm{T}^{-1} \tag{5} $$



Слика 1. Скица положаја координатних система

Други метод који се може користити за добијање једначине је уз помоћ трансформационог графа који је приказан на Слици 2. Преко њега може се директно исписати једначина непознатог фрејма.



Слика 2. Трансформациони граф 1.1

$$ {}^{C}_{0}\mathrm{T} = {}^{B}_{C}\mathrm{T}^{-1} \cdot {}^{B}_{A}\mathrm{T} \cdot {}^{0}_{A}\mathrm{T}^{-1} $$

Користећи основне ротационе матрице (Rx, Ry і Rz), могу се формирати коначне једначине задатих фрејмова (6), (7) и (8).

$$ {}^{O}_{A}T = T_{rotx} = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & \cos 60 & -\sin 60 & -11 \\ 0 & \sin 60 & \cos 60 & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 0,5 & -0.866 & -11 \\ 0 & 0.866 & 0,5 & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6) $$

$$ {}^{B}_{A}T = T_{roty} = \begin{bmatrix} \cos 45 & 0 & \sin 45 & 5 \\ 0 & 1 & 0 & 6 \\ -\sin 45 & 0 & \cos 45 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.707 & 0 & 0.707 & 5 \\ 0 & 1 & 0 & 6 \\ -0.707 & 0 & 0.707 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7) $$

$$ {}^{B}_{C}T = T_{rotz} = \begin{bmatrix} \cos 30 & -\sin 30 & 0 & -4 \\ \sin 30 & \cos 30 & 0 & 5 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0,5 & 0 & -4 \\ 0,5 & 0,866 & 0 & 5 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8) $$

Из једначине (5) увиђа се да је потребно одредити транспоноване матрице ${}^{O}_{A}T^{-1}$, једначина (9):

$$ -n^{T} \cdot p = -[1 \quad 0 \quad 0] \begin{bmatrix} 10 \\ -11 \\ 12 \end{bmatrix} = -10 $$

$$ -o^{T} \cdot p = -[0 \quad 0,5 \quad 0,866] \begin{bmatrix} 10 \\ -11 \\ 12 \end{bmatrix} = -4,892 \qquad {}^{O}_{A}T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 0,5 & 0,866 & -4,892 \\ 0 & -0,866 & 0.5 & -15,526 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9) $$

$$ -a^{T} \cdot p = -[0 \quad -0,866 \quad 0.5] \begin{bmatrix} 10 \\ -11 \\ 12 \end{bmatrix} = -15,526 $$

и ${}^{B}_{C}T^{-1}$ једначина (10):

$$-n^{T}\cdot p=-[0,866 \quad 0,5 \quad 0]\begin{bmatrix}-4\\5\\-2\end{bmatrix}=0,964$$

$$-o^{T}\cdot p=-[-0,5 \quad 0.866 \quad 0]\begin{bmatrix}-4\\5\\-2\end{bmatrix}=-6,33 \qquad {}^{B}_{C}T^{-1}=\begin{bmatrix}0,866 & 0,5 & 0 & 0,964\\ -0,5 & 0.866 & 0 & -6,33\\ 0 & 0 & 1 & 2\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (10)$$

$$-a^{T}\cdot p=-[0 \quad 0 \quad 1]\begin{bmatrix}-4\\5\\-2\end{bmatrix}=2$$

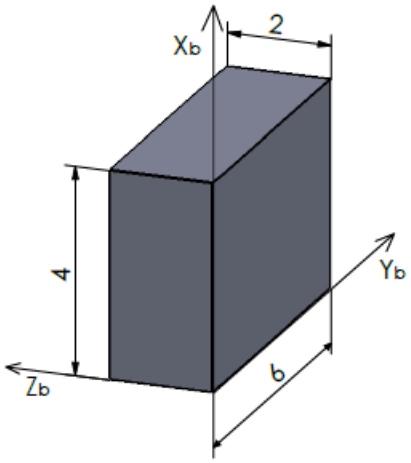
Затим заменом једначина (9) и (10) у једначини (5), израчунава се једначина непознатог фрејма (11)

$${}^{C}_{0}T=\begin{bmatrix}0,866 & 0,5 & 0 & 0,964\\ -0,5 & 0,866 & 0 & -6,33\\ 0 & 0 & 1 & 2\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0,707 & 0 & 0,707 & 5\\ 0 & 1 & 0 & 6\\ -0,707 & 0 & 0,707 & -8\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & -10\\ 0 & 0,5 & 0,866 & -4,892\\ 0 & -0,866 & 0,5 & -15,526\\ 0 & 0 & 0 & 1\end{bmatrix}=$$

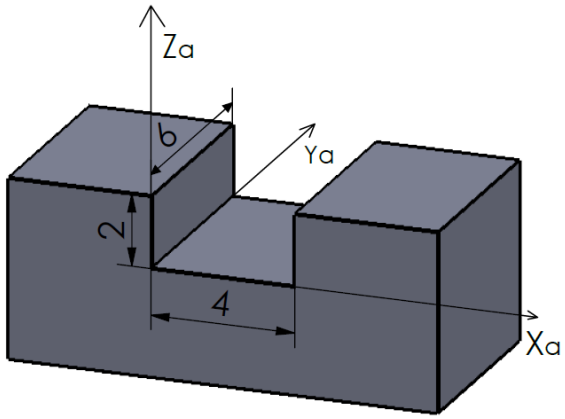
$${}^{C}_{0}T=\begin{bmatrix}0,612 & -0,28 & 0,739 & -9,78\\ -0,353 & 0,739 & 0,573 & 1,153\\ -0,707 & -0,612 & 0,353 & -9,907\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (11)$$

1.1.2 Уклапање дела у склоп са потребном оријентацијом

У другом делу ове тачке потребно је да се део B (Слика 3) одређене неповољне оријентације постави ротацијом, а затим и транслацијом, око оса непокретног координатног система, у повољан положај у односу на део A (Слика 4) да би се омогућило коректно уклапање у склоп AB, тако да се тачке r i q поклопе.

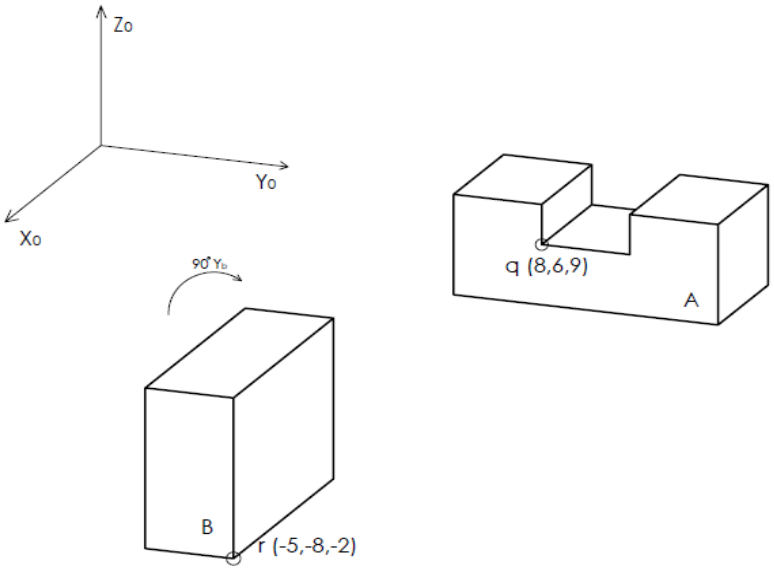


Слика 3. Део B



Слика 4. Део A

На Слици 5 је дат почетни положај оба дела у односу на почетни нулти непомични координатни систем, а у формули (12) положај тачака r и q у нултом координатном систему.



Слика 5. Почетни положаји делова

$$q\begin{bmatrix}8 & 6 & 9\end{bmatrix}$$
$$r\begin{bmatrix}-5 & -8 & -2\end{bmatrix} \quad (12)$$

У формули (13) дефинисаће се положај координатног система A у односу на нулти.

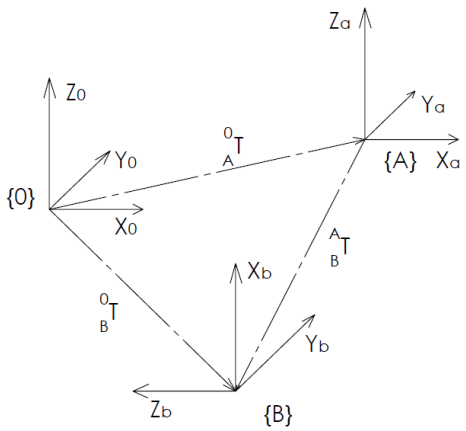
$$ {}^{O}_{A}T = T_{trans}\left(p_x, p_y, p_z\right) = T_{trans}\left(8,6,9\right) = \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13} $$

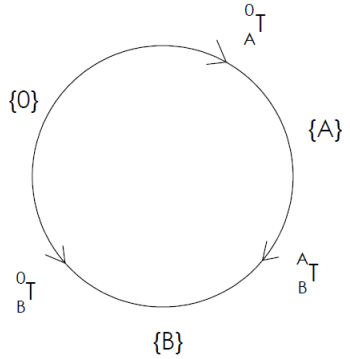
Пошто се део B мора и заротирати за угао од 90 степени, онда ће се извршити множење матрица ротације и транслације како би се поклопиле осе B координатног система и нултог, што је приказано једначином (14).

$$ {}^{O}_{B}T = T_{trans}\left(-5,-8,-2\right)\cdot T_{rot}\left(y,90°\right) = \begin{bmatrix} 1 & 0 & 0 & -5 \\ 0 & 1 & 0 & -8 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = $$

$$ {}^{O}_{B}T = \begin{bmatrix} 0 & 0 & 1 & -5 \\ 0 & 1 & 0 & -8 \\ -1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14} $$

На Сликама 6 и 7 су приказани скица и трансформациони дијаграм преко којих се може одредити непознати фрејм релацијом (15), који је у овом случају ${}^{A}_{B}T$:





Слика 6. Скица положаја координатних система ***Слика 7. Трансформациони граф 1.2***

$$ {}^{A}_{B}T = {}^{O}_{A}T^{-1}\cdot {}^{O}_{B}T \tag{15} $$

Из једначине (15) може се видети да је потребна транспонована матрица ${}^{O}_{A}T^{-1}$. Она је претстављена формулом (16).

$$-n^{T}\cdot p=-\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}8\\6\\9\end{bmatrix}=-8$$

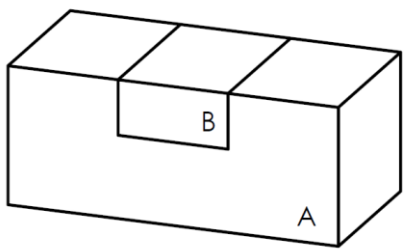
$$-o^{T}\cdot p=-\begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}8\\6\\9\end{bmatrix}=-6 \qquad {}^{O}_{A}T^{-1}=\begin{bmatrix}0 & 0 & 1 & -8\\0 & 1 & 0 & -6\\-1 & 0 & 0 & -9\\0 & 0 & 0 & 1\end{bmatrix} \tag{16}$$

$$-a^{T}\cdot p=-\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}8\\6\\9\end{bmatrix}=-9$$

Заменом (16) у (15) добија се тачно одређен међусобни положај делова који је одређен са (17), а изглед завршеног склопа AB дат је на Слици 8 :

$${}^{A}_{B}T={}^{0}_{A}T^{-1}\cdot{}^{0}_{B}T=\begin{bmatrix}1 & 0 & 0 & -8\\0 & 1 & 0 & -6\\0 & 0 & 1 & -9\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 0 & 1 & -5\\0 & 1 & 0 & -8\\1 & 0 & 0 & -2\\0 & 0 & 0 & 1\end{bmatrix}$$

$${}^{A}_{B}T=\begin{bmatrix}0 & 0 & -1 & -13\\0 & 1 & 0 & -14\\1 & 0 & 0 & -11\\0 & 0 & 0 & 1\end{bmatrix} \tag{17}$$



Слика 8. Изглед склопа

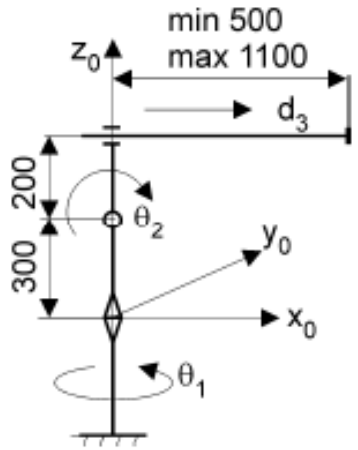
ROBOTIKA I VEŠTAČKA INTELIGENCIJA
DRUGI SAMOSTALNI RAČUNSKI ZADATAK
GRUPA 1

Zadatak 2:

Za prva tri stepena slobode (SS pozicioniranja) robota, slika 1, rešiti direktni kinematički problem.

Zadatkom obuhvatiti:

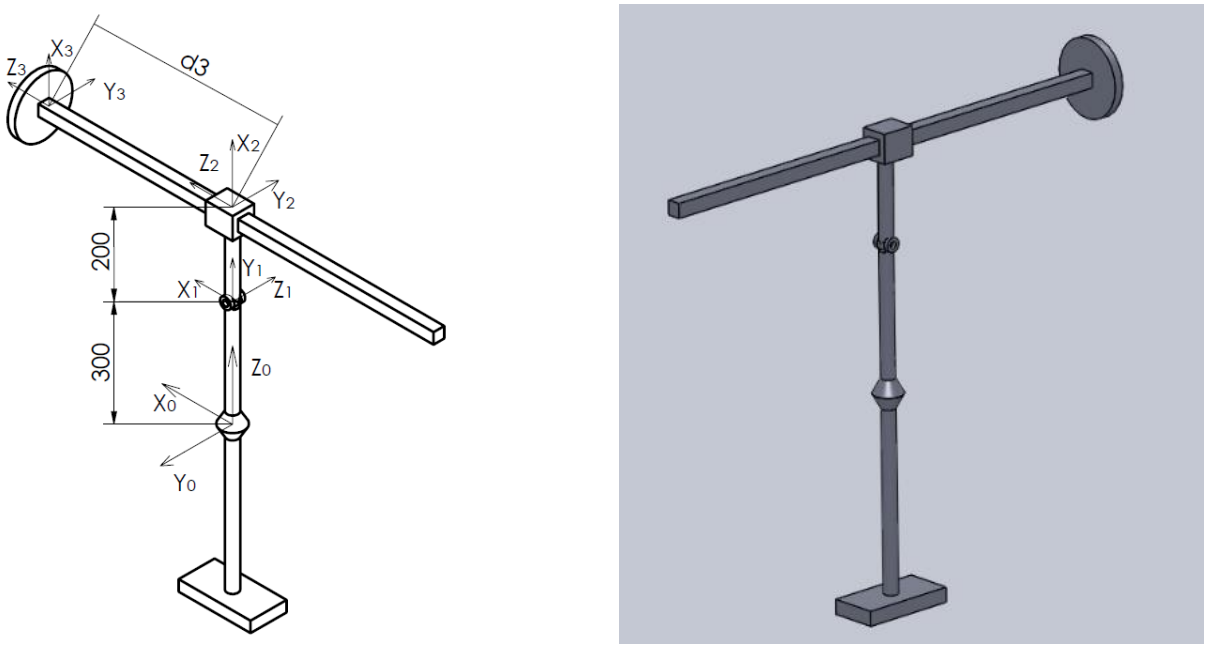
1. pridruživanje k.s. segmentima i određivanje D-H kinematičkih parametara segmenata,
2. formiranje matrica ${}^{i-1}_{i}A$, $i = 1,2,3$,
3. formiranje matrice ${}^{0}_{3}T$ koja specificira poziciju i orijentaciju vrha robota,
4. odrediti i nacrtati nulti položaj robota i
5. za vrednosti promenljivih $\theta_1 = 90^{\circ}$, $\theta_2 = 90^{\circ}$ i $d_3 = 700$, u odnosu na nulti položaj sračunati poziciju i orijentaciju i skicirati frejm ${}^{0}_{3}T$.



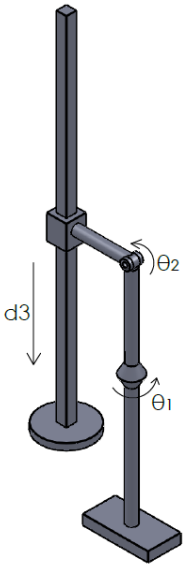
Slika 1. *Robot sa 3SS*

1.2 Решавање DKP

Друга тачка пројекта се заснивала на одређивању положаја врха трећег сегмента у радном простору робота у зависности од лако мерљивих унутрашњих координата које се мере помоћу енкодера. Задата конфигурација је сферног типа (RRT), где се од унутрашњих коорднината појављују θ_1, θ_2 за ротације првог и другог зглоба и d_3 за мерење транслаторног кретања трећег сегмента. Положај који је задат задатком претстављен је на Слици 9.



Слика 9. Задати положај манипулатора



Први корак је додела координатних система покретним сегментима. Да би то урадили на адекватан начин, морамо се одлучити за повољан нулти положај робота и он је приказан на Слици 10. Правило којим смо се водили је да су x осе свих координатних система сегмената у нултом положају међусобно паралелне и у истој оријентацији.

Попуњена таблица Денавит-Хартенбергових параметара налази се у табели 1.

i	α_i [°]	a_i [*mm*]	θ_i [°]	d_i [*mm*]
1	90	0	θ_1 *	300
2	90	200	θ_2 *	0
3	0	0	0	d3*

Табела 1. Д-Х параметри

Слика 10. Нулти положај

Следећи корак је да Д-Х параметре уврстимо у општу трансформациону матрицу ${}^{i-1}_{i}\mathrm{A}$ датом једначином (18).

$$ {}^{i-1}_{i}\mathrm{A} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18} $$

Када се Д-Х параметри убаце три пута за сваки одговарајући фрејм добијају се три трансформационе матрице, (19), (20) и (21).

$$ {}^{0}_{1}\mathrm{A} = \begin{bmatrix} c\theta_1 & 0 & s\theta_1 & 0 \\ s\theta_1 & 0 & -c\theta_1 & 0 \\ 0 & 0 & 1 & 300 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19} $$

$$ {}^{1}_{2}\mathrm{A} = \begin{bmatrix} c\theta_2 & 0 & s\theta_2 & 200c\theta_2 \\ s\theta_2 & 0 & -c\theta_2 & 200s\theta_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20} $$

$$ {}^{2}_{3}\mathrm{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3* \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21} $$

Функцијом (22) смо дефинисали потребну функцију трансформације, да бисмо дефинисали положај трећег сегмента у односу на непокретни координатни систем везан за постоље робота.

$$ {}^{0}_{3}\mathrm{T} = {}^{0}_{1}\mathrm{A}\,{}^{1}_{2}\mathrm{A}\,{}^{2}_{3}\mathrm{A} \tag{22} $$

Поступно ћемо помножити матрице (23), најпре прву и другу, затим и трећу са њима.

$$ {}^{0}_{3}\mathrm{T} = {}^{0}_{1}\mathrm{A}\,{}^{1}_{2}\mathrm{A}\cdot{}^{2}_{3}\mathrm{A} = \begin{bmatrix} c\theta_1c\theta_2 & s\theta_1 & c\theta_1s\theta_2 & 200c\theta_1c\theta_2 \\ s\theta_1c\theta_2 & -c\theta_1 & s\theta_1s\theta_2 & 200s\theta_1c\theta_2 \\ s\theta_2 & 0 & -c\theta_2 & 200s\theta_2+300 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3* \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23} $$

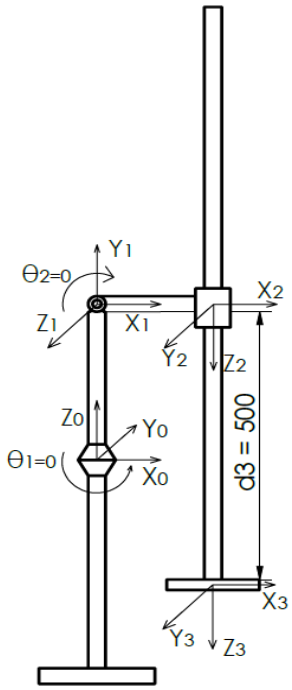
Долазимо до крајње матрице трансформације (24) преко које можемо одредити све координате позиционирања једноставном заменом унутрашњих параметара. Једначина (25) дефинише опште једначине позиционирања.

$$ {}^{0}_{3}\mathrm{T} = \begin{bmatrix} c\theta_1 \cdot c\theta_2 & s\theta_1 & s\theta_2 c\theta_1 & c\theta_1 \cdot (d_3 * s\theta_2 + 200 \cdot c\theta_2) \\ c\theta_2 \cdot s\theta_1 & -c\theta_1 & s\theta_1 s\theta_2 & s\theta_1 \cdot (d_3 * s\theta_2 + 200 \cdot c\theta_2) \\ s\theta_2 & 0 & -c\theta_1 & -d_3 * c\theta_2 + 200 \cdot s\theta_2 + 300 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24} $$

$$ \begin{aligned} Px &= c\theta_1 (d_3 s\theta_2 + 200 c\theta_2) \\ Py &= s\theta_1 (d_3 s\theta_2 + 200 c\theta_2) \\ Pz &= -d_3 c\theta_2 + 200 s\theta_2 + 300 \end{aligned} \tag{25} $$

1.2.1 Одређивање нултог положаја

Да бисмо одредили нулти положај у крајњу матрицу (24) постављамо параметре $\theta_1 = 0°$, $\theta_2 = 0°$ и $d_3 = \min = 500$. На Слици 11 је илустрован нулти положај, а матрично је претстављен у једначини (26).



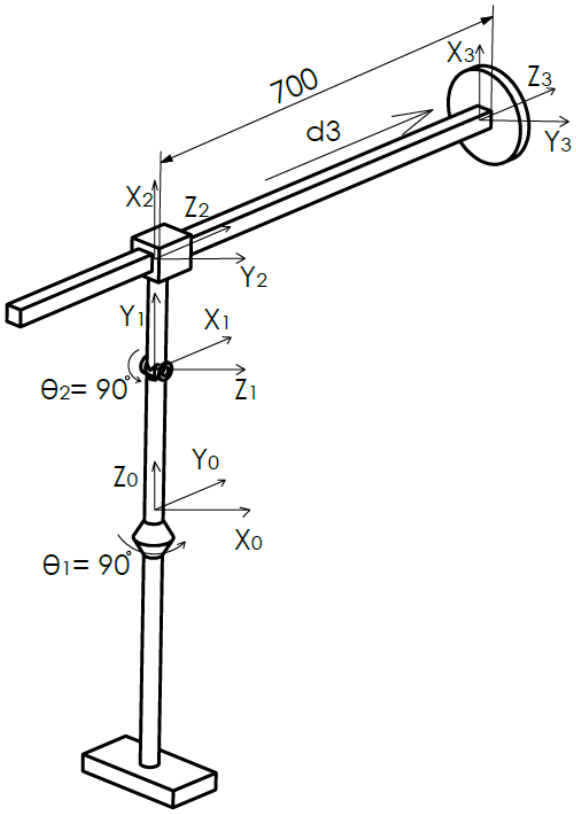
$$ {}^{0}_{3}\mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & 200 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -200 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{26} $$

$$ \begin{aligned} Px &= 200 \\ Py &= 0 \\ Pz &= -200 \end{aligned} \tag{27} $$

Слика 11. Нулти положај са котама

1.2.2 Одређивање траженог положаја

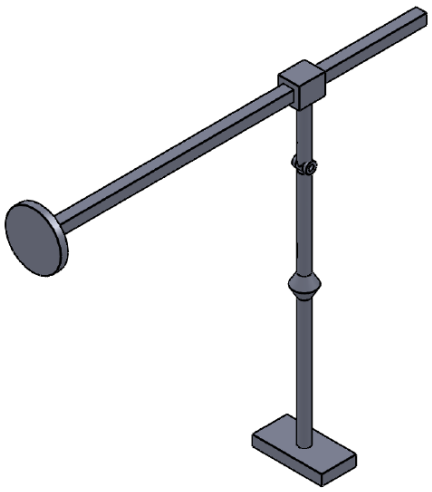
Аналогним поступком као за нулти положај, можемо пронаћи и позицију задатком траженог положаја. Вредности унутрашњих параметара су $\theta_1 = 90°$, $\theta_2 = 90°$ и $d3 = 700$, и њиховом заменом у матрицу (24) добијамо трансформациону матрицу овог положаја (28), а положај илуструјемо Сликом 12.



