УНИВЕРЗИТЕТ „ГОЦЕ ДЕЛЧЕВ“ – ШТИП

ФАКУЛТЕТ ЗА ИНФОРМАТИКА

ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И РОБОТИКА

Штип

Ѓорѓи Владимиров

РЕШЕНИЕ НА ПРОБЛЕМОТ НА ИНВЕРЗНА КИНЕМАТИКА НА РОБОТСКА РАКА СО ПОМОШ НА ANFIS

– МАГИСТЕРСКИ ТРУД –

Штип, јуни, 2020 година

Комисија за оценка и одбрана

Ментор:	**Сашо Коцески** **Проф. д-р, Факултет за информатика** **Универзитет „Гоце Делчев“ Штип**
Член:	**Владо Гичев** **Проф. д-р, Факултет за информатика** **Универзитет „Гоце Делчев“ Штип**
Член:	**Васко Кокаланов** **Доц. д-р, Факултет за информатика** **Универзитет „Гоце Делчев“ Штип**

Датум на одбрана: _____

Датум на промоција: _____

Рецензирани и објавени трудови

[1] Vladimirov Gjorgji and Koceski Saso (2019) *Inverse Kinematics Solution of a Robot Arm based on Adaptive Neuro Fuzzy Interface System.* International Journal of Computer Applications, 178 (39). pp. 10-14. ISSN 09758887

Решение на проблемот на инверзна кинематика на роботска рака со помош на AFIS

КРАТОК ИЗВАДОК:

Роботските раце се дизајнираат и се имплементираат како самоуправувачки роботи или пак се дел од уште посложени системи. Дизајнот и структурата на роботската рака зависи од повеќе фактори како што се тежината на работата што треба да ја врши, околината во која ќе работи и итн. Без разлика на дизајнот и структурата на роботската рака, секоја роботска рака се стреми кон претходно анализирано и планирано движење. Во овој процес се анализираат директна и инверзна кинематика. Директна кинематика е кога се одредува крајната положба на роботската рака, кога се познати параметрите за ротација и транслација. Инверзна кинематика е кога треба да се одредат параметрите за ротација и транслација за да се дојде до дадената, крајна положба. Во овој магистерски труд е преставено решение за решавање на проблемот на инверзна кинематика на роботска рака со помош на Adaptive Neuro Fuzzy Interface System (ANFIS). Направена е евалуација врз предложениот модел и резултатите се презентирани.

КЛУЧНИ ЗБОРОВИ:

Кинематика, инверзна кинематика, директна кинематика, DH параметри, ANFIS (Adaptive Neuro Fuzzy Interface System), Fuzzy logic, звоно членска функција, Гаусова членска функција, последователни и теоретски параметри, threejs, DOF.

Inverse kinematics solution of a robot arm based on ANFIS

ABSTRACT:

Nowadays, robot arms are used as standalone or intrinsic part of many robot systems in various fields and applications. The design and structure of robot arms varies depending on multiple constraints such as the tasks they have to perform, working environment in which they have to operate, the dimensions of the objects they have to peak, etc. Every robot arm, regardless of its design and structure, is aimed to perform some movement that has to be carefully analyzed and planned. During this process, usually two types of motion are analyzed. The first one aims at finding the position of the end effector when the angles between the robot arm links are known. This problem is usually denoted as direct kinematics. The second one, known as inverse kinematics, aims at solving the opposite problem i.e. to determine the angles between links when the position of the end effector is known. This paper presents an inverse kinematics solution of two degrees of freedom planar robot arm based on Adaptive Neuro Fuzzy Interface System (ANFIS). The proposed model is experimentally evaluated and the obtained results are discussed.

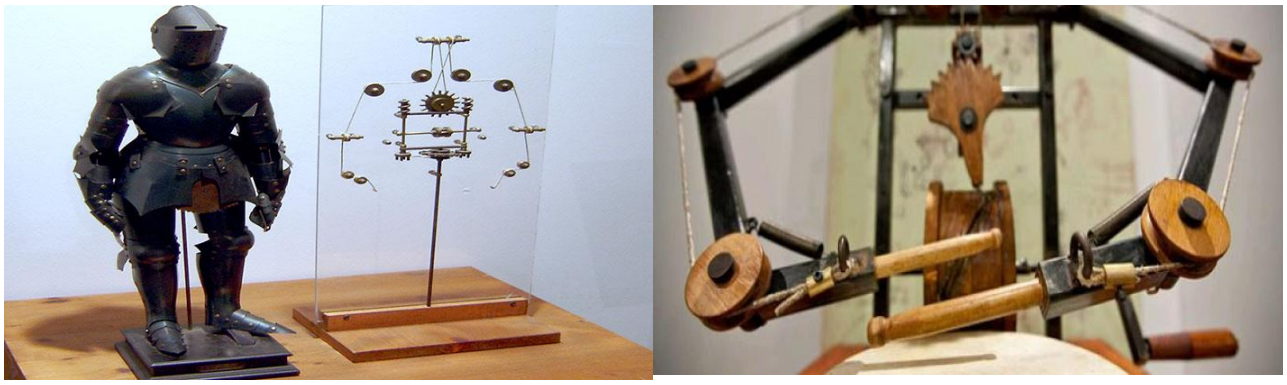
KEY WORDS:

Kinematics, Inverse kinematics, Direct kinematics, DH parameters, ANFIS (Adaptive Neuro Fuzzy Interface System), Fuzzy logic, Generalized bell function, Gaussian membership function, consequent and premise parameters, Cartesian space, Quaternion space, threejs, DOF.

Содржина

1. Вовед

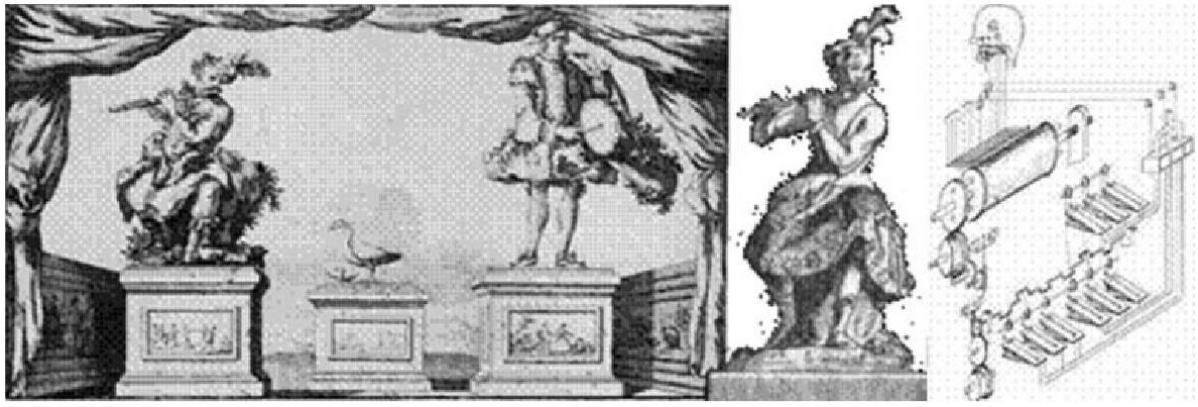
Уште од самите почетоци на роботиката, роботските раце се дизајнираат и имплементираат како самоуправувачки роботи или пак се дел од уште посложени системи. Историски гледано, првата софистицирана роботска рака ја има дизајнирано Леонардо да Винчи во 1495 година. Роботот бил составен од два независни дела. Долниот дел бил за долните екстремитети и биле со 3 DOF (степени на слобода), за колкот, коленото и стапалото, додека горниот дел бил за горните екстремитети - рацете.



Слика1 – Роботот на Леонардо да Винчи од 1495 год.

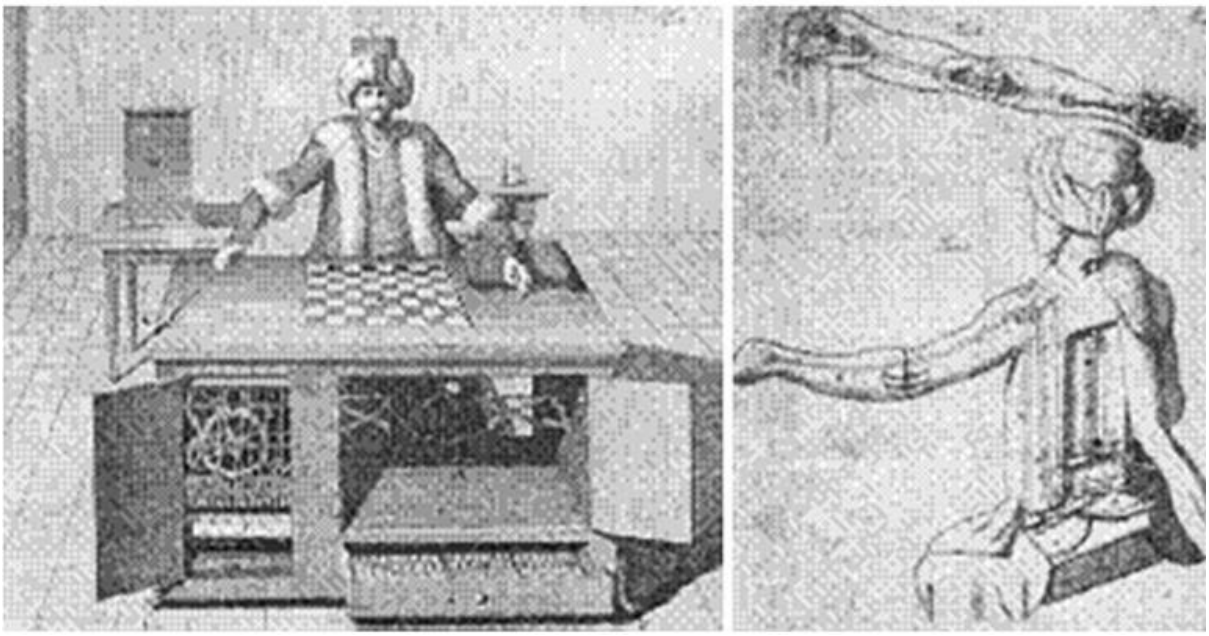
Раката била дизајнирана од 4 независни дела (4 зглоба), поврзана со аналоген контролер кој ја давал изворната енергија и во кој се наоѓал програмираниот дел од раката. Контролерот се наоѓал во градите на самиот робот и бил само за контролирање на рацете, додека долните екстремитети биле напојувани со надворешен извор.

Индустриската ера била следната искра за почетоците на роботиката и роботските раце. Џек де Вакасон бил талентиран механичар во 18-ти век. Роден е во 1709 година. Во 1738, тој дизајнирал автоматски свирач на флејта и го нарекол „Андроиде“. Во наредната година тој создал и автоматски тапанар. Ќе се разгледа свирачот на флејта бидејќи роботските раце се обработуваат во овој труд. Карактеристично за него е тоа што прстите на раката се мрдале како на човековата дланка.



Слика2 – Автоматски свирач на флејта

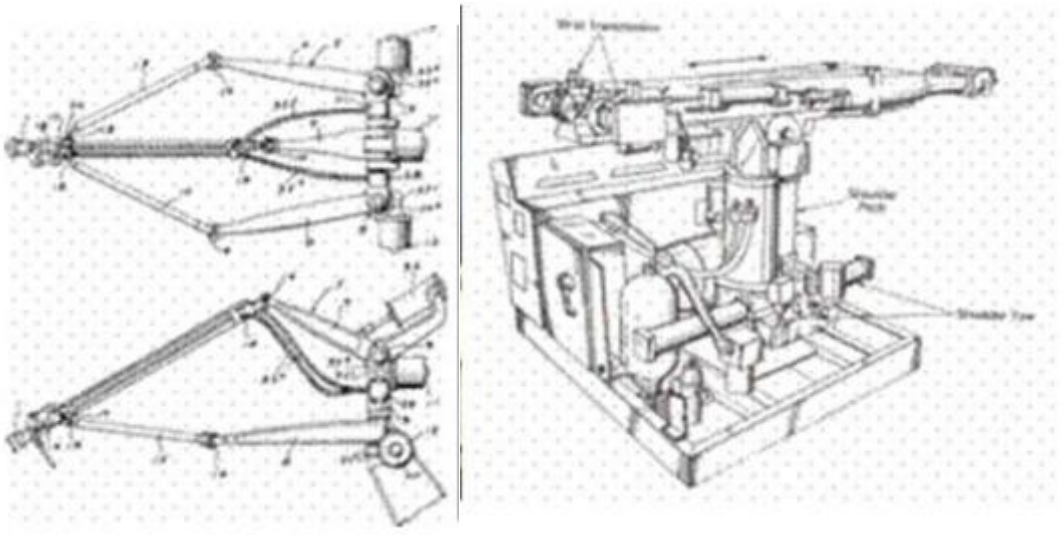
Друга, исто така софистицирана роботска рака - е левата рака на роботот за играње шах, дизајнирана во 1769 година, познат под името Турк (Турчин). Роботот бил изработен од дрво и можел да ја мрда главата, да ги затвора капаците на очите, но левата рака била многу покомплексна и сложена. Била контролирана од страна на „директор“, кој бил лоциран внатре во градниот кош. Секој прст си имал свои серии на кабли кои биле конектирани со „директорот“. Роботот Турк е прикажан на слика 3.



Слика 3 – Турк, роботската рака за играње шах

Првите современи роботски раце пред сè се користеле во индустријата, односно во производството и нивната задача била едноставно земање и преместување на производите од производната лента. Инспирација за овој напредок инженерите земале од флексибилноста на човековата рака.

Со пронаоѓањето на транзисторите и интегралните кола, дошло до голем напредок, не само во производството, туку и во научно-истражувачката дејност, односно се зголемила желбата за вештачка интелигенција. Полард има дизајнирано автоматска роботска рака која би се користела во индустријата за автоматско бојадисување. Овој дизајн никогаш не почнал да се употребува во индустријата. Харлод Руселунд создал слична роботска рака за автоматско бојадисување и тој додека работел за Де Вилбис ја имплементирал. Овие две роботски раце биле контролирани од страна на електрични контролори, што не е случај во претходните примери. Сепак и овие раце не биле толку многу совршени. *Unimate* во 1962 година ја претставил првата индустриска роботска рака, која била инсталирана во фабриката Генерал моторс. Роботската рака била дизајнирана од страна на Џорџ Девол и како и претходните роботски раце и оваа имала улога да бојадисува. Од оваа рака биле продадени 8500 парчиња. На слика 4 се прикажани дизајните на роботската рака на Полард и *Unimate*.



Слика4 – Првите индустриски роботски раце; Роботската рака на Полард (лево), Роботската рака на *Unimate* (десно)

Сега, роботските раце заземаат сè поголем замав и нивната употреба е проширена во повеќе сектори и во повеќе случаи. Ова не е резултат само на ниската цена на денешните материјали за дизајнирање на роботи, но и како резултат на развивањето на патерните за контрола како и развивањето на вештачката интелигенција (Artificial Intelligence AI). Зголемената популарност на вештачката интелигенција и нејзината имплементација во разни полиња [1], почнувајќи од туризам [2], роботика [8-11], како и во економијата [12], се резултат на техники и модели за имитација на човековата разумност, вештина на учење и можноста за подобрување со тек на времето.

Кога станува збор за истражување и развивање на нешто во роботиката, директната и инверзната кинематика се најчестите проблеми со кои може да се соочат инженерите. Кинематика е наука која го проучува движењето на телата, но притоа без да се земе предвид силата и моментот нанесена врз телото. Проблемот на директната кинематика е врската помеѓу зглобовите на роботската рака и позицијата и ориентацијата на самата роботска рака. Кажано со други зборови, проблемот на директната кинематика е откривање на позицијата и ориентацијата на роботската рака кога се дадени параметрите на зглобовите на раката.

Спротивно на проблемот на директната кинематика е проблемот на инверзната кинематика. Проблемот на инверзната кинематика може да се опише како откривање на параметрите на зглобовите (како што се аглите помеѓу два дела) кога е позната позицијата и ориентацијата на роботската рака.

Во овој труд најпрво ќе се разгледаат претходните научни истражувања и подоцна ќе се анализираат и имплементираат двете техники за решавање на проблемот на инверзна кинематика (математички и ANFIS) врз роботска рака со два зглоба која дејствува во 2Д простор. За процесот на имплементација и симулација на овие техники ќе биде развиена веб-апликација во .NET јазикот. Како завршна фаза, ќе се направи евалуација на решенијата и точноста на решавање на проблемот на инверзна кинематика со помош на експерименти во кои однапред се зададени траектории на движење на роботската рака.

2. Преглед на претходни научни истражувања

Решавањето на проблемот на директна кинематика е едноставен и не е воопшто комплициран за разлика од решавањето на проблемот на инверзната кинематика. Решавањето на проблемот на инверзната кинематика обично одзема многу време и не секогаш ги дава очекуваните резултати [13, 14].

Разни методологии за решавање на проблемот на инверзната кинематика на роботска рака се презентираат низ литературата која е достапна. Тие можат да се поделат во три главни категории: алгебарски [15], геометриски [16] и итеративни [17]. Проблемот со алгебарскиот метод е што не може секогаш да гарантира за резултатот. Геометрискиот метод може да гарантира за првите три зглоба, додека итеративниот нема да работи за недефинирани и може да конвергира до единечно решение во зависност од почетната положба.

Од друга страна, вештачката интелигенција и нејзините методи нудат многу голема флексибилност, заради можноста за унапредување и приближните решенија. Невронските мрежи и генетските алгоритами во комбинација се имплементирани за решавање на проблемот на инверзна кинематика на шестзглобен *Stanford* роботски манипулатор во [18]. Овде предложениот хибриден пристап ги користи најдобрите работи од невронските мрежи и техники за поголема прецизност на решенијата.

Во [19] авторите, предлагаат решение на проблемот на инверзна кинематика врз роботска рака користејќи продолжен генетски алгоритам (continuous genetic algorithm). Алгоритамот се состои од три фази и тоа: иницијализација, вкрстување и мутирање. Дизајниран е да дава мазни патеки на зглобовите и сепак тие да бидат одлично прецизни низ целиот координатен систем. Авторите нашле дека глаткоста најмногу зависи од првата фаза, односно иницијализацијата. Овде се докажува дека продолжениот генетски алгоритам дава подобри перформанси (време на извршување и конвергенција) од стандардниот генетски алгоритам.

Решение на проблемот на инверзна кинематика како во хаотична така и средина без пречки е презентиран во [20] каде што се користи бидирекционален метод на оптимизација. Идејата за бидирекционалено

пребарување е за да се забрза конвергенцијата во однос на традиционалното пребарување. Евалуација е направена на предложениот алгоритам со извршени експерименти на роботска рака со четири степени на слобода и резултатите се очекуваните.

Во [21] алгоритам т.н. летачки оган (firefly) е искористен за да се реши проблемот на инверзна кинематика на роботска рака. Овој пристап започнува со познат модел на директна кинематика и подоцна во алгоритмот firefly овој модел е искористен и се генерираат итеративни движења на зглобовите. Подоцна директната кинематика се користи за пресметување на релативните позиции во координатниот простор. Всушност firefly алгоритамот се користи за минимизирање на функцијата за пресметување на растојанието помеѓу добиените и очекуваните позиции. Врз овој алгоритам е направена евалуација со помош на дводелен и триделен систем. Резултатите потврдуваат конвергенција на сто извршени тестови.

Прецизно решение на проблемот на инверзна кинематика користејќи невронски мрежи е предложен во [22]. Авторите предлагаат дизајн на контролер базиран на невронска мрежа во Декартов координатниот систем. Овој дизајн покажува подобрување на перформансите во некои делови.

Постојат многу апликации во секојдневниот живот во кој има имплементирано роботска рака составена од два дела. Овој труд преставува решение на роботска рака составена од два дела и два зглоба која дејствува во единечна рамнина. Имајќи предвид дека е едноставен дизајн често се користи за тестирање на технологиите кои ќе ја подобрат работата на раката. Една таква технологија е во пневматските вештачки мускули (Pneumatic artificial muscles, PAMs). PAMs припаѓаат во групата неконвенционални управувачи со значаен сооднос помеѓу силата и тежината која може да се искористи за конструирање мек механизам кој може да биде во допир со човекот. За овој тип на проблем Adaptive Neural Fuzzy Interface System (ANFIS) е предложено како можно решение за инверзна кинематика [23].

Дизајн и имплементација на роботска рака со два DOF (Degrees Of Freedom) за учење на деца како да пишуваат и како да ги подобрат нивните способности за цртање е презентиран во [24]. Крајната траекторија и движење се

испланирани користејќи повеќесегментни равенки кои што се базирани на Adaptive Neuro-Fuzzy Interface System (ANFIS). Два ANFIS контролери се дизајнирани за моделирање на PMW директно. Влезовите на секој контролер се координатите коишто ја преставуваат посакуваната буква, додека излезите се две команди за Pulse Width Modulation (PMW) сервомоторот со кој ќе се активираат зглобовите на роботската рака.

Употребата на ANFIS не се ограничува само при имплементација на роботи и роботски системи, туку може да се искористи и при предвидување на квалитетот на водата [25]. Авторите, овде, ја истражуваат реката Брахмани и според pH вредноста, WQI (water quality index) и металите кои ги содржи на одредени места близу градовите и индустриите, со помош на ANFIS се предвидува колкав е квалитетот на водата на одредено место и дали е добар за консумирање од страна на човекот. Учењето на мрежата, овде, се врши со хибриден алгоритам и резултатите од експериментите се прикажани користејќи MATLAB.

3. Методи на решавање на проблемот на инверзна кинематика на роботска рака

3.1. Кинематика

Кинематика е наука која го проучува движењето на телата, но притоа без да се земе предвид силата или моментот нанесена врз тоа тело. Телото се движи ако ја промени положбата во однос на друго тело. Тоа друго тело се нарекува **референтно тело**, а доколку на тоа референтно тело се придружи и координатен систем тогаш тоа се нарекува **референтен систем**.

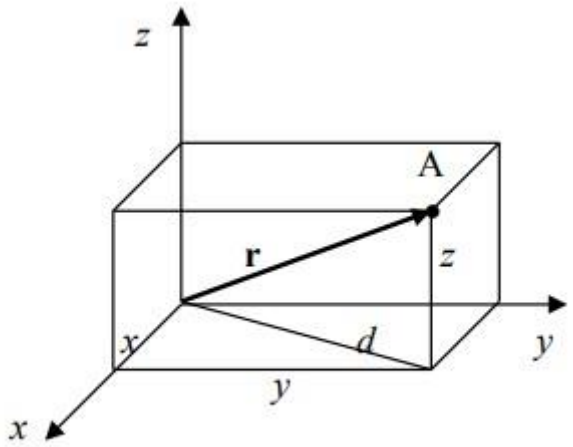
Секое движење и мирување на телата се релативни, што значи дека избраното тело се движи во однос на едно тело но во однос на друго мирува. Не постои апсолутно движење или апсолутно мирување на дадено тело, што значи дека не постои тело што се движи во однос на сите референтни тела и тело што би мирувало во однос на сите други. Еден таков пример е*: Дадено дрво мирува во однос на Земјата, но истовремено се движи во однос на Сонцето.*

Според дефиницијата за движење, може да се заклучи дека одредувањето на положбата на едно тело во однос на друго референтно тело е од витален

карактер. Така положбата на избраната материјална точка (избраното тело) во однос на референтното се одредува со помош на:

- Векторот на положба на телото (материјалната точка) $\vec{r}$
- Декартова правоаголна координата на материјалната точка -x, y, z.

На слика 5 се прикажани двата начина на одредување на положбата на телото (А) во однос на избран референтен систем.



Слика 5 - $\vec{r}$ векторот и декартова правоаголна координата

И во двата (за да може да се одреди положбата на материјалната точка случаи) потребни се, да се познати, по три податоци. Во случај положбата да треба да се одреди со помош на векторот за положба $\vec{r}$, потребно е: должината, правецот и насоката да се познати на тој вектор.

Во случај да се одредува положбата со помош на координата, потребно е да се знае должината на сите страни. Секогаш 3 податоци се потребни бидејќи нашиот простор е тродимензионален.

Движењата на телата можат да се поделат на две групи:

- **Транслаторно движење или транслација**
- **Ротациско движење или ротација**

Кинематика на транслација

Тело се движи транслаторно доколку секоја права линија, што е врзана за тоа тело остане паралелна на својата положба при движењето на телото.

Тело се движи ротациско доколку сите точки се движат по кружница и ако центарот на кружницата е во таа оска.

Основни величини на транслацијата се:

- Брзина $\vec{v}\ (m/s)$
- Забрзување $\vec{a}\ (m/s^2)$
- Време t (s)
- Изминат пат S (m)
- Поместување $\Delta\vec{r}\ (m)$
- Траекторија (m)

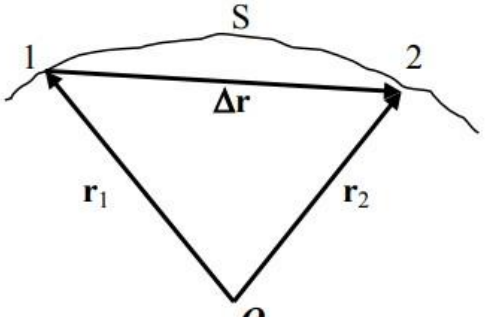
Основни комбинации на транслаторното движење што воедно се и основни карактеристики на ова движење можат да се прикажат математички како:

- Рамномерно праволиниско движење: $v = const, a = 0$
- Нерамномерно праволиниско движење: $v \neq const, a = a_t$
- Рамномерно криволиниско движење: $v \neq const,\ a = a_n$
- Нерамномерно криволиниско движење: $v \neq const, a = a_t + a_n$

Траекторијата на движење е замислената линија по која телото ќе се движи. Изминат пат е делот од траекторијата што телото го поминал при движењето. Патот/траекторијата може да биде **праволиниска и криволиниска**. Патот е скаларна величина. Криволиниски пат не може да биде вектор, но исто така патот не може да биде вектор ни кога е праволиниски, бидејќи нема насока. На слика 6 подолу е прикажан дел од еден криволиниски пат, ограничен помеѓу точките 1 и 2. Положбите на точките 1 и 2 се одредени врз основа на референтната точка О.

Векторот на поместување преставува векторска разлика на векторите за положба на почетната и крајната точка.

$$\Delta\vec{r} = \overrightarrow{r_2} - \overrightarrow{r_1} \qquad \textbf{(1)}$$



Слика 6 – Дел од криволиниски пат, ограничен помеѓу точките 1 и 2, во однос на референтната точка О

Поместувањето е вектор кој ги спојува почетната и крајната точка.

Времето во кинематиката се појавува во два облика и тоа:

- Временски координати t_1, t_2, t_3, ...
- Временски интервали Δt

Временските координати означуваат почеток, крај или некој момент на некое случување низ стандардното мерење на времето.

Временските интервали го означуваат времетраењето на дадената случка.

Пример: Автобус од Штип тргнал во 6 h, а во Скопје пристигнал во 8 h. Овде 6 и 8 преставуваат временски координати, односно $t_1 = 6, t_2 = 8$. Временскиот интервал на овој пример е 2 h, односно $\Delta t = t_2 - t_1$.

Брзината преставува вектор и е дефинирана со: бројна вредност, правец и насока. Бројната вредност на брзината одредува колку пат изминало телото во единица време.

Пример: Ако автомобил се движи со 50 km/h, тоа значи дека автомобилот ќе помине 50 km за време од 1h.

Правецот и насоката на векторот на брзината се одредува во зависност од обликот на движењето.

Слика7 – Пример за правец и насока на векторот за брзина во зависност од обликот на движење

Ако движењето е праволиниско, тогаш правецот на векторот на движење на телото е одреден по патот, додека насоката на векторот е одредена со насоката на движење на телото.

Ако движењето е криволиниско, тогаш правецот на векторот на брзината на телото во дадена точка, е одреден со тангентата на криволинискиот пат во

таа точка. Насоката на движење во таа точка е иста како и насоката на движење на самото тело.

Спрема брзината сите транслаторни движења можат да се поделат во четири типа:

- Рамномерни – брзината е константна, $\mathrm{v} = const$
- Нерамномерни - брзината не е константна, $\mathrm{v} \neq const$
- Праволиниски – кога правецот на векторот на брзината е константен
- Криволиниски – кога правецот на векторот на брзината не е константен

Се разликуваат два типа на брзина: средна брзина и моментална брзина.

Средна брзина кај транслаторното движење се добива кога целокупниот изминат пат ќе се подели со времето потребно да се помине тој пат.

$$\boldsymbol{v_{sr} = \frac{\Delta s}{\Delta t}} \qquad \textbf{(2)}$$

За да се дефинира моментална брзина, мора да се објасни што е момент. Ако земеме која било вредност на времето (час, минута, секунда...), колку и таа да е мала, секогаш постои вредност што е помала од таа. Затоа во физиката за момент се подразбира бесконечно мал интервал на времето, што се означува со: $\Delta t \to 0$. Тогаш:

Моментална брзина е еднаква на средната брзина, но измерена во бесконечно многу мал интервал.

$$v_m = v_{sr}\,, \qquad (\Delta t \to 0) \qquad (3)$$

Забрзувањето, исто како и брзината е вектор. Затоа и тоа е дефинирано со должина, насока и правец. Должината на забрзувањето преставува промената на брзината на движење на телото во единица време (временски интервал).

$$a = \frac{\Delta v}{\Delta t} \qquad (4)$$

Пример: Нека дадено тело се движи по прав пат, но неговата брзина да биде променлива. Да се претпостави дека брзината во точка 1 е помала отколку

брзината во точка 2, што значи дека телото забрзува. На слика 8 е прикажан пример за забрзување. Δv – кај забрзувањето преставува разлика од брзината во крајната точка и брзината од првата точка. $\Delta v = v_2 - v_1$. Временскиот интервал се одредува на идентичен начин, односно: $\Delta t = t_2 - t_1$.

Слика8 – Пример за забрзување

$$a = \frac{\Delta v}{\Delta t} = \frac{v_2 - v_1}{t_2 - t_1} = \frac{11 - 3}{9 - 5} = \frac{8m/s}{4s} = 2\ m/s^2$$

Од резултатот погоре ($2\ \mathrm{m/s^2}$) се гледа дека брзината на движење на телото се зголемува секоја секунда за 2 m/*s.* Па така во шестата секунда брзината на движење е 5 m/s, во седмата 7 m/s и итн.

Ако движењето на телото се намалува, тогаш забрзувањето има негативен предзнак, односно негативна вредност на забрзување, која укажува за колку телото ја намалува својата брзина при секоја секунда.

