UNIVERSITETI I PRISHTINËS "HASAN PRISHTINA"

FAKULTETI I INXHINIERISË MEKANIKE

DEPARTAMENTI I MEKATRONIKËS

TEMA E DIPLOMËS - MASTER

NGRITJA NË KOHË REALE E LAVJERRËSIT LINEAR TË PËRMBYSUR DUKE PËRDORUR REINFORCEMENT LEARNING

Mentor:
Prof. Asoc. Dr. Xhevahir Bajrami

Kandidat:
BSc Fisnik Kaçiu

Prishtinë, Republika e Kosovës, 2024

Abstrakt

Përparimi i shpejtë i inteligjencës artificiale ka çuar në integrimin në rritje të algoritmeve inteligjente të kontrollit në sistemet moderne të kontrollit, duke adresuar kërkesën në rritje për zgjidhje me performancë të lartë. Kjo tezë paraqet modifikimin e një kontrolluesi me gradient politik të përcaktuar thellësisht (DDPG) bazuar në mësimin përforcues të thellë (DRL) për të përmirësuar performancën e kontrollit të sistemit. Politika optimale e kontrollit e algoritmit DDPG rrjedh nga procesi i vendimmarrjes Markov dhe korniza Aktor-Kritik, me modifikime për të përmirësuar konvergjencën dhe stabilitetin. Duke rregulluar kapacitetin dhe bazën e përvojës së algoritmit DDPG, sistemi është në gjendje të shmangë optimizimet lokale dhe të përshpejtojë trajnimin. Për më tepër, struktura e rrjetit Kritik është përmirësuar për të zbutur mbivlerësimin e vlerës Q, duke shkurtuar kështu periudhën e konvergjencës me norma të ulëta të të mësuarit. Për të vërtetuar efektivitetin e qasjes së propozuar, një sistem kontrolli për lavjerrësin e vetëm të përmbysur (SIP) u zhvillua dhe u testua fillimisht në një mjedis simulimi, dhe më pas u testua në laboratorin e Mekatronikës, në Fakultetin e Inxhinierisë Mekanike të Universitetit të Prishtinës. Analiza krahasuese me kontrolluesit tradicionalë dhe qasjen standarde DDPG demonstron se kontrolluesi i përmirësuar DDPG arrin përgjigje më të shpejtë ndaj shqetësimeve, zvogëlim të zhvendosjes dhe devijimeve këndore, si dhe stabilitet superior. Ky kërkim gjithashtu ilustron potencialin e kombinimit të teknologjive të të mësuarit të thellë dhe të mësimit përforcues për të zgjidhur probleme komplekse të kontrollit, duke ofruar një zgjidhje të vlefshme për kontrollin inteligjent autonom në sisteme si kontrolli i shpejtësisë së motorit DC dhe stabilizimi i lavjerrësit të përmbysur. Rezultatet theksojnë aftësitë e përmirësuara të qëndrueshmërisë dhe konvergjencës së algoritmit të përmirësuar DDPG, duke ofruar njohuri të vlefshme për aplikimin e mësimit përforcues (RL) në sistemet tradicionale të kontrollit.

Fjalë kyçe: Gradient Politik i Përcaktuar Thellësisht (DDPG), Mësim Përforcues (RL), Sistemet e Kontrollit, Mësim i Thellë (DL), Sistemet Dinamike

Abstract

The fast progress of artificial intelligence has led to the increased use of intelligent control algorithms in modern control systems, meeting the growing need for high-performance solutions. This thesis presents the modification of a deep deterministic policy gradient (DDPG) controller based on deep reinforcement learning to improve system control performance. The DDPG algorithm's best control policy comes from the Markov decision process and the Actor-Critic framework, with changes made to improve convergence and stability. By adjusting the capacity and experience pool of the DDPG algorithm, the system can avoid getting stuck in local optima and can speed up training. Additionally, the Critic network structure is improved to reduce the overestimation of the Q-value, which shortens the convergence period at low learning rates. To test the effectiveness of the proposed approach, a control system for a single inverted pendulum was developed and first tested in a simulation environment, and then it was tested in the Mechatronics Laboratory at the Faculty of Mechanical Engineering, University of Prishtina. A comparative analysis with traditional controllers and the standard DDPG approach shows that the improved DDPG controller responds faster to disturbances, reduces displacement and angular deviations, and has better stability. This research also shows the potential of combining Deep Learning and Reinforcement Learning techniques to solve complex control problems, providing a practical solution for intelligent autonomous control in systems like DC motor speed control and inverted pendulum stabilization. The results show the improved robustness and convergence abilities of the DDPG algorithm, giving valuable insights into how reinforcement learning can be applied in traditional control systems.

Keywords: Deep Deterministic Policy Gradient, Reinforcement Learning, Control Systems, Deep Learning, Dynamical Systems

Përmbajtja

1. Hyrje 9

1.1 Qëllimet 9

1.2 Përmbledhje e tezës 9

2. Reinforcement Learning 11

2.1 Bazat e teorisë së rregullimit 12

2.1.1 Sistemet lineare të rregullimit 12

2.1.2 Sistemet jolineare të rregullimit 14

2.2 Reinforcement Learning për aplikimet optimale të rregullimit 14

2.3 Bazat e Reinforcement Learning 15

2.3.1 Prova-gabimi-të mësuarit 16

2.3.2 Kthimi 17

2.3.3 Procesi i Vendimmarrjes Markov 18

2.3.4 Funksioni i Vlerës 18

2.3.5 Ekuacioni i Bellman-it 19

2.3.6 Pritshmëria 20

2.3.7 Eksplorimi kundrejt shfrytëzimit 20

2.3.8 Qasjet ndaj RL 22

2.3.9 Qasja e bazuar në model: Programimi Dinamik 24

2.3.10 Qasjet pa model 28

2.3.11 Mësimi me Diferencë Temporale 29

2.4 Përfundimi: Disavantazhet e Mësimit Përforcues (RL) 31

3. Deep Reinforcement Learning 33

3.1. Bazat e Deep Learning 33

3.1.1. Rrjetet Neurale Artificiale (ANNs) 33

3.1.2. Procesi i të mësuarit 35

3.2. Algoritmet i Policy Gradient 38

3.2.1. Derivimi i Policy Gradient (PG) 38

3.2.2. Arkitektura aktor-kritik 39

3.3. Deep Q-learning 42

3.4. Deep Deterministic Policy Gradient 43

3.4.1. Përsëritja e Buffers 44

3.4.2. Rrjetet e Objektivit 44

3.4.3. Eksplorimi 45

3.5. Përfundimi: Diskutim mbi RL 45

4. Zbatimi i DDPG duke përdorur RL MATLAB Toolbox 47
4.1. Lavjerrësi i vetëm i përmbysur 47
4.1.1. Problemi i kontrollit të pozicionit 48
4.2.3. Skema e rregullimit 57
4.2.4. Sinjalet e vëzhgimit dhe veprimit 58
4.2.5. Sinjali i shpërblimit 59
4.2.6. Hiperparametrat dhe arkitekturat e rrjetit 59
4.2.7. Performanca 60
4.3. Diskutim mbi trajnimin RL 62
5. Kontrolli i Stabilizimit 64
5.1. Zbatimi në Kohë Reale 64
5.1.1. Aparatura 64
5.1.2. Specifikimet e Dizajnit 65
5.2. Rezultatet eksperimentale 65
5.2.1. Matricat e Peshës së Qëndrueshme 66
6. Përfundimet 76
7. Conclusions 77
8. Referencat 78

Lista e Figurave

Figura 2.1 Cikli i ndërveprimit agjent-mjedis tipik i të mësuarit përforcues (RL) 11
Figura 2.2 Bllok diagrami që përfaqëson një sistem rregullimi të mbyllur 12
Figura 2.3 Sistemi njëdimensional masë-sustë-amortizator 13
Figura 2.4 Të mësuarit përforcues (RL) kundrejt rregullimit tradicional 15
Figura 2.5 Cikli provë-gabim-mësim 17
Figura 2.6 Përgjigja e epsilonit gjatë trajnimit 21
Figura 2.7 Krahasimi i pendesës në lidhje me metodat e eksplorimit 22

Figura 3.1 Shembull i një modeli të thjeshtë për një neuron të vetëm [5] 34
Figura 3.2 Shembull i thjeshtë që tregon një rrjet të thellë neural me katër shtresa [5] 35
Figura 3.3 Grafiku i funksioneve të aktivizimit: (a) Sigmoid; (b) Tanh; (c) ReLU; (d) LeakyReLU 37
Figura 3.4 Faza e mësimit Aktor-Kritik: Hapi 1 40
Figura 3.5 Faza e mësimit Aktor-Kritik: Hapi 2 40
Figura 3.6 Faza e mësimit Aktor-Kritik: Hapi 3 41
Figura 3.7 Faza e mësimit Aktor-Kritik: Hapi 4 41
Figura 3.8 Faza e mësimit Aktor-Kritik: Hapi 5 42

Figura 4.1 Quanser IP02 Lavjerrësi i vetëm i përmbysur [29] 48
Figura 4.2 Skema lineare e lavjerrësit të përmbysur 49
Figura 4.3 Skema elektrike e motorit standard DC 54
Figura 4.4 Konvertimi i sistemit referues të Lavjerrësit të Vetëm të Përmbysur 58
Figura 4.5 Skema e përgjithshme e kontrollit 58
Figura 4.6 Këndi i Lavjerrësit të Përmbysur [deg] 61
Figura 4.7 Tensioni në hyrje të motorit [V] 61

Figura 5.1 Diagrami i vendosjes eksperimentale 64
Figura 5.2 Lavjerrësi i vetëm i përmbysur i montuar në një sistem servo Quanser IP02 65
Figura 5.3 Sekuencia e Ngritjës dhe Stabilizimit të Lavjerrësit të Vetëm të Përmbysur (SIP) Quanser IP02 66
Figura 5.4 Përparimi i trajnimit për peshat Q=diag(0.75,4,0,0), R=0.0003 68
Figura 5.5 Tensioni në hyrje të motorit për peshat Q=diag(0.75,4,0,0), R=0.0003 68
Figura 5.6 Pozicioni i karrocës (m) për peshat Q=diag(0.75,4,0,0), R=0.0003 69
Figura 5.7 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(0.75,4,0,0), R=0.0003 69
Figura 5.8 Përparimi i trajnimit për peshat Q=diag(5,50,0,0), R=0.002 70
Figura 5.9 Tensioni në hyrje të motorit për peshat Q=diag(5,50,0,0), R=0.002 70
Figura 5.10 Pozicioni i karrocës (m) për peshat Q=diag(5,50,0,0), R=0.002 71
Figura 5.11 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(5,50,0,0), R=0.002 71
Figura 5.12 Përparimi i trajnimit për peshat Q=diag(10,20,0,1), R=0.01 72
Figura 5.13 Tensioni në hyrje të motorit për peshat Q=diag(10,20,0,1), R=0.01 72
Figura 5.14 Pozicioni i karrocës (m) për peshat Q=diag(10,20,0,1), R=0.01 73
Figura 5.15 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(10,20,0,1), R=0.01 73
Figura 5.16 Përparimi i trajnimit për peshat Q=diag(800,150,1,1), R=0.1 74
Figura 5.17 Tensioni në hyrje të motorit për peshat Q=diag(800,150,1,1), R=0.1 74
Figura 5.18 Pozicioni i karrocës (m) për peshat Q=diag(800,150,1,1), R=0.1 75
Figura 5.19 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(800,150,1,1), R=0.1 75

Lista e Tabelave

Tabela 2.1 Përmbledhje e Iteracionit të Politikës kundrejt Iteracionit të Vlerave 28
Tabela 2.2 Krahasimi ndërmjet Q-learning dhe SARSA 31

Tabela 4.1 Parametrat fizikë të Lavjerrësit të Vetëm të Përmbysur (SIP) 56
Tabela 4.2 Hiperparametrat dhe arkitekturat e rrjetit [33] 60

Shkurtesat

AI	:	*Artificial Intelligence*
ANN	:	*Artificial Neural Network*
COG	:	*Centre Of Gravity*
DDPG	:	*Deep Deterministic Policy Gradient*
DL	:	*Deep Learning*
DNN	:	*Deep Neural Network*
DQN	:	*Deep Q-Network*
DRL	:	*Deep Reinforcement Learning*
DP	:	*Dynamic Programming*
EOM	:	*Equations Of Motion*
HJB	:	*Hamilton-Jacobi-Bellman*
LQR	:	*Linear Quadratic Regulator*
LQT	:	*Linear Quadratic Tracker*
LTI	:	*Linear Time-Invariant*
MDP	:	*Markov Decision Process*
ML	:	*Machine Learning*
MPC	:	*Model Predictive Control*
MSE	:	*Mean-Squared Error*
PG	:	*Policy Gradient*
PPO	:	*Proximal Policy Optimization*
RL	:	*Reinforcement Learning*
SARSA	:	*State-Action-Reward-next State-next Action*
SIP	:	*Single Inverted Pendulum*
TD	:	*Temporal Difference*
VI	:	*Value Iteration*

1. Hyrje

Teoria e rregullimit optimal, një disiplinë matematikore e mirëpërcaktuar, identifikon strategjitë optimale të rregullimit për sistemet dinamike duke optimizuar funksionet e kostos që kapin objektivat specifike të dizajnit. Një qasje kryesore për zgjidhjen e këtyre problemeve është Programimi Dinamik (DP), i cili siguron optimalitetin duke zgjidhur ekuacionin diferencial të pjesshëm Hamilton-Jacobi-Bellman (HJB). Metodat tradicionale të rregullimit optimal funksionojnë offline dhe kërkojnë njohuri të plota të dinamikës së sistemit. Inteligjenca Artificiale (AI) është bërë instrumentale në mundësimin e autonomisë adaptive, duke përfshirë zhvillimin e programeve kompjuterike që shfaqin sjellje inteligjente. Softuerët që demonstrojnë funksione njohëse si perceptimi, planifikimi dhe të mësuarit konsiderohen pjesë e AI-së. Machine Learning (ML), një nënfushë e AI-së, përqendrohet në zhvillimin e algoritmeve që zgjidhin probleme duke nxjerrë njohuri nga të dhënat. ML kategorizohet në tre lloje kryesore bazuar në riveprimin e të dhënave: Mësim i Mbikëqyrur, Mësim i Pambikëqyrur dhe Mësim Përforcues (RL). Në Mësimin e Mbikëqyrur, algoritmet përdorin të dhëna trajnimi të etiketuar për të ndërtuar modele që përgjithësojnë marrëdhëniet hyrje-dalje. Në të kundërtën, Mësimi i Pambikëqyrur përfshin zbulimin e modeleve të fshehura në të dhëna pa etiketa pa ndonjë riveprim. Mësimi Përforcues (RL) përfshin një agjent që merr vendime me kalimin e kohës për të optimizuar performancën afatgjatë, me veprime të ndërmarra në një cikël të mbyllur të ndikuar nga rezultatet e mëparshme. RL trajton problemet e vendimmarrjes sekuenciale në fusha të ndryshme, përfshirë rregullimin, kërkimin operacional, ekonominë dhe mjekësinë, duke gjetur zgjidhje për ekuacionin HJB në kohë reale. Kjo përshtatshmëri në kohë reale ka shtyrë studiuesit të zhvillojnë kontrollorë të bazuar në RL për të arritur rregullim optimal autonom.

1.1 Qëllimet

Objektivi kryesor i kësaj teze masteri është të shqyrtoj aplikimin e Deep Reinforcement Learning në dizajnin e rregullimit të sistemeve dinamike duke përdorur Reinforcement Learning Matlab Toolbox. Për të arritur këtë qëllim të përgjithshëm, janë përcaktuar disa detyra specifike:

1. Rishikimi i literaturës ekzistuese mbi Deep Reinforcement Learning me qëllim identifikimin e një algoritmi të përshtatshëm për aplikimet e rregullimit;
2. Fitimi i aftësive në përdorimin e Reinforcement Learning Matlab Toolbox;
3. Zhvillimi dhe zbatimi i një kontrolluesi me rregullim efektiv për të dyja sistemet lineare dhe jolineare si një implementim referencial.

1.2 Përmbledhje e tezës

Pas kapitullit hyrës, kjo tezë eksploron dy koncepte kryesore: Mësimi Përforcues (en. Reinforcement Learning) dhe Mësimi i Thellë (en. Deep Learning), duke kulmuar në aplikimin e tyre në sistemet dinamike duke përdorur Matlab. Struktura e tezës është si vijon:

1. **Hyrje**: Shpjegon prapavijën dhe objektivat e tezës.
2. **Reinforcement Learning**: Ofron një përmbledhje të plotë të gjendjes së artit në Mësimin Përforcues, duke pajisur lexuesin me mjete thelbësore për të hyrë në këtë fushë kërkimi dhe duke krahasuar terminologjin me rregullimin tradicional.
3. **Deep Reinforcement Learning**: Diskuton përdorimin e Rrjetave Neurale për Mësimin Përforcues, me një fokus të veçantë në algoritmin Deep Deterministic Policy Gradient (DDPG) për sistemet me kohë të vazhdueshme.

4. **Implementimi i DDPG duke përdorur RL MATLAB Toolbox**: Përshkruan dizajnin e sistemeve të kontrollit linear dhe jolinear për të kryer eksperimente të mësimit përforcues në aplikacione reale, siç janë modeluar në një mjedis të simuluar duke përdorur Reinforcement Learning Matlab Toolbox dhe Toolbox të rregullimit në kohë reale Quanser.
5. **Përmirësimi i të mësuarit bazuar në funksionin e shpërblimit plotësues për DLR**: Propozon një metodë për të gjeneruar të dhëna të marra në mënyrë efikase gjatë fazës së të mësuarit, duke përshpejtuar procesin dhe promovuar konvergjencën drejt zgjidhjeve optimale.
6. **Përfundime**: Përmbledh rezultatet e marra nga eksperimentet, duke ofruar disa njohuri dhe reflektime.
7. **Perspektivat**: Ofron përfundime nga perspektiva e punës së mundshme të ardhshme.

2. Reinforcement Learning

Mësimi përforcues (RL) ofron algoritme të fuqishme që janë të afta të përcaktojnë rregulluesit optimalë për sisteme lineare dhe jolineare, edhe në skenarë ku dinamikat janë përcaktuese, stokastike, të panjohura ose shumë të pasigurta. RL funksionon si një kuadër pa model, i aftë për të adresuar problemet e kontrollit optimal të formuluara si Procese Vendimmarrëse Markoviane (MDP) [1]. Ky kapitull kryesisht eksploron qasjet e AI-së ndaj RL nga perspektiva e inxhinierisë së kontrollit. Fillimisht, do të diskutohen konceptet themelore të teorisë së rregullimit dhe RL, veçanërisht ato që kanë të bëjnë me rregullimin tradicional, për të krijuar një kuptim themelor. Aftësia e RL për të menaxhuar sisteme dinamike komplekse pa kërkuar modele të qarta e bën atë jashtëzakonisht të vlefshme në fusha si robotika, automjetet autonome dhe sistemet adaptive të rregullimit. Duke shfrytëzuar RL, inxhinierët mund të dizajnojnë kontrollues që mësojnë politika optimale përmes ndërveprimit me mjedisin, duke përmirësuar vazhdimisht performancën bazuar në reagimet e marra. Përshtatshmëria e RL shtrihet në aplikacione të ndryshme, përfshirë automatizimin industrial, kujdesin shëndetësor dhe financat, ku vendimmarrja nën pasiguri është thelbësore. Për më tepër, përparimet e fundit në Deep Reinforcement Learning (DRL), i cili integron teknikat e Deep Learning me RL, kanë rritur ndjeshëm aftësinë për të trajtuar hapësira të mëdha të gjendjes dhe veprimit [2]. DRL ka treguar rezultate premtuese në detyra komplekse si luajtja e video lojërave, kontrollimi i krahëve robotikë dhe optimizimi i konsumit të energjisë në rrjete të zgjuara. Ndërsa thellohemi më shumë në qasjet e AI-së ndaj RL, bëhet e qartë se ky ndryshim paradigmatik në inxhinierinë e rregullimit ka potencialin të revolucionarizojë mënyrën se si ne dizajnojmë dhe zbatojmë sistemet e rregullimit.

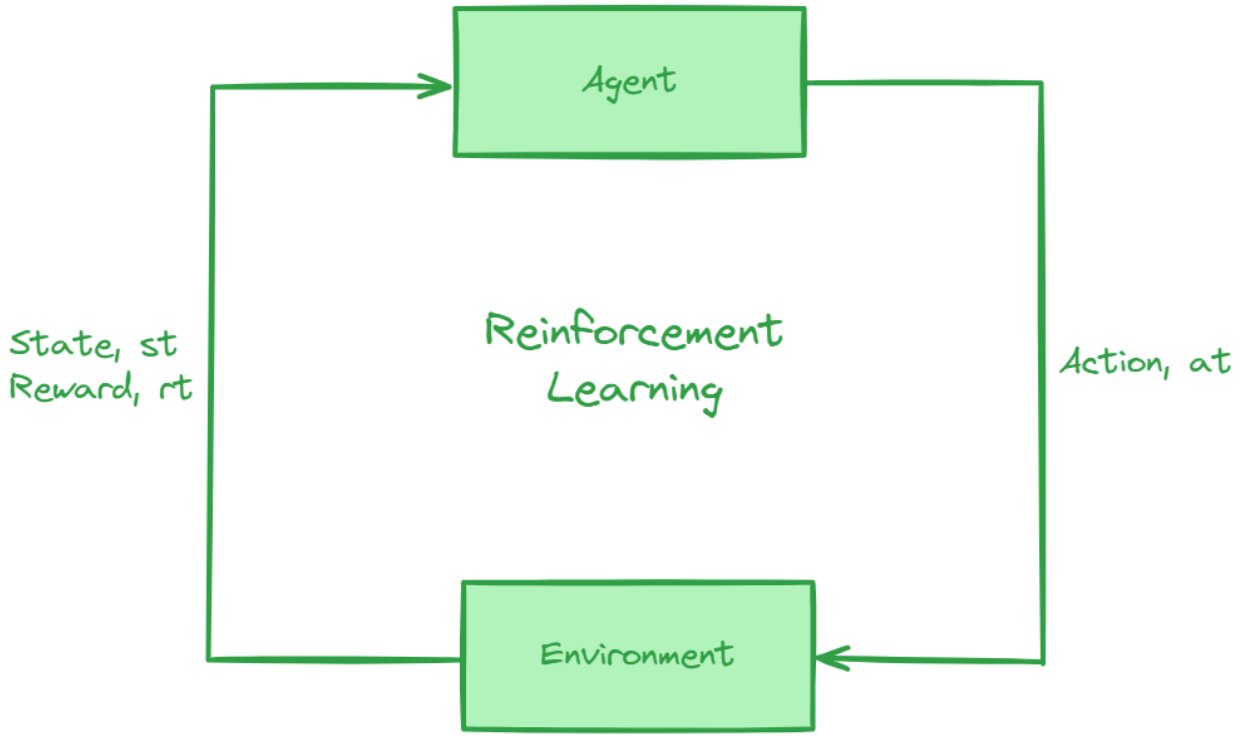


Figura 2.1 Cikli i ndërveprimit agjent-mjedis tipik i të mësuarit përforcues (RL)

2.1 Bazat e teorisë së rregullimit

Fusha e matematikës dhe inxhinierisë e njohur si rregullimi automatik studion se si të manipulojnë variablat hyrëse për të ndikuar në sjelljen e një sistemi përmes një hyrjeje komanduese. Konkretisht, një sistem rregullimi përbëhet nga një grup pajisjesh që menaxhojnë, komandojnë, drejtojnë ose rregullojnë sjelljen e pajisjeve të tjera për të arritur një performancë të dëshiruar. Me fjalë të thjeshta, një sistem rregullimi është një sistem që drejton sisteme të tjera për të arritur një gjendje specifike. Sistemet e rregullimit mund të kategorizohen gjerësisht në *sisteme rregullimi linearë* dhe *sisteme rregullimi jolinearë*. Sistemet e rregullimit linearë respektojnë parimet e homogjenitetit dhe aditivitetit, duke siguruar përgjigje të parashikueshme dhe proporcionale ndaj ndryshimeve të hyrjes. Në të kundërt, sistemet e rregullimit jolinearë nuk ndjekin këto parime dhe mund të shfaqin sjellje më komplekse.