Слика 12. Тражени положај

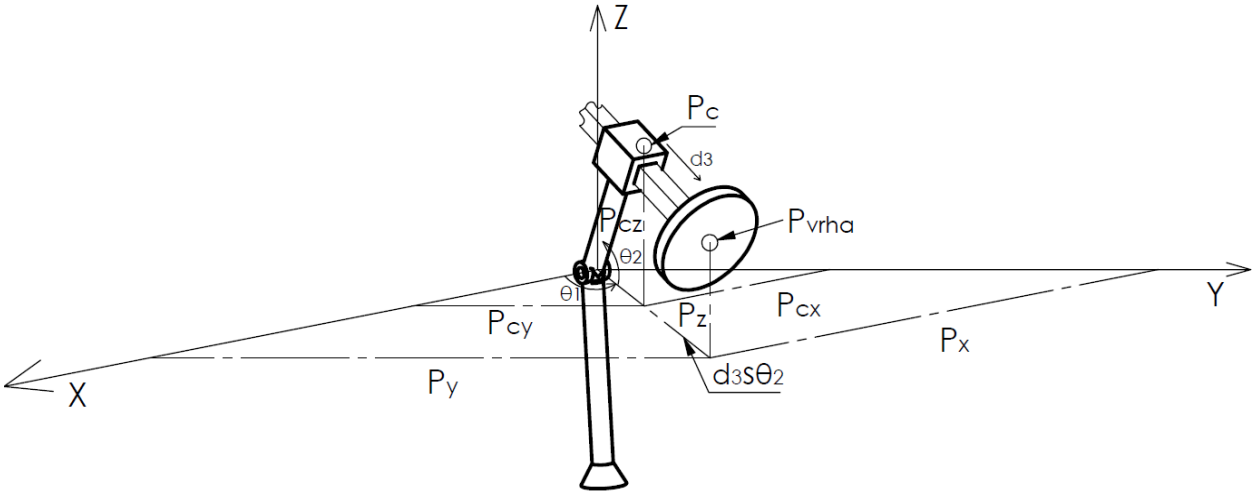
$$ {}^{0}_{3}\mathrm{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 700 \\ 1 & 0 & 0 & 500 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{28} $$

$$ \begin{aligned} \mathrm{Px} &= 0 \\ \mathrm{Py} &= 700 \\ \mathrm{Pz} &= 500 \end{aligned} \tag{29} $$



1.3 Решавање ИКР-а

Трећа тачка се заснива на супротном процесу од друге тачке: овде је потребно помоћу спољашњих координата одредити унутрашње. Слика 13 приказује скицу са које ћемо изводити формуле, а Слика 14 појашњава тај положај робота.



Слика 13. Скица произвољног положаја са разложеним величинама

Како би смо извршили задатак коректно потребно је да одредимо једначине унутрашњих координата, тј θ_1, θ_2 и d_3, преко спољашњих Px,Py i Pz.

Слика 14. Произвољни положај

Са слике 12 можемо одмах дефинисати једначину (30):

$$\theta_1 = \text{atan2}(Py, Px) \qquad (30)$$

„a“ је растојање између врха робота и другог зглоба и израчунава се преко:

$$a = \sqrt{Px^2 + Py^2 + (Pz - 300)^2} \qquad (31)$$

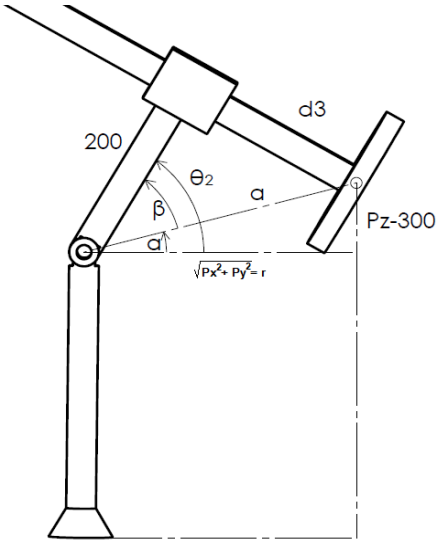
Затим изражавамо $d3$ преко питагорине теореме, где је „a“хипотенуза, а друга катета је 200мм, једначина (32).

$$d3 = \sqrt{a^2 - 200^2}$$

$$d3 = \sqrt{Px^2 + Py^2 + (Pz - 300)^2 - 200^2} \qquad (32)$$

Затим нам преостане само изражавање угла θ_2, који ћемо раздвојити на угао α и угао β, једначина (33)

$$\theta_2 = \alpha + \beta \qquad (33)$$



Слика 15. Појашњена скица

Са Слике 15, можемо извести формулу за бета угао (34) и алфа угао (35):

$$\beta = \operatorname{atan2}(d_3, 200) \tag{34}$$

$$\alpha = \operatorname{atan2}(Pz - 300, \sqrt{Px^2 + Py^2}) \tag{35}$$

Потом заменом једначине (32) у (34), а потом и (34) и (35) у једначину (33), долазимо до коначне једначине за θ_2 (36).

Са тиме је комплетиран IKP за ову сферну конфигурацију робота. Једначине (30), (32) и (36) претстављају његово решење.

$$\theta_2 = \operatorname{atan2}(\sqrt{Px^2 + Py^2 + (Pz - 300)^2 - 200^2}, 200) + \operatorname{atan2}(Pz - 300, \sqrt{Px^2 + Py^2}) \tag{36}$$

1.4 Програмирање Мицубишија

Као додатак првој делу пројектног задатка, било је потребно испрограмирати мобилног робота марке Mitsubishi да се креће и исписује оловком одабрани троцифрени број, у нашем случају 164. Код за исписивање је дат у наставку:

164

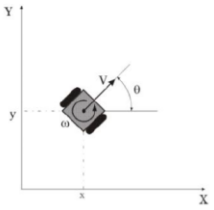
```
10 LPRINT  “NT”
20 LPRINT  “MP -40, 270, 310, -90, 0”
30 LPRINT  “TI 10”
40 LPRINT  “SP 2”
50 LPRINT  “DW 20, 0, 0”
60 LPRINT  “DW 0, 0, -9”
70 LPRINT  “DW 0, 40, 0”
80 LPRINT  “DW 0, 0, 9”
90 LPRINT  “DW 30, 0, 0”
100 LPRINT  “DW 0, 0, -9”
110 LPRINT  “DW -20, 0, 0”
120 LPRINT  “DW 0, -40, 0”
130 LPRINT  “DW 20, 0, 0”
140 LPRINT  “DW 0, 20, 0”
150 LPRINT  “DW -20, 0, 0”
160 LPRINT  “DW 0, 0, 9”
170 LPRINT  “DW 30, 20, 0”
180 LPRINT  “DW 0, 0, -9”
190 LPRINT  “DW 0, -20, 0”
200 LPRINT  “DW 20, 0, 0”
210 LPRINT  “DW 0, 0, 9”
220 LPRINT  “DW 0, 20, 0”
230 LPRINT  “DW 0, 0, -9”
240 LPRINT  “DW 0, -40, 0”
250 LPRINT  “DW 0, 0, 9”
260 LPRINT  “SP 4”
270 LPRINT  “NT”
280 END
```

Вештачка интелигенција

2.1.1 *Поставка задатка*

ПРОЈЕКТНИ ЗАДАТАК 1:

Мобилни робот (слика 1) креће се у равни према следећој коначној једначини кретања:

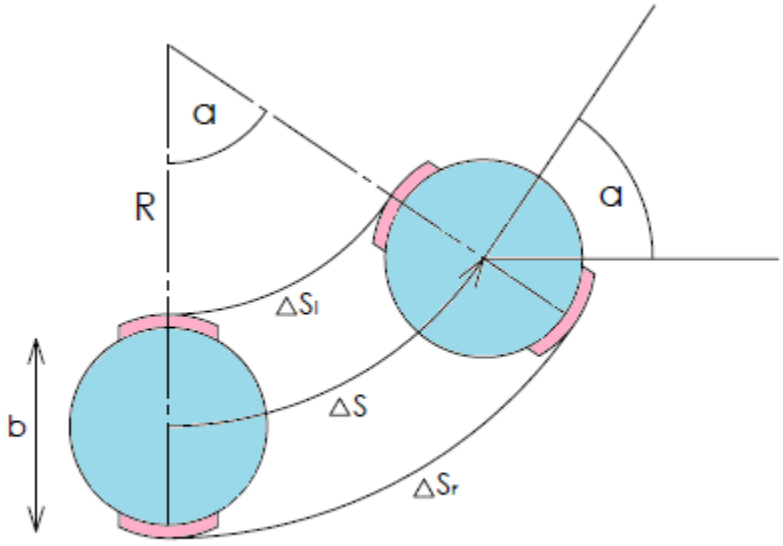


Слика 1:
Мобилни робот са два независна погонска точка

$$x' = \begin{Bmatrix} x \\ y \\ \theta \end{Bmatrix} + \begin{Bmatrix} \frac{\Delta s_d + \Delta s_l}{2} cos\left(\theta + \frac{\Delta s_d - \Delta s_l}{2b}\right) \\ \frac{\Delta s_d + \Delta s_l}{2} sin\left(\theta + \frac{\Delta s_d - \Delta s_l}{2b}\right) \\ \frac{\Delta s_d - \Delta s_l}{b} \end{Bmatrix} \quad (1)$$

Задатак 1.1.1

Математички доказати (извести у приказаном коначном облику) закон кретања мобилног робота.



Слика 16. Промена положаја робота и сви релевантни параметри

Са слике је могуће уочити следеће релације:

$$\Delta s_l = (R - \frac{b}{2})\Delta\Theta \quad (37)$$

$$\Delta s_d = (R + \frac{b}{2})\Delta\Theta \quad (38)$$

$$\Delta s = R\Delta\Theta \tag{39}$$

из којих се решавањем може добити следећи израз:

$$\Delta s = \frac{\Delta s_l + \Delta s_d}{2} \tag{40}$$

Такође, уколико посматрамо угао $\Delta\Theta$ и троугао над њим уочавамо следеће релације:

$$\frac{\Delta s_l}{R - \frac{b}{2}} = \frac{\Delta s_d}{R + \frac{b}{2}} \tag{41}$$

Из које се на основу релације (39) добија:

$$\Delta\Theta = \frac{\Delta s_l - \Delta s_d}{b} \tag{42}$$

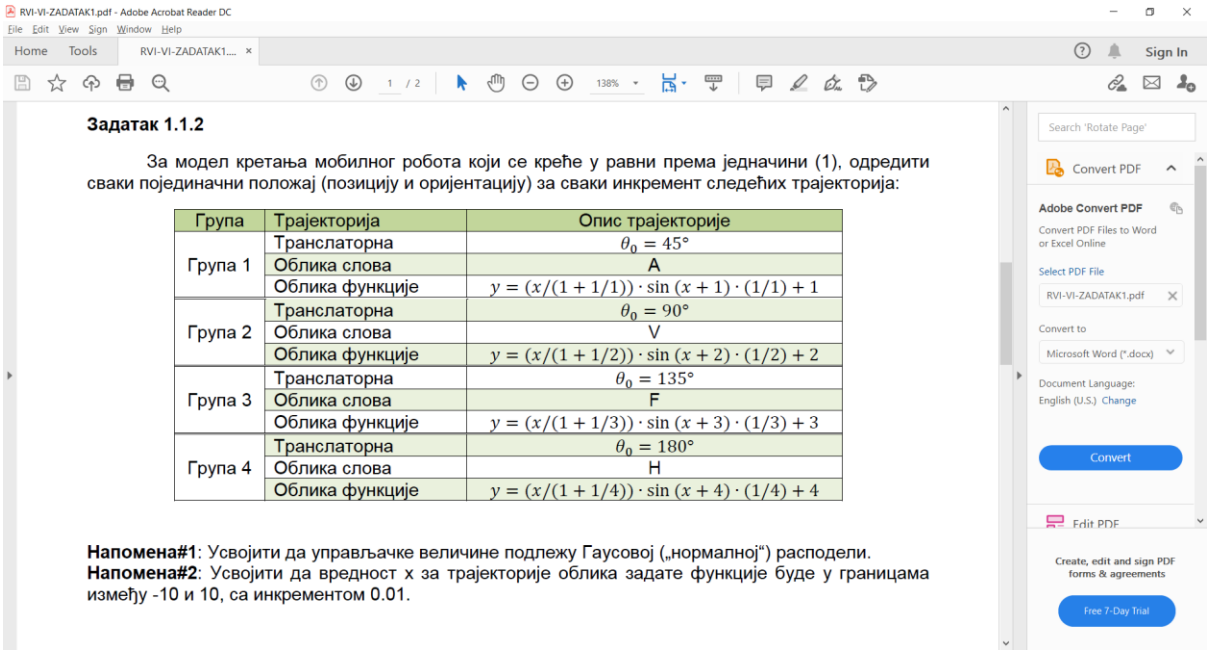
Δx представља растојање дуж x осе центра масе робота између два положаја робота Δy дуж осе y. Уз апроксимацију $\Delta s \approx \Delta d$ лако се види:

$$\Delta x = \Delta s \cos(\Theta + \frac{\Delta\Theta}{2}) \tag{43}$$

$$\Delta y = \Delta s \sin(\Theta + \frac{\Delta\Theta}{2}) \tag{44}$$

Комбинујући формуле 42, 43 и 44 можемо добити координате и оријентацију следећем положају, x', y' и Θ ' што је тражено у задатку и приказано на слици поставке задатка као формула 1.

Задатак 2.1.2



RVI-VI-ZADATAK1.pdf - Adobe Acrobat Reader DC

Задатак 1.1.2

За модел кретања мобилног робота који се креће у равни према једначини (1), одредити сваки појединачни положај (позицију и оријентацију) за сваки инкремент следећих трајекторија:

Група	Трајекторија	Опис трајекторије
Група 1	Транслаторна	$\theta_0 = 45°$
	Облика слова	A
	Облика функције	$y = (x/(1+1/1)) \cdot \sin(x+1) \cdot (1/1) + 1$
Група 2	Транслаторна	$\theta_0 = 90°$
	Облика слова	V
	Облика функције	$y = (x/(1+1/2)) \cdot \sin(x+2) \cdot (1/2) + 2$
Група 3	Транслаторна	$\theta_0 = 135°$
	Облика слова	F
	Облика функције	$y = (x/(1+1/3)) \cdot \sin(x+3) \cdot (1/3) + 3$
Група 4	Транслаторна	$\theta_0 = 180°$
	Облика слова	H
	Облика функције	$y = (x/(1+1/4)) \cdot \sin(x+4) \cdot (1/4) + 4$

Напомена#1: Усвојити да управљачке величине подлежу Гаусовој („нормалној“) расподели.
Напомена#2: Усвојити да вредност x за трајекторије облика задате функције буде у границама између -10 и 10, са инкрементом 0.01.

2.1.2.1 Транслаторно кретање под углом од 45°

Концептуално решење задатка

Посматра се робот који има оријентацију од 45°. План је да се зада наредба роботу, тако да се он креће транслаторно унапред. Да би се то постигло, точкови треба да му се окрећу једнаким брзинама, односно пређени пут левог и десног точка треба да буде једнак за одређени временски интервал.

Детаљан опис пројектног решења

Функција *simuliraj_kretanje* као улазне аргументе добија почетни положај робота: позицију (x,y) и орјентацију θ(teta), растојање између точкова робота b, број корака (инкремената) n и параметре нормалне расподеле m и sigma. Параметар m представља очекивану вредност нормалне расподеле што ће у нашем случају бити вредност једног корака робота, док параметар sigma представља стаднардно одступање од средње вредности тог корака. Ова функција враћа вектор положаја робота у свкаој итерацији.

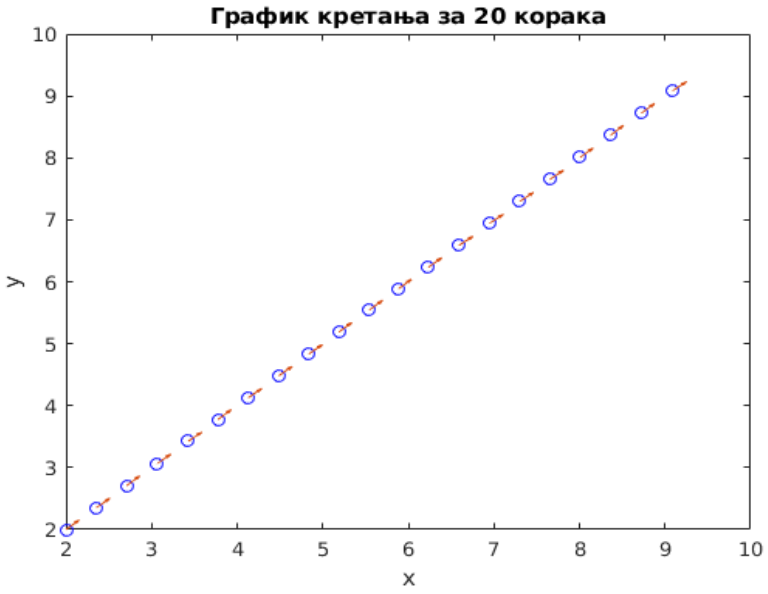
У вектору *koraci* рачунамо и смештамо све вредности ΔS_d и ΔS_l тј. пређени пут десног и левог точка. Иако ми задајемо функцији вредност корака коју желимо да десни и леви точак у итерацији изврше, морамо урачунати и грешку која се у реалности дешава због разних физичких ограничења. Грешку коју робот прави смо у свкаој итерацији генерисали из нормалне расподеле (m, sigma) и додавали на m (очекивану вредност корака).

У свакој итерацији се на основу формуле изведене у задатку 1.1.1. рачунају позиција и оријентација.

```
function [vektor] = simuliraj_kretanje(x,y,teta,b,n,m,sigma)
% n - broj koraka;
% sl i sd imaju N(m,sigma) raspodelu

koraci = [m + sigma*randn(1,n); m + sigma*randn(1,n)]; % deltaS - prvi
 red levi(deltaSl), drugi desni koraci(deltaSr)
vektor = zeros(3,n+1);
vektor(1,1)=x; vektor(2,1)=y; vektor(3,1)=teta;

for i = 2:n+1
    dodaj = [(koraci(2,i-1)+koraci(1,i-1))/2*cos(teta+(koraci(2,i-1)-
koraci(1,i-1))/2*b); ...
             (koraci(2,i-1)+koraci(1,i-1))/2*sin(teta+(koraci(2,i-1)-
koraci(1,i-1))/2*b); ...
             (koraci(2,i-1)-koraci(1,i-1))/b];
    vektor(:,i) = vektor(:,i-1) + dodaj;
end


end
```

Слика 17. Функција simuliraj_kretanje

```
clear all; close all; clc

vektor_koordinata = simuliraj_kretanje(2,2,pi/4,0.5,20,0.5,0.01);
axis equal
plot(vektor_koordinata(1,:),vektor_koordinata(2,:),'ob')
title('График кретања за 20 корака')
xlabel('x')
ylabel('y')
hold on
quiver(vektor_koordinata(1,:),vektor_koordinata(2,:),cos(vektor_koordinata(3,:)),sin(vektor_koordinata(3,:)),0.1)
```



Слика 18. Код и график симулације кретања за угао pi/4 и почетне координате 2,2

2.1.2.2 Слово А

Концептуално решење задатка

При анализи путање облика слова А, приметили смо да је потребно да се робот креће 4 пута транслаторно и 3 пута ротационо адекватним редоследом. Да бисмо описали ове две врсте кретања конструисали смо функције за ротацију и транслацију. Да би се робот коонтинуално кретао, крајњу позицију након транслаторног кретања користили смо за почетну позицију при ротацији и тако наизменично.

Детаљан опис пројектног решења

Функција translacija конструисана је на сличан начин као и код претходног задатка. Једино што је разлика јесте што смо уместо аргумената n и m користили променљиве daljina и br_koraka због тога што желимо да робот пређе унапред дефинисану раздаљину. Један корак је онда дефинисан променљивом d. Излаз из функције су вектор положаја после сваке итерације као и пређени путеви левог и десног точка.

```
function [vektor, koraci] = translacija(x,y,teta, daljina,br_koraka,sigma,b)

%d = linspace(0,sqrt((x1-x)^2+(y1-y)^2),10)
%d = sqrt((x1-x)^2+(y1-y)^2)/br_koraka;
d = daljina/br_koraka;
koraci = [d+sigma*randn(1,br_koraka); d+sigma*randn(1,br_koraka)];

vektor = zeros(3,br_koraka+1);
vektor(1,1)=x; vektor(2,1)=y; vektor(3,1)=teta;

for i = 2:br_koraka+1
    dodaj = [(koraci(2,i-1)+koraci(1,i-1))/2*cos(vektor(3,i-1)+(koraci(2,i-1)-koraci(1,i-1))/2*b); ...
            (koraci(2,i-1)+koraci(1,i-1))/2*sin(vektor(3,i-1)+(koraci(2,i-1)-koraci(1,i-1))/2*b); ...
            (koraci(2,i-1)-koraci(1,i-1))/b];
    vektor(:,i) = vektor(:,i-1) + dodaj;
end

vektor = vektor(:, 2:end)

end
```

Слика 19. Функција translacija, код

Када је потребно да се робот заротира, желимо да то изврши у истој тачки. Ротација у месту ће се догодити уколико оба точка прећу исти пут у различитим смеровима. У зависности од тога да ли се десни точак креће позитивно или негативно можемо управљати смером ротације. Иако задајемо угао за који желимо да се робот заротира, реализација таквог кретања извршава се преко прећеног пута точкова а не директним коришћењем вредности угла. Уколико је прећени пут једнак обиму точка, робот ће се вратити у почетни положај. Део обима точка који робот треба да пређе добијамо множењем обима точка b*pi и односа fi/2*pi. На основу овог резона смо добили прећени пут точка у једном кораку, d. С обзиром да желимо да се робот окрене у једном кораку али поново са неком грешком, генеришемо прећени пут левог и десног точка на следећи начин:

```
koraci = [-d+sigma*randn(1,1); d+sigma*randn(1,1)];          (45)
```

Излаз из функције су vektor и koraci. Vektor смо добили тако што смо на почетни вектор положаја (дефинисан са x, y и teta) додали вектор добијен помоћу формуле изведене у 2.1.1 и резултата ротације (koraci).

```
function [vektor, koraci] = rotacija(x,y, teta, b, fi,sigma)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
%d = b*pi*fi/(2*pi)
d = b*fi/2;
koraci = [-d+sigma*randn(1,1); d+sigma*randn(1,1)];

vektor = zeros(3,2);
vektor(1,1)=x; vektor(2,1)=y; vektor(3,1)=teta;

dodaj = [(koraci(2,1)+koraci(1,1))/2*cos(vektor(3,1)+(koraci(2,1)-koraci(1,1))/2*b); ...
            (koraci(2,1)+koraci(1,1))/2*sin(vektor(3,1)+(koraci(2,1)-koraci(1,1))/2*b); ...
            (koraci(2,1)-koraci(1,1))/b];
    vektor(:,2) = vektor(:,1) + dodaj;

vektor = vektor(:,2);

end
```

Слика 20. Функција rotacija, код

Још једна помоћна функција је ugao_izmedju_vektora која нам служи да бисмо дефинисали за који угао треба да се заротира робот (то је угао fi). Тај угао смо изразили из формуле $x \cdot y = |x||y|\cos(ugao(x,y))$ која представља скаларни производ између два вектора. Функција за улазне аргументе узима два вектора, затим рачуна по дефиницији њихов скаларни производ sk_proizvod, интензитете int1 и int2 и враћа угао између њих, ugao.

```
function [ugao] = ugao_izmedju_vektora(vek1,vek2)
%UNTITLED8 Summary of this function goes here
%   Detailed explanation goes here

sk_proizvod = sum(vek1.*vek2);
int1 = sqrt(sum(vek1.*vek1)); int2 = sqrt(sum(vek2.*vek2));
ugao = acos(sk_proizvod/(int1*int2));

end
```

Слика 21. Функција ugao_izmedju_vektora, код

У функцији main смо симулирали кретање дуж путање слова А. Задали смо да су пречници точкова b=0.5, а стандардно одступање sigma=0.005 За почетни положај смо одабрали координатни почетак а врх слова А смо сместили у тачку са координатама (2,5). За доњи десни

крај слова А узимамо тачку са координатама (4,0). За леву страну цртице слова А узимамо тачку са координама (1,2.5) а за десну (3,2.5). Почетни угао дефинисан је са:

```
pocTeta = atan2(vrhY-pocY,vrhX-pocX);          (46)
```

Описан део кода налази се на Слици 22.

```
clear all; close all; clc
b = 0.5; sigma=0.005;
pocX = 0; pocY = 0; vrhX = 2; vrhY=5; donjiDX = 4, donjiDY=0;
desniX=3; desniY=2.5; leviX=1; leviY=2.5;
pocTeta = atan2(vrhY-pocY,vrhX-pocX);

vektor = [pocX; pocY; pocTeta];
```

Слика 22. Поставка задатка, код

Прво морамо дефинисати раздаљину за транслаторно кретање до врха А помоћу које ћемо дефинисати корак. То смо израчунали на основу координата које су познате и формуле за интензитет вектора која ће дати удаљеност (daljina). Резултат функције translacija, vektor и koraci смештамо у векторе [I1, i2]. У овом делу кода нам vektor представља почетни положај, све положаје током транслације и положај након ротације а koraci пређене путеве точкова у сваком кораку транслације и након ротације. Векторе између којих желимо да одредимо угао за ротацију дефинисали смо на основу познавања координата почетних и крајњих тачака и стандардне дефиниције за вектор између две тачке. Да би се робот заротирао у позитивном смеру потребно је да се десни точак креће унапред а десни уназад и то смо обезбедили узимањем негативне вредности угла fi. Део кода описан у претходном пасусу налази се на Слици 23.

```
daljina= sqrt((vrhX-pocX)^2+(vrhY-pocY)^2);
[i1,i2] = translacija(pocX,pocY,pocTeta,daljina,10,sigma,b);


vektor = [vektor i1];
koraci = i2;

vek1 = [vrhX-pocX, vrhY-pocY]; vek2= [donjiDX-vrhX, donjiDY-vrhY];
fi = ugao_izmedju_vektora(vek1,vek2);

[i1,i2] = rotacija(vektor(1,end),vektor(2,end), vektor(3,end),b,-fi,sigma);
vektor = [vektor i1];
koraci = [koraci i2];
```

