Universidad del País Vasco
Euskal Herriko Unibertsitatea

VITORIA-GASTEIZKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE VITORIA-GASTEIZ

INGENIERITZA MEKANIKOKO GRADUA

GRADU AMAIERAKO LANA

IKASKUNTZA SAKONA (*DEEP LEARNING*) TEKNIKEN AZTERKETA ETA ERABILPEN PRAKTIKOA BURMUIN ERRETRAKTORE BATEN KONTROLEAN

Egilea:

Erik Gorospe Hernaez

Zuzendaria:

Ekaitz Zulueta Guerrero

VITORIA GASTEIZKO INGENIERITZA ESKOLA

2023

ESKERRAK

Eskerrak, Ekaitz Zuluetari proiektu guztiaren inguruan egindako lanagatik. Proiektuan zehar pazientziarekin guidatu nau, une oro nire zalantza eta galderei erantzunak bilatuz eta *"Reinforcement Learning"* mundua nire begitara zabaldu du.

Mila esker, Teo Ricori unibertsitatea aukeratzeko orduan aholkuak emateagatik, oso pozik nago egindako hautaketarekin. Horretaz gain, eskerrak, urtero izandako zalantza askotan laguntza eskaintzeagatik eta gradu amaierako lanaean hasierako guidapen bat emateagatik.

Eskerrik asko, Euskak Herriko Unibertsitateari Matlab lizentzia emateagatik; Izan ere, lanean erabilitako herramientarik garrantzitsuena izan da.

Análisis práctico para un control cerebral basado en técnicas de aprendizaje por refuerzo.

Resumen

El aprendizaje por refuerzo es una técnica que día a día adquiere más importancia. Su concepto base es el "prueba y error", en este, un agente, que será la representación virtual de nuestro robot, interactúa con un entorno para la realización de una tarea. A través de un sistema de recompensas el agente, después de un entrenamiento, finaliza la tarea con la solución más óptima para cumplir su objetivo.

Esta técnica últimamente se está poniendo en práctica en el área de la medicina. Pero no se ha realizado ningún intento para el uso de la misma a la hora de realizar la maniobra de retracción cerebral, la cual, se basa en abrir las dobleces naturales del cerebro para así poder acceder a partes más profundas de él. Esto genera tal estrés en el cerebro que puede conllevar a traumas post operatorios.

Por ello, en este trabajo se ha construido un agente y un entorno, además de un sistema de recompensas, que abordan este problema y proponen una solución óptima, adquirida a través del aprendizaje profundo.

Indartze Ikaskuntzaren tekniken analisia eta erabilpena burmuin erretraktore batentzako.

Laburpena

Indartze bidezko ikaskuntza egunez egun garrantzi handiagoa hartzen duen teknika da. Oinarrizko kontzeptua "proba eta errorea" da. Agente bat, gure robotaren irudikapen birtuala izango dena, ingurune batekin elkarreragiten du zeregin bat egiteko. Sari-sistema baten bidez eta entrenamendu baten ostean agenteak bere helburua betetzeko irtenbiderik onenarekin amaitzen du.

Teknika hau medikuntzaren arloan praktikan jartzen ari da azkenaldian. Halaber, ez da hura erabiltzeko saiorik egin garunaren erretrakzioa maniobra egiteko orduan. Maniobra hau burmuinaren tolesdura naturalak irekitzean oinarritzen da, horrela burmuinaren zati sakonagoetara iritsi ahal gaitezke. Honek estres handia eragiten du burmuinean eta ebakuntza osteko traumak eragin ditzazke.

Hori dela eta, lan honetan agente ingurune eta sari-sistema bat eraiki da, arazo honi aurre egiten diotenak. Honela arazoari konponbide optimo bat aurkitu zaio, ikaskuntza sakona erabiliz.

Aurkibidea

1. SARRERA ETA HELBURUAK.

Ibai Inziarte-Hidalgok, Irantzu Uriartek, Unai Fernandez-Gamizek, Gorka Sorrosalek eta Ekaitz Zuluetak idatzitako *"Robotic-Arm-Based Force Control in Neurosurgical Practice"* artikuluaren ikasketa jarraituz [1] gradu amaierako lana landu dut.

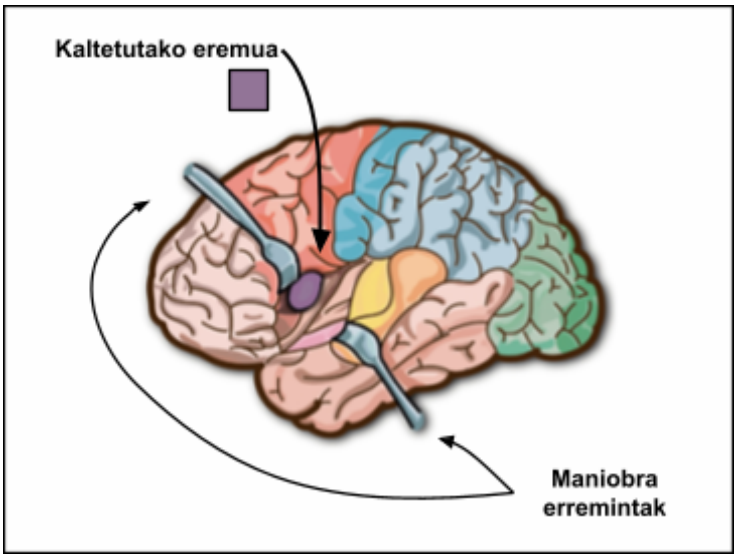
Artikuluan burmuin erretraktore robot baten kontrol optimoa egiten da. Amaieran, egindako experimentuaren ondorioak eta egin daitezkeen aurrerapenak eztabaidaten dira, non, *Reinfocement Learning* tekniken erabilpena proposatzen da. Hurrengo lanean proposamen hau landu da.

1.1. TESTUINGURUA.

Artikuluan [1] adierazten den moduan ikasketen arabera [2] urtero 22.6 milioi pertsona lesio neurolojikoak jasaten dituzte eta horietako 13.8 milioi ebakuntza-gelara joan beharra daukate. Esaterako, soilik Estatu Batuetan Kirurjilari Neurologikoen Amerikako Elkartearen (AANS-ren) arabera 50.000 neurokirurgiak egiten dira urtero, ordu erdi baino gehiago irauten dutenak. Garun ehunaren ebakuntza hasterakoan ordu laurden pasa daiteke, garunean neurrizko presioa ezartzen, ebakuntza osteko traumak eragin gabe. Denbora horren ondoren, garuna ebakuntza osteko traumak jasotzeko probabilitatea exponentzialki handitzen da minuturo [3].

Lan honetan **burmuin erretrakzioa** landuko dugu, lesio neurolojiko hauetan egin beharreko prozedura ohikoenetarikoa delako. Ebakuntza honetan kirurjilaria kaltetutako eremura iristeko *"brain retraction"* (burmuin erretrakzioa) deituriko maniobra egin behar du [4]; Izan ere, kaltetuko eremua burmuinaren barnealdean dago. Prozedura hau burmuinak dauzkan tolesdura naturalak beraien artean bereiztean datza, barneko eremura iristeko (Ikus **1.irudia**). Dena den, adituak [3]-n nabarmendu dute burmuinean egindako presioa garuneko ehuna hondatu dezakela era askotara. Presioa handiegia bada, burmuinatik zabaltzen hasten da deformazioak eragiten eta zainak ostopatzen, oxigenoaren bideraketa ostopatuz. Kaltetutako eremuaren larritasuna/garrantzia alderdi askoren baitan dago [5], esaterako, erretrakzioak eragindako presioaren zabalera, geometria, denbora, grabitatea, fluido galera, deformazio muga (20mm kasu ohikoetan) eta pertsona bakoitzaren burmuin

zurruntasuna. Burmuin kontusioak eta infartuak eman daitezke prozedura egiterakoan



[6,7,8,9]

1..Irudia: Burmuin erretrakzioaren irudikapena.

Adituek [10]-n giza biztanleriaren burmuin barne-arteriaren bolumena (rCBF) $10 - 13\, mL\, 100\, gm^{-1}\, min^{-1}$ baino handiagoa izan behar dela esaten dute, iskemiaren ondoriozko oinazeak saihesteko. Laha et al.-en ikerketen arabera [11], 70mmHg baino handiagoa izan behar da batez besteko arteria-presioaren eta burmuinaren erretrakzioa eragindako presioaren arteko kendura, garuna ez kaltetzeko. Bestalde, ezberdintasuna 200 mmHg baino handiagoa baldin bada, burmuina guztiz sendatuko da.

$$70\, mmHg \; < \left| P_{arteria} - P_{erretrakzioa} \right| \rightarrow \; Garunan\ egon\ daitezkeen\ kalte\ gehienak\ ekiditu.$$

$$200\, mmHg \; < \left| P_{arteria} - P_{erretrakzioa} \right| \rightarrow \; Garunan\ guztiz\ sendatuko\ da.$$

Oro har, klinikoki kasu postoperatorio esanguratsuenak %3-%9 inguru aldera gertatzen dira, ehuneko hau alda daiteke prozezuaren zailtasunaren arabera [12,13]. Buruan gertatzen diren kirurgietan espezifikoki gertatzen diren konplikazioen %10 garunaren erretrakzioarekin dute harremana, adibidez, hematomak, afasia, hemiparesia eta logalea eman daitezke.

Estadistikak ikusita, adituek egoera hobetzeko hainbat ekintza egitera bultzatu ditu. Nabarmenenak robot kirurujikoen inplementazioa, operazioaren efizentzia hobesteko gizakiak dituen limitazioak murriztuz kontrola medikuaren eskuetan jarriz [3], eta MIS *(Minimal Invasive Surgery)* [6], ebakuntza handiak ekiditzen dituen teknika, dira. Bi hauek konbinatuta **MIRS** *(Minimally Invasive Robotic Surgery)* sortu da. Honetan robotak erabiltzen dira medikuarekin/osasun-langilearekin batera lan egiteko, gizakiak dituen limitazioak murriztuz eta gaitasunak hedatuz. Pazienteak teknika berri hauek jasoz ikusitako onurak: eguneratzeko denbora laburragoak, postoperazio min gutxiagoa, ospitale egonaldi laburragoak, lehenetsitako leheneratze denbora laburragoak, sistema immunologikoaren gainkarga gutxitua eta ospitale kostu baxuagoak izan dira [7,8].

MIRS-ren barne Hoecklemann-ek hiru oinarri ezarri ditu [11]. Bat, sistema teleoperatuak dira, non, robota langilearen baitan lan egiten du interfaz baten bitartez. Bi, irudiz guidatutako sistemak, non, robota ejekutatzen du lehenago ezarritako plana sentsore eta jarraipen sistemek emandako informazioren laguntzaz eguneratuz. Eta hiru, orientazio aktiboa da, non, langilean robota eskuz guztiz kontrolatzen du [14].

1.2. HELBURUA.

Gure helburu nagusia *Reinforcement Learning*-en bitartez burmuin erretraktore baten kontrol optimoa lortzea da. Horretarako [1] artikuluan eraman zen prosezua eta izan zituzten erantzunak laburbilduko ditugu.

Egileek [1]-n egin zuten lehenengo gauza, abiaduraren ekuazio optimoa kalkulatzea izan zen. Abiadura hau, haiek zituzten bi helburuak betetzen zituen:

1. Robotak burmuina zabalduko du guk desiratutako posizioa lortu arte.
2. Iskemia kausa ditzazkeen ondorioak ahalik eta gehien ekidituko dira, indarra eta abiadura murriztuz.

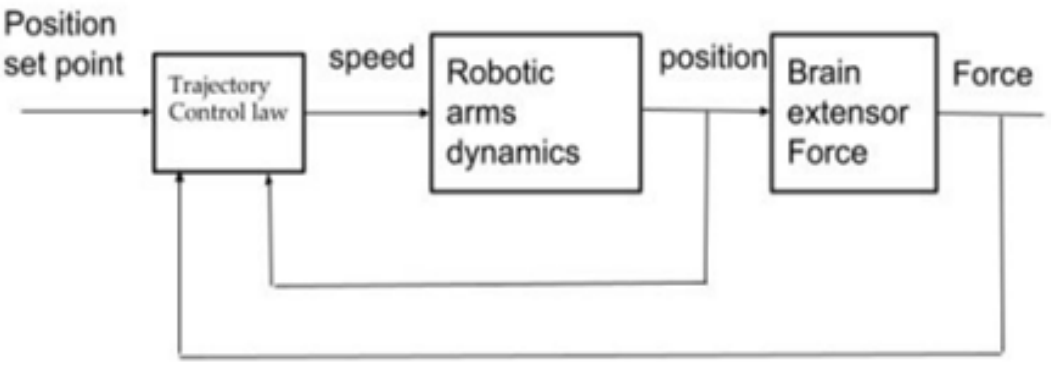
1.Ekuazioan ikus dezakegu [1]-n ateratako abiadura ekuazioa:

$$v(x) = \sqrt{\frac{\lambda_F (x_d - x)^2 F^2(x) + \lambda_{error} (x_d - x)^2}{2\lambda_v}} \quad (1)$$

1.Taula. [1]-n erabilitako sistema eta kontrol aldagaiak.

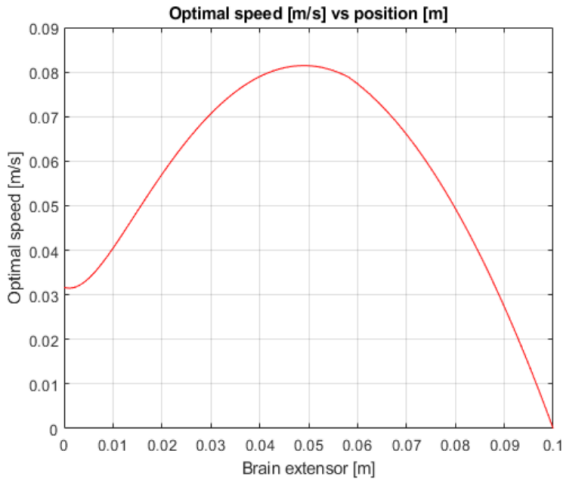
Aldagaiaren izena	Definizioa	Balioa unitateekin edo soilik unitateak
λv	Abiadura karratuaren terminoaren ponderazio-koefizientea.	10−3 s 2 m−2
λerror	Posizioaren karratuaren konfigurazio-errorearen ponderazio-koefizientea.	10−4 m−2
λF	Indar karratua terminoaren ponderazio-koefizientea.	1 N−2
x	Garuneko ehunen desplazamendua.	m
xd	Garuneko ehunen desplazamenduaren helburua.	m
F	Garuneko erretraktileek aplikatutako indarra.	N

Formula honekin batera kontrol sistema bat eraiki zen. **2.irudian** ikus dezakegun moduan kontrol honetan posizioa hasieran (x_t), hurrengo posizioa (x_{t+1}) eta hurrengo indarra (F_{t+1}) hartzen ziren sarrera moduan. Honekin eta **1.ekuazioarekin** robota mugitzeko behar zuen abiadura kalkultzen zuen momentu oro.

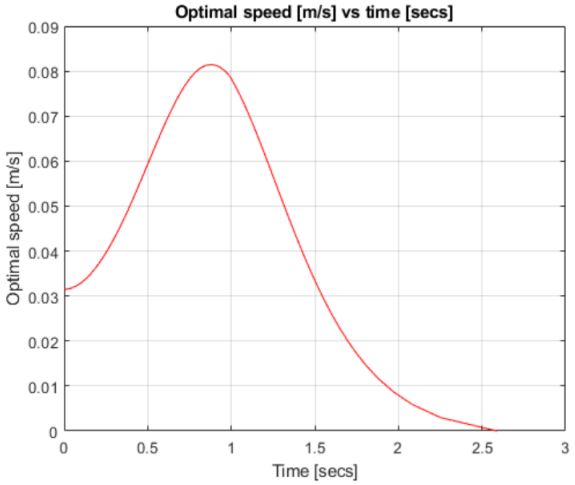


2.Irudia: [1]-n eraiki zen kontrol sistema.

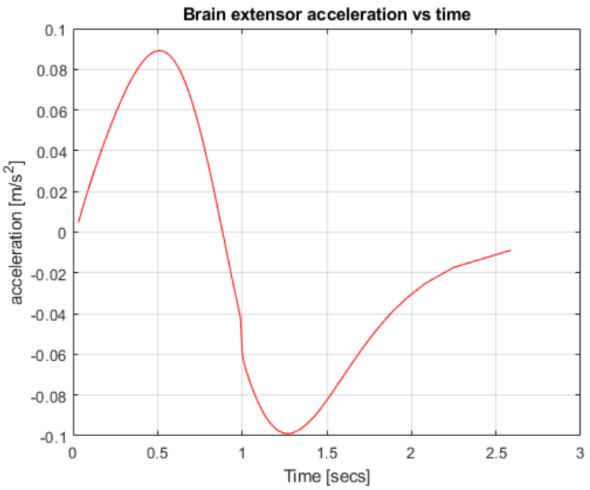
Kontrol sistema abian jartzerakoan hurrengo erantzunak atera zituzten (ikus **3, 4, 5 eta 6.irudiak**).



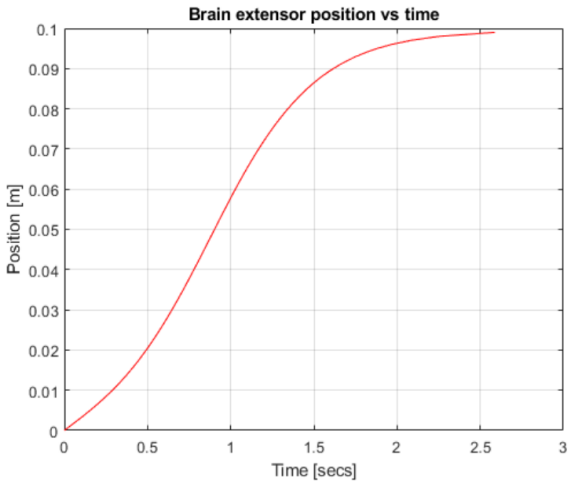
3.Irudia: Robotaren abiadura posizioaren baitan.



4.Irudia: Robotaren abiadura denboraren baitan.



5.Irudia: Robotaren azelerazioa denboraren baitan.



6.Irudia: Robotaren posizioa denboraren baitan.

Guk egindako ikasketa guztia eredu honetan oinarritu denez, gure helburua erantzun hauek isladatzea izango da. Halere, erantzun hauek bi helburutatik aterata daudenez, guk ere hoiek lortzen saiatuko gara.

1.3. MOTIBAZIOA

General Electric 1961-ean "*Unimate*" robota aurkeztu zuenetik, beso robotikoen garrantzia industrian gero eta handiagoa bilakatu da [15]. Hauek gizakia ordeztu dute edo berarekin lanean hasi dira, eremu arriskutsuetan langilearen segurtasuna hobestuz edo ekoizpen lerrotan efinzientzia eta produktibitatea handituz.

Roboten erabilpena ekarritako onurak, ideia beste lan-eremu askotara zabaltzea ekarri du, horietako bat medikuntza. Medikuntzarako beharrezkoa den ahalmen garrantzitsuenetarikoak zehaztasuna eta eraginkortasuna dira. Robota hauek betetzen baditu kirurjian erabili daiteke [16]. Halere, malgutasuna da gehienetan bilatuko den ezaugarria. Adibidez, makina-erreminta ia perfektu batek kadena zinematiko berean matxarda mota asko erabiltzeko ahalmena izango du, hau robotaren bizkortasunarekin batera, funtzio asko bete ahal dituen beso robotiko bat emango digu.