Në këtë tezë, fokusi është vetëm në *sistemet e rregullimit me riveprim* (en. feedback). Një sistem rregullimi me riveprim kontrollon daljen e tij duke përdorur matjen e daljes si një sinjal riveprimi. Ky sinjal riveprimi krahasohet me një *sinjal reference* për të gjeneruar një *sinjal gabimi të rregullimit*, i cili më pas përpunohet nga një kontrollues për të gjeneruar hyrjen e rregullimit të sistemit. Sistemi i rregullimit, i njohur si kombinimi i procesit dhe aktuatorit, ka daljen e tij *y* të matur duke përdorur sensorë. Bllok diagrami në figurën 2.2 ilustron një sinjal të përgjithshëm riveprimi.

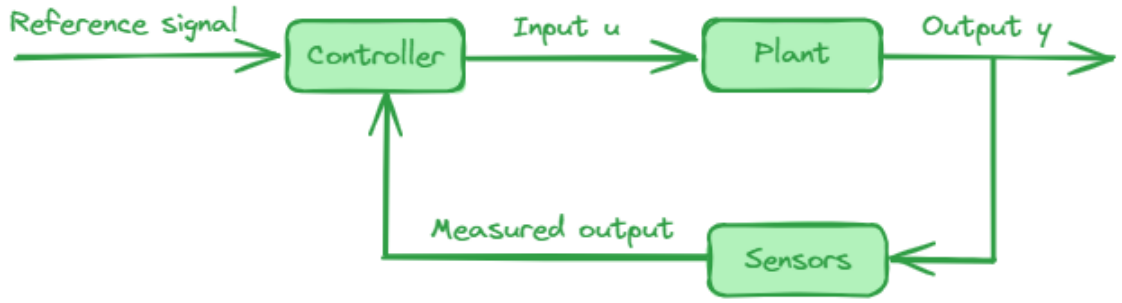


Figura 2.2 Bllok diagrami që përfaqëson një sistem rregullimi të mbyllur

Objektivat kryesore të rregullimit me riveprim janë të sigurojnë që variablat me interes në një proces ose sistem, të shikohen si sinjalet e daljes, ose të ndjekin trajektoret e referencës ose të mbahen afër pikave të caktuara të tyre. Riveprimi është thelbësor për tre arsye kryesore:

1. Për të kundërvepruar sinjalet e ndërhyrjes që ndikojnë në daljen.
2. Për të përmirësuar performancën e sistemit në prani të pasigurisë së modelit.
3. Për të stabilizuar një sistem të paqëndrueshme.

Përmes përdorimit të riveprimit, sistemet e rregullimit mund të arrijnë nivele të dëshiruara të performancës, pavarësisht pranisë së ndërhyrjeve dhe pasigurive.

2.1.1 Sistemet lineare të rregullimit

Sistemet lineare të rregullimit janë thelbësore në inxhinierinë e rregullimit dhe karakterizohen nga varësia e tyre nga ekuacionet diferenciale lineare. Këto ekuacione janë lineare në variablat e varur, derivatet e tyre në lidhje me variablin e pavarur (zakonisht koha) dhe funksionin e hyrjës ose

rregullimit. Sistemet lineare rregullohen nga parime kyçe të tilla si aditiviteti, homogjeniteti dhe superpozicioni:

- **Additiviteti:** Nëse një hyrje x_1 çon në një dalje y_1 dhe një hyrje x_2 çon në një dalje y_2, atëherë shuma e secilës $x_1 + x_2$ çon në shumën e çdo dalje $y_1 + y_2$.
- **Homogjeniteti:** Nëse një hyrje x çon në një dalje y, atëherë një hyrje e shkallëzuar kx (ku k është një konstant) çon në një dalje të shkallëzuar ky.
- **Superpozicioni:** Nëse një hyrje x_1 çon në një dalje y_1 dhe një hyrje x_2 çon në një dalje y_2, atëherë një kombinim i hyrjeve $k_1x_1 + k_2x_2$ (ku k_1 dhe k_2 janë konstante) çon në një kombinim të daljeve $k_1y_1 + k_2y_2$.

Një shembull ilustrues i një sistemi linear është sistemi njëdimensional masë-sustë-amortizator, siç tregohet në figurën 2.3.

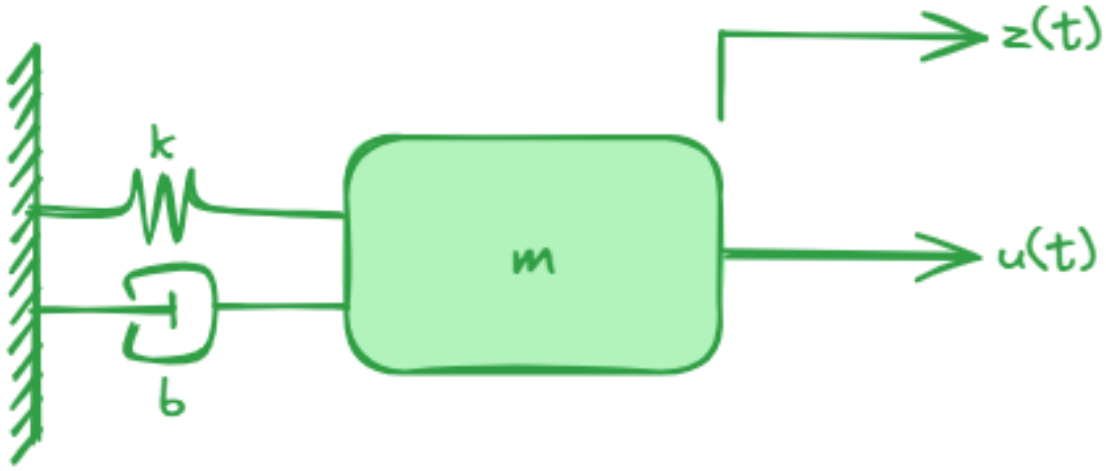


Figura 2.3 Sistemi njëdimensional masë-sustë-amortizator

Dinamika e këtij sistemi përshkruhet nga ekuacioni diferencial:

$$m\ddot{z} + b\dot{z} + k = u(t) \tag{2.1}$$

Në këtë ekuacion, m përfaqëson masën, b koeficientin e amortizimit, k konstantën e sustës dhe $u(t)$ forcën e jashtme të aplikuar në sistem. Duke prezantuar variablat $x_1 = z$ dhe $x_2 = \dot{z}$, ne mund të përfaqësojmë sistemin në formën e hapësirës së gjendjes:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u(t) \tag{2.2}$$

Përfaqësimi i përmbledhur i hapësirës së gjendjes për këtë sistem është:

$$\dot{x} = Ax + Bu \tag{2.3}$$

ku x është vektori i gjendjes, u është vektori i rregullimit të hyrjes, A është matrica e dinamikës së sistemit, dhe B është matrica e hyrjes. Dalja y e sistemit mund të shprehet si një kombinim linear i variablave të gjendjes:

$$y = Cx + Du \tag{2.4}$$

Këtu, C është matrica e daljes që përcakton cilat variabla të gjendjes maten, dhe D është matrica kthyese që lidh hyrjen direkt me daljen. Nëse matricat A, B, C, dhe D mbeten konstante me kalimin e kohës, sistemi klasifikohet si linear i pandryshueshëm në kohë (LTI).

2.1.2 Sistemet jolineare të rregullimit

Sistemet e rregullimit jolinear karakterizohen nga paaftësia e tyre për të respektuar parimin e superpozicionit, që do të thotë se dalja e tyre nuk është proporcionalisht e drejtpërdrejtë me hyrjen e tyre. Në inxhinieri, sistemet jolineare luajnë një rol thelbësor në teorinë e rregullimit sepse të gjitha sistemet praktike shfaqin sjellje jolineare në një masë të caktuar. Kjo jolinearitet shpesh është për shkak të faktorëve të tillë si saturimi, ku përgjigja e një sistemi bëhet e kufizuar për shkak të kufizimeve fizike, duke e parandaluar atë të japë energji të pafundme.

Një shembull i shkëlqyer i jolinearitetit në sistemet e rregullimit është kushti i saturimit, i cili ndodh sepse sistemet reale kanë kufij në energjinë që ato mund të prodhojnë ose të thithin. Ky sjellje kërkon zhvillimin dhe aplikimin e strategjive të specializuara të rregullimit të përshtatura për sistemet jolineare, duke siguruar performancë të saktë dhe efikase.

Ekuacionet e gjendjes dhe të daljes për sistemet jolineare zakonisht shprehen si më poshtë:

$$x(t) = f[x(t), u(t)]$$

$$y(t) = g[x(t), u(t)]$$

Në këto ekuacione, $x(t)$ përfaqëson vektorin e gjendjes, $u(t)$ hyrjen e rregullimit, $y(t)$ daljen, dhe f dhe g janë funksione jolineare që përshkruajnë dinamikën e sistemit dhe marrëdhëniet e daljes, përkatësisht. Këto ekuacione theksojnë kompleksitetin dhe nevojën për teknika të avancuara të rregullimit për të menaxhuar sjelljet jolineare në aplikimet praktike.

2.2 Reinforcement Learning për aplikimet optimale të rregullimit

Qëllimi kryesor i një sistemi të rregullimit është të përcaktojë hyrjet optimale të rregullimit për të arritur sjelljen e dëshiruar të një sistemi. Sistemet e rregullimit me riveprim përdorin dalje të matura si riveprim për të përmirësuar performancën dhe për të korrigjuar shqetësimet dhe gabimet e rastësishme. Inxhinierët dizajnojnë kontrollet për të përmbushur kërkesat e sistemit duke përdorur riveprimin e sistemit të rregullimit së bashku me një model të kombinuar të procesit dhe aktuatorit.

Mësimi i përforcimit (RL) ka një qëllim të ngjashëm me sistemet e rregullimit, por përdor një qasje të ndryshme. Një agjent RL mëson një politikë, e përfaqësuar nga një sekuencë veprimesh ose hyrjesh rregullimi, për të rregulluar një mjedis dinamik dhe për të rrit një funksion objektiv. Për shembull, merrni parasysh detyrën e parkimit të një automjeti duke përdorur një sistem drejtimi të automatizuar. Këtu, kompjuteri i automjetit (agjenti) duhet ta parkojë makinën në orientimin dhe vendndodhjen e duhur.

Kontrolluesi gjeneron hyrjet e rregullimit të drejtimit, frenimit dhe përshpejtimit (veprime) duke përdorur të dhëna nga kamerat, akselerometrat, xhiroskopët, një marrës GPS dhe lidar (vëzhgime). Këto hyrje rregullimi dërgohen te aktuatorët që rregullojnë automjetin. Vëzhgimet e rezultuara varen nga aktuatorët, sensorët, dinamika e automjetit, sipërfaqja e rrugës, era dhe shumë faktorë të tjerë më pak të rëndësishëm.

Ky formulim synon të tërheqë vëmendjen e ekspertëve të teorisë së rregullimit, pasi ajo përshtatet ngushtë me objektivat e fushës së tyre. Ndërsa teoria e rregullimit është kryesisht e bazuar në modele, ML është kryesisht i bazuar në të dhëna. Figura 2.4 ilustron një përfaqësim të përgjithshëm të bllok sistemit të rregullimit, i cili mund të përkthehet në një kornizë RL:

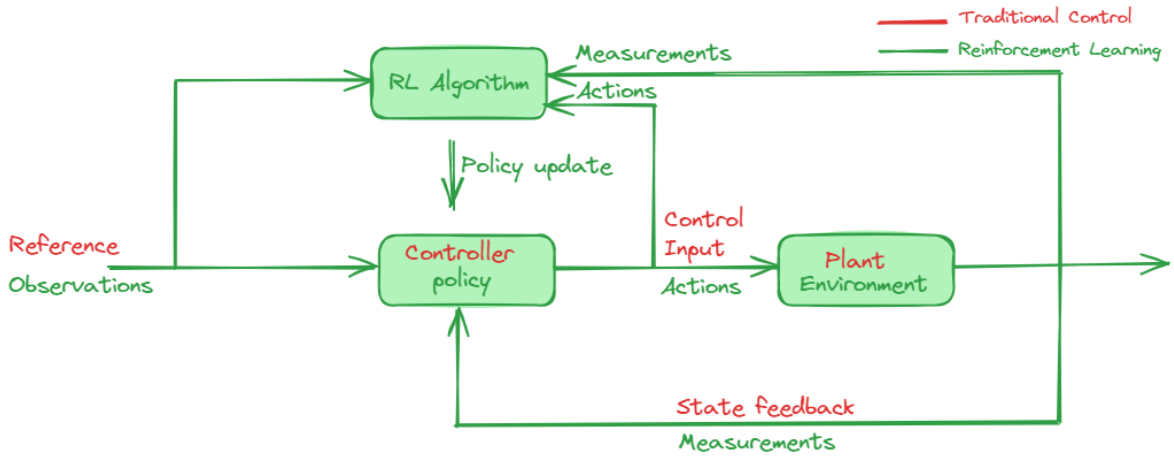


Figura 2.4 Të mësuarit përforcues (RL) kundrejt rregullimit tradicional

Nga perspektiva e RL, kombinimi i procesit dhe aktuatorit (plant), sinjali referues dhe llogaritja e gabimit konsiderohen pjesë e *mjedisit* (en. environment). Mjedisi mund të përfshijë gjithashtu elementë të tillë si zhurma e matjes, sinjalet e shqetësimit, filtrat dhe konverterët. *Vëzhgimet* (en. observation) janë vlera të matshme nga mjedisi që janë të dukshme për agjentin, në varësi të senzorëve të disponueshëm. Këto vëzhgime përdoren në algoritmet RL për të gjetur sekuencën optimale të hyrjeve të rregullimit.

Hyrjet e rregullimit janë *veprime* (en. action) të ndërmarra nga kontrolluesi. Sekuenca e veprimeve, e njohur si *politikë* (en. policy) (π), mësohet nga *agjenti* (en. agent) duke përdorur metoda të mësimit të përforcimit. Funksioni i *shpërblimit* (en. reward), i ngjashëm me një funksion kostoje në teorinë e rregullimit, mat performancën e agjentit, duke e ndihmuar atë të kuptojë nëse veprimet e tij janë në përputhje me performancën e dëshiruar të sistemit. Për shembull, funksionet e shpërblimit mund të dizajnohen për të minimizuar gabimin e gjendjes së qëndrueshme duke ulur njëkohësisht përpjekjet e rregullimit [3, 4, 5, 6, 7].

2.3 Bazat e Reinforcement Learning

Mësimi i përforcimit (RL) është një degë e Machine Learning (ML) që merret me problemin e marrjes së vendimeve sekuenciale. Ndryshe nga metodat tradicionale të marrjes së vendimeve që mund të mbështeten në një veprim të vetëm, RL thekson nevojën për një sekuencë veprimesh gjatë kohës për të arritur një rezultat të dëshiruar. Kjo është veçanërisht e dobishme në mjedise dinamike ku vendimet e menjëhershme ndikojnë në gjendjet dhe shpërblimet e ardhshme.

Korniza e RL është projektuar për të trajtuar probleme ku zgjidhja optimale kërkon konsiderimin e përfitimeve afatgjata dhe jo vetëm të përfitimeve të menjëhershme. Kjo përfshin një agjent që ndërvepron me një mjedis përmes veprimeve, merr riveprime në formën e shpërblimeve dhe rregullon politikën e tij për të rrit shpërblimin kumulativ me kalimin e kohës. Konceptet themelore në RL përfshijnë agjentin, mjedisin, gjendjet, veprimet, shpërblimet dhe politikën.

Ky seksion ofron një përmbledhje të plotë të këtyre koncepteve themelore në RL, duke u ofruar lexuesve një kuptim të fortë për të qasur dhe zbatuar algoritme të ndryshme RL. Algoritmet kryesore në këtë fushë përfshijnë Q-learning, Deep Q-Networks (DQN) dhe metodat e Policy Gradient. Secili prej këtyre algoritmeve ofron strategji unike për të balancuar eksplorimin dhe shfrytëzimin, duke siguruar që agjenti jo vetëm të shfrytëzojë informacionin e njohur, por edhe të zbulojë strategji të reja për të përmirësuar performancën.

Mësimi i përforcimit (RL) është i aplikueshëm gjerësisht në fusha të ndryshme, përfshirë robotikën, lojërat dhe modelimin financiar. Duke kuptuar bazat, studiuesit dhe praktikuesit mund të zhvillojnë sisteme inteligjente të afta për të mësuar dhe përshtatur në mjedise komplekse dhe të pasigurta [3, 4, 6, 7].

2.3.1 Prova-gabimi-të mësuarit

Mësimi i përforcimit (RL) përdor cikle të mësimit përmes provave dhe gabimeve për të rrit shpërblimin total që një agjent merr nga mjedisi. Kjo rritje mund të shihet si zvogëlim i funksionit të kostos, i ngjashëm me teorinë optimale të rregullimit, pasi funksioni i shpërblimit është formuluar bazuar në problemin e rregullimit dhe kërkesat e sistemit.

Mjedisi përbëhet nga variabla që lidhen me problemin dhe sistemin. Kombinimi i të gjitha vlerave të mundshme që këto variabla mund të marrin njihet si hapësira e gjendjes, e shënuar si S. Një grup specifik vlerash të vëzhguara në çdo kohë t është një gjendje s_t. Agjentët mund ose nuk mund të kenë qasje në gjendjen aktuale s_t të mjedisit, në varësi të disponueshmërisë së senzorëve. Variablat që agjenti mund të perceptojë në çdo kohë t quhen vëzhgime. Kombinimi i të gjitha vlerave të mundshme të këtyre variablave është hapësira e vëzhgimeve, e shënuar si O. Gjendja dhe vëzhgimi përdoren shpesh ndërsjelltas në komunitetin RL, pasi agjentët zakonisht lejohen të shohin gjendjen e brendshme të mjedisit, megjithëse kjo nuk është gjithmonë e vërtetë.

Në çdo gjendje s_t, mjedisi ofron një grup veprimesh at që agjenti mund të zgjedhë. Grupi i të gjitha veprimeve të mundshme në çdo gjendje quhet hapësira e veprimit, e shënuar si A. Figura 2.5 ilustron grafikisht ndërveprimin midis agjentit dhe mjedisit, i cili përsëritet sekuencialisht me kalimin e kohës. Në çdo hap kohor t, agjenti vëzhgon një gjendje $s_t \in S$, pastaj zgjedh një veprim $a_t \in A$ bazuar në politikën e tij, që përfaqëson sjelljen e agjentit në kohën t.

Politika mund të jetë *determinuese* ose *stohastike* dhe zakonisht shënohet si $\pi(a_t|s_t)$:

$$\pi(s_t, a_t) = P[a_t|s_t] \tag{2.4}$$

ku politika π ofron një probabilitet $P[a_t|s_t]$ për të ndërmarrë një veprim në mjedise stohastike ose zgjedh drejtpërdrejt veprimin në ato determinuese. Agjenti pastaj merr një vëzhgim të ri nga mjedisi në hapin tjetër kohor s_{t+1}, së bashku me një shpërblim skalar r_{t+1} të përcaktuar në ekuacionin 2.5, dhe gjendjen tjetër s_{t+1}. Shpërblimi skalar përfaqëson përfitimin e menjëhershëm pas kalimit nga gjendja s_t në s_{t+1} dhe ndërmarrjen e një veprimi të caktuar a_t.

$$r_{t+1} = \rho(x_t, u_t) \tag{2.5}$$

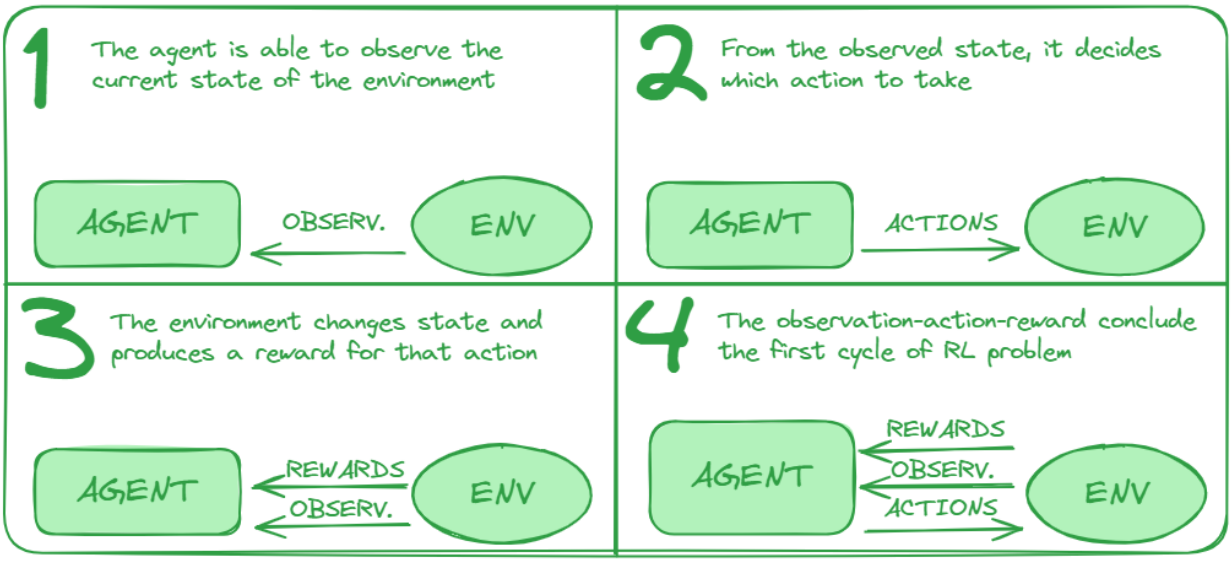


Figura 2.5 Cikli provë-gabim-mësim

Grupi i gjendjes, veprimit, shpërblimit dhe gjendjes së re të vëzhguar nga agjenti në një hap kohor specifik t quhet *përvojë* (2.6):

$$e_t = < s_t, a_t, r_{t+1}, s_{t+1} > \tag{2.6}$$

Çdo përvojë ofron një mundësi për të mësuar dhe për të përmirësuar performancën e agjentit. Detyra që po zgjidhet nga agjenti mund të ketë ose të mos ketë një përfundim natyror. Detyrat me një përfundim natyror, si lojërat, quhen *detyra episodike*. Përkundrazi, detyrat pa përfundim natyror, si mësimi i lëvizjes përpara, quhen *detyra të vazhdueshme*. Sekuenca e hapave kohorë nga fillimi deri në fund të një detyre episodike quhet *episod*. Agjentët mund të kërkojnë disa hapa kohorë dhe episode për të mësuar sjelljen e dëshiruar. E ngjashme me vendimmarrjen njerëzore, marrja e një shpërblimi të konsiderueshëm në një hap kohor specifik t nuk përjashton mundësinë e marrjes së një shpërblimi të vogël menjëherë pas tij, dhe nganjëherë mund të jetë më mirë të injorohet shpërblimi i menjëhershëm për të fituar më të mëdhenj më vonë [3, 4, 6, 7].