Забрзувањето постои за да може да се одреди промената на брзината на движење на дадено тело. Забрзување може да се очекува кај сите нерамномерни и криволиниски движења, додека кај праволиниските рамномерни движења не постои забрзување. Комплексноста кај забрзувањето е тоа што преставува вектор и ја менува својата должина кај нерамномерните движења и ја менува својата насока кај криволиниските движења. Бидејќи овие две промени се од различен тип/основа тие не можат да бидат пресметани со едно забрзување. Затоа во кинематиката постојат два типа на забрзување:

- Тангенцијално забрзување a_t
- Нормално забрзување a_n

Тангенцијалното забрзување ја одредува промената на вредноста на брзината и се јавува кај сите нерамномерни движења. Односно, одредува колку од вкупното забрзување делува на насоката на движење.

Нормалното забрзување ја одредува промената на правецот на брзината и се јавува кај сите криволиниски движења.

Рамномерно движење	$v = const$	$a_t = 0$, не постои
Нерамномерно движење	$v \neq const$	$a_t \neq 0$, постои
Праволиниско движење	Правец$v = const$	$a_n = 0$, не постои
Криволиниско движење	Правец$v \neq const$	$a_n \neq 0$, постои

Вкупното забрзување на дадено тело преставува векторски збир на тангенцијалното и нормалното забрзување.

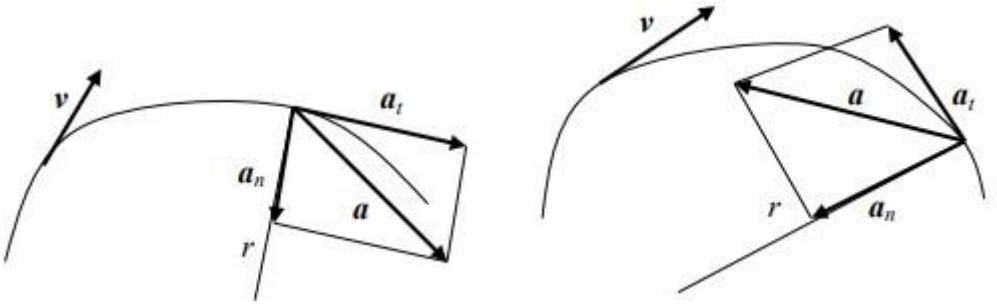
$$\vec{a} = \vec{a}_t + \vec{a}_n \qquad (5)$$

- Кај рамномерно праволиниско движење не постои ниту едно од двете забрзувања: $a = 0$
- Кај рамномерно криволиниско движење, телото има нормално забрзување, па вкупното забрзување ќе биде: $a = a_n$
- Кај нерамномерното праволиниско движење, телото има само тангенцијално забрзување и крајното забрзување ќе биде: $a = a_t$
- Кај нерамномерното криволиниско движење телото ги има и двете забрзувања и нормалното и тангенцијалното, па крајното забрзување ќе биде: $a = a_t + a_n$

Ќе се разгледаат два примера:

Во првиот пример движењето е забрзано криволиниско, додека во вториот пример движењето е успорено криволиниско. Од слика 9 може да се заклучи правецот и насоката на тангенцијалното и нормалното забрзување, како и зошто самите називи се такви.

Векторот на тангенцијалното забрзување секогаш се влече по тангента на криволиниското движење на телото (ако поместувањето на телото е праволиниско тогаш a_t се влече по таа траекторија). Насоката на a_t зависи дали движењето е забрзано или успорено. Ако движењето на телото е забрзано тогаш a_t се црта во насоката на движење на телото, но ако движењето е успорено тогаш a_t се црта со спротивна насока на движењето на телото.



Слика 9 – Векторот на тангенцијалното и нормалното забрзување, кај забрзување и успорување; лево забрзување, десно успорување

Векторот на нормалното забрзување секогаш се повлекува по должината на радиусот на искривување и секогаш е насочен кон центарот на кривата. Името нормално забрзување потекнува по правиот агол што векторот го прави со векторот a_t. Прикажано, исто така, на слика 9. Во физиката нормалното забрзување има повеќе имиња: центрипетално – насочено кон центарот, централно – исто како центрипеталното, радијално – по должината на радиусот на кривината.

Постојат средно и моментално забрзување.

Средно забрзување се пресметува кога вкупната промена на брзината ќе се подели со времето потребно да се направи таа промена.

$$a_{sr} = \frac{\Delta v}{\Delta t} \tag{6}$$

Како и кај моменталната брзина и овде се воведува поимот бесконечно краток временски интервал, кој одговара на поимот момент. **Моментално забрзување е еднакво на средното забрзување измерено во бесконечно мал временски интервал.**

$$a_{tr} = a_{sr},\ (\Delta t \to 0) \tag{7}$$

$$a_{tr} = \frac{\Delta v}{\Delta t},\ (\Delta t \to 0) \tag{8}$$

Рамномерно праволиниско движење

Кај ова движење вредноста на брзината е непроменлива, затоа што движењето е рамномерно, а правецот на брзината е непроменлив, затоа што движењето е

праволиниско. Значи телото кое се движи рамномерно праволиниски нема ни тангенцијално, ни нормално забрзување. Ова е единствено движење без забрзување.

Големини кои се застапени во ова движење се:

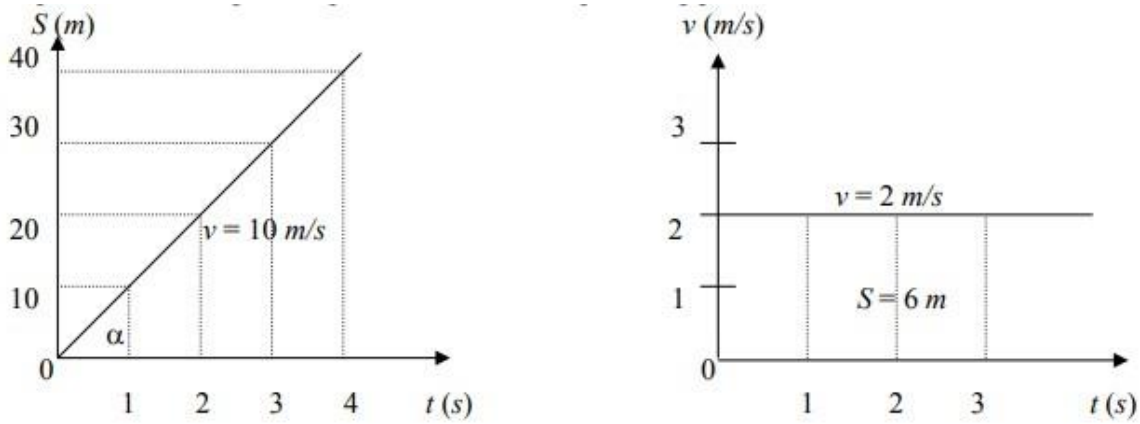
- Изминат пат *S (m)*
- Брзина *v(m/s)*
- Време *t(s)*

Формулата која ги поврзува овие големини гласи:

$$v = \frac{S}{t}, \quad S = v \cdot t, \quad t = \frac{S}{v} \tag{9}$$

Формулата може да се користи само кај рамномерни движења, т.е. движења кај кои вредноста на брзината е константна.

Подолу на графиконите се прикажани зависностите: изминатион пат во единица време и брзината во единица време.



Слика10 – Пример за рамномерно праволиниско движење на тело

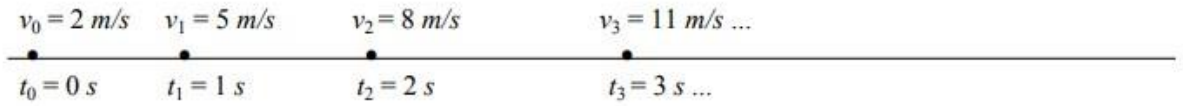
На првиот графикон прикажана е зависноста на изминатион пат од времето, со вредност на брзината од $v = 10\,m/s$. Аголот α помеѓу линијата и t оската зависи од брзината на телото. Колку е брзината на телото поголема, толку аголот α е поголем.

На вториот графикон е прикажана зависноста на брзината од времето, кога вредноста на брзината е $v = 2\,m/s$. Графикот е паралелен со оската t, затоа што брзината е непроменета низ целиот временски интервал. Од

графиконот се гледа дека површината под линијата, всушност преставува вредноста на изминатион пат S. Површината во овој случај е секогаш во вид на правоаголник каде што едната страна е времето t и е променлива, а другата страна е секогаш константа, брзината v.

Праволиниско движење со постојано забрзување

Во овој случај телото се движи праволиниски, а вредноста на неговата брзина постојано расте или опаѓа за иста вредност низ временскиот интервал. Пример



Слика 11 – Пример за праволиниско движење со постојано забрзување

Од слика 11 се гледа дека брзината се зголемува секоја секунда за $v = 3m/s$, па може да се заклучи дека тангенцијалното забрзување е $a_t = 3m/s2$ што воедно преставува и целокупното забрзување. Математички може да се престави:

$$a = a_t = \frac{\Delta v}{\Delta t} = \frac{v_3 - v_1}{t_3 - t_1} = \frac{11 - 5}{3 - 1} = \frac{6m/\ s}{2s} = 3\ m/s^2$$

Основни величини што се среќаваат кај ова движење се:

- Почетна брзина v_0
- Крајна брзина v
- Средна брзина v_{sr}
- Изминат пат S
- Забрзување a
- Време t

Формули што ги поврзуваат овие величини се:

$$a = \frac{\Delta v}{\Delta t} \qquad (10)$$

$$v = v_0 \pm a \cdot t \qquad (11)$$

$$v_{sr} = v_0 \pm \frac{a \cdot t}{2}, \quad v_{sr} = \frac{v_0 + v}{2}, \quad v_{sr} = \frac{S}{t} \tag{12}$$

$$S = v_0 \cdot t \pm \frac{a \cdot t^2}{2} \tag{13}$$

$$v^2 = v_0^2 \pm 2 \cdot a \cdot S \tag{14}$$

Изведување на формулата:$v = v_0 \pm a \cdot t$

Слика 12 – Пример за праволиниско забрзување

На примерот прикажан на слика 12 погоре, може да се види дека само вредноста за t_0има постојана вредност, односно 0. Останатите 3 величини може да имаат различни вредности. Па така:

$$a = \frac{\Delta v}{\Delta t} = \frac{v - v_0}{t - t_0} \tag{15}$$

бидејќи $t_0 = 0$:

$$a = \frac{v - v_0}{t} \tag{16}$$

$$a \cdot t = v - v_0 \tag{17}$$

$$\boldsymbol{v = v_0 + a \cdot t} \tag{18}$$

Изведување на формулата: $v_{sr} = v_0 + \frac{(a \cdot t)}{2}$

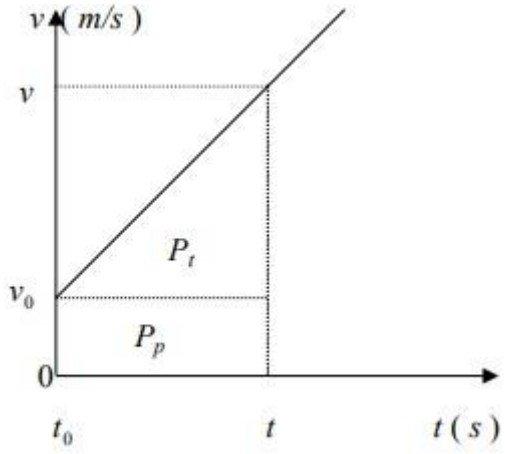
Од формулите: $v_{sr} = \frac{v_0 + v}{2}$и $v = v_0 + a \cdot t,$ се добива:

$$v_{sr} = \frac{(v_0 + v_0 + a \cdot t)}{2} = \frac{2\,v_0 + a \cdot t}{2} = \frac{2v_0}{2} + \frac{a \cdot t}{2} \Rightarrow$$

$$\boldsymbol{v_{sr} = v_0 + \frac{a \cdot t}{2}} \tag{19}$$

Изведување на формулата: $S = v_0 \cdot t + \frac{a \cdot t^2}{2}$

Оваа формула може да се изведе (покрај математички) и со помош на графиконот на кој се прикажува зависноста на брзината на телата од поминатото време.



Слика13 – Пример за изведување на формулата$S = v_0 \cdot t + \frac{a \cdot t^2}{2}$

Како и кај рамномерно праволиниско движење, изминатиот пат е еднаков на површината што кривата/линијата ја прави со временската оска t. Оваа површина може да се подели на:

- Површина на правоаголникот од почетната брзина
- Површина на правоаголниот триаголник

На тој начин изминатион пат се добива како збир од двете површини:$S = P_p + P_t$

$$S = v_0 \cdot t + \frac{(v - v_0) \cdot t}{2} \tag{20}$$

со замена на: $v = v_0 + a \cdot t$се добива:

$$S = v_0 \cdot t + \frac{(v_0 - v_0 + a \cdot t) \cdot t}{2} \Rightarrow$$

$$\boldsymbol{S = v_0 \cdot t + \frac{a \cdot t^2}{2}} \tag{21}$$

Изведување на формулата: $v^2 = v_0^2 + 2 \cdot a \cdot S$

Од формулата: $v = v_0 + a \cdot t$се изразува времето t =>

$$t = \frac{v - v_0}{a} \tag{22}$$

$$S = v_0 \cdot t + \frac{a \cdot t^2}{2} \Rightarrow$$

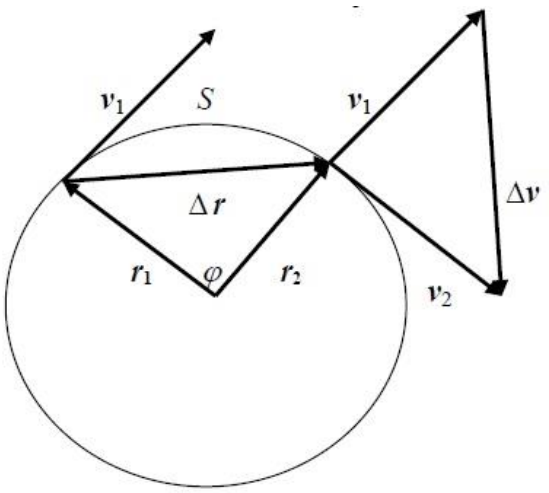
$$S = v_0 \cdot \frac{v - v_0}{a} + \frac{a}{2} \cdot \frac{(v - v_0)^2}{a^2} \Rightarrow$$

$$S = \frac{v^2 - v_0^2}{2a} \tag{23}$$

$$\boldsymbol{v^2 = v_0^2 + 2 \cdot a \cdot S} \tag{24}$$

Рамномерно кружно движење

Кога телото се движи по рамномерно кружна патека, тогаш може да се искористи истата формула за брзина која што ја имаме и кај движењето по рамномерно праволиниска патека $v = \frac{S}{t}$. Доколку за изминатиот пат се земе периметарот на кругот $S = O = 2\pi r$, тогаш времето кое што е потребно за негово поминување се нарекува период и се означува со $T(s)$.



Слика14 – Рамномерно кружно движење

$$v = \frac{2\pi r}{T} \tag{25}$$

Кај кружното движење се дефинира и величината наречена фреквенција, а таа ги означува бројот на вртежи кои што телото ги прави во единица време, односно во текот на една секунда. Ознаката за фреквенција е v, а мерната

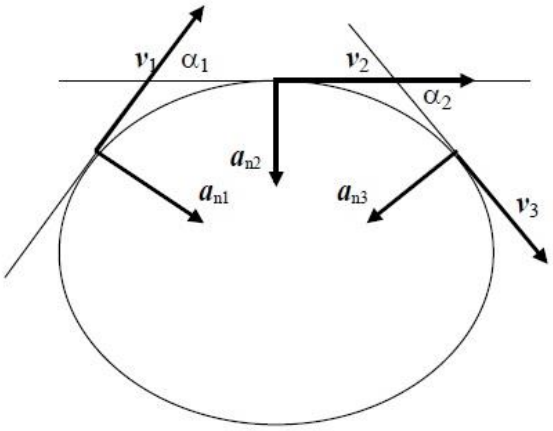
единица е Hz (херц). Периодот T и фреквенцијата v се реципроцитетни величини, т.е.

$$v = \frac{1}{T} \text{ или } T = \frac{1}{v}. \quad (26)$$

Кај рамномерното кружно движење бројната вредност на брзината е постојана, па затоа нема тангенцијално забрзување, но се менува правецот на брзината и затоа телото има нормално забрзување. Бројната вредност на нормалното забрзување е постојана (константа), затоа што кружната патека претставува правилна кривина, што значи дека на исти делови од патеката, правецот од векторот на брзината секогаш се менува за ист агол. Доколку на слика 14 се земе два еднакви дела од кружната патека, тогаш ќе се забележи дека промената на правецот на брзина на двата дела е еднаква. Промената на правецот на брзина може да се одреди со помош на аглите: $\alpha_1 и \alpha_2$. Тоа значи дека овие два агла се еднакви, односно $\alpha_1 = \alpha_2$. Тоа понатаму значи дека нормалните забрзувања: α_{n1} иα_{n2} исто така треба да имаат иста бројна вредност, затоа што одредуваат две еднакви промени по правецот на брзината. Па затоа од овде математички претставено, рамномерното кружно движење може да ги опише овие изрази:

$$v \neq const. a = a_n;\ a_n = const.$$

Притоа, треба да се има на ум, дека постојана (константна) е само бројната вредност на нормалното забрзување, додека неговиот правец е променлив,



Слика 15 – Рамномерно кружно движење
со два дела

затоа што од секоја точка од кружната патека е насочена кон центарот на кругот.

Сега ќе ја изведеме формулата за нормално забрзување користејќи ја слика15. Секое, па и нормалното забрзување се дефинира како промена на брзината во единица време, па така и ќе го запишеме, но во векторски облик:

$$\overrightarrow{a_n} = \frac{\Delta\vec{v}}{\Delta t}, \ a_n = \frac{\Delta v}{\Delta t}. \qquad (27)$$

Триаголникот што го сочинуваат двата позициони вектори: r_1 и r_2 со векторот на движење Δr, и триаголникот што го сочинуваат векторите на брзина: v_1 и v_2 со векторот Δv, се слични затоа што се рамностранi, а аголот φ помеѓу нивните краци е ист во двата триаголника (како агли со меѓусебно нормални краци, затоа што помеѓу радиусот и тангентата се наоѓа прав агол). Од овие сличности доаѓаме до следната пропорција:

$$\frac{\Delta v}{v} = \frac{\Delta r}{r}, \text{ (затоа што: } v_1 = v_2 = v \text{ и} r_1 = r_2 = r), \ \ \Rightarrow \Delta v = \frac{v}{r}\Delta r.$$

$$a_n = \frac{v}{r} \cdot \frac{\Delta r}{\Delta t} \qquad (28)$$

Ако се земе мал агол φ, тогаш не постои значајна разлика помеѓу: должината на поминатитот пат S и должината на векторот на движење Δr, т.е. $\Delta r \approx S$.

Поради тоа : $a_n = \frac{v}{r} \cdot \frac{\Delta r}{\Delta t}$, а како рамномерно движење : $\frac{S}{\Delta t} = v$, па оттука: $a_n = \frac{v}{r} \cdot v$ или конечно добиваме:

$$a_n = \frac{v^2}{r} \qquad (29)$$

Од оваа формула може да заклучиме дека бројната вредност на нормалното забрзување е константа, затоа што се добива како количник од вредностите кои што и самите се константи. Оваа формула е применлива и на нерамномерни кружни движења, но тогаш a_n претставува моментално забрзување додека v е моментална брзина.

Кинематика на ротацијата

Ротација. Аголна брзина и аголно забрзување

Транслација	Ротација
Изминат пат *S (m)*	Изминат агол $\varphi(rad)$
Брзина *v (m/s)*	Аголна брзина $\omega(rad/s)$
Забрзување *a(m/s²)*	Аголно забрзување $\alpha\ (rad/s^2)$
Време *t(s)*	Време$t\ (s)$

Од табеларниот приказ на двете движења на телата (транслаторно и ротациско), може да се заклучи дека трите транслаторни величини не можат да се користат при ротација на телата, додека единицата време е константна величина, која се користи кај двете движења.

1. *<u>Изминат пат и изминат агол</u>*

Кај транслаторното движење сите точки на телото изминуваат еднакви должини и затоа тука лесно се утврдува колкав пат изминало целото тело. Должината на изминатиот пат на сите точки од телото е иста со должината на изминатиот пат на телото.

При ротацијата на телото сите точки од телото не изминуваат иста должина, поточно точките кои се поблиску до оската поминуваат помал пат а точките кои се пооддалечени од оската на ротацијата поминуваат подолги должини на патот. Поради ова не е можно да се одреди колкав пат има поминато целото тело. Сепак има величина која е еднаква за сите точки од телото при ротацијата на истото а тоа е изминатиот алог. Затоа при ротацијата на телата наместо изминатиот пат се користи изминатиот агол како величина. Во физиката единица за мерење на изминат агол е 1 радијан.

$$\mathbf{1}\ \boldsymbol{rad} = \frac{\mathbf{360^\circ}}{\boldsymbol{2\pi}} = \mathbf{57,324^\circ} \qquad \textbf{(30)}$$

2. *<u>Брзина и аголна брзина</u>*

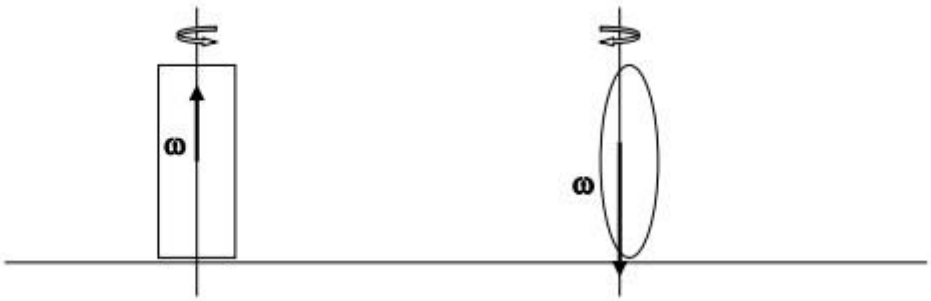
Кај транслацијата сите точки на телото поминуваат еднаква должина за исто време односно имаат иста брзина. Затоа е можно да утврдиме со колкава

брзина се движи целото тело бидејќи сите точки од телото се движат со иста брзина.

Кај ротацијата точките од телото не се движат со иста брзина туку точките што се поблиску до оската се движат побавно, со помала брзина од точките кои се пооддалечени од оската на ротацијата и кои при ротирање се движат со поголема брзина. Поради ова не е возможно да се утврди брзината на телото бидејќи таа брзина не се однесува на сите точки од телото.

Меѓутоа, сите точки од телото при ротацијата за исто време поминуваат исти агол, со тоа сите точки имаат иста аголна брзина. Затоа во ротацијата, наместо брзина се користи аголна брзина.

Правецот на векторот од аголната брзина е на оската на ротацијата на телото , а неговата насока е одредена со правилото на десната рака: свитканите прсти од десната рака ја покажуваат насоката на вртењето на телото а испруженият палец го одредува насоката на векторот на аголната брзина.



Слика16–Правец и насока на ротација

3. Забрзување и аголно забрзување

Во транслацијата забрзувањето служи да ја одреди промената на брзината.

Во ротацијата аголното забрзување се користи да се одреди промената на аголната брзина.

4. Време

Времето како величина е константа, независна од видот на движењата на телата , па затоа се користи и во транслацијата и во ротацијата.

Постојат четири основни типа на движење и тоа :

Транслација	**Ротација**
Рамномерно – $v = const$	Рамномерно – $\omega = const$
Нерамномерно – $v \neq const$	Нерамномерно - $\omega \neq const$
Праволиниско – правец $\vec{v} = const$	Без процесија - $\vec{\omega} = const$
Криволиниско – правец $\vec{v} \neq const$	Со процесија - $\vec{\omega} \neq const$

Во кинематиката постои потполна аналогија помеѓу Законот за транслаторно и законот за ротациско движење. Поточно, четири можни комбинации на движења во транслацијата имаат свои четири аналогни комбинации на движење во ротацијата.

- Рамномерно праволиниско движење во транслацијата е аналогно на рамномерна ротација без процесија.

- Нерамномерно праволиниско движење во транслацијата е аналогно на нерамномерна ротација без процесија.

- рамномерно криволиниско движење во транслацијата е аналогно на рамномерна ротација со процесија.

- Нерамномерно криволиниско движење во транслацијата е аналогно на нерамномерна ротација со процесија.

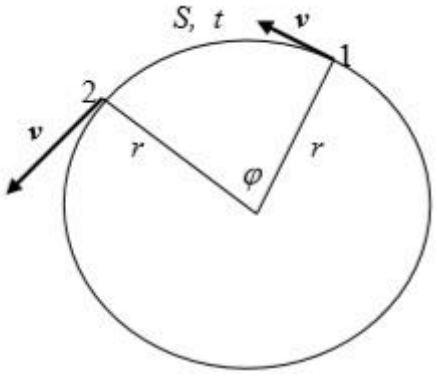
Меѓутоа аналогијата на овие движења се проширува и на други дефиниции што значи дека за одредени транслаторни величини можат да се користат аналогни ротациони величини прикажани во табелата со што настануваат одредени аналогни ротациони закони.

Транслација	**Ротација**
Рамномерно праволиниско движење	Рамномерна ротација без процесија
$v = \frac{S}{t}$	$\omega = \frac{\varphi}{t}$

Праволиниско движење со постојано забрзување	Ротација без процесија со постојано аголно забрзување
$v = v_0 \pm a \cdot t$	$\omega = \omega_0 \pm \alpha \cdot t$
$v_{sr} = v_0 \pm \frac{a \cdot t}{2}$	$\omega_{sr} = \omega_0 \pm \frac{\alpha \cdot t}{2}$
$S = v_0 \cdot t \pm \frac{a \cdot t^2}{2}$	$\varphi = \omega_0 \cdot t \pm \frac{\alpha \cdot t^2}{2}$
$v^2 = v_0^2 \pm 2 \cdot a \cdot S$	$\omega^2 = \omega_0^2 \pm 2 \cdot \alpha \cdot \varphi$

Аналогијата на поимите за транслација и ротација е применлива и во одредени дефиниции. Како пример во транслацијата ја имаме следна дефиниција: „**Бројната вредност на забрзувањето одредува колкава е промената на брзината во единица време**“, а во ротацијата имаме аналогна дефиниција која се добива со проста замена на транслаторни големини со ротациони : „**Бројната вредност на аналогното забрзување одредува колкава е промената на аголната брзина во единица време**“. Единствено аналогијата не е применлива при цртање на векторски транслаторни и ротациони големини. Од ова произлегува дека може да се направи поврзување на транслаторни со соодветни ротациони големини единствено ако постои движење кое истовремено може да биде и транслаторно и ротациско. Такво движење е кружното движење.

Пример за кружно движење на точка е прикажана на слика 17.



Слика17 – Пример за кружно движење на точка

Движењето на една точка на кружницата се врши во временски период (t). Точката во тој период изминува пат (s). Патот кој што го

изминува точката претставува кружен лак односно дел од кружницата. Точката се движи со рамномерна брзина *(v)*. Од ова се гледа дека по опишувањето на движењето на овој начин се работи за транслаторно движење бидејќи во изразувањето се користат транслаторни величини, големини. Ова движење може да се опише и со користење на следните големини: за време *(t)* точката го има изминато аголот (φ) движејќи се со рамномерна аголна брзина (ω) При ова објаснување се користени ротационите големини со што и самото движење е ротација.

Бидејќи кружниот лак се добива со множење на радиусот од кружницата со соодветниот централен агол (искажан во радијани) се добива равенство помеѓу изминатиот пат и изминатиот агол.

$$S = r \cdot \varphi \tag{31}$$

Врската помеѓу брзината и аголната брзина се добива ако веќе формулираната равенка се подели со времето за која точката го поминала патот, односно аголот.

$$\frac{S}{t} = r \cdot \frac{\varphi}{t} \tag{32}$$

Бидејќи движењето е рамномерно следува:

$$v = \frac{S}{t}; \qquad \omega = \frac{\varphi}{t} \;\Rightarrow$$
$$\boldsymbol{v = r \cdot \omega} \tag{33}$$

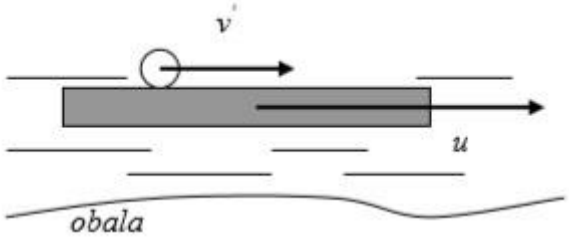
Ако движењето по кружницата е забрзано и телото во точката **1** се наоѓа во времето t_1 со брзина v_1, а подоцна во точката **2** се наоѓа во времето t_2со зголемена брзина v_2 тогаш

$$a = \frac{\Delta v}{\Delta t} = \frac{v_2 - v_1}{t_2 - t_1} = \frac{r \cdot \omega_2 - r \cdot \omega_1}{t_2 - t_1} = r \cdot \frac{\omega_2 - \omega_1}{t_2 - t_1} = r \cdot \frac{\Delta \omega}{\Delta t} = r \cdot \alpha \quad \Rightarrow$$

$$\boldsymbol{a = r \cdot \alpha} \tag{34}$$

Класичен закон за собирање на брзина

Брзината на телото кое е во корелација со друго тело кое исто така се движи се добива со собирање на брзините на телата доколку истите се движат во ист правец и иста насока. Како пример се наведува движењето на едно топче кое се движи по дрвена површина (сплав) која плива во некоја река.



Слика18 – Пример сплав на река со топче на сплавот

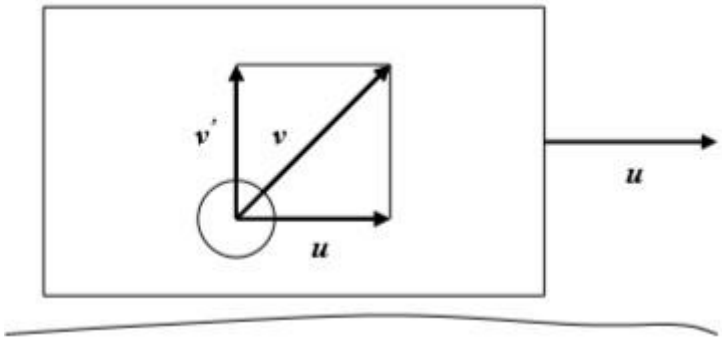
Топчето се движи со брзина *v'* во правец и насока на течењето на реката со брзина *u*. Брзината на топчето е *v= v' + u*

Во случај топчето и дрвената површина да се движат во ист правец а спротивни насоки , брзината на топчето *е v= v'- u*

Ако векторите на брзината $\vec{v}$и $\vec{u}$ имаат различни правци се применува класичниот закон за собирање на брзина во векторска форма $\vec{v} = \vec{v} + u$

Брзината на топчето кое се движи во правец кој е нормален (агол од 90°) на правецот по кој се движи дрвената површина во реката може да се пресмета и со користење на Питагоровата теорема.

$$v^2 = (v)^2 + u^2$$



Слика19 – Пример сплав на река

Принцип на релативност

Според Галилеј, принцип на релативност е „**избрана механичка појава ќе се случува на потполно ист начин во различни инерцијални системи**“.