Слика 23. Део кода за узлазно кретање

Слично, дефинишемо раздаљину од врха ка десном дну слова и бележимо положаје и пређене путеве точкова након транслације и ротације за угао pi што је приказано на Слици 24.

```
daljina= sqrt((vrhX-donjiDX)^2+(vrhY-donjiDY)^2);

[il, i2] = translacija(vektor(1,end),vektor(2,end), vektor(3,end),daljina ,10,sigma,b);
vektor = [vektor il];
koraci = [koraci i2];

[il, i2] = rotacija(vektor(1,end),vektor(2,end), vektor(3,end),b,pi,sigma);
vektor = [vektor il];
koraci = [koraci i2];
```

Слика 24. Део кода који описује кретање од врха ка десном углу слова A

Сада морамо да вратимо робота до позиције (3,2.5). У тој позицији је потребно да се робот заротира ка последњој тачки (1, 2.5). Угао се одређује преко скаларног производа вектора силазне путање и цртице слова. Угао fi је позитиван због тога што робот мора да се заротира ка лево. Транслација до тачке и ротација приказани су на Слици 25.

```
daljina= sqrt((desniX-donjiDX)^2+(desniY-donjiDY)^2)
[il, i2] = translacija(vektor(1,end),vektor(2,end), vektor(3,end),daljina ,5,sigma,b);

vektor = [vektor il];
koraci = [koraci i2];

vek1 = [desniX-donjiDX, desniY-donjiDY]; vek2= [leviX-desniX, leviY-desniY];
fi = ugao_izmedju_vektora(vek1,vek2);

[il, i2] = rotacija(vektor(1,end),vektor(2,end), vektor(3,end),b,fi,sigma);
vektor = [vektor il];
koraci = [koraci i2];
```

Слика 25. Транслација до (3,2.5) и ротација ка (1, 2.5)

Након транслације робота до позиције (1, 2.5) робот стаје и кретање је завршено. Последњи део кога налази се на Слици 26, а резултат кретања на Слици 27.

```
daljina= sqrt((desniX-leviX)^2+(desniY-leviY)^2);

[il, i2] = translacija(vektor(1,end),vektor(2,end), vektor(3,end),daljina ,5,sigma,b);
vektor = [vektor il];
koraci = [koraci i2];
kraj = [0;0];
koraci = [koraci kraj];

axis equal
plot(vektor(1,:),vektor(2,:),"ob")
hold on
quiver(vektor(1,:),vektor(2,:),cos(vektor(3,:)),sin(vektor(3,:)),0.1, "r--")
```

Слика 26. Завршетак кретања, код

Слика 27. Симулација кретања робота

2.1.2.3 Функција

По услову задатка било је потребно формирати обучавајуће парове тако да су улазни вектори у опсегу (-10,10) са кораком 0.01, а излазни израчунати по функцији y. У свакој тачки рачуна се оријентација на основу чињенице да је извод је тангента на криву а тангента представља правац криве у односу на x осу. По познатој математичкој формули извод је тангенс угла који правац криве заклапа са x осом па се одатле лако добија угао, teta. Задати су параметри sigma, стандардна грешка, и b пречник точка. Описана поставка задатка у коду налази се на Слици 28.

```
x=-10:0.01:10;
y=(x/2).*sin(x+1)+1;
izvod=sin(x+1)/2+(x/2).*cos(x+1);
teta=atan(izvod);
h=length(x);
sigma=0.0005;
b=0.5;
```

Слика 28. Поставка задатка, код

У овом задатку захтевано је да за задату координату израчунамо пређени пут левог и десног точка, односно да у свакој тачки трајекторије решавамо ИКП. На основу формуле изведене у 2.1.1 познавајући пречник точка b могуће је изразити $\triangle S_l, \triangle S, \triangle S_d$ у функцији од $\triangle x, \triangle y$ и $\triangle \theta$.

Формирана је функција predjeni_put у којој се рачунају пређени путеви десног dsd и левог точка dsl на основу изведених формула. Код помоћу ког смо формирали функцију налази се на Слици 29**,** у коме су примењене формуле за ИКП.

```
function [ dsl,dsd ] = predjeni_put( x,y,xl,yl,teta,tetal,b )

a=sqrt((xl-x)^2+(yl-y)^2);
c=b*(tetal-teta)/2;
dsd=a+c;
dsl=a-c;

end
```

Слика 29. Функција predjeni_put

Креирана је for петља унутар које се за сваку вредност задатих x и y рачунају пређени путеви левог и десног точка, и меморишу у vektor-y у виду матрице 2xlength(x). Код за петљу налази се на Слици 30.

```
vektor=zeros(2,1);

for i=1:(h-1)
    [dsl,dsd]=predjeni_put(x(i),y(i),x(i+1),y(i+1),teta(i),teta(i+1),b);
    vektor=[vektor [dsl;dsd]];
end
```

Слика 30. Петља за генерисање пређеног пута левог и десног точка

Иако је познато коју вредност корак левог и десног точка треба да има, зна се да ће у реалности постојати грешка када робот учини тај корак. Грешку је могуће генерисати из нормалне расподеле (vektor(i ,j), sigma). Дакле, очекивана вредност корака се мења у свакој итерацији јер се узимају вредности добијене из ИКП-а. За то је погодно формирати нову петљу унутар које ће се сваки пут генерисати пређени пут са грешком. Затим је потребно са тим пређеним путевима точкова израчунати где ће робот заиста да се нађе (због грешке то неће нужно бити у координатама на основу којих су израчунати пређени путеви). Код се налази на Слици 31.

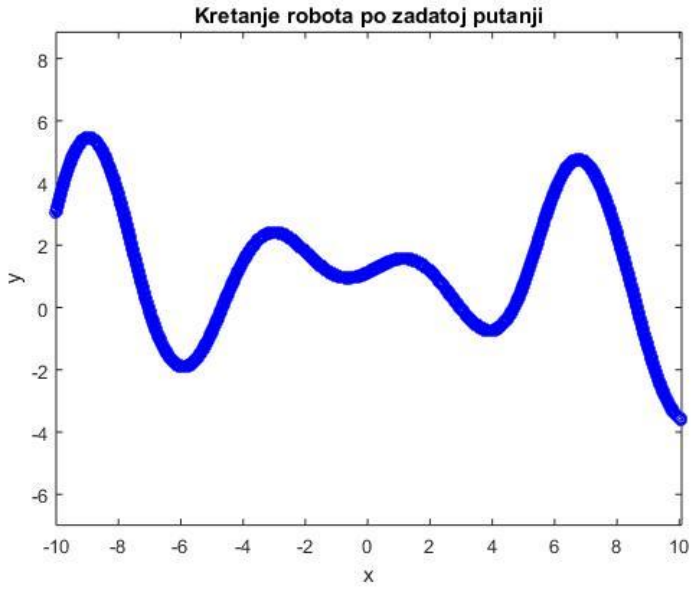
```
polozaji = [x(1); y(1); teta(1)];

vektor=vektor(:,2:h);

for i=1:length(vektor)
    levi = vektor(1,i) + sigma*randn(1,1);
    desni = vektor(2,i) + sigma*randn(1,1);
    dodaj = [((levi+desni)/2)*cos(polozaji(3,i)+(desni-levi)/(2*b));
             ((levi+desni)/2)*sin(polozaji(3,i)+(desni-levi)/(2*b));
             (desni-levi)/b];
    novi = polozaji(:,i) + dodaj;
    polozaji = [polozaji novi];
end
```

Слика 31. Код за генерисање положаја на основу пређених путева

Резултати се смештају у ulaz и vector и тако су формирани улазни и излазни обучавајући парови који представљају положај робота и пређени пут точкова и налазе се на Слици 32. График кретања робота по задатој путањи налази се на слици 33.

```
vektor=[vektor [0;0]];
ulaz=[x;y;teta];
```

Слика 32. Формирање обучавајућих парова



Слика 33. График кретања робота по путањи

2.2.1.1 Обучавање мреже за слово А

На основу резултата 1.1.2 извршити анализу на основу које ће бити изабране компоненте улазног и излазног вектора вештачке неуронске мреже са простирањем сигнала унапред. Изабрати оптималну структуру **„backpropagation"** вештачке неуронске мреже, тако да се коришћењем **MATLAB** и/или **„BPnet"** софтвера, кроз машинско учење, из простора улазног вектора (задатак 1.1.2) изврши пресликавање у простор излазног вектора (задатак 1.1.2). Утврдити, помоћу симулације коришћењем **MATLAB** и/или **„BPnet"** софтвера, утицај фактора **„backpropagation"** вештачке неуронске мреже на процес учења, кроз анализу броја скривених слојева и броја неурона у њима, тежинских коефицијената, утицаја параметара учења и грешке учења, и то тако да се оправданост изабраног оптималног решења верификује сходно постављеном циљу пресликавања.

Експериментални резултати и анализа

Анализа на основу које су изабране компоненте улазног и излазног вектора вештачке неуронске мреже извршена је у 2.1.2.2. Помоћу симулације коришћењем **MATLAB** софтвера, утврђени су утицаји различитих фактора који утичу на обучавање мреже. Резултати симулација налазе се у табели 2. У њој се могу видети, у називима колона, који су параметри разматрани. Најбоље обучена мрежа одређена је према брзини конвергенције и минималној вредности грешке и она је остварена у експерименту под редним бројем 3.

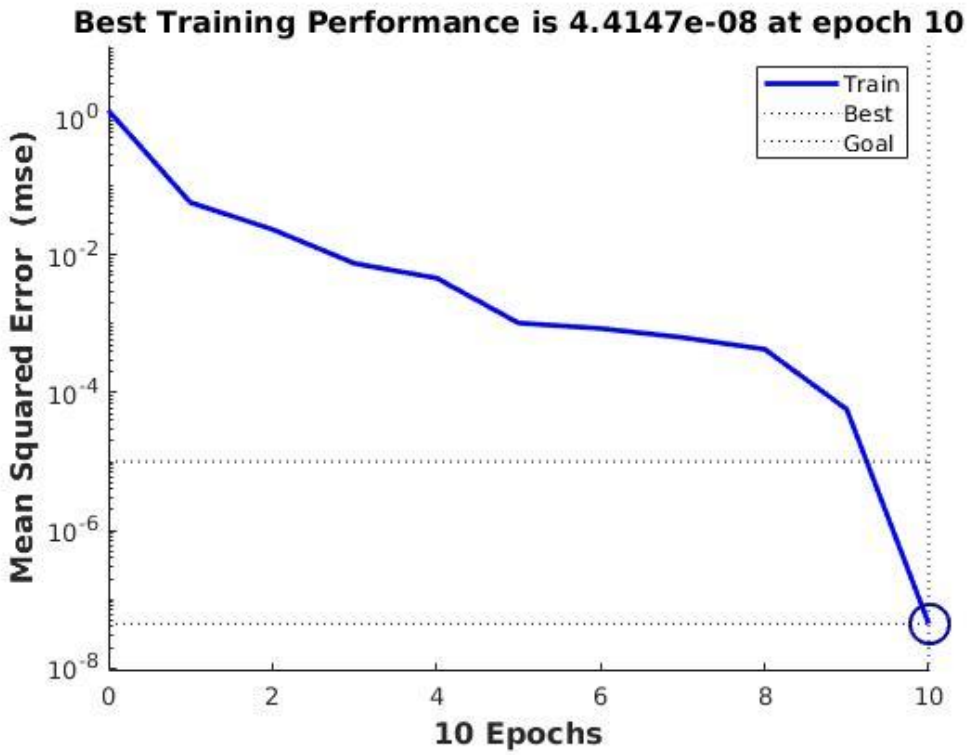
редни број експеримента	број неурона/ слојева	активациона функција	алгоритам обучавања	параметар учења	остварена грешка	број итерација од могућих 1000	t[s]
1	[4, 8]	logsig, tansig	trainlm	0.01	9.98E-06	44	1
2	[4, 8]	logsig, logsig	trainlm	0.01	9.97E-06	148	13
3	[20, 20]	logsig, logsig	trainlm	0.01	3.26E-09	13	2
4	[20, 20, 20]	logsig, logsig, logsig	trainlm	0.01	6.67E-06	10	1
5	[20, 20]	logsig, logsig	trainrp	0.01	9.98E-06	495	2
6	[20, 20]	logsig, logsig	trainrp	0.05	1.57E-05	1000	3
7	[20, 20]	logsig, tansig	trainrp	0.05	1.00E-05	846	3
8	[20, 20]	logsig, tansig	trainr	0.05	1.83E-05	1000	108
9	[20, 20]	logsig, tansig	trainr	0.01		1000	
10	[20, 20]	hardlim, hardlim	trainlm	0.01	1.97E-02	6	0
11	[20, 20, 20]	hardlim, hardlim, hardlim	trainlm	0.01	1.14E-02	5	0
12	[20, 20]	logsig, logsig	trainlm	0.1	5.66E-06	22	6
13	[20, 20]	logsig, logsig	trainlm	0.005	2.20E-06	13	1
14	[20, 20]	logsig, logsig	traingd	0.01	4.45E-02	1000	6
15	[20, 20]	poslin, poslin	traingd	0.01	2.08E-02	1000	7
16	[40, 40]	logsig, logsig	trainlm	0.01	9.76E-07	18	8
17	[20, 40, 20]	logsig, logsig, logsig	trainlm	0.01	6.00E-06	13	6
18	[20, 40, 20]	tansig, tansig, tansig	trainlm	0.01	9.35E-06	9	3
19	[20, 20]	logsig, logsig	traingdx	0.01	4.21E-02	1000	7
20	[20, 20]	logsig, logsig	traingda	0.01	9.67E-02	1000	3

Табела 2 – Резултати обучавања неуронске мреже за слово А

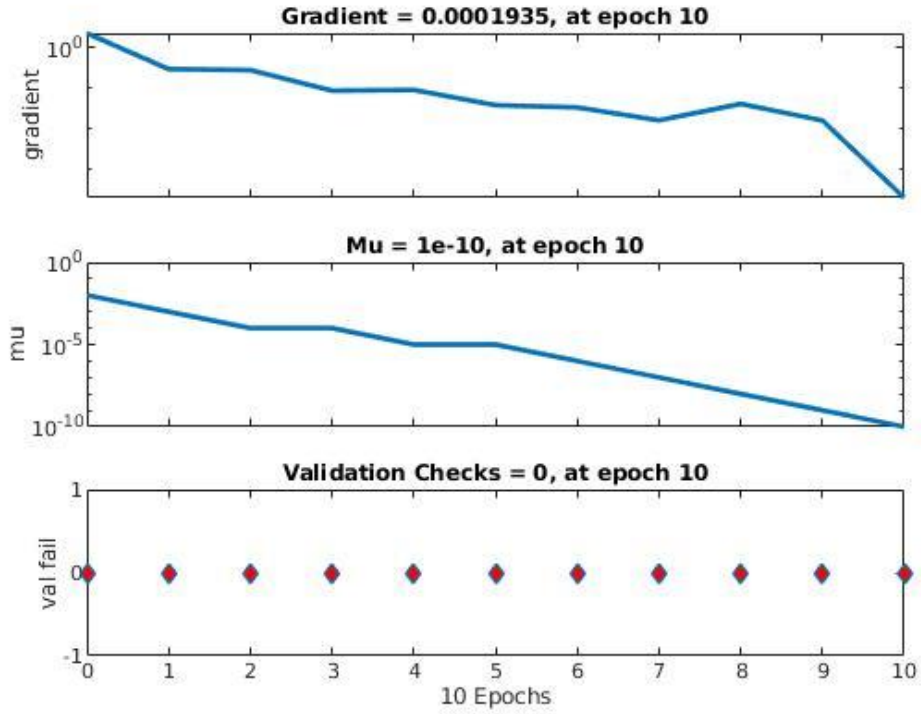
У првом експерименту разматрана је комбинација две активационе функције, ‘logsig’ I ‘tansig’ са малим бројем неурона у 2 слоја као и алгоритмом обучавања trainlm, и резултати су упоређени са истом конфигурацијом али са ‘logsig’ активационом функцијом у оба слоја. У следећем експерименту повећан је број неурона и достигнута боља (мања) грешка обучавања мреже. У 4. експерименту је испробана зависност резултата од броја слојева и приметно је да се мрежа боље обучи уколико је број слојева мањи. Затим је разматран утицај промене алгоритма учења на најбољу конфигурацију као и на комбинацију активационих функција из првог експеримента (6,7). С обзиром да ова комбинација постиже задату грешку учења разматрали смо да ли резултати за ову комбинацију могу бити бољи мењањем алгоритама учења и параметра учења (8,9) али су број итерација и трајање били велики.У експериментима (10,11) мрежа је обучавана са активационом функцијом ‘hardlim’ али је грешка била већа за 7 редова величине и захтевана вредност није достигнута. У експериментима (12-14,16,19 и 29) разматран је утицај промене параметра учења и алгоритма учења на најбољу конфигурацију (ону која је до сада дала најмању грешку), али резултати нису превазишли оне из 3. експеримента. За крај, разматран је утицај (15,17 и 18) активационих функција ‘poslin’ и ‘tansig’ у различитим конфигурацијама на вредност постигнуте грешке.

```
clear all; close all; clc

%load('obucavajuci_ulaz');
%load('obucavajuci_izlaz');
load('ulaz1');
load('izlaz1');

ulaz = [vektor(:,1:2:20) vektor(:,21) vektor(:,22:2:41) vektor(:,42) vektor(:,43:2:52) vektor(:,53) vektor(:,53:2:64)];
izlaz = [koraci(:,1:2:20) koraci(:,21) koraci(:,22:2:41) koraci(:,42) koraci(:,43:2:52) koraci(:,53) koraci(:,53:2:64)];

sim_skup = [vektor(:,2:2:20) vektor(:,21) vektor(:,23:2:41) vektor(:,42) vektor(:,44:2:52) vektor(:,53) vektor(:,54:2:64)];

net = newff(ulaz,izlaz,[20,20], {'logsig','logsig'}, 'trainlm');

net.trainParam.show = 100;
net.trainParam.mu = 0.01;
%net.trainParam.lr = 0.01;
net.trainParam.epochs = 1000;
net.trainParam.goal = 10^-5;
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;
%net.trainParam.min_grad=1e-100;

net = train(net,ulaz,izlaz);
y = sim(net,sim_skup(:,:));
```

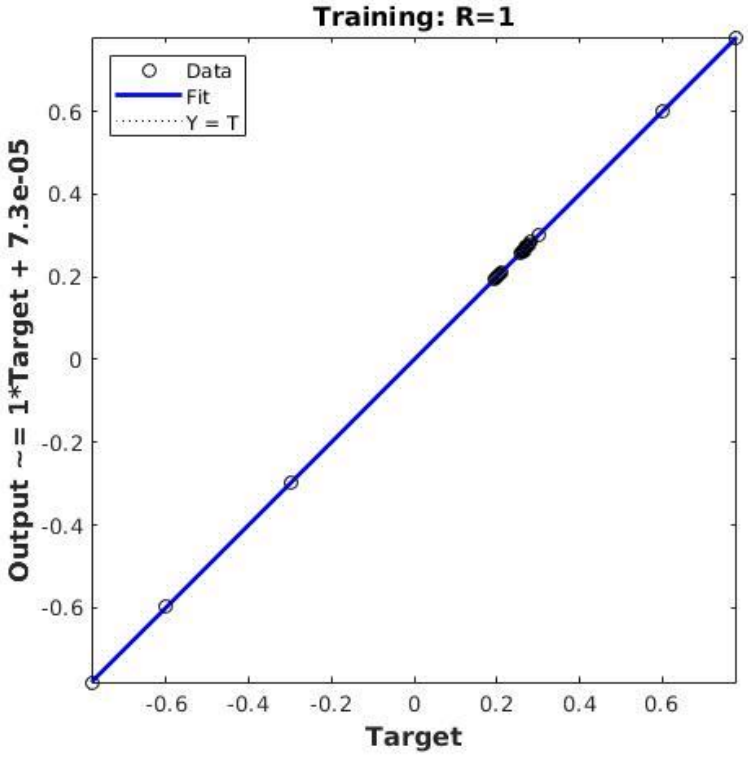
Слика 34- Код помоћу ког је обучена мрежа за оптималне вредности параметара



Слика 35- график за најбоље достигнуту вредност при обучавању мреже



Слика 36- постигнута грешка (слово А)



Слика 37- Линеарна регресија (слово А)

2.2.1.2 Обучавање мреже за функцију

Експериментална анализа и резултати

Анализа на основу које су изабране компоненте улазног и излазног вектора вештачке неуронске мреже извршена је у (име поглавља). Помоћу симулације коришћењем **MATLAB** софтвера, утврђени су утицаји различитих фактора који утичу на обучавање мреже. Резултати симулација налазе се у табели (број). У њој се могу видети, у називима колона, који су параметри разматрани. Најбоље обучена мрежа одређена је према брзини конвергенције и минималној вредности грешке и она је остварена у експерименту под редним бројем 1.

редни број експеримента	број неурона/ слојева	активациона функција	алгоритам обучавања	параметар учења	остварена грешка	број итерација до конвергенције	t[s]
1	[10,8]	logsig	trainlm	0.01	2.31E-05	21	1
2	[30,30]	logsig	trainlm	0.01	2.26E-05	15	10
3	[10,10,10,10]	logsig	trainlm	0.01	2.24E-05	31	5
4	[30,30,30,30,]	logsig	trainlm	0.01	2.20E-05	15	57
5	[10]	logsig	trainlm	0.01	2.29E-05	105	7
6	[10,8]	logsig	trainlm	0.1	2.25E-05	80	8
7	[10,8]	logsig	trainlm	0.25	2.30E-05	55	2
8	[10,8]	logsig	trainlm	0.7	2.30E-05	26	2
9	[10,8]	softmax	trainlm	0.7	2.25E-05	102	91
10	[10,8]	hardlim	trainlm	0.7	1.10E-04	8	1
11	[10,10,10]	hardlim	trainlm	0.7	1.10E-04	8	1
12	[10,8]	hardlim	trainlm	0.1	1.10E-04	7	1
13	[10,8]	hardlim	trainlm	0.005	1.30E-04	6	1
14	[10,8]	hardlim	trainrp	0.005	1.20E-04	68	0
15	[10,8]	logsig	trainrp	0.1	3.26E-05	66	0
16	[10,8]	logsig	trainrp	0.05	3.26E-05	55	0
17	[10,8]	logsig	trainrp	0.005	2.81E-05	47	0
18	[10,8]	logsig	trainbr	0.005	2.30E-05	179	14
19	[10,8]	logsig	traingd	0.005	5.24E-03	1000	4
20	[10,8]	logsig	traingdx	0.005	9.28E-05	346	1

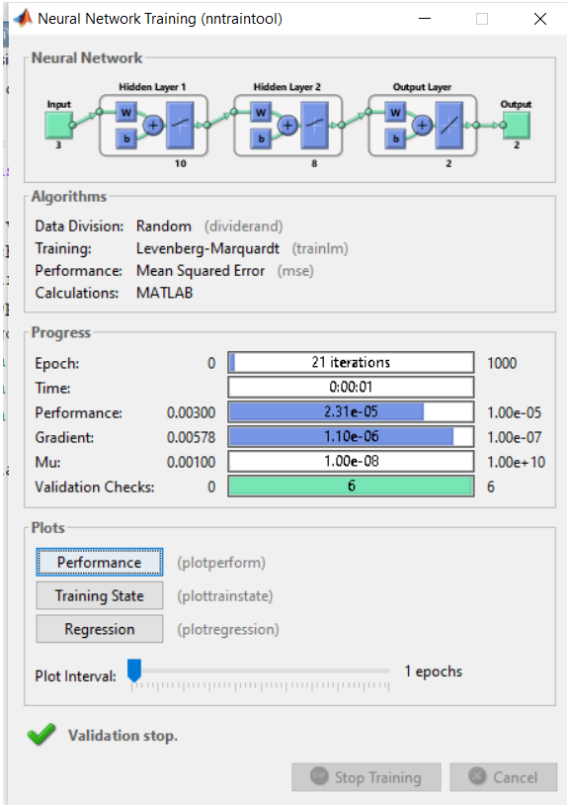
Табела 3. Резултат обучавања неуронске мреже за функцију

Прво смо испитали утицај броја неурона и слојева на брзину конвергенције и вредност грешке. Приметили смо да не постоји разлика у обучености мреже уколико је у питању број слојева, али утиче уколико је у питању број неурона. Нпр. експерименти 2 и 3 имају сличан број неурона а различит број слојева, а експерименти 2 и 4 имају значајну разлику у броју неурона и броју слојева. Тако смо закључили да је најбоља мрежа она која има мањи број неурона. Да бисмо испитали доњу границу броја неурона извршили смо експеримент 5 и видели да је резултат лошији од првог. Такође смо испробали како се досадашња најбоља конфигурација мења са повећавањем параметра учења и закључили смо да иако је грешка незнатно различита, број итерација и време извршавања расту (експерименти 6-8).