2.3.2 Kthimi

Vlera e kthimit G_t është shpërblimi total i grumbulluar që një agjent merr me kalimin e kohës, i përcaktuar matematikisht si vijon (2.7):

$$G_t(\tau) = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad \gamma \in [0,1] \tag{2.7}$$

Këtu, γ përfaqëson faktorin e zbritjes, i cili është thelbësor për të siguruar që vlera e kthimit të mbetet e kufizuar në raste ku shuma e shpërblimeve zgjatet pafundësisht. Ky faktor tregon gjithashtu një preferencë për shpërblimet e menjëhershme mbi ato të ardhshme. Objektivi kryesor i agjentit është të *rrisë* këtë *shpërblim kumulativ*. Në këtë kontekst, τ është përcaktuar si sekuenca e gjendjeve dhe veprimeve brenda mjedisit, siç tregohet në ekuacionin 2.8:

$$\tau = (s_0, a_0, s_1, a_1, \dots) \tag{2.8}$$

Në këtë sekuencë, s_0 përfaqëson daljen e matur ose gjendjen në hapin kohor $t = 0$, ndërsa a_0 shënon hyrjen e rregullimit të aplikuar në sistem ose veprimin e ndërmarrë në hapin kohor $t = 0$. Në mënyrë të ngjashme, a_1 është dalja e matur ose gjendja në hapin kohor $t = 1$, dhe a_1 është hyrja e rregullimit të aplikuar në sistem ose veprimi i ndërmarrë nga agjenti në hapin kohor $t = 1$, dhe kështu me radhë.

2.3.3 Procesi i Vendimmarrjes Markov

Procesi i Vendimmarrjes Markov (MDP) ofron një kornizë të fortë matematikore për modelimin e ndërveprimit ndërmjet një agjenti, politikës së tij dhe mjedisit.

Ai karakterizohet nga tufa (S, A, r, P):

1. Një grup gjendjesh, S, që përfshin të gjitha gjendjet e mundshme të mjedisit, duke përfshirë daljet e matshme diskrete ose të vazhdueshme.
2. Një grup veprimesh të mundshme, $A(s)$, që përfshin të gjitha veprimet ose hyrjet e rregullimit të zbatueshme në çdo gjendje.
3. Një funksion shpërblimi $r_{t+1} = \rho(x_t, u_t)$, që përfaqëson shpërblimin e menjëhershëm të vëzhguar pas kalimit nga gjendja s_t në s_{t+1} përmes veprimit a_t.
4. Një model kalimi, $P(s_{t+1}|s_t, a_t)$, që tregon probabilitetin e kalimit në gjendjen s_{t+1} pas marrjes së veprimit a_t në gjendjen s_t.

Këto gjendje kalimi i përmbahen *pronësisë së Markovit*, që nënkupton se probabiliteti i arritjes në gjendjen s_{t+1} varet vetëm nga gjendja aktuale s_t, pa marrë parasysh asnjë gjendje paraprake. Nëse informacioni i plotë për gjendjen e mjedisit është i qasshëm për agjentin, mjedisi quhet *plotësisht i vëzhgueshëm*; përndryshe, është *pjesërisht i vëzhgueshëm*.

Për më tepër, hapësirat e veprimit dhe të gjendjes mund të jenë diskrete ose të vazhdueshme. Një hapësirë veprimi diskrete përmban një grup të kufizuar veprimesh, ndërsa një hapësirë veprimi e vazhdueshme përfshin një numër të pafund veprimesh të mundshme. Natyra e hapësirës së veprimit ndikon ndjeshëm në qasjen për zgjidhjen e problemit të mësimit të përforcimit (RL) [8].

Identifikimi i natyrës diskrete ose të vazhdueshme të hapësirave të veprimit dhe të gjendjes është thelbësor për të përcaktuar teknikat më të përshtatshme të RL-së. Kjo kuptimësie vendos bazat për trajtimin e problemeve më të avancuara të RL-së, të cilat do të hulumtohen në seksionet në vijim. [9]

2.3.4 Funksioni i Vlerës

Funksioni i vlerës $V_\pi(s_t)$ është vlera e pritur E_π e kthimit të gjendjes s_t kur ndiqet politika π siç përcaktohet në ekuacionin 2.9. Praktikisht, ai përfaqëson shpërblimin e pritur që agjenti mund të mbledhë në të ardhmen duke filluar nga gjendja aktuale s_t. Sinjali i shpërblimit r_{t+1} përfaqëson vetëm një vlerë lokale të shpërblimit në kuptimin që merr parasysh vetëm gjendjen, ndërsa funksioni i vlerës ofron një pamje më të gjerë të shpërblimeve të ardhshme: është një *lloj parashikimi i shpërblimeve*.

$$V_\pi(s_t) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t] = E_\pi[r_{t+1} + \gamma G_{t+1} | s_t] \tag{2.9}$$

Me fjalë të tjera, funksioni i vlerës vlerëson se sa e mirë është një gjendje, e përfaqësuar nga vlera e pritur E_π nën politikën π. Vini re se nuk merr parasysh veprimet, por vetëm sa mirë është të jesh në atë gjendje të veçantë $V_\pi(s_t)$ në lidhje me detyrën specifike. Duke marrë parasysh se sa mirë është të jesh në një gjendje duke marrë parasysh edhe veprimin e ndërmarrë nga agjenti, është e nevojshme t'i referohemi funksionit të vlerës së veprimit, i njohur gjithashtu si *funksioni-Q* ose *funksioni i vlerës-Q* i përcaktuar në ekuacionin 2.10 dhe i shënuar si $Q_\pi(s_t, a_t)$, i cili jep kthimin e pritur për kryerjen e veprimit a_t në gjendjen a_t në një hap të përgjithshëm kohor t, nën politikën π:

$$Q_\pi(s_t, a_t) = E_\pi[G_t|s_t, a_t] = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots |s_t, a_t] = E_\pi[r_{t+1} + \gamma G_{t+1} |s_t, a_t] \quad (2.10)$$

Në të dyja këto funksione të vlerës, $G_t(s_t)$ është vlera e kthimit siç përcaktohet në ekuacionin 2.7. Funksioni optimal i vlerës i shënuar si $V^*(s_t)$ është ai që jep vlerën maksimale krahasuar me të gjitha funksionet e tjera të vlerës siç përcaktohet në ekuacionin 2.12:

$$V^*(s_t) = \max_\pi V_\pi(s_t) \quad \forall s \in S \quad (2.11)$$

$$V^*(s_t) = \max_a \sum_{s_{t+1} \in S} P(s_{t+1}, r_{t+1}|s_t, a_t) + [r_{t+1} + \gamma V^*(s_{t+1})] \quad \forall s \in S \quad (2.12)$$

Me fjalë të tjera, ai jep kthimin e pritur maksimal kur fillohet në gjendjen s_t, dhe duke vepruar sipas politikës optimale π^* më pas. Nga ana tjetër, funksioni optimal i vlerës së veprimit i shënuar si $Q^*(s_t, a_t)$ dhe i përcaktuar në ekuacionin 2.13 është ai që jep vlerën maksimale krahasuar me të gjitha funksionet e tjera të vlerës së veprimit:

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t) \quad \forall s \in S, a \in A \quad (2.13)$$

$$Q^*(s_t) = \max_a \sum_{s_{t+1} \in S} P(s_{t+1}, r|s_t, a_t) + \left[r_{t+1} + \gamma \max_{a_{t+1}} Q_\pi(s_{t+1}, a_{t+1})\right] \quad \forall s \in S, a \in A \quad (2.14)$$

dhe jep kthimin e pritur kur fillohet nga gjendja s_t, duke kryer veprimin a_t, dhe pastaj duke vepruar sipas politikës optimale π^* më pas. Ka një lidhje të rëndësishme midis funksionit optimal të vlerës së veprimit dhe veprimit optimal: politika optimale në s_t do të zgjedhë veprimin që rrit kthimin e pritur kur fillohet në gjendjen s_t. Prandaj, nëse $Q^*(s_t, a_t)$ është i njohur, veprimi optimal $a^*(s_t)$ mund të merret drejtpërdrejt siç tregohet në ekuacionin 2.15.

$$a^*(s_t) = \arg\max_a Q^*(s_t, a_t) \quad (2.15)$$

2.3.5 Ekuacioni i Bellman-it

Qëllimi i agjentit është të përafrojë një politikë optimale π^*, e cila rrit vlerën e çdo gjendje, duke dhënë V^* dhe Q^* si funksionet optimale të vlerës së gjendjes dhe të vlerës së veprimit. *Programimi Dinamik (DP)* është një metodë optimizimi që thjeshton një problem të ndërlikuar si RL duke e ndarë atë në nën-probleme më të thjeshta që mund të zgjidhen më pas. Ideja është të shfrytëzojmë faktin që funksionet e vlerës përmbushin *Ekuacionin e Optimalitetit të Bellman-it*. Sipas Barto dhe Mahadevan (2003) [10], ekuacionet janë si më poshtë:

$$V^*(s_t) = \max_{\pi} V_{\pi}(s_t) \quad \forall s \in S \tag{2.16}$$

$$V^*(s_t) = \max_{a} \sum_{s_{t+1} \in S} P(s_{t+1}, r_t | s_t, a_t) + [r_{t+1} + \gamma V^*(s_{t+1})] \quad \forall s \in S \tag{2.17}$$

$$Q^*(s_t, a_t) = \max_{\pi} Q_{\pi}(s_t, a_t) \quad \forall s \in S, a \in A \tag{2.18}$$

$$Q^*(s_t) = \max_{a} \sum_{s_{t+1} \in S} P(s_{t+1}, r_{t+1} | s_t, a_t) + \left[r_{t+1} + \gamma \max_{a_{t+1}} Q_{\pi}(s_{t+1}, a_{t+1}) \right] \quad \forall s \in S, a \in A \tag{2.19}$$

Këtu, $P(s_{t+1})$ është *probabiliteti i gjendjes së tranzicionit*, që përfaqëson probabilitetin e vëzhgimit të funksionit të shpërblimit r_{t+1} dhe gjendjes s_{t+1} nga gjendja e mëparshme s_t pas kryerjes së veprimit a_t.

2.3.6 Pritshmëria

Meqenëse objektivi është të rritet shpërblimi afatgjatë i së ardhmes, *pritshmëria* e dhënë në ekuacionin 2.20 përdoret si shuma e peshuar e të gjitha rezultateve të mundshme të shumëzuara me probabilitetet e tyre. Me fjalë të tjera, pritshmëria, e njohur gjithashtu si *mesatarja*, llogaritet nga shuma e produktit të çdo vlere të gjendjes dhe probabilitetit të saj:

$$E[f(x)] = \sum_{x} P(x) f(x) \tag{2.20}$$

Në këtë ekuacion, $P(x)$ përfaqëson probabilitetin e ndodhjes së një variabli të përgjithshëm të rastit x, dhe $f(x)$ është një funksion i përgjithshëm që tregon vlerën e gjendjes x. Duke integruar probabilitetet e rezultateve të ndryshme me vlerat e tyre përkatëse, kjo formulë ofron një masë të vetme të pritshmërive të shpërblimeve të ardhshme.

2.3.7 Eksplorimi kundrejt shfrytëzimit

Një problem i rëndësishëm në RL është balancimi midis *eksplorimit* dhe *shfrytëzimit*. Për të arritur shpërblime të larta, agjenti duhet të zgjedhë veprime duke aplikuar një hyrje kontrolli që ka provuar më parë dhe di që është i mirë. Për të zbuluar këto veprime, agjenti duhet të zgjedhë veprime që nuk janë testuar duke përdorur pasojat e vëzhguara nga mjedisi. *Eksplorimi* i referohet agjentit RL që eksploron mjedisin për të mbledhur më shumë informacion, ndërsa *shfrytëzimi* do të thotë ndjekja e politikës më të mirë aktuale nga përvojat e mbledhura në të kaluarën dhe të disponueshme për agjentin për të fituar sa më shumë shpërblim të jetë e mundur. Me fjalë të tjera, shpërblimet afatshkurtra duhet të sakrifikohen në mënyrë që të gjendet një politikë e mirë në afat të gjatë duke eksploruar gjendjet që ende nuk janë parë nga agjenti. Sfida kryesore që lind në projektimin e sistemeve RL është në balancimin e tregtisë midis eksplorimit dhe shfrytëzimit. Në një mjedis stohastik, veprimet duhet të merren mjaftueshëm mirë për të marrë një vlerësim të shpërblimit të pritur. Një agjent që ndjek ekskluzivisht eksplorimin ose shfrytëzimin është i destinuar të jetë më pak se efektiv. Ai bëhet më keq se rastësia e pastër (d.m.th., një agjent i rastësishëm).

Ekzistojnë disa zgjidhje të tilla si eksplorimet naive dhe më të përdorurat quhen eksplorime $greedy$, $\varepsilon - greedy$ dhe $\varepsilon - greedy$ $të$ $dekompozuar$. Këto teknika janë mjaft të thjeshta për t'u kuptuar dhe zbatuar në praktikë, por ato vuajnë nga një mangësi e madhe, që është se kanë një pendesë nën-optimale. Në fakt, pendesa e të dyjave $greedy$ dhe $\varepsilon - greedy$ rritet në mënyrë lineare me kalimin e kohës. Kjo është e thjeshtë për t'u kuptuar intuitivisht pasi $greedy$ do të fokusohej në një aktivitet që, edhe pse mund të ketë prodhuar rezultate pozitive në të kaluarën, nuk ishte në fakt veprimi më i mirë. Pra, $greedy$ do të vazhdojë të përfitojë nga kjo situatë ndërsa injoron mundësitë e mundshme. Me fjalë të tjera, ai shfrytëzon tepër shumë.

Nga ana tjetër, $\varepsilon - greedy$ eksploron tepër shumë sepse edhe kur një veprim duket të jetë optimal, metoda vazhdon të alokojë një përqindje të caktuar të kohës për eksplorim, duke humbur kështu mundësitë dhe duke rritur pendesën totale. Formulimi i saj është mjaft i thjeshtë:

$$le\ të\ jetë\ \epsilon\ \in\ (0,1): \begin{cases} a^*(s_t) = arg\max_a Q^*(s_t, a_t)\ me\ probabilitet\ (1-\epsilon) \\ veprim\ i\ rastësishëm \qquad me\ probabilitet\ \epsilon \end{cases} \tag{2.21}$$

Veprimi $a^*(s_t)$ i shprehur në ek. 2.21 merret nga agjenti sipas politikës më të mirë aktuale π. Vini re se në fillim të trajnimit, probabiliteti për të bërë eksplorim do të jetë i madh me një vlerë të lartë ϵ, kështu që shumicën e kohës agjenti do të eksplorojë. Ndërsa trajnimi vazhdon dhe për pasojë funksionet Q përmirësohen në vlerësimet e tyre, është e nevojshme të zvogëlohet gradualisht vlera ϵ pasi agjenti do të ketë nevojë për gjithnjë e më pak eksplorim dhe më shumë shfrytëzim:

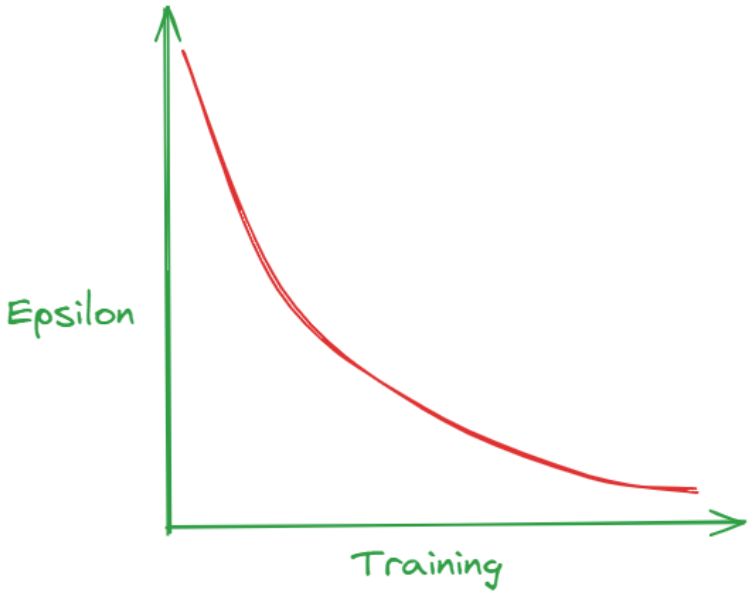


Figura 2.6 Përgjigja e epsilonit gjatë trajnimit

Nga ana tjetër, ndërsa koha kalon, qasjet $\varepsilon - greedy$ të dekompozuara përpiqen të zvogëlojnë përqindjen e caktuar për eksplorim. Grafiku në figurën 2.7 tregon se si kjo mund të rezultojë në pendesën më të mirë.

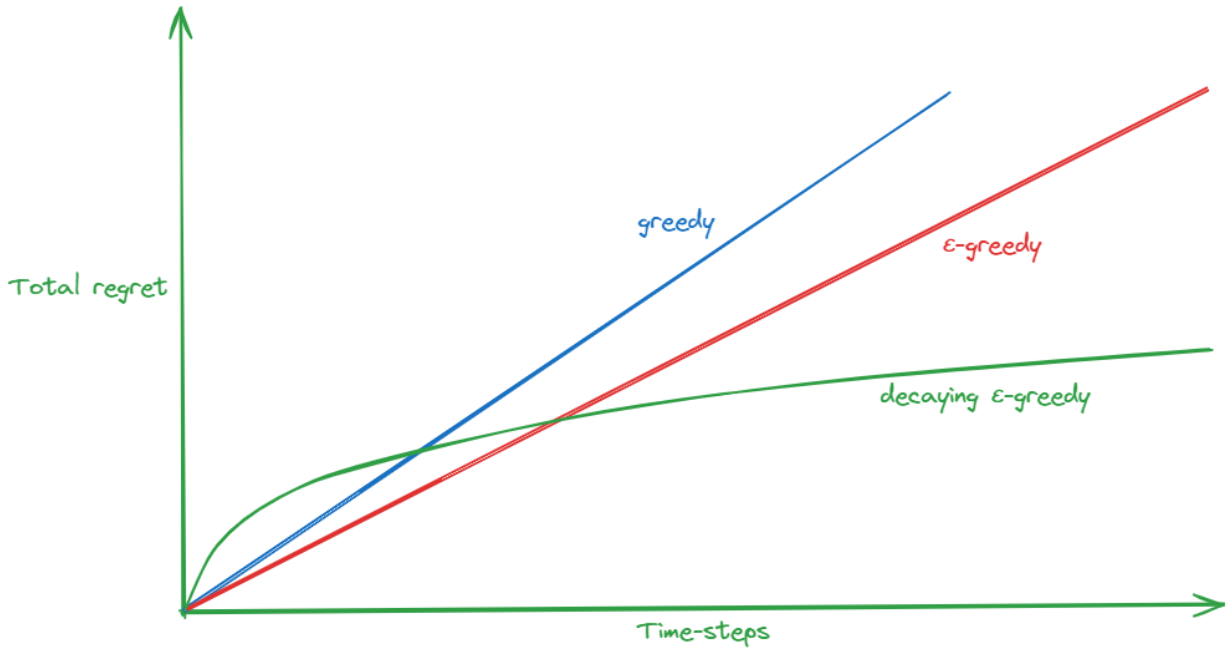


Figura 2.7 Krahasimi i pendesës në lidhje me metodat e eksplorimit

Vështirësia qëndron në aftësinë për të kryer procesin e shuarjes në mënyrë të duhur.

2.3.8 Qasjet ndaj RL

Çdo agjent në një kornizë të mësimit përforcues (RL) funksionon bazuar në një algoritëm specifik të dizajnuar për të optimizuar shpërblimet që merr nga mjedisi i tij. Çdo lloj algoritmi RL ka karakteristika dhe funksionalitete unike. Kuptimi i këtyre dallimeve është thelbësor për zgjedhjen e algoritmit më të përshtatshëm të përshtatur për kërkesat specifike të një problemi të dhënë. Peizazhi i algoritmeve RL është i gjerë dhe vazhdimisht në zhvillim, duke e bërë sfiduese paraqitjen e një përmbledhje gjithëpërfshirëse. Megjithatë, dallimet e theksuara në këtë seksion synojnë të kapin aspektet më kritike të rëndësishme për këtë tezë, pa pretenduar të jenë shteruese.

Algoritmet e mësimit përforcues (RL) mund të klasifikohen gjerësisht në kategori të ndryshme bazuar në strategjitë e tyre të mësimit dhe teknikat e zbatimit. Klasifikime të zakonshme përfshijnë algoritmet pa model kundrejt algoritmeve të bazuara në model, metodat e bazuara në vlerë kundrejt metodave të bazuara në politikë, dhe algoritmet me politikë në veprim kundrejt atyre me politikë jashtë veprimit [11]. Algoritmet pa model, si Q-learning dhe SARSA, mësojnë drejtpërdrejt nga ndërveprimet me mjedisin pa pasur nevojë për një model të mjedisit, ndërsa algoritmet e bazuara në model, si Dyna-Q, përdorin një model për të simuluar dhe planifikuar veprimet. Metodat e bazuara në vlerë fokusohen në vlerësimin e funksioneve të vlerës, me shembuj të spikatur përfshirë Q-learning dhe Deep Q-Networks (DQN). Metodat e bazuara në politikë, si REINFORCE dhe Optimizimi Proksimal i Politikës (PPO), optimizojnë drejtpërdrejt politikën që dikton veprimet e agjentit. Algoritmet me politikë në veprim mësojnë vlerën e politikës që po realizohet nga agjenti, ndërsa algoritmet me politikë jashtë veprimit, si Q-learning, mësojnë vlerën e një politike optimale pavarësisht veprimeve të agjentit [12].

Këto dallime janë thelbësore për të kuptuar avantazhet dhe kufizimet e qasjeve të ndryshme të RL dhe përshtatshmërinë e tyre për aplikacione të ndryshme. Në kontekstin e kësaj teze, fokusi do të

jetë në shpjegimin e këtyre dallimeve për të ofruar një kuptim të plotë të metodave RL të zbatueshme për problemin në fjalë.

2.3.8.1 Komponentët e të mësuarit

Një dallim kryesor midis algoritmeve të mësimit përforcues (RL) vjen nga mënyrat e ndryshme se si këto algoritme shfrytëzojnë komponentët e agjentit. Këto strategji mund të shpjegohen duke përdorur *metrika të politikës*, *modelit* dhe *funksionit të vlerës* të përcaktuara në seksionet 2.3.1 dhe 2.3.4.

Një nga aspektet kritike të një algoritmi RL është nëse agjenti ka qasje në modelin e mjedisit. Ky model i mundëson agjentit të parashikojë kalimet e gjendjes dhe shpërblimet. Një metodë *pa model* nuk e shfrytëzon modelin e mjedisit për të zgjidhur problemin RL. Në vend të kësaj, veprimet e rregullimit të aplikuara nga agjenti bazohen në vëzhgimet e drejtpërdrejta të gjendjes aktuale. Agjenti vëzhgon mjedisin dhe zgjedh hyrjen më të mirë të rregullimit në përputhje me rrethanat. Kjo qasje bie në kundërshtim me metodat e *bazuara në model*, ku agjenti ka njohuri të plotë për mjedisin, përfshirë edhe bimen e sistemit. Metodat e bazuara në model, nga ana tjetër, supozojnë se agjenti ka njohuri të plotë për mjedisin, përfshirë edhe kombinimin e procesit dhe aktuatorit. Këto metoda mund të jenë të dobishme në mjedise përcaktuese, ku agjenti mund të shfrytëzojë të gjitha informacionet e disponueshme për të mësuar një politikë optimale [13]. Megjithatë, marrja e njohurive të sakta për mjedisin shpesh është sfiduese për shkak të faktorëve si parametrat e panjohur të sistemit ose kompleksiteti i vetë sistemit. Si rezultat, metodat pa model janë bërë më të njohura dhe të zhvilluara në mënyrë të gjerë dhe të testuara.