Beso robotiko hauen kontrola konplexua da. Robot familia bakoitza kontrol programa bat behar du eta hau kostu ekonomikoa asko handitzen du.

Makina hauekin erlazionatuta dauden kontrol numeriko tradizionalak beharrezkoa den zehaztasuna ematen dute, baina hau lortzeko beharrezkoa den programazioa handia eta zaila da. Horretaz gain, normalean kontrol hauek ez dute beso-robotiko guzti hauen malgutasuna aprobetxatzen; Azken batean, robotaren atal kantitatea handitzen dugun bitartean, metodo algebraiko hauen zailtasuna (alderantzizko zinematika kalkulatzea) exponentzialki handitzen da.

Guk erabiliko dugun estrategian (RL-n), inguru birtual bat eraikiko dugu, sarrera moduan guk diseinatuko dugun robota emango dituen aldagaiak sartuz. Gure robotaren ekintzak hasieran, entrenamendu gabe, guztiz auzazkoak izango dira, hau da, mugimendua kaotikoa izango da eta simulazioetan isaladatuko den moduan, ez dugu estrategiarik izango arazoari aurre egiteko. Dena den, soilik sari funtzio bat sortuz eta simulazio asko/entrenamendu bat izanez, gure makina arazoaren erantzun optimoa "ikasiko" du.

Reinforcement Learning berarekin ekartzen duen koste komputazionalak handiak badira ere, teknika hauen onurak arazoaren sinplifikazioa eta malgutasunaren erabilpena dira.

1.4. ANTOLAKETA

Lana hurrengo antolaketa eramango du. 2.kapituluan lana ulertzeko *Reinforcement Learning*-etaz beharrezkoa den eremu teorikoak landuko ditugu, algoritmo familia klasikoen azalpenak eta familiak osatzen dituzten subalgoritmoen (*Actor-Critic*) azalpen laburrak emanez. Horretaz gain, aurrerago atera ziren algoritmo eboluzioen artean (DDPG, SAC, A2C…) guk lanean erabiliko duguna aukeratuko dugu, gure arazoaren lagin efizientzia, ekintza-eremua eta estabilitatea aztertuz.

Ondoren, 3.kapituluan guk aukeratutako algoritmoaren azalpena emango da, hau ulertzeko beharrezkoa den informazio guztia emanez.

Bukatu aurretik, 4.kapituluan gure arazoari aurre egiteko beharrezkoa den kodigo guztiaren azalpena emango dugu, optimizazio prozesua gehituz.

Bukatzeko, ondorioak eta etorkizunean eman daitezkeen aurrerapenak eztabaidatuko ditugu.

2. *REINFORCEMENT LEARNING* (Indartze Ikaskuntza)(RL).

Hurrengo azalpena Richard S. Sutton y Andrew G. Barto ikerlariek idatzi zuten *"Reinforcement Learning: An introduction"* liburuan [17] oinarrituta egongo da.

Reinforcement Learning ideia biologiatik dator; Azken batean, animaliek bizitzan saiakuntza anitzen akatsetik ikasten dute. Gaztelaniaz kontzeptu honi "prueba y error" deitzen zaio. RL-k idei hau hartu eta adimen artifizialetan sartzen du, arazoen konponketarako.

Baina, nola jakingo du adimen artifizial bat ikasketan hartu beharreko bide egokia? Honetarako sarietan (*reward*-etan) oinarritzen den sistema bat eraikitzen da[1]. **Sari sistema** funtzio batean oinarrituta dago, **sari-funtzioa** (*reward function*). Funtzioa guk izango ditugun helburuak isladatuko ditu, ondorioz, zenbat eta, sari gehiago jaso, orduan eta, arazoaren erantzun optimori hurbilago egongo gara.

Sari guztien batuketari "**Itzulera"** (retorno) deituko diogu, kontuz, hau sari indibidualarekin (*reward*-arekin) ez nahastea, bibliografiaren artikulu askotan gertatzen den moduan. Itzulera erabiltzen da soilik amaierako egoera garrantzia duen arazoetan, lehiaketa moduko arazoetan, non, askotan garrantzia duen gauza bakarra irabaztea da. Irabazteak itzulera postibo bat izango du eta galtzeak, aldiz, negatiboa.

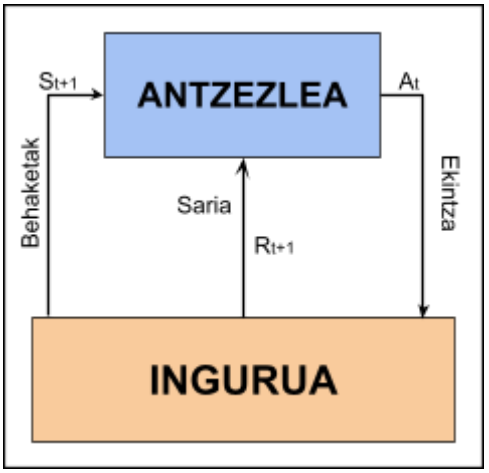
RL teknikarekin irtenbidea ateratzen zaion arazo batean, gure robota errepresentatzen duen kodigo/programazioari "**Agentea**" deitzen zaio. Agentea berarekin eta inguruarekin interakzionatzen du, ekintzen bitartez.

Ingurua egin beharreko atazaren errepresentazioa da. Inguruak agenteak aukeratutako **ekintzei erreakzionatzen** du, honela agentearekin komunikatuz. Ekintza hauen ondorioz, ingurua aldatzen da bere **egoera** modifikatuz. Agenteak egoera hau ikus eta baloratu dezake (ikus **7.irudia**).

Egoera atazan garrantzi handia duten aldagaiez osatuta egongo da, horregatik espazio bektorial propio bat izango du, **egoera-espazioa** deituko duguna. **Behaketak** ere aldagaiez osatuta egongo dira, hauek izango dira agenteak ekintzak aukeratzeko

[1] Paulov-ek erreflexu ikasiaren edo erantzunaren ikasketekin antzekotasuna du [18].

ingurutik jasoko duen informazioa. Behaketek beste espazio bektorial bat osatuko dute (**behaketa-espazioa**).



7.Irudia: RL- ikasketa prosezuaren eskema simplifikatua.

Agentearen portaera funtzio matematiko baten moduan errepresenta daiteke, honek aldagai moduan behaketak izango ditu. Funtzio honi **politika** deituko diogu eta agentearen portaera definituko du ingurumenaren egoera desberdinen aurrean. Bi politika mota desberdintzea garrantzitsua da. Alde batetik, **politika deterministikoa** taula batean errepresenta daiteke egon daitezkeen ekintza-egoera pareak finituak direlako. Bestetik, ekintza-egoera pareak infinituak direnean politikari **estokastikoa** deituko diogu.

2.1. ZERGATIK RL?

Imagina dezagun, zaborra klasifikatzeko hiru artikulazio eta amaieran matxarda bat duen beso robotikoa. Kontrol optimo bat izateko egin ditzazkeen mugimendu guztiak kalkulatu beharko ditugu. Horretarako **alderantzizko zinematika** (cinematica inversa) erabiliko genuke. Kalkulu hauek askotan oso konplexuak dira eta gehienetan robotaren konplexutasunarekin batera kalkuluen zailtasuna exponentzialki handitzen da. Kasu hoberenean, kalkulu guztiak egin ondoren robotaren posizioa (estatika), mugimendua (zinematika) eta indarra (dinamika) kontrolatu genezake. Metodo zaharkitu hau denbora asko dakar eta malgutasun txikia du. Horretaz gain, portaera inteligentea simulatzea oso zaila izango da.

RL teknika erabiliz, arazoa sinpleagoa bilakatzen da. Soilik beso robotikoa bere baitan mugitu daitekeen ingurumen bat diseinatuz eta sarietan oinarritutako sistema bat eraikiz, robota guztiz ikas dezake portaera egoki bat izaten. Halere, sarietan oinarritutako sistema eraikitzea ez da hain erraza ez badakizu lortu nahi dituzun helburuak eta neurtu ahal diren aldagaiak zehazki zeitzuk diren; Izan ere, lan honeta edukiko dugun ataz garrantzitzuena eta zailena izango da.

2.2. ALGORITMOAREN AUKERAKETA

RL-ren barruan algoritmo kantitate handia dago. Halere, guk ez ditugu guztiak azalduko debora asko dakarrelako. Horregatik, lehenengo, guk erabiliko dugun algoritmoaren aukeraketa egingo dugu, ondoren (3.atalean) algortimo hori ulertzeko behar diren azalpen guztiak emanez.

Aukeratu ditzazkegun algortimo guztiak sare neuronalen erabilpena implementatzen dituzte, honi *"Deep Learning"* (ikaskuntza sakona) deitzen zaio.

2.2.1. Sample efficiency. Zenbat data beharko dugu, errendimendu handienera iristeko? (Steps)

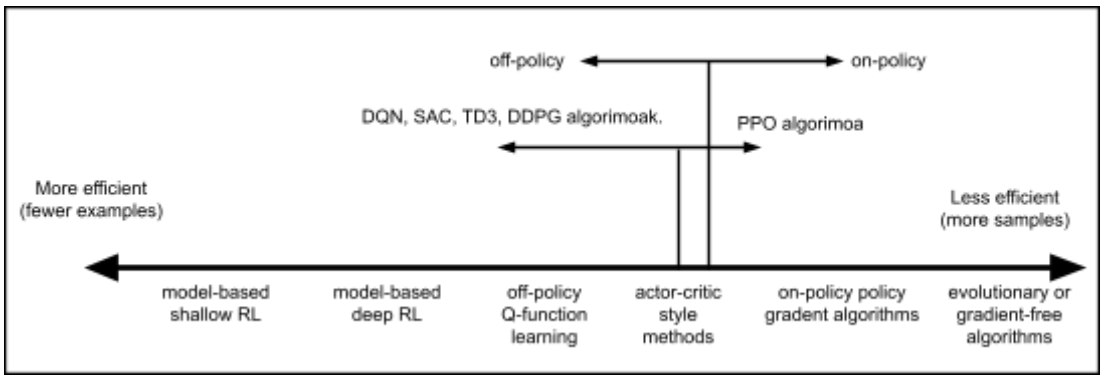
Sample efficiency edo **lagin eraginkortasuna** garrantzitsua da, ingurua simulatzea ezinezkoa denean eta entrenamedua inguru fisiko batean egin behar denean [17,19]. Izan ere, robotaren limitazio fisikoak ez dute simulazio azkarrak egiteko ahalmena ematen; Azken batean, ordenagailu simulazio batek bere abiadura handitu dezake bideo baten bizkortasunaren modura, baina errealitatean robotaren limite fisikoak gainditzea ezinezkoa da.

Inguru birtual batean simulatzen badugu ere, askotan, entrenamendua gauzatzeko behar dugun data errealitatetik jasotzen da. Adibidez, *"trading"* egiterakoan, data eguneraketa denbora finko bat du eta ezin dugu hau murriztu. Honek robota denbora jakin batean egin ditzazkeen simulazioak murrizten ditu.

Horretaz gain, lagin eraginkortasuna konputazio denbora txikiagoak ekar ditzazke. Dena den, inguru birtual batean paralelizazioarekin batera denbora desberdintasun hau hautemaezina da.

Gure kasuan, simulazio guztiak inguru birtual batean gauzatuko dira eta beharrezko informazioaren eguneraketa inguru birtualaren baitan geratuko da. Ondorioz, lagin eraginkortasuna, gure kasuan, ez du garrantzi handirik izango.

RL-ek dituen algoritmoak efizientziarekiko eta lagin kantiteareiko klasifikatuta daude **8.irudian**. "*Off-policy*" algoritmoek, lagin berri bat sortu gabe politika aldatu dezakete, *"on-policy"* algoritmoek ,aldiz, lagin berri bat sortzen dute politika aldatzen duten bakoitzean.



8.Irudia: RL- algoritmoen sailkapena efizientziarekiko eta lagin kantiteareiko.

Gure kasuan *"sample efficiency"* garrantzia handirik ez duenez *"actor-critic style methods"* inguruan mantenduko gara. Aukeratu dezakegun algoritmoa, ondorioz, DQN, SAC [20], TD3 [21], DDPG [23] edo PPO [24] izan daitezke.

2.2.2. Egonkortasuna.

Antzezle-kritiko metodologian (*actor-critic style methods*) dauden algoritmoen artean, bat aukeratzeko, erabiliko dugun hurrengo sailkapena egonkortasunean oinarrituko da.

Zer da **egonkortasuna**? Kasu honetan, egonkortasuna izango dugun dataren aniztasuna neurtuko du. Gure kasuan, data askotan alda daiteke gizakibakoitza bere baitan desberdina delako, ondorioz, gure agentea egoera asko aztertu beharko ditu.

Ondorioz, genituen algoritmoen artean egonkortasun txikiena duenez, PPO algoritmoa baztertuko dugu.

2.2.3. Ekintza: Diskretuak edo jarraituak.

Ekintzak diskretuak edo jarraiak izatea, algoritmoen aukeraketan garrantzi handia duen alderdia da. Ekintza diskretuak izatea, gure antzezleak dituen aukerak finituak

direla adierazten du, adibidez, mahai jolas askotan gure pertsoanaia mugitzeko ditugun aukerak limitatuak dira atera dugun daduaren zenbakiarekiko. Ekintza jarraituak izatea, aldiz, antzezleak dituen aukerak infituak direla jakinarazten du, esaterako, kotxe bat gidatzerakoan azeleragailua duen posizioan infituak izan daitezke.

Geratzen zaizkigun algoritmoak hauek dira, eta ekintza mota hauekin lan egin dezakete:

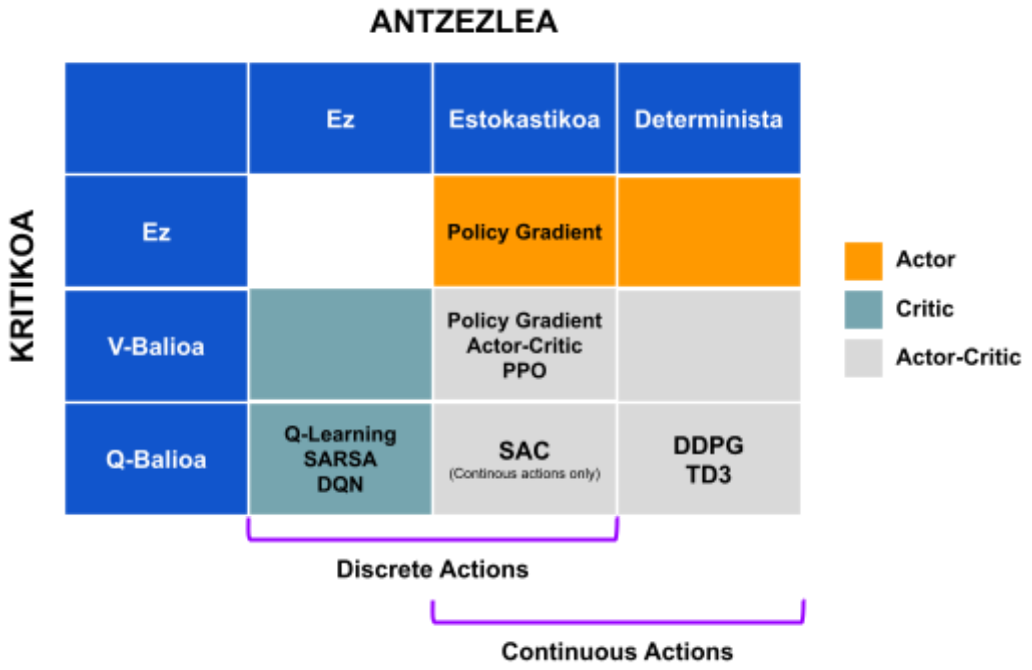
- DQN → Soilik ekintza diskretuekin lan egin dezake.
- TD3 → Soilik ekintza jarraituekin lan egin dezake.
- DDPG → Soilik ekintza jarraituekin lan egin dezake.
- SAC → Bai ekintza diskretu bai jarraituekin egin dezake lan.
- PPO → Bai ekintza diskretu bai jarraituekin egin dezake lan. (ia da baztertuta).

Gure ekintzak jarraituak dira, robota burmuina irekitzerakoan hartu ditzazkeen posizio balioak infinituak direlako.

Hau ikusita, soilik SAC, DDPG eta TD3 algoritmoak izango dira egokiak gure RL kontrola burutzeko.

2.2.4. Antzezle eta kritiko desberdinak.

Aukeraketa bukatzeko eta soilik algoritmo bat hautatzeko gure antzezle eta kritikoa aztertuko ditugu (ikus 3.1..atala honene azalpen bat edukitzeko). **9.Irudian** ikus dezakegu algoritmo guztiak antzezle eta kritikoen baitan bananduta.



9.Irudia: RL- algoritmoen bigarren sailkapena antzezle eta kritikoarekiko.

Alde batetik, gure antzezleak aukeratuko dituen ekintza abiadura izango da. Gure goiko limitea 0,1 m/s izango da eta behekoa 0 m/s. Aukeratu nahi dugun balioa ezaguna izango denez gure **antzezlea determinista** izango da.

Bestetik, gure kritikoaren balorazioa bi alde hartu ditzazke: ekintza baloratu edo egoera baloratu. Gure robotan garrantzia duen aldea, robotak duen abidura izango denez, gure kritikoa ekintza baloratzea aukeratuko dugu. Horregatik **Q-balio kritikoa** aukeratu dugu.

Hau jakinda hartu ditzazkegun algoritmo bakarrak DDPG eta TD3 dira.

2.2.5. Erantzuna.

Bi hoien artean **DDPG** ***(Deep Deterministic Policy Gradient)*** aukeratuko dugu eraginkorragoa delako, eta ia da gure arazoari hobeto moldatzen delako.

3.- DDPG *(Deep Deterministic Policy Gradient)* ALGORITMOA

Algoritmo honetan *"Actor-Critic"* metodologia (ikus **3.1.atala**) funtzionamenduan jartzen da. Agentea bitan banatzen da: **antzezlea eta kritikoa.**

Antzezlearen funtzioa egoera bat ekintza batean bilakatzea izango da, μ(s). Kritikoarena, aldiz, ekintzari balio numerioko bat ematea izango da (Q-balioa). Antzezlearen funtzioa kritioaren barnean sartzen badugu, Q-balio funtzioa honela geratzen da: Q(s,μ(s)).

Bi funtzio hauek betetzeko sare neuronalen (ikus **3.3.atala**) erabilpena inplementatuko da. Gure helburua bi hauen pisuak optimizatzea da, gure arazoaren konponbide hoberena aurkitzeko. Horretarako gure politikaren gradientea lortu behar dugu, pisuak ze norabidean eguneratu behar ditugun jakiteko.

DDPG-n politikak **deterministikoa** da, Q(s,μ(s)) funtzioaren gradienteak kritikoa kalkulatzen du, antzezleak aukeratutako ekintzen bitartez. Era honetan, sistema guztia gradientearen optimizazioarekin batera aldatzen da (ikus **2.ekuazioa**).

$$\nabla_s Q(s, \mu(s)) = \nabla_s Q(s, \mu(s)) \nabla_{\theta_\mu} \mu(s) \tag{2}$$

3.1. ACTOR-CRITIC METODOA

Actor-critic metodologia *value-based* eta *policy-seach* arteko nahasketa bat da.