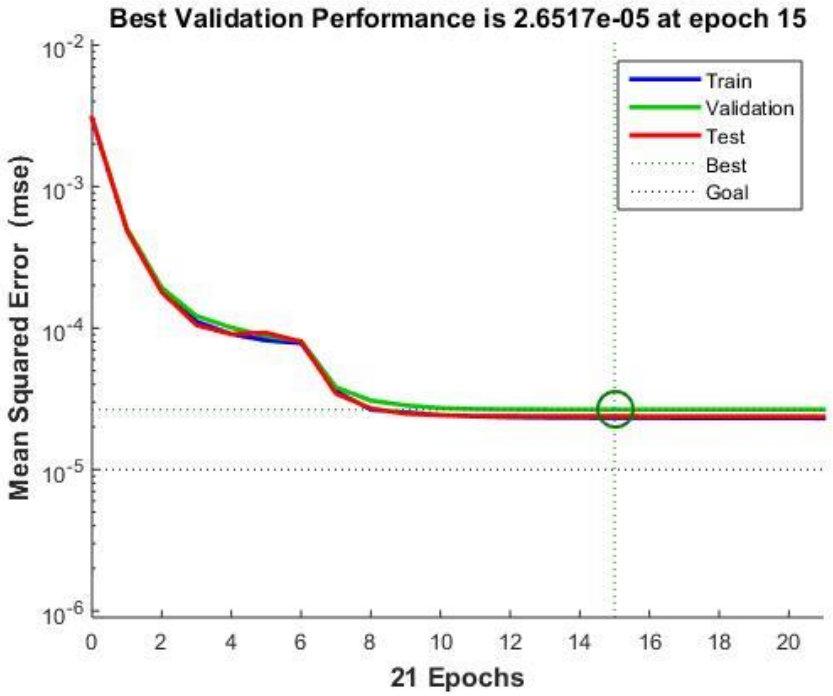
За најбољу конфигурацију броја неурона и слојева, [10,8], разматран је у експерименту 9 и 10 утицај промене активационе функције. Пошто ‘softmax’ има дуже време конвергенције и број итерација до конвергенције, нисмо га даље разматрали. Конфигурацију са активационом функцијом ‘hardlim’ смо мењали како за различит број слојева тако и за различит параметар учења, не би ли смо проверили да ли постоји бољи резултат него у првом експерименту (11-13). За најбољу конфигурацију за ‘hardlim’ променили смо алгоритам учења али тај експеримент (14) није дао боље резултате од првог. Вратили смо се на кофнигурацију у првом експерименту и испробали још неколико различитих алгоритама (15-20) и закључили да је најбоља конфигурација остварена у првом експерименту. Код помоћу ког смо вршили обучавање неуронске мреже налази се на слици 38. На слици 39. налази се резултат обучавања. На слици 40 налази се најбољи валидациони перформанс, на слици 41 резултати метода оцене експерименталних резултата, линеарном регресијом а на слици 42 резултати грешке за време обучавања.

```
clc; clear all; close all;

load('ulaz_sinus');
load('izlaz_sinus');

net=newff(ulaz, vektor,[10,8],{'logsig','logsig'},'trainlm');
net.trainParam.show = 100;
net.trainParam.lr = 0.01;
net.trainParam.epochs = 1000;
net.trainParam.goal = 10^-5;
%net.divideParam.trainRatio = 1; 0.7
%net.divideParam.valRatio = 0;0.15
%net.divideParam.testRatio = 0;0.15

net=train(net,ulaz,vektor);
```

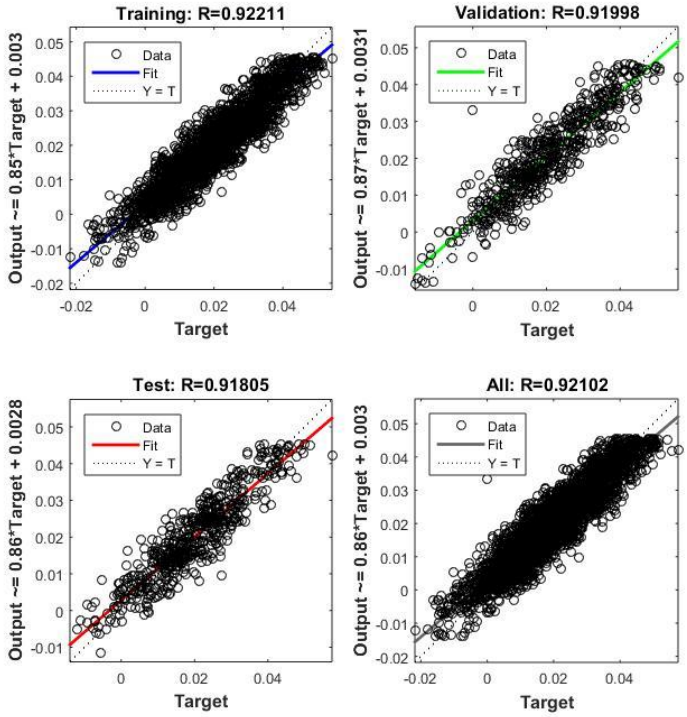
Слика 38. *Део кода који приказује параметре за најбоље обучену мрежу (Matlab)*



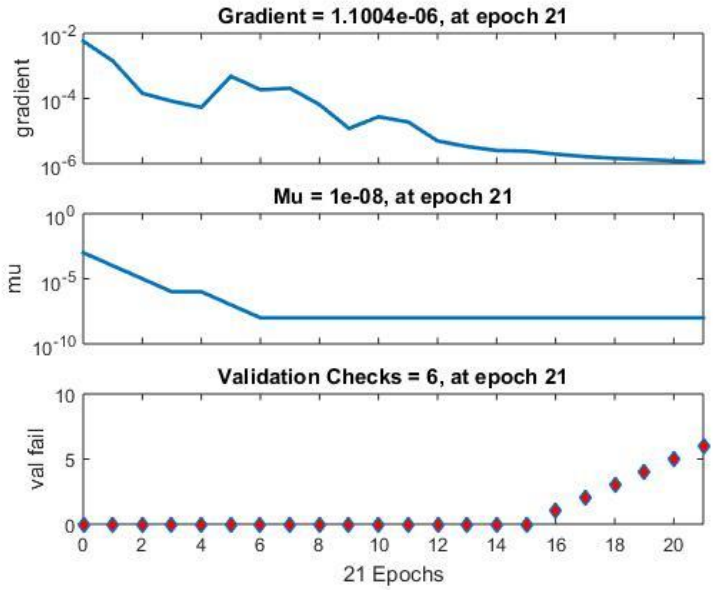
Слика 39. резултат обучавања најбоље мреже (Matlab)



Слика 40. најбољи валидациони перформанс



Слика 41. Линеарна регресија (функција)



Слика 42. Резултати грешке (функција)

3. Препознавање цифара

Поставка проблема

Детаљан опис задатка приказан је на **Слици 43**.

Задатак: Применом вештачких неуронских мрежа у *MATLAB* апликативном софтверу извршити класификацију слика на којима се налазе ручно исписане цифре. Улазни подаци (*Obucavanje_slike*) и излазни подаци (*Obucavanje_klase*) се налазе у фајлу „*ObucavanjeCifre.mat*“, у оквиру *Moodle* електронске учионице. Након обучавања, извршити тестирање и оцену грешке вештачке неуронске мреже применом кода „*TestiranjeCifre.m*“, који се такође налази на Moodle-у. Обучавање вршити док грешка класификације не буде мања од 10% (0.1).

Напомена#1: За процес класификације подесити да у излазном слоју неурона буде компетитивна активациона функција („*softmax*“ или „*compet*“).

Потребно је:

1. Анализирати утицај броја неурона на брзину обучавања и тачност;
2. Анализирати утицај броја скривених слојева на брзину обучавања и тачност;
3. Анализирати утицај активационих функција и алгоритама обучавања на брзину обучавања и тачност;
4. Анализирати утицај параметра учења на брзину обучавања и тачност;
5. Одредити грешку учења и тестирања за сваку обучену мрежу;
6. Сачувати све тестиране вештачке неуронске мреже.

Написати извештај и приложити одговарајуће резултате у електронској форми. У извештају резултате приказати табеларно. Дати коментаре и дискутовати добијена решења.

Слика 43. Опис проблема са сајта предмета Роботика и вештачка интелигенција

Концепцијско решење проблема

Из скупа добијених података формирани су обучавајући парови и анализирани су утицаји промена различитих параметара у мрежи. Идеја је била да се мења један по један параметар и да у свакој промени задржава најбољи резултат. За сваку обучену мрежу посматра се ‘performace’, линеарна регресија, као и крива конвергенције грешке учења вештачке неуронске мреже.

Детаљан опис пројектног решења

У овом делу пројектног задатка било је потребно помоћу 6000 обучавајућих парова обучити мрежу да препознаје задате бројеве са грешком класификације мањом од 0.1. Скуп улазних вектора (Obucavanje_slike) задат је у форми 196x60000, где свака колона представља један улазни вектор.

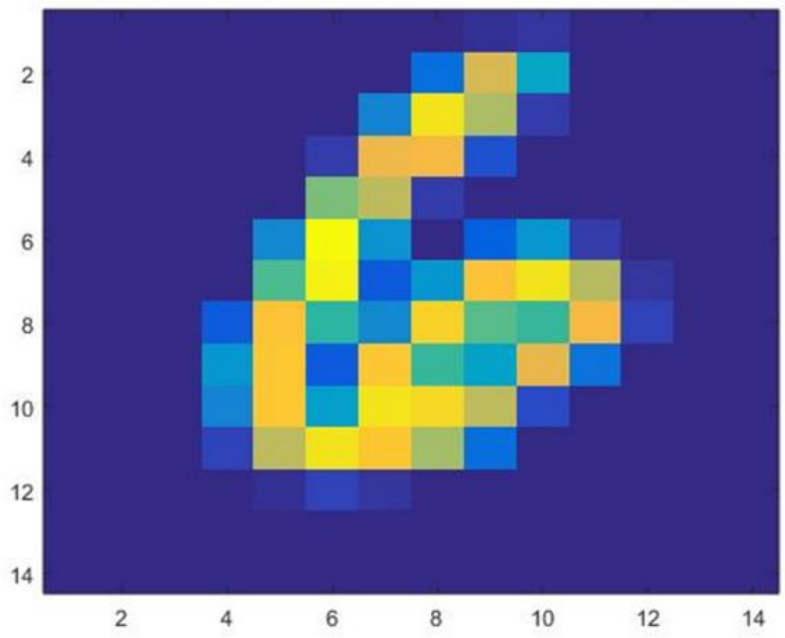
Слика 44: Приказ облика улазног (Obucavanje_slike) и излазног (Obucavanje_klase) вектора

У том вектору описан је насумичан број од 0 до 10, који представља ручно исписане цифре, помоћу вредности пиксела које се крећу од 0 до 255. С обзиром да је димензија улазног вектора 196x1 и да је на почетку било познато да он представља слику ручно исписане цифре, може се приметити да се овде ради о слици димензије 14x14 те је, ради боље визуализације проблема, конструисана функција која прво смешта улазни вектор у матрицу 14x14, а затим ту матрицу исцртава да би се видео број. Линија кода помоћу које је визуелизован вектор је

imagesc(matrizuj_sliku(Obucavanje_slike(:,14))) (47)

```
1   function [matra] = matrizuj_sliku(vektor)
2   %UNTITLED2 Summary of this function goes here
3   %   Detailed explanation goes here
4   matra = zeros(14,14);
5   for i = 1:14
6       for j = 1:14
7           matra(i,j) = vektor((j-1)*14+i);
8       end
9   end
10  end
11
```

Слика 45. Код који визуализује улазни вектор



Слика 46. Визуелизација вектора улаза (Obucavanje_slike(:,14))

Вектор излаза је вектор димензије 10x1 и има јединицу у врсти чији редни број одговара броју на слици, а на свим осталим местима вектор има 0. Вектори излаза су приказани на слици 47.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11																			
12																			
13																			
14																			

Слика 47. Табела вектора излаза (Obucavanje_klase);

Оно што је даље било потребно учинити јесте дефинисати мрежу и обучити је до тренутка док грешка класификације није достигла вредност мању од 0.1 . Да бисмо дошли до оптималне комбинације броја неурона и скревених слојева, алгоритма учења, параметра учења и активационе функције, користили смо се методом у којој смо прво бирали алгоритам учења, затим број слојева и неурона и активационе функције. За сваку пожељну горе наведену комбинацију мењали смо параметар учења у опсегу од минимално 0.0001 до 0.7 и посматрали

грешку учења, број итерација који је био потребан да се мрежа обучи, као и грешку класификације. Грешка класификације је добијена из дела кода за тестирање који је био дат у прилогу.

Broj eksperimenta	Broj neurona i slojeva	Aktivacione funkcije u skrivenom sloju	Aktivaciona funkcija u izlaznom sloju	Algoritam obučavanja	Parametar učenja	Broj iteracija	Ostvarena vrednost greške ucenja	Ostvarena greška klasifikacije
1	[20,15,10]	logsig	softmax	trainlm	0.05	neodređen		
2	[20,15,10]	logsig	softmax	trainlm	0.1	neodređen		
3	[20,15,10]	logsig	softmax	trainrp	0.05	227	0.000402	0.0739
4	[20,15,10]	logsig	softmax	trainrp	0.1	183	0.000437	0.0718
5	[20,15,10]	logsig	softmax	trainrp	0.25	281	0.000427	0.0754
6	[20,15,10]	logsig	softmax	trainrp	0.7	270	0.000697	0.0722
7	[30,30]	logsig	softmax	trainrp	0.05	213	0.00023	0.0571
8	[30,30]	logsig	softmax	trainrp	0.1	173	0.000333	0.0583
9	[30,30]	logsig	softmax	trainrp	0.25	8	0.000349	0.8737
10	[30,30]	logsig	softmax	trainrp	0.7	6	0.0236	0.9114
11	[30,30]	logsig	softmax	trainrp	0.0001	272	0.00037	0.0517
12	[30,30,30]	logsig	softmax	trainrp	0.1	166	0.000474	0.0603
13	[30,30,30]	logsig	softmax	trainrp	0.25	281	0.000434	0.0606
14	[30,30,30]	logsig	compet	trainrp	0.25	0	0	0.902
15	[30,30,30]	logsig	compet	trainrp	0.01	0	0	0.8964
16	[30,30]	logsig	compet	trainrp	0.01	0	0	0.9042
17	[30,30]	logsig	softmax	traingdx	0.01	225	0.248	0.2378
18	[30,30]	logsig	softmax	traingdx	0.25	149	0.2849	
19	[30,30]	logsig	softmax	traingdx	0.001	273	0.0294	0.171
20	[30,30]	logsig	softmax	traingdx	0.0001	317	0.036	0.1921
21	[60,60]	logsig	softmax	traingdx	0.001	281	0.0235	0.268
22	[30,30,30]	logsig	softmax	traingdx	0.001	307	0.000010	0.8185
23	[30,30]	softmax	softmax	trainrp	0.1	123	0.000693	0.1134
24	[30,30]	softmax	softmax	trainrp	0.0001	80	0.00275	0.1061

Табела 4: Резултат обучавања мреже

Прешли смо алгоритам' trainrp' који представља 'backpropagation' алгоритам учења који се користи код мрежа са сигмоидним функцијама у скривеним слојевима које услед својих особина имају мали градијент иако оптималне вредности тежина нису постигнуте. Овај алгоритам служи да отклони овакав проблем.

С обзиром да се најчешће користе сигмоигне активационе функције у скривеним слојевима, њих смо искористили за почетну конфигурацију [25,10,5] (ово је одабрано произвољно). Задатак је био да се на излазном слоју искористи или 'softmax' или 'compet' активациона функција а ми смо за почетак одабрали 'softmax'. Оптимална вредност параметра учења је углавном између 0.25 и 0.7, међутим кренули смо од мање због тога да се не би догодило да баш у овом случају параметар учења прескочи глобални минимум грешке и заузме локални. Наш приступ је био да смо све претходно описане параметре држали фиксним и мењали параметар учења од мањег ка већем. Као што се из табеле 4 може видети, грешка класификације је за цео опсег параметара учења до на другу децималу константа и задовољава тражене услове али смо желели да видимо да ли је могуће достићи још бољи резултат.

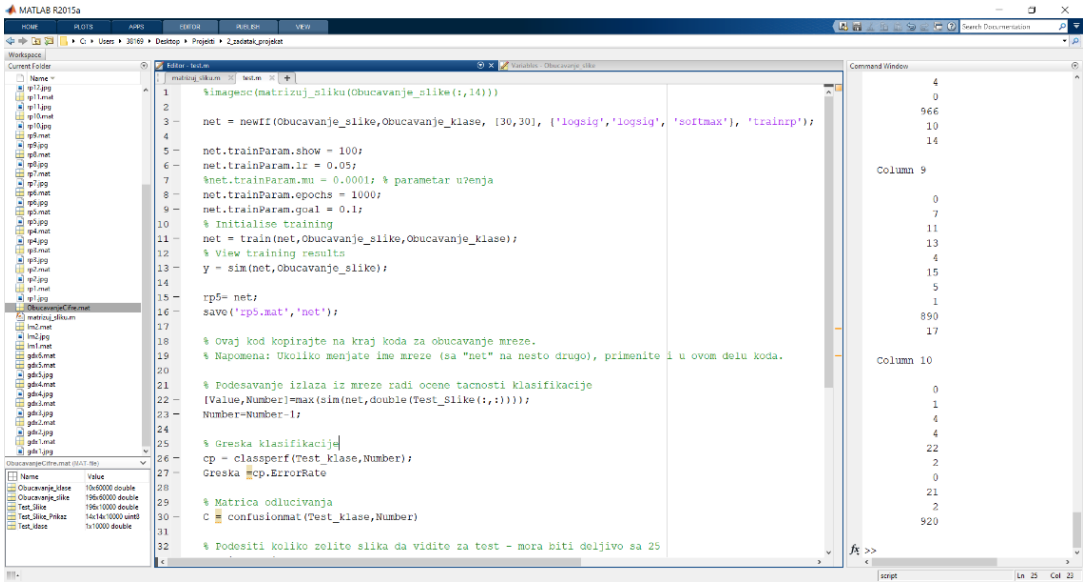
Променили смо број неурона и слојева док је остатак параметара остао исти. Оно што се догодило јесте да се грешка смањила али се и број итерација до остварења конвергенције. Завршавањем инкрементације параметра учења до 0.7 увидели смо да је најбољи резултат и даље за вредности параметра учења 0.05 и 0.1, као и за 0.0001 које смо испробали приметивши да мрежа бива боље обучена за нижи параметар учења. Проверили смо да ли се ситуација мења повећањем броја слојева и увидели да је грешка и даље боља него у случају када смо имали мањи број неурона у сваком слоју појединачно, али да је ипак боља претходна конфигурација. Иако смо достигли жељени резултат, интересовало нас је како ће се понашати наш систем у

случају када искористимо на излазном слоју ‘compet’ функцију. Резултати су за више различитих параметара били незадовољавајући те из тог разлога даље нисмо користили ову функцију.

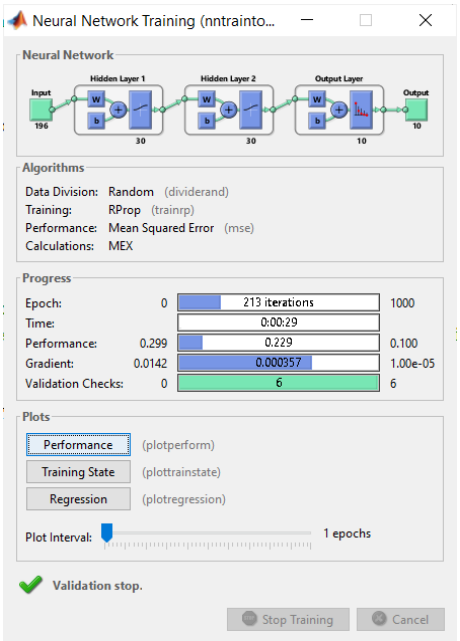
Претходно откривену најбољу конфигурацију неурона и ‘softmax’ смо испробали за traingdx алгоритам учења стандардним принципом, али су грешке за ред величине горе. Као последњу опцију испробали смо ‘softmax’ и за скривени слој јер постоји и за то могућност иако то није најчешће коришћено. Грешке нису биле боље од конфигурације коју смо већ навели.

Експериментални резултати

На **Слици 48** налази се приказ кода који је дао најбољу грешку, 0.0571.



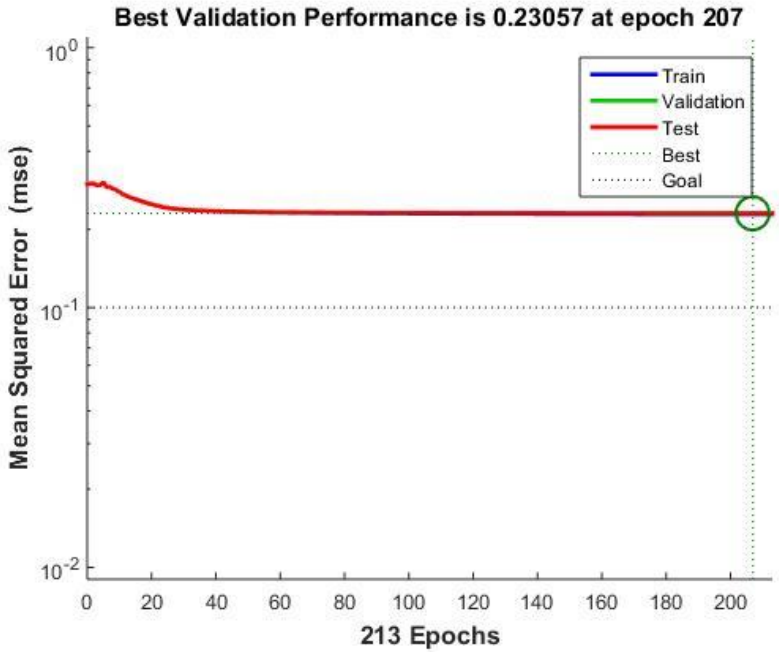
Слика 48. Приказ кода за обучавање мреже



На **Слици 49** налази се резултат обучавања мреже. Видимо да је достигнута конвергенција за 213 епоха од граничне вредности која износи 1000.

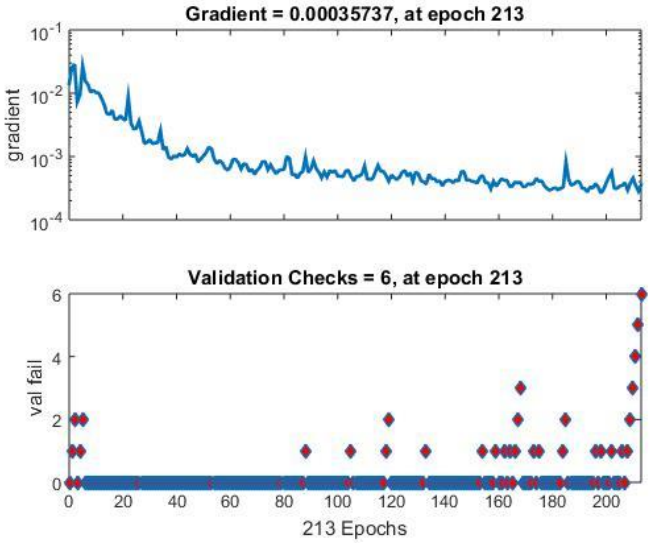
Слика 49. Резултат обучавања мреже

Иако је грешка постигнута за 213 итерација, најбоља вредност ‘validation performance’ постигнута је за 207 итерација и то је приказано на **Слици 50**.

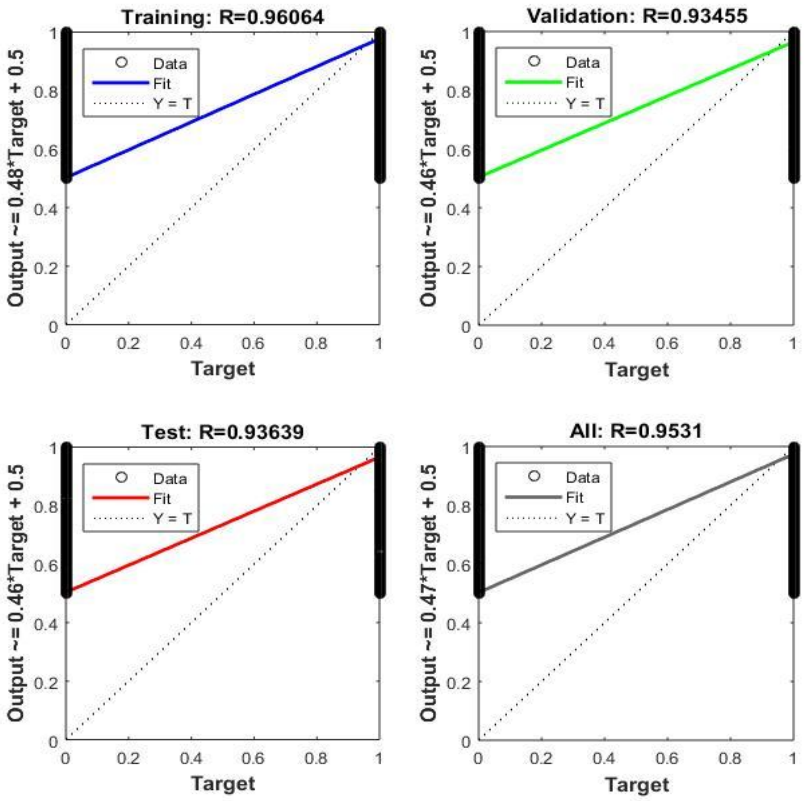


Слика 50. Најбољи резултат ‘validation performance’

На Слици 51 приказани су графици грешке за време обучавања, на слици 52 су графици оцене линеарном регресијом, на слици 53 код који смо користили при обучавању мреже, на слици 54 види се вредност грешке за најбоље обучену мрежу, а на слици 55 се види резултат класификације мреже након тестирања.