Një dallim tjetër thelbësor midis algoritmeve RL është përdorimi i politikës ose funksionit të vlerës si elementi qendror i metodës. Metodat e bazuara në politikë fokusohen në përafrimin e politikës së agjentit, zakonisht e përfaqësuar si një shpërndarje probabilistike mbi veprimet. Këto metoda synojnë të optimizojnë drejtpërdrejt sjelljen e agjentit, por mund të kërkojnë ndërveprime të gjera me mjedisin, duke i bërë ato më pak të efektshme në mostra. Në të kundërt, metodat e bazuara në vlerë përfshijnë agjentin në gjetjen e sjelljes optimale në mënyrë indirekte duke shfrytëzuar funksionet e vlerës. Në vend që të fokusohen në shpërndarjen probabilistike të veprimeve, metodat e bazuara në vlerë përcaktojnë vlerën e të gjitha veprimeve të mundshme dhe zgjedhin atë më të mirën. Ndryshe nga metodat e bazuara në politikë, metodat e bazuara në vlerë mund të përfitojnë nga burime shtesë si të dhënat e politikës së vjetër ose replay buffer.

Në përmbledhje, kuptimi i këtyre dallimeve është thelbësor për zgjedhjen e qasjes së duhur RL për një problem të caktuar. Kjo tezë kryesisht fokusohet në qasjet pa model për shkak të thjeshtësisë dhe qëndrueshmërisë së tyre në aplikime të ndryshme. Megjithatë, njohja e forcave dhe kufizimeve të të dyja metodave, me dhe pa model, është thelbësore për përparimin e kërkimeve RL dhe aplikimeve të tyre praktike.

2.3.8.2 Qasjet e të mësuarit

Shumë sfida rregullimi në robotikë dhe drejtim të automatizuar kërkojnë struktura të avancuara, jolineare të rregullimit. Zgjidhjet për këto sfida shpesh përfshijnë metoda si programimi i fitimeve, rregullimi i qëndrueshëm dhe rregullimi modelor parashikues jolinear (MPC). Megjithatë, zbatimi i këtyre metodave zakonisht kërkon një nivel të lartë ekspertize nga inxhinieri i rregullimit dhe një sistem të aftë për matje *online*. Rregullimi i fitimeve dhe parametrave, për shembull, mund të jetë mjaft kompleks.

Në të kundërten, mësimi përforcues (RL) ofron një paradigmë të ndryshme ku sjellja e agjentit mund të mësohet ose *online* ose *offline*. Në një mjedis online, procesi i mësimit ndodh paralelisht me ndërveprimin e vazhdueshëm të agjentit me mjedisin, duke mbledhur vazhdimisht informacione të reja. Në anën tjetër, mësimi *offline* mbështetet në një dataset fiks, duke përparuar drejt mësimit me të dhëna të kufizuara.

Një dallim tjetër i madh në algoritmet RL është qasja e tyre ndaj mësimit të politikës. Algoritmet *on-policy* varen shumë nga të dhënat e trajnimit të marra sipas politikës aktuale, duke përdorur vetëm të dhënat e mbledhura me politikën e fundit. Në anën tjetër, metodat *off-policy* mund të shfrytëzojnë të dhëna nga burime të ndryshme përveç përvojës së drejtpërdrejtë, si për shembull buferat e përvojës së gjerë nga episodet e kaluara. Këto bufera zakonisht zgjidhen në mënyrë të rastësishme për t'i bërë të dhënat më afër të qenit të pavarura dhe të shpërndara në mënyrë identike [14].

Nën-seksioni vijues do të shpjegojë qasjen e përdorur shpesh bazuar në model, *Programimin Dinamik* (DP). DP është thelbësor për t'u kuptuar pasi përmban të gjitha idetë dhe kornizat kryesore që mbështesin shumicën e algoritmeve RL.

2.3.9 Qasja e bazuar në model: Programimi Dinamik

Programimi Dinamik (DP) është një qasje kyçe e përdorur për të trajtuar problemet e mësimit përforcues (RL). Në thelb, DP është një metodë e gjithanshme që dekompozon problemet komplekse në nënpjesë më të menaxhueshme. Duke zgjidhur këto nënpjesë, ne mund të agregojmë zgjidhjet e tyre për të adresuar në mënyrë gjithëpërfshirëse problemin fillestar kompleks. Kjo metodë është veçanërisht efektive për problemet e bazuara në Procesin e Vendimit Markovian (MDP), pasi supozon njohuri të plota të sistemit, duke mundësuar kështu përcaktimin e rezultatit më të mirë të arritshëm.

DP ofron një kornizë të fuqishme për zgjidhjen e problemeve të bazuara në MDP duke shfrytëzuar njohuri të plota të sistemit. Kjo kornizë është kryesisht e aplikueshme për problemet e bazuara në model, ku modeli i mjedisit është i përcaktuar mirë. Natyrshmëria e DP përfshin përdorimin e një ose më shumë vlerave të vlerësuara gjatë fazës së përditësimit, gjë që mund të bëjë që rezultatet të varen shumë nga vlerësimet fillestare. Kjo karakteristikë nënvizon ndjeshmërinë e metodave DP ndaj kushteve fillestare.

Metodat kryesore të DP përfshijnë *përsëritjen e politikës* dhe *përsëritjen e vlerës*. Përsëritja e politikës përfshin vlerësimin dhe përmirësimin e përseritjeve të politikave derisa të gjendet një politikë optimale. Përsëritja e vlerës, nga ana tjetër, fokusohet në përditësimin e funksionit të vlerës derisa të konvergjojë në funksionin optimal të vlerës. Të dyja metodat ilustrojnë fuqinë e DP në zgjidhjen sistematike të problemeve RL duke i ndarë dhe rafinuar në mënyrë të përseritur zgjidhjet [15].

2.3.9.1 Përsëritja (Iteracioni) e politikës (policy)

Iteracioni i Politikës është një qasje e dizajnuar për të gjetur politikën optimale π^* duke rafinuar në mënyrë iterative një politikë fillestare të rastësishme. Kjo metodë përbëhet nga dy hapa kryesore: *Vlerësimi i Politikës* 2.23 dhe *Përmirësimi i Politikës* 2.24.

Hapi i Vlerësimit të Politikës synon të vlerësojë politikën aktuale duke aplikuar në mënyrë iterative Ekuacionin e Bellmanit, siç është shpjeguar në Seksionin 2.3.5. Duke filluar me një funksion vlerash të rastësishme $V_\pi(s_t)$ për të gjitha gjendjet, ekuacioni i Bellmanit përdoret për të përditësuar

këto vlera, duke siguruar që ato të reflektojnë shpërblimet e pritura të ardhshme. Ky proces iterativ vazhdon deri sa funksioni i vlerës të konvergjojë.

Më pas, hapi i Përmirësimit të Politikës synon të përmirësojë politikën aktuale duke zgjedhur veprime që japin shpërblime më të larta. Kjo bëhet duke vlerësuar funksionin e vlerës së veprimit $Q_\pi(s_t, a_t)$, i cili jep vlerën e marrjes së një veprimi specifik a_t në gjendjen aktuale s_t dhe ndjekjes së politikës ekzistuese π më pas. Nëse një politikë e re π' jep një vlerë më të lartë se politika e mëparshme π, ajo konsiderohet më e mirë. Ky përmirësim iterativ vazhdon deri sa nuk ndodhin më modifikime në politikë, duke rezultuar në politikën optimale π^*.

Hapat e Vlerësimit dhe Përmirësimit të Politikës mund të përmblidhen si më poshtë:

1. **Vlerësimi i Politikës**: vlerësimin e funksionit të vlerës V_π për një politikë të dhënë π duke përdorur kopje të sinkronizuara duke iteruar Ekuacionin e Bellmanit. Rregulli i përditësimit është:

$$V(s_t) \leftarrow E_\pi[r_{k+1} + \gamma V_\pi(s_{t+1})] \tag{2.23}$$

2. **Përmirësimi i Politikës**: përmirësimin e politikës duke vepruar në mënyrë të etur me respekt për funksionin e vlerës të vlerësuar $V_\pi(s_t)$. Veprimi më i mirë për secilën gjendje zgjidhet për të përmirësuar π:

$$\pi(s_t) \leftarrow arg\max_a E_\pi\,[r_{t+1} + \gamma V_\pi(s_{t+1})] \tag{2.24}$$

Ky proces iterativ, i njohur si Iteracioni i Politikës, ndalet sapo politika të stabilizohet dhe të mos ndryshojë më, duke treguar se politika optimale π^* është gjetur. Pseudo-algoritmi për Iteracionin e Politikës është paraqitur më poshtë:

Algorithm 1: Policy Iteration Algorithm

Data: θ: a small number
Result: V: a value function s.t. $V \approx v_*, \pi$: a deterministic policy s.t. $\pi \approx \pi_*$
Function *PolicyIteration* **is**
  /* Initialization */
  Initialize $V(s)$ arbitrarily;
  Randomly initialize policy $\pi(s)$;
  /* Policy Evaluation */
  $\Delta \leftarrow 0$;
  while $\Delta < \theta$ **do**
    for $each\ s \in S$ **do**
      $v \leftarrow V(s)$;
      $V(s) \leftarrow \sum_{s',r'} p(s', r|s, \pi(s))[r + \gamma V(s')]$;
      $\Delta \leftarrow max(\Delta, |v - V(s)|)$;
    end
  end
  /* Policy Improvement */
  policy-stable $\leftarrow true$;
  for $each\ s \in S$ **do**
    old-action $\leftarrow \pi(s)$;
    $\pi(s) \leftarrow arg\max_a \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')]$;
    if *old-action* $!= \pi(s)$ **then**
      policy-stable $\leftarrow false$;
    end
  end
  if $policy - stable$ **then**
    return $V \approx v_*$ and $\pi \approx \pi_*$;
  else
    go to Policy Evaluation;
  end
end

ku $V(s) = V(s_t)$ 2.9, V^{π} është funksioni$-V$ optimal i përcaktuar në 2.12, $V_{\pi} = V\pi$ është funksioni$-V$ sipas politikës aktuale π, π^* është politika optimale që duhet gjetur, S është hapësira e gjendjes, $s' = s_{t+1}$.

Vini re se θ është parametri që përcakton saktësinë: sa më shumë që vlera të jetë afër 0, aq më i saktë do të ishte vlerësimi.

2.3.9.2 Përsëritja (Iteracioni) e vlerës

Qasja e dytë e përdorur nga Programimi Dinamik (DP) për të zgjidhur Proceset e Vendimeve Markoviane (MDP) është *Iteracioni i Vlerave (VI)*. Iteracioni i Vlerave është një teknikë iterative që alternon midis vlerësimit të politikës dhe përmirësimit të politikës derisa të konvergjojë në politikën optimale π^*. Ndryshe nga Iteracioni i Politikës, VI përdor një version të modifikuar të

vlerësimit të politikës për të përcaktuar funksionin e vlerës $V(s_t)$ dhe më pas mëson politikën optimale nga ai.

Iteracioni i Vlerave thjeshton procesin duke kombinuar hapat e vlerësimit të politikës dhe përmirësimit të politikës në një operacion të vetëm përditësimi. Kjo metodë llogarit vlerën e secilës gjendje nën politikën aktuale dhe më pas përditëson politikën bazuar në këto vlera derisa funksioni i vlerës të stabilizohet dhe të mos mund të bëhen përmirësime të mëtejshme. Qëllimi është të rafinohet iterativisht funksioni i vlerës dhe politika derisa të gjendet zgjidhja optimale.

Pseudo-algoritmi përkatës për Iteracionin e Vlerave është si më poshtë:

Algorithm 2: Value Iteration Algorithm

Data: θ: a small number
Result: π: a deterministic policy s.t. $\pi \approx \pi_*$
Function *ValueIteration* **is**
  /* Initialization */
  Initialize $V(s)$ arbitrarily, except $V\ (terminal)$;
  $V\ (terminal) \leftarrow 0$;
  /* Loop until convergence */
  $\Delta \leftarrow\ 0$;
  while $\Delta <\ \theta$ **do**
    for $each\ s\ \in\ S$ **do**
      $v\ \leftarrow\ V(s)$;
      $V(s) \leftarrow \max_a \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')]$;
      $\Delta \leftarrow\ max(\Delta, |v\ -\ V(s)|)$;
    end
  end
  /* Return optimal policy */
    return π s.t. $\pi(s) \leftarrow arg\max_a \sum_{s',r'} p(s', r|s, a)[r\ +\ \gamma V(s')]$;
end

2.3.9.3 Iteracioni i Politikës kundrejt Iteracionit të Vlerave

Iteracioni i Politikës dhe Iteracioni i Vlerave janë të dy algoritme të programimit dinamik që përdoren për të gjetur një politikë optimale π^* në një mjedis të Mësimit Përforcues (RL). Të dyja përdorin një hap të vetëm përpara dhe përditësime të modifikuara të Bellman-it, siç është diskutuar më parë.

Dallimet kryesore ndërmjet këtyre dy qasjeve janë të theksuara si më poshtë:

- Në Iteracionin e Politikës, pika fillestare është një politikë fikse. Përkundrazi, Iteracioni i Vlerave fillon duke zgjedhur funksionin e vlerës. Në të dy algoritmet, hapi i përmirësimit zbatohet në mënyrë iterative deri sa të arrihet konvergjenca.
- Politika përditësohet përmes algoritmit të Iteracionit të Politikës, ndërsa algoritmi i Iteracionit të Vlerave përditëson funksionin e vlerës drejtpërdrejt. Megjithatë, çdo iteracion i të dy algoritmeve përditëson në mënyrë indirekte funksionin e vlerës së gjendjes dhe politikën.

- Funksioni i Iteracionit të Politikës përfshin dy faza për çdo iteracion: vlerësimin e politikës në fazën e parë dhe përmirësimin e politikës në fazën e dytë. Duke marrë maksimumin mbi funksionin e shfrytëzimit për të gjitha veprimet e mundshme, funksioni i iteracionit të vlerave kombinon këto dy faza.
- Algoritmi i iteracionit të vlerave është më i thjeshtë pasi kombinon dy fazat e iteracionit të politikës në një operacion të vetëm përditësimi. Megjithatë, ai kalon përmes të gjitha veprimeve të mundshme për të gjetur vlerën maksimale të veprimit, duke e bërë atë më të rëndë në aspektin e llogaritjeve.
- Të dy algoritmet janë të garantuar që do të konvergojnë në një politikë optimale eventualisht. Megjithatë, algoritmi i Iteracionit të Politikës konvergjon brenda më pak iteracioneve, duke e bërë atë më të shpejtë se algoritmi i Iteracionit të Vlerave [16].

Si përfundim, dallimet kryesore midis këtyre algoritmeve mund të përmblidhen si më poshtë:

Iteracioni i Politikës	**Iteracioni i Vlerave**
Fillon me një politikë të rastësishme	Fillon me funksionin e vlerave të rastësishme
Algoritmi është më kompleks	Algoritmi është më i thjeshtë
Garantohet të konvergjojë	Garantohet të konvergjojë
Më e lirë për t'u llogaritur	Më e shtrenjtë për t'u llogaritur
Kërkon pak përsëritje për të konvergjuar	Kërkon shumë përsëritje për të konvergjuar
Më i shpejt	Më i ngadaltë

Tabela 2.1 Përmbledhje e Iteracionit të Politikës kundrejt Iteracionit të Vlerave

2.3.10 Qasjet pa model

Hipoteza kryesore e metodave të Programimit Dinamik (DP) është të kesh një njohuri të plotë të mjedisit. Megjithatë, kjo nuk është gjithmonë e mundur nga një perspektivë praktike, veçanërisht për sistemet shumë komplekse ose mjediset me pasiguri të konsiderueshme. Në skenarë të tillë, agjenti ose rregulluesi duhet të nxjerrë informacion përmes përvojës duke përdorur metoda pa model. Këto metoda operojnë nën supozimin se nuk ka njohuri paraprake rreth tranzicioneve të gjendjes dhe shpërblimeve.

Qasjet pa model janë thelbësore për sistemet e rregullimit ku modelimi i mjedisit është sfidues ose i pamundur. Një nga teknikat kryesore pa model është mësimi me *Diferencë Temporale (TD)*. Mësimi TD kombinon ide nga metodat *Monte Carlo* dhe *programimi dinamik*. Ajo përditëson vlerën e një gjendjeje bazuar në ndryshimin (ose diferencën temporale) midis vlerave të gjendjeve të njëpasnjëshme. Kjo metodë lejon agjentin të mësojë direkt nga përvoja e papërpunuar pa pasur nevojë për një model të dinamikës së mjedisit.

Metodat e mësimit me Diferencë Temporale, të tilla si TD(0), SARSA (Gjendje-Veprim-Shpërblim-Gjendje-Veprim), dhe Q-Learning, janë përdorur gjerësisht në aplikime të ndryshme. TD(0) është forma më e thjeshtë, duke përditësuar vlerën e gjendjes aktuale bazuar në shpërblimin e vëzhguar dhe vlerën e vlerësuar të gjendjes së ardhshme. SARSA, një metodë në politikë, përditëson funksionin e vlerës së veprimit bazuar në veprimet e politikës aktuale. Në kontrast, Q-mësimi, një metodë jashtë politikës, synon të mësojë funksionin optimal të vlerës së veprimit në mënyrë të pavarur nga veprimet e agjentit [17].

Këto qasje pa model ofrojnë zgjidhje të fuqishme për sistemet e rregullimit, veçanërisht në mjedise ku ndërtimi i një modeli të saktë është i papraktikueshëm. Duke u mbështetur në përvojën dhe përditësimet iterative, ato mundësojnë agjentin të adaptohet dhe të optimizojë sjelljen e tij në mënyrë efektive.

2.3.11 Mësimi me Diferencë Temporale

Mësimi me Diferencë Temporale (TD) është një koncept thelbësor në Mësimin Përforcues (RL). Metodat TD kategorizohen si pa model sepse i lejojnë agjentit të mësojë drejtpërdrejt nga përvojat pa u mbështetur në një model të paracaktuar të mjedisit. Në mësimin TD, agjenti përditëson vlerësimet e tij të vlerës bazuar në shpërblimet e vëzhguara dhe vlerën e vlerësuar të gjendjes së ardhshme, një proces i njohur si *bootstrapping*. Kjo do të thotë se vlerësimet përditësohen duke përdorur vlerësimet e tjera të mësuara të funksionit pa pritur rezultatin përfundimtar. Mësimi TD adreson *problemin e parashikimit*, ku qëllimi është të vlerësohet funksioni i vlerës. Rregulli i përditësimit për funksionin $-V$ në mësimin TD është i përcaktuar si:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2.25}$$

ku α përfaqëson normën e mësimit ($\alpha \in (0,1)$) dhe γ është faktori i zbritjes ($\gamma \in (0,1)$). Norma e mësimit α përcakton në çfarë mase informacioni i ri zëvendëson informacionin e vjetër. Për shembull, nëse $\alpha = 1$, agjenti përfshin plotësisht shpërblimin e ri, ndërsa nëse $\alpha = 0$, agjenti injoron informacionin e ri.

Termi $[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$ është i njohur si gabimi i TD, i cili është thelbësor për përditësimin e funksionit të vlerës:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.26}$$

Komponentët që përbëjnë gabimin TD referohen si *Objektivi TD*:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) \tag{2.27}$$

Teknikat e mësimit TD për parashikim aplikohen në metoda të ndryshme RL, duke përfshirë:

- **SARSA** (Gjendje-Veprim-Shpërblim-Gjendje tjetër-Veprim tjetër): Një metodë e politikës në veprim ku funksioni i vlerës së veprimit përditësohet bazuar në veprimet e marra nga politika aktuale.
- **Q-learning**: Një metodë e politikës jashtë veprimit që synon të mësojë funksionin optimal të vlerës së veprimit pavarësisht nga politika aktuale duke marrë parasysh veprimin më të mirë të mundshëm në gjendjen e ardhshme.

Këto qasje pa model ofrojnë zgjidhje të qëndrueshme për sistemet e rregullimit, veçanërisht në mjedise ku ndërtimi i një modeli të saktë është i pamundur. Duke u mbështetur në përvojë dhe përditësime iterative, ato i lejojnë agjentit të adaptojë dhe optimizojë sjelljen e tij në mënyrë efektive.

2.3.11.1Mësimi SARSA

SARSA (Gjendja-Veprimi-Shpërblimi-Gjendja-Veprimi) është një metodë e mësimit të Diferencës Temporale (TD) që është *on-policy* dhe *online*, e cila merr emrin e saj nga sekuenca e veprimeve dhe gjendjeve që vlerëson: $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. Kjo metodë operon duke përdorur parimin e parashikimit TD për të mësuar funksionin e vlerës së veprimit, $Q(s_t, a_t)$, në vend të

funksionit të vlerës së gjendjes, $V(s_t)$. Është një teknikë në politikë sepse vlerëson funksionin e vlerës së veprimit, $Q_\pi(s_t, a_t)$, për *politikën aktuale* $\pi(s_t)$ dhe njëkohësisht përditëson politikën $\pi(s_t)$ për të qenë lakmitare në lidhje me funksionin e vlerës së veprimit të vlerësuar, $Q_\pi(s_t, a_t)$. Përditësimi i funksionit $-Q$ arrihet përmes bootstrapping, siç është formuluar në Ekuacionin 2.25, dhe ndjek rregullin në Ekuacionin 2.28 pas çdo kalimi në një gjendje jo-terminal s_t.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{2.28}$$

Nëse haset një gjendje terminale s_{t+1}, $Q(s_{t+1}, a_{t+1})$ vendoset në zero. Është demonstruar se SARSA konvergjon në funksionin optimal të vlerës së veprimit, $Q^*(s_t, a_t)$, dhe politikën optimale, $\pi^*(s_t)$, me kusht që të gjitha çiftet gjendje-veprim të vizitohen në një numër të pafundëm herë dhe politika gradualisht bëhet plotësisht lakmitare. Në thelb, agjenti ndërvepron me mjedisin, duke përditësuar politikën e tij bazuar në veprimet e ndërmarra dhe shpërblimet përkatëse të marra.

Pseudo-algoritmi i SARSA-s ilustrohet më poshtë:

Algorithm 3: SARSA (on-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $epsilon > 0$;
Initialize $Q(s_t, a_t)$, for all $s_t \in S$, $a_t \in A$, arbitrarily except that $Q(terminal, \cdot) = 0$;
For each episode:
  Initialize s_t;
    For each step of episode:
      Choose a_t from s_t using policy derived from $Q(s_t, a_t$ (e.g., $-$greedy));
      Take action at, observe r_{t+1}, s_{t+1};
      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$;
      $s_t \leftarrow s_{t+1}$;
      $a_t \leftarrow a_{t+1}$;
    end;
end;

2.3.11.2Q-learning

Një nga përparimet e rëndësishme në fushën e Mësimit Përforcues (RL) është zhvillimi i algoritmit të rregullimit *off-policy*, i njohur si *Q-learning*. Ky algoritëm fokusohet në përditësimin e funksionit $-Q$ në një mënyrë të ngjashme me SARSA, me ndryshimin kryesor që qëndron në qasjen ndaj bootstrap-it. Në algoritmin *Q-learning*, veprimi i ardhshëm zgjidhet bazuar në një politikë lakmitare, që do të thotë se veprimi i zgjedhur është ai që rrit funksionin $-Q$ të gjendjes së ardhshme. Ky proces është përmbledhur në Ekuacionin 2.29:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1})\right] \tag{2.29}$$

Këtu, α përfaqëson normën e mësimit brenda intervalit $(0,1)$, dhe γ tregon faktorin e zbritjes, gjithashtu brenda intervalit $(0,1)$.

Pseudo-algoritmi për Q-learning është përshkruar si më poshtë:

Algorithm 4: Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $epsilon > 0$;
Initialize $Q(s_t, a_t)$, for all $s_t \in S$, $a_t \in A$, arbitrarily except that $Q(terminal, \cdot) = 0$;
For each episode:
  Initialize s_t;
    For each step of episode:
      Choose a_t from s_t using policy derived from $Q(s_t, a_t$ (e.g., $-$greedy));
      Take action at, observe r_{t+1}, s_{t+1};
      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_t, a) - Q(s_t, a_t) \right]$;
      $s_t \leftarrow s_{t+1}$;
    end;
end;

Në këtë kontekst, a i referohet veprimit që rrit funksionin $-Q$ në një hap të veçantë kohor t. Duke përditësuar vazhdimisht vlerat Q në këtë mënyrë, *Q-learning* synon të konvergjojë në një politikë optimale që rrit shpërblimin kumulativ për agjentin që ndërvepron brenda mjedisit të tij.