Ајнштајн, подоцна, го проширил овој закон на сите физички појави. Јасно е дека Галилеј не бил во можност да ги разгледува сите физички

појави затоа што во негово време физиката не постоела како наука, туку постоела само механиката која подоцна, односно, денес знаеме дека е дел од физиката.

Инерцијален систем, според дефиниција, е секое тело кое нема забрзување, односно мирува или се движи со рамномерно праволиниско движење. Најдобар пример за инерцијален систем е некое превозно средство, пример: авион, камион, подморница, автобус..., а за механичка, односно физичка појава, пример е осцилаторното движење на клатното.

За пример земаме два авиона од кои едниот мирува, а другиот се движи рамномерно - праволиниски. Во двата авиона се стават две еднакви клатна кои се занишуваат со ист почетен отклон. Според Галилеј и според Ајнштајн клатната ќе осцилираат на ист начин. Различно осцилирање на клатната ќе има доколку еден од авионите се движи забрзано , успорено или криволиниски. Од овој експеримент може да се заклучи дека инерцијалните системи им овозможуваат рамноправни услови за одвивање на сите физички појави , а тоа е всушност причината поради која сите појави во тие системи се одвиваат на ист начин.

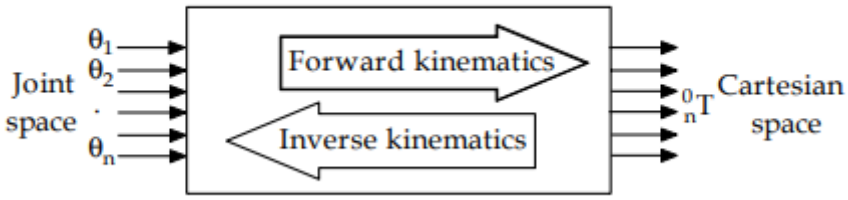
Роботска кинематика

Роботска кинематика преставува аналитичко проучување на движењето на роботот. Дизајнирањето на правилно и посакувано движење на роботскиот манипулатор е важен чекор при работата на даден роботски манипулатор. Постојат две главни полиња при изучување на кинематиката и тоа: **Декартов (Cartesian space) системисистем на Кватерниони (Quaternion space)**. Трансформацијата помеѓу два Декартови координатни система се прави со ротации и транслации. Постојат многу начини за преставување на ротации, како на пример: Ојлеровите агли (Euler angels), векторите на Гибс (Gibbs vector), Кеијли-Клаин параметри (Cayley-Klein parameters), ортогонални матрици (orthonormal matrices), кватарнионите на Хамилтон (Hamilton's quaternions). Од сите овие репрезентации, во роботиката, најкористени се хомогените трансформации со ортогоналните матрици 4x4. Денавит и Хартенберг покажаа дека трансформацијата меѓу два зглоба на даден робот е искажана преку четири параметри. Овие параметри се познати како **DH**

параметри и веќе се како стандардни параметри при опис и дизајн на кинематиката за даден робот.

Иако кватарнионите преставуваат подобра репрезентација на ротација, тие не се толку користени како хомогените трансформации во роботиката. Двоен кватарнион може да прикаже и ротација и транслација во форма на трансформациски вектор. Додека ориентацијата на дадено тело со помош на хомогени трансформации е одредена со девет елементи, двојниот кватарнион, овој број, го намалува на четири. Со ова се обезбедува поголема искористеност на меморијата и побрза пресметка.

Во денешно време роботската кинематика се дели на директна и инверзна кинематика. Директната кинематика е едноставна и нема некоја поголема комплексност при изучување, дизајнирање и пресметување на равенките. За даден робот секогаш може да се најде или напише решение за директна кинематика. Од друга страна инверзната кинематика преставува многу покомплексен проблем отколку директната кинематика. Решенијата за инверзна кинематика во реалниот свет се скапи и одземат многу долго време. Врската меѓу директната и инверзната врска е прикажана подолу на слика 20.



Слика 20 – Насока на податоците при решавање на директна и иверзна кинематика

Две најпопуларни техники за решавање на проблем врзан со инверзната кинематика се аналитички метод и техника на нумерички методи. Во првата техника, аглите на зглобовите се наоѓаат аналитички преку дадена конфигурација. Додека со втората техника се користи некој нумерички метод како Јакобиев, Гаус, Гаус-Њутон и итн.

Како трета техника за решавање на проблем врзан со инверзна кинематика е комбинацијата на Fuzyy System и невронски мрежи. Таа комбинација се

нарекува ANFIS (Adaptive Neuro Fuzyy Interface System). Во наредните поглавја ќе се разгледа на кој начин функционира оваа техника и нејзина имплементација.

Директна кинематика

Роботски манипулатор е составен од серија на роботски делови т.н. „врски“(links) кои меѓусебно се врзани. Пресметката на позицијата и ориентацијата на роботот врз претходна варијација на зглобовите се нарекува директна кинематика. Кај роботската рака, директна кинематика се добива со ротации и транслации на зглобовите. Со помош на матрици ова аналитички би изгледало вака:

$$
\begin{aligned}
{}^{i-1}_{i}T &= R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)Q_i(d_i) \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{35}
$$

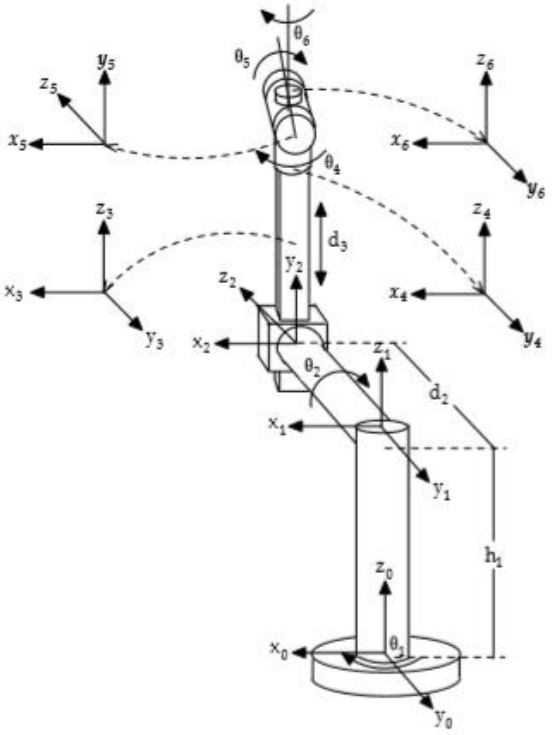
каде што R_xи R_zпреставуваат ротации, D_xи Q_i се транслации и $c\theta_i$и $s\theta_i$преставуваат скратеници од $cos\theta_i$и $sin\theta_i$ сооедветно.

$$
{}^{base}_{end-effector}T = {}^{0}_{1}T\,{}^{1}_{2}T \dots {}^{n-1}_{n}T \tag{36}
$$

Како алтернатива може да се престави со помош на матрица

$$
{}^{base}_{end-effector}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}
$$

Пример како изгледаат трансформациите и транслациите аналитички, ќе се земе предвид роботска рака со 6 зглоба. Роботската рака е преставена на слика 21.



Слика 21 – Роботска рака со 6 составни зглобови

DH параметрите се прикажани во табелата:

i	θ_i	α_{i-1}	a_{i-1}	d_i
1	θ_1	0	0	h_1
2	θ_2	90	0	d_2
3	0	-90	0	d_3
4	θ_4	0	0	0
5	θ_5	90	0	0
6	θ_6	-90	0	0

(38)

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & h_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & -d_2 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & 0 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{6}T = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_6 & c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$
\begin{aligned}
r_{11} &= -s\theta_6(c\theta_4 s\theta_1 + c\theta_1 c\theta_2 s\theta_4) - c\theta_6(c\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) + c\theta_1 s\theta_2 s\theta_5) \\
r_{12} &= s\theta_6(c\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) + c\theta_1 s\theta_2 s\theta_5) - c\theta_6(c\theta_4 s\theta_1 + c\theta_1 c\theta_2 s\theta_4) \\
r_{13} &= s\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) - c\theta_1 c\theta_5 s\theta_2 \\
r_{21} &= s\theta_6(c\theta_1 c\theta_4 - c\theta_2 s\theta_1 s\theta_4) + c\theta_6(c\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - s\theta_1 s\theta_2 s\theta_5) \\
r_{22} &= c\theta_6(c\theta_1 c\theta_4 - c\theta_2 s\theta_1 s\theta_4) - s\theta_6(c\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - s\theta_1 s\theta_2 s\theta_5) \\
r_{23} &= -s\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - c\theta_5 s\theta_1 s\theta_2 \\
r_{31} &= c\theta_6(c\theta_2 s\theta_5 + c\theta_4 c\theta_5 s\theta_2) - s\theta_2 s\theta_4 s\theta_6 \\
r_{32} &= -s\theta_6(c\theta_2 s\theta_5 + c\theta_4 c\theta_5 s\theta_2) - c\theta_6 s\theta_2 s\theta_4 \\
r_{33} &= c\theta_2 c\theta_5 - c\theta_4 s\theta_2 s\theta_5 \\
p_x &= d_2 s\theta_1 - d_3 c\theta_1 s\theta_2 \\
p_y &= -d_2 c\theta_1 - d_3 s\theta_1 s\theta_2 \\
p_z &= h_1 + d_3 c\theta_2
\end{aligned}
$$

(39)

Инверзна кинематика

За разлика од директна кинематика, инверзна кинематика е многу покомплексна и сложена за имплементација. Роботскиот манипулатор од колку повеќе делови и зглобови е составен толку повеќе имплементацијата станува покомплексна. Може слободно да се каже дека комплексноста зависи од бројот на зглобови на роботскиот манипулатор. Во наредните поглавја, фокусот ќе биде задржан на имплементација на инверзната кинематика. Овде како за пример ќе се спомне роботскиот манипулатор што беше опфатен во поглавјето за директна кинематика, се со цел да се покаже комплексноста на имплементација на инверзна кинематика врз даден роботски манипулатор. Ќе се користи исто како и за директна кинематика алгебарски метод со помош на матрици. Од равенката:

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{0}_{1}T(q_1){}^{1}_{2}T(q_2){}^{2}_{3}T(q_3){}^{3}_{4}T(q_4){}^{4}_{5}T(q_5){}^{5}_{6}T(q_6) \tag{40} $$

за да се најде решението за инверзна кинематика за првиот зглоб (q_1) како функција од познатите елементи на ${}^{base}_{end-effector}T$, ќе биде помножена со инверзната матрица на $[{}^{0}_{1}T(q_1)]^{-1}$по што се добива:

$$ [{}^{0}_{1}T(q_1)]^{-1}{}^{0}_{6}T = [{}^{0}_{1}T(q_1)]^{-1}{}^{0}_{1}T(q_1){}^{1}_{2}T(q_2){}^{2}_{3}T(q_3){}^{3}_{4}T(q_4){}^{4}_{5}T(q_5){}^{5}_{6}T(q_6) \tag{41} $$

Познато е дека доколку дадена матрица се помножи со својата инверзна матрица се добива единечна матрица, односно $[{}^{0}_{1}T(q_1)]^{-1}{}^{0}_{1}T(q_1) = I$. Откука следува:

$$ [{}^{0}_{1}T(q_1)]^{-1}{}^{0}_{6}T = {}^{1}_{2}T(q_2){}^{2}_{3}T(q_3){}^{3}_{4}T(q_4){}^{4}_{5}T(q_5){}^{5}_{6}T(q_6) \tag{42} $$

За да се најдат и другите променливи на сличен начин се помножуваат сите зглобови, односно:

$$ [{}^{0}_{1}T(q_1){}^{1}_{2}T(q_2)]^{-1}{}^{0}_{6}T = {}^{2}_{3}T(q_3){}^{3}_{4}T(q_4){}^{4}_{5}T(q_5){}^{5}_{6}T(q_6) \tag{43} $$

$$ [{}^{0}_{1}T(q_1){}^{1}_{2}T(q_2){}^{2}_{3}T(q_3)]^{-1}{}^{0}_{6}T = {}^{3}_{4}T(q_4){}^{4}_{5}T(q_5){}^{5}_{6}T(q_6) \tag{44} $$

$$ [{}^{0}_{1}T(q_1){}^{1}_{2}T(q_2){}^{2}_{3}T(q_3){}^{3}_{4}T(q_4)]^{-1}{}^{0}_{6}T = {}^{4}_{5}T(q_5){}^{5}_{6}T(q_6) \tag{45} $$

$$ [{}^{0}_{1}T(q_1){}^{1}_{2}T(q_2){}^{2}_{3}T(q_3){}^{3}_{4}T(q_4){}^{4}_{5}T(q_5)]^{-1}{}^{0}_{6}T = {}^{5}_{6}T(q_6) \tag{46} $$

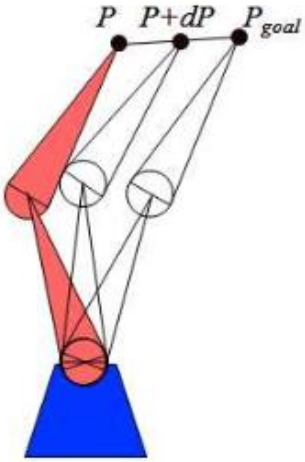
Дванаесетте нелинеарни елементи на матрицата од десно се или нули, или константи или функции од q_2 до q_6. Ако елементите од левата страна кои преставуваат функции од q_1се изедначат со елементите од десната страна, тогаш зглобот q_1 може да се пресмета како функции од $r_{11}r_{12} \dots r_{33}$, p_x, p_y, p_zИ фиксните параметри на роботската рака. Кога ќе се добие q_1другите зглобови ќе се добијат на сличен начин.

Од овој пример се гледа колку е комплексна инверзната кинематика за имплементација. Нема да се навлегува сега во детали на овој пример бидејќи, во наредните поглавја ќе биде разгледана подетално имплементацијата на инверзната кинематика врз роботска рака со два зглоба која се движи во 2D простор.

3.2. Јакобиев метод за решавање на проблем со инверзна кинематика

Имплементација на инверзна кинематика врз роботска рака всушност се состои од имплементација на еден или повеќе алгоритами во самиот софтвер на самата роботска рака.

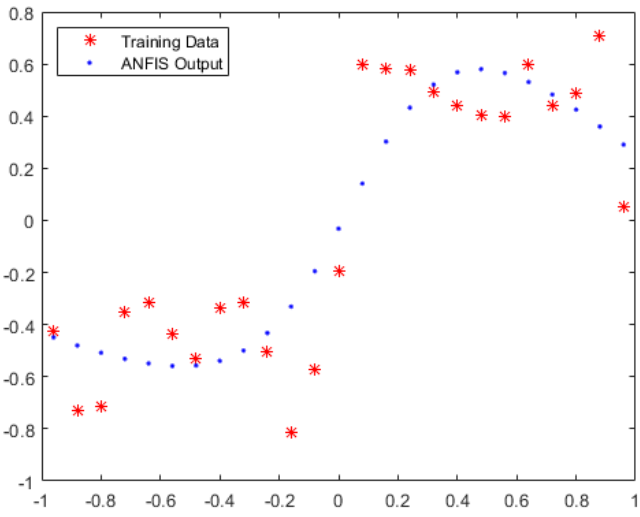
Инверзен Јакобиев метод работи со интерации. На тој начин итеративно аглите меѓу зглобовите се менуваат, така што при секоја интерација се повеќе и повеќе роботската рака се приближува до крајната позиција и ориентација. Јакобиевиот метод преставува матрица со димензии n x m, која содржи диференцијални промени на q врз диференцијалните промени на P (dP). На слика 22 сликовито е прикажан инверзниот Јакобиев метод.



Слика 22 - Инверзен Јакобиев метод

3.3. Adaptive Neuro Fuzzy Interface System (ANFIS)

Adaptive Neuro Fuzzy Interface System или на скратено ANFIS, всушност преставува комбинација на fuzzy логика и невронска мрежа. Невронските системи имаат повеќе влезни параметри но еден излез. Системите fuzzy имаат можност да го престават сеопфатното јазично знаење, со што ќе работи по дадени правила (како пример човечки експерт да ги напише правилата за работа). Но, системите fuzzy немаат механизам за автоматско стекнување или прилагодување на овие правила. Од друга страна невронските мрежи се приспособливи системи кои можат да се истренираат и прилагодат на множество од примери. Па така комбинацијата на невронските мрежи и системите fuzzy се нарекува ANFIS која се користи во нелинеарни апликации. При имплементација на ANFIS потребно е да има множество на податоци за тренинг со кои подоцна ќе биде пресметан и излезот од методот ANFIS. Пример на податоците за тренинг заедно со излезите на ANFIS се прикажани на графот подолу.



Слика 23 Пример: Податоци за тренинг и излезите на ANFIS

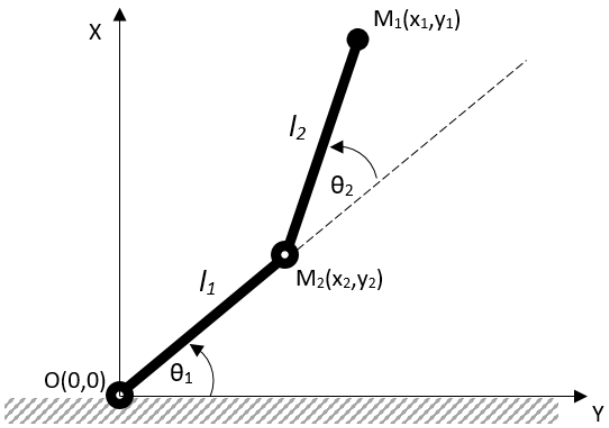
ANFIS имплементација е направена во овој труд и ќе биде подетално разгледана во следните поглавја. Со помош на математички метод ќе биде проверена имплементацијата на ANFIS и ќе види со колкава грешка пресметува ANFIS.

4. Осврт кон алгебарско аналитичкиот метод за решавање на проблемот на инверзна кинематика на роботска рака

Во овој труд анализиран е **алгебарско аналитички метод**, односно ***алгебарски метод со елементи од аналитичка геометрија***. Роботската рака работи во рамнина, односно ќе работи во дводимензионален простор и се состои од два зглоба. Долниот зглоб е фиксиран за земјата / почетната точка и во понатамошен контекст таа ќе биде статична.

Влезните параметри преставуваат координатите, додека аглите се излезите, што воедно преставува и излезите на инверзната кинематика.

Механичкото движење на роботската рака ќе ја вршат степ мотори, односно зглобовите на самата роботска рака ќе бидат степ мотори. Еден степ на самиот степ мотор, всушност преставува минималната аголна промена што еден зглоб може да ја направи. Бидејќи роботската рака може да има физички рестрикции при нејзино движење во дадена рамнина, при имплементација треба да бидат доставени и минималните и максималните агли на зглобовите (Θ_{min}, Θ_{max}). Пример за овој метод е прикажан на слика 24.



Слика24 – Позиција на роботската рака

4.1. Одредување на позицијата на вториот зглоб

Крајната или посакуваната точка е $M_1(x_1,y_1)$. За таа цел ќе биде разгледан триаголникот ΔOM_1M_2. Должините на сите три страни се познати и исто така познато е каде се наоѓаат во просторот двете точки. Единствено не е позната третата точка на триаголникот и таа точка преставува позиција на вториот зглоб каде роботската рака треба да се прекрши $M_2(x_2, y_2)$. За нејзина пресметка ќе бидат решени повеќе равенки.

$$\begin{cases} \boldsymbol{l_1{}^2 = x_2{}^2 + y_2{}^2} \quad \rightarrow \quad \boldsymbol{x_2{}^2 = l_1{}^2 - y_2{}^2} \\ \boldsymbol{l_2{}^2 = (x_1{}^2 - x_2{}^2)^2 + (y_1{}^2 - y_2{}^2)^2} \end{cases} \tag{47}$$

Овој систем на квадратни равенки може да се поедностави и реши:

$$l_2{}^2 = x_1{}^2 - 2x_1{}^2x_2{}^2 + x_2{}^2 + y_1{}^2 - 2y_1{}^2y_2{}^2 + y_2{}^2 \tag{48}$$

$$l_2{}^2 = x_1{}^2 + x_2{}^2 + y_1{}^2 + y_2{}^2 - 2(x_1x2 + y_1y_2) \tag{49}$$

$$l_2{}^2 = x_1{}^2 + l_1{}^2 - y_2{}^2 + y_1{}^2 + y_2{}^2 - 2(x_1x2 + y_1y_2) \tag{50}$$

$$l_2{}^2 = l_1{}^2 + x_1{}^2 + y_1{}^2 - 2(x_1x2 + y_1y_2) \tag{51}$$

$$2(x_1x_2 + y_1y_2) = l_1 - l_2 + x_1{}^2 + y_1{}^2 \tag{52}$$

$$C = x_1x_2 + y_1y_2 \tag{53}$$

$$D = l_1 - l_2 + x_1{}^2 + y_1{}^2 \tag{54}$$

$$2C = D \rightarrow C = \frac{D}{2} \quad \rightarrow \quad \boldsymbol{C = \frac{l_1 - l_2 + x_1{}^2 + y_1{}^2}{2}} \tag{55}$$

$$C = x_1\sqrt{l_1{}^2 - y_2{}^2} + y_1y_2 \tag{56}$$

$$C - y_1y_2 = x_1\sqrt{l_1{}^2 - y_2{}^2}/^2 \tag{57}$$

$$(C - y_1y_2)^2 = x_1{}^2(l_1{}^2 - y_2{}^2) \tag{58}$$

$$C^2 - 2Cy_1y_2 + y_1{}^2y_2{}^2 - x_1{}^2l_1{}^2 + x_1{}^2y_2{}^2 = 0 \tag{59}$$

$$\boldsymbol{ay^2 + by + c = 0} \rightarrow \boldsymbol{y_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}} \quad \rightarrow \tag{60}$$

$$y_{2_1} = \frac{2Cy_1 + \sqrt{4C^2{y_1}^2 - 4({x_1}^2 + {y_1}^2)(-{y_1}^2{y_1}^2 + C^2)}}{2({x_1}^2 + {y_1}^2)} \quad \textbf{(61)}$$

$$y_{2_2} = \frac{2Cy_1 - \sqrt{4C^2{y_1}^2 - 4({x_1}^2 + {y_1}^2)(-{y_1}^2{y_1}^2 + C^2)}}{2({x_1}^2 + {y_1}^2)} \quad \textbf{(62)}$$

Со равенките (61) и (62) ќе се пресметау-координататанаточкатаM2ќе се пресмета. Откако ќе се добие вредност за у координатата со помош на $x_2{}^2 = l_1{}^2 - y_2{}^2$ ќе ја пресметаме и *x* координата. Од дадената равенка се забележува дека можни се четири решенија за *x* координатата, равенка (63) и равенка (64).

$$x_{2_1} = \sqrt{{l_1}^2 - {y_{2_1}}^2} x_{2_3} = -\sqrt{{l_1}^2 - {y_{2_1}}^2} \quad (63)$$

$$x_{2_2} = \sqrt{{l_1}^2 - {y_{2_2}}^2} x_{2_4} = -\sqrt{{l_1}^2 - {y_{2_2}}^2} \quad (64)$$

Па така се вкупно за M_2 точката ќе добиеме 4 можни точки .

$$A(x_{2_1}, y_{2_1}), B(x_{2_2}, y_{2_2}), C(x_{2_3}, y_{2_1}), D(x_{2_4}, y_{2_2})$$

Но наоѓањето на овие точки не значи дека роботската рака може на секоја од овие да дојде, иако математичките решенијата се точни. На самиот почеток е кажано дека секоја роботска рака е ограничена со должина и насока и агол на движење. Па така треба да се изврши проверка за секоја од овие точки дали ги задоволуваат сите овие правила, односно дали да се пресметаат аглите Θ_1 и Θ_2 и да се провери дали се помеѓу Θ_{min}, $\Theta_{max.}$

Слична таква проверка треба да се направи и за должината на деловите на роботската рака, односно да се пресмета должината на роботската рака со помош на равенката за должина на растојание меѓу две точки, равенка (65). Пресметаното растојание треба да биде исто или приближно на веќе познатите вредности на роботската рака, односно равенките (66) и (67).

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad \textbf{(65)}$$

$$d_1 \approx l_1 \quad \textbf{(66)}$$

$$d_2 \approx l_2 \quad \textbf{(67)}$$

4.2. Одредување на аглите Θ_1 и Θ_2

Откако ќе се добијат координатите на сите можни позиции на зглобот, потребно е да се пресметаат аглите на соодветните зглобови, односно Θ_1 и Θ_2. Оваа пресметка се прави со цел да се заврши имплементацијата на инверзната кинематика за дадениот проблем, но и да се направи последната проверка и филтрација. Филтрирањето се прави на тој начин што откако ќе се добијат аглите се проверуваат дали се во опсег на спецификацијата на роботската рака ($\Theta_{min} \leq \Theta_{1/2} \leq \Theta_{max}$).

За пресметување на аглите ќе се користи формулата за пресметување на агли меѓу два вектора.

$\vec{a} = \{a_x, a_y\}$,

$\vec{b} = \{b_x, b_y\}$,

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|a| \cdot |b|} \tag{68}$$

Каде што,

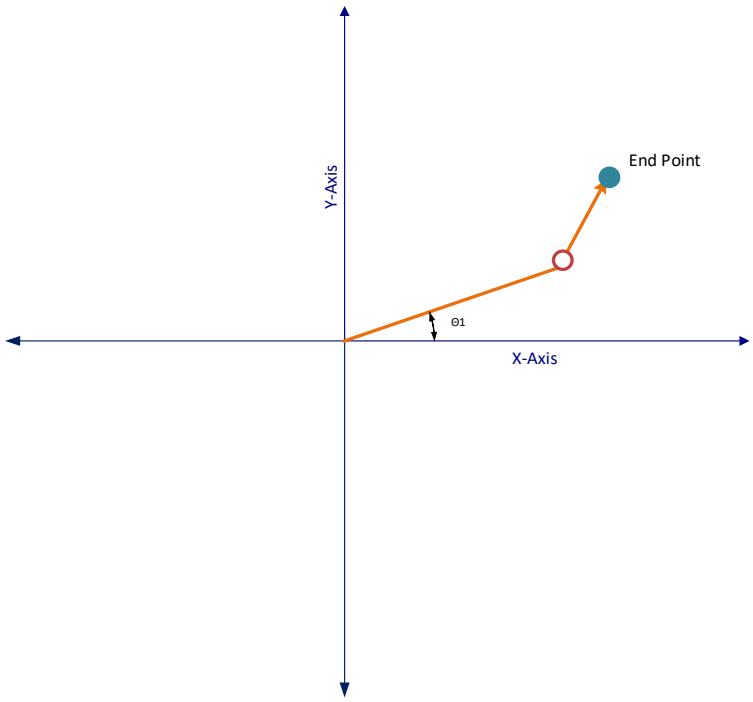
$$\vec{a} \cdot \vec{b} = a_x \cdot b_x + a_y \cdot b_y \tag{69}$$

$$|a| = \sqrt{{a_x}^2 + {a_y}^2} \tag{70}$$

$$|b| = \sqrt{{b_x}^2 + {b_y}^2} \tag{71}$$

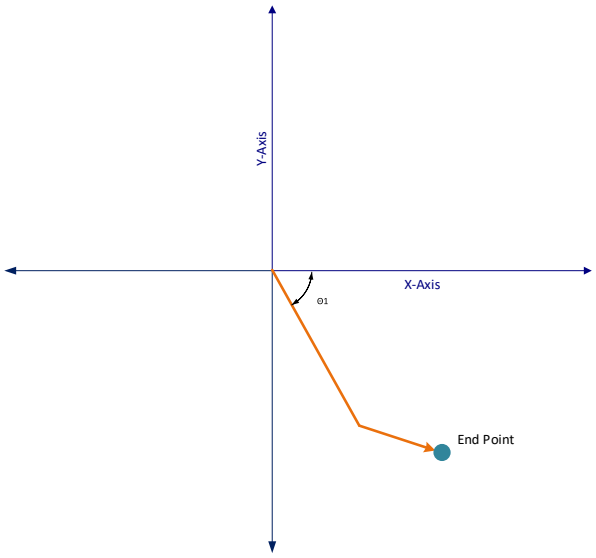
Одредување на Θ_1

Од равенката (68) се добива вредност на аголот изразена во радијани, кој всушност е агол помеѓу првиот дел на раката и X оската. Оваа вредност за аголот е точна само доколку раката се наоѓа во првиот и вториот квадрант, бидејќи раката се движи во обратна насока на стрелките на часовникот и како што е наведено на самиот почеток секоја роботска рака има свои лимити. За вектор $\vec{a}$ќе биде земен векторот$\vec{a} = (100, 0)$.