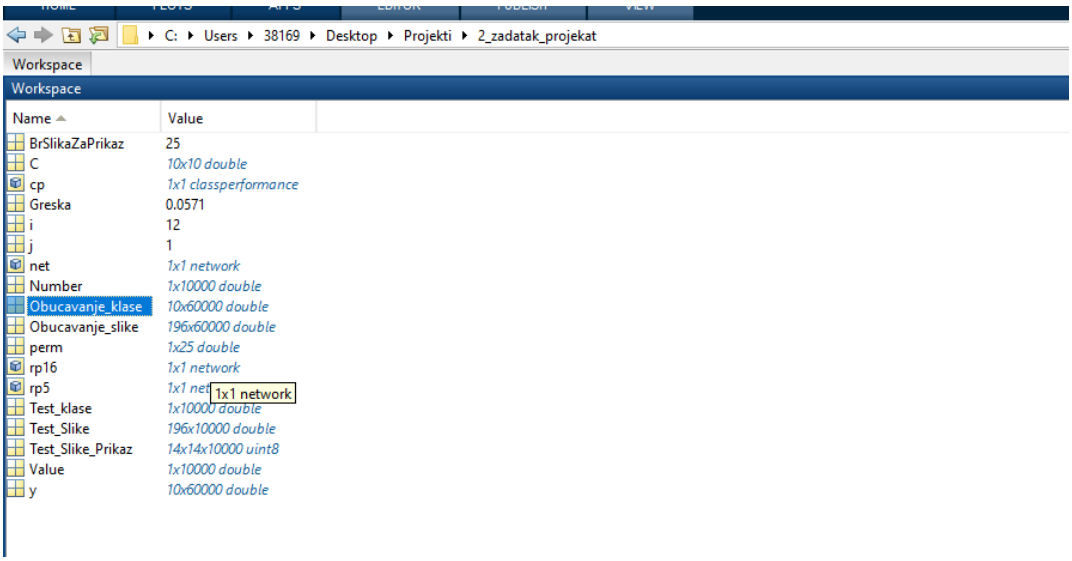
Слика 51. График грешке у току обучавања



Слика 52. Линеарна регресија

```
% Greska klasifikacije
cp = classperf(Test_klase,Number);
Greska =cp.ErrorRate

% Matrica odlucivanja
C = confusionmat(Test_klase,Number)

% Podesiti koliko zelite slika da vidite za test - mora biti deljivo sa 25
BrSlikaZaPrikaz= 25;
perm = randperm(10000, BrSlikaZaPrikaz);
for j=1:BrSlikaZaPrikaz/25
    figure()
    for i=1:BrSlikaZaPrikaz/2
        subplot(5,5,i)
        imshow(Test_Slike_Prikaz(:,:,perm((j-1)*(BrSlikaZaPrikaz/2)+i)))
        if Number(perm((j-1)*(BrSlikaZaPrikaz/2)+i)) ~=Test_klase(perm((j-1)*(BrSlikaZaPrikaz/2)+i))
        title (['\color{red}Net=',num2str(Number(perm((j-1)*(BrSlikaZaPrikaz/2)+i))),' Real=',num2str(T
        else
         title (['\color{blue}Net=',num2str(Number(perm((j-1)*(BrSlikaZaPrikaz/2)+i))),' Real=',num2str
        end
    end
end

% Mogucnosti za odabir Algoritama obucavanja i aktivacionih funkcija - - pokrenite redove ispod bez '%'
```

Слика 53. Код за тестирање мреже



Слика 54. Грешка класификације

Слика 55. Резултат тестирања

4. Обучавање мреже за апроксимацију функције

Детаљан опис задатка приказан је на слици 56,57 и 58.

Задатак#1: Применом вештачких неуронских мрежа у *MATLAB* апликативном софтверу извршити моделирање следећих функција:

Група	Функције
	$y=\sum_{i=1}^{n}-x_i\cdot\sin\left(\sqrt{\lvert x_i\rvert}\right),\quad n=3,\ \mathrm{x}=[-50,-49,...,50]$
Група 1	$y=\left(\frac{1}{500}+\sum_{j=1}^{5}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1},\ \mathrm{x}=[-5,-4.8,...,5]$ $a=\begin{bmatrix}-32 & -16 & 0 & 16 & 32\\ 32 & 16 & 0 & -16 & -32\end{bmatrix}$

Слика 56. Поставка задатка 3.1 и 3.2

Напомена#1: Параметар *n* представља димензију улазног вектора задате функције $\left(y=f\left(x_1,x_2,...,x_n\right)\right)$.

Напомена#2: Извршити претпроцесирање - скалирање података пре обучавања (*Миљковић З., Алексендрић Д., Вештачке неуронске мреже – збирка решених задатака са изводима из теорије (II издање), Универзитет у Београду – Машински факултет, Београд, 2018., 102. страна*)

Слика 57. Додатак уз текст 3.1 и 3.2

Потребно је:

1. Формирати скуп обучавајућих парова;
2. 70% скупа обучавајућих парова користити за обучавање вештачке неуронске мреже, 15% за валидацију и 15% за тестирање;
3. Анализирати утицај броја неурона на брзину обучавања и тачност;
4. Анализирати утицај броја скривених слојева на брзину обучавања и тачност;
5. Анализирати утицај активационих функција и алгоритама обучавања на брзину обучавања и тачност;
6. Анализирати утицај параметра учења на брзину обучавања и тачност;
7. Одредити грешку учења и тестирања за сваку обучену мрежу;
8. Сачувати све тестиране вештачке неуронске мреже.

Написати извештај и приложити одговарајуће резултате у електронској форми. У извештају резултате приказати табеларно. Дати коментаре и дискутовати добијена решења.

Слика 58. Тражене тачке

Видимо да слике 56, да задатак можемо поделити у два дела. Обучавање вештачке неуронске мреже која користи функцију која се налази у плавом реду табеле, зваће се у наставку мрежа 3.1, а мрежа која користи другу функцију зваће се мрежа 3.2.

4.1.Мрежа прве функције

Концепцијско решење проблема

Користећи задату функцију, формирани су обучавајући парови. Улазни вектор у мрежу је тродимензиони елемент из домена функције, док се излаз из мреже пореди са сликом функције од тог елемента. Обучено је око 20 мрежа са различитим параметрима: алгоритмом обучавања, бројем неурона, активационим функцијама у њима и параметрима учења. Бирана је најбоља мрежа, тако што су посматране вредности перформанси на валидационом скупу. Свака мрежа са одређеним, горе наведеним параметрима, тренирана је више пута са различитим иницијалним тежинама, како би се имао бољи увид у перформансе те врсте мреже.

Детаљан опис пројектног решења

Прво су генерисане вредности функција у свим тачкама домена, а затим су преобликовани, тј. секвенцирани, тако да вештачка неуронска мрежа може да их користи за обучавање. На слици 59, приказан је програмски код који је описао рачунару ове инструкције.

```
x = -50:50;
n = 3;

ulaz1_multidim = zeros(length(x),length(x),length(x),3);
izlaz1_multidim = zeros(length(x),length(x),length(x));

for i1 = 1:length(x)
    for i2 = 1:length(x)
        for i3 = 1:length(x)
            ulaz1_multidim(i1,i2,i3,:) = [x(i1) x(i2) x(i3)];
            pom = [x(i1),x(i2),x(i3)];
            izlaz1_multidim(i1,i2,i3) = sum(-pom.*sin(sqrt(abs(pom))));
        end
    end
end

ulaz1 = reshape(ulaz1_multidim, [length(x)^3, 3])';
ulaz1 = ulaz1(3:-1:1,:);
izlaz1 = reshape(izlaz1_multidim, [length(x)^3, 1])';
```

Слика 59. Код који генерише и спрема обучавајуће парове за мрежу

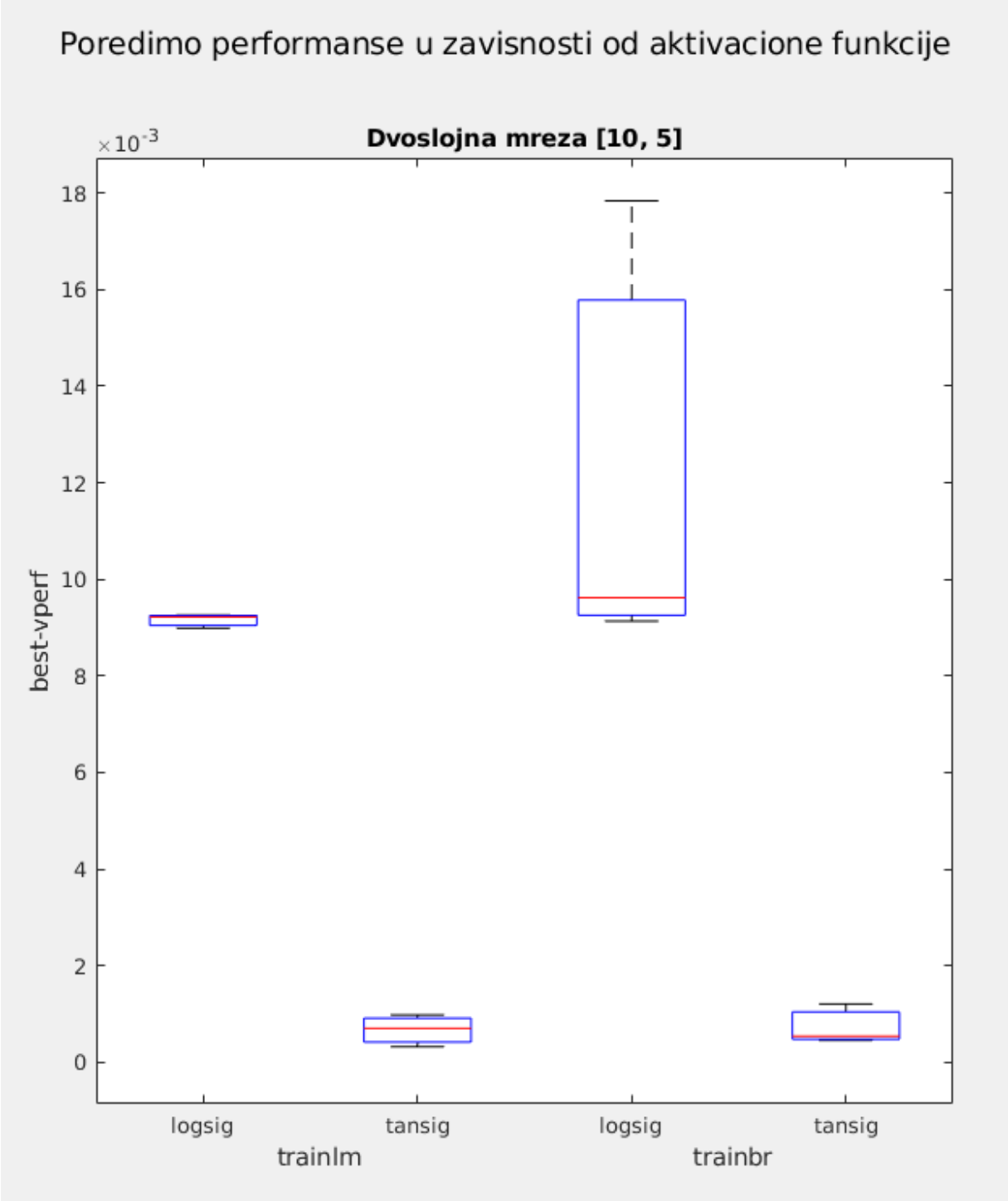
Следећи корак је био скалирање обучавајућих парова на интервал [0,1], затим тренирање мрежа дела прве фазе обучавања. (слика 60) Испробане су активационе функције ‘logsig’ и ‘tansig’, за мрежу која има структуру скривених слојева [10, 5], користи алгоритам обучавања ‘trainbr’ и

параметар учења 0.5. Оба типа мреже су позвана на тренинг 3 пута са различитим иницијалним тежинама.

```
for i = 1:3
    ulaz1(i,:) = (ulaz1(i,:)-min(ulaz1(i,:)))./(max(ulaz1(i,:))-min(ulaz1(i,:)));

end

izlaz1 = (izlaz1-min(izlaz1))./(max(izlaz1)-min(izlaz1));
f={'logsig','tansig'}
for i = 1 : length(f)
    t=char(f{i})
        for k=1:3
            net = newff(ulaz1,izlaz1,[10, 5],{t,t,t},'trainbr');
            net.trainParam.show = 100;
            net.trainParam.lr = 0.5; % ne vazi za 'trainlm'
            net.trainParam.mu = 0.6 % parametar uzenja
            net.trainParam.epochs = 70;
            net.trainParam.goal = 1e-5;
            net.trainParam.max_fail=20000;
            net.divideParam.trainRatio = 0.7;
            net.divideParam.valRatio = 0.15;
            net.divideParam.testRatio = 0.15;
            [net,tr] = train(net,ulaz1,izlaz1,'useParallel','yes');

            validation(k)=tr.best_vperf
            mreze_ukupno(k)={net}
            tr_ukupno(k)={tr}
        end


    [vred,poz] = min(validation);
    naj_mreza{i} = mreze_ukupno{poz};
    naj_tr{i} = tr_ukupno{poz};
    rezultati(i,:) = validation;

end
```

Слика 60: Скалирање обучавајућих парова и тренирања

Аналогно је урађено уз измену алгоритма обучавања са ‘trainlm’ и упоређене су перформансе сва 4 типа мрежа које смо обучили. (Слика 61)



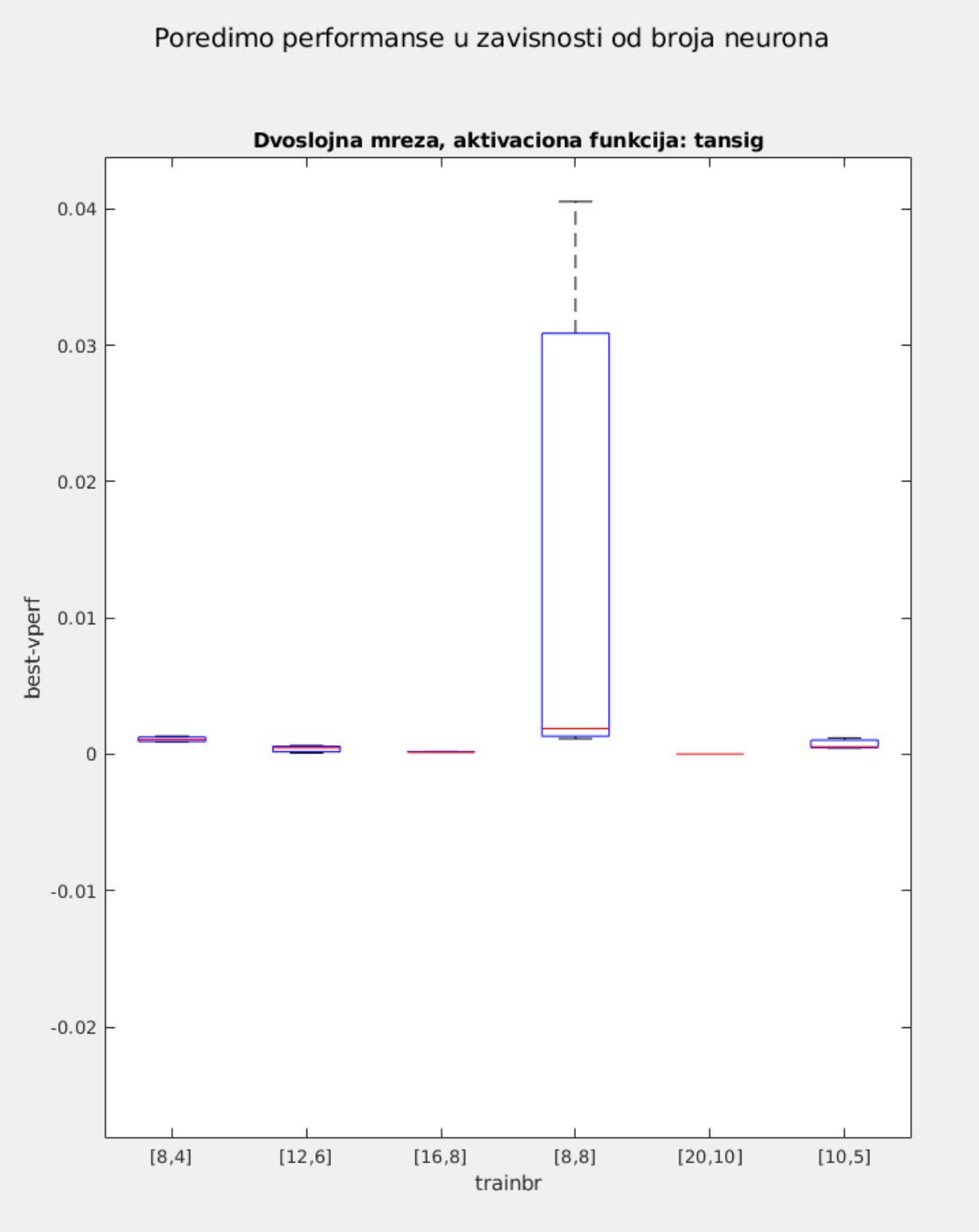
Слика 61: Поредба перформанси 4 типа мрежа из прве фазе обучавања

Може се видети да најбоље перформансе има мрежа која је користила ‘tansig’ активациону функцију и ‘trainbr’ алгоритам обучавања, те ће исти параметри бити кориштени у даљој потрази за оптималном мрежом.

У другој фази обучавања, испробано је више комбинација, са различитим бројем неурона у скривеним слојевима и упоређене су перформансе обучених мрежа (Слика 62 ,63 и 64)

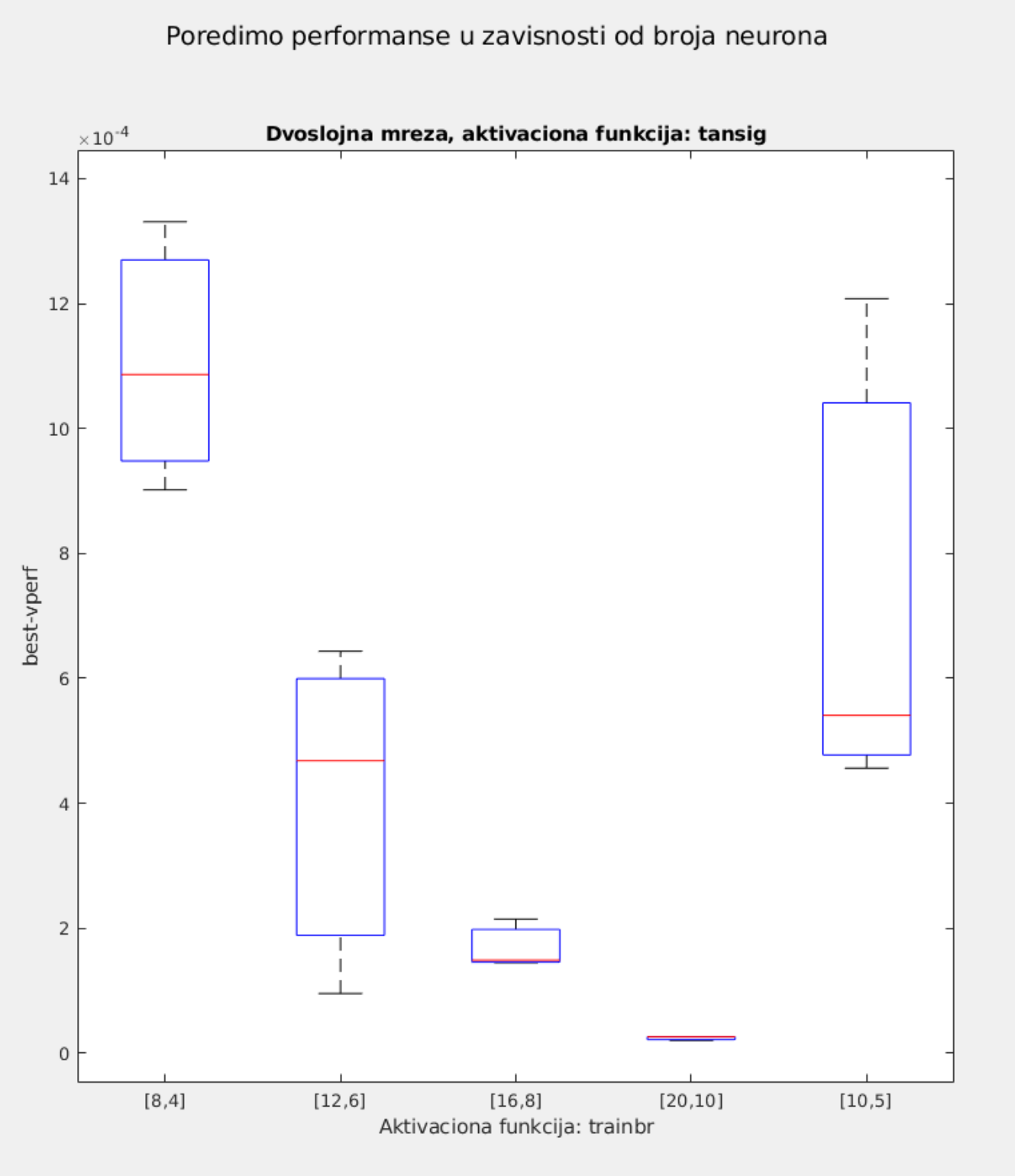
```
broj_neurona = {[8,4],[12,6],[16,8],[8,8], [20,10]};
for i =1:length(broj_neurona)
    t='tansig';
    for k=1:3
            net = newff(ulaz1,izlaz1,broj_neurona{i},{t,t,t},'trainbr');
            net.trainParam.show = 100;
            net.trainParam.lr = 0.5; % ne vazi za 'trainlm'
            net.trainParam.mu = 0.6 % parametar ucenja
            net.trainParam.epochs = 70;
            net.trainParam.goal = 1e-5;
            net.trainParam.max_fail=20000;
            net.divideParam.trainRatio = 0.7;
            net.divideParam.valRatio = 0.15;
            net.divideParam.testRatio = 0.15;
            [net,tr] = train(net,ulaz1,izlaz1,'useParallel','yes');

            validation(k)=tr.best_vperf;
            mreze_ukupno(k)={net};
            tr_ukupno(k)={tr};
    end
    [vred,poz] = min(validation);
    naj_mreza3{i} = mreze_ukupno{poz};
    naj_tr3{i} = tr_ukupno{poz};
    rezultati3(i,:) = validation;
end
```

Слика 62: Обучавање мрежа са различитим бројем неурона



Слика 63: Поређење перформанси мрежа са различитим бројем неурона

Избацујемо из разматрања комбинацију са [8, 8] скривеним слојевима, због лошијих перформанси.