Value-based metodoak Q-funtzioa optimizatzen du, antzezleak aukera ditzazkeen ekintzen artean banaketa bat ezartzeko. Estatu eta ekintza diskretuak dauden ingurune batean, Q-funtzioa erabiliz, balio bat eman ahal diogu egoera bakoitzean eman daitekeen egoera-ekintza pare guztiei (s,a). ***Q-Learning*** **algoritmoa** balio hoiek guztiekin taula bat eraikitzen eta optimizatzen du.

Hartutako erabakiaren balio-funtzioa edo **Q-funtzioa**, izena berak esaten duen moduan, erabaki baten balioa ematen du, politika batez deteminatuta, ingurumenaren egoera konkretu batean. Matematikoki honela adierazi dezakegu.

$$Q_\pi(s_t, a_t) = E_\pi(G_\pi | S_t = s, A_t = a) \tag{3}$$

Qπ matematikoki determinatzea oso zaila da, halere, hurbilketak egiteko moduak daude.

Policy-search*h metodologian** politika funtzio optimo bat bilatzen du zuzenean, balio funtzioa kalkulatu gabe. Hau da, sare neuronalak ez ditugu ekintza-egoera pareen balioak emateko entrenatzeko, baizik eta, egoera partikular baten egin beharreko ekintza aukeratzeko. *Policy-search* metodoen barnean ***policy-gradient submetodoa da erabiltzen den gehiena. Honetan sare neuronalaren aldagaiak gradientearen maldarekiko eguneratzen dira.

Egonkortasuna da *policy-gradient* metodoaren arazo larriena, espero ditugun probabilitateak eta sariaren (*"reward-aren"*) balioak aldakortasun handia dutelako.

Balio-funtzioan gradientea kalkulatzeko metodoa honakoa da:

$$\nabla_{\theta} U(\theta) \leftarrow \sum_{t=0}^{H} \nabla log \pi_{\theta}(a_t|s_t) G_t \tag{4}$$

Funtzioan ikus dezakegun moduan gradientearen menpekotasuna balio-funtzioaren erantzunen bata besteraekiko oso handia da. Efektu hau txikitzeko *"baseline"* kontzeptua atera zen. ***Baseline*** espero ditugun balio funtzioen batabesteko da.

Actor critic metodologian *baseline*-a ingurumenaren egoeraren menpe dago.

Egoera balio intrinseko bat badu (*baseline*), egoera-ekintzaren balioa **8.ekuazioan** isaldatzen den moduan emango da.

$$Q(a, s) = V(s) + A(s, a) \tag{5}$$

Kasu honetan, V(s) "*baseline"* balioa izango da eta A(a,s) trantsizioaren balioa emango digu.

V(s)-ren balioa hasiera batean ezezaguna da eta sare neuronalen bitartez kalkulatu dezakegu, honi **"Kritikoa"** deitzen zaio. Beste era batera esanda, kritikoak ingurua aurkitzen den egoerari balio bat ematen dio, V(s).

Ekintzak aukeratzen dituen bigarren sare neuronala **"Antzezlea"** deitzen da. Kritikoarekin batera, antzezlearen sare neuroalaren pisuen balioen aldaketa optimiza dezakegu, pisu hauek gure **politika** (π) sortuko dute.

Praktikan sare neuronal hauek, batera egiten dute lan (ikus **1.algoritmoa**).

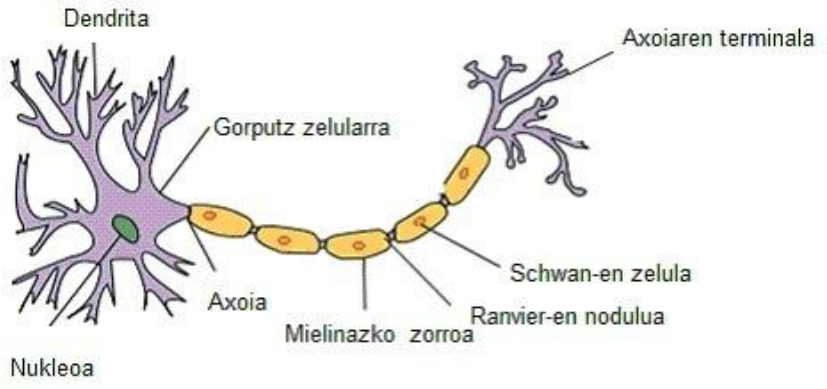
1.Algoritmoa: Actor-Critic

```
1.  Hasieratu programa.Sare neuronalaren balioak (pisuak) ausazkoak.
2.  while not done
3.        Play N steps in the environment with π0 saving epoch of (st,at,rt).
4.        if (done)
5.              R = 0
6.        else
7.              R=Vtheta(st)
8.        end
9.        Update backwards:
10.       for i = t-1…tstart
11.             R ← ri + γR
12.             Accumulate policy gradients.
13.             Accumulate value gradients.
14.             Update networks with gradients.
15.       end
16. end
17. return π
```

3.2. SARE NEURONALAK

Sare neuronaletaz lehen aldiz 60ko hamarkadan entzun zen Marvin Minsky eta Seymour Papert *"percepetron"* ideiari buruz egindako publikazio batean. Neuronaren portaera antzerazten duen elementu teorikoa da *"percepetron"* [24]. Halere, kontzeptua *Machine Learning* teknikak handitzen ari ziren momentu berean popularizatu zen

Neurona bat estimulatzean, dituen dentritetik egiten da (ikus **10.irudia**). Horretan tentsio igoera bat ematen da 35mV-ak gainditzen dituena. Dentritak, tentsio hori jasota, 69mv eta 80mv artean dagoen tentsio pultso bat eragiten du axonaren zehar. Axon-a beste neurona batzuen dentritekin konekatatuta dago.



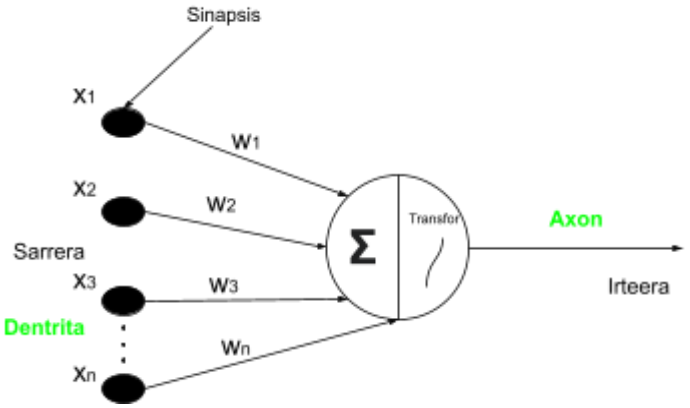
10.Irudia: Neurona baten atalak.

"*Percepetron*" elementua jokaera hau imitatzen du **aktibazio funtzio** baten bidez. Funtzio bat baino gehiago daude jokaera hau errepresentatzeko, "sigmoide" funtzioa (**6.ekuazioa**), aldiz, garrantzitsuenetariko bat da:

$$sigm(x) = \frac{1}{1+e^{-x}} \tag{6}$$

Perceptron-ak neurona-birtualak deitu daitezke. Neurona birtual hauek geruzetan multzokatzen dira, non, normalean geruza baten neurona guztiak hurrengoaren neurona guztiekin konektatzen dira. Konexio mota honi **"MLP"** ***"Multy Later Perceptron"*** (Ikus **12.irudia**) deitzen zaio.

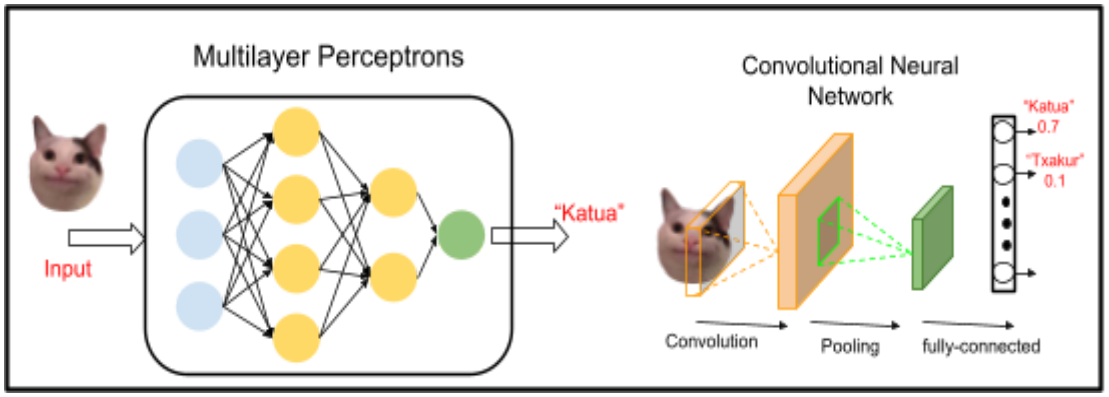
Konexio bakoitza zenbaki bat dauka, **pisua** deitzen dena. Ondorioz, "*perceptron*" baten aktibazio funtzioa, aurreko "*perceptron*" guztien balioak bider sartzen den konexioaren pisua eta guzti hauen gehiketa da (ikus **11.irudia** eta **8.ekuazioa**).



11.Irudia: "Perceptron" baten eredua, neurona batekin antzekotasuna antemantzeko.

$$AF = \sum_{t=1}^{n} (x_t * w_t) \quad (8)$$

Machine Learning-en gure agentea ikasi dezan, sare neuronal hauen pisuak aldatu behar dira, hau egiteko erabiltzen den teknika *"backpropagation*" deitzen da. Teknika honetan aktibazio funtzioaren deribatuak kalkultu behar dira, ondoren, katearen araua aplikatuz, hurrengo deribatuak kalkulatzeko, amaieran, pisuak pausu oro eguneratzeko. Zorionez, guk erabilitako software-an (Matlab-en) teknika hau automatikoki ejekutatzen da. Programadorea egin beharreko ataza bakarrak: sare neuronal mota aukeratzea, haren arkitektura eaikitzea eta geruzak aukeratzea izanda.



12.Irudia: MLP eta CNN arteko desberdintasunak.

Bi familia desberdinetan sailkatu ditzazkegu sare neuronalak (**12.irudia**): MLP *(MultiLayer Perceptron)*, funtzioak edo egoera bektorialak arazoa definitzen dituzten egoeratan gehien erabiltzen direnak, edo CNN *(Convolutional Neuron Network)*, normalean argazkiak sailkatzeko edo klasifikatzeko erabiltzen direnak.

4. INGURUMENA ETA AGENTEAREN SORKETA.

Ariketa osoa Matlab-en programatu da. Egin beharreko lehenengo gauza ingurua eta antzezlea eraikitzea izango da. Bi hauek bai simulink barruan, bai kodearekin sortu beharko ditugu. Hurrengo azalpena, sofware-aren oinarrizko ezagutza bat izateko emango da.

4.1. KODEA MATLAB-EN

4.1.1. Ingurua.

Ingurumena sortzeko egin beharreko gauza garrantzitsuena, **behaketen eta ekintzen informazioa** sortzea da. Gure kasuan agentea bi behaketa, indarra eta posizioa, jasoko ditu eta ekintza bakar bat, abiadura, emango du.

Behaketek eta ekintzek balio infinituak jaso ditzazkete, ondorioz, "*rlNumericSpec()"* [25] funtzioa erabiliko dugu. Funtzio honetan aldagaien kantitatea bektore moduan adierazten da, esaterako, behaketak bi direnez, bi balio dituen bektore bakarra ([2 1]) sartuko dugu funtzioaren barnealdean (ikus **2.algoritmoa**).

Honekin batera, aldagai asko sortuko ditugu, **2.taulan** ikus dezakegun moduan. Aldagai guzti hauek kodean sortu arren, simulink-en dute garrantzia.

2.Taula: Sortu beharreko aldagai guztiak.

Aldagaia	Deskribapena	Balioa
λ_1	Sari funtzioan distantziarem garrantzia kontrolatzen du.	0.9
λ_2	Sari funtzioan abiadura garrantzia kontrolatzen du	0.2
λ_3	Sari funtzioan indarraren garrantzia kontrolatzen du.	0.2
Ogden Model Data	Deformazio indarra kalkulatzeko erabiltzen da.	.mat
$x_{desired}$	Lortu nahi dugun distantzia burmuin tolesduren artean.	0.1
T_s	Simulazioa pausu bakoitzaren denbora.	0.02
$X_{theshold}$	Distantzia maximoa izan dezakeen errorea.	0.01
Penalty	Simulazio osoan, helburua lortu ez bada, ezarritako zigorra.	-15

Ingurua sortzeko, sortu ditugun behaketa eta ekintza informazio guztiaren laguntzarekin, "*rlSimulinkEnv()"* [26] funtzioa erabili dugu. Inguruarekin ia da *"ResetFcn"* funtzioa, simulazio guztiak hasterakoan ejekutatzen diren lehenengo kode

lerroak, dator. Guk sortutako *"cleanstart"* funtzioaz ordeztu dugu (Ikus **3.algoritmoa**).

2.Algoritmoa: Inguruaren sorketa.

```
1. prozedura
2. Behaketa informazioa ← Bi behaketa: indarra eta posizioa.
3. Ekintza informazioa ← Ekintza bakarra: Abiadura.
4. λ1 ← 0.9
5. λ2 ← 0.2
6. λ3 ← 0.2
8. OgdenModelData ← load('OgdenModelData')
9. xdesired ← 0.1
10. Ts ← 0.02
11. Λx ← 0.01
12. Penalty← -15
13. env ←Ingurumena sortu behaketa eta ekintza informazioarekin..
14. env ← @cleanstart. ResetFcn funtzioa ordezkatu
```

3.Algoritmoa: Cleanstart funtzioa.

```
1. funtzioa cleanstar(in)
2. x ← 0
3. F ← 0
4. xaurrekoa ← 0
5. ẋ← 0
6. bueltan x,F, xaurrekoa, ẋ
```

4.1.2. Agentea.

Agentea sotzeko alde batetik kritikoa eta bestetik antzezlea sortu beharko dugu.

Bai kritikoa bai antzezlea sortzeko egin beharreko lehenengo gauza, sare neuronalak sortzea izango da (Ikus **5.0.2.atala**). Sare neuronal hoberenak aukeratzeko hainbat saiakerak egin beharko ditugu, 5.0.2.atalean ikus dezakegun moduan.

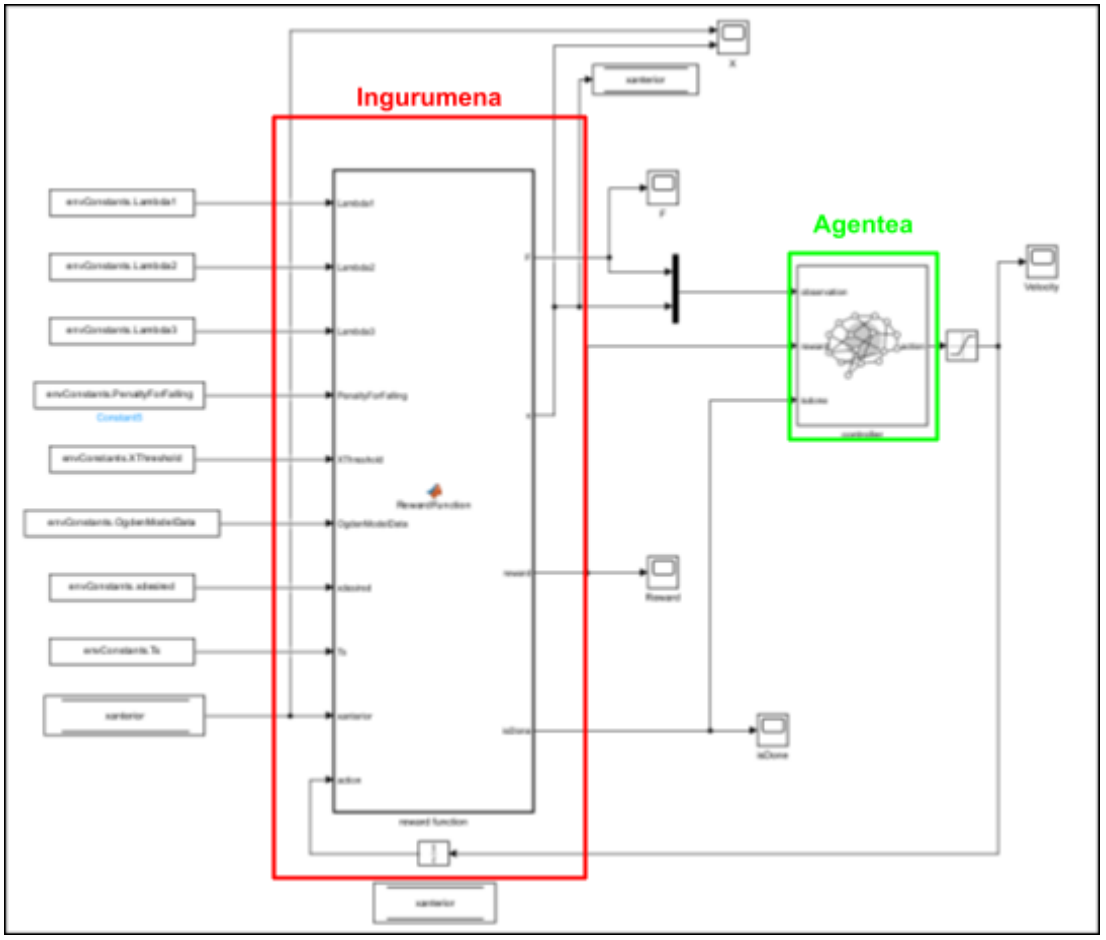
Ia da sare neuronalak sortuta, antzezlea *"rlContinousDeterministicActor()"* [27] funtzioarekin sortuko dugu eta kritikoa *"rlQValueFuction()"* [28] funtzioarekin eraikiko dugu.

Bi hauek agente bakar batean batzeko, *"rlDDPGAgent()"* [29] funtzioa erabiliko dugu.

Funtzio guzti hauen barnean aukerak egongo dira, guk aurrerago optimizatuko ditugunak, gure ataza sortzeko hoberenak lortu harte (Ikus **5.0.1.atala**).

4.2. SIMULINK EREDUA.

Simulink eredua barnealdea nola ikusten den, **13.irudian** ikus dezakegu.



13.Irudia: Simulink eredua.