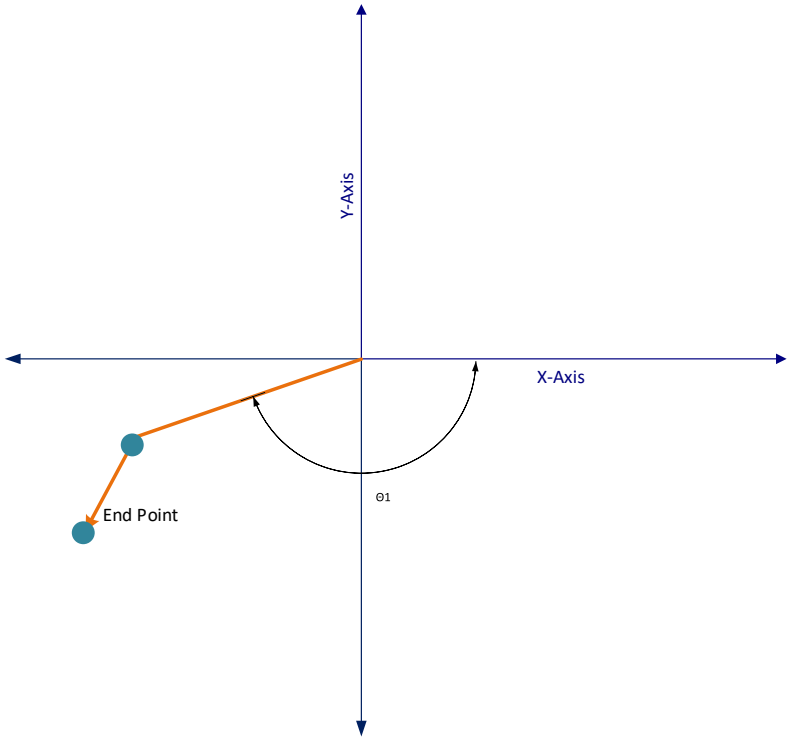
Слика 25 – Пример, аголот Θ_1 добиен со равенката доколку првиот зглоб се наоѓа во првиот квадрант

Во третиот и четвртиот квадрант, за да се добие посакуваниот агол, потребно е аголот добиен со равенката (68) да се одземе од 2π. Ова се случува бидејќи векторот $\vec{a}$е денифиран со почеток (0,0) и насока кон десно односно $\vec{a} = (100, 0)$ Аголот што ќе се добие со равенката (68) е прикажан на слика 26 -*Пример, аголот Θ_1 добиен со равенката доколку првиот зглоб се наоѓа во четвртиот квадрант*..



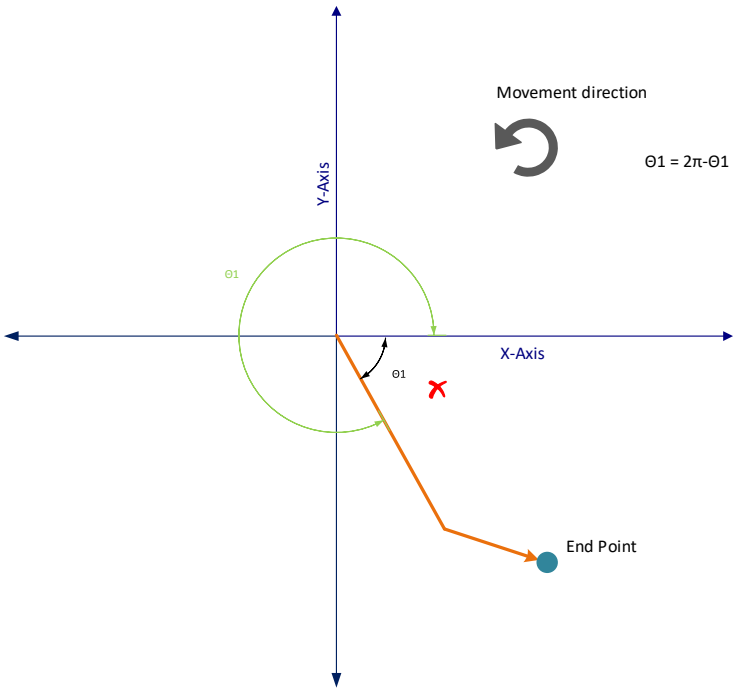
Слика 26 - Пример, аголот Θ_1 добиен со равенката доколку првиот зглоб се наоѓа во четвртиот квадрант

Истото важи и доколку првиот зглоб се наоѓа во третиот квадрант. Аголот што ќе биде пресметан со равенката (68) е прикажан на слика 27.

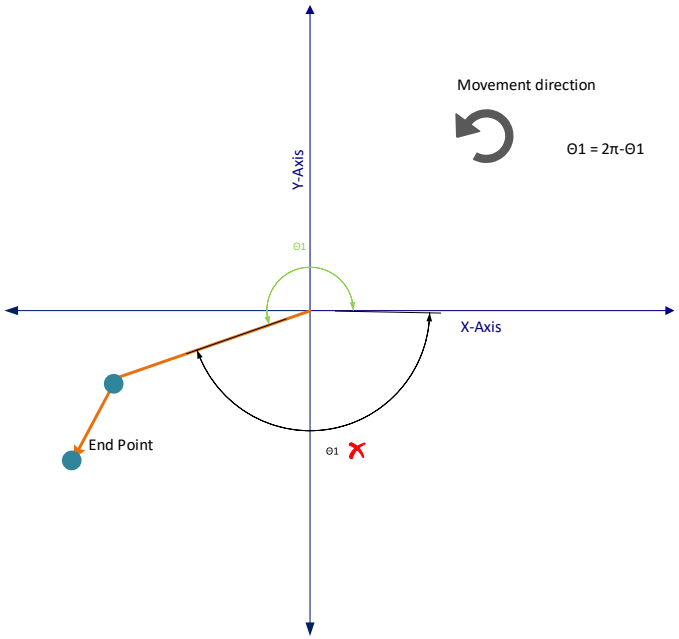


Слика 27- Пример, аголот Θ_1 добиен со равенката доколку првиот зглоб се наоѓа во третиот квадрант

Аголот што всушност ќе треба во оваа имплементација ќе треба е надворешниот агол од добиениот или сликовито прикажано, посакуваниот агол е:

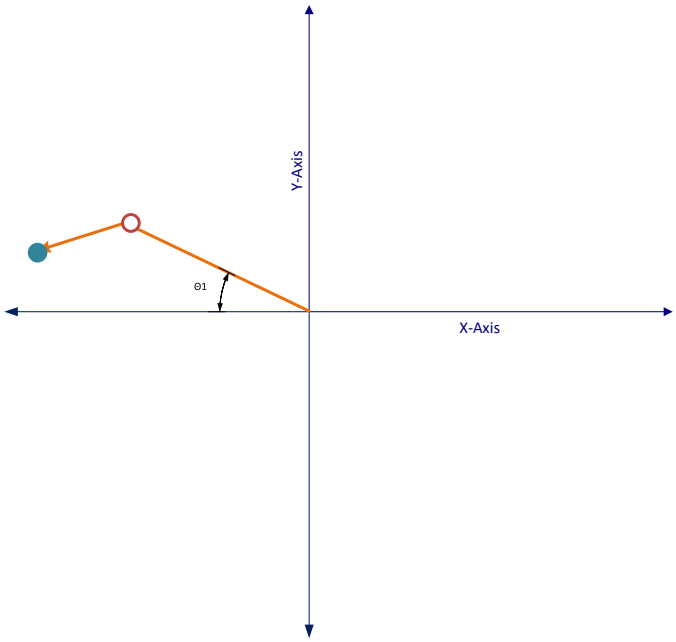


Слика 28 - Пример; Добиениот агол со равенката во овој случај е грешен, доколку првиот зглоб се наоѓа во четвртиот квадрант. Посакуваниот аголот Θ_1 е надворешниот означен со зелено.



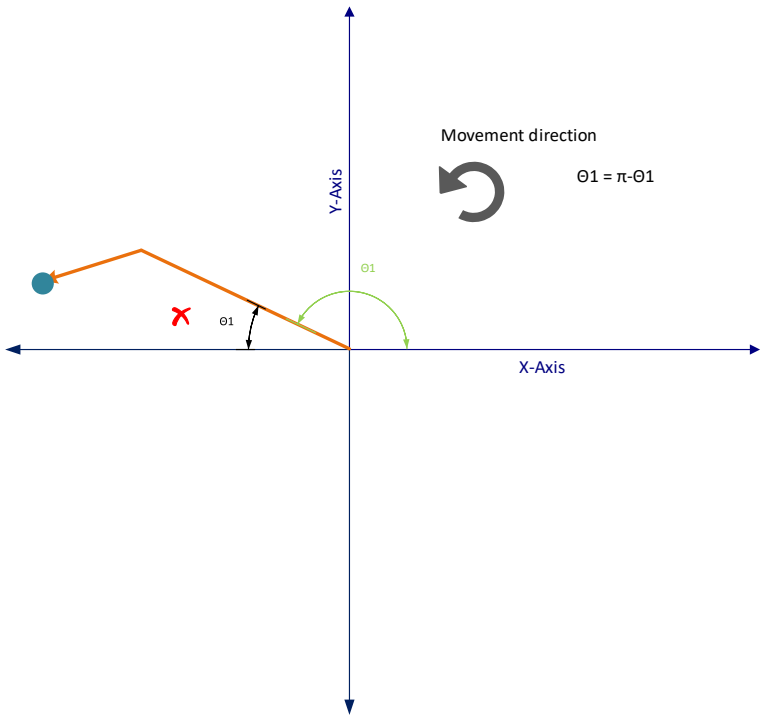
Слика 29 - Пример; Добиениот агол со равенката во овој случај е грешен, доколку првиот зглоб се наоѓа во третиот квадрант. Посакуваниот аголот Θ_1 е надворешниот означен со зелено

Доколку локацијата на првиот зглоб надмине повеќе од 90 степени, но за вектор $\vec{a}$ се земе да биде $\vec{a} = (-100, 0)$, тогаш аголот добиен од равенката ќе преставува аголот помеѓу првиот дел од роботската рака и X оската, но гледано во вториот квадрант (слика 30).



Слика 30 - Добиениот агол со равенката доколку векторот на x оската е дефиниран како $\vec{a} = (-100, 0)$

Математички овој добиен агол е точниот агол помеѓу два вектора, но во реална имплементација наспоменавме дека роботската рака ќе се движи обратно од стрелките на часовникот почнувајќи од првиот квадрант. Добиениот агол во овој случај е грешен. Посакуваниот агол што ќе треба да се пресмета доколку раката се наоѓа во II квадрант е прикажан на слика 31.



Слика 31 - Пример; Добиениот агол со равенката доколку векторот на x оската е дефиниран како $\vec{a} = (-100, 0)$, во овој случај е грешен. Посакуваниот агол е надворешниот означен со зелено.

Добиениот агол треба да се одземе од π.

Затоа при имплементација многу важен фактор е векторот a. Треба внимателно да се одреди неговата насока и според тој вектор да се имплементира целото решение.

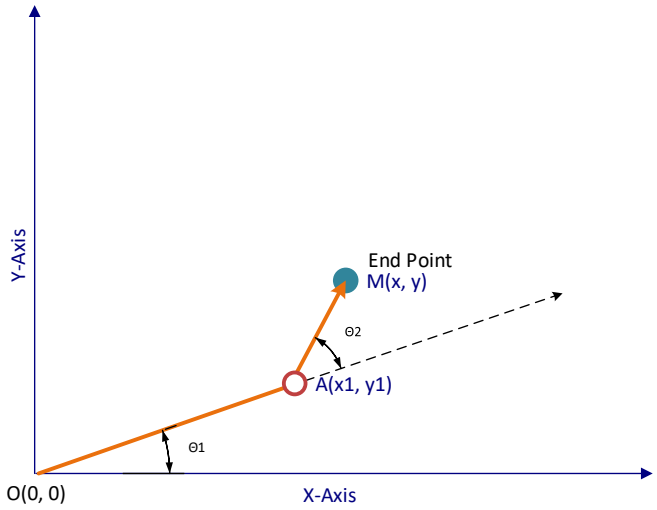
За вектор $\vec{a}$ ***кој е дефиниран како*** $\vec{a} = (100, 0)$***, аголот:***

- ***I квадрант:*** $\theta = \theta$ ***(аголот добиен со равенката (22) е посакуваниот)***
- ***II квадрант:*** $\theta = \theta$
- ***III квадрант:*** $\theta = 2\pi - \theta$
- ***IV квадрант:*** $\theta = 2\pi - \theta$

Одредување на Θ_2

За разлика од Θ_1, Θ_2 аголот се одредува на поспецифичен начин. Овде аголот Θ_2 зависи од положбата на зглобот на роботската рака. Со други

зборови кажано, зависи од положбата и насоката на првиот дел од роботската рака (векторот $\overrightarrow{OA}$).



Слика 32 – Аголот Θ_2, во однос на Θ_1 и крајната положба

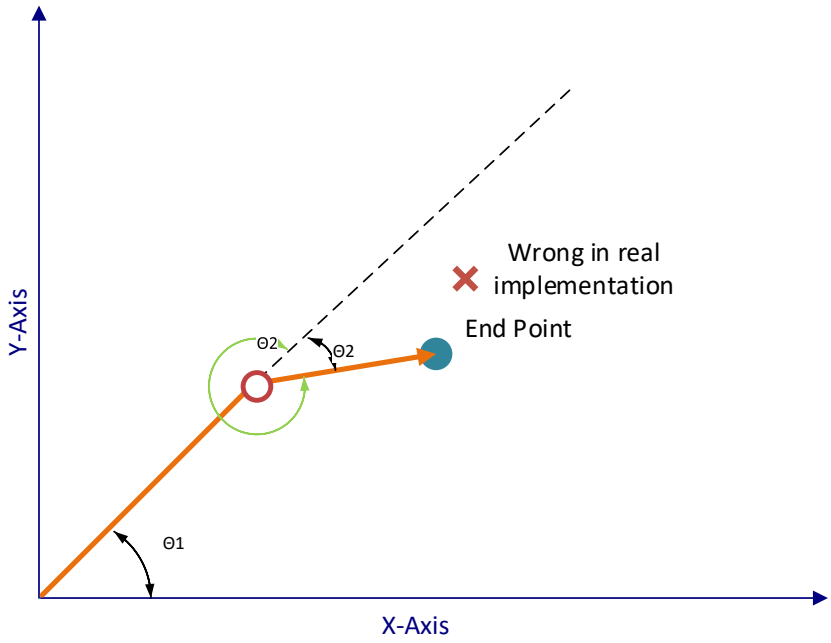
Од слика 32 се забележува дека за да се одреди аголот потребни податоци се:

- Векторот $\overrightarrow{OA}$
- Крајната посакувана положба на роботската рака (точката M)

И во овој случај ќе се искористи равенката (68) за да се одреди аголот Θ_2, но притоа треба да се земе предвид одредена комплексност.

Во првиот случај (за аголот Θ_1) се користи статичната X-оска и првиот променлив дел на роботската рака, но овде имаме два променливи дела, односно првиот дел и вториот дел на роботската рака (зависен од првиот дел). Аголот Θ_2 зависи од положбата на зглобот односно точката A.

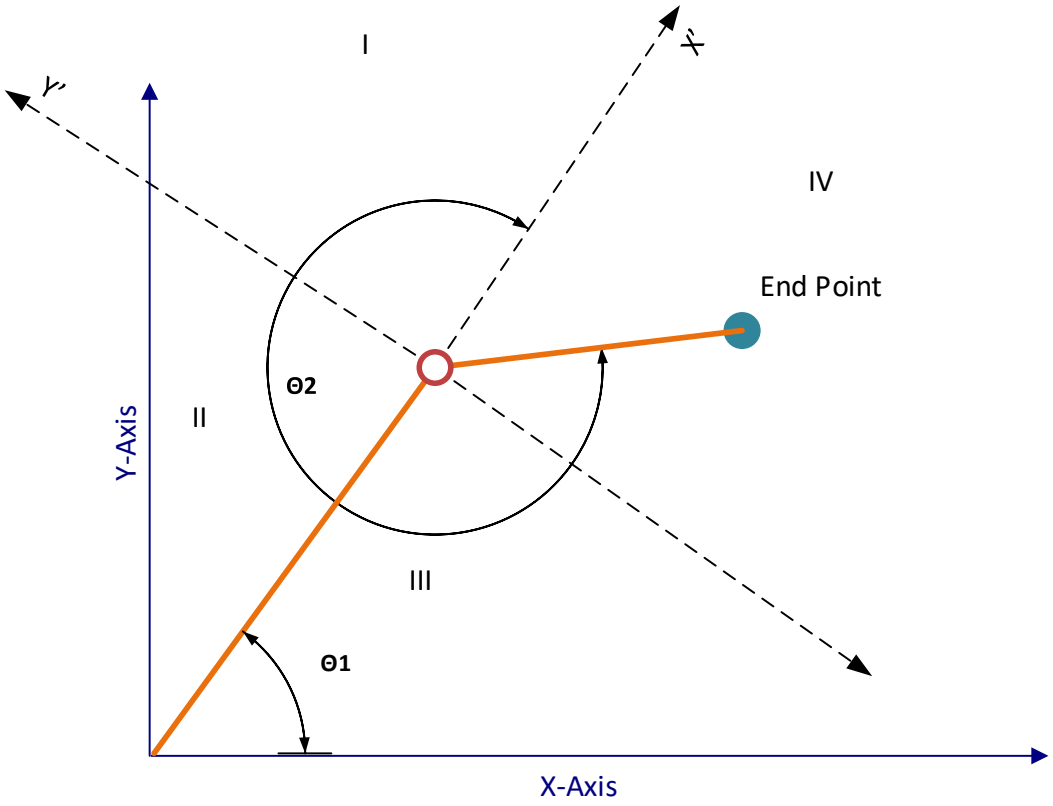
За да се пресмета аголот со равенката (68), најпрво треба да се добие положбата на вториот зглоб односно да се дефинира точката A и ќе се дефинираат векторите $\vec{a} = \overrightarrow{OA}$ и $\vec{b} = \overrightarrow{AM}$. Но овде може да се јави сличен проблем како при пресметување на Θ_1. Односно на слика 33 се гледа:



Слика 33 – Пример, вредноста на аголот Θ2 добиена со равенката е грешен доколу раката се движи во обратна насока на стрелките на часовникот

Во овој случај равенката математички ќе го даде точниот агол, но за наша употреба ни треба надворешниот агол, односно $2\pi - \theta_2$.

За да се определи дали добиениот агол од равенката (68) е точниот или треба да се определи надворешниот агол ќе се опише следното сценарио. Од слика 33 се забележува дека почетокот на аголот θ_2 започнува од насоката на векторот $\vec{a}$. Па така може слободно да се каже дека точката *A* преставува нов координатен почеток, а векторот $\vec{a}$ преставува новата замислена x' оска. За да се добие y'оска низ точката *A*, треба да се одреди нормален вектор на векторот $\vec{a}$. Ова е прикажано наслика 34.



Слика 34 – Формирање на вториот координатен систем, кој е задолжен за аголот Θ2

За да се определи y' оската, односно за да се определи нормалниот вектор на веќе познат вектор ќе се искористи условот:

$$\vec{a} \cdot \vec{b} = |a||b| \cos\theta \tag{72}$$

Бидејќи $\cos 90 = 0$ тогаш следува дека:

$$\vec{a} \cdot \vec{b} = 0 \tag{73}$$

$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y \tag{74}$$

Од равенката (74) ќе се определи единичниот векторот $\vec{b}$ кој ќе биде нормален на векторот $\vec{a}$и следно што ќе се пресмета е во кој квадрант се наоѓа точката M од ново создадениот квадратен систем. Откако ќе се дознае во кој квадрант се

наоѓа за да се добие посакуваниот агол ќе се пресмета со истата логика како и аголот θ_1.

- I квадрант: $\theta = \theta$ (аголот добиен со равенката (68) е посакуваниот)
- II квадрант: $\theta = \pi - \theta$
- III квадрант: $\theta = \pi + \theta$
- IV квадрант: $\theta = 2\pi - \theta$

5. ANFIS (Adaptive Neuro Fuzzy Interface System) како метод за решавање на проблемот на инверзна кинематика на роботска рака

5.1. Што е ANFIS

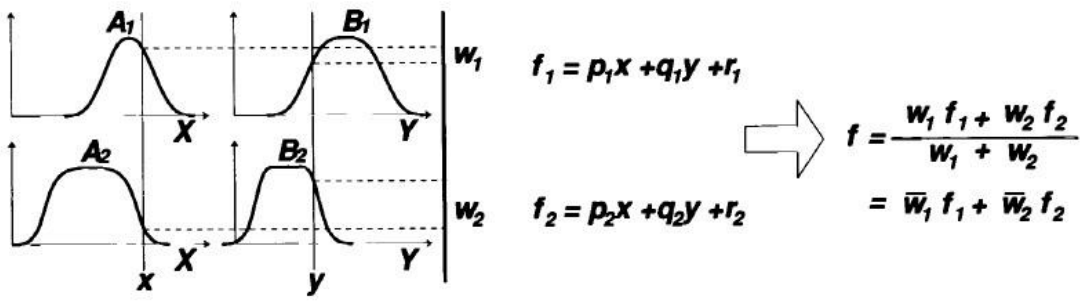
Adaptive Neuro Fuzzy Interface System или на скратено ANFIS, всушност преставува комбинација на логика fuzzy и невронска мрежа. Невронските системи имаат повеќе влезни параметри, но еден излез. Системите fuzzy имаат можност да го престават сеопфатното јазично знаење, со што ќе работи по дадени правила (како пример, човечки експерт да ги напише правилата за работа). Но, системите fuzzy немаат механизам за автоматско стекнување или прилагодување на овие правила. Од друга страна невронските мрежи се приспособливи системи кои можат да се истренираат и прилагодат на множество од примери. Па така комбинацијата на невронските мрежи и системите fuzzy се нарекува ANFIS која се користи во нелинеарни апликации. При имплементација на ANFIS потребно е да има множество на податоци за тренинг со кои подоцна ќе биде пресметан и излезот од методот ANFIS.

5.2. Архитектура на ANFIS

Архитектурата на ANFIS најдобро ќе се разбере со помош на даден пример. Да се претпостави дека за еден фази систем имаме два влеза *x* и *y* и еден излез *z*. Постојат и две правила:

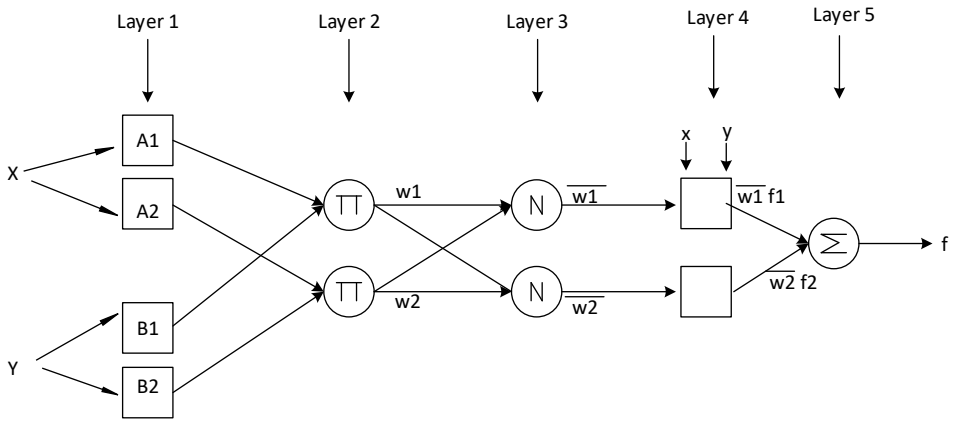
- Правило 1: Ако x е A_1 и y е B_1, тогаш $f_1 = p_1x + q_1y + r_1$
- Правило 2: Ако x е A_2 и y е B_2, тогаш $f_2 = p_2x + q_2y + r_2$

Графички приказ на правилата:



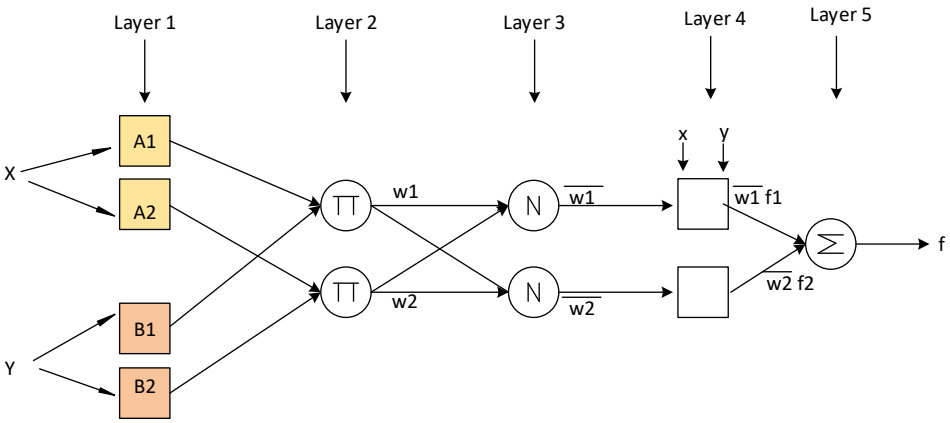
Слика 35 – Пример за графички приказ на правилата

Еквивалентно на приказот архитектурата на ANFIS е:



Слика 36 – Архитектура на ANFIS

5.2.1. Ниво 1 (Layer 1)



Слика 37 – Ниво 1 од архитектурата на ANFIS

Секое квадратче *I* овде преставува **прилагоден јазол** (**adaptive node**) со јазлеста функција:

$$O_{1,i} = \mu A_i(x), \qquad \text{за } i = 1,2 \text{ или} \tag{75}$$

$$O_{1,i} = \mu B_{i-2}(y), \qquad \text{за } i = 3,4 \tag{76}$$

Со други зборови $O_{1,i}$ **е членска оценка** (**membership grade**) на множеството fuzzy A_i, B_i.

Членската функција може да биде која било соодветна параметарска функција, како на пример, општата функција ѕвоно (**generalized bell function**):

$$\boldsymbol{\mu_A(x) = \frac{1}{1+\left|\frac{x-c_i}{a_i}\right|^{2b}},} \tag{77}$$

При што a_i, b_i, c_i, се параметрите. Овие параметри на ова ниво се познати како **теоретски параметри (premise parameters**).

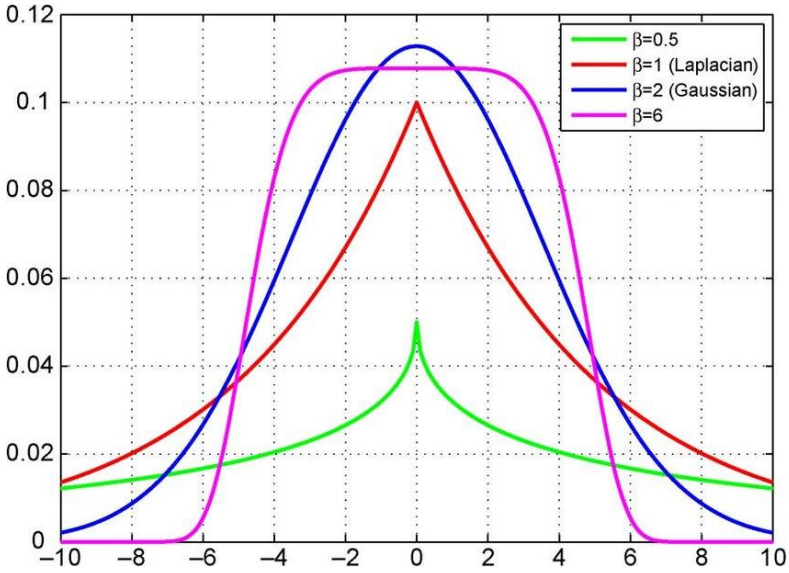
Друга функција која може да се искористи како членска функција и покажала добри резултати е **Гаусовата функција (Gaussian membership function)**:

$$\boldsymbol{\mu_A(x) = exp\left[-\frac{(x-c_i)^2}{2{a_i}^2}\right]} \tag{78}$$

Општата Гаусова членска функција може да се изрази како:

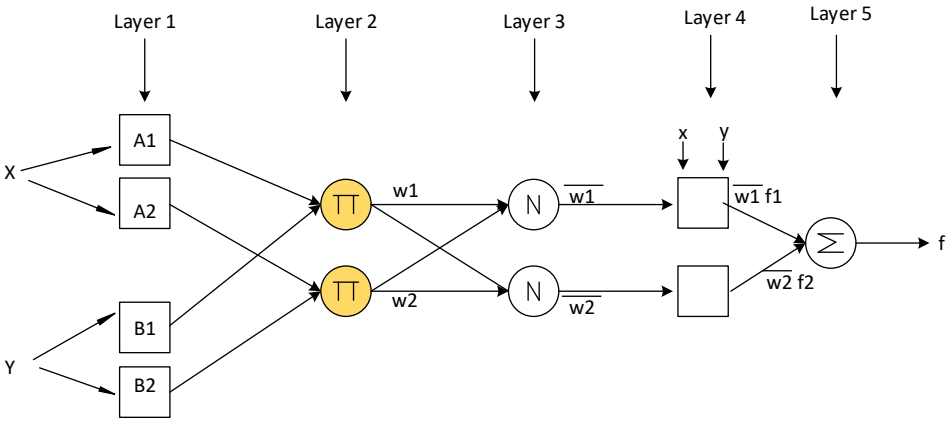
$$\mu_A(x) = exp\left[-\left(\frac{x-c_i}{a_i}\right)^{\beta}\right] \tag{79}$$

Каде β преставува фактор за облик на гаусовата функција. Доколку $\beta = 2$ тогаш функцијата ќе има класичен гаусов облик, но доколку $\beta = 0.5$ или $\beta = 6$ тогаш функцијата ќе има триаголен односно трапезоиден облик.



Слика 38 – Облик на Гаусовата функција во однос на факторот β

5.2.2. Ниво 2 (Layer 2)



Слика 39 – Ниво 2 од архитектурата на ANFIS

Овде секој јазол преставува фиксен јазол (П), каде излез преставува продукт на сите влезни сигнали, односно:

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_i}(y),\ i = 1,2 \tag{80}$$

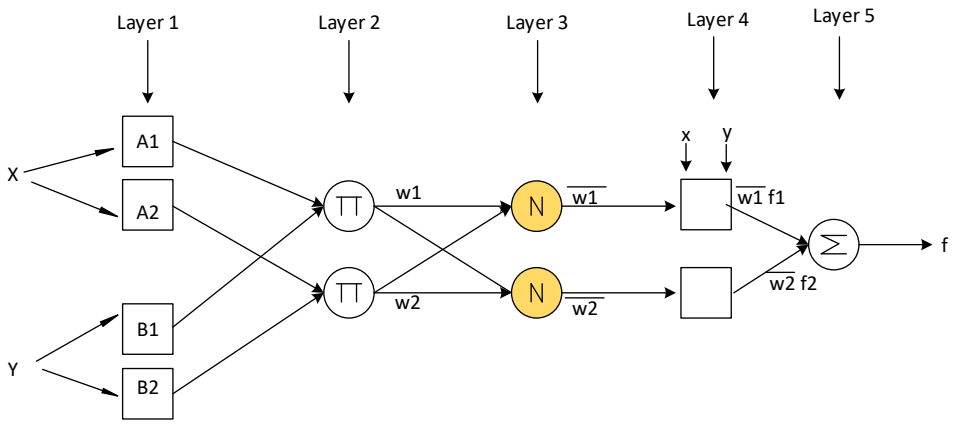
Излезот на секој јазол овде преставува јачината на даденото правило. Како функција овде може да се искористи која било T-norm операција која fuzzy ја подржува.