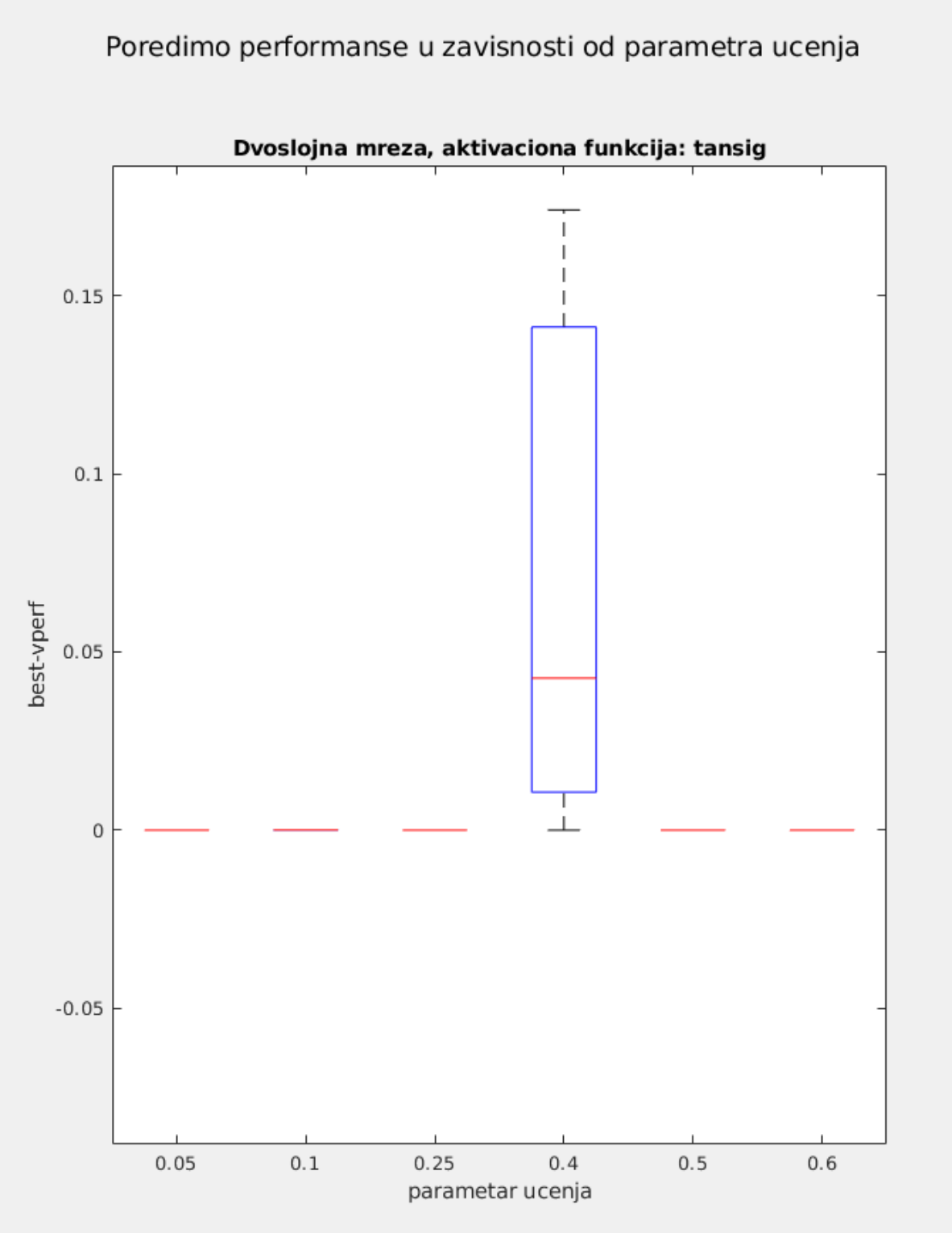


Слика 64: Поређење перформанси мрежа са слике 9 без комбинације [8,8]

Видимо да најбоље перформансе има мрежа са структуром скривених слојева [20, 10], те ћемо ту структури користити надаље у потрази за оптималном мрежом. За крај ћемо потражити и оптималан параметар учења. (Слика 65, 66,67 и 68)

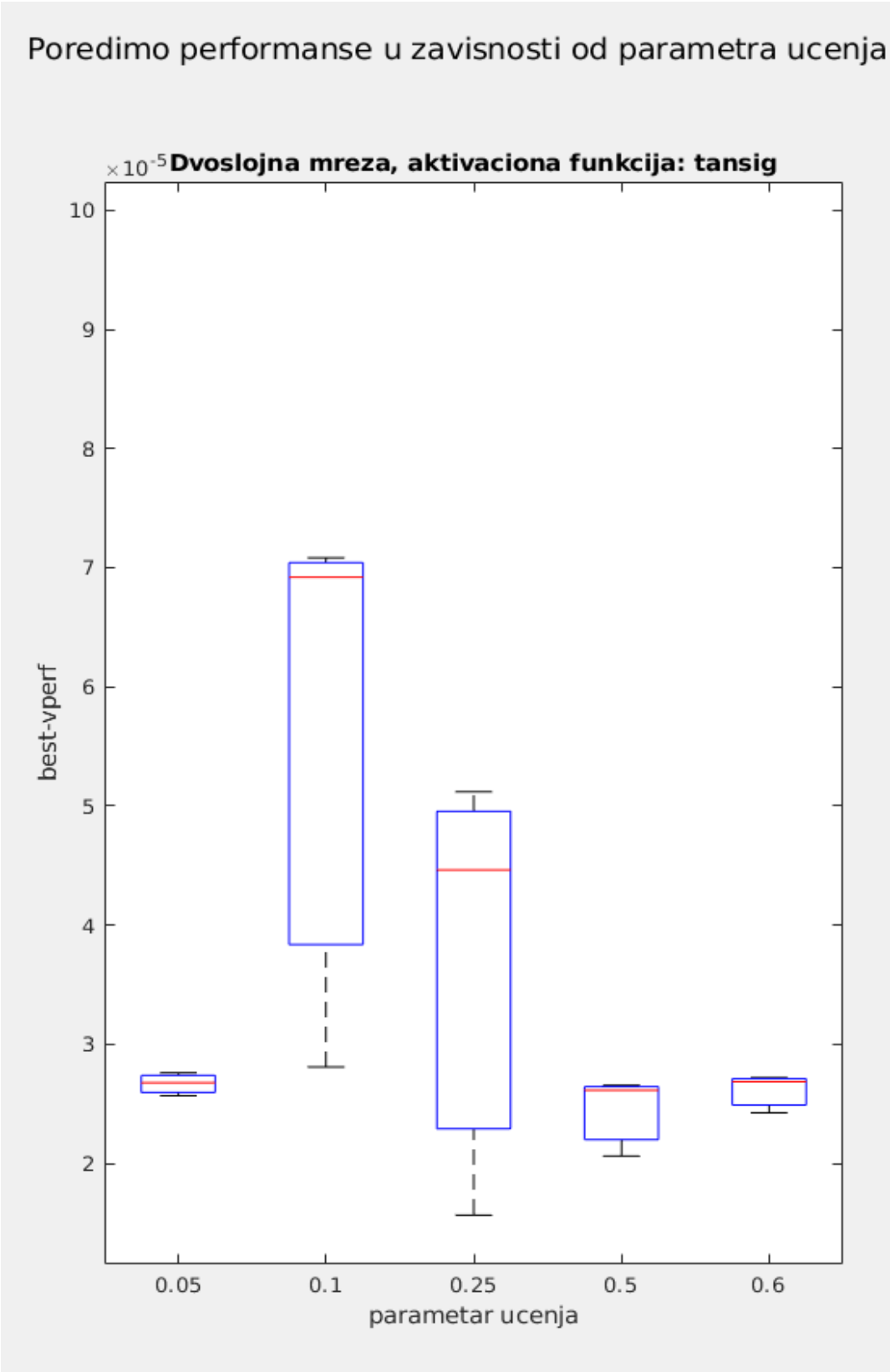
```
parametri = [0.05, 0.1, 0.25, 0.4, 0.6];
for i=1:length(parametri)
    t='tansig';
    for k=1:3
        net = newff(ulaz1,izlaz1,[20,10],{t,t,t},'trainbr');
        net.trainParam.show = 100;
        net.trainParam.lr = parametri(i); % ne vazi za 'trainlm'
        net.trainParam.mu = parametri(i); % parametar ucenja
        net.trainParam.epochs = 70;
        net.trainParam.goal = 1e-5;
        net.trainParam.max_fail=20000;
        net.divideParam.trainRatio = 0.7;
        net.divideParam.valRatio = 0.15;
        net.divideParam.testRatio = 0.15;
        [net,tr] = train(net,ulaz1,izlaz1,'useParallel','yes');

        validation(k)=tr.best_vperf;
        mreze_ukupno(k)={net};
        tr_ukupno(k)={tr};
    end

    [vred,poz] = min(validation);
    naj_mreza4{i} = mreze_ukupno{poz};
    naj_tr4{i} = tr_ukupno{poz};
    rezultati4(i,:) = validation;
end
```

Слика 65: Обучавање мрежа са различитим параметрима учења



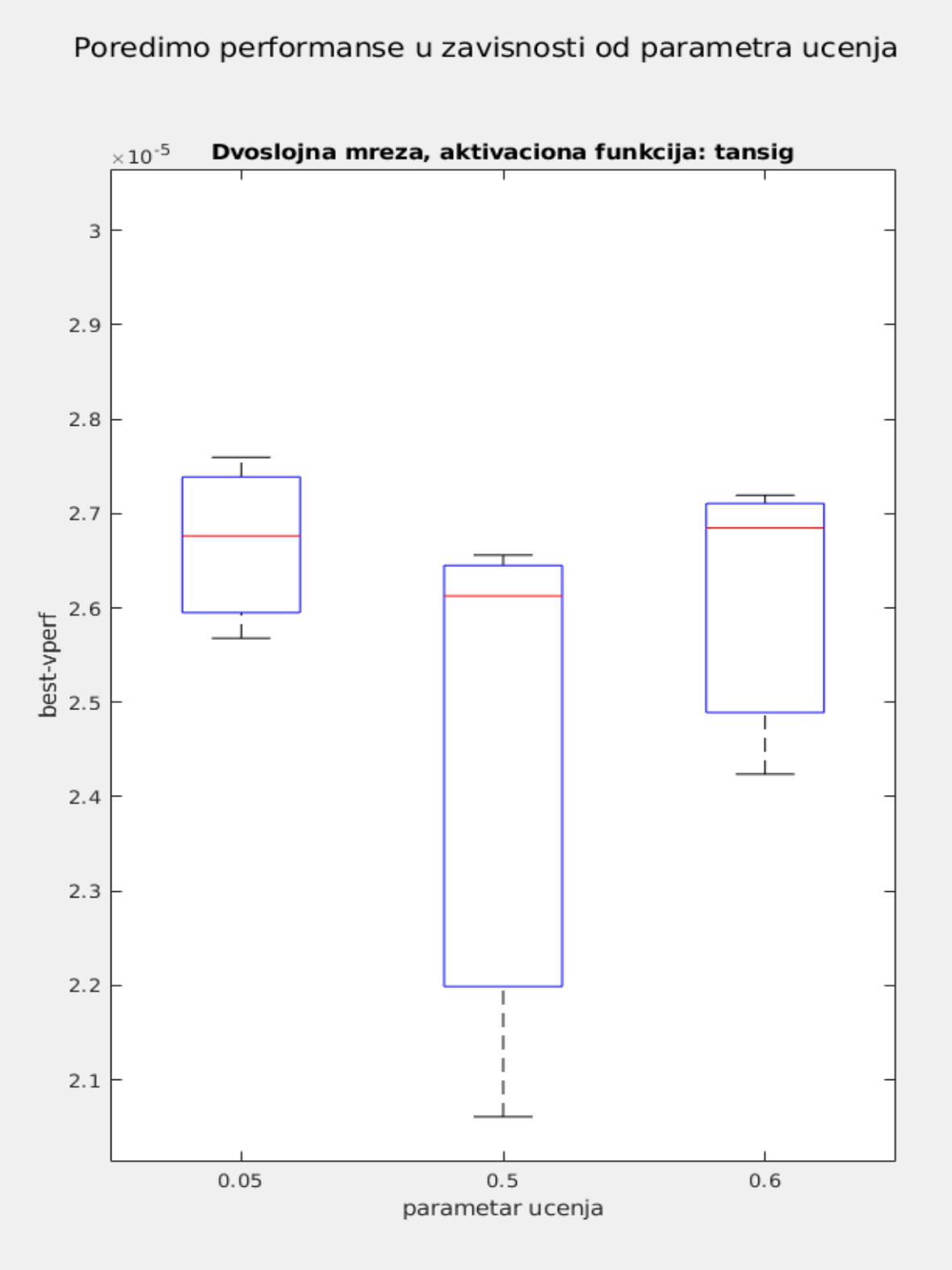
Слика 66: Поређење перформанси мрежа са различитим параметрима учења

Елиминшемо из разматрања комбинацију са 0.4 параметром учења због лошијих перформанси.



Слика 67: Ажурирана слика 9, без мреже са параметром учења 0.4

За крај приказујемо 'boxplot' са три кандидата за најбоље обучену мрежу.



Слика 68: Три кандидата за најбоље обучену мрежу

Закључак

Најбоља обучена мрежа има структуру скривених слојева [20,10], ‘tansig’ активационе функције и користила је ‘trainbr" алгоритам бучавања и параметар учења 0.5.

4.2 Мрежа друге функције

Концепцијско решење проблема

Користећи задату функцију, формирани су обучавајући парови. Улазни вектор у мрежу је тродимензиони елемент из домена функције, док се излаз из мреже пореди са сликом функције од тог елемента. Обучено је преко 100 мрежа са различитим параметрима: алгоритмом обучавања, бројем скривених слојева, бројем неурона, активационим функцијама у њима и параметрима учења. Бирана је најбоља мрежа, тако што су посматране вредности перформанси на валидационом скупу. Свака мрежа са одређеним, горе наведеним параметрима, тренирана је више пута са различитим иницијалним тежинима, како би се имао бољи увид у перформансе те врсте мреже.

Детаљан опис пројектног решења

Прво су генерисане вредности функција у свим тачкама домена, а затим су преобликовани, тј. секвенцирани, тако да вештачка неуронска мрежа може да их користи за обучавање. На слици 69, приказан је програмски код који је описао рачунару ове инструкције.

```
x = -5:0.2:5;

ulaz2_multidim = zeros(length(x),length(x),2);
izlaz2_multidim = zeros(length(x),length(x));


a = [-32 -16 0 16 32; 32 16 0 -16 -32];

for i1 = 1:length(x)
    for i2 = 1:length(x)
        xtr = [x(i1) x(i2)];
        ulaz2_multidim(i1,i2,:) = xtr;
        ypom = 1/500;
        for j = 1:5
            pom1=0;
            for i=1:2
                pom1 = pom1 + (xtr(i)-a(i,j))^6;
            end
            ypom = ypom + 1/(j+pom1);
        end
        izlaz2_multidim(i1,i2) = 1/ypom;
    end
end

ulaz2 = reshape(ulaz2_multidim, length(x)^2,2)';
ulaz2 = ulaz2(2:-1:1,:);
izlaz2 = reshape(izlaz2_multidim, length(x)^2,1)';
```

Слика 69. Код који генерише и спрема обучавајуће парове за мрежу

Следећи корак је био скалирање обучавајућих парова на интервал [0,1]. (слика 70)

```
for i = 1:2
    ulaz2(i,:) = (ulaz2(i,:)-min(ulaz2(i,:)))./(max(ulaz2(i,:))-min(ulaz2(i,:)));
end

izlaz2 = (izlaz2-min(izlaz2))./(max(izlaz2)-min(izlaz2));
```

Слика 70: Скалирање обучавајућих парова

Сада су обучавајућу парови спремни за тренирање вештачких неуронских мрежа. У првој фази обучавања је испробано више мрежа које имају [8, 4] структуру скривених слојева, алгоритам обучавања ‘trainlm’, параметар учења 0.5 и имају различите активационе функције у скривеним слојевима. Сваку мрежу истог типа тренирамо 10 пута, са различитим иницијалним тежинским

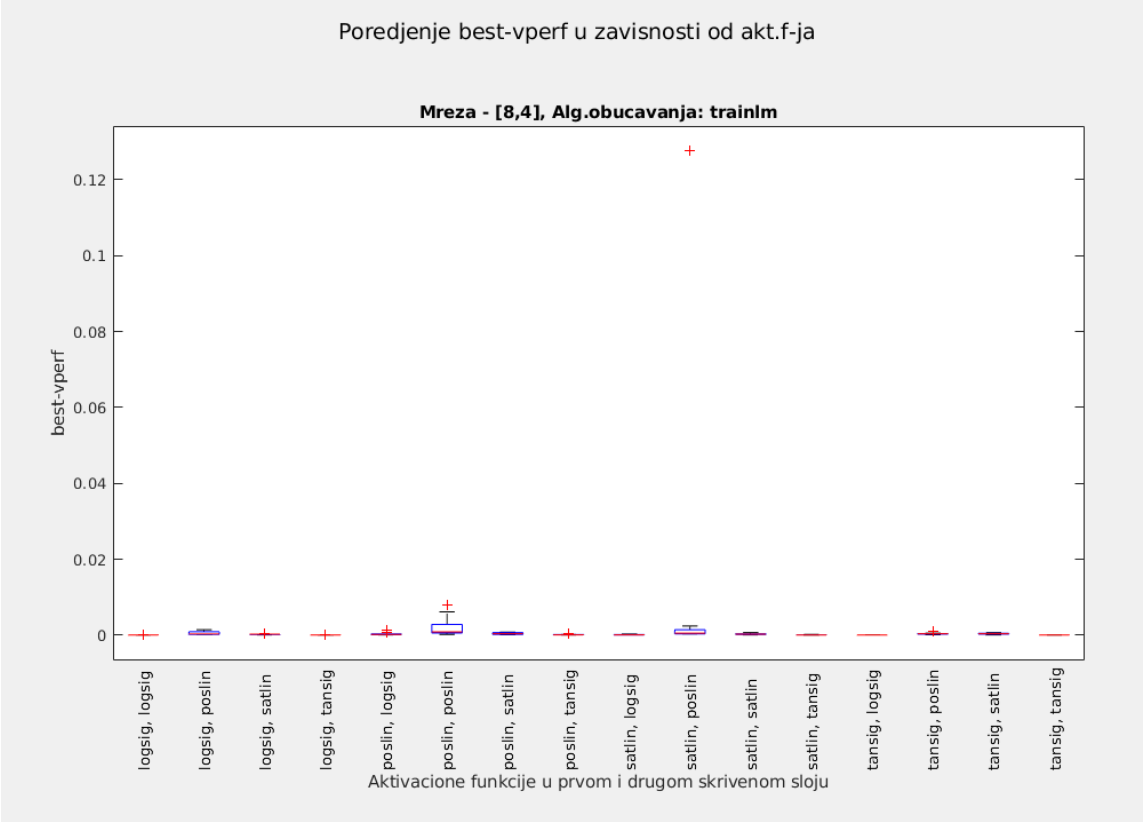
односима. (Слика 71.) Активационе функције су комбинације следећих функција: "logsig", "poslin", "satlin" и "tansig".

```
akt_fje = {"logsig", "poslin", "satlin", "tansig"};

for i = 1:length(akt_fje)
    for j =1:length(akt_fje)
        oznake((i-1)*4+j) = strcat(char(akt_fje{i}),", ",char(akt_fje{j}));
        for k = 1:10
            net = newff(ulaz2,izlaz2,[8, 4], {char(akt_fje{i}), char(akt_fje{j})}, 'trainlm');

            net.trainParam.show = 100;
            net.trainParam.lr = 0.5; % ne vazi za 'trainlm'
            net.trainParam.mu = 0.5; % parametar ucenja
            net.trainParam.epochs = 100;
            net.trainParam.goal = 1e-5;
            net.trainParam.max_fail=20000;
            net.divideParam.trainRatio = 0.7;
            net.divideParam.valRatio = 0.15;
            net.divideParam.testRatio = 0.15;

            [net, tr] = train(net,ulaz2,izlaz2);

            rezultati(k)=tr.best_vperf;
            sve_tr(k) = {tr};
            sve_net(k) = {net};
        end

        [vred,poz] = min(rezultati);
        naj_mreza{(i-1)*length(akt_fje)+j} = sve_net{poz};
        naj_tr{(i-1)*length(akt_fje)+j} = sve_tr{poz};
        svi_rez((i-1)*length(akt_fje)+j,:) = rezultati;
    end
end
```

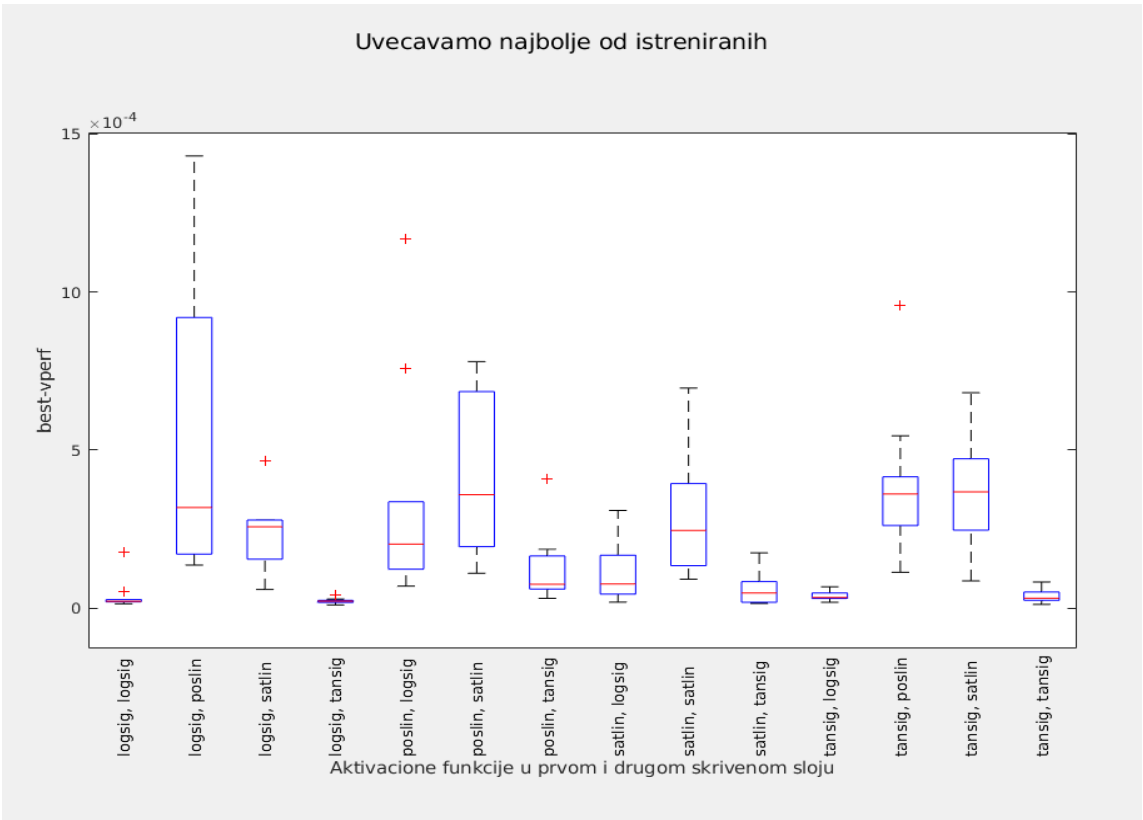
Слика 71: Прва фаза обучавања

Користимо функцију ‘boxplot’ за преглед перформанси обучених мрежа. Резултате мрежа у првој фази можемо видети на сликама 72, 73 и 74.

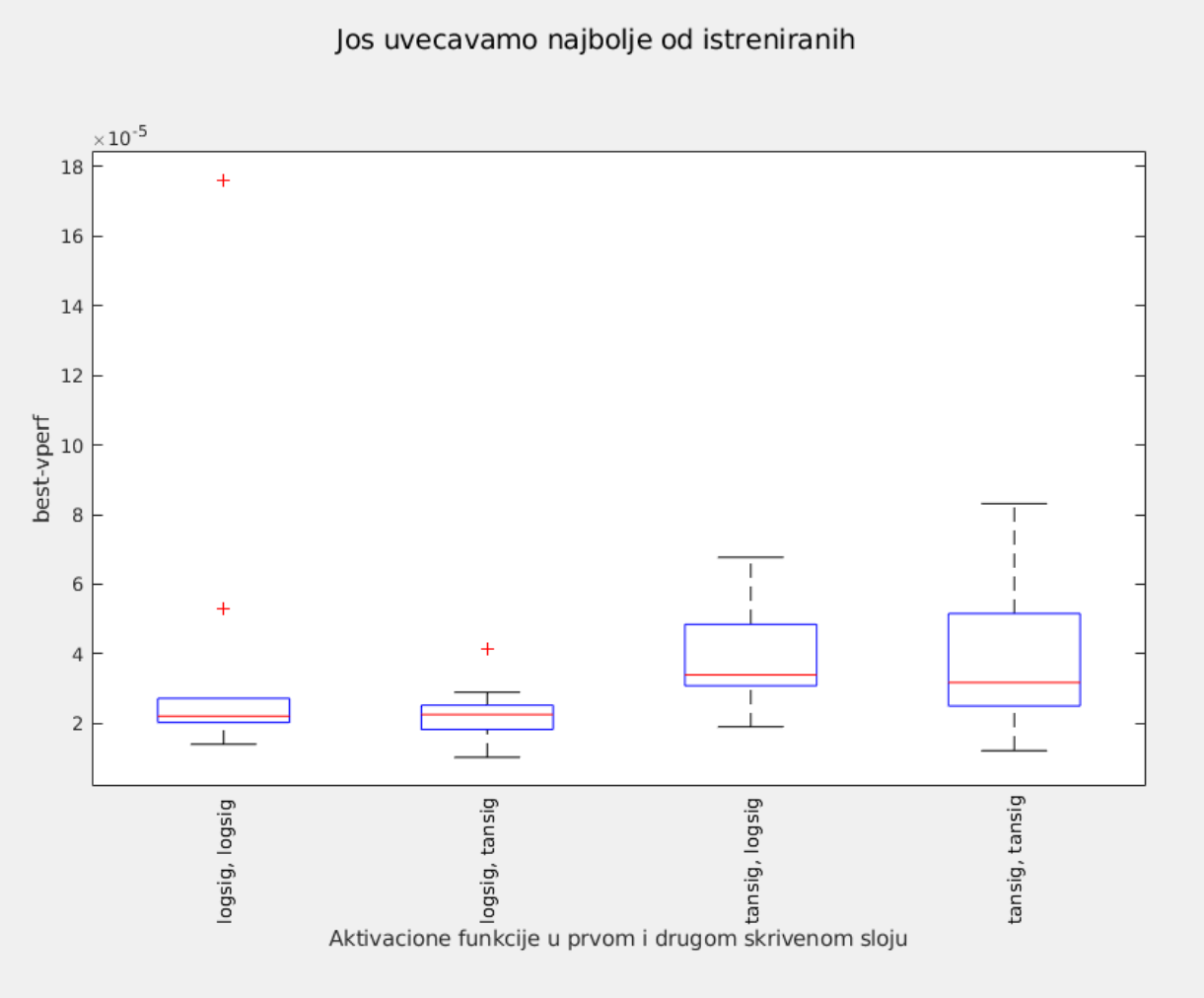
Постепеним елиминисањем мрежа са најлошијим перформансама, видимо да је најбоља мрежа она која има активационе функције ‘logsig’ и ‘tansig’ у првом и другом скривеном слоју, респективно.