4.2.1. Ingurua.

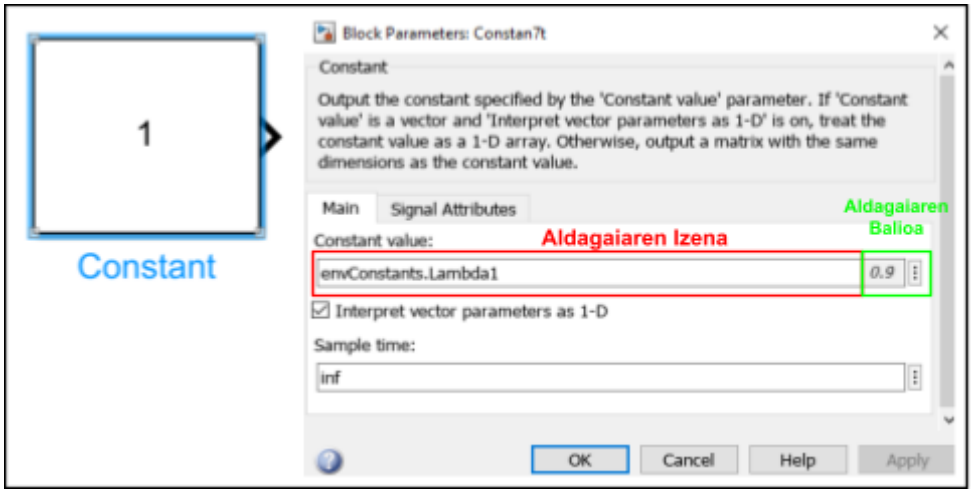
Ingurua sortzeko, askotan, simulink-en blokeen konbinazioak erabiltzen dira, halere, metodo hau ez da dagoen bakarra. Gure kasuan, gure Malab-en ezagutza simulink-ena handiagoa denez, Matlab funtzio bat simulink barnean sortu dugu, *"MATLAB Function"* blokea erabiliz (ikus **14.irudia**). Honen barnean **behaketak (posizioa eta indarra) eta sari futzioa** kalkulatzen dira, **bukaerako baldintzekin** batera (ikus **4.algoritmoa**).

14.Irudia: *"MATLAB Function"* blokearen eredua.

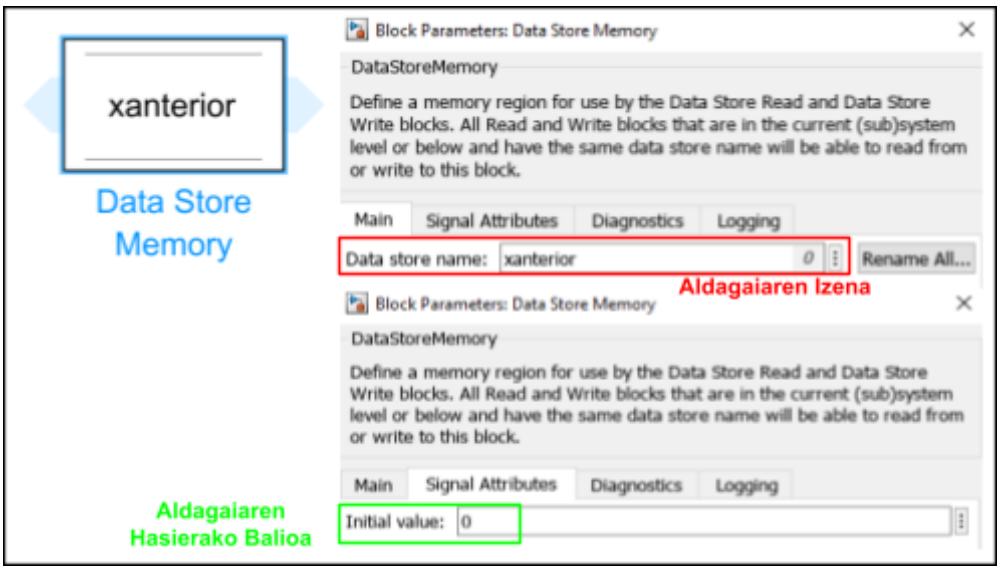
4.Algoritmoa: Behaketen eta sari funtzioaren kalkuluak "MATLAB Function" blokearen barnean.

1: **funtzioa** RewardFunction($\lambda_1, \lambda_2, \lambda_3, Penalty, \Lambda_x, OgdenModelData, T_s, x_{ant}, v, x_{desiratua}$)
2: $\dot{x} \leftarrow v$
3: $x \leftarrow x_{ant} + \dot{x} * T_s$
4: $F_0 \leftarrow$ Interpolazioa(OgdenModelData,0)
5: $F \leftarrow$ Interpolazioa(OgdenModelData,x)
6: $F \leftarrow F - F_0$
7: Bukaerako baldintzen konprobaketa. (**5.Algoritmoa** ikusi)
8: IsDone ← 0 or 1
9: Hartutako ekintza dakarren saria kalkulatu. (**6.Algoritmoa** ikusi)
10: Saria ← Balioa(F,x,$\dot{x}$,IsDone,Penalty)
11: **bueltan** x, F, Saria

Horretaz gain, kodean sortutako aldagaiak simulink ereduan sartzeko *"constant"* blokea erabili dugu (ikus **15.irudia**), barneko aukeretan aldagaiaren izena jarriz. Ez hori bakarrik, *"Data Store Memory"* blokearen laguntzarekin posizioaren jarraipena egiteko ahalmena izan dugu (ikus **16.irudia**).



15.Irudia: *"Constant"* blokearen eredua.



16.Irudia: *"Data Store Memory"* blokearen eredua.

4.2.1.1- Bukaerako baldintzak:

Bukaerako baldintzetaz hitz egiten dugunean, bi gauzei egiten diogu erreferentzia. Alde batetik, simulazioa bukatu daiteke gure agentea debekatuta dagoen bide bat hartu duelako. Gure simulazioan horrelako hiru baldintza egongo dira:

1. Agenteak burmuina gu nahi genuena baino gehiago ireki du.
2. Posizioaren aldagaia balio negatiboak hartu ditu. Hau errealitatean ezinezkoa izango zen.
3. Burmuina irekitzen dugun bitartean, robota burmuina ixten hasi da.

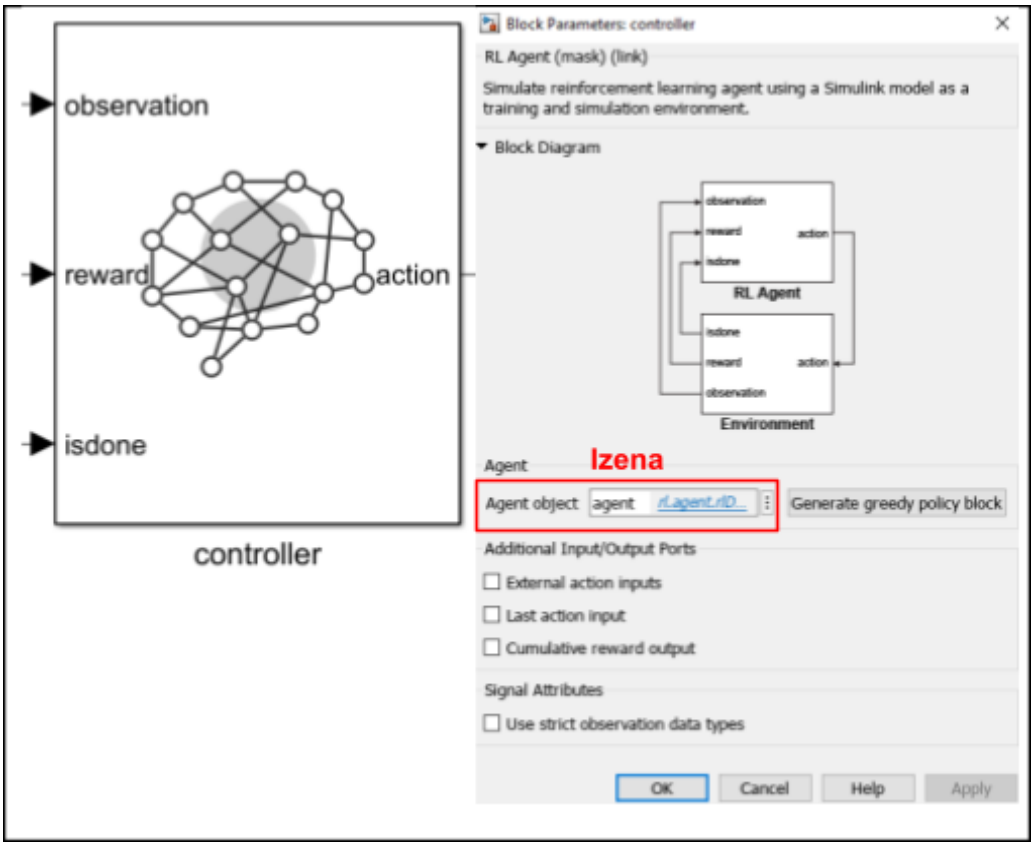
Bestetik, simulazioa bukatu daiteke agentea helburua lortu duenean, kasu honetan ez zaio zigorrik ematen eta agentea premiatzen da. Baina, gure simulazioan hau ez da implementatuko; Azken batean, agenteak burmuina desiratutako posizioan irekita mantendu beharko du, ez soilik iritsi.

5.Algoritmoa: Bukaerako baldintzak.

1: **prosezua**
2: **if** x > xdesiratua+ XThreshold ⟶ Agentea burmuina gu nahi genuena baino gehiago irekit du.
3: isDone ⟵ 1;
4: **else if** x<0 ⟶ Posizioaren aldagaia balio negatiboak hartu ditu.
5: isDone ⟵ 1;
6: **else if** x-xaurrekoa<0 ⟶ Robota burmuina ixten hasi da.
7: isDone ⟵ 1;
8: **else**
9: isDone ⟵ 0;
10:**endif**

4.2.2. Antzezlea.

Antzezlea egiteko MatLab-en soilik bloke bakar bat erabili behar dugu, *"controller"* blokea. Ikus **17.irudia** blokearen irudia ikusteko. Oso garrantzitsua da blokearen aukeretan lehen Matlab kodean sortutako agentearen izena jartzea, simulatzerakoan bi hauek erraz erlazionatu daitezen.



17.Irudia: *"controller"* blokearen eredua.

5. SAIAKERAK ETA OPTIMIZAZIOA

Gure arazoari irtenbide egokia aurkitzeko asmoarekin eta ahal dugun entrenamendu zehatzena lortzeko ideiarekin hainbat saiakera egin ditugu. Gure programazioaren hiru atal modifikatu dira. Atal hoiek Matlab aukerak, sare neuronalak eta sari *(reward)* funtzioa dira.

Halere, ezinezkoa da banaka banaka bat aukeratzea; Izan ere, entrenamendua futzionatzeko guztiak behar ditugu. Ondorioz, aukera hoberenak ez badira ere guztien oinarri bat ezarri dugu, honetik aurrera hobetzeko.

5.0. OINARRIA.

5.0.1. MatLab aukerak

MatLab aukeretan gure oinarrizko programa honetan, aukera guztien balio lehenetsiak hutsi ditugu. Askotan, ataz ez oso konplexuetarako soilik aukera hauekin agenteren entrenamendua eta funtzionamendua egokia delako.

3.Taulan ikusi daitezke aukerak, hoien deskribapena eta balio lehenetsiak kritikoa eta antzezlearen kasuan. **4.Taulan**, aldiz, agentearen aukerak ikustarazi ditzazkegu. Bukatzeko **5.taulan** entrenamenduaren aukerak agertzen dira.

3.Taula: *"rlOptimizerOptions()"* [30] funtzioa erabilita, antzezle eta kritikoaren aukera garrantzitsuenak.

Izena	Deskribapena	Balioa
Learn Rate - Ikaskuntza tasa	Aktorearen edo kritikaren funtzioaren hurbiltzailearen prestakuntzan erabilitako ikaskuntza-tasa, eskalar positibo gisa zehaztua.	0.01
Gradient Threshold - Atalase gradientea	Atalase gradientearen balioa, aktorearen entrenamenduan edo funtzio kritikoaren hurbilketan erabiltzen dena, Inf edo eskalar positibo gisa zehaztua.	Inf
Gradien Threshold Method - Atalase gradientearen metodoa	Atalase gradientea gainditzen duten balio gradienteak klikatzeko erabilitako metodo espezifikoa.	12norm
L2 Regularization Factor - L2 regulazio faktoretza.	Gaindoikuntza murrizteko modua.	0.0001
Algorithm - Algoritmoa	Algoritmoa aktorearen edo funtzio kritikoaren hurbiltzailea trebatzeko erabiltzen da.	"adam"

4.Taula: *"rlDDPGAgentOptions()"* [31] funtzioa erabilita, agentearen aukera garrantzitsuenak.

Izena	Deskribapena	Balioa
Noise Options - Zarata aukerak	Zarata modeloaren aukerak, OrnsteinUhlenbeckActionNoise objektu gisa zehaztuak.	Ikus [31]
Actor optimizer options - Aktore optimizatzailearen aukerak	Antzezlearen aukerak sartzen dira aukera honetan, kasu honetan predeterminatuak egondo dira	Pred
Critic optimizer options - Kritiko optimizatzailearen aukerak	Kritikoaren aukerak sartzen dira aukera honetan, kasu honetan predeterminatuak egondo dira	Pred
Mini Batch Size - Mini sortaren tamaina	Auzasko experientzia sortaren tamaina	64
Num Steps To Look Ahead - Predikzio kantitatea	Politikaren balioa kalkulatzeko erabili diren etorkizuneko sarien kopurua	1
Experience Buffer Length - Gordetako Experientzia Kantitatea	Ondoren, politikaren balioa eguneratzeko erabiltzen aukeratu ditzazkegun experientzia kantitatea.	10000
Sample TIme - Lagin-denbora.	Simulazioaren *Step* bakoitza iraungo duen denbora.	1
Discount Factor - Deskontu-faktorea	Deskontu-faktorea etorkizuneko sariei aplikatzen zaie entrenamenduan zehar, 1 edo txikiagoa den eskalar positibo gisa zehaztua.	0.99

5.Taula: *"rlTrainingOptions()"* [32] funtzioa erabilita, entrenamendu aukera garrantzitsuenak:

Izena	Deskribapena	Balioa
Max Episodes - Pasarte maximoak	Agentea trebatzeko pasarte/kapitulu kopuru maximoa.	500
Max Steps Per Episode - Pausu kantitate maximoa	Egon ditzazkeen pausu kantitate maximoa pasarte/kapitulu bakoitzean. Pausu denbora, Lagin-denboraren berdina da.	Pred
ScoreAveragingWindowLength - Bataz bestekoak kalkulatzeko leihoa.	Bataz bestekoa kalkulatzeko erabltzen diren kapitulu kantitatea	5
Stop Training Criteria - Entrenamendua bertan behera geratzeko arrazoia.	Aukera desberdina entrenamendua bukatzeko.	*Average Steps*
Stop Training Value - Entrenamendua gelditzeko balioa.	Aurreko aukera ze balio eduki behar duen entrenamendua bertan behera uzteko.	500
Save Agent Criteria - Agentea gordetzeko arrazoia..	Aukera desberdina agentearen egoera gordetzeko.	*none*

Save Agent Value - Agentea gordetzeko balioa.	Aurreko aukera ze balio eduki behar duen agentea gordetzeko.	none
Plots - Grafikak	Entrenamendua nola dijoan ikusteko aukera edo ezer ez ikusteko aukera.	*"training-progress"*

5.0.2. Sare neuronalak

Eraiki ditugun bi sare neuronalak MLP motakoak dira. Hauek eraikin partikular bat daukate, normalean, guztiz konektatutako geruzataz (*fully connected layers*) osatuta daude eta hauen artean relu geruzak (*relu layers*) jartzen dira.

Guztiz konetatutako geruzak neurona kantitate partikular bat dute. Hauek, lehen aipatu dugun moduan, pisuak eta sesgoak dituzte. Geruza hauen neurona bakoitzak hurrengoaren eta aurrekoaren neurona guztiekin konektatzen dira, funtzio luze eta egoki bat sortzeko.

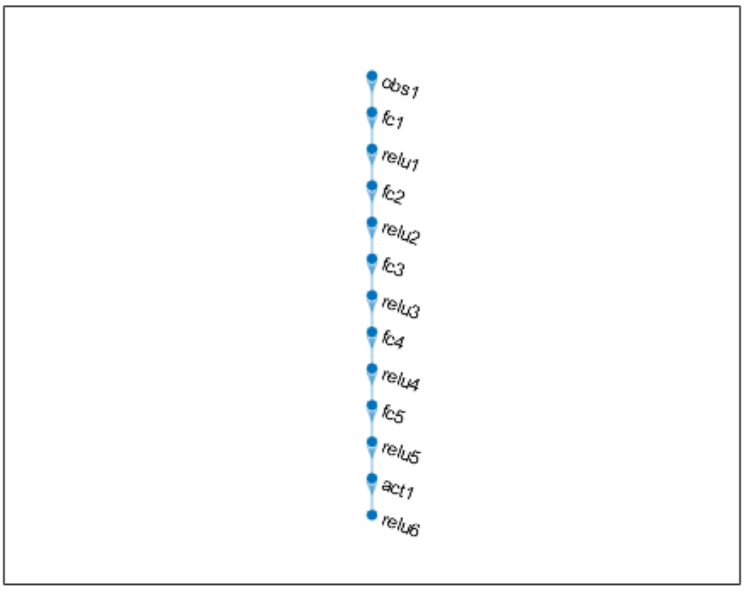
Relu geruzak informazioa simplifikatzen dute balio negatiboak ezabatuz. Honela simulazioa azkarragoa da.

Matlab-en sare neuronal bat sortzeko nahi ditugun geruza guztien izendapen bat egin behar dugu, ia da jarri nahi ditugun ordenean. Geruza guztiek guk zehazten ditugun izen eta aldagai desberdinak dituzte. Ikus [33] guk erabilitako geruzen informazio gehiago irakurtzeko.

Hasierako geruza sarrera geruza bat (*Input layers*) izango da, hau da, sarrera kantitatea zehaztu beharko dugu. Gure kasuan, behaketak geruzan sartzen diren aldagaiak badira, bi sarrera egongo dira eta ekintza bada sartzen den aldagaia, soilik sarrera bakarra izango dugu.

5.0.2.1. Antzezlearen oinarrizko sare neuronala:

Antzezlearen oinarrizko sare neuronala **sei guztiz konektutako geruzataz eta sei relu geruzataz** osatu dugu. Horretaz gain, sarrera geruza bat jarri dugu sare neuronalaren aurrekaldean (ikusi **18.irudia**).

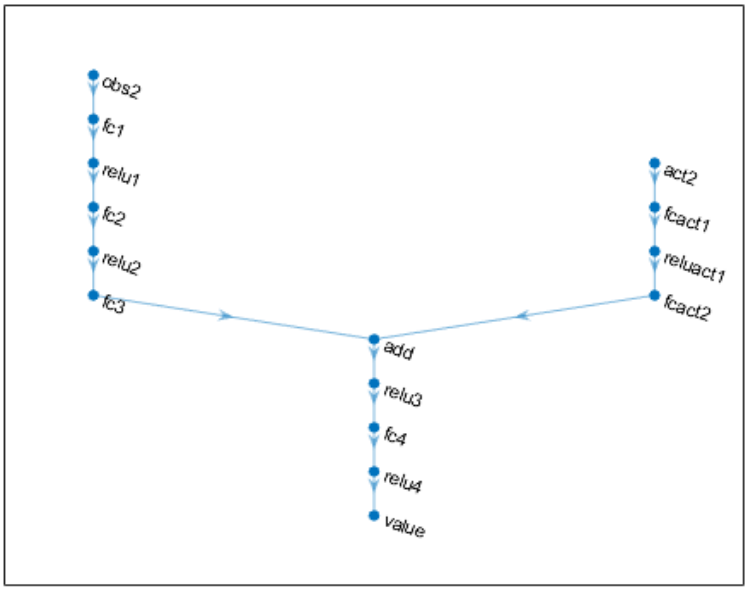


18.Irudia: Antzezlearen sare neuronala.

Sare neuronalaren geruzak ikusteko, *"layerGraph()"* [34] funtzioa erabili dugu, honela **18.irudia** atera da.