5.2.3. Ниво 3 (Layer 3)

Кај третото ниво, секој јазол е фиксен јазол (N), каде излез преставува продукт на сите влезни сигнали од претходното ниво. Излезот овде на секој *i*-ти јазол преставува процент на јачината на извршување на *i*-тото правило во однос на збирот од сите јачини на извршување на правилата.

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2}, i = 1,2 \tag{81}$$

Излезите на ова ниво се нарекуваат **нормализирани јачини на извршување (normalized firing strengths).**



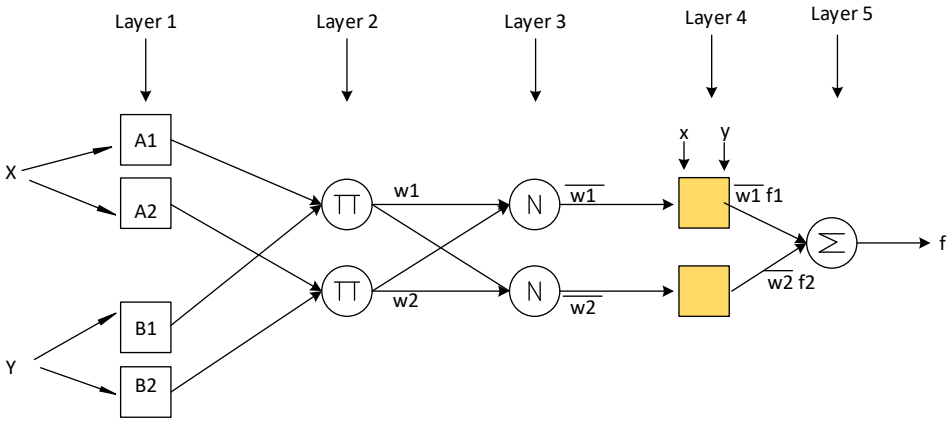
Слика 40 – Ниво 3 од архитектурата на ANFIS

5.2.4. Ниво 4 (Layer 4)

Секој јазол овде е прилагодлив јазол со функција:

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i), \tag{82}$$

Каде $\overline{w_i}$се нормализираните јачини на извршување од ниво 3 и p_i, q_i, r_i се параметри на ова ниво. Овие параметри на ова ниво уште се викаат и **последователни параметри (consequent parameters).**

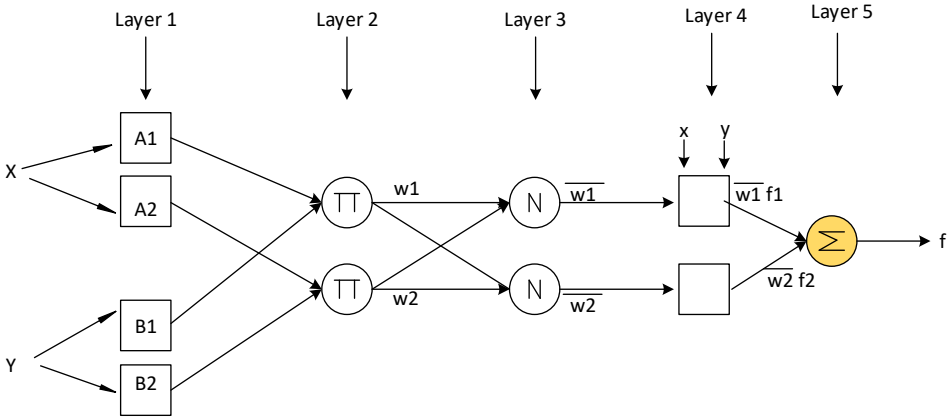


Слика 41 – Ниво 4 од архитектурата на ANFIS

5.2.5. Ниво 5 (Layer 5)

Единичниот јазол овде е фиксен јазол кој го калкулира излезот на самиот процес и тоа преку сумирање на сите влезни сигнали од претходното ниво.

$$output = O_{5,i} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \tag{83}$$



Слика 42 – Ниво 5 од архитектурата на ANFIS

5.3. Хибриден алгоритам за учење

Кога теоретските параметри (premise parameters) се фиксни тогаш излезот (резултатот) може да се претстави како линеарна комбинација на последователните параметри (consequent parameters).

$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_1}{w_1 + w_2} f_2$$

$$= \overline{w_1}(p_1 x + q_1 y + r_1) + \overline{w_2}(p_2 x + q_2 y + r_2)$$

$$= (\overline{w_1} x)p_1 + (\overline{w_1} y)q_1 + (\overline{w_1})r_1 + (\overline{w_2} x)p_2 + (\overline{w_2} y)q_2 + (\overline{w_2})r_2 \quad (84)$$

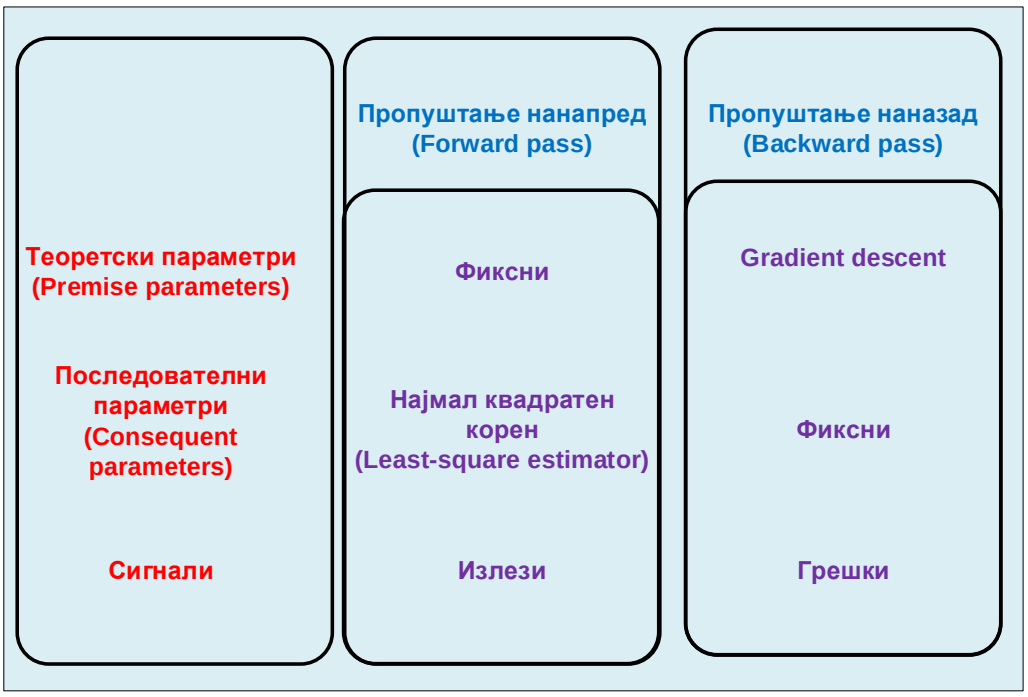
При имплементација на ANFIS систем и неговиот алгоритам за учење, целта е да се пресметаат теоретските и последователните параметри (**premise and consequent parameters**).

Хибридниот алгоритам за учење се состои од 2 дела:

- Пропуштање нанапред (Forward pass).
- Пропуштање наназад (Backward pass).

Во пропуштањето нанапред од хибридниот начин на учење, излезите од секое ниво одат нанапред до нивото 4 (Layer 4) и последователните параметри (consequent parameters) се пресметуваат со помош алгоритамот на најмал корен (least squares). Додека во пропуштање наназад грешката патува назад и теоретските параметри (premise parameters) се ажурираат користејќи gradient descendent.

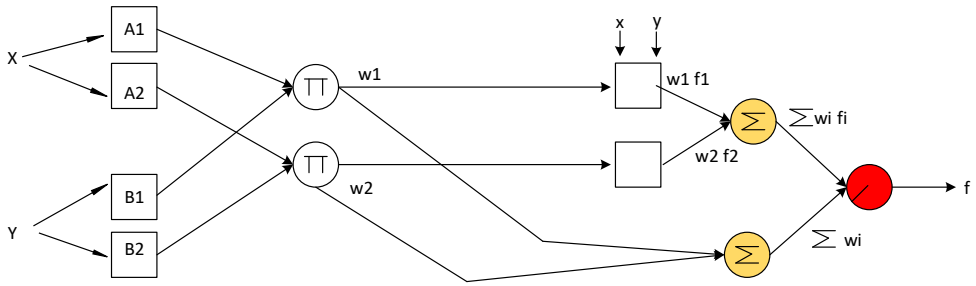
На слика 43 се прикажани разликите меѓу овие два дела.



Слика 43 – Хибриден начин на тренирање на ANFIS

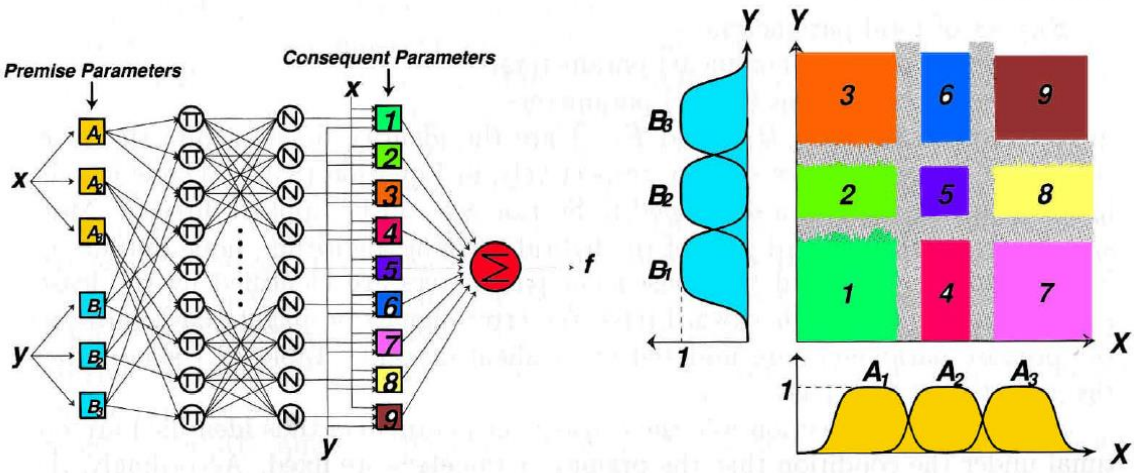
5.4. Примери на архитектура ANFIS

Забелешка за претходната архитектура е тоа што треба да се наспомене дека оваа архитектура не е единечна и може да се прилагоди во зависност од потребите на системот. Како пример структурата може да изгледа вака:



Слика 44 – Пример за модификација на архитектурата на ANFIS

Друг пример на ANFIS архитектура во која има два влезни сиглана и 9 правила би изгледала вака:



Слика 45 – Пример на архитектура на ANFIS со повеќе влезови

5.4.1. Пример за добивање на излезот од веќе истренирана ANFIS мрежа

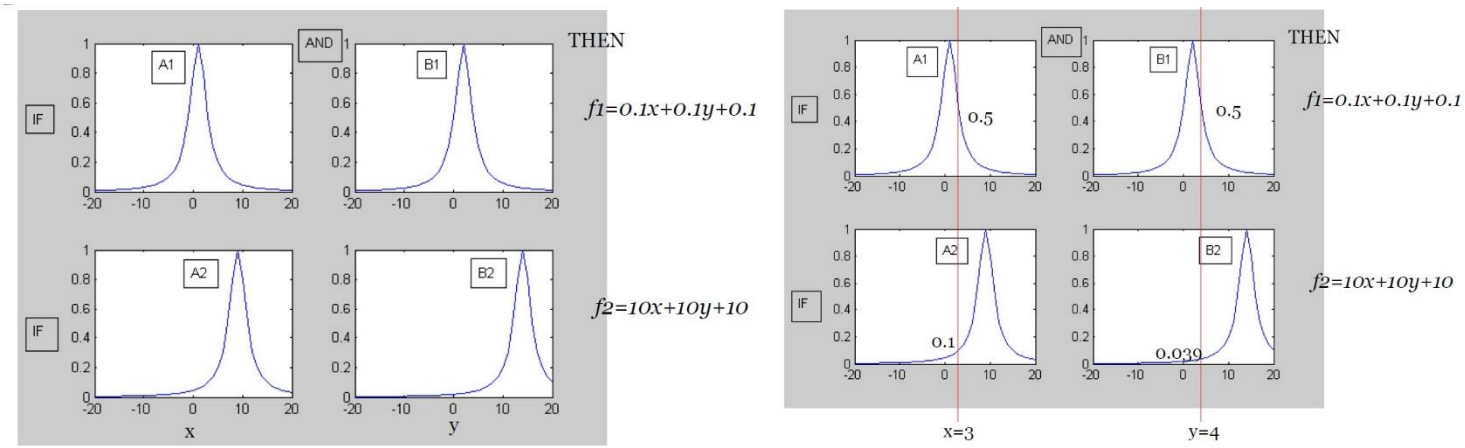
Rule 1: IF x is small (A1) AND y is small (B1) THEN f1=small
Rule 2: IF x is large (A2) AND y is large (B2) THEN f2=large

A1: $\mu_{A1}(x)=\frac{1}{1+\left|\frac{x-1}{2}\right|^2}$ B1: $\mu_{B1}(y)=\frac{1}{1+\left|\frac{y-2}{2}\right|^2}$ $f1=0.1x+0.1y+0.1$

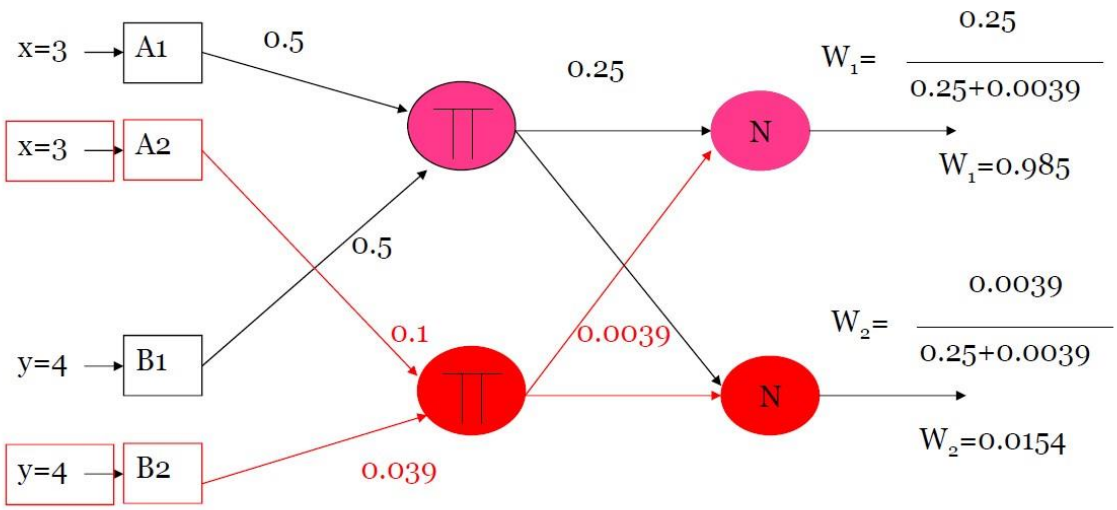
A2: $\mu_{A2}(x)=\frac{1}{1+\left|\frac{x-9}{2}\right|^2}$ B2: $\mu_{B2}(y)=\frac{1}{1+\left|\frac{y-14}{2}\right|^2}$ $f2=10x+10y+10$

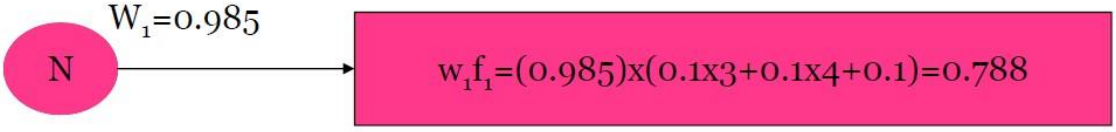
Given the trained fuzzy system above and input values
of x=3 and y=4, find output of the Sugeno fuzzy system

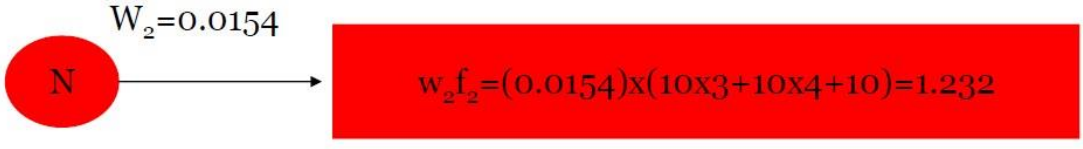
Во овој пример за членска функција (membership function) е земена општата функција ѕвоно (generilsed bell function).

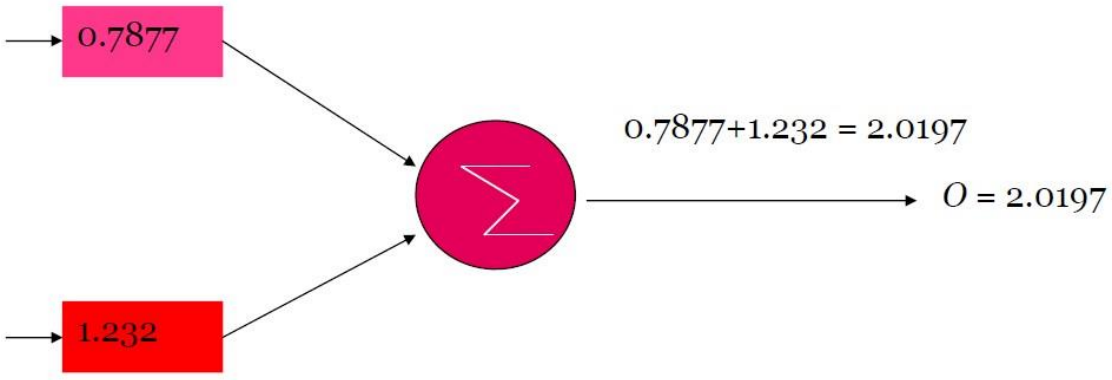


Слика 46 – Членска функција ѕвоно


x=3
A1
0.5
x=3
A2
0.25
W_1= 0.25 / 0.25+0.0039
N
W_1=0.985
0.5
0.1
0.0039
W_2= 0.0039 / 0.25+0.0039
y=4
B1
N
W_2=0.0154
y=4
B2
0.039


W_1=0.985
N
w_1f_1=(0.985)x(0.1x3+0.1x4+0.1)=0.788


W_2=0.0154
N
w_2f_2=(0.0154)x(10x3+10x4+10)=1.232


0.7877
0.7877+1.232 = 2.0197
O = 2.0197
1.232

6. Веб базирана имплементација на ANFIS за решавање на проблемот на инверзна кинематика на роботска рака

Симулацијата на аналитичкиот метод ќе биде разгледана со помош на нејзина имплементација во програмскиот јазик .Net, поточно ќе биде развиена MVC (ModelViewController) апликација која ќе служи за симулација на инверзната кинематика на роботската рака.

Апликацијата ќе биде .NET MVC, која влезните податоци, односно карактеристиките за роботската рака ќе се внесуваат преку text input полиња и преку даден Controller ќе се изврши калкулацијата и резултатот пак ќе се врати на делот View каде што со JavaScript framework (Three.js) ќе бидат прикажани на UI (User Interface).

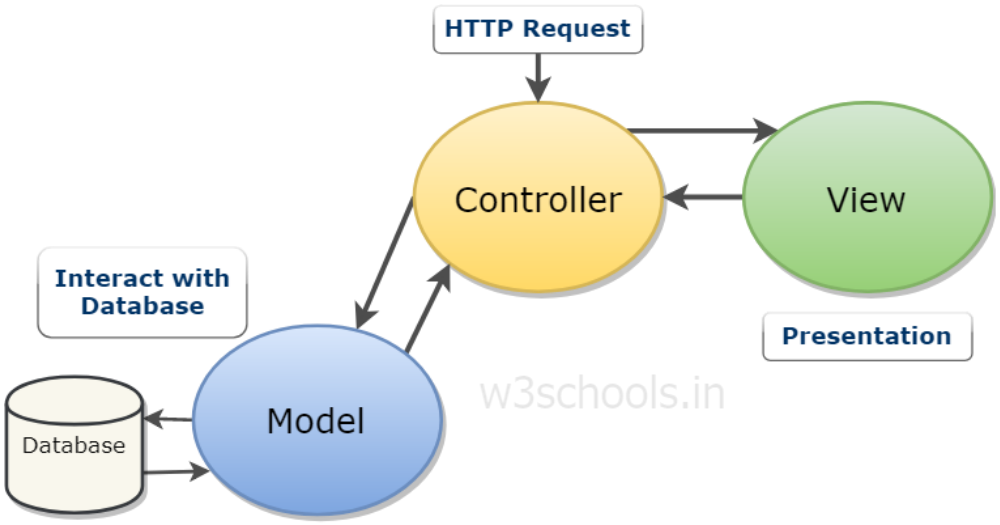


Fig: MVC Architecture

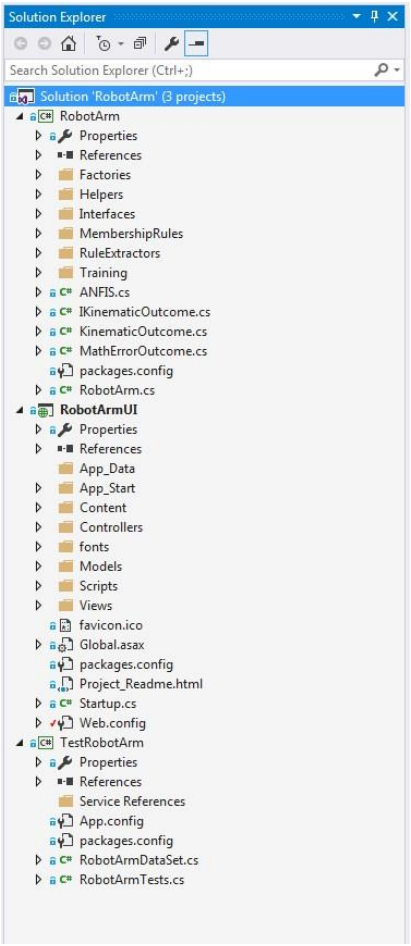
Слика 47 – MVC(ModelViewController) дијаграм

Технологии што се користени за симулација:

- **.NET, MVC (Model View Controller)**
  - **UI (UserInterface)**
    - **JQuery**
    - **Three.js**
    - **CSS**
    - **HTML and Razor from MVC**
  - **Back End (WorkForce)**
    - **Controller from MVC**
    - **Async and multithreading Programming in C# 7**

Како IDE во оваа симулација ќе се користи Visual Studio 2017.

Архитектурата на самото решение е составена од три потпроекти (апликации):



Слика 48 – Структура на проектите

1. Проект за самата имплементација на инверзната кинематика врз роботската рака. Нејзините методи, сите помошни методи и итн. (RobotArm)
2. Апликација за прикажување и обработување на резултатите добиени од некој од методите од првиот проект. Овој проект е MVC. (RobotArmUI.)
3. Проект во кој се тестираат функционалностите на инверзната кинематика. (TestRobotArm.)

Имплементацијата на инверзната кинематика во проектот е добиена со помош на пристапот TDD (Test Driven Development).

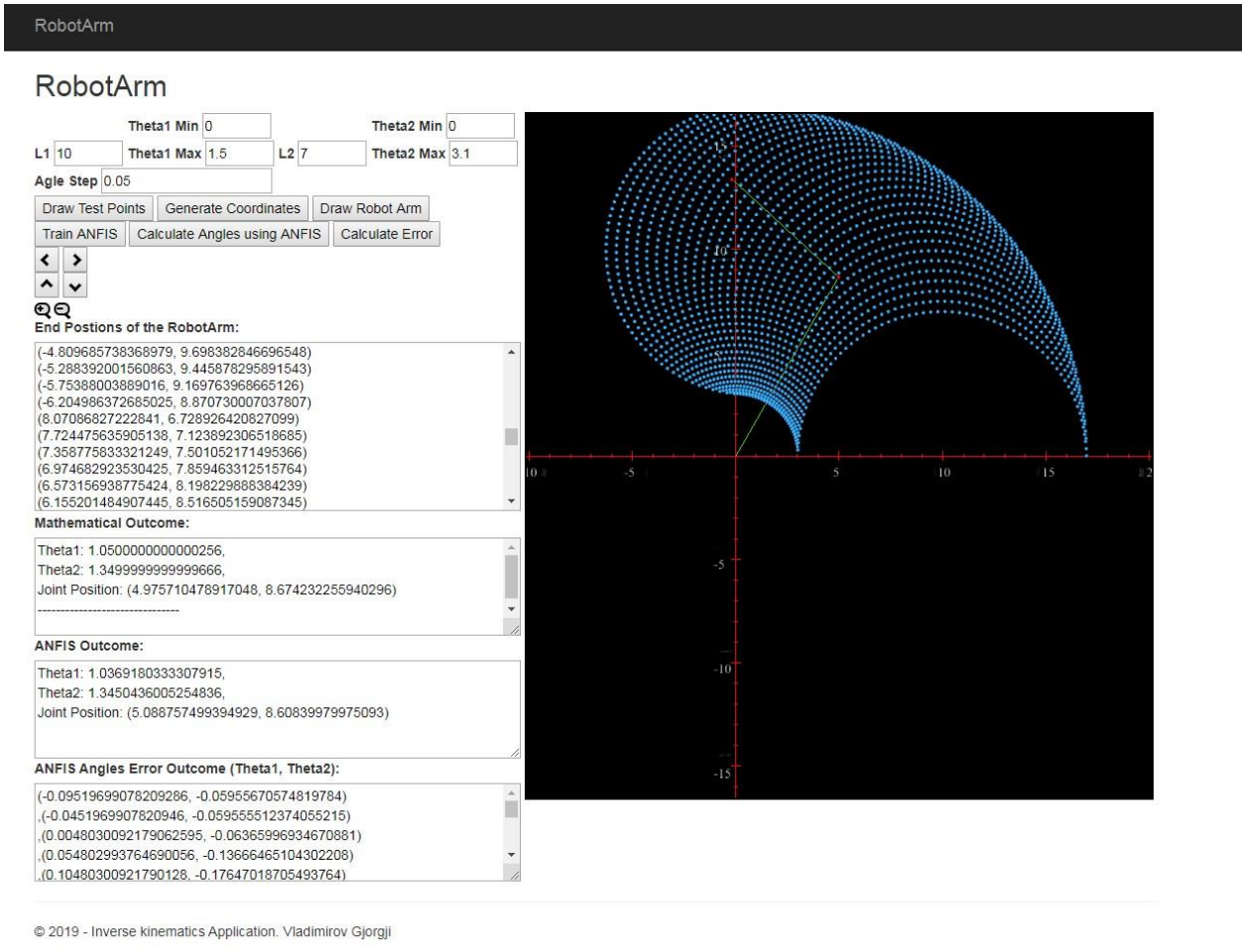
Одбрана е оваа архитектура бидејќи, многу попрегледно и полесно за одржување е кога е поделена на повеќе помали делови кои взаемно комуницираат. Но и покрај овие работи главна цел за тоа зошто имплементацијата е одвоена од прикажувањето е поради независност и повторно искористување во натамошни проекти. Многу полесно за

имплементација ќе биде доколку треба да се искористи друг проект за UI, на пример да кажеме дека UI треба да биде имплементиран во React Js.

При имплементацијата на овој проект, запазени се сите протоколи и стандардизација при кодирање во .net јазикот.

6.1. UI Development

На слика 49 е прикажан изгледот на апликацијата. Од десната страна се наоѓа координатниот систем на кој ќе бидат прикажани сите можни крајни точки и ќе се исцрта самата роботска рака при дадена точка со помош наалгебарско аналитичкиот метод. Од левата страна се наоѓаат сите влезни параметри, излезни полиња во кои ќе се испишуваат резултатите, копчињата со кои се управува координатниот систем(сцената), копчињата за повикување на одредена функционалност и известувањето кој ја изработил.



Слика 49 – Приказ на веб-апликацијата, која е решение на проблемот на инверзна кинематика на роботска рака

6.1.1 Three.js имплементација

Three.js е JavaScript библиотека и се користи при дизајн и прикажување на 3Д анимации, графикони и слични работи на веб-пребарувач. Основата на Three.js се заснова на WebGL и е open source. Оваа JavaScript библиотека за прв пат била пуштена во употреба во 2000-та година и набрзо добива на популарност и со тоа доаѓа нејзиното усовршување.

WebGL всушност преставува OpenGL јазик за веб-пребарувачите. Скоро сите функционалности и трансформации овозможени со OpenGL се пресликани во WebGL.

При самиот почеток, најпрво мора да се импортира скриптата во самата веб-страна сѐ со цел подоцна да може да се користи во другите скрипти. Тоа е овозможено со:

```
<scriptsrc="~/Scripts/three.min.js"></script>
```

Како и во OpenGL така и овде најпрво мора да се креира и воспостави сцена на која ќе се презентираат и отсликуваат сите трансформации. Целата логика и методи кои ќе служат за управување со сцената се дефинирани во посебен *.js фаил именуван како RobotArmScript.js.

Како што е напоменато претходно, најпрвин ќе се иницијализира/рендерира нова сцена на која ќе се работи. За таа цел креиран е метод InitializeScene(). Имплементацијата на овој метод е преставена во *[Додаток 1].* Овој метод содржи

пет основни елементи за работа и тоа: **дефинирање на сцена, дефинирање на камера, дефинирање на рендер за сцената, дефинирање на светлина и исцртување на координатните оски.**

Се дефинира каде ќе биде сместена сцената, односно во кој елемент. Се задаваат широчината и висината на сцената. Следно се дефинира нова сцена од самата библиотека. За секоја сцена потребна е камера, односно да се дефинира од кој агол ќе биде набљудувана сцената од самиот корисник. Подолу е преставен методот за иницијализација на камерата. Откако ќе се иницијализира камерата, потребно е рендер кој ќе ја рендерира сцената при

зададена промена во неа. Тој рендер е многу важен елемент бидејќи се додека не се рендерира сцената промените на самата сцена нема да бидат видливи.