Слика 72: Све перформансе мрежа из прве фазе

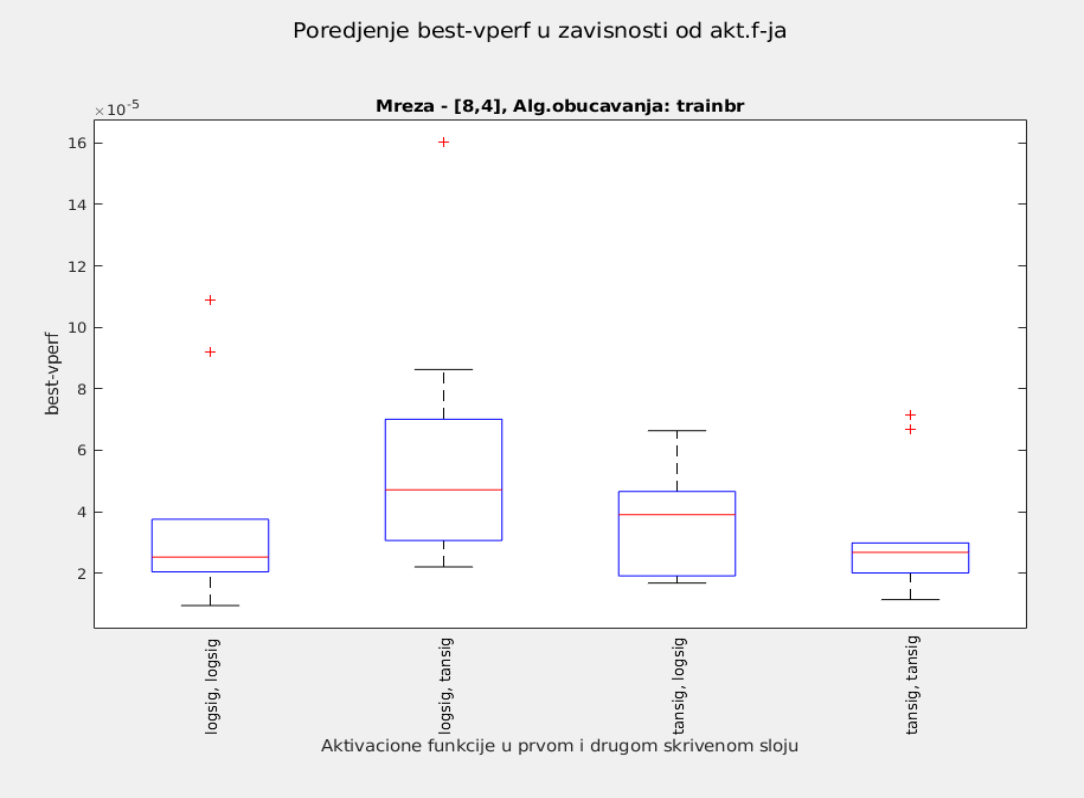


Слика 73: Одстрањујемо из разматрања најлошије мреже са слике 72



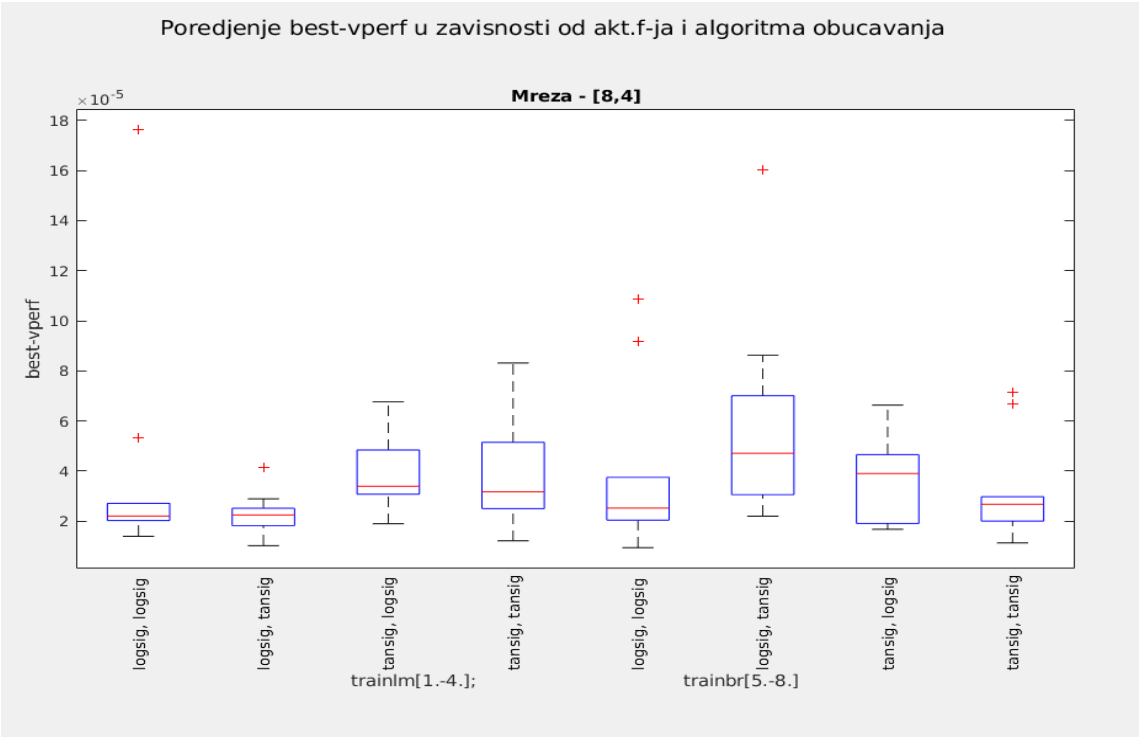
Слика 74: Одстрањујемо из разматрања најлошије мреже са слике 73

Види се да за овај проблем, најбоље резултате дају комбинације активационих фукнција ‘logsig’ и ‘tansig’, те се у даљој потрази оптималне мреже не користи више "poslin" и "satlin". Узимајући у обзир ово, у другој фази обучавања понавља се поступак, са ажурираним скупом активационих функција и алгоритмом обучавања ‘trainbr’. Видимо да је овај пут најбоља комбинација ‘tansig'-‘tansig’.Слика 10)



Слика 75: Резултати друге фазе обучавања

Упоређени су резултати са слике 74 и слике 75 у једном ‘boxplot’-у. (Слика 76)



Слика 76: Поређење перформанси алгоритама учења ‘trainlm’ и ‘trainbr’

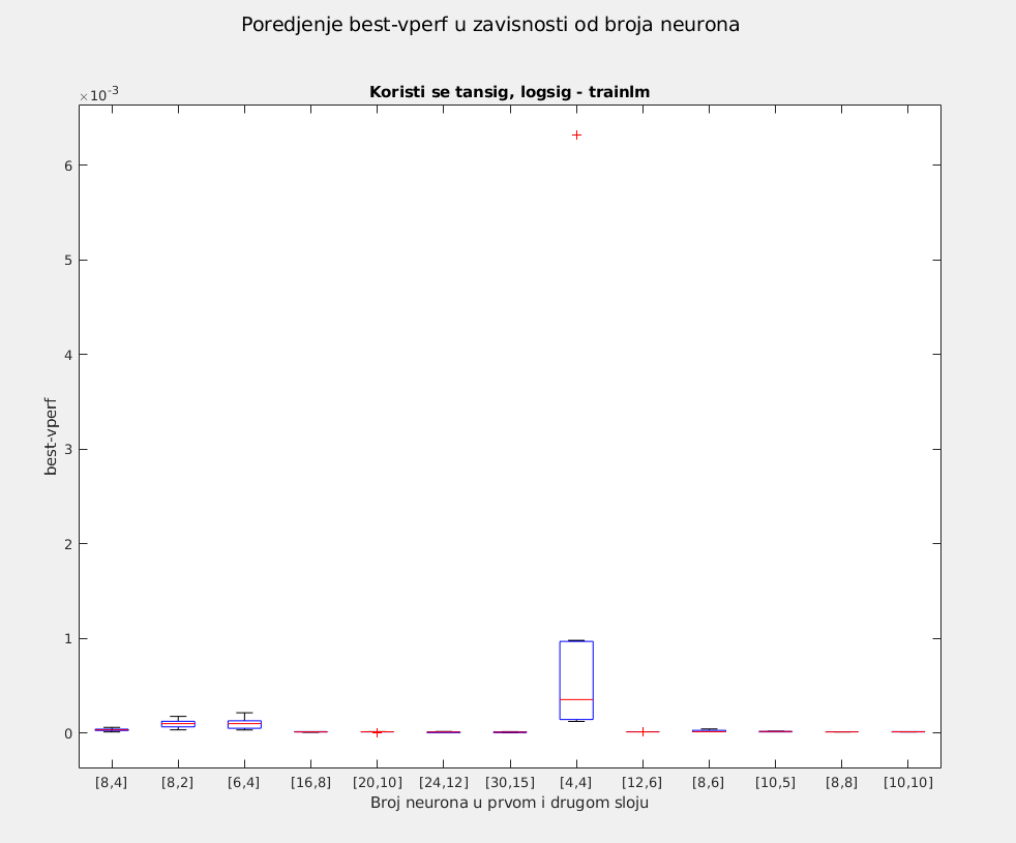
Види се да боље резултате даје ‘trainlm’ алгоритам обучавања, те ће се он користи у даљој потрази оптималне мреже. Најбоља пронађена мрежа за сада користи ‘logsig’, ‘tansig’ активационе функције и алгоритам учења ‘‘trainlm’.

У трећој фази је истраживан оптималан број неурона мреже са 2 скривена слоја. Одабран је скуп потенцијално добрих комбинација броја неурона у скривеним слојевима и са њим су обучене мреже. (Слика 77)

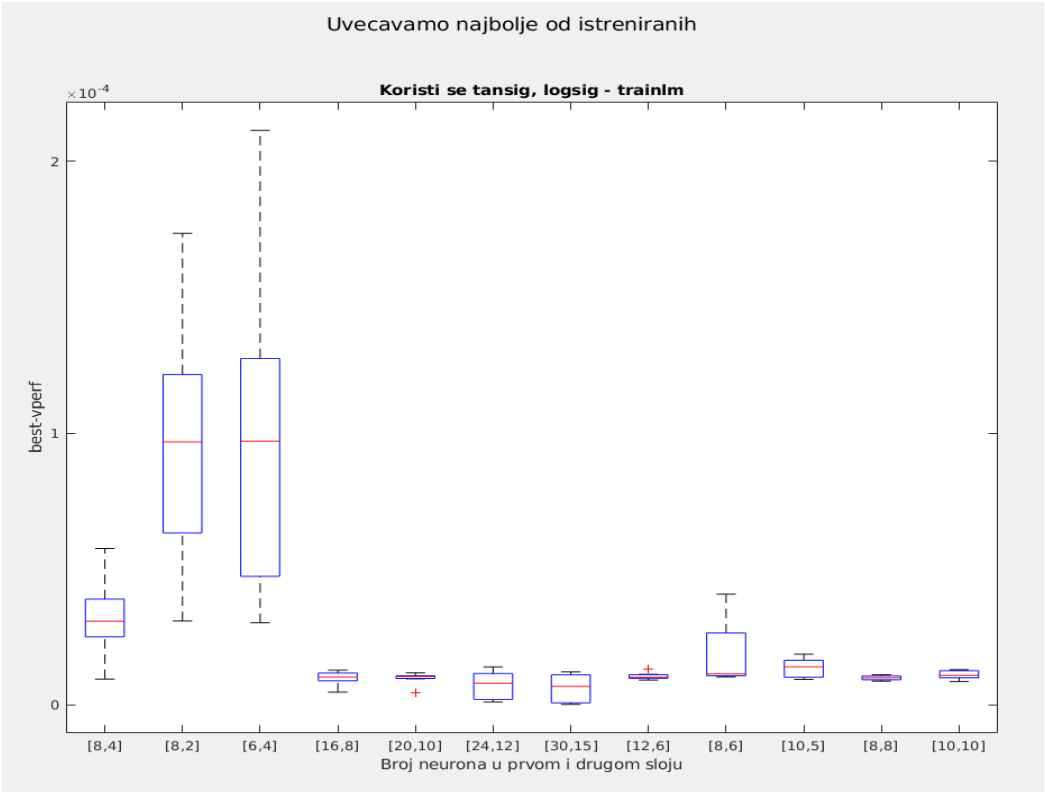
```
broj_neurona = {[8,4],[8,2],[6,4],[4,2], [4,4], [12,6]};

for l = 1:length(broj_neurona)
    oznake4(l) = broj_neurona(l);
    for k = 1:10
        net = newff(ulaz2,izlaz2,[8, 4, 2], {char(akt_fje{i}), char(akt_fje{j})}, 'trainlm');

        net.trainParam.show = 100;
        net.trainParam.lr = 0.5; % ne vazi za 'trainlm'
        net.trainParam.mu = 0.5; % parametar ucenja
        net.trainParam.epochs = 100;
        net.trainParam.goal = 1e-5;
        net.trainParam.max_fail=20000;
        net.divideParam.trainRatio = 0.7;
        net.divideParam.valRatio = 0.15;
        net.divideParam.testRatio = 0.15;

        [net, tr] = train(net,ulaz2,izlaz2);

        rezultati4(k)=tr.best_vperf;
        sve_tr4(k) = {tr};
        sve_net4(k) = {net};
    end

    [vred,poz] = min(rezultati4);
    naj_mreza4{l} = sve_net4{poz};
    naj_tr4{l} = sve_tr4{poz};
    svi_rez4(l,:) = rezultati4;
end
```

Слика 77: Тренирање мрежа са различитим бројем неурона у скривеним слојевима

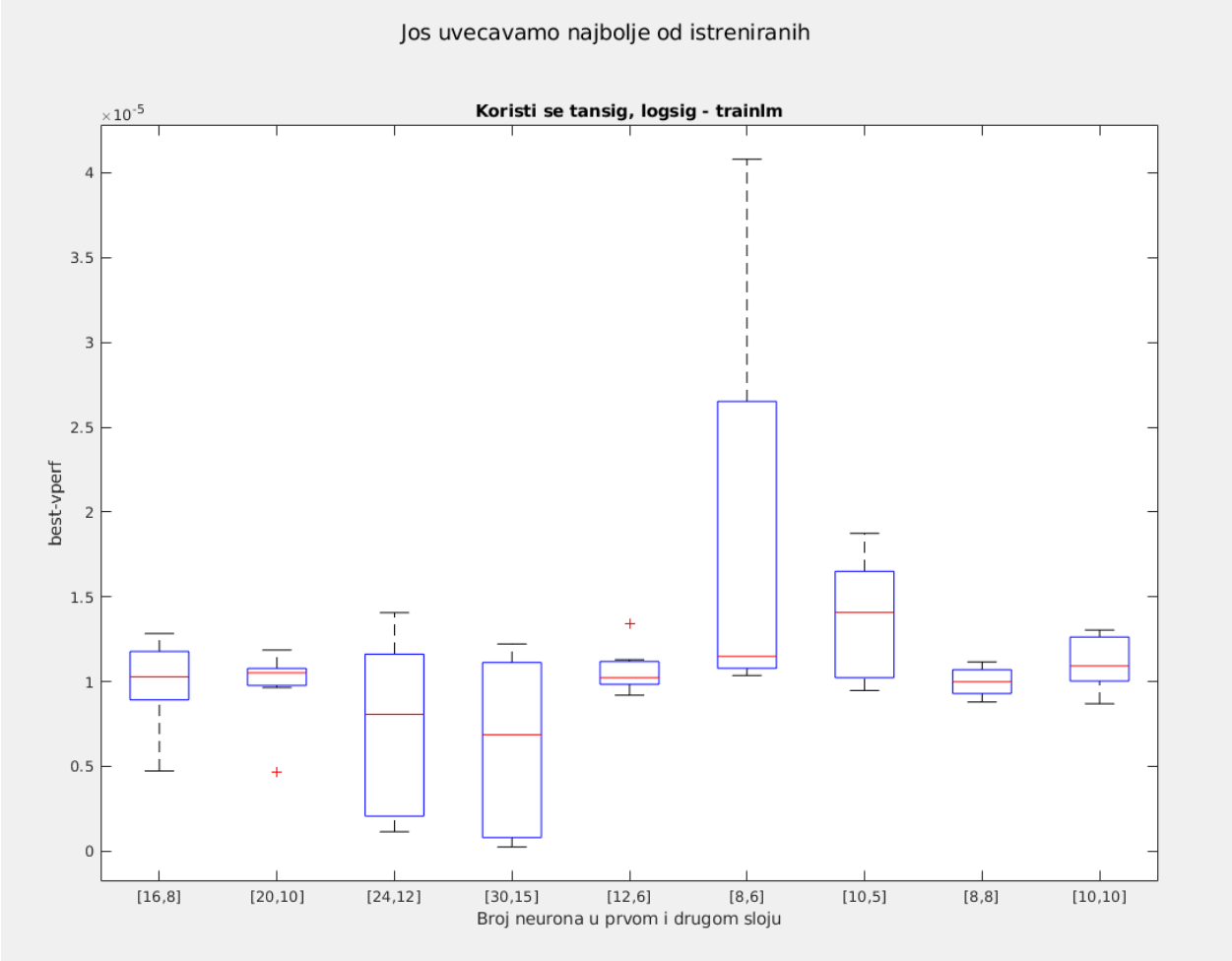
На сликама се 78, 79 и 80 се види потрага за најбољом мрежом из треће фазе обучавања.



Слика 78: Поређење перформанси мрежа са различитим бројем неурона



Слика 79: Елиминисање лоших мрежа са слике 78



Слика 80: Елиминисање лоших мрежа са слике 79

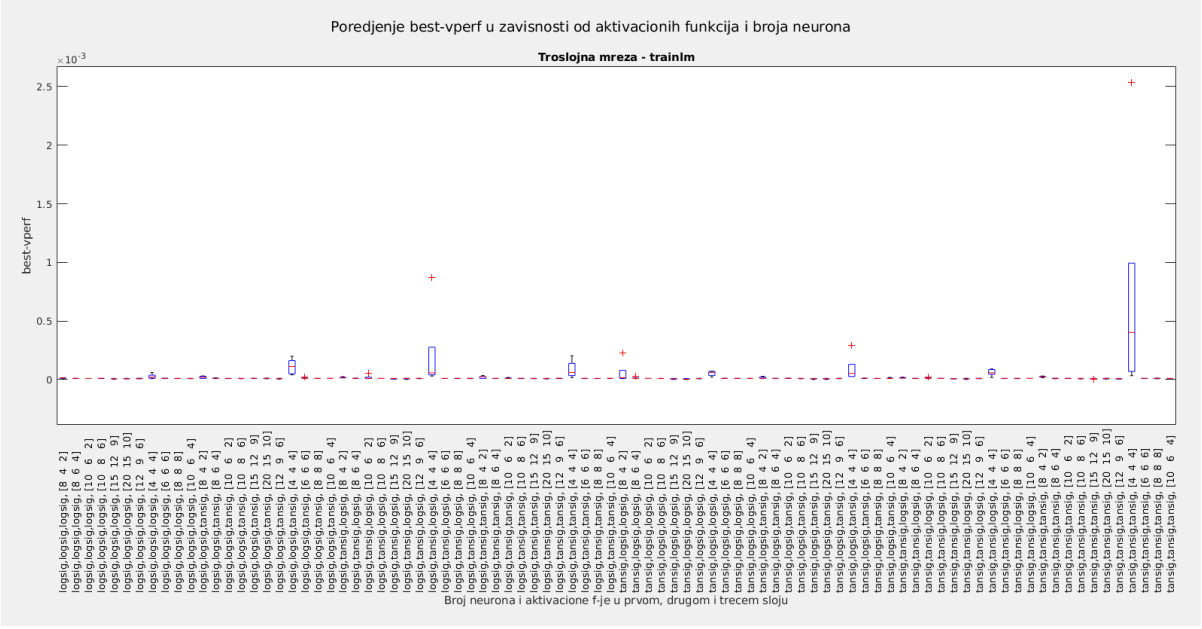
Можемо да уочимо да најбоља мрежа, пронађена за сад, има структуру скривеног слоја [30, 15], активационе функције ‘logsig’ и ‘tansig’ у првом и другом скривеном слоју редом и алгоритам обучавања ‘trainlm’.

У четвртој фази, тражи се оптимална мрежа са 3 скривена слоја. (Слика 81)

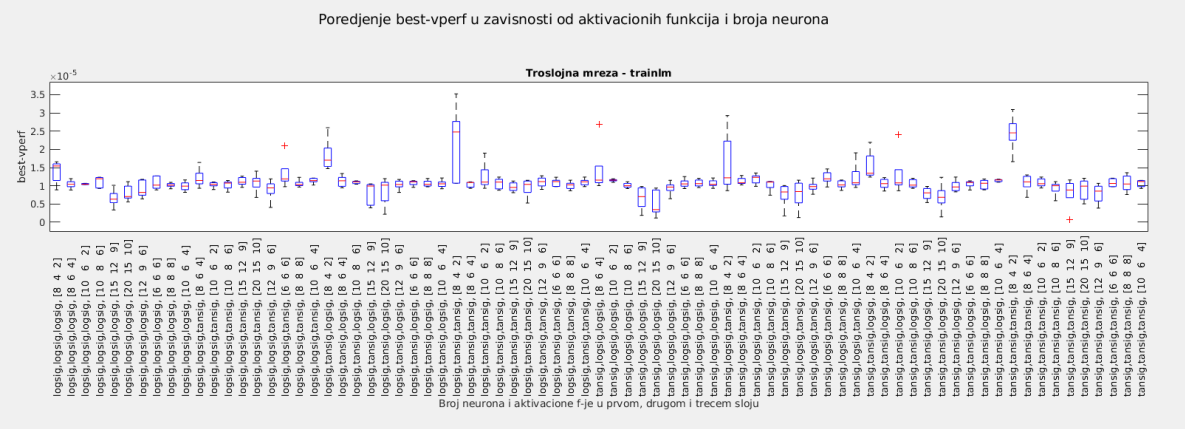
```
akt_fje = {"logsig","tansig"};
broj_neurona = {[8,4,2],[8,6,4],[10,6,2],[10,8,6],[15,12,9],[20,15,10] [12,9,6], [4,4,4],[6,6,6],[8,8,8],[10,6,4]};
for i = 1:length(akt_fje)
    for j = 1:length(akt_fje)
        for m = 1:length(akt_fje)
            for l = 1:length(broj_neurona)
                oznake6{(i-1)*2*2*length(broj_neurona)+(j-1)*2*length(broj_neurona)+ (m-1)*length(broj_neurona)+l} =...
                    strcat(char(akt_fje{i}),', ',char(akt_fje{j}),', ',char(akt_fje{m}),', [',num2str(broj_neurona{l}),']');
                for k = 1:5
                    net = newff(ulaz2,izlaz2,broj_neurona{l}, {char(akt_fje{i}),char(akt_fje{j}),char(akt_fje{m})}, 'trainlm');
                    net.trainParam.show = 100;
                    net.trainParam.lr = 0.5; % ne vazi za 'trainlm'
                    net.trainParam.mu = 0.5; % parametar ucenja
                    net.trainParam.epochs = 100;
                    net.trainParam.goal = 1e-5;
                    net.trainParam.max_fail=20000;
                    net.divideParam.trainRatio = 0.7;
                    net.divideParam.valRatio = 0.15;
                    net.divideParam.testRatio = 0.15;

                    [net, tr] = train(net,ulaz2,izlaz2);
                    rezultati6(k)=tr.best_vperf;
                    sve_tr6(k) = {tr};
                    sve_net6(k) = {net};
                end

                [vred,poz] = min(rezultati6);
                naj_mreza6{(i-1)*2*2*length(broj_neurona)+(j-1)*2*length(broj_neurona)+ (m-1)*length(broj_neurona)+l} = sve_net6{poz};
                naj_tr6{(i-1)*2*2*length(broj_neurona)+(j-1)*2*length(broj_neurona)+ (m-1)*length(broj_neurona)+l} = sve_tr6{poz};
                svi_rez6((i-1)*2*2*length(broj_neurona)+(j-1)*2*length(broj_neurona)+ (m-1)*length(broj_neurona)+l,:) = rezultati6;
            end
        end
    end
end
```

Слика 81: Обучавање мрежа у четвртој фази

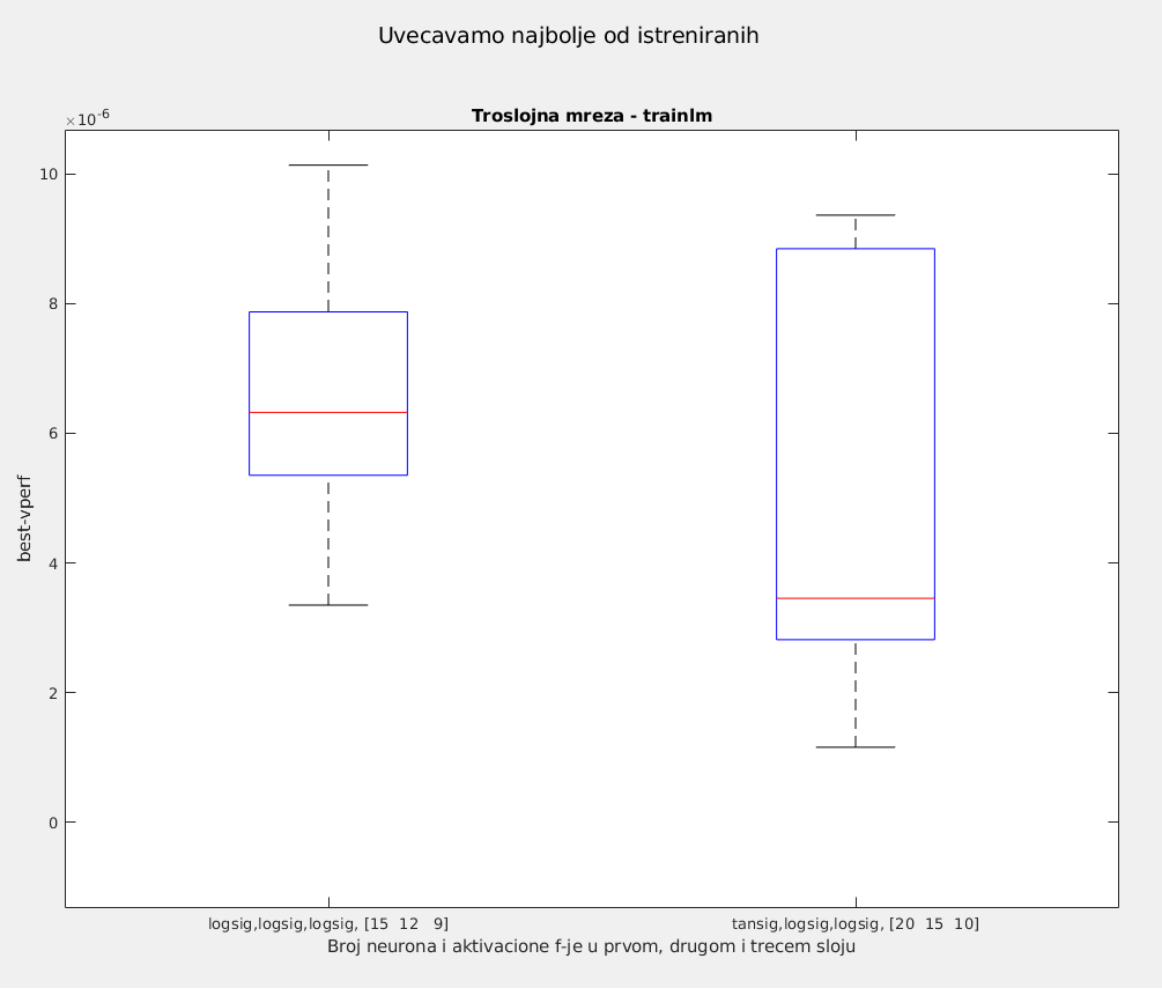
На сликама 82, 83 и 84 види се претрага најбоље од свих обучених мрежа у четвртој фази.