5.0.2.2. Kritikoaren oinarrizko sare neuronala:

Kritikoaren oinarrizko sare neuronala **sei guzti konektutako geruzataz eta bost relu geruzataz** osatu dugu. Horretaz gain, sarrera geruza bi jarri ditugu; Izan ere, kritikoa ekintza eta behaketak aztertuko ditu (ikusi **19.irudia**).



19. Irudia: Kritikoaren sare neuronala [34].

5.0.3. Sari (*reward*) funtzioa.

Sari ekuazioa hurrengoa izango da:

$$R = -\lambda_1 \cdot (x_{desired} - x)^2 \cdot F^2 + \lambda_1 \cdot \lambda_3 \cdot (x_{desired} - x)^2 + \lambda_2 \cdot \dot{x} + Penalty \qquad (3)$$

Funtzio honetan ikus dezakegun moduan, abiaduraren eta indarraren kontrola izango dugu. Indarra, beti posizioaren baitan egongo denez, posizio errorearekin batera ezarri da. Bukatzeko, *"Penalty"* aldagaia soilik bukaerako baldintza bat betetzen denean jarriko da, zigor moduan.

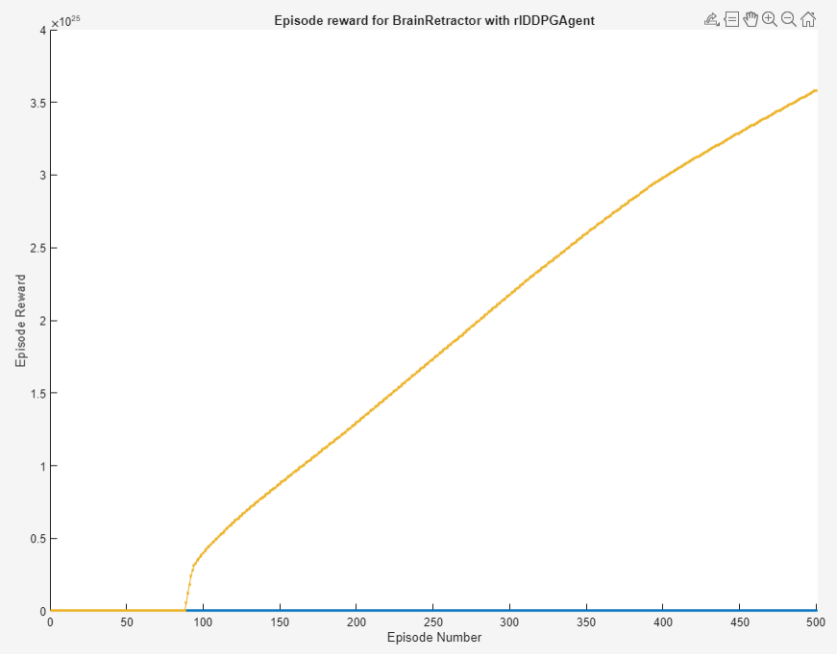
Hasiera batean, gure helburu handiena desiratutako posiziora heltzea denez lamden balioak **6.taulan** agertzen direnak dira.

6.Taula: Lamden oinarrizko balioak, ondoren aldatuko ditugunak.

Lamda 1 (λ_1)	0.9
Lamda 2 (λ_2)	0.2
Lamda 3 (λ_3)	0.2

5.0.4. Erantzunak.

Entrenamendua egin ondoren **20.irudian** ikus daitekeen moduan, ez dugu erantzun onik izan, erantzun sub-optimo batean ostopatuta geratzen da.



20.Irudia: Entrenamenduaren grafika oinarrizko aukerekin.

5.1- MatLab aukeren hobekuntza eta saiakerak.

Lortu nahi dugun lehenengo helburua burmuina nahi dugun posiziora irekitzea da. Horretaz gain, behin irekita agentea garuna irekita mantendu beharko du. Horretarako aldatu behar dugun lehenengo gauza antzezle, kritiko eta agentearen aukerak dira, ez sari funtzioa (ia da garrantzi handiena distantzia duelako) ez sare neuronalak (suposatu dugulako nahiko handia dela helburua lortzeko).

Lehenengo, entrenamenduaren aukerak ezarri ditugu, **20.irudia** ikusita aldaketa handiak behar zirela ikusi dugulako. Saiakera askoren ostean, **7.taulan** ematen diren balioak nahiko onak direla ikusi ditugu.

7.Taula: Entrenamendu aukera berriak.

Izena	Arrazoia	Balioa
Max Episodes - Pasarte maximoak	Kapitulu kantitate handiago bat behar genuen, gehiegizko kapituluak badaude ere, guk ez dugu guztiak erabiliko.	3000
Max Steps Per Episode - Pausu kantitate maximoa	Lagin denbora 0.02 izanez, 4s-tan kontsideratzen dugu gure robota helurua lortzeko denbora izan duela.	200
ScoreAveragingWindowLength - Bataz bestekoak kalkulatzeko leihoa.	Bataz bestekoak ateratzeko, kapitulu kantitatea handiagoa denez, hau berdin handitu behar dugula kontsideratu dugu.	50
Stop Training Criteria - Entrenamendua bertan behera geratzeko arrazoia.	Sari funtzioa du garrantzi handiena gure simulazioetan, ondorioz kalkulatutako batabestekoa horretan oinarrituko da.	*Average Reward*
Stop Training Value - Entrenamendua gelditzeko balioa.	Ez du garrantzirik izango, oraindik entrenamendua nola dijoan ikusi behar dugulako.	$1e^{-3}$
Save Agent Criteria - Agentea gordetzeko arrazoia..	Sari funtzioa du garrantzi handiena gure simulazioetan, ondorioz kalkulatutako batabestekoa horretan oinarrituko da.	*Episode Reward*
Save Agent Value - Agentea gordetzeko balioa.	Ez du garrantzirik izango, oraindik entrenamendua nola dijoan ikusi behar dugulako. Halere, balio minimo bat ezarriko dugu, agenteak gure minimoa lortzen duen ikusteko.	-0.5

Aurrerago, agentearen aukerak modifikatu ditugu. **8.Taulan** ikus daitekeen moduan ez ditugu aukerak asko eraldatu; Izan ere, ia da zeudenak nahiko onak ziren.

8.Taula: Agentearen aukera berriak

Izena	Arrazoia	Balioa
Noise Options.Standard Deviation	[31] ikusi daitekeen moduan, StandardDeviation*sqrt(Ts) ekuazioa hartu dugun ekintzaren balio-tartearen %1-%10 izatea gomendatzen da.	0.035
Noise Options.Variance Decay Rate	Antzezlea gero eta aukera desberdin gutxiago hartzea nahi dugu.	$1e^{-5}$
SampleTime	Hasiera batean egoki irudi zaigun lagin-denbora ezarriko dugu.	0.02
Mini Batch Size - Mini sortaren tamaina	Aurreko balio gehiegizkoa dela suposatu dugu, eta honek ahalmen komputazional handia eskatzen du.	128
Experience Buffer Length - Gordetako Experientzia Kantitatea	Experientzia balio handiagoa hartuko ditugu oraingoz.	$1e^{6}$

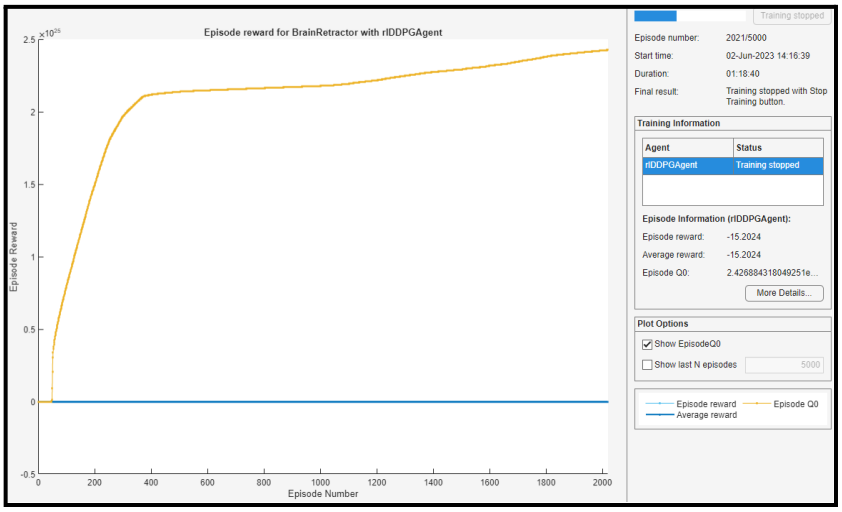
(Agentearen eta entrenamenduaren aukerak behin betiko bertsioan ez genituen asko aldatu. Halere, ikus "A. Eranskina" amaieran izandako aukera guztiak ikusteko.)

Ondoren, kritikoaren eta antzezlearen aukerak eraldatu ditugu. Prosezu hau luzeagoa izan da **abiadura ikasketa** (*"Learn Rate"*) zehazteko hainbat simulazio egin ditugalako.

Abiadura ikasketa handiegia bada simulazioak erantzun sub-optimoak aurkitu ditzazke, baina txikiegia bada denbora gehiegi egon daiteke erantzuna bilatzen edo inoiz ez aurkitu. Ondorioz, guk hurrengo prozedura eraman dugu:

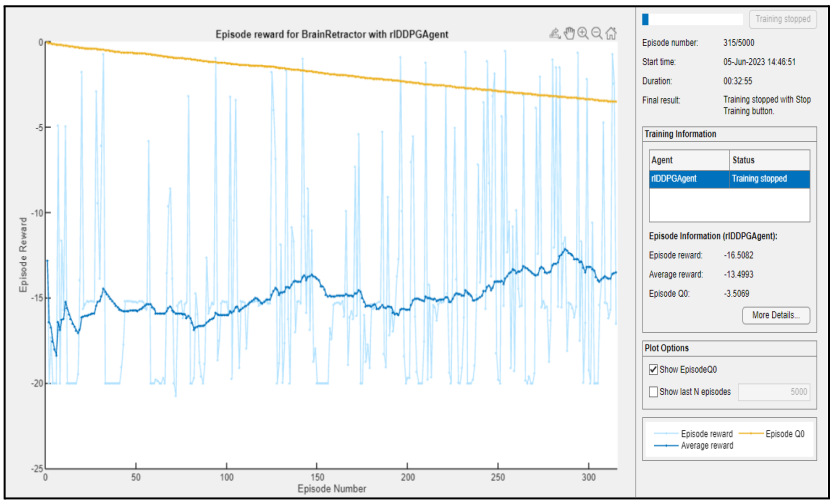
- Ikasketa abiaduraren balio lehenetsiarekin hasi gara.
- Ikasketa abiaduraren balioa handiaegia bazen, hau jaitsi dugu zati hamar eginez.
- Ikasketa abiadura, aldiz, txikiegia bazen, hau igo dugu aurreko ikasketa abiadura eta honen artean ezartzeko.

21.Irudian ikus daitekeen moduan, "*LearnRate"* balioa handiegia denean gure robota hasieran proba batzuk egiten ditu, baina behin ikustia robota ahal zen guztia irekiz lortutako sariaren balioa hain txarra ez zela, erantzun horretan buklean geratu da. Erantzun hau irtenbide suboptimo bat kontsideratu dezakegu.



21.Irudia: Entrenamenduaren grafika abiadura ikasketa balio handiegiarekin.

22.Irudian aldiz, agentea ez du abiadura normal batean ikasten, *"LearRate"*-aren balio txikiegia delako.

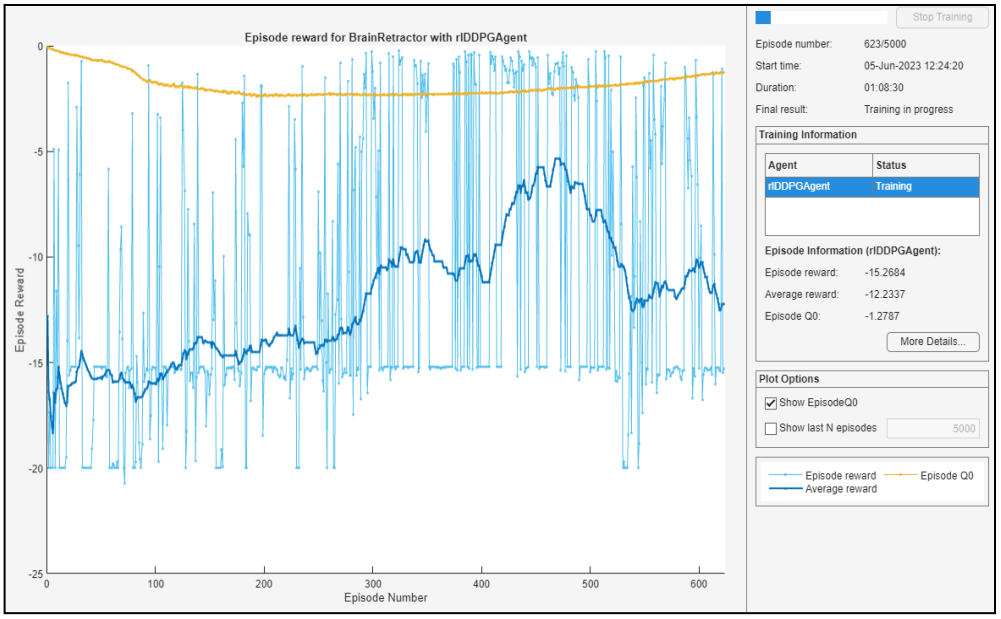


22.Irudia: Entrenamenduaren grafika abiadura ikasketa balio handiarekin.

Amaieran, ikaste abiadura eta beste aukerak **9.taulan** agertzen diren balioak dituzte, entrenamendua **23.irudian** ikusten den moduan ematen delako. Irudian ikus dezakegu ikasketa noranzko egoki bat hartzen duela (goranzko norabidea), denbora tarte egoki batean.

9.Taula: Antzezle eta kritikoaren aukera berriak.

Antzezlea.	Learn Rate - Ikaskuntza tasa	$1e^{-5}$
	Gradient Threshold - Atalase gradientea	1
Kritikoa.	Learn Rate - Ikaskuntza tasa	$1e^{-4}$
	Gradient Threshold - Atalase gradientea	1



23.Irudia: Entrenamenduaren grafika abiadura ikasketa egoki batekin.

5.2. Sare Neuronalak eta guk izandako saihakerak.

Ia da lehenengo helburua lortuta, entrenamendu denbora murrizten saiatu gara. Horretarako, sare neuronalei aldaketa egin diegu.

Sei saiakerta egin ditugu. Alde batetik, sare neuronalen geruza kantitatea murriztu ditugu, sei geruza gehiegizkoak zirela uste genuelako. Bestetik, geruza bakoitzean zeuden neurona kopurua eraldatu ditugu, 100 neurona seguraski gehiegi delako gure helburua lortzeko eta entrenamenduari denbora gehitzen diolako.

Bi aldaketa hauen konbinazio guztiak egin ondoren **10 eta 11.tauletan** laburbilduta daude jasotako erantzunak.

10.Taula: Antzezle eta kritikoaren sare neuronalak 6 guztiz konektatutako geruzekin eta neurona kantitatea aldatuz (ikus “C Eranskina” irudiak ikusteko).

Neurona Kantitatea	Helburua lortu?	500 kapitulu denbora?
100	Bai.Goranzko tendentzia txkiia	1 ordu.
50	Bai. Goranzko tendentzia handia.	35 minutu.
10	Ezi	20 minutu.

11.Taula: Antzezle eta kritikoaren sare neuronalak 3 guztiz konektatutako geruzekin eta neurona kantitatea aldatuz (ikua “C Eranskina” irudiak ikusteko).

Neurona Kantitatea	Helburua lortu?	500 kapitulu denbora?
100	Bai. Goranzko tendentzia txikia.	40 minutu.
50	Bai. Goranzko tendentzia handiena du.	25minutu.
10	Ez.	20 minutru.

10 eta 11.tauletan ikusten den moduan, hoberen jardun duen sare neuroanala 3 geruza dituena da, bakoitzan 50 neuronez osatua. Gainera, denbora egokia egon da kapitulu bakoitzean, RL arazo bati aurre egiteko.

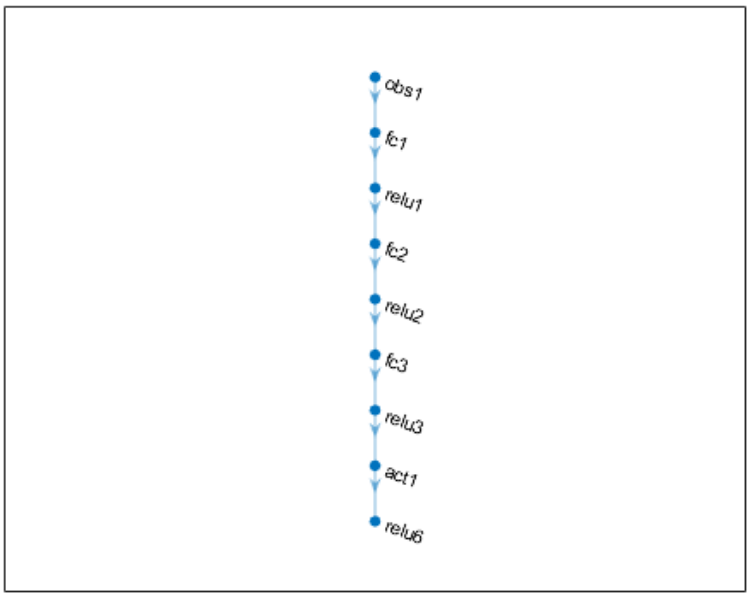
Horregatik, hemendik aurrera erabiliko dugun sare neuronalak hauek dira.

5.2.1. Antzezlearen behin betiko sare neuronala.

- Sarrera geruza *(Input Layer)*: Datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da. Bi neurona izango ditu.
- Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 50 neurona dela ikusi dugu. Konexio geruza guztiak, barnean dituen pisuekin, gure robotak egingo dituen indarrak kontrolatuko dute.
- Lehenengo ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu, funtzionamendua azkarrago izateko.
- Bigarren konexio geruza *(Fully Connected Layer)*: 50 neurona.
- Bigarren ReLu geruza *(ReLu Layer)*.

- Hirugarren konexio geruza *(Fully Connected Layer)*: 50 neurona.
- Hirugarren ReLu geruza *(ReLu Layer)*.
- Azkenengo konexio geruza *(Fully Connected Layer)*: Neurona bakarra. Neurona kantitatea ekintzak izango dituen aldagaien kantitateari berdina izango da. Gure kasuan abiadrua da kontrolatu dezakeguna.
- Amaierako ReLu geruza *(ReLu Layer)*.

Geruza bakoitzaren informazio zehatza ikusteko joan [33]. Antzezlearen sare neuronalaren irudikapena **24.irudian** ematen da.