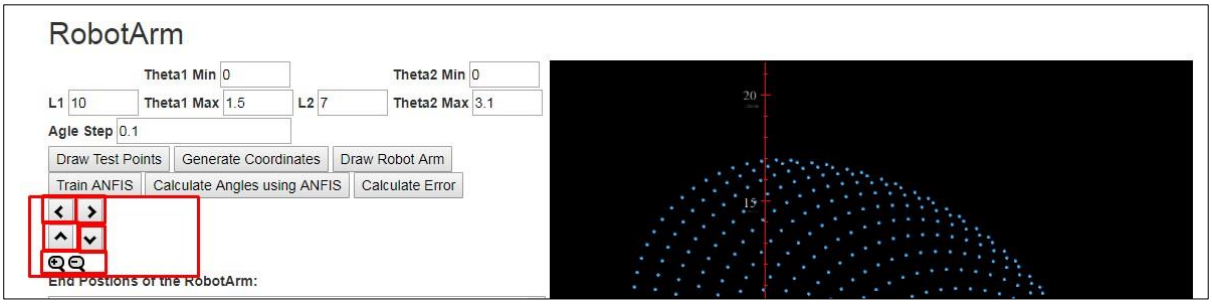
```
renderer.render(scene, camera);
```

Секоја сцена исто така треба да има и светло. Светлото е иницијализирано со помош на методот *initLight();*кој ќе биде прикажан подолуво *[додаток 1].* Бидејќи овде се збори за иницијализација на нова сцена во главниот метод дефиниран е и исцртување на почетната сцена и координатни оски со методот *drawAxes().* Откако горе наведените работи успешно ќе завршат си помош на дефенираниот и иницијализираниот рендер, сцената ќе ја рендерираме.

Се знае дека на сцената ќе бидат прикажувани крајните точки во кои може роботската рака да достигне, односно работната околина на самата роботска рака. Овие точки треба да бидат некако прикажани. Точка како поим точка е нешто толку мало и не може да се прикаже, затоа како решение на ова на сцената секоја точка ќе биде прикажана како топка со центар во дадената точка (x, y, z) и **радиус** од **16 пиксели**. Методот кој ќе се користи низ целиот процес е дефиниран во *[додаток 2]* како *drawPoint(x, y, z, options).*

За симулација, цртање на роботската рака откако ќе се дознае позицијата на вториот зглоб ќе се користи*drawRobotArm(jointPoint, endPoint)* методата изразена во *[додаток 2].*

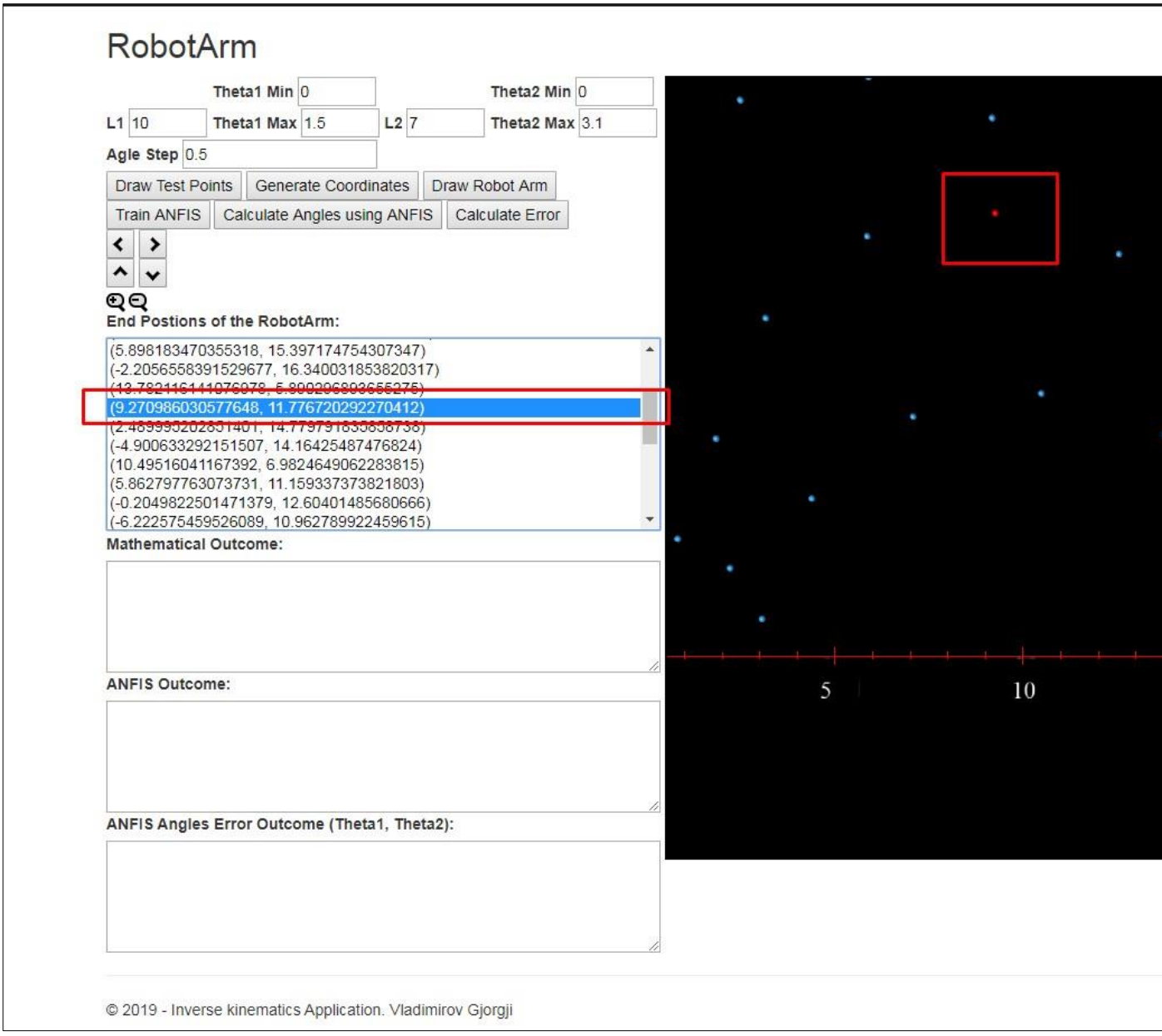
На почетокот е напоменатно дека самата сцена е интерактивна со тоа што ќе може да се поместува зумира и поместува лево и десно камерата.



Слика 50 – Копчиња за движење низ координатниот систем (контролирање на камарата)

Оваа интеракција е овозможена со едноставно поместување на позицијата на камерата. Имплементацијата на методот за интеракција е прикажан *во [додаток 3].*

Втората интеракција овозможена врз сцената/координатниот систем е при селектирање/означување на одредена позиција од листата со конечните точки на сцената се прикажува нејзината положба.



Слика 51 – Интеракција со координатниот систем. Селектирање на крајна позиција.

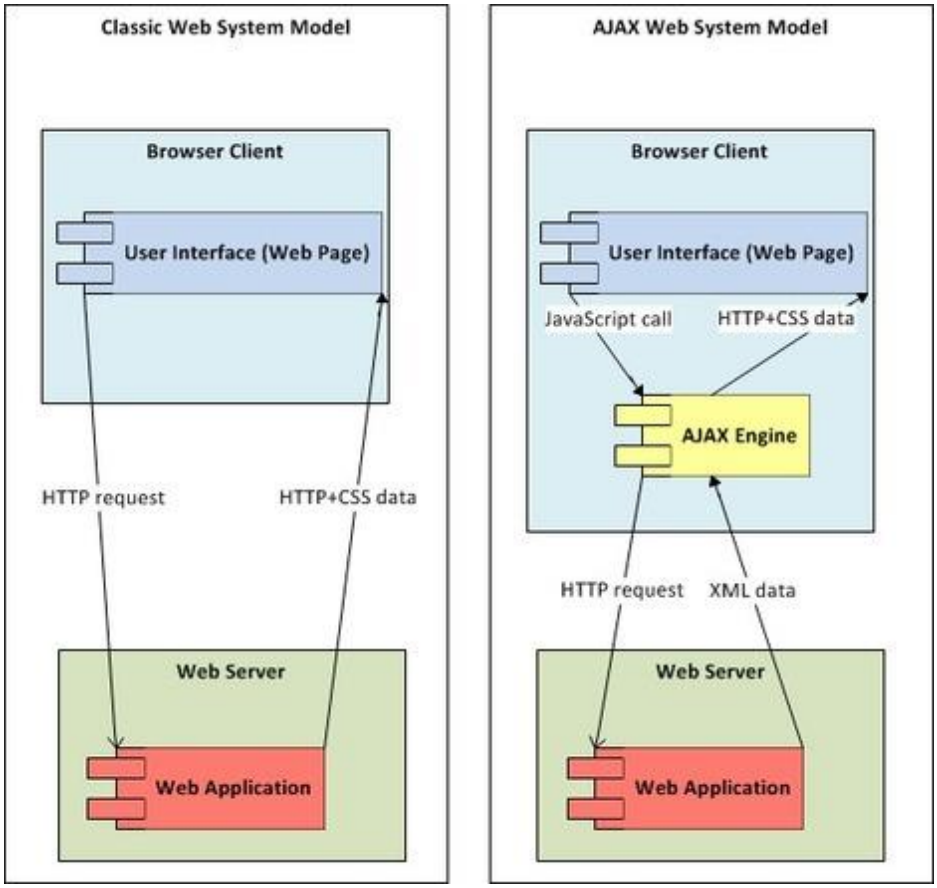
Имплементацијата на овој метод е прикажан во [додаток 4].

За бришење на сите елементи од сцената или за бришење на исцртаната роботска рака дефинирани се два метода кои се слични меѓу себе. Нивната имплементација е прикажана во [додаток 5]. Едниот ги брише сите елементи и повторно ја иницијализира сцената со координатни оски, додека другиот само последните исцртувања на роботската рака.

6.1.2 Поврзување на backend-от со frontend-от

Поврзувањето, односно пресметувањето на самата инверзна кинематика и прикажување на резултатите на сцена со горенаведените методи е овозможена со помош на ***ajax*** повици до дефинираниот **Controller** и резултатот од повикот се пренесува на скриптата RobotArmScript. Пример за еден таков проток ќе го земеме повикот за пресметување на сите можни позиции и нивно прикажување на сцената. Имплементацијата е во [додаток 6].

Разликата помеѓу класичните HTTP повици до серверот и овие со помош на AJAX engine, е во тоа што повиците AJAX се вршат со JavsScript и тие се интерпретираат во позадина без повторно да се повика целата страна. Разликата е прикажана на дијаграмот.



Слика 52 – Класичен HTTPS повик и HTTP повик со AJAX

6.2. Backend имплементација

За имплементација споменато е дека се користи асинхроно програмирање низ паралелни процеси имплементирани во C# програмскиот јазик.

За полесно објаснување, имплементацијата може да се подели на два дела. Првиот дел ја сочинува имплементацијата на аналитичкиот метод и вториот дел за имплементација на ANFIS.

6.2.1 Алгебарско-аналитички метод

Како што е споменато уште во претходното поглавје каде што се анализира аналитичкиот метод, кажано е дека главен фактор е откривањето каде ќе се наоѓа вториот зглоб на роботската рака. Дефиниран е метод *CalculateArmJoint(Point endPoint)* кој на излез враќа објект од тип

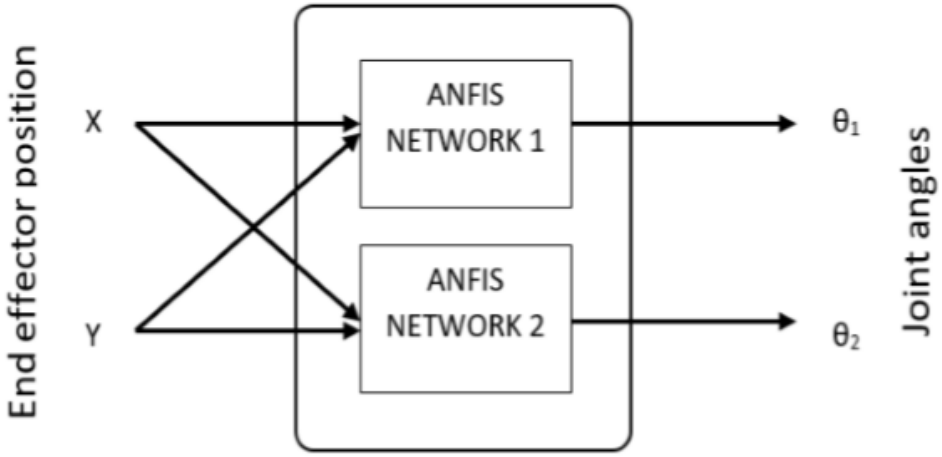
KinematicOutcome кој во себе ги содржи координатите на вториот зглоб и аглите на зглобовите(Θ_1 и Θ_2).

```
publicinterfaceIKinematicOutcome {
        double Theta1 { get; }
        double Theta2 { get; }
        PointJointPosition{ get; }
}
```

Бидејќи се работи со double вредности и при пресметките самиот процесор работи со многу децимали, мора и да се дефинира со која грешка ќе се работи. Односно која ќе биде толеранцијата при пресметките. Овде толеранцијата е земена да биде $\mathbf{10^{-7}}$**.** Целосната имплементација на овој метод е прикажана во [Додаток 7]. Најпрво во методот се одредува сите можни позиции на вториот зглоб и откако ќе се добијат позициите се пристапува кој одредување на двата агли θ_1, θ_2. Како валидација се имплементирани два метода и тоа: едниот за да провери дали аглите се во опсег на минималната и максималната вредност за дадениот агол од можните позиции и вториот метод за да провери инверзно дали должината на деловите на роботската рака се предефинираните вредности.

6.2.2. Имплементација на ANFIS

Во овој случај потребни ќе бидат две ANFIS мрежи. И двете мрежи ќе бидат истренирани со истите влезни податоци, но едната ќе биде за пресметување на θ_1а другата за пресметување на θ_2.



Слика 53 – Архитектура на две ANFIS мрежи за двата агли

Дефинирана е класа ANFIS, која ќе ги содржи правилата и врз тие правила ќе може да се одреди резултатот за дадени влезни параметри. Таа пресметка ќе се врши со методата Inference. Имплементацијата е прикажана во [додаток 8].

Податоци за тренирање на мрежата ќе се определат со помош на следниот код:

```
l1 = 10; % length of first arm
l2 = 7; % length of second arm

theta1 = 0:0.1:pi/2; % all possible theta1 values
theta2 = 0:0.1:pi; % all possible theta2 values

[THETA1,THETA2] = meshgrid(theta1,theta2); % generate a grid of theta1 and
theta2 values

X = l1 * cos(THETA1) + l2 * cos(THETA1 + THETA2); % compute x coordinates
Y = l1 * sin(THETA1) + l2 * sin(THETA1 + THETA2); % compute y coordinates
```

Овде веќе мрежата е истренирана и правилата веќе се иницијализирани и истренирани. За полесна употреба и полесна иницијализација на објект од оваа класа дефиниран е Builder со кој ќе се креира мрежата ANFIS. Builder-от ќе биде статичен и ќе содржи еден метод а тоа е методот Build кој како излез ќе биде истренирана ANFIS мрежа [додаток 8].

Од самата дефиниција на методот се гледа дека како влезни аргументи треба да се проследат: влезните податоци како мултидимензионална низа, излезните податоци исто така како мултидимензионална низа потребни за тренирање на системот, максимален број на интерации за тренирање, објект од тип ITraining со кој се тренира системот/мрежата при секоја интерација и објект од IRuleExtractor кој содржи метод за вадење на правила односно за преместување на Centroids и Consequences вредности што се потребни за тренирање на системот. Овој IRuleExtractor во самиот метод со помош на Factory за генерирање на правила се добива листата од правила, но при користење мора да се наспомене и од кој тип ќе бидат правилата. [додаток 8]

Како што е споменато, за секој агол од роботската рака ќе се креира и истренира мрежа ANFIS и подоцна со секоја мрежа ќе се пресметуваат саканите агли. Во кодот во [додаток 8] секоја мрежа ќе се тренира во посебна нишка со што двете мрежи ќе се тренираат паралелно и независно една од друга. Како резултат ќе се добие поголема брзина на тренирање.

Како што може да се види во оваа имплементација ќе се користи Гаусовата членска функција во правилата. Имплементацијата на Гаусовата членска функција е пресликана во метод од формулата дадена во едно од претходните поглавја. Во [додаток 9] е прикажана целата имплементација на членската функција.

$$\mu_A(x) = exp\left[-\frac{(x-c_i)^2}{2a_i{}^2}\right] \tag{85}$$

6.3. Експеримент и имплементација

Експериментите што ќе бидат имплементирани за евалуација, ќе бидат имплементирани, според следниот IExperimentinterfa, се:

```
publicinterfaceIExperiment {
IEnumerable<Point>ExperimentPositions{ get; }
IEnumerable<IKinematicOutcome>ActualOutputs{ get; }
IEnumerable<IKinematicOutcome>AnfisOutputs{ get; }
  Task<IEnumerable<MathErrorOutcome>>CalculateError();

IEnumerable<Point>GeneratePositions(double radius, double step, doubleshiftX,
doubleshiftY);

voidSetActualOutputs(IEnumerable<IKinematicOutcome>actualOutputs);
voidSetAnfisOutputs(IEnumerable<IKinematicOutcome> outputs);
```

```
}
```

Секој експеримент треба да содржи точки врз кои ќе се тестираат решенијата. Својството *ExperimentPositions* кое е листа од точки. Ова својство треба да се полни со помош на методот *GeneratePositions*. Овој метод како прв аргумент ја има големината на траекторијата. Во овој случај тоа може да биде радиус на кружница, должина на страна од квадрат и сл. Како втор аргумент треба да се наведе растојанието од една до друга точка која треба да се генерира. Трет и четврт аргумент се транслации по x и по y оската, односно за колку мерни единици треба да се помести генерираната траекторија по x, односно по y оската.

Откако ќе се добијат експерименталните точки, истите тие треба да се искористат и да се пресметаат решенијата со алгебарско-аналитичкиот метод и ANFIS. Решенијата со помош на ***Set…*** методите се ставаат во експерименталниот објект и може да се повика ***CalculateError***() методот. Овој метод служи за пресметка на отстапувањето на реалното решение со она добиено со помош на ANFIS. Може да се забележи дека излезот од овој метод е **Task<..>**што значи дека е асинхрон метод и при имплементација може да се имплементира во нова нишка (*Thread*).

Овде може да се воочи дека пресметувањето на отстапувањето и штелувањето на решенијата за сите експерименти е ист па затоа може да се имплементира како апстрактна класа што сите експерименти ќе ја наследуваат. Ќе се дефинира апстрактна класа Experiment, прикажана е во [додаток 10].

6.3.1 Кружна траекторија – имплементација

Имплементацијата на овој експеримент се заснова на имплементирање на методот за генерирање на експериментални точки. Ќе се дефинира класа CircleExperiment која се наследува од апстрактната класа Experiment.

```
publicclassCircleExperiment : Experiment {

publicoverrideIEnumerable<Point>GeneratePositions
   (double radius, double step = 0.1745, doubleshiftX = 0, doubleshiftY = 0) {
double counter = 0.0;
```

```
var result = new List<Point>();
while(counter <Math.PI * 2) {
var x = (radius * Math.Cos(counter)) + shiftX;
var y = (radius * Math.Sin(counter)) + shiftY;
result.Add(new Point() { X = x, Y = y });
      counter += step;
    }
ExperimentPositions = result;
returnresult;
  }
}
```

Растојанието помеѓу две точки доколку не е проследено ќе изнесува *0.1745* мерни единици.

6.3.2 Квадратна траекторија имплементација

Исто како и за кружното движење и овде единствен метод што треба да се имплементира е методот за генерирање на експериментални точки. Разликата овде е тоа што нема да се користи радиус туку должина на страната на квадратот.

```
publicclassSquareExperiment : Experiment {

publicoverrideIEnumerable<Point>GeneratePositions
   (double side, double step = 0.1745, doubleshiftX = 0, doubleshiftY = 0) {
var result = new List<Point>();
for(doublei = -side; i<= side; i+=step) {
var side1Point = new Point { X = i + shiftX, Y = side + shiftY };
var side2Point = new Point { X = i + shiftX, Y = -side + shiftY };
var side3Point = new Point { X = side + shiftX, Y = i + shiftY };
var side4Point = new Point { X = -side + shiftX, Y = i + shiftY };

result.AddRange(new[] { side1Point, side2Point, side3Point, side4Point });
    }

ExperimentPositions = result;
returnresult;
  }
}
```

6.3.3 Синусоидна траекторија - имплементација

Како и во претходните експерименти и овде методот за генерирање на

експериментални точки треба да биде имплементиран. Овде првиот параметар ќе биде должината на синусоидата.

```
publicclassSinExperiment : Experiment {

publicoverrideIEnumerable<Point>GeneratePositions
   (double length, double step = 0.1745, doubleshiftX = 0, doubleshiftY = 0) {
var result = new List<Point>();

for(doublei = 0; i<= length; i+= step) {
result.Add(new Point {
        X = i+shiftX,
        Y = Math.Sin(i)+shiftY
      });
    }

ExperimentPositions = result;
returnresult;
  }
}
```

Откако ќе се добијат точките и отстапувањето тоа треба да се проследи назад кон корисникот и неговиот UI, со помош на акција регистрирана во самиот контролер. Кога резултатот ќе се добие на UI делот на ист начин како што е прикажана работната околина на роботската рака така и овде со истиот метод се прикажуваат експерименталните точки. Имплементацијата на овие методи е прикажана во [додаток 11].

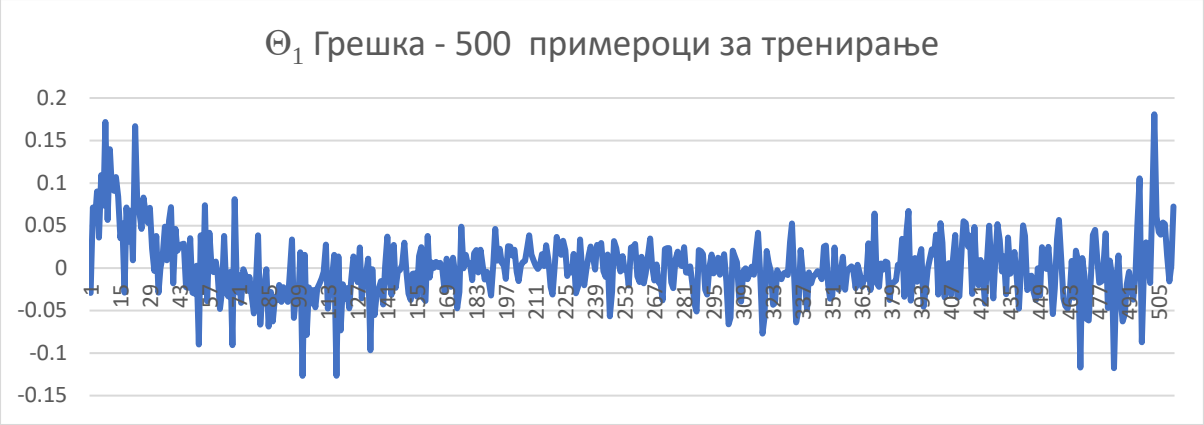
7. Евалуација на имплементираното решение - експериментални резултати

За да се провери колку е точна имплементацијата и колку точно ги дава самата ANFIS мрежа ќе треба да се направат одредени тестирања. Во конкретниот случај ќе се направи тестирање, односно отстапувањето на ANFIS резултатот со алгебарско аналитичкиот. Споредбата се врши на тој начин што најпрвин ќе се одберат посакувани точки кои ќе преставуваат тест точки и веднаш потоа ќе се пресметаат резултатите(потребните аглиθ_1и θ_2) за тие точки во двата метода оделно и ќе стојат во меморија. За крај ќе се одземат резултатите едни од други и со тоа се добива колку отстапува даден агол од реалниот (потребниот). Ќе се претпостави дека аналитичкиот метод е посакуваниот агол. Со други зборови кажано за дадена точка откако ќе се пресметаат двете решенија, аголот $\theta_{1\mathrm{M}}$ ќе се одземе од $\theta_{1\mathrm{A}}$ исто така и $\theta_{2\mathrm{M}}$ ќе се одземе од $\theta_{2\mathrm{A}}$. Псевдо кодот за овој потег е:

$$dataSet = CalculateDataSet(\dots)$$
$$TrainAnfis(dataSet)$$
$$O_M = CalculateOutcome(dataSet) - \text{Алгебарско аналитичко}$$
$$O_A = CalculateOutcomeUsingANFIS(dataSet) - ANFIS$$
$$Error = O_M - O_A$$

За тестирање ќе се разгледаат две сценарија и тоа во првото сценарио при тренирање на роботската рака ќе земеме дека раката се движи со 0.1, односно за тренирање ќе бидат земени приближно 500 примероци. Додека во вториот пример, ќе штелуваме раката да се движи со 0.05 и со тоа ќе се зголемат примероците од 500 на приближно 2 000 примероци. И во двата примера ќе се земат ист број на правила при тренирање на мрежите, по 25 правила. Со овие точки ќе биде и претходно истренирана мрежата ANFIS.

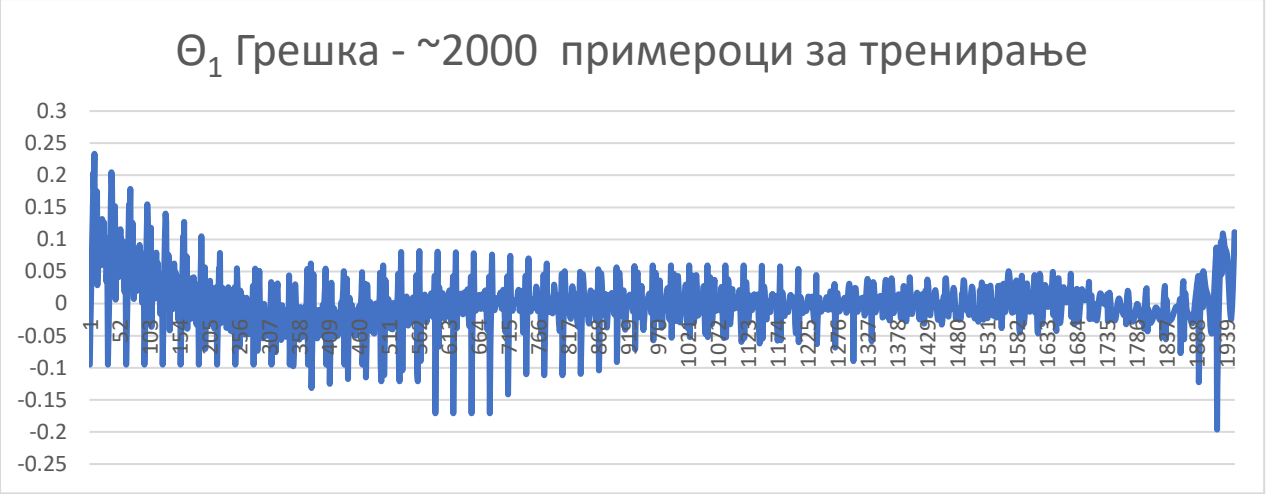
Резултатот (отстапувањето на аголот) за првиот пример се гледа на сликите подолу за θ_1и θ_2 одделно.



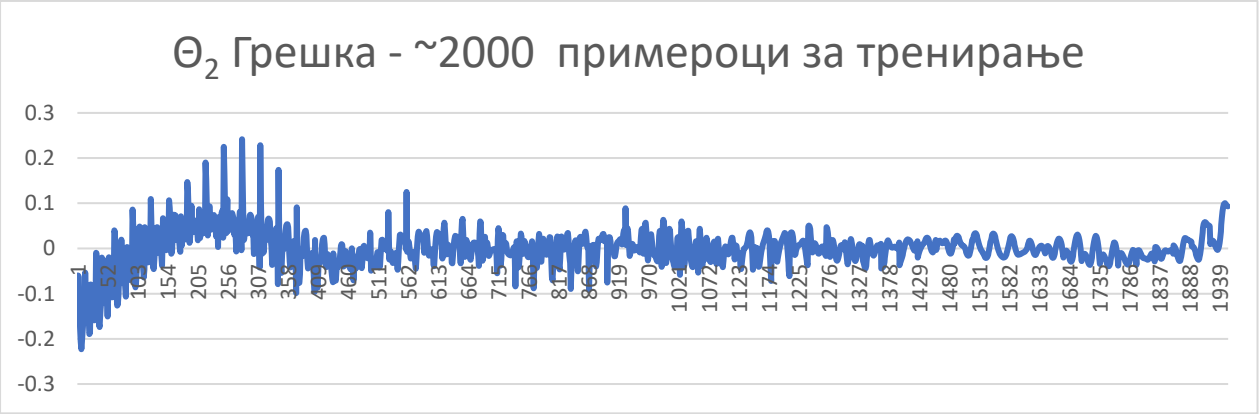
Слика 54 – Графички приказ на грешката на аголот Θ_1 со 500 примероци

Слика 55 - Графички приказ на грешката на аголот Θ_2 со 500 примероци

Доколку примероците за тренирање ги зголемиме на приближно 2000 ќе се забележи веднаш друга вредност за отстапувањето на аглите θ_1 и θ_2 односно:



Слика 56 - Графички приказ на грешката на аголот Θ_1 со 2000 примероци



Слика 57 - Графички приказ на грешката на аголот Θ_2 со 2000 примероци

Од самите графикони ќе се заклучи дека со зголемување на примероците за тренирање на самата ANFIS мрежа се зголемува и точноста, односно се намалува грешката на пресметување. Ова подобрување повеќе се забележува кај аголот θ_2.