Слика 82: Приказ перформанси обучених мрежа у четвртој фази



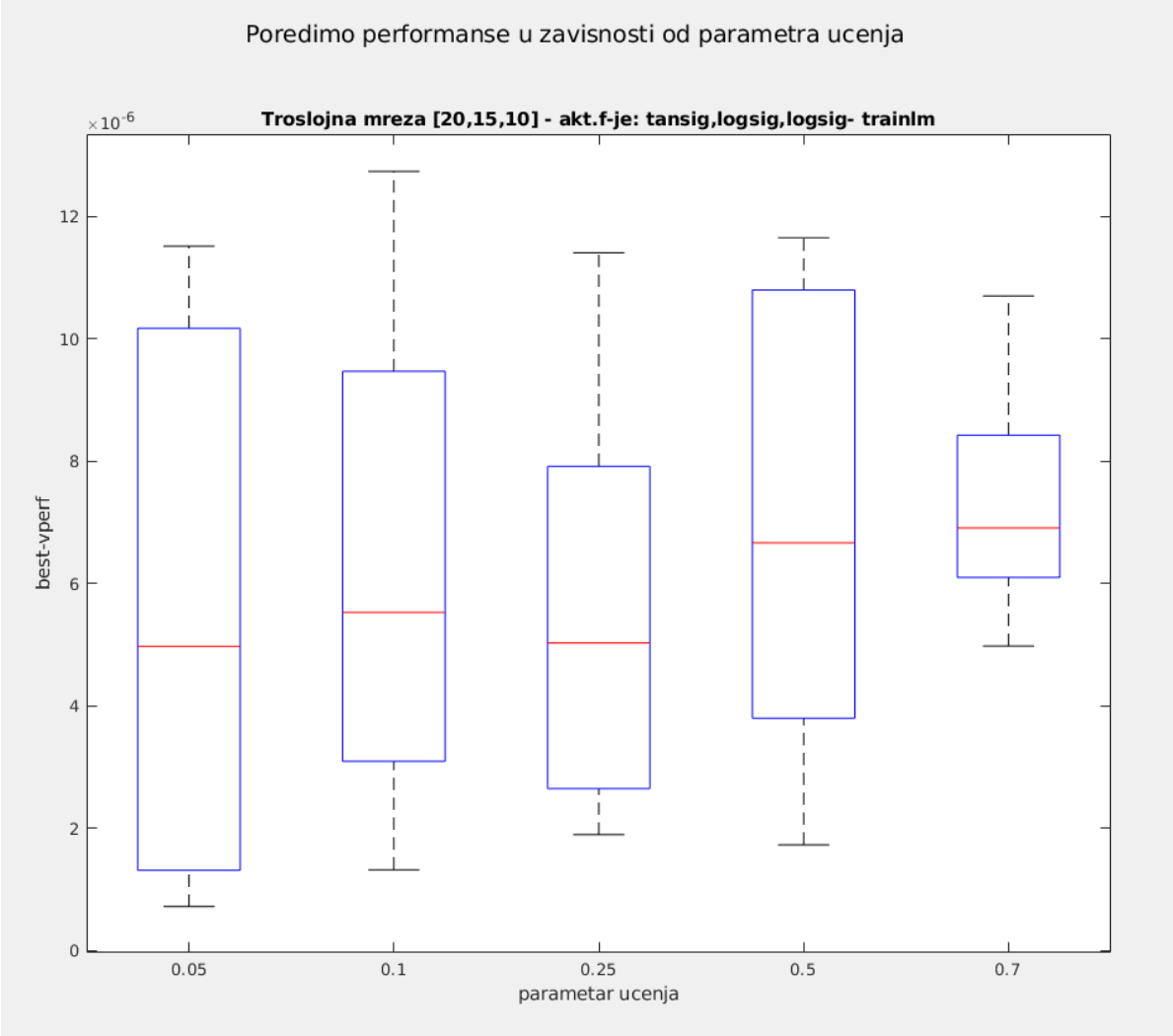
Слика 83: Елиминишу се најлошије мреже са слике 82



Слика 84: Две најбоље мреже са слике 83

Видимо да је најбоља мрежа са 3 слоја користила број неурона у слојевима - [20, 15, 10], активационе функције {‘tansig’, ‘logsig’ ,‘logsig’} и алгоритам учења ‘trainlm’.

У наставку ће се још испитати оптималан параметар учења за ову трослојну мрежу. (Слика 85)



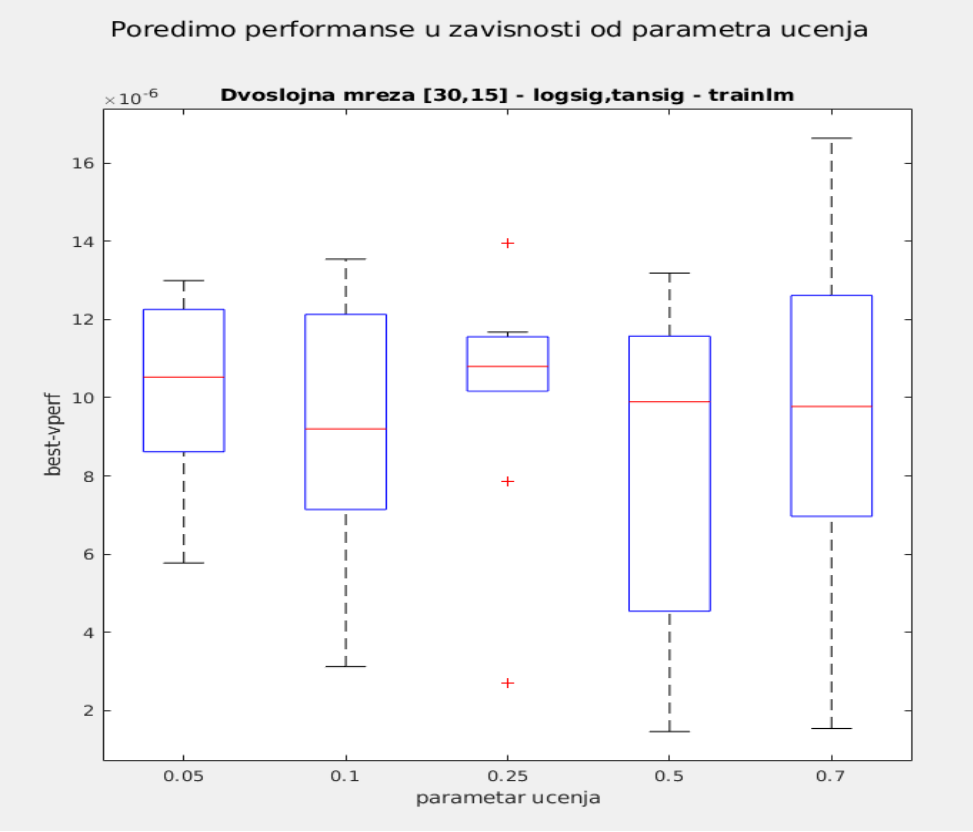
Слика 85: Перформансе најбоље трослојне мреже у зависности од параметра учења

Види се са слике 85 да је најбоља обучена трослојна мрежа користила параметар учења 0.25.

Проверава се најбољи параметар учења за најбољу двослојну мрежу.(Слика 86 и слика 87)

```
parametri = [0.05, 0.1, 0.25, 0.5, 0.7];
for i = 1:length(parametri)
    for k=1:10
        net = newff(ulaz2,izlaz2,[30,15], {'logsig','tansig'}, 'trainlm');

        net.trainParam.show = 100;
        net.trainParam.lr = parametri(i); % ne vazi za 'trainlm'
        net.trainParam.mu = parametri(i); % parametar ucenja
        net.trainParam.epochs = 100;
        net.trainParam.goal = 1e-5;
        net.trainParam.max_fail=20000;
        net.divideParam.trainRatio = 0.7;
        net.divideParam.valRatio = 0.15;
        net.divideParam.testRatio = 0.15;

        [net, tr] = train(net,ulaz2,izlaz2,'useParallel','yes');

        rezultati8(k)=tr.best_vperf;
        sve_tr8(k) = {tr};
        sve_net8(k) = {net};
    end
    [vred,poz] = min(rezultati8);
    naj_mreza8{i} = sve_net8{poz};
    naj_tr8{i} = sve_tr8{poz};
    svi_rez8(i,:) = rezultati8;
end
```

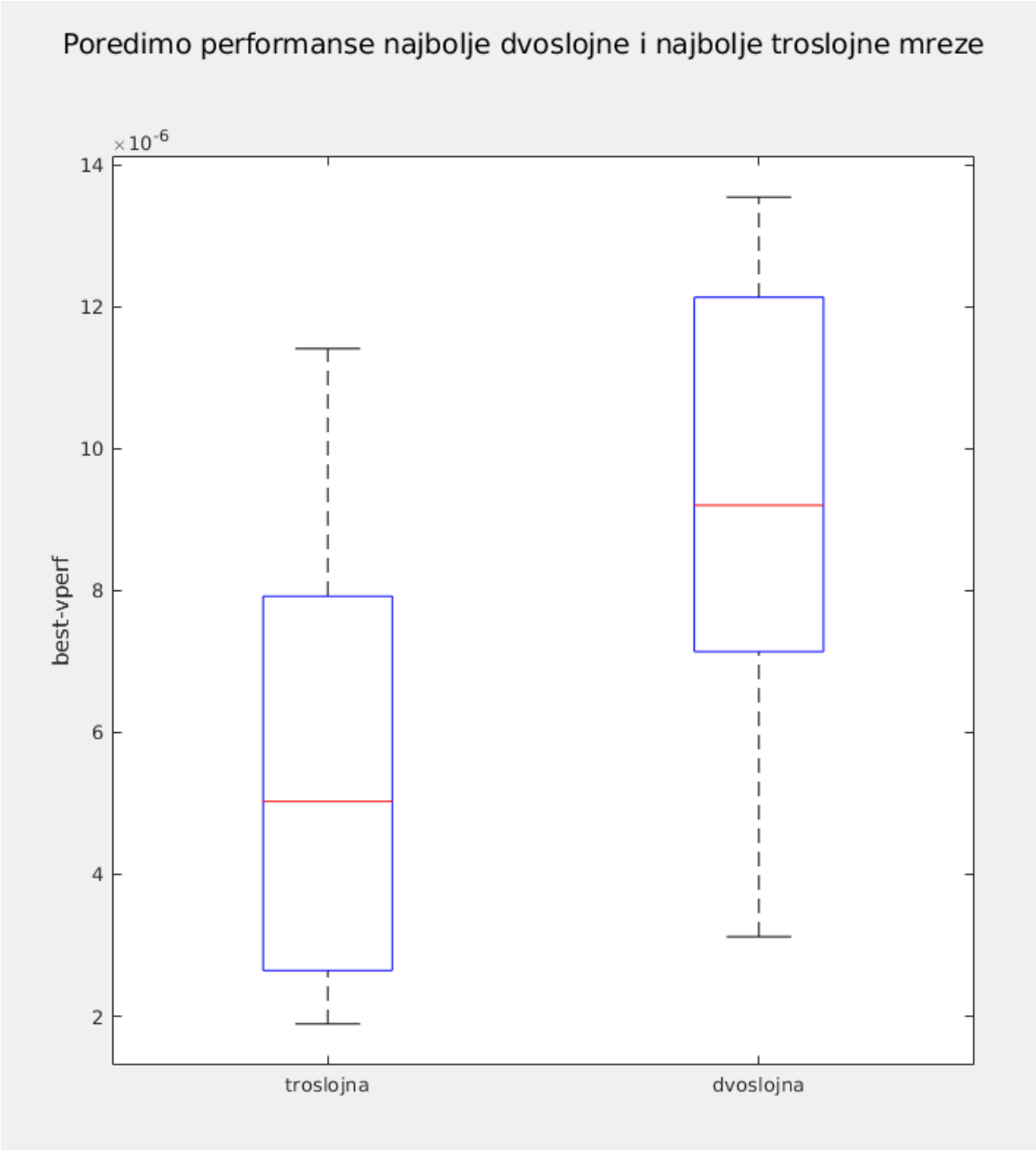
Слика 86: Обучавамо најбољу двослојну мрежу са различитим параметрима учења



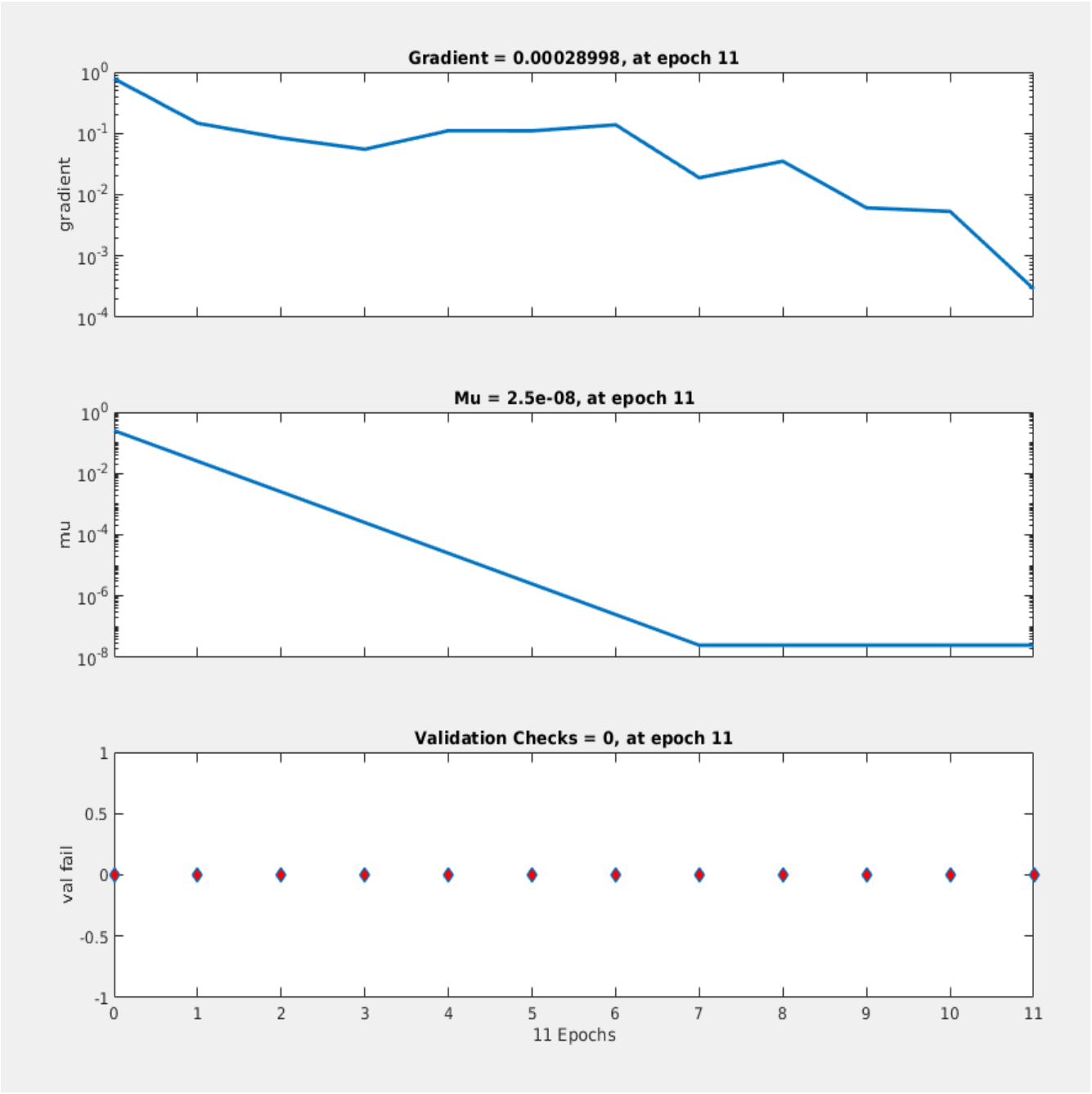
Слика 87: Поредимо најбољу двослојну мрежу са различитим параметрима учења

Види се да је најбоље обучена за вредност параметра учења 0.1.

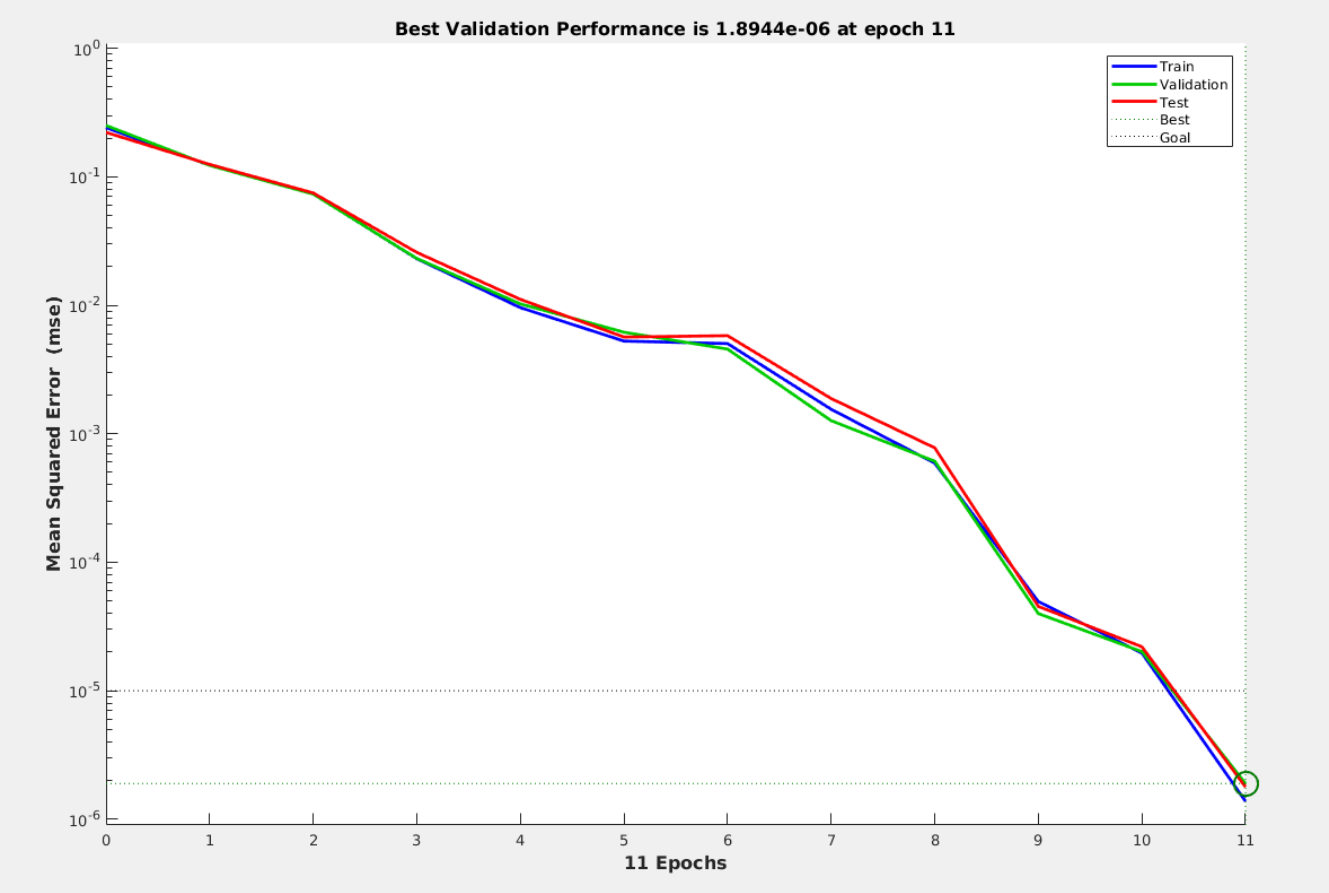
За крај се пореди најбоља трослојна мрежа и најбоља двослојна мрежа. (Слика 88.)



Слика 88: Поредимо перформансе најбоље трослојне мреже и најбоље довслојне мреже



Слика 89. Графици за најбољу мрежу



Слика 90. График грешке током обучавања најбоље мреже

Закључак

Пошто најбоља трослојна мрежа има боље перформансе од најбоље двослојне мреже, њу бирамо за оптималну мрежу. Дакле најбоља обучена мрежа има структуру скривених слојева [20,15,10], активационе функције {'tansig', 'logsig', 'logsig'} и користила је алгоритам обучавања 'trainlm' и параметар учења 0.25.

Закључак

Након рада овог пројектног задатка имамо јаснију слику о комплексности трансформације реалног света у поимање робота. Посебан изазов у томе представља стварање „памети", тј. учења робота да предвиђа различите исходе и прилагођава њима своје понашање на најоптималнији начин. Можемо закључити да ће се човек све више ослањати на употребу машина са развијеном перцепцијом, за обављање најразличитијих интелектуално захтевних послова. Наш посао је да то обезбедимо, али не заборављајући потребе осталих људи.

Литература

[1] Миљковић, З., Славковић, Н., Петровић, М., Предавања на предмету Роботика и вештачка интелигенција, Универзитет у Београду - Машински факултет, Београд, 2020.

[2] Миљковић, З., Алексендрић, Д., Вештачке неуронске мреже - Збирка решених задатака са изводима из теорије, Универзитет у Београду - Машински факултет, Београд, 2018.

[3] http://www.mathworks.com/

[4] https://stackexchange.com/

[5] Michael Nielsen, Neural Networks and Deep Learning- free online book, Dec 2019

[6] K. Kipli et al., Performance of Levenberg-Marquardt Backpropagation for Full Reference Hybrid Image Quality Metrics- 2012 - (google schoolar)