2.3.11.3SARSA kundrejt Q-learning

Dallimi më i rëndësishëm midis *SARSA* dhe *Q-learning* qëndron në mënyrën se si funksioni $-Q$ përditësohet pas çdo veprimi. *Q-learning*, një metodë Off-policy, përditëson vlerat e funksionit të tij Q duke përdorur vlerën Q të gjendjes së ardhshme s_{t+1} me veprimin lakmitar a_{t+1} që rrit funksionin $-Q$ në hapin e ardhshëm kohor. Në thelb, ai vlerëson shpërblimin e përgjithshëm të ardhshëm të zbritur për secilin funksion $-Q$ duke supozuar se ndiqet një politikë lakmitare (greedy), edhe nëse politika aktuale nuk është lakmitare.

Nga ana tjetër, *SARSA*, një metodë On-policy, përditëson vlerat e funksionit të tij Q duke përdorur vlerën Q të gjendjes së ardhshme s_{t+1} me veprimin a_{t+1} të nxjerrë nga politika aktuale. Kjo do të thotë që SARSA vlerëson shpërblimin e përgjithshëm të ardhshëm të zbritur për secilin funksion $-Q$ duke supozuar se politika aktuale vazhdon të ndiqet.

Tabela e mëposhtme thekson ndryshimet kryesore midis algoritmeve Q-learning dhe SARSA:

Q-learning	**SARSA**
1. Lëviz një hap duke zgjedhur a_t nga hapësira e veprimeve A	1. Lëviz një hap duke zgjedhur a_t nga hapësira e veprimeve $\pi(s_t)$
2. Vëzhgo r_{t+1}, s_{t+1}	2. Vëzhgo $r_{t+1}, s_{t+1}, a_{t+1}$
3. Përditëso funksionin $-Q$ $Q(s_t, a_t)$	3. Përditëso funksionin $-Q$ $Q(s_t, a_t)$
4. Përditëso politikën $\pi(s_t) \leftarrow arg\max_a(s_t, a_t)$	4. Përditëso politikën $\pi(s_t) \leftarrow Q(s_{t+1}, a_t)$

Tabela 2.2 Krahasimi ndërmjet Q-learning dhe SARSA

2.4 Përfundimi: Disavantazhet e Mësimit Përforcues (RL)

Algoritmet e diskutuar në këtë kapitull janë të dizajnuara për të punuar me sisteme që kanë gjendje dhe veprime të përcaktuara mirë dhe diskrete. Brenda këtij kuadri, tabelat bazuar në funksionin $-V$ ose funksionin $-Q$ janë ndërtuar, ku numri i rreshtave korrespondon me madhësinë e hapësirës së

gjendjeve dhe numri i kolonave korrespondon me hapësirën e veprimeve. Megjithatë, kjo qasje mund të çojë në sfida të rëndësishme. Për shembull, funksioni $-V$ kërkon një hyrje për çdo gjendje, ndërsa funksioni $-Q$ kërkon një hyrje për çdo çift gjendje-veprim.

Për më tepër, mësimi i vlerës së çdo gjendjeje individualisht mund të jetë i ngadaltë, një problem i njohur si mallkimi i përmasave. Një mënyrë për të adresuar këtë problem është përmes përdorimit të *Rrjetave Neurale Artificiale (ANN)* si *përafrues funksionesh*. Integrimi i teorisë së rrjeteve neurale, i njohur zakonisht si *Deep Learning*, me *Mësimin Përforcues (RL)*, rezulton në atë që njihet si *Mësim Përforcues i Thellë (DRL)*.

Në kapitullin e ardhshëm, ne do të prezantojmë konceptet themelore nga *Deep Learning*, duke përgatitur terrenin për një eksplorim më të detajuar të metodave të *Deep Reinforcement Learning*.

3. Deep Reinforcement Learning

Mësimi Përforcues (RL) ka përparuar ndjeshëm me integrimin e teknikave të Mësimit të Thellë (DL). Përpjekjet e fundit kërkimore në bashkimin e këtyre dy fushave kanë çuar në krijimin e sistemeve, algoritmeve dhe agjentëve të fuqishëm të Mësimit të Thellë Përforcues (DRL), të cilët kanë demonstruar arritje të jashtëzakonshme [18]. Koncepti themelor i DRL është të vlerësojë vlerat e funksionit $-V$ dhe funksionit $-Q$ në vend që t'i llogarisë ato drejtpërdrejt, duke adresuar problemin e ndëshkimit të dimensionit të përmendur në seksionin 2.4:

$$V(s_t) \approx V_\pi(s_t) \tag{3.1}$$

$$Q(s_t, a_t) \approx Q_\pi(s_t, a_t)$$

Përdorimi i *Rrjetave Neurale Artificiale (ANN)* si vlerësues jo vetëm që redukton kohën e trajnimit për sisteme me dimensione të larta, por gjithashtu kërkon më pak hapësirë në memorie [19]. Kjo qasje formon një urë ndërmjet Mësimit Përforcues tradicional (RL) dhe përparimeve të fundit në teorinë e Mësimit të Thellë (DL) [4, 20].

Ky kapitull do të fokusohet në algoritmet pa model, të cilat përshtaten me natyrën e punës së paraqitur. Seksionet e mëposhtme do të diskutojnë gjendjen më të avancuar dhe teoritë kryesore pas kornizës së Deep RL, duke përfshirë një përmbledhje të përdorimit të Rrjetave Neurale si përafrues funksionesh. Për më tepër, do të prezantohen dy algoritme të thella aktor-kritik të përdorura për zgjidhjen e problemeve të kontrollit në këtë tezë: metoda e ***Gradientit të Politikës së Deterministëve të Thellë (DDPG)***.

3.1. Bazat e Deep Learning

Deep Learning (DL) përfshin përdorimin e aproksimimit të funksioneve jolineare me shumë shtresa, më së shumti të zbatuara përmes Rrjetave Neurale. Në dallim nga të qenit një degë e veçantë e Machine Learning (ML), DL përbëhet nga një sërë teknikash dhe metodologjish të synuara për të përdorur rrjetat neurale për të adresuar detyra të ndryshme të ML. Duke shfrytëzuar këto teknika të avancuara, DL mundëson krijimin e modeleve që mund të përpunojnë dhe të mësojnë nga sasi të mëdha të dhënash, duke e bërë atë veçanërisht efektiv për detyra të tilla si përpunimi i imazheve dhe zërit.

Mësimi i Thellë Përforcues (DRL) i referohet specifikisht zbatimit të DL për të trajtuar problemet e Mësimit Përforcues (RL). Kjo qasje përmirëson aftësitë e RL tradicional duke përfshirë aftësitë e fuqishme të nxjerrjes së veçorive dhe njohjes së modeleve të DL, duke lejuar trajtimin e mjediseve më komplekse dhe me përmasa të larta [21].

Ky seksion përfshin *bazat e Mësimit të Thellë (DL)*, duke ofruar një përmbledhje gjithëpërfshirëse të koncepteve dhe parimeve bazë që mbështesin këtë fushë. Ajo gjithashtu do të shqyrtojë se si këto koncepte integrohen me Mësimin Përforcues (RL) për të krijuar sisteme të fuqishme dhe efikase të të mësuarit.

3.1.1. Rrjetet Neurale Artificiale (ANNs)

Rrjetet Neurale Artificiale (ANNs) fillimisht u zhvilluan për të simuluar mënyrën se si funksionojnë truri i kafshëve dhe njerëzve. Megjithëse janë një thjeshtëzim i proceseve biologjike, ato kanë provuar të jenë jashtëzakonisht të dobishme si përafrues funksionesh. Këto ANNs quhen *rrjete* sepse përbëhen nga shumë neurone (nyje) të lidhura së bashku, ashtu si truri i njeriut. Çdo

neuron merr shumë hyrje, të quajtura dendrite, nga neuronët e mëparshëm. Kur një kombinim linear i këtyre hyrjeve tejkalon një potencial të caktuar, ai dërgon sinjalin përmes një daljeje të vetme të quajtur akson. Matematikisht, faza e përpunimit për çdo neuron j përfshin marrjen e hyrjeve, *llogaritjen e shumës së peshuar* të tyre dhe pastaj shtimin e një *biasi* b:

$$in_j = \sum_{i=0}^{n} wx_{i,j} + b \tag{3.2}$$

ku indeksi i shtrihet në të gjitha nyjet në shtresën e mëparshme të lidhura me të. Më pas, aplikohet një funksion aktivizimi g, duke prodhuar daljen:

$$a_j = g(in_j) = g(\sum_{i=0}^{n} wx_{i,j} + b) \tag{3.3}$$

Këto neurone janë të organizuara në *shtresa* të ndryshme, të cilat mund të ndahen në tri kategori kryesore: *shtresat e hyrjes, shtresat e fshehura* dhe *shtresat e daljes*. Çdo rrjet neural ka një ose më shumë shtresa hyrëse ku të dhënat e hyrjes futen në rrjet si dhe një ose më shumë shtresa dalëse që gjenerojnë rezultatin përfundimtar bazuar në detyrën e rrjetit. Shtresat e hyrjes dhe daljes janë të lidhura përmes shtresave të fshehura që përmbajnë dinamikën e rrjetit në lidhje me detyrën e tij. Numri i shtresave të fshehura varet nga kompleksiteti i vlerësuar i funksionit të aktivizimit.

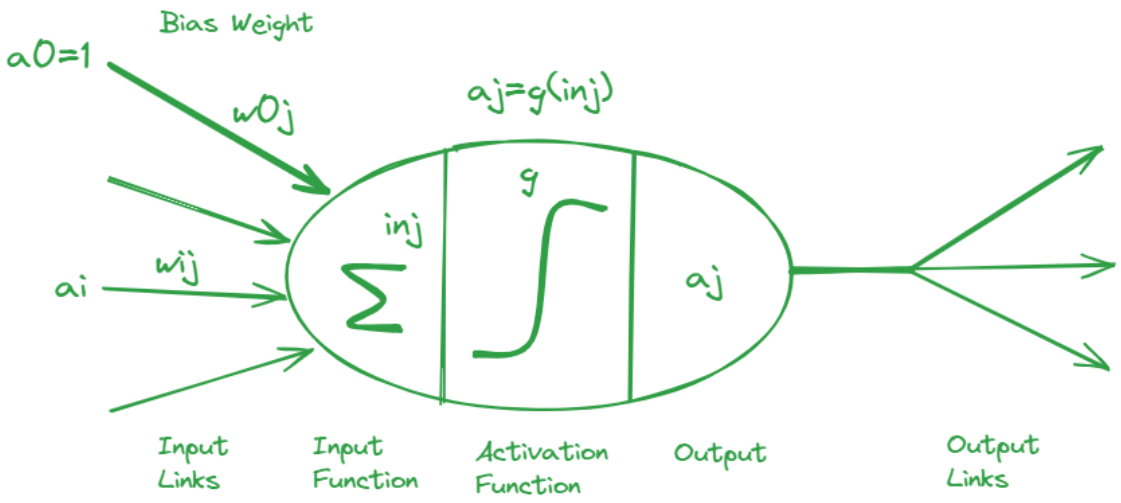


Figura 3.1 Shembull i një modeli të thjeshtë për një neuron të vetëm [5]

Çdo nyje në këto rrjete zbaton një klasifikues linear, por rrjeti në tërësi mund të përafrojë funksione jolineare gjithashtu. Një provë e hershme e teoremës së përafrimit universal tregoi se dy shtresa të fshehura mund të përshkruajnë çdo funksion, dhe më vonë u tregua se edhe një shtresë e vetme e fshehur është e mjaftueshme për të përfaqësuar çdo funksion të vazhdueshëm. Kjo arrihet duke ndryshuar peshat e ndryshme të nyjeve në rrjet, të referuara kolektivisht si *parametra të rrjetit* dhe zakonisht të shënuara si θ. Sfida kryesore atëherë bëhet gjetja e një kombinimi peshash që lejon rrjetin të prodhojë një funksion që përafrohet ngushtësisht me funksionin e synuar, bazuar në procesin e të mësuarit [19].

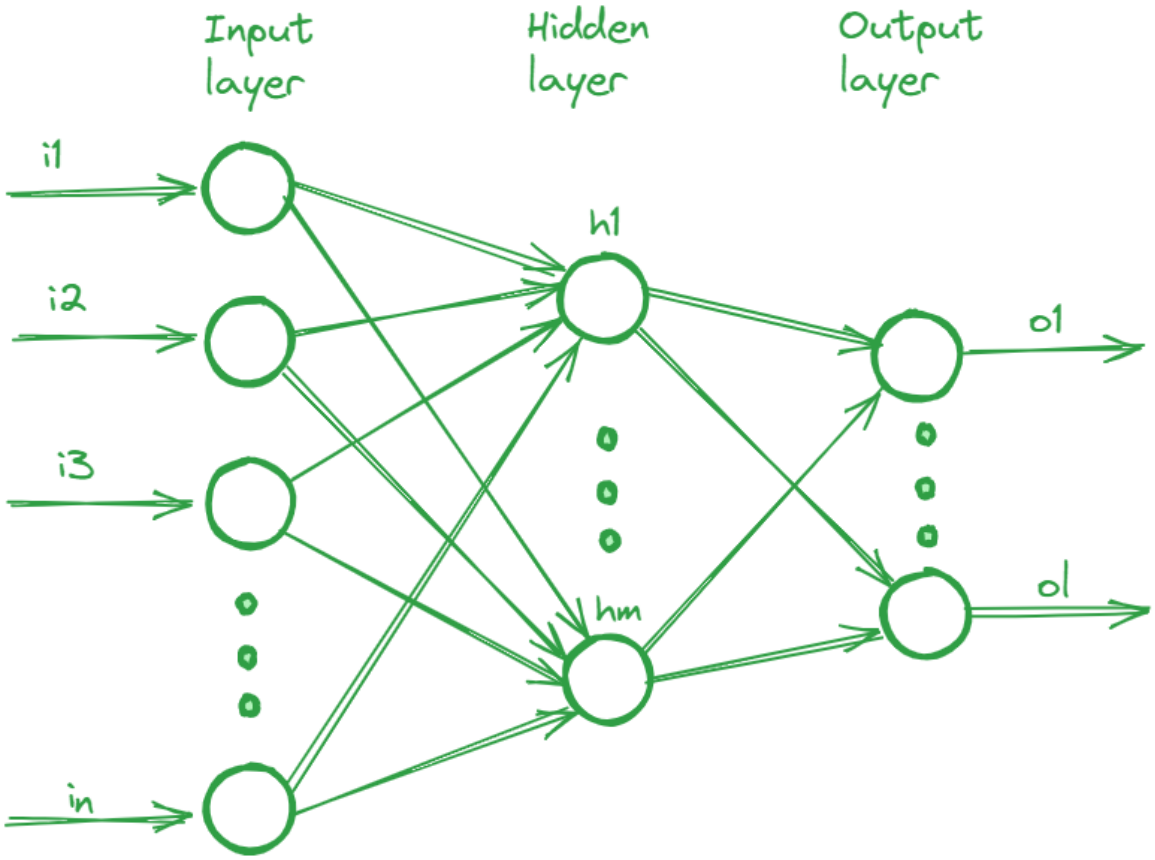


Figura 3.2 Shembull i thjeshtë që tregon një rrjet të thellë neural me katër shtresa [5]

3.1.2. Procesi i të mësuarit

Procesi i mësimit në rrjetet neuronale fokusohet në identifikimin e grupit optimal të parametrave, θ, për të arritur përafrimin më të mirë të mundshëm të funksionit për një objektiv të specifikuar. Ky proces përfshin disa hapa kryesorë:

1. **Rruga Përpara:** Vektori i veçorive hyrëse X kalon përmes rrjetit neural për të gjeneruar një rezultat të parashikuar $\hat{Y} = f(X, \theta)$. Ky hap përfshin llogaritjen e aktivizimeve të secilës shtresë neuronesh njëra pas tjetrës deri sa të arrihet shtresa përfundimtare e daljes.
2. **Llogaritja e Humbjes:** Rezultati i parashikuar $\hat{Y}$ krahasohet me $vlerën\ aktuale\ y$ duke llogaritur $funksionin\ e\ humbjes\ L(\theta)$. Funksioni i humbjes mat mospërputhjen midis daljes së rrjetit neural dhe vlerës së vërtetë. Një nga funksionet e humbjes më të përdorura është *Gabimi Mesatar i Katëruar (MSE)*, i cili përcaktohet si:

$$L(y, \hat{y}) = (y_\theta - y)^2 \tag{3.4}$$

   Ky hap mat se sa mirë përputhen parashikimet e rrjetit me të dhënat aktuale.

3. **Back-propagation:** Procesi i mësimit vazhdon duke llogaritur gradientin e funksionit të humbjes në lidhje me secilin parametër në rrjet. *Back-propagation* përdoret për të përhapur këto gradiente mbrapa nëpër rrjet, duke përdorur rregullin e zinxhirit për të llogaritur

derivatet e funksioneve të përbëra. Për shembull, nëse $y = g(x)$ dhe $z = f(g(x))$, derivati $\frac{dz}{dx}$ llogaritet si:

$$y = g(x), z = f\big(g(x)\big), \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x} \tag{3.5}$$

Ky hap është thelbësor për të kuptuar se si ndryshimet në parametrat e rrjetit ndikojnë në humbjen totale.

4. **Përditësimi i Parametrave:** Hapi përfundimtar në procesin e mësimit përfshin përditësimin e peshave të të gjitha neuroneve. Kjo zakonisht bëhet duke përdorur metodën e gradient descent, e cila synon të minimizojë funksionin e humbjes duke rregulluar parametrat e rrjetit në drejtim të kundërt të gradientit të humbjes. Rregulli i përditësimit për gradient descent është:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \nabla \theta_L \tag{3.6}$$

ku α është norma e mësimit, një hiperparametër që kontrollon madhësinë e hapave të përditësimeve.

5. **Rregullimi:** Për të siguruar që rrjeti neural të gjeneralizojë mirë mbi të dhëna të panjohura, përdoren teknikat e rregullimit për të parandaluar overfitting dhe underfitting. Rregullimi përfshin shtimin e një termi ndëshkimi në funksionin e humbjes, i cili ndihmon në mbajtjen e kompleksitetit të modelit nën kontroll. Funksioni i modifikuar i humbjes me rregullim është:

$$L'(\theta) = L(y, \hat{y}) + \lambda \Omega(\theta) \tag{3.7}$$

ku λ është faktori i rregullimit. Dy metoda të zakonshme të rregullimit janë *rregullimi* L^1 dhe L^2. *Rregullimi* L^1 ndëshkon shumën absolute të peshave:

$$L^1(\theta) = L(y, \hat{y}) + \lambda \frac{1}{2} ||\theta||^2 \tag{3.8}$$

Rregullimi L^2 ndëshkon shumën e katrorëve të peshave:

$$L^2(\theta) = L(y, \hat{y}) + \lambda \frac{1}{2} ||\theta|| \tag{3.9}$$

Këto teknika ndihmojnë në arritjen e një ekuilibri midis kompleksitetit të modelit dhe performancës.

3.1.2.1. Funksionet e aktivizimit

Funksionet e aktivizimit janë thelbësore në futjen e jo-linearitetit në rrjetet neuronale, duke përmirësuar aftësinë e tyre për të modeluar funksione komplekse dhe jo-lineare. Disa nga funksionet e aktivizimit më të përdorura përshkruhen më poshtë dhe paraqiten në Figurën 3.3:

$$Sigmoid \rightarrow g(x) = \frac{1}{1+e^{-x}}$$

$$Hyperbolic\ Tangent \rightarrow g(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (3.10)$$

$$Rectified\ Linear\ Unit\ (ReLu) \rightarrow g(x) = \max(0,x)$$

$$Leaky\ Rectified\ Linear\ Unit\ (Leaky\ ReLu) \rightarrow g(x) = \max(\alpha,x)$$

ku $\alpha \in (0,1)$ zakonisht vendoset në 0.01.

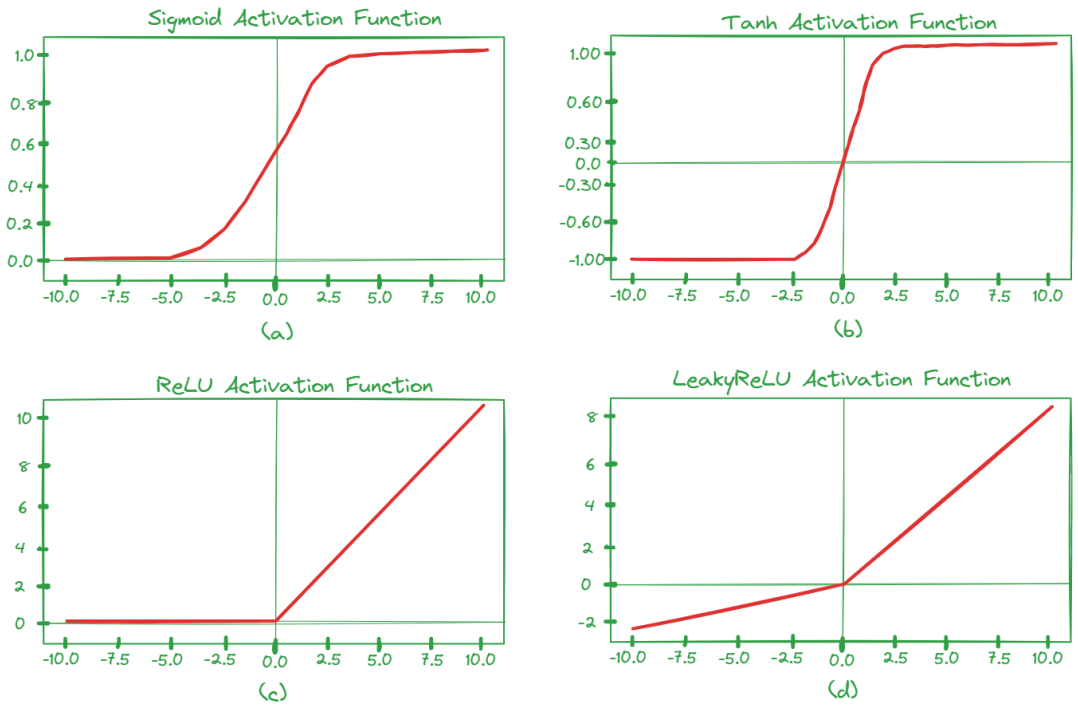


Figura 3.3 Grafiku i funksioneve të aktivizimit: (a) Sigmoid; (b) Tanh; (c) ReLU; (d) LeakyReLU

Në praktikë, rrjetet neuronale shpesh përdorin *ReLU* në shtresat e fshehura për shkak të efikasitetit të tij llogaritës dhe natyrës së tij jo-saturuese. Ndërkohë, *Tanh* preferohet në shtresat e daljes për të normalizuar rezultatin ndërmjet -1 dhe 1.

Këto funksione aktivizimi formojnë bazën e fazës së mësimit në rrjetet neuronale, veçanërisht në detyrat e mësimit përforcues (RL). Në RL, algoritmet që përdorin rrjetet neuronale për të vlerësuar drejtpërdrejt inputet e kontrollit quhen metoda të *Gradientit të Politikës (PG)*. Nga ana tjetër, algoritmet aktor-kritik përdorin rrjetet neuronale për të vlerësuar funksionin $-Q$, duke përcaktuar më pas hyrjet optimale të kontrollit.

Seksionet e ardhshme do të shqyrtojnë këto metoda në detaje, duke çuar në një kuptim të plotë të algoritmit të *Deep Deterministic Policy Gradient (DDPG)* të përdorur në këtë tezë.