7.1. Експерименти по дадена траекторија

За подобра анализа на точноста и грешката ќе се зададе дадена траекторија од точки и за секоја точка ќе се пресмета инверзната кинематика со помош на претходно истренираниот систем ANFIS и истиот ќе се спореди со резултатот од аналитичкиот метод. Ќе бидат зададени три различни траектории. Сите траектории ќе бидат дел од работниот простор на роботската

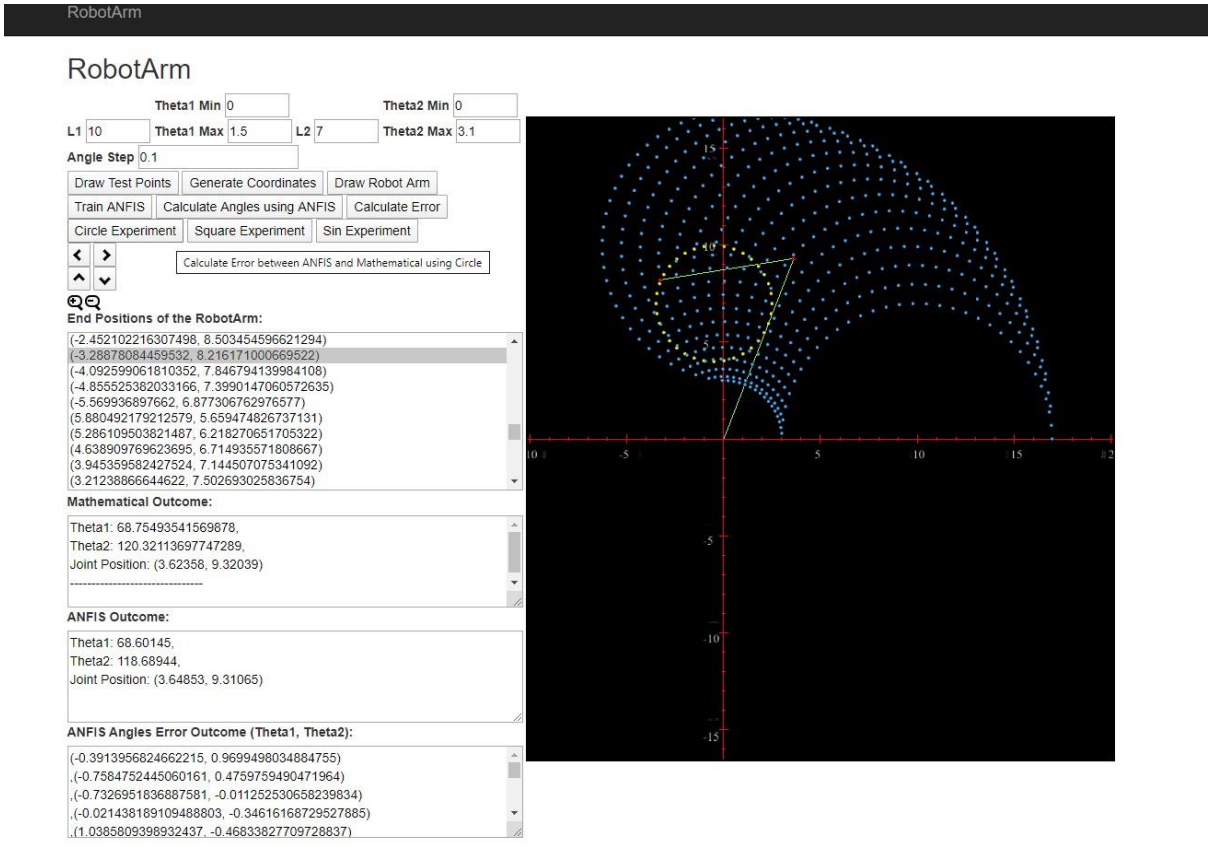
рака. Траекториите ќе бидат од познат облик и тоа: кружно, квадратно и синусоидно движење.

7.1.1 Кружна траекторија

Кружното движење е дефинирано со радиус на кружницата и степен на зголемување на точките.

За овој тип на експеримент ќе се земе дека радиусот на кружницата е 3 мерни единици, аголот помеѓу две точки е *0.1745 радијани*. Целата оваа кружницата ќе биде транслирана по x и по y оската и тоа по x оската за -0.5 мерни единици, додека по y оската за 7 мерни единици.

Пример е прикажан на слика 58.



Слика 58 – Приказ на кружната траекторија во координатниот систем

Резултатите добиени за грешката за првиот и вториот агол се прикажани на следните графикони.

Слика 59- Графички приказ на грешката на аголот Θ_1, кај кружниот експеримент изразена во радијани

Слика 60- Графички приказ на грешката на аголот Θ_2, кај кружниот експеримент изразена во радијани

Грешката на овие резултати се изразени во радијани. Човековото работење со агли, поголема визуелна перцепција има доколку аглите се прикажани во степени, па затоа доколку се претворат во степени се добива истата крива само со различни броеви на у оската, односно:

Слика 61 - Графички приказ на грешката на аголот Θ_1, кај кружниот експеримент изразен во степени

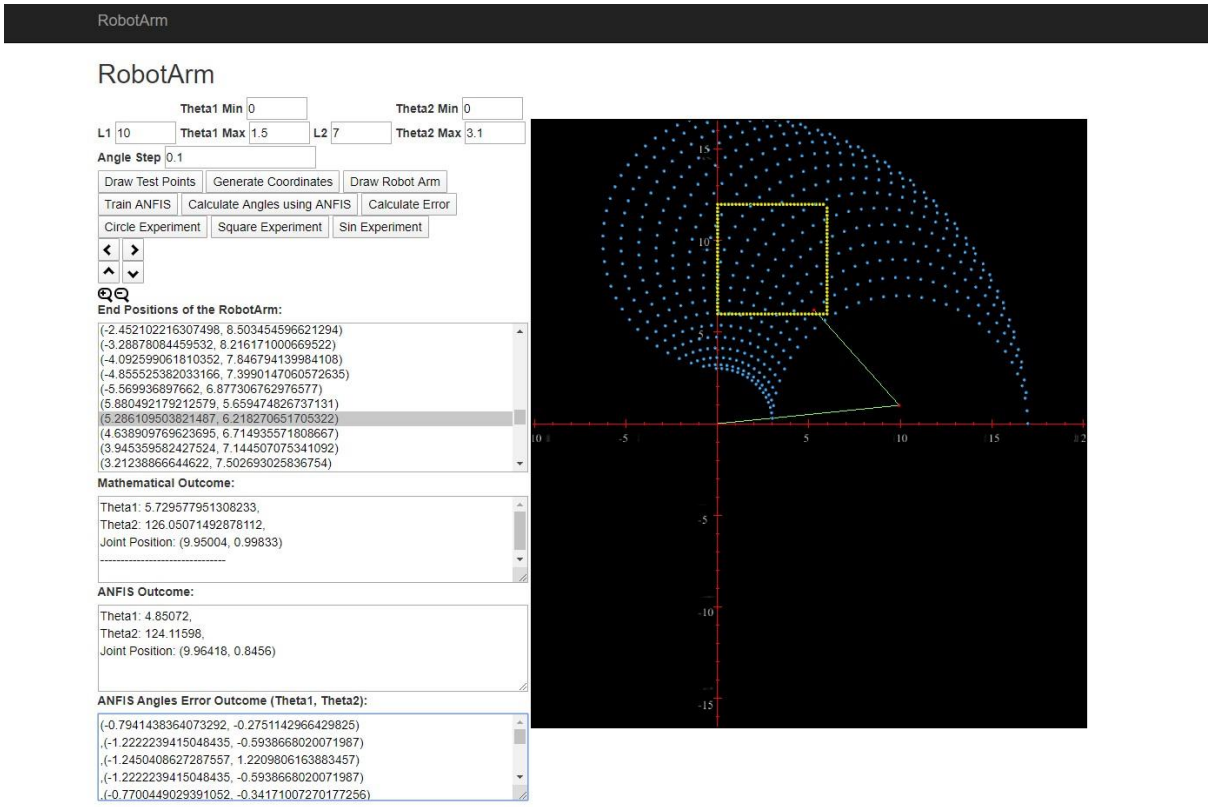
Слика 62 - Графички приказ на грешката на аголот Θ_2, кај кружниот експеримент изразен во степени

Од овие графикони се забележува дека за првиот агол отстапувањето за првиот агол е во опсег помеѓу **-1.5** и **3.7 степени** низ целата кружница која е составена од 37 точки. Додека за вториот ако е во опсег помеѓу **-1.14 и 1.6 степени** низ целата кружница која е составена од 37 точки.

7.1.2 Квадратна траекторија

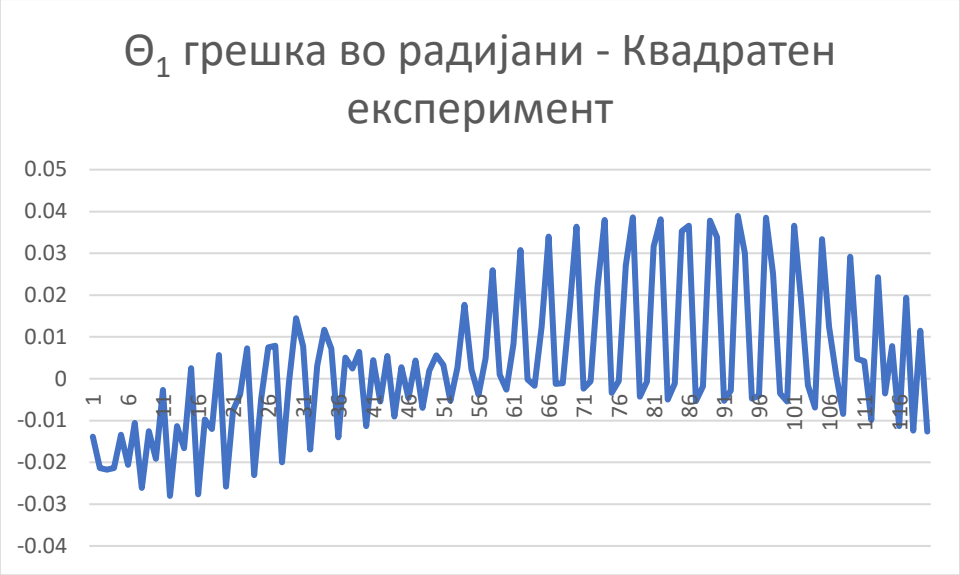
Квадратното движење е дефинирано со должина на страната на квадратот, растојанието помеѓу две точки и доколку е потребна транслација по x и по y оската.

За конкретната роботска рака, земено е страната да биде со големина од 3 мерни единици, а растојанието помеѓу две точки да биде 0.2 мерни единици. Квадратот во конкретниот случај ќе го сочинуваат 120 точки. Истиот квадрат ќе биде поместен за 3 мерни единици на x оската и 9 мерни единици на y оската. Експериментот е преставен на слика 63.

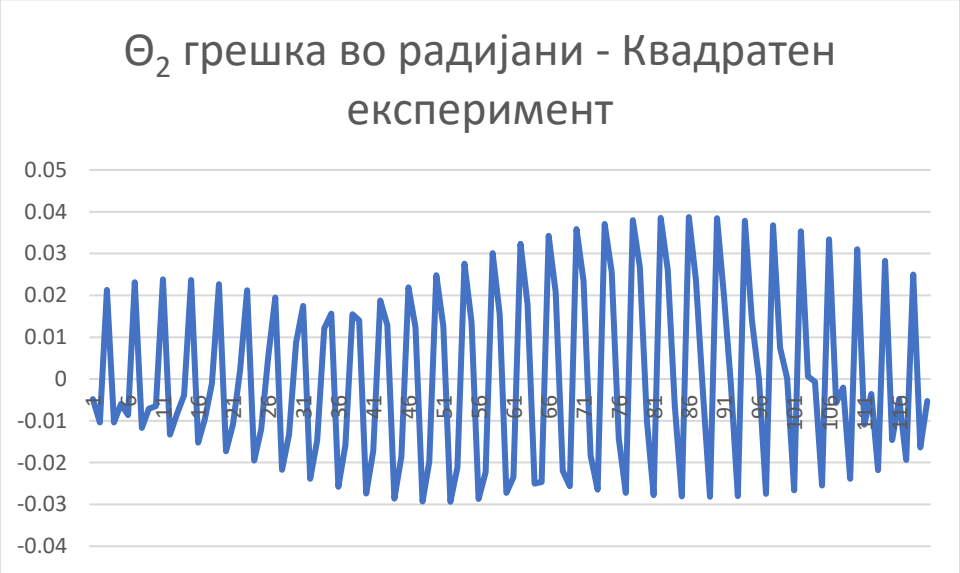


Слика 63 – Приказ на квадратната траекторија во квадратниот систем

Резултатите добиени за грешката за првиот и вториот агол се прикажани на следните графикони.

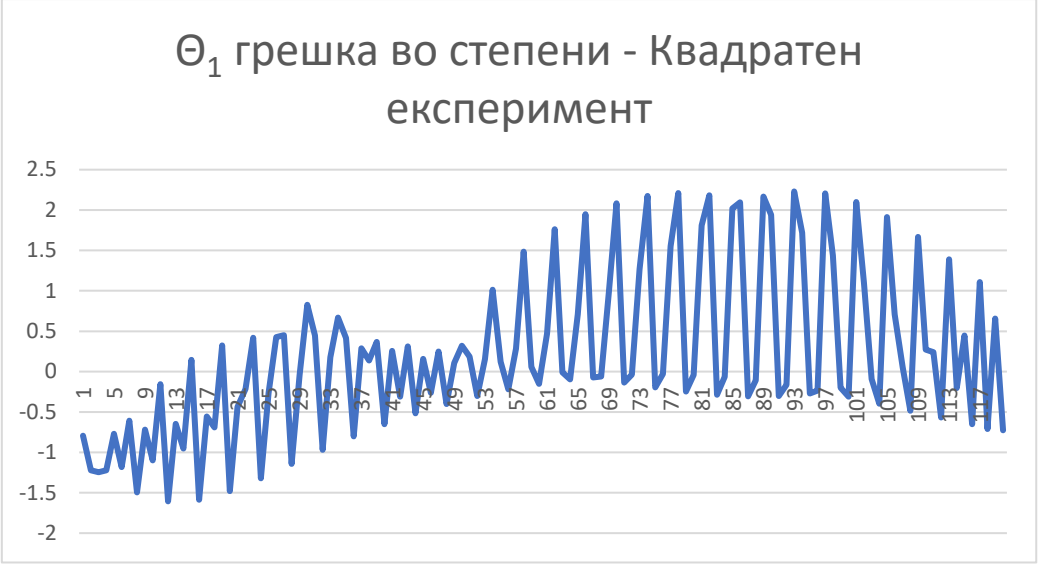


Слика 64 - Графички приказ на грешката на аголот Θ_1, кај квадратниот експеримент изразен во радијани

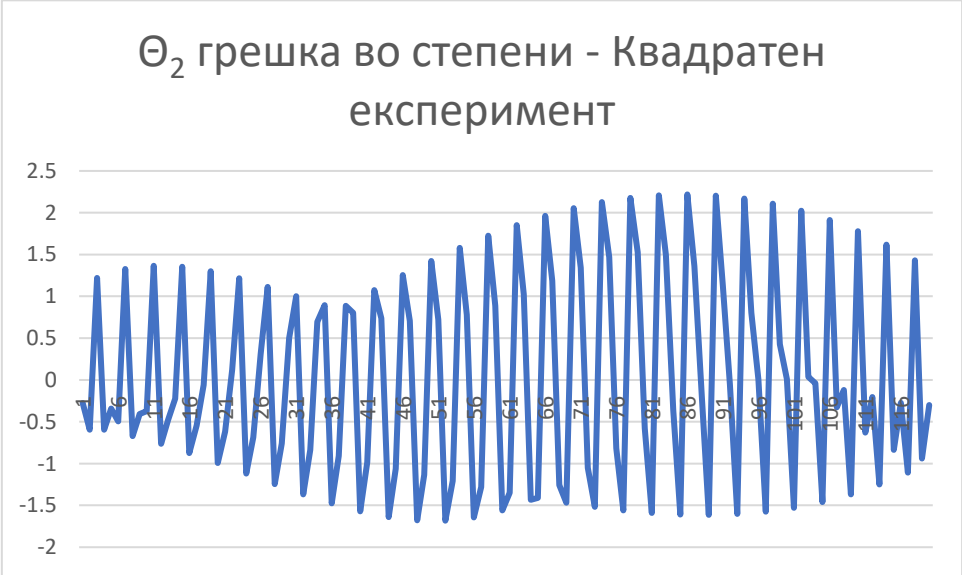


Слика 65 - Графички приказ на грешката на аголот Θ_2, кај квадратниот експеримент изразен во радијани

Грешката на овие резултати се изразени во радијани. Човековото работење со агли, поголема визуелна перцепција има доколку аглите се прикажани во степени, па затоа доколку се претворат во степени се добива истата крива, само со различни броеви на у оската, односно:



Слика 66 - Графички приказ на грешката на аголот Θ_1, кај квадратниот експеримент изразен во степени



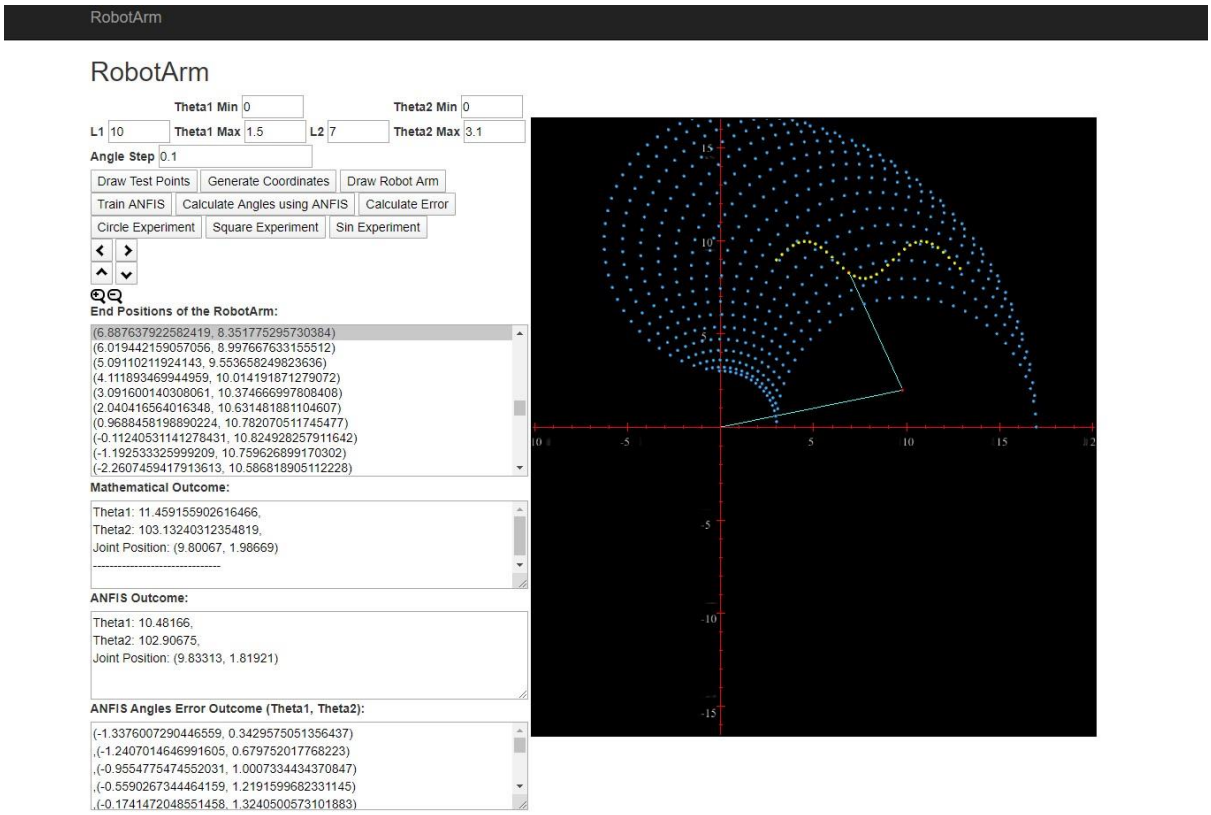
Слика 67 - Графички приказ на грешката на аголот Θ_2, кај квадратниот експеримент изразен во степени

За овој експеримент грешката за првиот агол е во опсег помеѓу **-1.6 и 2.2 степени**, додека за вториот агол отстапувањето е исто така во опсег помеѓу **-1.6 и 2.2 степени**.

7.1.3 Синусоидна траекторија

Синусуидното движење е дефинирано со должина на синусоидната траекторија, растојанието помеѓу две точки и поместување на почетокот по x и по y оската.

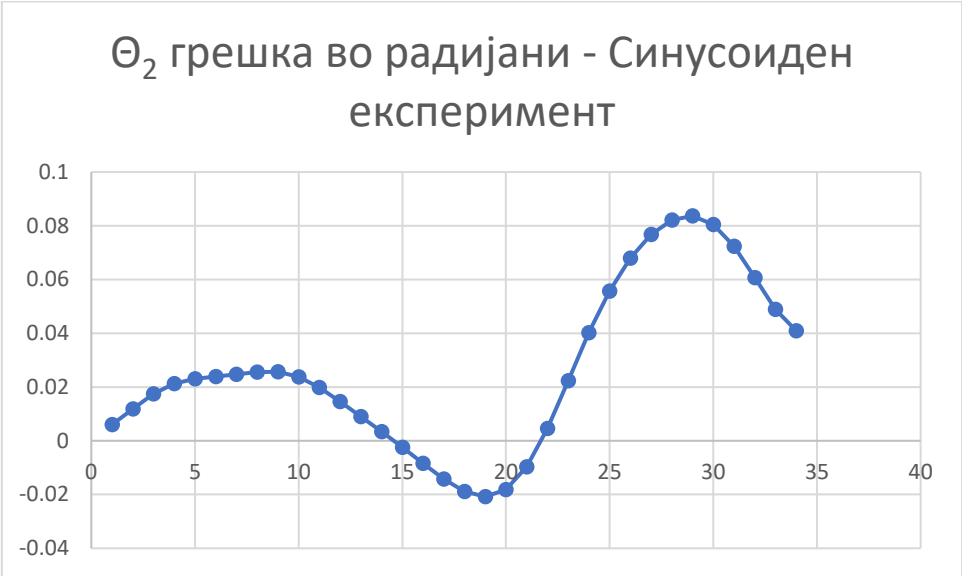
За конкретната роботска рака ќе се земе дека должината ќе биде 10 мерни единици, растојанието помеѓу две точки ќе биде 0.3 мерни единици, а стартот на движењето ќе биде во точката M(3, 9). Сликовит приказ на целата траекторија во работната околина е прикажана на слика 68.



Слика 68 – Приказ на синусоидната траекторија во координатниот систем

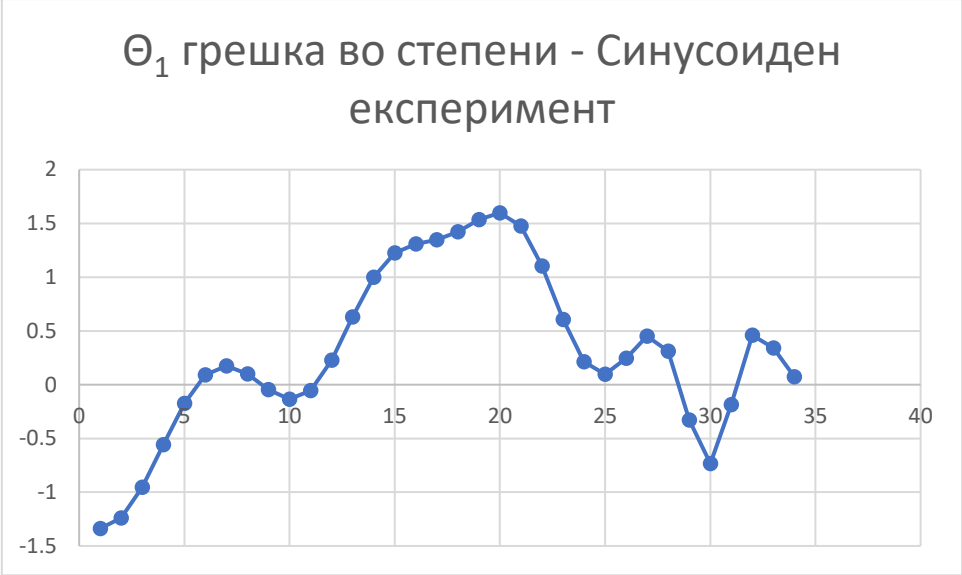
Резултатите добиени за отстапувањето на аглите за овој експеримент се прикажани на следните графикони.

Слика 69 - Графички приказ на грешката на аголот Θ_1, кај синусоидниот експеримент изразен во радијани

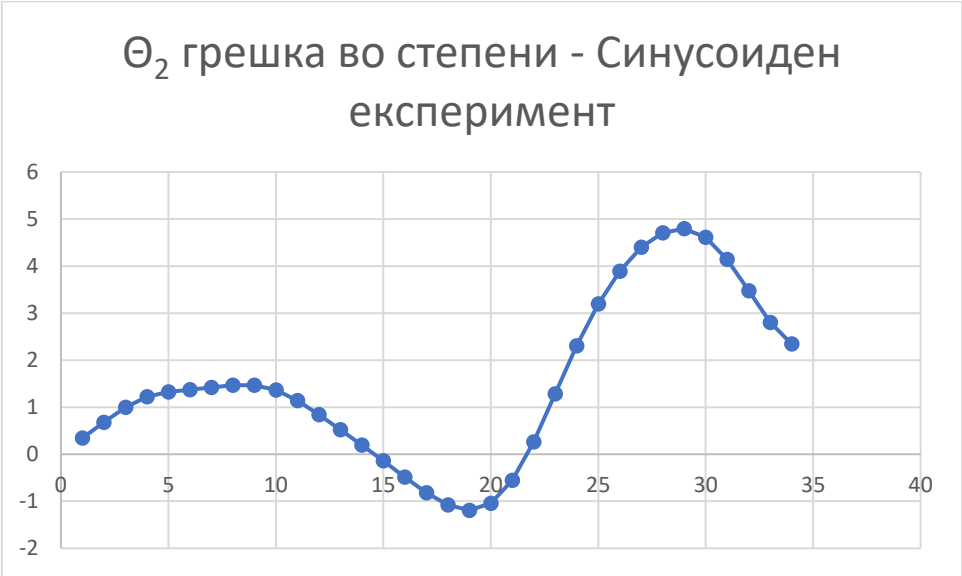


Слика 70 - Графички приказ на грешката на аголот Θ_2, кај синусоидниот експеримент изразен во радијани

Овие резултати се прикажани во радијани. За полесна анализа исто како и во претходните експерименти ќе се претворат во степени.



Слика 71 - Графички приказ на грешката на аголот Θ_1, кај синусоидниот експеримент изразен во степени



Слика 72 - Графички приказ на грешката на аголот Θ_2, кај синусоидниот експеримент изразен во степени

Од графиконите може да се забележи дека за првиот агол отстапувањето е во опсег помеѓу **-1.3 и 1.6 степени**, додека за вториот агол отстапувањето е во опсег помеѓу **-1.19 и 4.8 степени**. Во оваа траекторија може да се каже дека е најсложена и аголот најмногу може да отспани и да доведе до некоја грешка.

8. Заклучок

Роботските раце обично се имплементираат за земање, преместување и слични операции врз објекти од различни големини. Решавањето на проблемот на директната кинематика е едноставно и нема некоја поголема комплексност. Единствено треба да се пресметаат сите ротации и транслации и ќе се добие крајната положба. Решавањето на проблемот на инверзна кинематика може да изискува многу и може да биде многу комплексен.

Во овој магистерски труд беше презентирано решение на проблемот на инверзна кинематика на роботска рака која работи во дадена рамнина и составена од два дела(два зглоба). Преставени беа две решенија. Првото решение се базира на **алгебарско-аналитички метод**. Односно, алгебарско решение со елементи со аналитичка геометрија. Додека второто решение е базирано на **ANFIS (Adaptive Neuro Fuzzy Iterface System).**За второто решение преставено е алгоритам за пресметување на работната околина во рамнината. Второто решение, односно решението базирано на ANFIS е споредено со алгебарско аналитичкото решение кое е земено како валидно решение.

Извршени се три типа на експерименти и резултатите се прикажани графички и се образложени. Грешката просечно се движи помеѓу **-0.03 и 0.08 радијани.** Изразено во степени грешката за аглите просечно се движи помеѓу **-1.6 и 2.2 степени**. Со оваа грешка може да се каже дека решението е прифатливо за поголем дел од апликациите каде што треба и може да се имплементира.

Доколку е потребно подобрување за да се добие уште поголема точност, тогаш може да се разгледа подобрување кај алгоритамите за учење и бројот на интерации за учење како и бројот на правила заедно со бројот на податоци за тренирање на мрежата.

Користена литература

[1] Loshkovska, Suzana, and SasoKoceski, eds. ICT innovations 2015: Emerging technologies for better living. Vol. 399. Springer, 2015.

[2] Koceski, Saso, and Biljana Petrevska. "Empirical evidence of contribution to e-tourism by application of personalized tourism recommendation system." Annals of the AlexandruIoanCuza University-Economics 59, no. 1 (2012): 363-374.

[3] Trajkovik, Vladimir, Elena Vlahu-Gjorgievska, SasoKoceski, and Igor Kulev. "General assisted living system architecture model." In International Conference on Mobile Networks and Management, pp. 329-343. Springer, Cham, 2014.

[4] Stojanov, Done, and SasoKoceski. "Topological MRI prostate segmentation method." In Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on, pp. 219-225. IEEE, 2014

[5] Koceski, Saso, and NatasaKoceska. "Evaluation of an assistive telepresence robot for elderly healthcare." Journal of medical systems 40, no. 5 (2016): 121.

[6] Stojanov, Done, Aleksandra Mileva, and SašoKoceski. "A new, space-efficient local pairwise alignment methodology." Advanced Studies in Biology 4, no. 2 (2012): 85-93.

[7] Koceski, Saso, and NatasaKoceska. "Challenges of videoconferencing distance education-a student perspective." International Journal of Information, Business and Management 5, no. 2 (2013): 274.

[8] Koceski, Saso, NatasaKoceska, and Ivica Kocev. "Design and evaluation of cell phone pointing interface for robot control." International Journal of Advanced Robotic Systems 9, no. 4 (2012): 135.

[9] Koceski, Saso, StojanchePanov, NatasaKoceska, PierluigiBeomonte Zobel, and Francesco Durante. "A novel quad harmony search algorithm for grid-based path finding." International Journal of Advanced Robotic Systems 11, no. 9 (2014): 144.

[10] Koceska, Natasa, SasoKoceski, Francesco Durante, PierluigiBeomonte Zobel, and TerenzianoRaparelli. "Control architecture of a 10 DOF lower limbs exoskeleton for gait rehabilitation." International Journal of Advanced Robotic Systems 10, no. 1 (2013): 68

[11] Serafimov, Kire, DimitrijaAngelkov, NatasaKoceska, and SasoKoceski. "Using mobile-phone accelerometer

for gestural control of soccer robots." In Embedded Computing (MECO), 2012 Mediterranean Conference on, Bar, Montenegro, pp. 140-143. 2012.

[12] Koceska, Natasa, and SasoKoceski. "FinancialEconomic Time Series Modeling and Prediction Techniques–Review." Journal of Applied Economics and Business 2, no. 4 (2014): 28-33.