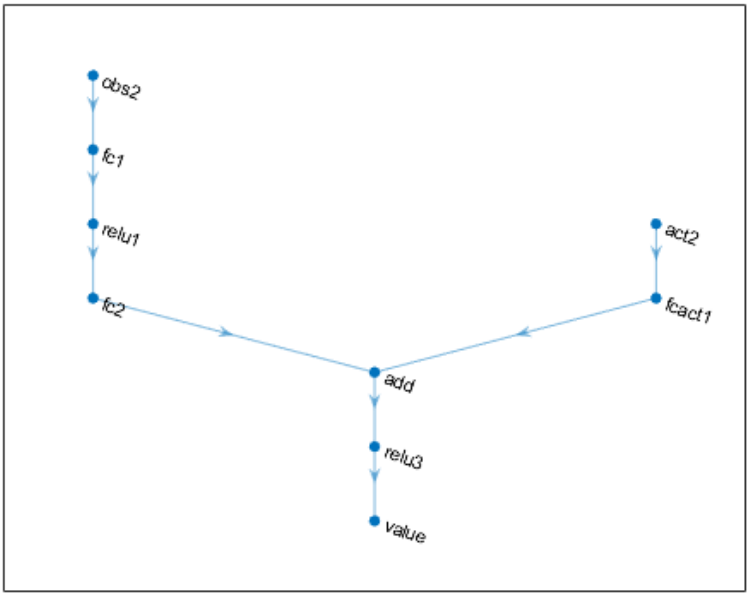
24.Irudia: Antzezlearen behin betiko sare neuronalaren irudikapena.

5.2.2. Kritikoaren behin betiko sare neuronala.

- Lehenengo sarrera ilara (behaketak):
    - Sarrera geruza *(Input Layer)*: Behaketa datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da, 2 behaketa ditugunez 2 neurona izango ditugu.
    - Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 50 neurona dela ikusi dugu.
    - ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu.
    - Bigarren konexio geruza *(Fully Connected Layer)*: 50 neuronekin.

- Bigarren sarrera ilarau (ekintzak):
    - Sarrera geruza *(Input Layer)*: Ekintza datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da. Ekintza bakarra ditugunez neurona bat izango ditugu.
    - Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 100 neurona dela ikusi dugu.
- Irteera ilera bakarra:
    - Batuketa geruza *(Additon Layer)*: Bi ilarak batzen ditu.
    - ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu.
    - Azkenengo konexio geruza *(Fully Connected Layer)*: Neurona bakarra. Hemendik Q-balio funtzioa antzezlera eramango da.

Geruza bakoitzaren informazio zehatza ikusteko joan [33]. Kritikoaren sare neuronalaren irudikapena **25.irudian** ematen da.



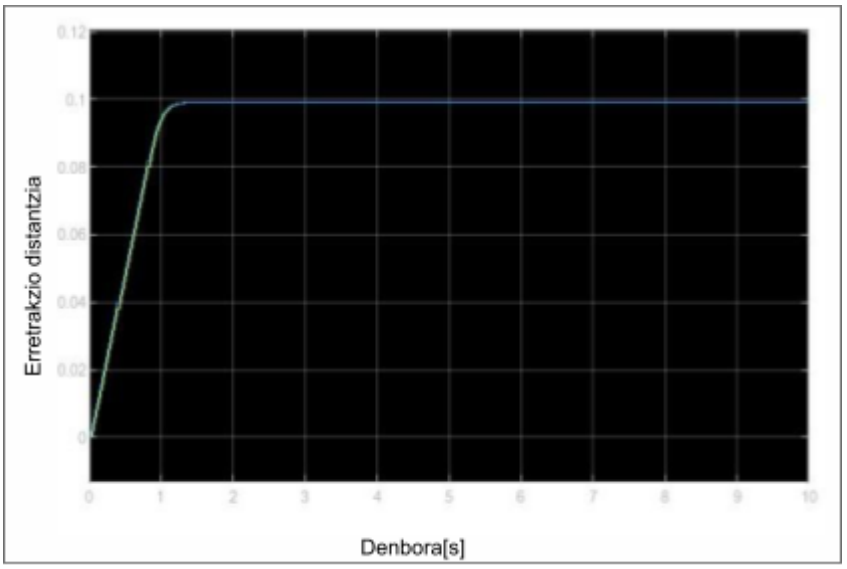
25.Irudia: Kritioaren behin betiko sare neuronalaren irudikapena.

5.3. Sari (reward) funtzioa eta saiakerak.

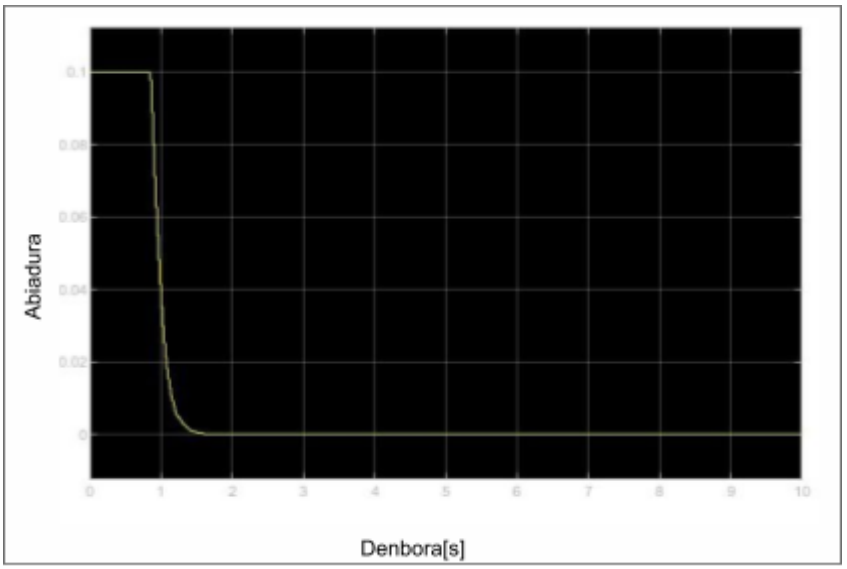
Egindako programarekin ia da, helburua lortzen duen sistema bat eraiki dugu. **26, 27, 28.Irudietan** ikus dezakegun moduan, gorde ditugun agenteen artean hoberena

hautatuz, hurrengo grafikak sortu ditugu. Indarra, abiadura eta posizioa denborarekiko ikus ditzazkegu.

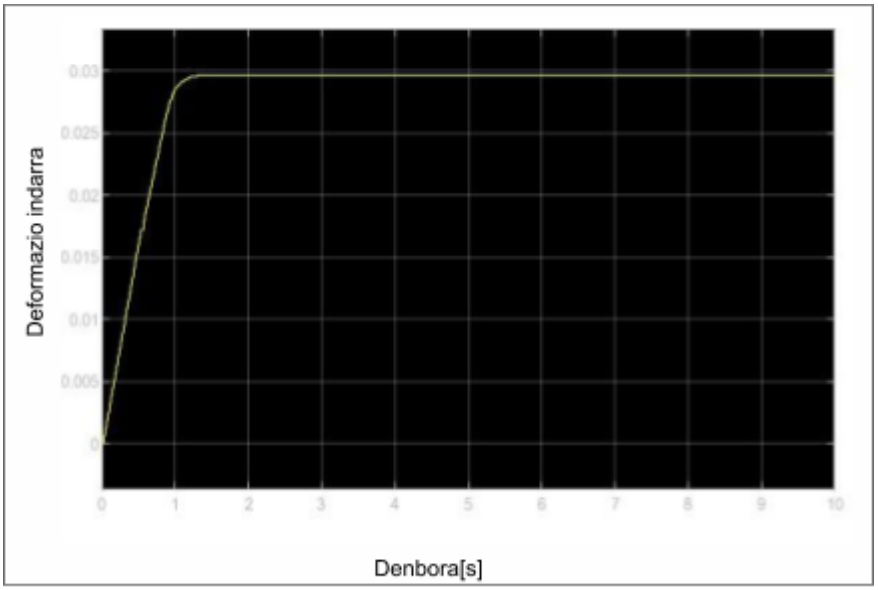
Ikus dezakegun moduan, guk ezarritako sari funtzioarekin antzezleak hasieran abiadura maximoa ezartzea ondoren guztik gelditzeko aukera onena dela erabaki du.



26.Irudia: Erretrakzio posizioa denboraren baitan.(Guztiz ez optimizatua)



27.Irudia: Erretrakzio abiadura denobraren baitan.(Guztiz ez optimizatua)



28.Irudia: Deformazio indarra denboraren baitan.(Guztiz ez optimizatua)

Hauek egokiak irudi dezakete, baina ez dira [1] artikuluan atera zituzten erantzun berak. Horregatik, *reward* funtzioa eta haren lamdak aldatuta, bigarren helburua lortzeko froga asko egin ditugu.

Frogak egin ondoren, ez dugu lortu pasaden artikuloaren erantzun berdinak. Gure helburua lortzeaz hurbilen egon garen kasuan sari funtzio berri batekin (**4.ekuazioa**) eta lamden balio berriekin (**12.taulan**) izan da.

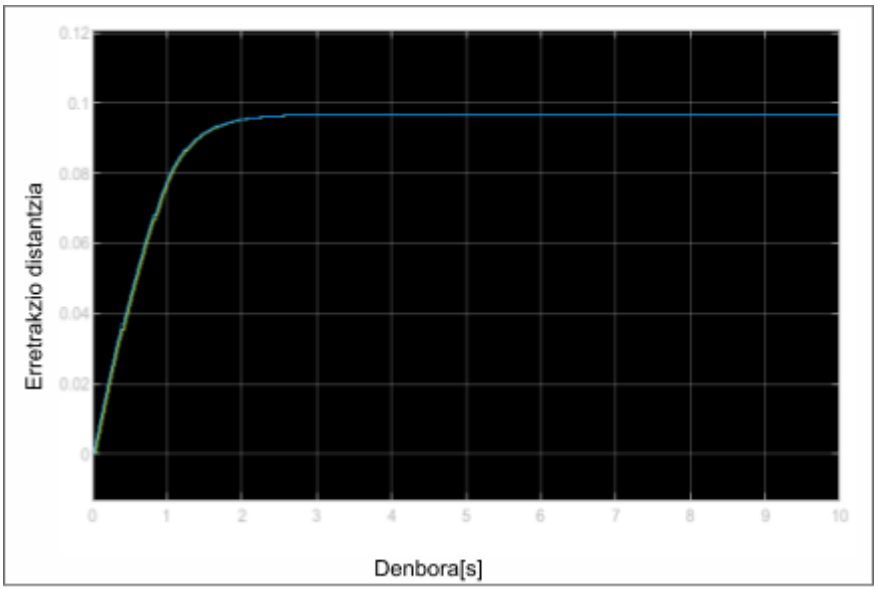
$$R = -\left(\left(\frac{(\lambda_1 \cdot ((x_{desired} - x)^2)) \cdot F^2 + \lambda_1 \cdot \lambda_3 \cdot ((x_{desired} - x)^2)}{(\dot{x} + 0.0001)}\right) + \lambda_2 \cdot \dot{x}\right) + Penalty \quad (4)$$

12.Taula: Lamden behin betiko balioak.

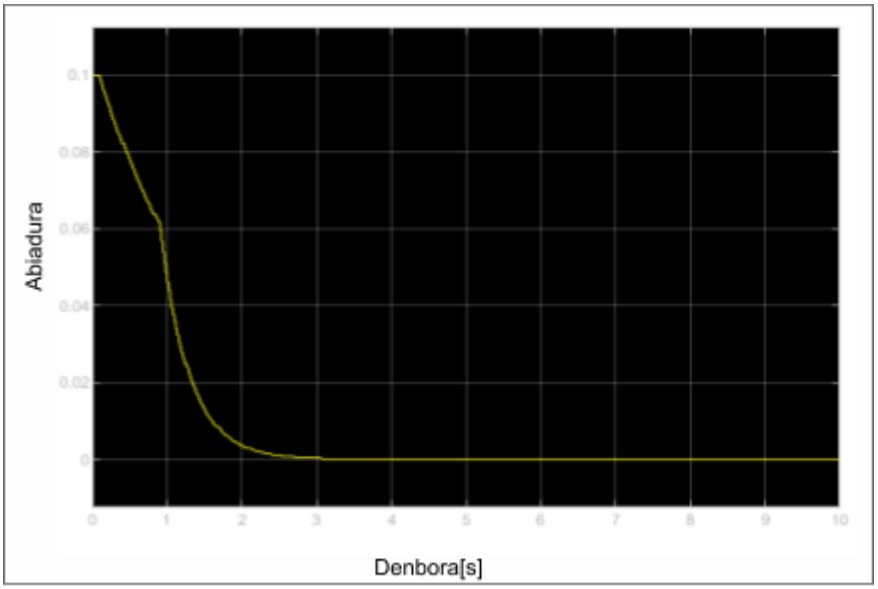
Lamda 1 (λ_1)	1
Lamda 2 (λ_2)	0.1
Lamda 3 (λ_3)	0.1

6. ERANTZUNAK.

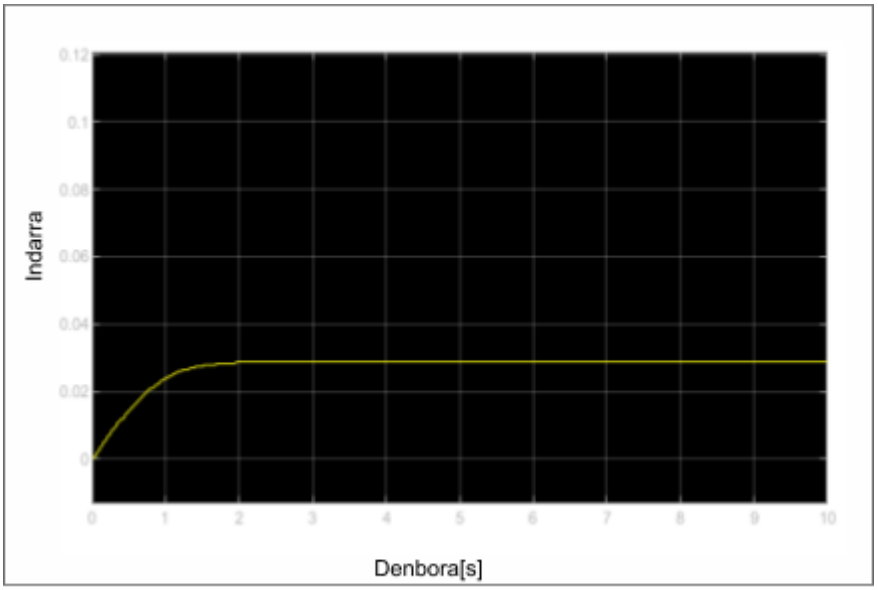
Optimizazio prosezuaren ondoren eta ahal izan ditugun sare neuronal, Matlab aukera eta sari futzio hoberenak atera ostean, **29, 30 eta 31.irudietan** ikus dezakegu gure agente hoberenak ateratako erantzuna (Ikus "A Eranskina" amaierako programa osoa ikusteko).



29.Irudia: Erretrakzio posizioa denboraren baitan.

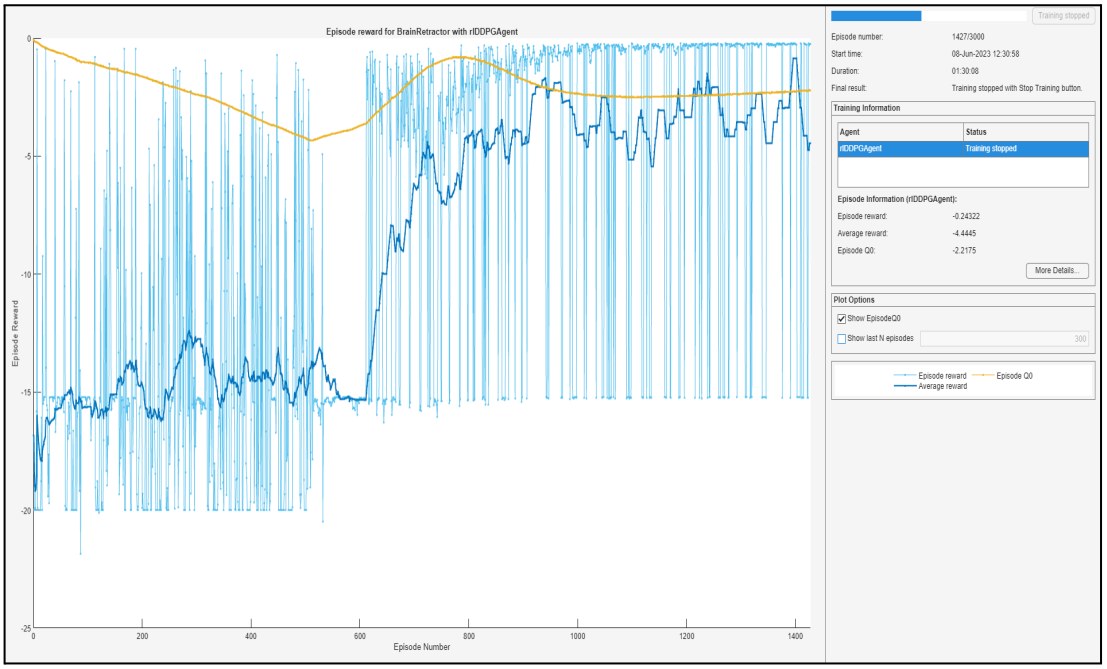


30.Irudia: Erretrakzio abiadura denboraren baitan.



31.Irudia: Erretrakzio indarra denboraren baitan.

Agenteak gordetzeko eginiko entrenamendu baten grafika **32.irudian** ikus daiteke.



32.Irudia: Erretrakzio posizioa denoraren baitan.

7. ONDORIOAK ETA ETORKIZUNERAKO AURRERAPENAK.

[1] Artikulua emandako ezagutzak eta indartzeko ikaskuntza-teknika sakonak erabiliz kontrol-arazo bati irtenbide bat bilatu diogu. Bi helburu ezarri genituen arazo honi aurre egiteko:

1. Robota burmuina zalbaldu beharko du guk desiratutako posizioa lortu arte.
2. Iskemia kausa ditzazkeen ondorioak ahalik eta gehien ekidituko dira, indarra eta abiadura murriztuz.

Agente funtzional bat sortu dugu. Hasieran, agentea politika gabe eta ausazko ekintzak aukeratuz hasi da. Hobekuntza prozesu baten ondoren, gure lehenengo helburua gainditzea lortu dugu. Halere, bigarren helburua lortze arren eginiko saiakera kantitatea handia bada ere, ez dugu gure abiadura grafika (**30.irudia**), erreferentzia grafikaren (**4.irudia**) berdina izatea lortu.

Ondoriozta dezakegu, arazoa sari futzioarekin eralzionatuta dagoela, seguraski, distantziari askoz garrantzi asko eman diogulako abiadurari baino. Hau konpontzeko baitan, garrantzia orekatzen saiatu gara, baina une oro, gure agentea erantzun suboptimo batean trabatu da.

Bi konpobide proposatuko ditugu honi aurre egiteko. Alde batetik, sari futzioa orekatu eta sare neuronalak handitu beharko dira. Honekin batera, ikaste abiadura eta aukerak doituz. Honek konputazio ahalmen handiago bat eskatuko du, baina baliteke bigarren helburua lortzeko gai izatea.

Bestetik, sare neuronalak eta aukerak mantenduz soilik *reward* funtzioa aldatzea izango da gure bigarren proposamena. Horretarako, arazoa beste ikuspuntu batetik konpontzen saiatuko gara. Kasu honetan sari funtzioan garrantzi gehien duen aldagaia abiadura izango da, honela robota hau minimizatzen saiatuko da. Honek, seguraski, agentea erantzun suboptimo batera bultzatuko du, non, abiadura 0 izango da momentu oro. Hau ekiditzeko sari futzioari sari bat gehituko diogu gure agentea lehenengo helburua lortu duenean eta zigor bat simulazioaren amaieran helburua lortu ez badu.