3.2. Algoritmet i Policy Gradient

Ndryshe nga Q-Learning, algoritmet e gradientit të politikës (PG) fokusohen në optimizimin e drejtpërdrejtë të politikës për të rritur shpërblimin e pritur duke gjeneruar një trajektore τ. Qëllimi është të mësohet politika optimale π^* që ofron veprimin më të mirë nga çdo hap kohor t deri në kohën terminale T.

Matematikisht, metodat e gradientit të politikës synojnë të maksimizojnë funksionin e objektivit $J(\theta)$ duke rregulluar parametrat θ të rrjetit neural që parametron funksionin e politikës. Ky objektiv shprehet si:

$$J(\theta) = E\left[\sum_{t=0}^{T-1} r_{t+1}\right] \tag{3.11a}$$

Parametrat optimalë të politikës θ^* gjenden duke zgjidhur problemin e mëposhtëm të optimizimit:

$$\theta^* = arg\max_{\theta} J(\theta) \tag{3.11b}$$

Meqenëse kjo përfshin një *problem të maksimizimit*, politika optimohet duke përdorur ngritjen e gradientit. Kjo përfshin llogaritjen e gradientit të funksionit të objektivit në lidhje me parametrat e politikës θ dhe përditësimin e këtyre parametrave në drejtimin e gradientit. Rregulli i përditësimit është dhënë nga:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_\theta J(\theta) \tag{3.12}$$

ku α është norma e mësimit që kontrollon madhësinë e hapit në drejtimin e gradientit. Është e rëndësishme të theksohet se *ngritja e gradientit* është e kundërta e *zbritjes së gradientit* (siç është përcaktuar në Ekuacionin 3.6) dhe ajo përditëson parametrat θ_t në drejtimin pozitiv të gradientit të performancës së politikës, siç matet nga termi $\nabla_\theta J(\theta)$.

3.2.1. Derivimi i Policy Gradient (PG)

Duke zëvendësuar termin e pritshëm me përkufizimin e tij, rezultati i mëposhtëm merret:

$$J(\theta) = E\left[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta|\right] = \sum_{t=i}^{T-1} P(s_t, a_t|\tau) r_{t+1} \tag{3.13}$$

ku i është një pikë e rastësishme fillestare në një trajektore, dhe $P(s_t, a_t|\tau)$ është probabiliteti i ndodhjes së s_t, a_t duke pasur parasysh trajektoren t_τ.
Duke diferencuar të dy anët në lidhje me parametrin e politikës θ dhe duke përdorur pronën $\frac{\partial \log f(x)}{\partial x} = \frac{f'(x)}{f(x)}$, ne marrim:

$$\frac{\partial J}{\partial \theta} = \nabla_\theta J(\theta) \sim \sum_{t=i}^{T-1} \nabla_\theta \log P(s_t, a_t|\tau) r_{t+1} \tag{3.14}$$

Vini re se shpërblimi i pritshëm është zëvendësuar me mostra të rastësishme të episodeve. Duke përdorur përkufizimin e kthimit siç është përcaktuar në Ekuacionin (2.7), ne marrim:

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t) G_t \tag{3.15}$$

Termi i parë mat se sa e mundshme është trajektorja nën politikën aktuale π, dhe duke e shumëzuar me kthimin, ajo rritet në rast të shpërblimeve pozitive të larta. Përkundrazi, mundësia e një politike zvogëlohet nëse rezulton në një shpërblim të lartë negativ. Në thelb, gradienti i politikës rrit probabilitetet për veprime të mira dhe zvogëlon ato të veprimeve të këqija.

Hapat Kryesorë të Algoritmeve të Gradientit të Politikës janë:

1. Kryerja e një trajektore duke përdorur politikën aktuale π.
2. Ruajtja e logaritmit të probabiliteteve dhe vlerave të shpërblimeve në çdo hap t.
3. Llogaritja e shpërblimit të ardhshëm kumulativ të zbritur në çdo hap t.
4. Llogaritja e gradientit të politikës dhe përditësimi i parametrit të politikës θ.
5. Përsëritja e hapave 1-4 deri sa të arrihet politika optimale π^*.

Avantazhi kryesor i algoritmeve të gradientit të politikës është stabiliteti i tyre në konvergjencë pasi ato përditësojnë politikën aktuale drejtpërdrejt në çdo hap t në vend që të rinovojnë funksionet e vlerës nga të cilat të nxjerrin politikën. Për më tepër, algoritmet e gradientit të politikës mund të trajtojnë hapësira të veprimeve të pafundme dhe të vazhdueshme sepse agjenti vlerëson veprimin drejtpërdrejt në vend që të llogaritë vlerën Q për çdo veprim të mundshëm. Një përfitim tjetër është aftësia e tyre për të mësuar politika stohastike, të cilat mund të jenë të dobishme në kontekste të pasigurta ose në mjedise pjesërisht të vëzhgueshme [22].

Megjithë këto avantazhe, algoritmet e gradientit të politikës kanë një mangësi të rëndësishme: ato kanë tendencën të konvergjojnë në një maksimum lokal në vend të optimumit global. Kjo është për shkak se ato bazohen në logaritmet e probabiliteteve, duke rezultuar në gradientë të zhurmshëm, gjë që mund të shkaktojë mësim të paqëndrueshëm ose konvergjencë në një politikë jo optimale.

Për të adresuar këtë çështje, një qasje hibride e njohur si *arkitektura Aktor-Kritik* u prezantua. Kjo qasje synon të shfrytëzojë pikat e forta të metodave të gradientit të politikës dhe metodave të bazuara në vlera për të përmirësuar stabilitetin dhe konvergjencën.

3.2.2. Arkitektura aktor-kritik

Qasjet e gradientit të politikës së pastër tentojnë të mësojnë ngadalë dhe janë sfiduese për t'u zbatuar në aplikacione online për shkak të variancës së lartë në vlerësime. Megjithatë, metodat e Diferencës Temporale (TD) të diskutuar në Seksionin 2.3.11 mund të adresojnë këto çështje.

Metodat aktor-kritik integrojnë teknikat e *gradientit të politikës* dhe TD për të mësuar njëkohësisht një politikë dhe një funksion$-Q$, duke përdorur këtë të fundit për bootstraping. Politika mësohet nga rrjeti nervor *aktor*, i cili kontrollon veprimet e agjentit. Funksioni$-Q$ mësohet nga rrjeti nervor *kritik*, i cili vlerëson cilësinë e veprimeve të marra nga aktori. Konkretisht, aktori është përgjegjës për politikën, ndërsa kritiku vlerëson funksionin e vlerës (p.sh., funksionin Q-value).

Në kontekstin e Mësimit të Thellë të Përforcimit (DRL), metodat aktor-kritik mund të përfaqësohen duke përdorur rrjete nervore si përafërsues funksionesh: aktori përdor gradientë të nxjerrë nga metoda e gradientit të politikës për të rregulluar parametrat e politikës, ndërsa kritiku vlerëson funksionin e përafërt të vlerës për politikën aktuale π. Faza e mësimit realizohet përmes ndërveprimit midis rrjeteve nervore aktor dhe kritik me mjedisin në çdo hap kohor t si vijon:

1. Aktori është një rrjet që përpiqet të marrë veprimin më të mirë a_t që maksimizon rezultatin e tij bazuar në gjendjen aktuale s_t.

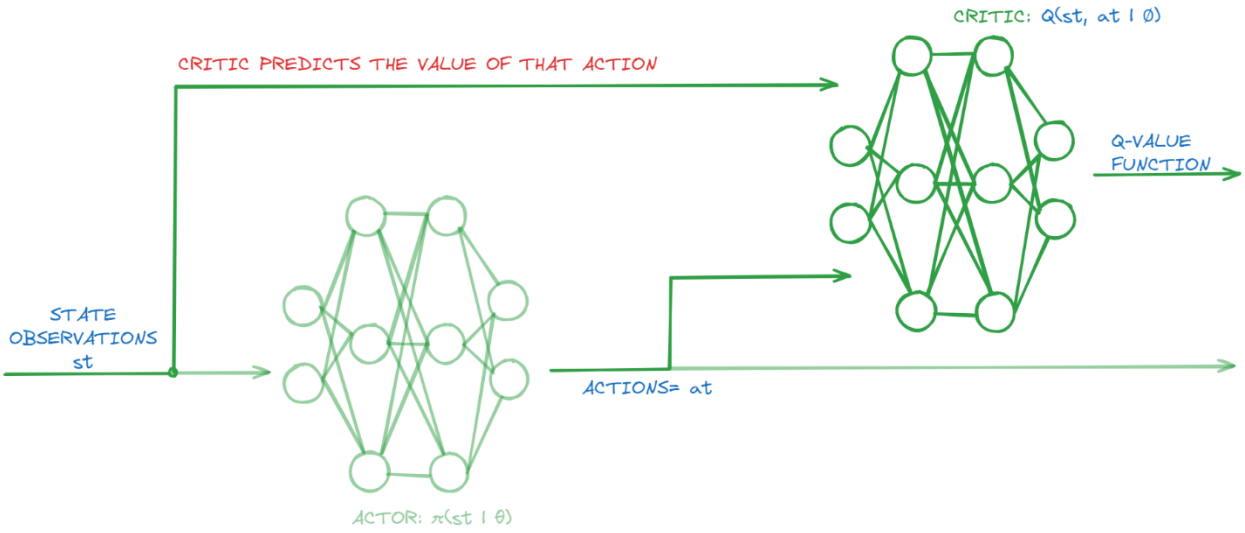


Figura 3.4 Faza e mësimit Aktor-Kritik: Hapi 1

2. Kritiku është një rrjet i dytë që vlerëson funksionin $-Q$ duke marrë si hyrje veprimin e marrë nga aktori a_t në hapin kohor t dhe vëzhgimet nga mjedisi s_{t+1}, dhe r_{t+1}.

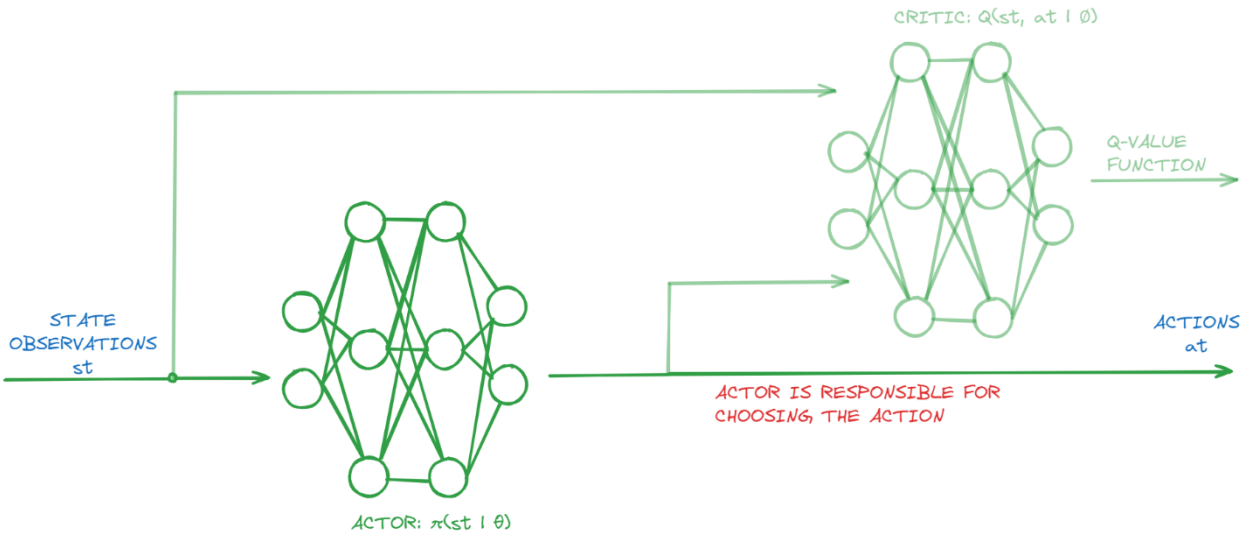


Figura 3.5 Faza e mësimit Aktor-Kritik: Hapi 2

3. Kritiku pastaj përdor shpërblimin r_{t+1} nga mjedisi për të përcaktuar saktësinë e parashikimit të tij të vlerës duke llogaritur gabimin, i cili përcaktohet si diferenca midis vlerës së re të përafërt të gjendjes së mëparshme (*vlera e synuar*) dhe vlerës së vjetër të

gjendjes së mëparshme nga *vlera aktuale* e rrjetit kritik: *gabimi* TD siç përcaktohet në Ekuacionin 2.26 dhe përdoret nga metodat TD. Vlera e re e përafërt bazohet në shpërblimin e marrë dhe vlerën e zbritur të gjendjes aktuale. Gabimi i jep kritikut një ndjesi nëse gjërat shkuan më mirë apo më keq sesa pritej.

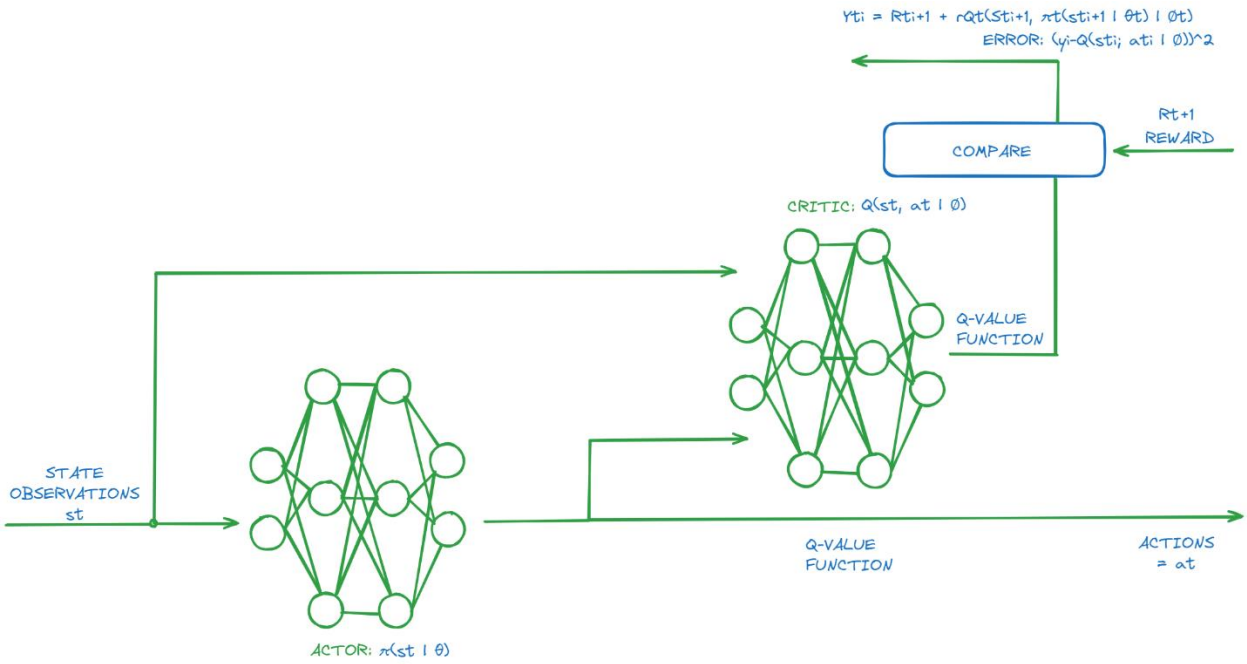


Figura 3.6 Faza e mësimit Aktor-Kritik: Hapi 3

4. Kritiku përdor këtë gabim për t'u përditësuar në të njëjtën mënyrë si do të bënte një funksion i vlerës, duke siguruar parashikime më të mira herën tjetër kur të jetë në këtë gjendje.

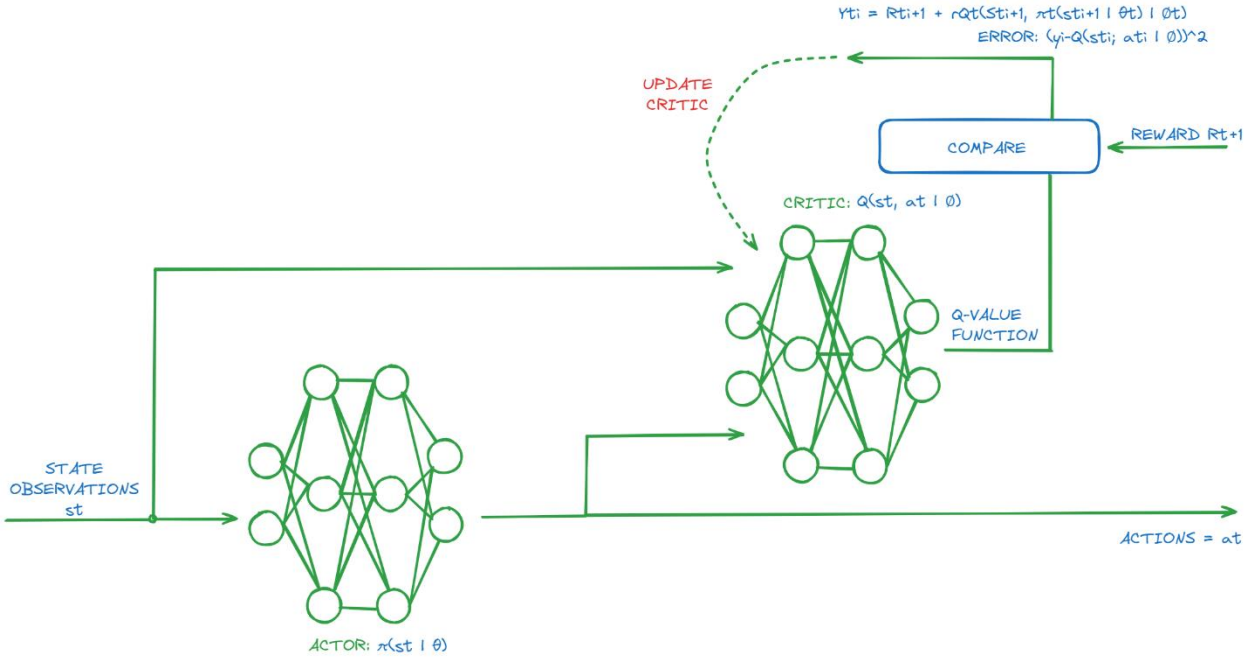


Figura 3.7 Faza e mësimit Aktor-Kritik: Hapi 4

5. Aktori gjithashtu përditësohet duke përdorur daljen e kritikut për të rregulluar probabilitetin e tij për të marrë atë veprim specifik përsëri në të ardhmen.

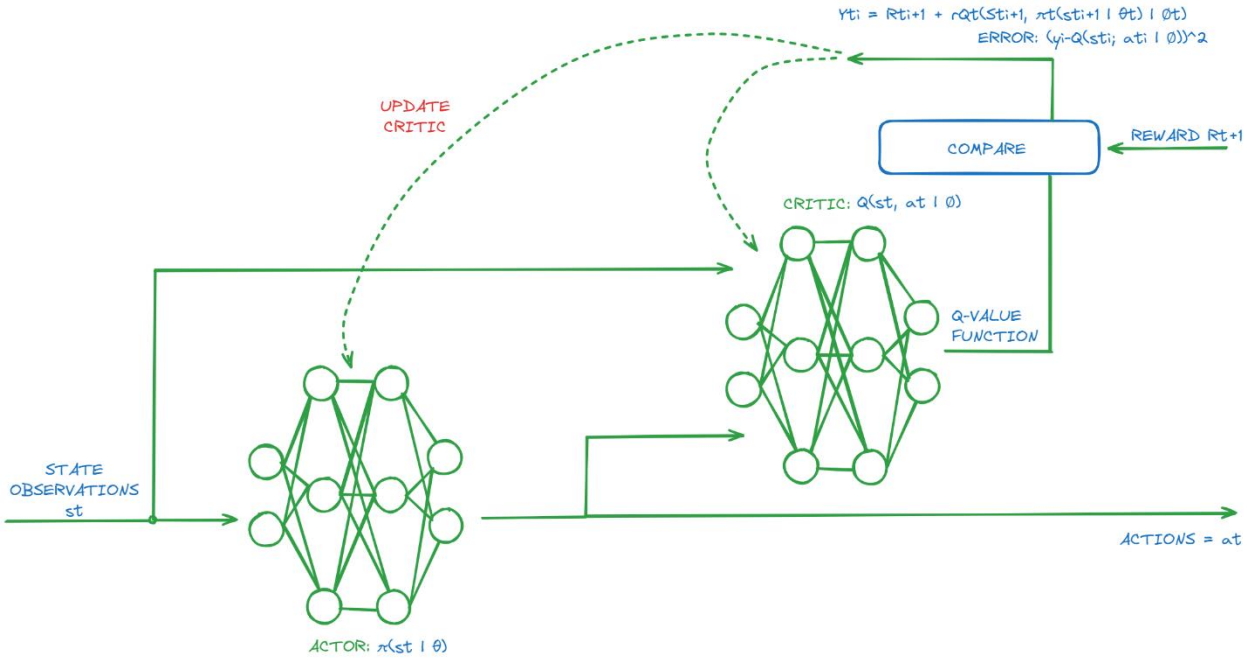


Figura 3.8 Faza e mësimit Aktor-Kritik: Hapi 5

Kombinimi i algoritmeve të gradientit të politikës (PG) dhe teknikave TD tani konsiderohet standard për zgjidhjen e problemeve të Mësimit të Përforcimit (RL) për shkak të performancës dhe stabilitetit të tyre. Shumica e algoritmeve moderne mbështeten në arkitekturën aktor-kritik dhe e zgjerojnë këtë ide bazë në teknika më të sofistikuara dhe komplekse, si algoritmi i Gradientit të Politikës Deterministe të Thellë (DDPG), i cili është një nga teknikat më të përdorura për sistemet e rregullimit, siç do të shpjegohet në seksionin e ardhshëm [23, 24].

3.3. Deep Q-learning

Q-Learning është një algoritëm shumë i përdorur në Mësimin përforcues (RL). Fillimisht, konsiderohej i paqëndrueshëm kur përdorej me rrjete nervore, duke kufizuar aplikimin e tij në detyra dhe probleme që përfshinin hapësira gjendjesh me dimensionalitet të kufizuar. Megjithatë, është demonstruar se algoritmet dhe teknikat Q-Learning mund të përdoren në mënyrë efektive me rrjete nervore. Ky algoritëm ka arritur performancë në nivel njerëzor në shtatë lojëra video në konsolën Atari 2600 duke përdorur vetëm imazhe të papërpunuara të pikselëve si hyrje. Futja e rrjeteve nervore ka zgjatur emrin e algoritmit nga Q-learning në Deep Q-Learning ose Deep Q-Network (DQN).

DQN është një metodë e përafrimit të funksionit të RL që funksionon vetëm me hapësira veprimi dhe gjendjeje të pandërprera. Ajo përfaqëson një evolucion të metodës Q-Learning, ku tabela e veprimit të gjendjes zëvendësohet nga një rrjet nervor i vetëm, rrjeti Kritik, siç përcaktohet në arkitekturën aktor-kritik.

Në këtë algoritëm pa model, online dhe off-policy, mësimi nuk konsiston në përditësimin e një tabele, por në rregullimin e peshave të neuronëve që përbëjnë rrjetin përmes backpropagation.

Mësimi i funksionit të vlerës në DQN bazohet në modifikimin e peshave sipas funksionit të humbjes L_t:

$$L_t = \left(E\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) \right] - Q(s_t, a_t) \right)^2 \tag{3.16}$$

ku $E\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) \right]$ përfaqëson kthimin optimal të pritur, ndërsa $Q(s_t, a_t)$ është rezultati i rrjetit nervor në hapin kohor t.

Gabimet e llogaritura nga funksioni i humbjes përhapen prapa përmes rrjetit duke përdorur backpropagation, duke ndjekur logjikën e zbritjes së gradientit. Gradienti tregon drejtimin e rritjes më të madhe të një funksioni dhe lëvizja në drejtimin e kundërt zvogëlon gabimin. Sjellja e politikës përdor një qasje $\epsilon - greedy$ për të siguruar eksplorim të mjaftueshëm.