[13] Kucuk, Serdar, and Zafer Bingul. "Robot kinematics: Forward and inverse kinematics." In Industrial Robotics: Theory, Modelling and Control. IntechOpen, 2006.

[14] Pérez-Rodríguez, Rodrigo, Alexis Marcano-Cedeño, Úrsula Costa, Javier Solana, César Cáceres, Eloy Opisso, Josep M. Tormos, Josep Medina, and Enrique J. Gómez. "Inverse kinematics of a 6 DoF human upper limb using ANFIS and ANN for anticipatory actuation in ADLbased physical Neurorehabilitation." Expert Systems with Applications 39, no. 10 (2012): 9612-9622.

[15] J. J. Craig, Introduction to Robotics: Mechanisms and Controls, Addison-Wesley, Reading, MA, 1989.

[16] G. C. S. Lee, Robot Arm Kinematics, Dynamics and Control, Computer, Vol. 15, Issue. 12, pp. 62-79, 1982.

[17] J. U. Korein, N. I. Balder, Techniques for generating the goal-directed motion of articulated structures, IEEE Computer Graphics and Applications, Vol. 2, Issue. 9, pp. 71-81, 1982.

[18] KöKer, RaşIt. "A genetic algorithm approach to a neuralnetwork-based inverse kinematics solution of robotic manipulators based on error minimization." Information Sciences 222 (2013): 528-543.

[19] Momani, Shaher, Zaer S. Abo-Hammour, and Othman MK Alsmadi. "Solution of inverse kinematics problem using genetic algorithms." Applied Mathematics & Information Sciences 10, no. 1 (2016): 225.

[20] Ram, R. V., P. M. Pathak, and S. J. Junco. "Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling." Mechanism and Machine Theory 131 (2019): 385-405.

[21] Dereli, Serkan, and RaşitKöker. "Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of the results in terms of speed and accuracy." Inverse Problems in Science and Engineering (2019): 113.

[22] Almusawi, Ahmed RJ, L. CananDülger, and SadettinKapucu. "A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242)." Computational intelligence and neuroscience 2016 (2016).

[23] Hošovský, A., J. Piteľ, K. Židek, M. Tóthová, J. Sárosi, and L. Cveticanin. "Dynamic characterization and simulation of two-link soft robot arm with pneumatic muscles." Mechanism and Machine Theory 103 (2016): 98-116.

[24] Raheem, Firas A., Hind Z. Khaleel, and Mostafa K. Kashan. "Robot Arm Design for Children Writing Ability Enhancement using Cartesian Equations based on ANFIS." In 2018 Third Scientific Conference of Electrical Engineering (SCEE), pp. 150-155. IEEE, 2019.

[25] Sahoo, Mrunmayee& Patra, Kanhu&Khatua, K.. (2015). Inference of Water Quality Index Using ANFIA and PCA. Aquatic Procedia. 4. 1099-1106. 10.1016/j.aqpro.2015.02.139.

[26] V. Sazdovski, P. M.G. Silson, A. Tsourdos “Attitude Determination from Single Camera Vector Observations”, IEEE Conference on Intelligent Systems, London, UK, 2010.

[27] Vladimirov, Angel and Koceski, Saso (2019) Attitude Determination of Unmanned Aerial Vehicle using Single Camera Vector Observations. International Journal of Computer Applications, 178 (41). pp. 15-21. ISSN 0975 – 8887

[28] Panov, Stojanche, and SasoKoceski. "Area coverage in wireless sensor network by using harmony search algorithm." In 2014 3rd Mediterranean Conference on Embedded Computing (MECO), pp. 210-213. IEEE, 2014.

[29] Koceska, Natasa, SasoKoceski, Francesco Durante, PierluigiBeomonte Zobel, and TerenzianoRaparelli. "Control architecture of a 10 DOF lower limbs exoskeleton for gait rehabilitation." International Journal of Advanced Robotic Systems 10, no. 1 (2013): 68.

[30] Moran, Michael. (2007). Evolution of robotic arms. Journal of Robotic Surgery. 1. 103-111. 10.1007/s11701-006-0002-x.

[31] Koceski, Saso, and NatasaKoceska. "Evaluation of an assistive telepresence robot for elderly healthcare." Journal of medical systems 40, no. 5 (2016): 121

[32] Prusak, A., O. Melnychuk, H. Roth, Ingo Schiller, and Reinhard Koch. "Pose estimation and map building with a time-of-flight-camera for robot navigation." International Journal of Intelligent Systems Technologies and Applications 5, no. 3/4 (2008): 355-364

[33] Benini, Alessandro, Matthew J. Rutherford, and Kimon P. Valavanis. "Experimental evaluation of a real-time GPU-based pose estimation system for autonomous landing of rotary wings UAVs." Control Theory and Technology 16, no. 2 (2018): 145-159.

[34] Schmitz, Gabriel, Tiago Alves, Renato Henriques, Edison Freitas, and Ebrahim El'Youssef. "A simplified approach to motion estimation in a UAV using two filters." IFAC-PapersOnLine 49, no. 30 (2016): 325-330.

[35] Koceski, Saso, NatasaKoceska, PierluigiBeomonte Zobel, and Francesco Durante. "Characterization and modeling of a 3D scanner for mobile robot navigation." In 2009 17th Mediterranean Conference on Control and Automation, pp. 79-84. Ieee, 2009

[36] Petrescu, Relly Victoria, Raffaella Aversa, Bilal Akash, Ronald Bucinell, Juan Corchado, Antonio Apicella, and Florian Ion Petrescu. "Inverse kinematics at the anthropomorphic robots, by a trigonometric method." American Journal of Engineering and Applied Sciences 10, no. 2 (2017): 394-411.

[37] Tolani, Deepak, Ambarish Goswami, and Norman I. Badler. "Real-time inverse kinematics techniques for anthropomorphic limbs." Graphical models 62, no. 5 (2000): 353-388.

[38] Wang, L-CT, and Chih-Cheng Chen. "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators." IEEE Transactions on Robotics and Automation 7, no. 4 (1991): 489-499.

[39] Siciliano, Bruno. "The Tricept robot: Inverse kinematics, manipulability analysis and closed-loop direct kinematics algorithm." Robotica 17, no. 4 (1999): 437-445.

[40] Aristidou, Andreas, Joan Lasenby, YiorgosChrysanthou, and Ariel Shamir. "Inverse kinematics techniques in computer graphics: A survey." In Computer Graphics Forum, vol. 37, no. 6, pp. 35-58. 2018.

[41] Petrescu, Florian Ion, and Relly Victoria Petrescu. "Direct and inverse kinematics to the anthropomorphic robots." (2016): 109-124.

[42] Kim, YoungBeom, Byung-Ha Park, Kwang-Mo Jung, and JungHyun Han. "Data-driven Shoulder Inverse Kinematics." arXiv preprint arXiv:1612.07353 (2016).

[43] Karaboga, Dervis, and Ebubekir Kaya. "Adaptive network based fuzzy inference system (ANFIS) training approaches: a comprehensive survey." Artificial Intelligence Review 52, no. 4 (2019): 2263-2293.

[44] Barak, Sasan, and S. SaeedehSadegh. "Forecasting energy consumption using ensemble ARIMA–ANFIS hybrid algorithm." International Journal of Electrical Power & Energy Systems 82 (2016): 92-104.

[45] Tien Bui, Dieu, KhabatKhosravi, Shaojun Li, HimanShahabi, Mahdi Panahi, Vijay P. Singh, Kamran Chapi et al. "New hybrids of anfis with several optimization algorithms for flood susceptibility modeling." Water 10, no. 9 (2018): 1210.

[46] AbdollahiKhosroshahi, Hossein, and MohammadaliBadamchizadeh. "Design and Implementation of a new mechanism for a planar robotic arm." Modares Mechanical Engineering 18, no. 9 (2019): 58-68.

[47] Chen, Wenyu, Jia Du, Wei Xiong, Yue Wang, Shueching Chia, Bingbing Liu, Jierong Cheng, and Ying Gu. "A noise-tolerant algorithm for robot-sensor calibration using a planar disk of arbitrary 3-D orientation." IEEE Transactions on Automation Science and Engineering 15, no. 1 (2016): 251-263.

[48] Cesconeto, Emanuel Moutinho, and Eduardo André Perondi. "Development of a hydraulic robotic arm–Determination of the kinematic parameters." In International

Conference on Innovation, Engineering and Entrepreneurship, pp. 537-544. Springer, Cham, 2018.

[49] Albahari, Joseph, and Ben Albahari. C# 7.0 in a nutshell: the definitive reference. " O'Reilly Media, Inc.", 2017.

[50] Leijen, Daan. "Structured asynchrony with algebraic effects." In Proceedings of the 2nd ACM SIGPLAN International Workshop on Type-Driven Development, pp. 16-29. 2017.

[51] Yuan, Shuguang, HC Stephen Chan, and Zhenquan Hu. "Implementing WebGL and HTML5 in macromolecular visualization and modern computer-aided drug design." Trends in biotechnology 35, no. 6 (2017): 559-571.

[52] Chang, Xin, KivancYuksel, and Władysław Skarbek. "WebGL and web audio software lightweight components for multimedia education." In Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2017, vol. 10445, p. 104452H. International Society for Optics and Photonics, 2017.

[53]Freeman, Adam. "Pro asp. net mvc 5 platform." In *Pro ASP. NET MVC 5 Platform*, pp. 3-8. Apress, Berkeley, CA, 2014.

[54] Galloway, Jon, Phil Haack, Brad Wilson, and K. Scott Allen. *Professional ASP. NET MVC 4*. John Wiley & Sons, 2012.

Додаток 1

Дефинирање на сцената во WebGL каде што ќе се врши пресликувањето, на роботската рака, како и дефинирањето на основните елементи на креираната сцена:

```
functioninitializeScene() {
var scene3d = document.getElementById("robotArmGraphic");

varwidthR = $("#robotArmGraphic").width();
varheightR = $("#robotArmGraphic").height();
        CANVAS_WIDTH = widthR;
        CANVAS_HEIGHT = heightR;
        projector = newTHREE.Projector();
mouseVector = new THREE.Vector3();

       // SCENE
        scene = newTHREE.Scene();

       // CAMERA
initCamera(CANVAS_WIDTH, CANVAS_HEIGHT);

       // RENDERER Initializing
        renderer = newTHREE.WebGLRenderer();
renderer.setClearColor(0x000, 1.0);
renderer.setSize(CANVAS_WIDTH, CANVAS_HEIGHT);

       // Draw Axes on scene
drawAxes();

       // LIGHT
initLight();

       // FINISH SCENE SETUP
        scene3d.appendChild(renderer.domElement);
renderer.render(scene, camera);

    }
```

```
functioninitCamera(CANVAS_WIDTH, CANVAS_HEIGHT) {
         camera = newTHREE.PerspectiveCamera(45, CANVAS_WIDTH / CANVAS_HEIGHT, 0.1,
1000);
camera.position.x = 5;
camera.position.y = 0;
camera.position.z = 40;
    }

functioninitLight() {
var spot1 = newTHREE.SpotLight(0xffffff);
spot1.position.set(0, 0, 20);
scene.add(spot1);
    }

functiondrawAxes() {
varaxisX = newTHREE.Geometry();
axisX.vertices.push(new THREE.Vector3(-100, 0, 0));
axisX.vertices.push(new THREE.Vector3(100, 0, 0));
var material = newTHREE.LineBasicMaterial({ color: 0xFF0000, linewidth: 1 });
var line = newTHREE.Line(axisX, material);
scene.add(line);
```

```
varaxisY = newTHREE.Geometry();
axisY.vertices.push(new THREE.Vector3(0, -100, 0));
axisY.vertices.push(new THREE.Vector3(0, 100, 0));
var material = newTHREE.LineBasicMaterial({ color: 0xFF0000, linewidth: 1 });
var line1 = newTHREE.Line(axisY, material);
scene.add(line1);

drawAxisDashes();

    }



functiondrawAxisDashes() {
      var material = newTHREE.LineBasicMaterial({ color: 0xFF0000, linewidth: 1});
for (vari = -100; i<= 100; i++) {
if (i === 0) continue;
vardashHeight = i % 5 === 0 ?0.25 : 0.10;

if (i % 5 === 0) drawAxisNumber(i);
vardashX = newTHREE.Geometry();
dashX.vertices.push(new THREE.Vector3(i, dashHeight, 0));
dashX.vertices.push(new THREE.Vector3(i, -dashHeight, 0));
varlineX = newTHREE.Line(dashX, material);
scene.add(lineX);
vardashX = newTHREE.Geometry();
dashX.vertices.push(new THREE.Vector3(dashHeight, i, 0));
dashX.vertices.push(new THREE.Vector3(-dashHeight, i, 0));
varlineX = newTHREE.Line(dashX, material);
scene.add(lineX);
        }
    }
```

```
functiondrawAxisNumber(number) {
var x = document.createElement("canvas");
var xc = x.getContext("2d");
x.width = x.height = 64;
xc.shadowBlur = 0;
xc.fillStyle = "white";
xc.font = "30pt arial bold";
xc.fillText(number, 0, 64);

varxm = newTHREE.MeshBasicMaterial({ map: newTHREE.Texture(x), transparent: true });
xm.map.needsUpdate = true;

varmeshX = newTHREE.Mesh(newTHREE.CubeGeometry(1, 1, 0), xm);
meshX.position.x = number + 0.20;
meshX.position.y =  -0.5;
meshX.position.z = 0;
meshX.doubleSided = false;
scene.add(meshX);
varmeshY = newTHREE.Mesh(newTHREE.CubeGeometry(1, 1, 0), xm);
meshY.position.x = -0.5;
meshY.position.y =  number + 0.1;
meshY.position.z = 0;
meshY.doubleSided = false;
scene.add(meshY);
    }
```

Додаток 2

Метод за исцртување на топка, која ќе преставува можна позиција на роботската рака при симулацијата.

```
functiondrawPoint(x, y, z, options) {
    options = options !== undefined ? options : {};
constballGeometry = newTHREE.SphereGeometry(0.1, 16, 16);
constballMaterial = newTHREE.MeshPhongMaterial({ color: options.type !== undefined ? (options.type
=== "robotArmComponent" || options.type === "highlightPosition") ? 0xFF0000 :options.type ===
"experiment" ? 0xFFFB00 : 0x33aaff  : 0x33aaff });
const ball = newTHREE.Mesh(ballGeometry, ballMaterial);
scene.add(ball);
ball.position.z = z;
ball.position.x = x;
ball.position.y = y;
points.push(ball);
if (options.type !== undefined &&options.type === "robotArmComponent") {
        ball.name = "robotArmComponent";
latestRobotArmComponents.push(ball);
    }
elseif (options.type !== undefined &&options.type === "highlightPosition") {
        ball.name = "highlightPosition";
highlightedPoint = ball;
    } elseif (options.type !== undefined &&options.type === "experiment") {
        ball.name = "experiment";
experimentComponents.push(ball);
    }
}
```

```
functiondrawRobotArm(jointPoint, endPoint) {
const material = newTHREE.LineBasicMaterial({ color: randomColor() });
constrobotArm = newTHREE.Geometry();
robotArm.vertices.push(new THREE.Vector3( 0, 0, 0) );
robotArm.vertices.push(new THREE.Vector3( jointPoint.x, jointPoint.y,
jointPoint.z));
robotArm.vertices.push(new THREE.Vector3( endPoint.x, endPoint.y, endPoint.z));
const line = newTHREE.Line(robotArm, material);
 line.name = `robotArmComponent`;
scene.add(line);
latestRobotArmComponents.push(line);
drawPoint(jointPoint.x, jointPoint.y, jointPoint.z, {type: "robotArmComponent"});
drawPoint(endPoint.x, endPoint.y, endPoint.z, { type: "robotArmComponent" });
}
```

Додаток 3

Метод за поместување на камерата, односно за движење низ координатниот систем.

```
functionchangeCameraPosition(side) {
switch (side) {
case"left":
camera.position.x -= 5;
break;
```

```
case"right":
camera.position.x += 5;
break;
case"up":
camera.position.y += 5;
break;
case"down":
camera.position.y -= 5;
break;
case"in":
camera.position.z -= 5;
break;
case"out":
camera.position.z += 5;
break;
default:
  }
}
```

Додаток 4

Метод за означување на крајна позиција во координатниот систем.

```
functionhighlightPosition(x, y, z) {
if (highlightedPoint !== undefined) {
constselectedObject = scene.getObjectByName(highlightedPoint.name);
scene.remove(selectedObject);
        }
drawPoint(x, y, z, { type: "highlightPosition" });
renderScene();
}
```

Додаток 5

Метод за чистење на сцената и повторно поставување на почетните параметри. Вториот метод за прикажување на компонентите на претходно исцртаната роботска рака.

```
   functionclearScene() {
while (scene.children.length> 0) {
scene.remove(scene.children[0]);
        }
drawAxes();
initLight();
renderScene();
   }

functionclearLatestRobotArmComponents() {
if (latestRobotArmComponents.length === 0) return;
latestRobotArmComponents.map(object => {
constselectedObject = scene.getObjectByName(object.name);
scene.remove(selectedObject);
```

```
        });
latestRobotArmComponents = [];
renderScene();
    }
```

Додаток 6

Метод за чистење и пресметување на сите можни позиции на роботската рака.

```
$("#Calculate").click(function () {
showLoader();
RobotArm.ClearScene();
            $('#positionsList').find('option').remove();
var l1 = $("#txtL1").val();
var l2 = $("#txtL2").val();
var theta1Min = $("#txtTheta1Min").val();
var theta1Max = $("#txtTheta1Max").val();
var theta2Min = $("#txtTheta2Min").val();
var theta2Max = $("#txtTheta2Max").val();
varagleStep = $("#txtAgleStep").val();
$.ajax({
                method: "POST",
                url: '@Url.Action("GetCoordinates", "RobotArm")',
                data: { l1: l1, l2: l2, theta1Min: theta1Min, theta1Max: theta1Max,
theta2Min: theta2Min, theta2Max: theta2Max, agleStep: agleStep },
dataType: "json",
                success: function (data) {
hideLoader();
                    console.log(data);
if (data.Success == false) {
                        alert(data.Message);
return;
                    }
data.Positions.map((point, index) => {
var x = point.X;
var y = point.Y;
var z = point.Z;
                        $("#positionsList").append(`<option value=${index} data-
value='{"x": ${x}, "y": ${y}, "z": ${z}}'>(${x}, ${y})</option>`);
RobotArm.DrawPoint(x, y, z);
                    });
RobotArm.RenderScene();
                },
fail:function(data) {
                    alert(data);
hideLoader();
                }

            });
});
```

Додаток 7

Метод за решавање на инверзна кинематика со помош на алгебарско аналитичкиот метод. Добивање на параметрите на вториот зглоб како што се координатни точки и аглите Θ_1, Θ_2.

```
publicTask<IEnumerable<KinematicOutcome>>CalculateArmJoint(PointendPoint)
{
/*
Formulas for calculation:
    y1 = (A + D)/B
    y2 = (A - D)/B
    x1 = +-Sqrt(L1^2 - y1^2)
    x2 = +-Sqrt(L1^2 - y2^2)
    ----------------------------------------------------------------
    A = 2*c*y;
    B = 2*(y^2 + x^2);
    D = Sqrt(m - n*o);
    c = (L1^2 - L2^2 +x^2 +y^2)/2.0;
    m = 4*c^2*y^2
    n = 4*(y^2+x^2);
    o = (c^2-(x^2*L1^2));
*/
returnTask.Run(() => {
var x = endPoint.X;
var y = endPoint.Y;
var c = (Math.Pow(L1, 2) - Math.Pow(L2, 2) + Math.Pow(x, 2) + Math.Pow(y, 2)) / 2.0;
var m = 4 * (Math.Pow(c, 2) * Math.Pow(y, 2));
var n = 4 * (Math.Pow(y, 2) + Math.Pow(x, 2));
var o = Math.Pow(c, 2) - x * x * Math.Pow(L1, 2);
var A = 2.0 * c * y;
var B = 2 * (Math.Pow(y, 2) + Math.Pow(x, 2));
var d = m - n * o;
// *** Error allowed of 10^-7 if d is negative
if(d < 0) {
if(d * Math.Pow(10, 7) < 0)
      d = 0;
else
thrownewException($"Cannot calculate joint point for this end point: x: {x} y: {y}");
    }
    d = Math.Sqrt(d);
var y1 = (A + d) / B;
var y2 = (A - d) / B;
var x1 = Math.Sqrt(Math.Pow(L1, 2) - Math.Pow(y1, 2));
var x2 = Math.Sqrt(Math.Pow(L1, 2) - Math.Pow(y2, 2));

varlistOfPoints = new List<Point> {
new Point {X = x1.Round(), Y = y1.Round(), Z = 0}, new Point {X = x2.Round(), Y = y2.Round(), Z = 0},
new Point {X = -x1.Round(), Y = y1.Round(), Z = 0}, new Point {X = -x2.Round(), Y = y2.Round(), Z = 0}
}.Select(RoundWithTolerance).Distinct(newCustomPointEquality());


return
listOfPoints.Select(point =>
newKinematicOutcome(FindTheta1WhenJointPointGiven(point), FindTheta2WhenJointPointGiven(point, endPoint),
point))
.Where(outcome =>
outcome.Theta1.Between(Theta1Min, Theta1Max, true)
&&outcome.Theta2.Between(Theta2Min, Theta2Max, true)
&&ValidateJointPointWithEndAndZeroPoint(outcome.JointPosition, endPoint)
            );
 });
}
```

Додаток 8

Дел од методите и класите користени за имплементација на ANFIS.

```
publicclassANFIS {
privateintinputDim, outputDim, numOfRules;

privateIRule[] ruleBase;

publicIRule[] RuleBase { get { returnruleBase; } set { ruleBase = value; } }

public ANFIS(IList<IRule>RuleSet) {
if (RuleSet == null || RuleSet.Count == 0)
thrownewException("Ruleset is empty");
this.numOfRules = RuleSet.Count;
ruleBase = RuleSet.ToArray();
  }

publicdouble[] Inference(double[] x) {
returnInference(x, ruleBase);
  }

publicstaticdouble[] Inference(double[] x, IList<IRule>RuleBase) {
intNumOfRules = RuleBase.Count;

intOutputDim = RuleBase[0].Z.Length;

double[] firings = newdouble[NumOfRules];
double[] y = newdouble[OutputDim];
doublefiringSum = 0.0;

for (inti = 0; i<NumOfRules; i++) {
      firings[i] = RuleBase[i].Membership(x);
firingSum += firings[i];
    }

for (inti = 0; i<NumOfRules; i++)
for (int C = 0; C <OutputDim; C++)
      y[C] += firings[i] / firingSum * RuleBase[i].Z[C];


returny;
  }

}
```

```
publicstaticclassANFISBuilder<R>whereR :IRule, new() {
static Logger _log = newLogger("ABuilder", InternalTraceLevel.Default,
TextWriter.Null);
publicstatic ANFIS Build(double[][] input, double[][] output,
IRuleExtractorRuleExtractor, ITraining trainer, intMaxIterations) {
    _log.Info("Start...");
    _log.Info($"Constructing initial rule set with
[{RuleExtractor.GetType().Name}]");
varruleBase = RuleSetFactory<R>.Build(input, output, RuleExtractor).Select(z => z
asIRule).ToList();
    _log.Info($"Get {ruleBase.Count} initial rules.");
int epoch = 0;

doubletrnError = 0.0;
```

```
Console.WriteLine();
Console.WriteLine();
do {
trnError = trainer.Iteration(input, output, ruleBase);
      _log.Info($"Epoch {epoch}, training error {trnError}");

if (double.IsNaN(trnError)) {
        _log.Info("Failure! Training error is NAN.");
thrownewException("Failure! Bad system design.");
      }
    } while(!trainer.isTrainingstoped() && epoch++ <MaxIterations);


    ANFIS fis = newANFIS(ruleBase);
    _log.Info("Done");
returnfis;
  }
}
```

```
varruleBase = RuleSetFactory<R>.Build(input, output, RuleExtractor)
       .Select(z => z asIRule).ToList();
```

```
public Task<bool>TrainANFIS(intruleNumber, intmaxIterations,
booluseAnalicitalOutcomeForTraining = false) {
returnTask.Run(() => {
if(!IsDataSetCalculated)
thrownewApplicationException("DataSet is not calculated or provided.");

varsampleSize = Positions.Count() - 1;
vardynamicObj = useAnalicitalOutcomeForTraining ?Positions.Select(x =>new{ Point =
x, KinematicOutCome = CalculateArmJoint(x).GetAwaiter().GetResult().FirstOrDefault()
}) : null;

var input = useAnalicitalOutcomeForTraining
            ? dynamicObj.Select(x =>x.Point).ConvertToANFISParameter()
            : Positions.ConvertToANFISParameter();
var theta1ANFIS = Task.Run(() => {
var sPropTheta1 = newStochasticQprop(sampleSize);
var extractorForTheta1 = newKMEANSExtractorIO(ruleNumber);
varexpectedOutcome = useAnalicitalOutcomeForTraining
                  ? dynamicObj.Select(x =>new[] {
x.KinematicOutCome.Theta1.ConvertRadiansToDegrees() }).ToArray()
                  : AnglesGrid.First().ConvertToANFISParameter();
      Theta1ANFIS = ANFISBuilder<GaussianRule>.Build(input,
expectedOutcome, extractorForTheta1, sPropTheta1, maxIterations);
    });
var theta2ANFIS = Task.Run(() => {
var sPropTheta2 = newStochasticQprop(sampleSize);
var extractorForTheta2 = newKMEANSExtractorIO(ruleNumber);
var expectedOutcome2 = useAnalicitalOutcomeForTraining
                ? dynamicObj.Select(x =>new[] {
x.KinematicOutCome.Theta2.ConvertRadiansToDegrees() }).ToArray()
                : AnglesGrid.Last().ConvertToANFISParameter();

      Theta2ANFIS = ANFISBuilder<GaussianRule>
.Build(input, expectedOutcome2, extractorForTheta2, sPropTheta2, maxIterations);
    });

Task.WaitAll(theta1ANFIS, theta2ANFIS);
```

```
IsANFISTrained = true;
returntrue;
  });
}
```

```
public Task<KinematicOutcome>CalculateAngelsUsingANFIS(Point endPoint) {
if(!IsANFISTrained)
thrownewApplicationException("ANFIS is not trained");
returnTask.Run(() => {
var theta1 =
Theta1ANFIS?.Inference(endPoint.CovertToANFISParameter()).FirstOrDefault() ?? 0;
var theta2 =
Theta2ANFIS?.Inference(endPoint.CovertToANFISParameter()).FirstOrDefault() ?? 0;

varjointPoint = FindJointPointWhenAngleGiven(theta1.ConvertDegreesToRadians());

returnnewKinematicOutcome(theta1.Round(), theta2.Round(), jointPoint);
  });
}
```

Додаток 9

Имплементација на Гаусовата членска функција.

```
publicdouble Membership(double[] x) {
double exponent = 0.0;
for (inti = 0; i<xdim; i++)
    exponent += pow2((x[i] - parameters[i]) / parameters[i + xdim]);

returnMath.Exp(-0.5 * exponent);
}
```

Додаток 10

Имплементација на експериментите.

```
publicabstractclassExperiment :IExperiment {
publicIEnumerable<IKinematicOutcome>ActualOutputs{ get; privateset; }
publicIEnumerable<IKinematicOutcome>AnfisOutputs{ get; privateset; }
publicIEnumerable<Point>ExperimentPositions{ get; protectedset; }

public Task<IEnumerable<MathErrorOutcome>>CalculateError() {
varexpectedOutcomes = ActualOutputs?.Where(x => x != null).ToArray();
varactualOutcomes = AnfisOutputs?.Where(x => x != null).ToArray();
if(expectedOutcomes == null
       || actualOutcomes == null
       || expectedOutcomes.Length != actualOutcomes.Length)
thrownewArgumentException("Size of mathematical and anfis does not match.");
returnTask.Run(() => {
returnexpectedOutcomes.Select((t, i) =>CalculateMathError(t, actualOutcomes[i])).Where(x => x !=
null);
    });
  }

publicabstractIEnumerable<Point>GeneratePositions
   (double radius, double step = 0.1745, doubleshiftX = 0, doubleshiftY = 0);
```

```
publicvoidSetActualOutputs(IEnumerable<IKinematicOutcome>actualOutputs) {
ActualOutputs = actualOutputs;
  }

publicvoidSetAnfisOutputs(IEnumerable<IKinematicOutcome> outputs) {
AnfisOutputs = outputs;
  }

protectedvirtualMathErrorOutcomeCalculateMathError
           (IKinematicOutcome mathematical, IKinematicOutcomeanfis) {
if(double.IsNaN(mathematical.Theta1)
    || double.IsNaN(mathematical.Theta2)
    || double.IsNaN(anfis.Theta1)
    || double.IsNaN(anfis.Theta2))
thrownewException("Some angle is NaN");
var result = newMathErrorOutcome {
      Theta1Error = (mathematical.Theta1 - anfis.Theta1),
      Theta2Error = (mathematical.Theta2 - anfis.Theta2)
    };
returnresult;
  }
}
```

Додаток 11

Методи за извршување на експериментите.

```
$("#circleExperiment").click(() => {
leturl = '@Url.Action("CircleExperiment", "RobotArm")';
let data = { radius: 3, step: 0.1745, shiftX: -0.5, shiftY: 7 };
doExperiment(url, data);
});

$("#squareExperiment").click(() => {
leturl = '@Url.Action("SquareExperiment", "RobotArm")';
let data = { side: 3, step: 0.2, shiftX: 3, shiftY: 9 };
doExperiment(url, data);
});

    $("#sinExperiment").click(() => {
leturl = '@Url.Action("SinExperiment", "RobotArm")';
let data = { length: 10, step: 0.3, shiftX: 3, shiftY: 9 };
doExperiment(url, data);
});
```

```
functiondoExperiment(endpointUrl, experimentData) {
showLoader();
RobotArm.ClearExperiment();
$.ajax({
        method: "POST",
        url: endpointUrl,
```

```
        data: experimentData,
dataType: "json",
        success: function (data) {
            console.log(data);
if (data.Success == false) {
                alert(data.Message);
hideLoader();
return;
            }
            $("#txtAnfisErrorOutcome").val(parseMathErrorResults(data.Outcome));
data.ExperimentPositions.map((point, index) => {
var x = point.X;
var y = point.Y;
var z = point.Z;
RobotArm.DrawPoint(x, y, z, {type: "experiment"});
            });
RobotArm.RenderScene();
hideLoader();
        },
fail:function (data) {
            alert(data.Message);
hideLoader();
        }

    });
}
```