Hau guztiz aparte, gure programak duen muga handienetako bat, entrenamenduaren ondoren, nahi dugun erretrakzio distanzia aldatzen badugu ere, agenteak bere portaera modifikatzen ez duela da. Horregatik, erretrakzio distantzia berrietarako agentea berriz

entrenamendu bat jasan behar du. Hau konpontzeko, agentea entrenamendu bakarrean desiratutako distantzia desberdinetarako entrenatu beharko dugu. Horretarko, lehen sortutako "*cleanstart*" funtzioan desiratutako distantzia ausazko aldagai baten moduan sartzea proposatzen dugu. Ikus "**B eranskina"** honen adibide/saiakera bat ikusteko. Honekin batera, seguraski, sare neuronal eta Matlab aldagei aldaketak egin beharko zaie.

8. AURREKONTUA.

KONTZEPTUA	Koste Basikoa	Amortizatutako Kostea
Ekipo Informatikoak:		
Ordenagailu pertsonala:*	1.500€	77€
Intel(R) Core(TM) i5-4460 CPU@ 3.20GHz		
8,0 GB DDR3		
NVIDIA GeForce GTX 960		
Acer Aspire 5 A515-51G-59ST*	500€	26€
Programa informatikoak:		
Matlab R2023b	6.000€	309€
Eskulana:		
450 ordu, 50€/h	22.500€	22.500€
GASTU TOTALAK		**22.912€**
Enpresa-gastuak:		
%15 zeharkako kostuak		26.349€
%20 mozkin industriala		31.619€
GUZTIRA, BEZik GABE		**31.619€**
%21 BEZa		38.258€
AURREKONTUA GUZTIRA		**38.258€**
*4 urterako amortizazioa		

Bibliografia

1. Ibai Inziarte-Hidalgo; Irantzu Uriarte; Unai Fernandez-Gamiz; Gorka Sorrosal, Ekaitz Zulueta. Robotic-Arm-Based Force Control in Neurosurgical Practice. 2023. [Esteka]
2. Dewan, M.C.; Rattani, A.; Fieggen, G.; Arraez, M.A.; Servadei, F.; Boop, F.A.; Johnson, W.D.; Warf, B.C.; Park, K.B. Global Neurosurgery: The Current Capacity and Deficit in the Provision of Essential Neurosurgical Care. Executive Summary of the Global Neurosurgery Initiative at the Program in Global Surgery and Social Change. J. Neurosurg. 2019, 130, 1055–1064. [Esteka]
3. Bennett, M.H.; Albin, M.S.; Bunegin, L.; Dujovny, M.; Hellstrom, H.; Jannetta, P.J. Evoked Potential Changes during Brain Retraction in Dogs. Stroke 1977, 8, 487–492. [Esteka]
4. Dai, Z. Improvement of General Design Theory and Methodology with Its Application to Design of a Retractor for Ventral Hernia Repair Surgery. Master's Thesis, University of Saskatchewan, Saskatoon, SK, Canada, March 2019
5. Dujovny, M.; Wackenhut, N.; Kossovsky, N.; Leff, L.; Gómez, C.; Nelson, D. Biomechanics of Vascular Occlusion in Neurosurgery. Acta Neurol. Lat. 1980, 26, 123–127.
6. Fukamachi, A.; Koizumi, H.; Nukui, H. Postoperative Intracerebral Hemorrhages: A Survey of Computed Tomographic Findings after 1074 Intracranial Operations. Surg. Neurol. 1985, 23, 575–580. [Esteka]
7. Kalfas, I.H.; Little, J.R. Postoperative Hemorrhage: A Survey of 4992 Intracranial Procedures. Neurosurgery 1988, 23, 343–347. [Esteka]
8. Rosenørn, J. The Risk of Ischaemic Brain Damage during the Use of Self-Retaining Brain Retractors. Acta Neurol. Scand. 1989, 79, 1–30. [Esteka]

9. Yokoh, A.; Sugita, K.; Kobayashi, S. Clinical Study of Brain Retraction in Different Approaches and Diseases. Acta Neurochir. 1987, 87, 134–139. [Esteka].

10. Bell, B.A.; Symon, L.; Branston, N.M. CBF and Time Thresholds for the Formation of Ischemic Cerebral Edema, and Effect of Reperfusion in Baboons. J. Neurosurg. 1985, 62, 31–41. [Esteka]

11. Hoeckelmann, M.; Rudas, I.J.; Fiorini, P.; Kirchner, F.; Haidegger, T. Current Capabilities and Development Potential in Surgical Robotics. Int. J. Adv. Robot. Syst. 2015, 12, 61. [Esteka]

12. DeLorenzo, C.; Papademetris, X.; Staib, L.H.; Vives, K.P.; Spencer, D.D.; Duncan, J.S. Volumetric Intraoperative Brain Deformation Compensation: Model Development and Phantom Validation. IEEE Trans. Med. Imaging 2012, 31, 1607–1619. [Esteka]

13. Dai, Z. Improvement of General Design Theory and Methodology with Its Application to Design of a Retractor for Ventral Hernia Repair Surgery. Master's Thesis, University of Saskatchewan, Saskatoon, SK, Canada, March 2019

14. DeLorenzo, C.; Papademetris, X.; Staib, L.H.; Vives, K.P.; Spencer, D.D.; Duncan, J.S. Volumetric Intraoperative Brain Deformation Compensation: Model Development and Phantom Validation. IEEE Trans. Med. Imaging 2012, 31, 1607–1619. [Esteka]

15. Shimon Nof. Handbook of Industrial Robotics. John Wiley & Sons, 1999.

16. Michael E. Moran. Evolution of robotic arms. Journal of Robotic Surgery, 2007.

17. Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. TheMIT press, 2014-2015. [Esteka]

18. Ivan.P.Pavlov. Los reflejos condicionados. 1997 [Esteka]

19. Simon S. Du; Sham M.; Ruosong Wang; Lin F. Yang. Is a Good Representation Sufficient For Sample Efficient Reinforcement Learning. ICLR 2020. [Esteka]
20. Cornel Secara and Luige Vladareanu. Iterative strategies for obstacle avoidance of a redundant manipulator. Proceedings of the Romanian Academy - Series A: Mathematics, Physics, Technical Sciences, Information Science, 9, 01 2010.
21. Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018 [Esketa]
22. Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019 [Esteka]
23. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017
24. M. Minsky and S. A. Papert. Perceptrons: An Introduction to Computational Geometry. 2017 [Online]. [Esteka]. [Accessed: 05-Jun-2023]
25. rlNumericSpec (MatLab Documentation). Introduced in R2019a. Create continuous action or observation data specifications for reinforcement learning environments. [Esteka]
26. rlSimulinkEnv (MatLab Documentation). Introduced in R2019a. Create a reinforcement learning environment using a dynamic model implemented in Simulink.. [Esteka]
27. rlContinuousDeterministicActor (MatLab Documentation). Introduced in R2022a. Deterministic actor with a continuous action space for reinforcement learning agents. [Esteka]
28. rlQValueFunction (MatLab Documentation). Introduced in R2022a. Q-Value function approximator object for reinforcement learning agents. [Esteka]
29. rlDDPGAgent (MatLab Documentation). Introduced in R2022a. Deep deterministic policy gradient (DDPG) reinforcement learning agent. [Esteka]

30. rlOptimizerOptions (MatLab Documentation). Introduced in R2022a. Optimization options for actors and critics.[Esteka]

31. rlDDPGAgentOptions (MatLab Documentation). Introduced in R2019a. Options for DDPG agent. [Esteka]

32. rlTrainingOptions (MatLab Documentation). Introduced in R2019a. Options for training reinforcement learning agents. [Esteka]

33. Layer (MatLab Documentation). Introduced in R2016a. Deep learning geruza. [Esteka]

34. layerGraph (MatLab Documentation). Introduced in R2017b. Graph of network layers for deep learning. [Esteka]

A ERANSKINA

Amaierako programa.

1. MatLab programa osoa:

```
clc
clear
%Simulink eredua ireki eta exekutatu.
mdl = 'SimulinkEreduaBurmuinErretrakzioa';
open_system(mdl)

%Behaketa eta ekintza informazioa zehaztu.
    %Bi behaketa: F(indarra) eta x (posizio/distantzia).
    %Ekintza bakarra: xdot(abiadura).
ObservationInfo = rlNumericSpec([2 1]);
ActionInfo = rlNumericSpec([1 1],"Lowerlimit",0,"Upperlimit",0.1);

%Ereduaren parametroak:(Reward funtzioan garrantzia zer duen aldatzeko
erabilitakoak)
envConstants.Lambda1 = 1; %Distanzia eta indarraren garrantziaren kontrola.
envConstants.Lambda2 = 0.1; %Abiaduraren garrantziaren kontrola.
envConstants.Lambda3 = 0.1; %Distantziaren garrantziaren kontrola.
%Ogden Model ireki deformazio indarra kalkulatzeko.
load('OgdenModelData.mat')

%Ereduaren Parametroak:(Array batean parametro guztiak jarri, Simulink ereduan
sartzeko)
envConstants.OgdenModelData=OgdenModelData;
envConstants.xdesired=0.1;
   %Lagin-denbora.
envConstants.Ts = 0.02;
   %Baimendutako Errorea.
envConstants.XThreshold=0.01;
```

```
    %Zigorra bukaerako aukerak betetzen badira.
envConstants.PenaltyForFalling = -15
%INGURUAREN sorketa
env=rlSimulinkEnv("SimulinkEreduaBurmuinErretrakzioa","SimulinkEreduaBurmui
nErretrakzioa/controller",ObservationInfo,ActionInfo);

%Ordezkatu ResetFcn gure "cleanstart" funtzioarekin.
env.ResetFcn = @cleanstart;

rng(0)

%Neurona kantitatea guztiz konektatutako geruzetan.
NeuQuantity=50;
%SARE NEURONALA sortu antzezlarentzako.
actnet = [
    featureInputLayer(2,"Name","obs1")
    fullyConnectedLayer(NeuQuantity,"Name","fc1")
    reluLayer("Name","relu1")
    fullyConnectedLayer(NeuQuantity,"Name","fc2")
    reluLayer("Name","relu2")
    fullyConnectedLayer(NeuQuantity,"Name","fc3")
    reluLayer("Name","relu3")
    fullyConnectedLayer(1,"Name","act1")
    reluLayer("Name","relu6")];

%Sare neuronala irudikatu.
plot(layerGraph(actnet));

%Antzezlea eraiki sortutako Sare Neuronalarekin eta aukerekin.
actor = rlContinuousDeterministicActor(actnet,ObservationInfo,ActionInfo);

%SARE NEURONALA sortu kritikoarentzako.
```

```
obsPath = [featureInputLayer(2,"Name","obs2")
    fullyConnectedLayer(NeuQuantity,"Name","fc1")
    reluLayer("Name","relu1")
    fullyConnectedLayer(NeuQuantity,"Name","fc2")
    additionLayer(2,"Name","add")
    reluLayer("Name","relu3")
    fullyConnectedLayer(1,"Name","value")];
actPath = [featureInputLayer(1,"Normalization","none","Name","act2")
    fullyConnectedLayer(NeuQuantity,"Name","fcact1")];

%Sare neuronala lotu.
qvalnet = layerGraph(obsPath);
qvalnet = addLayers(qvalnet,actPath);
qvalnet = connectLayers(qvalnet,"fcact1","add/in2");

%Sare neuronala irudikatu.
plot(qvalnet);

%Kritikoa eraiki sortutako Sare Neuronalarekin eta aukerekin.
critic=rlQValueFunction(qvalnet,ObservationInfo,ActionInfo,ObservationInputNames
="obs2",ActionInputNames="act2");

%Antzezle eta kritikoaren aukerak ezarri:
actorOpts = rlOptimizerOptions(LearnRate=1e-05,GradientThreshold=1);
criticOpts = rlOptimizerOptions(LearnRate=1e-04,GradientThreshold=1);

%Agentearen aukerak ezarri:
agentOpts = rlDDPGAgentOptions(...
    SampleTime=envConstants.Ts,...
    ExperienceBufferLength=1e6,...
    MiniBatchSize=128,...
    ActorOptimizerOptions=actorOpts,...
```

```
    CriticOptimizerOptions=criticOpts,...
    DiscountFactor=0.99);
agentOpts.NoiseOptions.Variance = 0.035;
agentOpts.NoiseOptions.VarianceDecayRate = 1e-5

%Agentea sortu, antzezle eta kritikoarekin.
agent = rlDDPGAgent(actor,critic,agentOpts);

%Entrenamenduaren aukerak ezarri.
maxEpisodes = 3000;
trainOpts = rlTrainingOptions(...
    'MaxEpisodes',maxEpisodes,...
    'MaxStepsPerEpisode',200,...
    'ScoreAveragingWindowLength',50,...
    'Verbose',false,...
    'Plots','training-progress',...
    'StopTrainingCriteria','AverageReward','StopTrainingValue',-1e-3,...
    'StopTrainingValue',maxEpisodes,...
    'SaveAgentCriteria','EpisodeReward',...
    'SaveAgentValue',-0.215);
%Simulink ererduan gure agentea entrenatu.
info = train(agent,env,trainOpts);
```

2. **“Cleanstart” funtzioa:**

```
function [in] = cleanstart(in)
%Hay que crear las variables en el Workspace de Simulink
%Funcion de reseteo cada vez que empieza una simulacion u/o entrenamieto
mdl="SimulinkEreduaBurmuinErretrakzioa";
in = setVariable(in,"x",0,"Workspace",mdl);
in = setVariable(in,"F",0,"Workspace",mdl);
in = setVariable(in,"xdot",0,"Workspace",mdl);
in = setVariable(in,"xanterior",0,"Workspace",mdl);
end
```

3. Simulink programa osoa:

```
function [F,x,reward,isDone] = RewardFunction(Lambda1, Lambda2,
Lambda3,PenaltyForFalling,XThreshold,OgdenModelData,xdesired,Ts,xaurrekoa,ekin
tza)
xdot = ekintza;
%Posizioa/distanzia kalkulatu.
x = xaurrekoa + Ts*xdot;
% Ogden Model Data-rekin ekuazioa ezarrita deformazio indarraren kalkulua
% egin.
F0=interp1(OgdenModelData(:,2),OgdenModelData(:,1),0,'linear','extrap');
F=interp1(OgdenModelData(:,2),OgdenModelData(:,1),x,'linear','extrap');
F=F-F0;
%AMAIERA BALDINZAK KONPROBATU.
if x > xdesired + XThreshold %Gehiegi ireki ez dugula konprobatu.
isDone=1;
elseif x<0 %Balio negatinoak hartu ez ditugula konprobatu.
isDone=1;
elseif x-xaurrekoa<0 %Irekitzen geudenenan atzerako mugimendua egin ez dugula
konprobatu.
isDone=1;
else
isDone=0;
end
epsilon=0.01;
%Saria kalkualtu.
if isDone == 0
reward=-(((Lambda1*((xdesired-x)^2)*F^2+Lambda1*Lambda3*((xdesired-x)^2))/(x
dot+epsilon))+Lambda2*(xdot^2));
else
reward=-(((Lambda1*((xdesired-x)^2)*F^2+Lambda1*Lambda3*((xdesired-x)^2))/(x
dot+epsilon))+Lambda2*(xdot^2));
reward = reward+PenaltyForFalling;
```

```
end
end
```

B ERANSKINA

Desiratutako erretrakzio distantziaren aldaketa, entrenamenduaren kapitulu bakoitzean. Programa hau, soilik ideia bat emateko egin da, ez da optimizatu ez frogatu.

1. **MatLab programa osoa:**

```
clc
clear
%Simulink eredua ireki eta exekutatu.
mdl = “EraldatutakoSimulinkEreduaBurmuinErretrakzioa”;
open_system(mdl)

%Behaketa eta ekintza informazioa zehaztu.
    %Hiru behaketa: F(indarra), x (posizio/distantzia) eta Xdesired (desiratutako distantzia).
    %Ekintza bakarra: xdot(abiadura).
ObservationInfo = rlNumericSpec([3 1]);
ActionInfo = rlNumericSpec([1 1],"Lowerlimit",0,"Upperlimit",0.1);

%Ereduaren parametroak:(Reward funtzioan garrantzia zer duen aldatzeko erabilitakoak)
envConstants.Lambda1 = 1; %Distanzia eta indarraren garrantziaren kontrola.
envConstants.Lambda2 = 0.1; %Abiaduraren garrantziaren kontrola.
envConstants.Lambda3 = 0.1; %Distantziaren garrantziaren kontrola.
%Ogden Model ireki deformazio indarra kalkulatzeko.
load('OgdenModelData.mat')

%Ereduaren Parametroak:(Array batean parametro guztiak jarri, Simulink ereduan sartzeko)
envConstants.OgdenModelData=OgdenModelData;
  %Lagin-denbora.
envConstants.Ts = 0.02;
  %Baimendutako Errorea.
```

```
envConstants.XThreshold=0.01;
   %Zigorra bukaerako aukerak betetzen badira.
envConstants.PenaltyForFalling = -15
%INGURUAREN sorketa
env=rlSimulinkEnv("EraldatutakoSimulinkEreduaBurmuinErretrakzioa","Eraldatutak
oSimulinkEreduaBurmuinErretrakzioa/controller",ObservationInfo,ActionInfo);

%Ordezkatu ResetFcn gure "EraldatuakoCleanstart" funtzioarekin.
env.ResetFcn = @EraldatuakoCleanstart;

rng(0)

%Neurona kantitatea guztiz konektatutako geruzetan.
NeuQuantity=50;
%SARE NEURONALA sortu antzezlarentzako.
actnet = [
   featureInputLayer(2,"Name","obs1")
   fullyConnectedLayer(NeuQuantity,"Name","fc1")
   reluLayer("Name","relu1")
   fullyConnectedLayer(NeuQuantity,"Name","fc2")
   reluLayer("Name","relu2")
   fullyConnectedLayer(NeuQuantity,"Name","fc3")
   reluLayer("Name","relu3")
   fullyConnectedLayer(1,"Name","act1")
   reluLayer("Name","relu6")];

%Sare neuronala irudikatu.
plot(layerGraph(actnet));

%Antzezlea eraiki sortutako Sare Neuronalarekin eta aukerekin.
actor = rlContinuousDeterministicActor(actnet,ObservationInfo,ActionInfo);
```

```
%SARE NEURONALA sortu kritikoarentzako.
obsPath = [featureInputLayer(2,"Name","obs2")
    fullyConnectedLayer(NeuQuantity,"Name","fc1")
    reluLayer("Name","relu1")
    fullyConnectedLayer(NeuQuantity,"Name","fc2")
    additionLayer(2,"Name","add")
    reluLayer("Name","relu3")
    fullyConnectedLayer(1,"Name","value")];
actPath = [featureInputLayer(1,"Normalization","none","Name","act2")
    fullyConnectedLayer(NeuQuantity,"Name","fcact1")];

%Sare neuronala lotu.
qvalnet = layerGraph(obsPath);
qvalnet = addLayers(qvalnet,actPath);
qvalnet = connectLayers(qvalnet,"fcact1","add/in2");

%Sare neuronala irudikatu.
plot(qvalnet);

%Kritikoa eraiki sortutako Sare Neuronalarekin eta aukerekin.
critic=rlQValueFunction(qvalnet,ObservationInfo,ActionInfo,ObservationInputNames
="obs2",ActionInputNames="act2");

%Antzezle eta kritikoaren aukerak ezarri:
actorOpts = rlOptimizerOptions(LearnRate=1e-05,GradientThreshold=1);
criticOpts = rlOptimizerOptions(LearnRate=1e-04,GradientThreshold=1);
%Agentearen aukerak ezarri:
agentOpts = rlDDPGAgentOptions(...
    SampleTime=envConstants.Ts,...
    ExperienceBufferLength=1e6,...
    MiniBatchSize=128,...
    ActorOptimizerOptions=actorOpts,...
```

```
    CriticOptimizerOptions=criticOpts,...
    DiscountFactor=0.99);
agentOpts.NoiseOptions.Variance = 0.035;
agentOpts.NoiseOptions.VarianceDecayRate = 1e-5

%Agentea sortu, antzezle eta kritikoarekin.
agent = rlDDPGAgent(actor,critic,agentOpts);

%Entrenamenduaren aukerak ezarri.
maxEpisodes = 3000;
trainOpts = rlTrainingOptions(...
    'MaxEpisodes',maxEpisodes,...
    'MaxStepsPerEpisode',200,...
    'ScoreAveragingWindowLength',50,...
    'Verbose',false,...
    'Plots','training-progress',...
    'StopTrainingCriteria','AverageReward','StopTrainingValue',-1e-3,...
    'StopTrainingValue',maxEpisodes,...
    'SaveAgentCriteria','EpisodeReward',...
    'SaveAgentValue',-0.215);
%Simulink ererduan gure agentea entrenatu.
info = train(agent,env,trainOpts);
```