Aspekti kryesor që e bën DQN efektiv është përdorimi i riprodhimit të përvojës. Me këtë teknikë, përvoja e agjentit $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ e mbledhur në çdo hap kohor t ruhet në një dataset të formës $D = (e_t, e_{t+1}, \dots, e_T)$ të njohur si *memorja e riprodhimit*.

Trajnimi kryhet përmes një teknike mini-batch, dmth., duke marrë një nën-vëllim të mostrave të përvojës të nxjerra rastësisht nga kjo memorje riprodhimi. Kjo teknikë lejon përdorimin e përvojave të kaluara në më shumë se një përditësim të rrjetit. Për më tepër, nën-vëllimi i zgjedhur rastësisht nga memorja e riprodhimit ndihmon në prishjen e korrelacionit të fortë ndërmjet përvojave të njëpasnjëshme, duke zvogëluar kështu variancën midis përditësimeve.

3.4. Deep Deterministic Policy Gradient

Gradienti i Politikës Deterministike të Thellë (DDPG) është një algoritëm aktor-kritik pa model dhe off-policy, i dizajnuar për hapësira veprimi të vazhdueshme, i prezantuar nga Dr. Lillicrap et al. [3] në vitin 2015. Ky algoritëm është veçanërisht i përshtatshëm për mjedise me hapësira veprimi të vazhdueshme, si shumë detyra fizike të rregullimit, dhe ka demonstruar performancë të lartë në këto skenare [25]. DDPG shtrin dy algoritme themelore: Deep Q-Networks (DQN) dhe Deterministic Policy Gradient (DPG).

DDPG përdor disa teknika të avancuara, duke përfshirë riprodhimin e përvojës dhe rrjetet e synuara, të cilat përmirësojnë stabilitetin dhe efikasitetin e mësimit. Algoritmi njëkohësisht mëson një funksion të vlerës $Q(s_t, a_t)$ dhe një politikë π. Ai përdor të dhëna *off-policy* dhe Ekuacionin e Bellman për të mësuar funksionin $-Q$, i cili më pas ndihmon në mësimin e politikës $\emptyset$:

$$Q_\pi(s_t, a_t) = E_{s_{t+1}\sim E}\left[r_{t+1} + \gamma E_{a_{t+1}\sim\pi}(Q_\pi(s_{t+1}, a_{t+1})) \right] \tag{3.17}$$

ku r_{t+1} është shpërblimi nga mjedisi pas veprimit a_t në hapin e kohës t, $s_{t+1}\sim E$ tregon se tranzicioni është mostruar nga mjedisi E, dhe $a_{t+1}\sim\pi$ do të thotë se veprimi është mostruar nga politika π. Për një politikë deterministike, e shënuar si μ, pritshmëria e brendshme mund të anashkalohet:

$$Q_\mu(s_t, a_t) = E_{s_{t+1}\sim E}\left[r_{t+1} + \gamma Q_\phi(s_{t+1}, \mu(s_{t+1})) \right] \tag{3.18}$$

DDPG mund të mësojë *off-policy* duke përdorur tranzicionet e gjeneruara nga një politikë tjetër β, pasi pritshmëria varet vetëm nga mjedisi. Duke përdorur politikën greedy nga Q-learning, $\mu(s_t) =$

$arg\max_a Q(s_t, a_t)$, dhe duke përfaqësuar funksionin $-Q$ si një afrim funksioni i parametrizuar nga θ_Q, gabimi mesatar i katrorit përdoret si një funksion humbjeje i ngjashëm me qasjen aktor-kritik:

$$L(\theta_Q) = E_{s_t \rho^\beta, r_{t+1} \sim E}\left[(y_t - Q(s_t, a_t)|\theta_Q)^2\right], \tag{3.19}$$

ku

$$y_t = r_{t+1} + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta_Q), \tag{3.20}$$

dhe ρ^β është tranzicioni i zbritur i gjendjes për politikën β. Termi y_t shpesh quhet *vlera e synuar*. DDPG përdor një qasje aktor-kritik ku aktori është një afrim i parametrizuar i një politike deterministike $\mu(s_t|\theta_\mu)$, dhe kritiku është një afrim i parametrizuar i funksionit të vlerës së veprimit $Q(s_t, a_t|\theta_Q)$, të dyja të përfaqësuara nga rrjete nervore të thella. Kritiku $Q(s_t, a_t)$ mësohet duke përdorur ekuacionin e Bellman të ngjashëm me Q-learning, ndërsa aktori mësohet duke përdorur gradientin e politikës. Gradienti i politikës është thjesht gradienti i pritshëm i funksionit të vlerës së veprimit:

$$\nabla_{\theta_\mu} J \approx E_{s_t \sim \rho^\beta}[\nabla_{\theta_\mu} Q(s_t, \mu(s_t|\theta_\mu)|\theta_Q)] \tag{3.21}$$

$$= E_{s_t \sim \rho^\beta}[\nabla_{a_t} Q(s_t, \mu(s_t)|\theta_Q) \nabla_{\theta_\mu} \mu(s_t|\theta_\mu)] \tag{3.22}$$

3.4.1. Përsëritja e Buffers

Trajnimi i rrjetave nervore shpesh supozon se mostrat janë të pavarura dhe të shpërndara në mënyrë identike (i.i.d.). Megjithatë, në mësimin përforcues (RL), mostrat vijnë nga ndërveprime të njëpasnjëshme me mjedisin, duke shkelur këtë supozim. Për më tepër, shfrytëzimi efektiv i optimizimeve të harduerit kërkon trajnim në mini-batches në vend të mësimit online.

Për të adresuar këtë sfidë, përdoret një buffer i ripërsëritjes së përvojës, në metodën Deep Q-learning, në të cilin gjenerohen dhe ruhen në një *memorie përsëritëse* të ndryshme tuple përvoje $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ [26]. Kjo memorie ruan një grup të kufizuar të përvojave të kaluara, duke siguruar që të dy, aktori dhe kritiku, të përditësohen në çdo hap kohor duke përdorur një mini-batch të përzgjedhur uniformisht nga ky buffer. Kjo qasje siguron mostra të pakorrelacionuara për trajnim. Kur bufferat arrijnë kapacitetin e tyre, mostrat e vjetra hidhen për t'i bërë vend të rejave.

Përdorimi i bufferave të përsëritjes ofron përparësi të rëndësishme, si thyerja e korrelacionit midis mostrave të njëpasnjëshme dhe përmirësimi i stabilitetit dhe efikasitetit të procesit të mësimit. Duke mundësuar ripërdorimin e përvojave të kaluara, bufferat e përsëritjes rrisin shpejtësinë e konvergjencës dhe performancën e përgjithshme të algoritmeve RL.

3.4.2. Rrjetet e Objektivit

Për të siguruar mësimin e qëndrueshëm me rrjetet nervore të thella, algoritmi DDPG (Gradienti i Politikës së Përcaktuar të Thellë) përdor një teknikë që përfshin rrjetet e objektivit. Rrjeti kritik $Q(s_t, a_t|\theta_Q)$ përditësohet vazhdimisht ndërkohë që përdoret për të llogaritur vlerën e objektivit të përcaktuar në ekuacionin 3.20 të funksionit të humbjes. Kjo do të thotë që përditësimi i parametrave varet nga parametrat θ që po përditësohen. Kjo varësi mund të çojë në divergjencë në përditësimin e funksionit $-Q$, duke e bërë procesin e mësimit të paqëndrueshëm.

Për të zbutur këtë çështje, algoritmi krijon kopje të të dy rrjeteve, atij të aktorit dhe kritikut, të cilët shënohen si $\mu'(s_t|\theta_{Q_\mu})$ dhe $Q'(s_t, a_t|\theta_{Q'})$, përkatësisht. Këto rrjete objektivi përdoren posaçërisht

për të llogaritur vlerat e objektivit, duke siguruar kështu më shumë qëndrueshmëri në procesin e mësimit.

Ideja themelore është që peshat e rrjeteve të objektivit të inicializohen si kopje të peshave të rrjeteve të aktorit dhe kritikut, por të përditësohen më ngadalë për t'i mbajtur ato të fiksuara për disa hapa kohorë. Kjo arrihet përmes përditësimeve të "butë", të cilat shprehen matematikisht si më poshtë:

$$\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'} \tag{3.23}$$

$$\theta_{\mu'} \leftarrow \tau\theta_\mu + (1-\tau)\theta_{\mu'} \tag{3.24}$$

Në këto ekuacione, τ është një hiperparametër që zakonisht caktohet me një vlerë të vogël (p.sh. 0.001). Mekanizmi i ngadaltë i përditësimit siguron që rrjetet e objektivit të evoluojnë gradualisht, duke përmirësuar ndjeshëm qëndrueshmërinë e përgjithshme të mësimit duke parandaluar ndryshimet e shpejta që mund të çojnë në paqëndrueshmëri dhe divergjencë.

3.4.3. Eksplorimi

Siç u diskutua më parë, balanca midis *eksplorimit* dhe *shfrytëzimit* është një sfidë e rëndësishme në të mësuarit përforcues, veçanërisht në hapësirat e veprimit të vazhdueshëm. Duke qenë se DDPG (Gradienti i Politikës së Përcaktuar të Thellë) funksionon off-policy, problemi i eksplorimit mund të shkëputet nga vetë algoritmi i të mësuarit.

Për të lehtësuar eksplorimin nga agjenti DDPG, zhurma e marrë nga një proces zhurme N futet në politikën e aktorit kur zgjidhet një veprim gjatë trajnimit:

$$\pi(s_t) = \mu\left(s_t \middle| \theta_\mu\right) + N \tag{3.25}$$

Kjo teknikë njihet si zhurma e veprimit. Ky proces ndihmon në gjenerimin e zhurmës të korrelacionuar në kohë, që është e dobishme për eksplorim duke zbutur zhurmën me kalimin e kohës, duke siguruar një sjellje eksploruese më të qëndrueshme.

Për të minimizuar shkallën e dështimit gjatë trajnimit, kështillohet që të ruhet varianca e zhurmës midis 1% dhe 10% të veprimit maksimal të rregullimit të hyrjes. Ky interval siguron që zhurma e shtuar është e mjaftueshme për të nxitur eksplorimin pa destabilizuar procesin e trajnimit.

Për më tepër, hulumtimet e fundit sugjerojnë alternativa si zhurma Gaussiane ose zhurma e parametrave adaptivë, të cilat mund të jenë më të thjeshta për t'u zbatuar dhe ende efektive në ruajtjen e një balance midis eksplorimit dhe shfrytëzimit. Secila metodë ka avantazhet e saj dhe mund të zgjidhet bazuar në kërkesat specifike të detyrës në fjalë [27].

3.5. Përfundimi: Diskutim mbi RL

Në këtë pikë, mund të duket se thjesht vendosja e një mjedisi, vendosja e agjentit RL në të dhe lënia e kompjuterit për të zgjidhur të gjitha problemet është e mjaftueshme ndërsa inxhinieri merret me detyra të tjera. Megjithatë, edhe nëse vendoset një agjent ideal dhe algoritmi RL konvergjon në një zgjidhje, ende ekzistojnë sfida të rëndësishme, duke përfshirë nevojën për ekspertizë për të interpretuar saktë dhe për të përmirësuar rezultatet.

Këto sfida reduktohen në dy pyetje kryesore:

- Si mund të jemi të sigurt se zgjidhja do të funksionojë?
- A mund të përmirësohet zgjidhja edhe nëse duket se është efektive?

Çështja e parë ka të bëjë me kompleksitetin e politikës, e cila përbëhet nga një rrjet nervor me një numër të madh peshash, përthyerjesh dhe funksionesh aktivizimi jolineare. Ndërveprimi i këtyre elementeve krijon një funksion të sofistikuar që mapon vëzhgimet e nivelit të lartë në veprime të nivelit të ulët. Ky funksion është në thelb një kuti e zezë për projektuesin, i cili mund të ketë një intuitë për funksionimin e tij, por nuk mund të kuptojë plotësisht arsyet e vlerave specifike. Një problem tjetër i lidhur me këtë qasje është fuqia kompjuterike dhe memorja e nevojshme. Koha e nevojshme për të aplikuar këto algoritme mund të ndryshojë ndjeshëm, nga disa minuta në disa orë, në varësi të fuqisë kompjuterike të disponueshme dhe hiperparametrave të përdorur. Kjo variabilitet mund të jetë një kufizim i madh, veçanërisht kur krahasohen sisteme të ndryshme. Nga ana tjetër, metodat tradicionale të rregullimit mund të bëhen të papërshtatshme ose të pamundura kur përballen me sisteme shumë jolineare, hapësira të mëdha të gjendjeve dhe veprimeve, ose sisteme që janë të vështira për t'u modeluar (p.sh., një robot që ecën). Megjithatë, këto metoda janë të përshtatshme kur njësia nuk është tepër komplekse dhe dinamika e saj është mirë e kuptuar. Për shkak të këtyre problemeve, zbatimi i algoritmit DDPG, i cili do të diskutohet në kapitullin e ardhshëm, kërkoi shumë kohë për rregullimin e hiperparametrave. Ky proces përfshinte testime të gjera për të arritur një performancë të mirë nga agjenti në sistemin që do të kontrollohej. Procesi i rregullimit duhej të merrte parasysh si kohën e trajnimit, ashtu edhe numrin e madh të hiperparametrave të përfshirë.

4. Zbatimi i DDPG duke përdorur RL MATLAB Toolbox

Reinforcement Learning Toolbox në MATLAB ofron funksione dhe blloqe të ndryshme thelbësore për trajnim politikash duke përdorur algoritme RL, përfshirë Deep Deterministic Policy Gradient (DDPG). Këto politika mund të përdoren për të zhvilluar kontrollorë dhe algoritme të marrjes së vendimeve për sisteme komplekse. Politikat mund të zbatohen duke përdorur Deep Neural Networks (DNNs), ekuacione polinomiale, ose tabela kërkimi. Toolbox-i lejon trajnimin e politikave duke i mundësuar atyre të ndërveprojnë me mjedise të modeluara duke përdorur skripta MATLAB ose blloqe Simulink. Për më tepër, toolbox-i lehtëson vlerësimin e algoritmeve, eksperimentimin me cilësimet e hiperparametrave dhe monitorimin e progresit të trajnimit.

Në këtë seksion, do të demonstrohet implementimi i algoritmit DDPG në sisteme dinamike lineare dhe jolineare për kontroll optimal për të vlerësuar performancën e algoritmit. Implementimi do të analizohet hap pas hapi duke ndjekur diagramin e mëposhtëm:

1. **Formulimi i Problemit:** Përcaktimi i detyrës që agjenti duhet të mësojë, duke përfshirë se si agjenti ndërvepron me mjedisin nga perspektiva e vëzhgimeve dhe veprimeve, dhe çdo qëllim parësor dhe dytësor që agjenti duhet të arrijë.
2. **Krijimi i Mjedisit:** Përcaktimi i mjedisit në të cilin operon agjenti, duke përfshirë ndërfaqen midis agjentit dhe mjedisit dhe modelin dinamik të mjedisit.
3. **Përcaktimi i Shpërblimit:** Hartimi i sinjalit të shpërblimit që agjenti përdor për të matur performancën e tij në lidhje me qëllimet e detyrës dhe përshkrimi se si ky sinjal llogaritet nga mjedisi.
4. **Krijimi i Agjentit:** Përcaktimi i agjentit, duke përfshirë përkufizimin e një përfaqësimi të politikës dhe konfigurimin e algoritmit të të mësuarit të agjentit dhe hiperparametrat e tij.
5. **Trajnimi i Agjentit:** Trajnon përfaqësimin e politikës së agjentit duke përdorur mjedisin e përcaktuar, sinjalin e shpërblimit dhe algoritmin e të mësuarit.
6. **Validimi i Agjentit:** Vlerësimi i performancës së agjentit të trajnuar duke simuluar agjentin brenda mjedisit për të siguruar që ai plotëson kriteret e dëshiruara të performancës. [28]

4.1. Lavjerrësi i vetëm i përmbysur

Kontrollimi i ekuilibrit të një Lavjerrësi të Përmbysur të montuar në një pajisje Quanser IP02, duke përdorur algoritmin DDPG, është zbatuar me sukses. Lavjerrësi i përmbysur është një problem klasik në teorinë e rregullimit, shpesh i përdorur për të ilustruar dhe zhvilluar strategji të ndryshme rregullimi.

Sistemi Quanser IP02 është përdorur në këtë studim. Siç tregohet në Figurën 4.1, Lavjerrësi i Përmbysur përfshin një karrocë që lëviz përgjatë një shtrati dhe një shufër që lëkundet rreth boshtit të karrocës. Megjithë thjeshtësinë e dukshme, mekanika e lavjerrësit paraqet një kompleksitet të konsiderueshëm, që kërkon ekuacione të gjata për të përshkruar lëvizjen e tij saktësisht.

Enkoderët përdoren për të matur pozicionin e karrocës (d.m.th., këndi i motorit DC) dhe këndin e lavjerrësit, si dhe normat e ndryshimit të tyre, duke përkufizuar daljen e sistemit:

$$y_t = y_{t_{meas}} = [x \quad \alpha \quad \dot{x} \quad \dot{\alpha}] \tag{4.1}$$

Sistemi Quanser IP02 është i njohur për aftësinë e tij për të lehtësuar studimin e strategjive të rregullimit linear dhe jolinear, duke e bërë atë një platformë të shkëlqyer për zbatimin dhe testimin e algoritmeve të avancuara si DDPG.

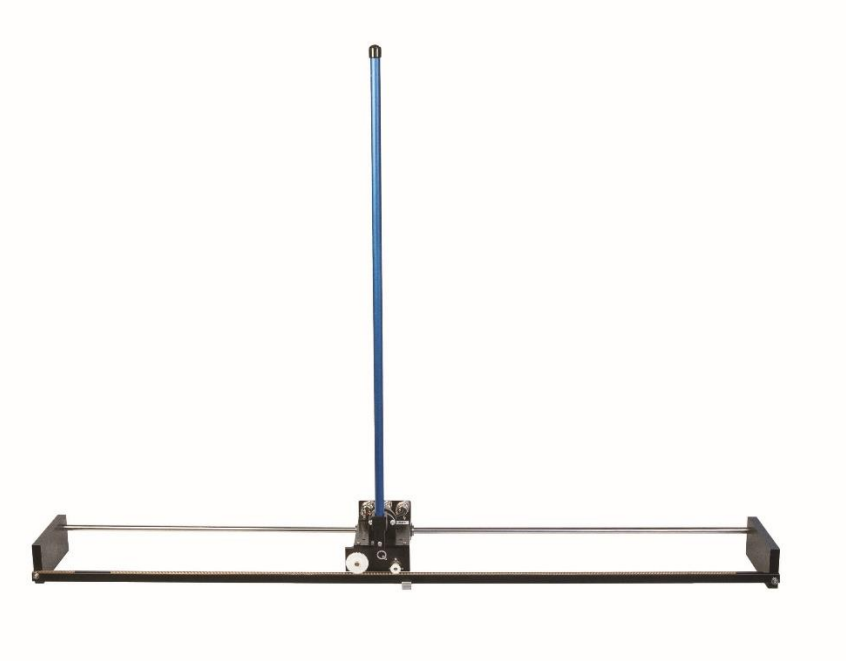


Figura 4.1 Quanser IP02 Lavjerrësi i vetëm i përmbysur [29]

4.1.1. Problemi i kontrollit të pozicionit

Kontrolli i një lavjerrësi mbetet një nga sfidat themelore në fushën e sistemeve të rregullimit. Stabilizimi i lavjerrësit në pozicionin e drejtë është një detyrë kritike, shpesh e arritur duke përdorur një teknikë të rregullatorit linear kuadratik (LQR), e cila është e njohur për efektivitetin e saj. Qëllimi kryesor i zbatimit të algoritmit model-pavarur Deep Deterministic Policy Gradient (DDPG) është që të mundësojë agjentin jo vetëm të mësojë dinamikën e sistemit, por gjithashtu të stabilizojë atë dhe të lëkundë lavjerrësin në pozicionin vertical [30]. Ky pozicion përcaktohet në origjinën e sistemit të referencës: $\alpha_0 = 0\ [rad]$ dhe $x_0 = 0\ [cm]$.

Në këtë kontekst, për problemin Linear Quadratic Tracker (LQT) me horizont të pafund, qëllimi është të projektohet një kontrollues optimal për sistemin. Ky kontrollues siguron që dalja $y_t = y_{t_{meas}} = [x \quad \alpha \quad \dot{x} \quad \dot{\alpha}]$ të ndjekë trajektoren e pozicionit të referencës $y_{t_{ref}} = [x \quad \alpha \quad \dot{x} \quad \dot{\alpha}] = [0 \quad 0 \quad 0 \quad 0]$. Kontrolluesi vepron mbi diferencën midis këtyre dyjave, e cila përcaktohet si gabimi i kontrollit e_t:

$$e_t = y_{t_{meas}} - y_{t_{ref}} \tag{4.2}$$

Për këtë studim, konsiderohet Lavjerrësi i Përmbysur i montuar në një pajisje servo Quanser IP02. Dinamika e sistemit përfshin aftësinë e lavjerrësi për të lëkundur dhe stabilizuar në pozicionin vertikal, pavarësisht nga sfidat dhe kompleksitetet e natyrshme të lëvizjes së lavjerrësit. Enkoderët përdoren për të matur pozicionin dhe normën e ndryshimit të krahut rrotativ dhe lidhjes së lavjerrësit, duke ofruar reagime kritike për kontrollin [29].

4.1.2. Krijimi i Mjedisit

Mjedisi i përdorur për të trajnuar agjentin është një model dinamik jolinear i sistemit të lavjerrësit të përmbysur, të montuar në një pajisje Quanser IP02. Ky mjedis është i përcaktuar në bllokun e modelit të lavjerrësit Simulink Quanser IP02 të ofruar nga MATLAB, i cili përmban të gjitha ekuacionet e dinamikës reale të lavjerrësit të përmbysur dhe parametrat fizikë të harduerit. Sistemi përbëhet nga një krah motorik, i cili është i aktivizuar nga një motor DC servo, me një krah lavjerrësi të varur në fund të tij. Ky sistem është sfidues për t'u kontrolluar sepse është i nën-vepruar, shumë jolinear dhe me fazë jo minimale.

4.1.2.1. Modelizimi i Sistemit

Modeli i lavjerrësit të vetëm të përmbysur të montuar në një sistem servo Quanser IP02 është paraqitur në Figurën 4.2. Ky konfigurim përfshin një karrocë të aktivizuar nga një motor DC që drejton lëvizjen e lavjerrësit. Pozita e karrocës, x_c, dhe këndi i lavjerrësit, α, maten të dyja duke përdorur enkoder. Drejtimi pozitiv i x_c përcaktohet si në të djathtë kur shikohet karroca, dhe α është zero kur lavjerrësi është plotësisht i drejtë.

Karroca ka një masë M_c dhe një gjatësi L_c, me momentin e saj të inercisë të dhënë nga:

$$J_c = \frac{M_c {L_c}^2}{3} \tag{4.3}$$

Lidhja e lavjerrësit, e lidhur në fund të karrocës, ka një gjatësi totale L_p dhe qendra e masës së saj ndodhet në $\frac{L_p}{2}$. Momenti i inercisë së lavjerrësit rreth qendrës së masës së tij është:

$$J_p = \frac{M_p {L_p}^2}{3} \tag{4.4}$$

ku M_p është masa totale e lidhjes së lavjerrësit. Këndi i lavjerrësit α rritet pozitivisht kur rrotullohet në drejtimin kundër akrepave të orës (CCW).