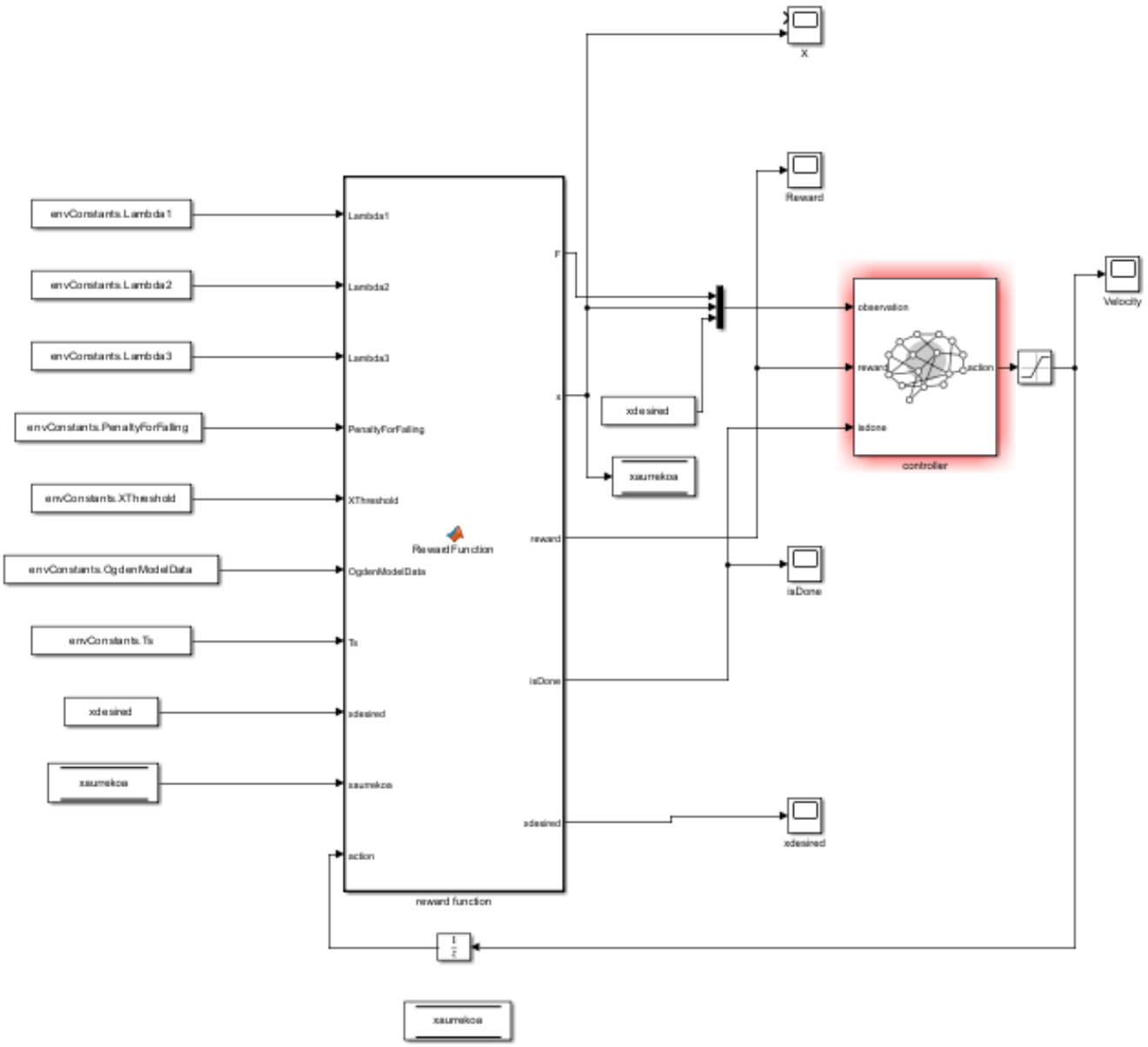
2. **“Cleanstart” funtzioa:**

```
function [in] = cleanstart(in)
%Hay que crear las variables en el Workspace de Simulink
%Funcion de reseteo cada vez que empieza una simulacion u/o entrenamieto
mdl="EraldatutakoSimulinkEreduaBurmuinErretrakzioa";
in = setVariable(in,"x",0,"Workspace",mdl);
in = setVariable(in,"F",0,"Workspace",mdl);
in = setVariable(in,"xdot",0,"Workspace",mdl);
in = setVariable(in,"xdesired",randi([1, 5]) / 10,"Workspace",mdl)
end
```

3. Simulink programa osoa eta argazkia:

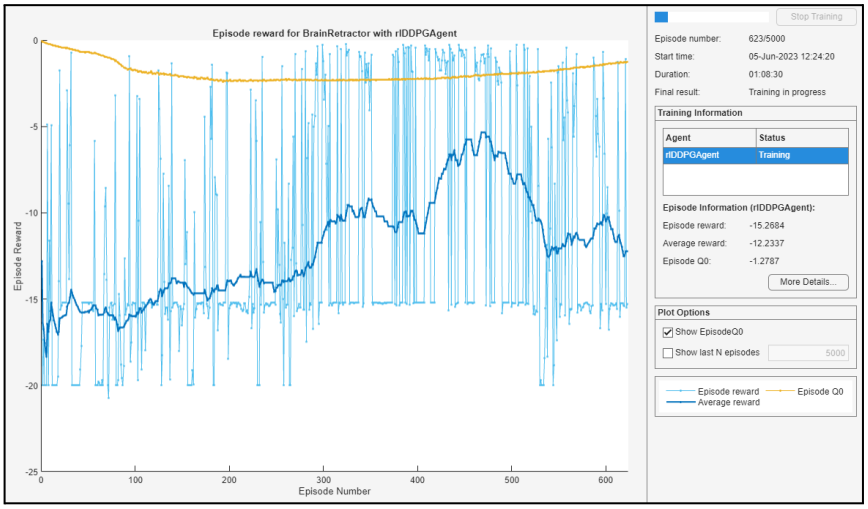
```
function [F,x,reward,isDone,xdesired] = RewardFunction(Lambda1, Lambda2,
Lambda3,PenaltyForFalling,XThreshold,OgdenModelData,Ts,xdesired,xaurrekoa,ekin
tza)
xdot = ekintza;
%Posizioa/distanzia kalkulatu
x = xaurrekoa + Ts*xdot;
% Ogden Model Data-rekin ekuazioa ezarrita deformazio indarraren kalkulua
% egin.
F0=interp1(OgdenModelData(:,2),OgdenModelData(:,1),0,'linear','extrap');
F=interp1(OgdenModelData(:,2),OgdenModelData(:,1),x,'linear','extrap');
F=F-F0;
%AMAIERA BALDINZAK KONPROBATU.
if x > xdesired + XThreshold %Gehiegi ireki ez dugula konprobatu.
isDone=1;
elseif x<0 %Balio negatinoak hartu ez ditugula konprobatu.
isDone=1;
elseif x-xaurrekoa<0 %Irekitzen geudenenan atzerako mugimendua egin ez dugula
konprobatu.
isDone=1;
else
isDone=0;
end
epsilon=0.01;
%Saria kalkualtu.
if isDone == 0
reward=-(((Lambda1*((xdesired-x)^2)*F^2+Lambda1*Lambda3*((xdesired-x)^2))/(x
dot+epsilon))+Lambda2*(xdot^2));
else
reward=-(((Lambda1*((xdesired-x)^2)*F^2+Lambda1*Lambda3*((xdesired-x)^2))/(x
dot+epsilon))+Lambda2*(xdot^2));
reward = reward+PenaltyForFalling;
```

```
end
end
```

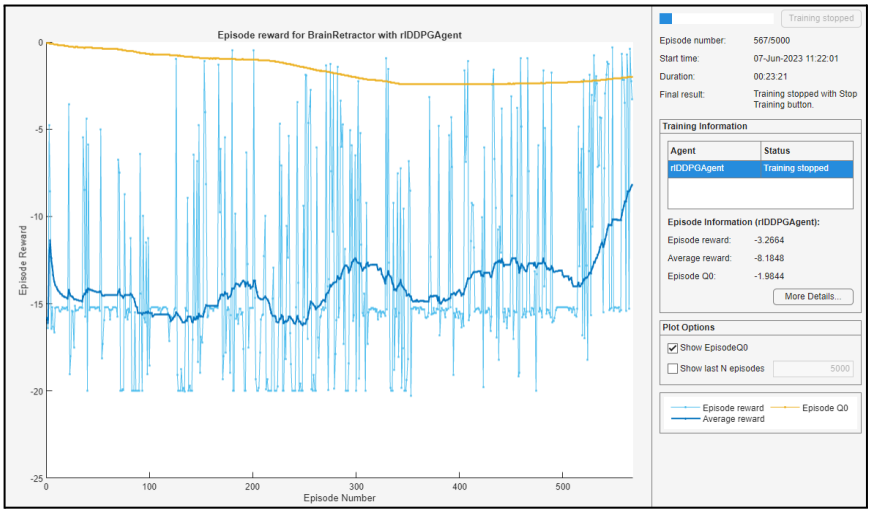

X
Reward
Velocity
envConstants.Lambda1
envConstants.Lambda2
envConstants.Lambda3
envConstants.PenaltyForFalling
envConstants.XThreshold
envConstants.OgdenModelData
envConstants.Ts
xdesired
xaumekca
Lambda1
Lambda2
Lambda3
PenaltyForFalling
XThreshold
RewardFunction
OgdenModelData
Ts
xdesired
xaumekca
action
reward function
F
x
reward
isDone
xdesired
xaumekca
observation
reward
isdone
action
controller
isDone
xdesired
1
z
xaumekca

C ERANSKINA

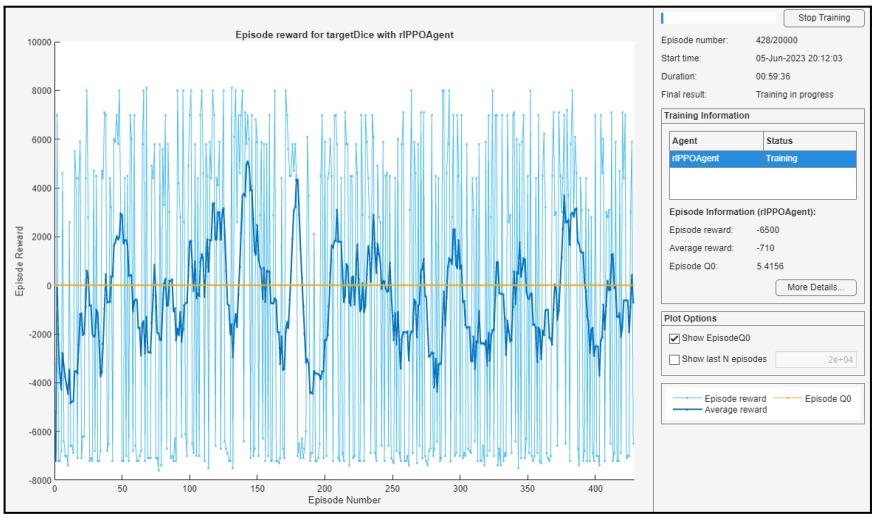
1. **Sare neuronalen frogak 6 geruza eta neurona kantitate desberdinekin.**
   a. **6 geruza, 100 neurona:**



   b. **6 geruza, 50 neurona:**

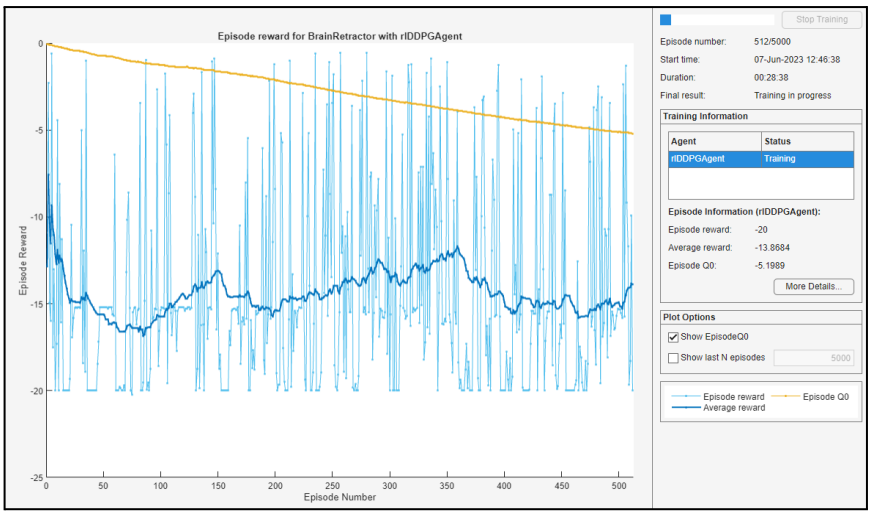


   c. **6 geruza 10 neurona:**

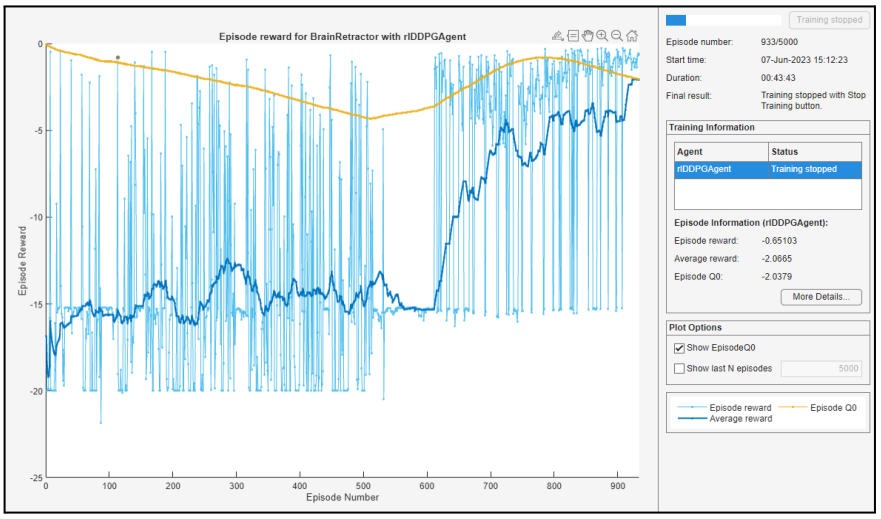


2. **Sare neuronalen frogak 3 geruza eta neurona kantitate desberdinekin.**

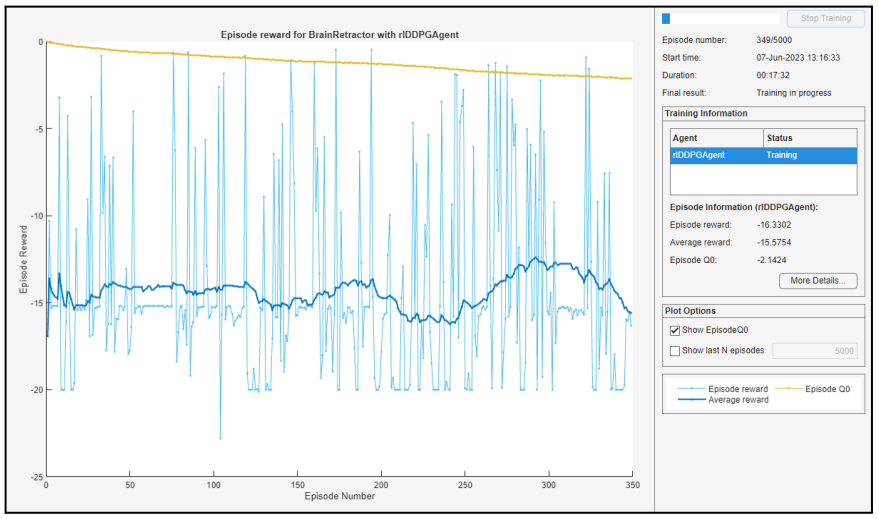
   a. **3 geruza, 100 neurona:**



   b. **3 geruza, 50 neurona:**



   c. **3 geruza 10 neurona**



GANTT DIAGRAMA

ATAZA	Hasiera	Bukaera	1.Astea (01/03/2023-05/03/2023)	2.Astea (06/03/2023-12/03/2023)	3.Astea (13/03/2023-19/03/2023)	4.Astea (20/03/2023-26/03/2023)	5.Astea (27/03/2023-02/04/2023)	6.Astea (03/04/2023-09/04/2023)	7.Astea (10/04/2023-16/04/2023)	8.Astea (17/04/2023-23/04/2023)
Informazio eskuraketa eta ikasketa.	1.Astea	3.Astea								
RL-ren ulerketa.	2.Astea	15.Astea								
DDPG agentearen sorketa.	3.Astea	4,Astea								
DDPG agentearen simulink eredua.	4.Astea	6.Astea								
DDPG agentearen kodigo eguneratu.	5.Astea	6.Astea								
Simulazioak eta entrenamenduaren optimizazioa	5.Astea	11.Astea								
Entrenamenduen frogapenak	6.Astea	12.Astea								
TFG-aren idazketa osoa.	11.Astea	15.Astea								

ATAZA	Hasiera	Bukaera	9.Astea (24/04/2023-30/04/2023)	10.Astea (01/05/2023-07/05/2023)	11.Astea (08/05/2023-14/05/2023)	12.Astea (15/05/2023-21/05/2023)	13.Astea (22/05/2023-28/05/2023)	14.Astea (29/05/2023-04/06/2023)	15.Astea (05/06/2023-11/06/2023)
Informazio eskuraketa eta ikasketa.	1.Astea	3.Astea							
RL-ren ulerketa.	2.Astea	15.Astea							
DDPG agentearen sorketa.	3.Astea	4,Astea							
DDPG agentearen simulink eredua.	4.Astea	6.Astea							
DDPG agentearen kodigo eguneratu.	5.Astea	6.Astea							
Simulazioak eta entrenamenduaren optimizazioa	5.Astea	11.Astea							
Entrenamenduen frogapenak	6.Astea	12.Astea							
TFG-aren idazketa osoa.	11.Astea	15.Astea							