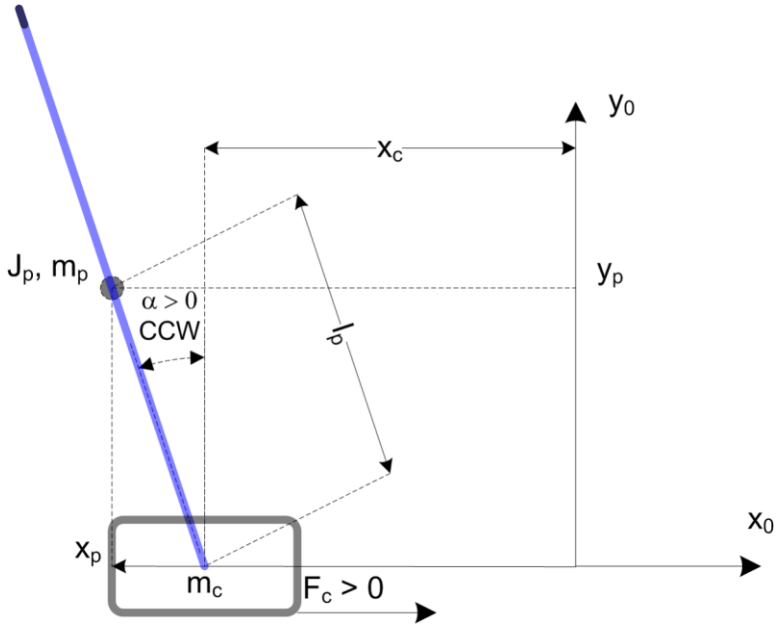


Figura 4.2 Skema lineare e lavjerrësit të përmbysur

4.2.2.1. Ekuacionet e Lëvizjes

4.2.2.1.1. Metoda e Lagranzhit

Metoda e energjisë së Lagranzhit është një qasje analitike e fuqishme e përdorur për të nxjerrë modelin dinamik të një sistemi. Në këtë metodë, fokusohemi në forcën drejtuese, F_c, e cila gjenerohet nga motori DC dhe transmetohet te karroca përmes pinionit të motorit, duke e konsideruar atë si hyrjen kryesore. Hapi i parë në zbatimin e metodës së Lagranzhit përfshin përcaktimin e Lagranzhit të sistemit. Lagranzhi përcaktohet si ndryshimi ndërmjet energjisë totale kinetike dhe energjisë totale potenciale të sistemit. Duke llogaritur saktë këto përbërës energjie, ne mund të nxjerrim ekuacionet e lëvizjes që përshkruajnë dinamikën e sistemit. Kjo qasje ofron një mënyrë sistematike për të trajtuar kompleksitetet e përfshira në modelimin e sistemeve dinamike, veçanërisht kur kemi të bëjmë me shkallë të shumëfishta lirie ose kufizime.

Në kontekstin e sistemit tonë, energjia kinetike përfshin kontribute nga lëvizja translative e karrocës dhe lëvizja rrotulluese e pinionit të motorit. Energjia potenciale, nga ana tjetër, është zakonisht e lidhur me forcat gravitacionale që veprojnë mbi karrocën. Pasi të formulohet Lagranzhit, ne aplikojmë ekuacionet Euler-Lagrazhit për të marrë ekuacionet dinamike që udhëheqin sistemin. Kjo metodë jo vetëm që thjeshton procesin e nxjerrjes së këtyre ekuacioneve, por gjithashtu siguron që parimet e ruajtjes së energjisë të respektohen në mënyrë të qenësishme.

4.2.2.1.2. Energjia potenciale

Energjia totale potenciale, $\boldsymbol{V}_T$, në një sistem përfaqëson energjinë e ruajtur për shkak të pozicionit ose konfigurimit të tij. Kjo energji mund t'i atribuohet llojeve të ndryshme të punës së kryer në sistem, të tilla si ngritja kundër gravitetit ose kompresimi i një suste. Specifikisht, energjia potenciale mund të kategorizohet në energji potenciale gravitacionale, e cila buron nga zhvendosja vertikale e një objekti, dhe energji potenciale elastike, e cila lidhet me deformimin e materialeve elastike si sustat.

Në kontekstin e sistemit të Lavjerrësit të Vetëm të Përmbysur (SIP), energjia potenciale është kryesisht për shkak të efekteve gravitacionale. Meqenëse lëvizja e karrocës është e kufizuar në një plan horizontal pa zhvendosje vertikale, energjia totale potenciale përcaktohet vetëm nga lartësia e lavjerrësit në raport me pikën e tij më të ulët. Shprehja matematikore për energjinë potenciale gravitacionale të lavjerrësit jepet nga:

$$\boldsymbol{V}_T = M_p g y_p = M_p g l_p \cos\alpha \tag{4.5}$$

Ky ekuacion nxjerr në pah varësinë e energjisë potenciale nga pozicioni i lavjerrësit, duke forcuar konceptin që parimet e ruajtjes së energjisë janë në veprim në sistemet dinamike [31].

4.2.2.1.3. Energjia Kinetike

Në këtë seksion, ne do të përcaktojmë energjinë totale kinetike të sistemit, e shënuar si $\boldsymbol{K}_T$. Energjia kinetike mat sasinë e energjisë që posedon një sistem për shkak të lëvizjes së tij. Për sistemin e Lavjerrësit të Vetëm të Përmbysur (SIP), energjia totale kinetike rrjedh nga kombinimi i energjive kinetike translative dhe rrotulluese të karrocës së motorizuar dhe lavjerrësit të përmbysur të montuar në të.

Së pari, energjia kinetike translatore e karrocës së motorizuar, $\boldsymbol{K}_{ct}$, shprehet si:

$$\boldsymbol{K}_{ct} = \frac{1}{2} M \dot{x}^2 \tag{4.6}$$

ku M është masa e karrocës dhe $\dot{x}$ është shpejtësia e saj.

Energjia kinetike rrotulluese e karrocës, K_{cr}, buron nga rrotullimi i boshtit dalës të motorit DC, e cila jepet nga:

$$\boldsymbol{K}_{cr} = \frac{1}{2} J_m \omega_m^2 \tag{4.7}$$

Këtu, J_m është momenti i inercisë së motorit, dhe ω_m është shpejtësia këndore e boshtit të motorit. Sipas specifikimeve teknike të motorit DC dhe planetareve të ingranazheve, shpejtësia këndore e boshtit të motorit ω_m mund të lidhet me shpejtësinë këndore të pinionit të motorit ω me:

$$\omega_m = K_g \omega \tag{4.8}$$

Duke marrë parasysh mekanizmin e ingranazheve dhe pinionit, kjo mund të rishkruhet në terma të shpejtësisë së karrocës si:

$$K_g \omega = \frac{K_g \dot{x}}{r_{mp}} \tag{4.9}$$

Duke zëvendësuar ekuacionet (4.8) dhe (4.9) në (4.7), energjia kinetike rrotulluese e karrocës mund të reformulohet si:

$$\boldsymbol{K}_{cr} = \frac{J_m K_g^2 \dot{x}^2}{2 r_{mp}^2} \tag{4.10}$$

Për lavjerrësin e vetëm të përmbysur, masa e tij supozohet të jetë e përqendruar në qendrën e gravitetit (COG). Shpejtësia e COG, v_{COG}, jepet nga:

$$v_{COG} = \sqrt{\dot{x}_p^2 + \dot{y}_p^2} \tag{4.11}$$

Bazuar në përkufizimin e sistemit referues në Figurën 4.2, koordinatat kartesiane të qendrës së gravitetit të lavjerrësit janë:

$$x_p = x_c - l_p \sin\alpha \;\; dhe \; y_p = l_p \cos\alpha \tag{4.12}$$

Duke diferencuar ekuacionin (4.11), mund të rishkruajmë ekuacionin (4.10) si:

$$v_{COG} = \sqrt{\dot{x}_c^2 - 2 l_p \cos\alpha \, \dot{x}_c \dot{\alpha} + l_p^2 \dot{\alpha}^2} \tag{4.13}$$

Kështu, energjia kinetike translatore e lavjerrësit, $\boldsymbol{K}_{pt}$, është:

$$\boldsymbol{K}_{pt} = \frac{1}{2} M_p v_{COG}^2 = \frac{1}{2} M_p \left(\dot{x}_c^2 - 2 l_p \cos\alpha \, \dot{x}_c \dot{\alpha} + l_p^2 \dot{\alpha}^2 \right) \tag{4.14}$$

Gjithashtu, energjia kinetike rrotulluese e lavjerrësit në COG, $\boldsymbol{K}_{pr}$, jepet nga:

$$\boldsymbol{K}_{pr} = \frac{1}{2} I_p \dot{\alpha}^2 \tag{4.15}$$

ku momenti i inercisë së lavjerrësit në COG, I_p, është:

$$I_p = \int_{-l_p}^{l_p} r^2 \frac{M_p}{2l_p} dr = \frac{1}{3} M_p l_p^2 \tag{4.16}$$

Prandaj, energjia totale kinetike $\boldsymbol{K}_T$ e sistemit, e cila është shuma e katër energjive kinetike individuale të dhëna nga ekuacionet (4.6), (4.10), (4.14), dhe (4.15), mund të shprehet si:

$$\boldsymbol{K}_T = \frac{1}{2}\left(M + M_p + \frac{J_m K_g^2}{r_{mp}^2}\right)\dot{x}^2 - M_p l_p \cos\alpha\, \dot{x}\dot{\alpha} + \frac{2}{3} M_p l_p^2 \dot{\alpha}^2 \tag{4.17}$$

4.2.2.1.4. Ekuacionet e Lagranzhit

Metoda e Lagranzhit përdoret për të gjetur ekuacionet e lëvizjes së sistemit, e cila është një metodë sistematike e përdorur shpesh për sisteme më të ndërlikuara si robotët manipulues me nyje të shumta.

Më konkretisht, ekuacionet që përshkruajnë lëvizjet e krahut rrotullues dhe lavjerrësit në lidhje me tensionin e motorit do të merren duke përdorur ekuacionin Euler-Lagrange. Lagrangiani, L, jepet nga diferenca midis energjisë totale kinetike dhe energjisë totale potenciale,

$$\boldsymbol{L} = \boldsymbol{K}_T - \boldsymbol{V}_T \tag{4.18}$$

Zëvendësimi i ekuacioneve (4.5) dhe (4.17) në (4.18) jep:

$$\boldsymbol{L} = \frac{1}{2}\left(M + M_p + \frac{J_m K_g^2}{r_{mp}^2}\right)\dot{x}^2 - M_p l_p \cos\alpha\, \dot{x}\dot{\alpha} + \frac{2}{3} M_p l_p^2 \dot{\alpha}^2 - M_p g l_p \cos\alpha \tag{4.19}$$

Me përkufizim, dy ekuacionet e Lagranzhit për sistemin tonë janë:

$$\frac{\partial^2}{\partial t \partial \dot{x}} \boldsymbol{L} - \frac{\partial}{\partial x} \boldsymbol{L} = F_c - B_{eq}\dot{x} \tag{4.20}$$

dhe

$$\frac{\partial^2}{\partial t \partial \dot{\alpha}} \boldsymbol{L} - \frac{\partial}{\partial \alpha} \boldsymbol{L} = B_p \dot{\alpha} \tag{4.21}$$

ku B_{eq} është koeficienti i viskozitetit ekuivalent në pinionin e motorit, dhe B_p është koeficienti i viskozitetit ekuivalent në boshtin e lavjerrësit. Kështu, ekuacionet (4.20) dhe (4.21) përfshijnë fërkimin në formën e viskozitetit ekuivalent. Sidoqoftë, duhet të theksohet se në këtë model, fërkimi i tipit Coulomb (jo-linear) i aplikuar në karrocë dhe forca në karrocë për shkak të veprimit të lavjerrësit janë neglizhuar.

Duke përdorur ekuacionin (4.19), ana e majtë e ekuacionit (4.20) mund të shprehet si:

$$\frac{\partial^2}{\partial t \partial \dot{x}} \boldsymbol{L} - \frac{\partial}{\partial x} \boldsymbol{L} = \frac{\partial}{\partial t} ((M + M_p + \frac{J_m K_g^2}{r_{mp}^2}) \dot{x} - M_p l_p \cos \alpha \, \dot{\alpha}) \tag{4.22}$$

$$\frac{\partial^2}{\partial t \partial \dot{x}} \boldsymbol{L} - \frac{\partial}{\partial x} \boldsymbol{L} = \left(M + M_p + \frac{J_m K_g^2}{r_{mp}^2} \right) \ddot{x} + M_p l_p \sin \alpha \, \dot{\alpha}^2 - M_p l_p \cos \alpha \, \ddot{\alpha} \tag{4.23}$$

Në mënyrë të ngjashme, ana e majtë e ekuacionit (4.21) mund të shkruhet si:

$$\frac{\partial^2}{\partial t \partial \dot{\alpha}} \boldsymbol{L} - \frac{\partial}{\partial \alpha} \boldsymbol{L} = \frac{\partial}{\partial t} \left(-M_p l_p \cos \alpha \, \dot{x} + \frac{4}{3} M_p l_p^2 \dot{\alpha} \right) - M_p l_p g \sin \alpha \tag{4.24}$$

$$\frac{\partial^2}{\partial t \partial \dot{\alpha}} \boldsymbol{L} - \frac{\partial}{\partial \alpha} \boldsymbol{L} = -M_p l_p \cos \alpha \, \ddot{x} + \frac{4}{3} M_p l_p^2 \ddot{\alpha} - M_p l_p g \sin \alpha \tag{4.25}$$

Duke zëvendësuar ekuacionin (4.23) në ekuacionin (4.20), ne marrim:

$$\left(M + M_p + \frac{J_m K_g^2}{r_{mp}^2} \right) \ddot{x} + M_p l_p \sin \alpha \, \dot{\alpha}^2 - M_p l_p \cos \alpha \, \ddot{\alpha} = F_c - B_{eq} \dot{x} \tag{4.26}$$

Në mënyrë të ngjashme, zëvendësimi i ekuacionit (4.25) në ekuacionin (4.21) jep:

$$-M_p l_p \cos \alpha \, \ddot{x} + \frac{4}{3} M_p l_p^2 \ddot{\alpha} - M_p l_p g \sin \alpha = -B_p \dot{\alpha} \tag{4.27}$$

Zgjidhja e ekuacioneve (4.26) dhe (4.27) për derivatet e rendit të dytë të x dhe α rezulton në dy ekuacionet jo-lineare:

$$\begin{aligned} \ddot{x} = &- \frac{3 r_{mp}^2 B_p \cos \alpha \, \dot{\alpha}}{l_p D(\alpha)} - \frac{4 M_p l_p r_{mp}^2 \sin \alpha \, \dot{\alpha}^2}{D(\alpha)} - \frac{4 r_{mp}^2 B_{eq} \dot{x}}{D(\alpha)} \\ &+ \frac{3 M_p r_{mp}^2 g \cos \alpha \sin \alpha}{D(\alpha)} + \frac{4 r_{mp}^2 F_c}{D(\alpha)} \end{aligned} \tag{4.28}$$

dhe

$$\begin{aligned} \ddot{\alpha} = &- \frac{3 \left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2 \right) B_p \dot{\alpha}}{M_p l_p^2 D(\alpha)} - \frac{3 M_p r_{mp}^2 \cos \alpha \sin \alpha \, \dot{\alpha}^2}{D(\alpha)} \\ &- \frac{3 r_{mp}^2 B_{eq} \cos \alpha \, \dot{x}}{l_p D(\alpha)} + \frac{3 \left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2 \right) g \sin \alpha}{l_p D(\alpha)} \\ &+ \frac{3 r_{mp}^2 \cos \alpha \, F_c}{l_p D(\alpha)} \end{aligned} \tag{4.29}$$

ku $D(\alpha) = 4 M r_{mp}^2 + M_p r_{mp}^2 + 4 J_m K_g^2 + 3 M_p r_{mp}^2 \sin^2 \alpha$. Ekuacionet (4.28) dhe (4.29) përfaqësojnë Ekuacionet e Lëvizjes (EOM) të sistemit të lavjerrësit të vetëm të përmbysur (SIP).

4.2.2.1.5. Konvertimi i hyrjes në tension

Në implementimin tonë në kohë reale, hyrja e sistemit është e barabartë me tensionin e motorit DC të karrocës, V_m. Prandaj, duhet të konvertojmë forcën drejtues të gjeneruar nga motori DC që vepron mbi karrocën përmes pinionit të motorit në hyrje të tensionit. Për ta bërë këtë, do të përdorim skemën elektrike të qarkut të armaturës së një motori standard DC, siç tregohet në Figurën 4.3.

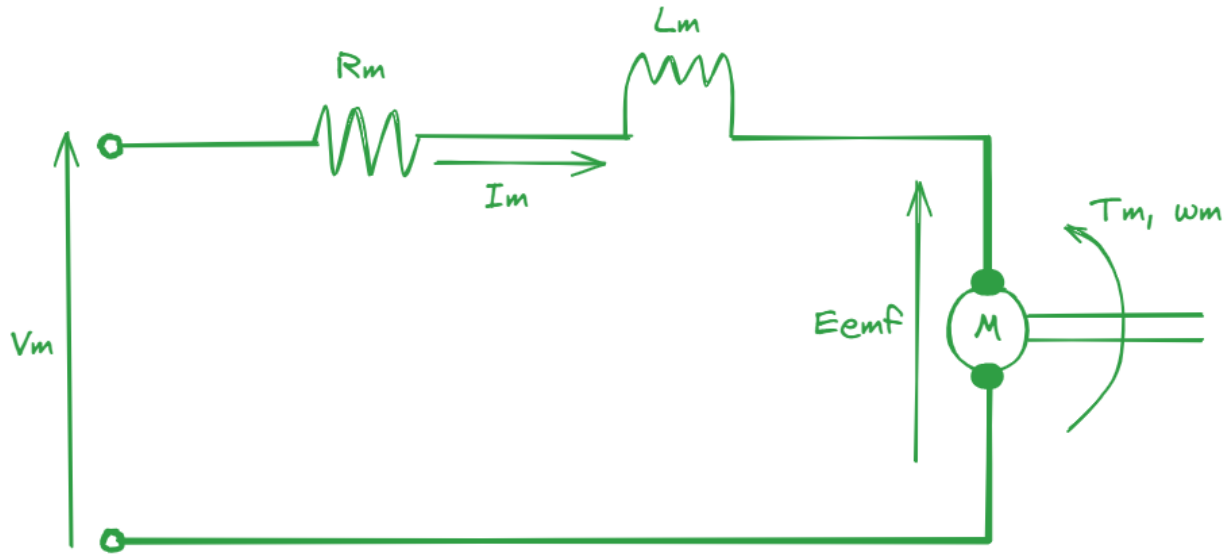


Figura 4.3 Skema elektrike e motorit standard DC

Forca drejtues F_c mund të shprehet si:

$$F_c = \frac{K_g T_m}{r_{mp}} \tag{4.30}$$

Sipas ligjit të tensionit të Kirchhoff-it, shuma e drejtuar e ndryshimeve të potencialit elektrik rreth çdo qarku të mbyllur është zero. Për sistemin tonë, kjo përkthehet në:

$$V_m - R_m I_m - L_m \left(\frac{d}{dt} I_m\right) - E_{emf} = 0 \tag{4.31}$$

Duke qenë se L_m është shumë më i vogël se R_m, mund të neglizhojmë induktancën e motorit, duke e thjeshtuar ekuacionin në:

$$I_m = \frac{V_m - E_{emf}}{R_m} \tag{4.32}$$

Tensioni i forcës elektromotore të kundërt (EMF) i gjeneruar nga motori është proporcional me shpejtësinë e boshtit të motorit, i shprehur si $E_{emf} = K_m \omega_m$. Duke e zëvendësuar këtë në ekuacionin (4.32), marrim:

$$I_m = \frac{V_m - K_m \omega_m}{R_m} \tag{4.33}$$

ku K_m është konstanta e EMF-së së kundërt. Për më tepër, momenti i forcës T_m i prodhuar nga motori DC lidhet me rrymën I_m me:

$$T_m = K_t I_m \tag{4.34}$$

Duke zëvendësuar ekuacionet (4.33) dhe (4.34) në ekuacionin (4.30), marrim:

$$F_c = \frac{K_g K_t (V_m - K_m \omega_m)}{R_m r_{mp}} \tag{4.35}$$

Duke përfshirë ekuacionet (4.8) dhe (4.9) në ekuacionin (4.35) dhe duke e riorganizuar, përftojmë:

$$F_c = -\frac{K_g K_t K_m \dot{x}}{R_m r_{mp}^2} + \frac{K_g K_t V_m}{R_m r_{mp}} \tag{4.36}$$

Duke përdorur ekuacionin (4.36) për të konvertuar forcën drejtues në hyrje të tensionit, ne mund të rishkruajmë ekuacionet e lëvizjes (EOM) të dhëna në ekuacionet (4.28) dhe (4.29) si:

$$\begin{aligned} \ddot{x} = &-\frac{3 r_{mp}^2 B_p \cos\alpha\, \dot{\alpha}}{l_p D(\alpha)} - \frac{4 M_p l_p r_{mp}^2 \sin\alpha\, \dot{\alpha}^2}{D(\alpha)} - \frac{4\left(R_m r_{mp}^2 B_{eq} + K_g^2 K_t K_m\right)\dot{x}}{R_m D(\alpha)} \\ &+ \frac{3 M_p r_{mp}^2 g \cos\alpha \sin\alpha}{D(\alpha)} + \frac{4 r_{mp} K_g K_t V_m}{R_m D(\alpha)} \end{aligned} \tag{4.37}$$

dhe

$$\begin{aligned} \ddot{\alpha} = &-\frac{3\left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\right) B_p \dot{\alpha}}{M_p l_p^2 D(\alpha)} - \frac{3 M_p r_{mp}^2 \cos\alpha \sin\alpha\, \dot{\alpha}^2}{D(\alpha)} \\ &- \frac{3\left(R_m r_{mp}^2 B_{eq} + K_g^2 K_t K_m\right)\cos\alpha\, \dot{x}}{R_m l_p D(\alpha)} \\ &+ \frac{3\left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\right) g \sin\alpha}{l_p D(\alpha)} + \frac{3 r_{mp} K_g K_t \cos\alpha\, V_m}{R_m l_p D(\alpha)} \end{aligned} \tag{4.38}$$

Parametrat fizikë të Lavjerrësit të Vetëm të Përmbysur (SIP) janë detajuar në Tabelën 4.1.

Parametri	**Përshkrimi**	**Vlera**
B_p	Koeficienti i zbutjes viskoze, siç shihet në boshtin e lavjerrësit	0.0024 $[\frac{N \cdot m \cdot s}{rad}]$

B_{eq}	Koeficienti ekuivalent i zbutjes viskoze siç shihet në pinionin e motorit	5.4 $[\frac{N \cdot m \cdot s}{rad}]$
g	Konstanta gravitacionale	9.81 $[\frac{m}{s^2}]$
I_p	Momenti i inercisë së lavjerrës, rreth qendrës së gravitetit të tij	8.359E-003 $[kg \cdot m^2]$
J_p	Momenti i inercisë së lavjerrësit në varsen e tij	3.344E-002 $[kg \cdot m^2]$
J_m	Momenti i inercisë së rotorit	3.90E-007 $[kg \cdot m^2]$
K_g	Raporti i dhëmbëzorit të Kutisë Planetare	3.71
K_t	Konstanta e momentit të forcës së motorit	0.00767$[\frac{N \cdot m}{A}]$
K_m	Konstanta e Forcës Elektro-Motore të Kundërt (EMF)	0.00767$[\frac{V \cdot s}{rad}]$
l_p	Gjatësia e Lavjerrësit nga Pika e Lidhjes deri te Qendra e Gravitetit	0.3302 $[m]$
M_w	Masa e Peshës së Karrocës	0.37 $[kg]$
M	Masa e Karrocës me Peshë Shtesë	0.57+M_w $[kg]$
M_p	Masa e Lavjerrësit	0.230 $[kg]$
R_m	Rezistenca e Armaturës së Motorit	2.6 $[\Omega]$
r_{mp}	Rrezja e Pinionit të Motorit	6.35E-003 $[m]$

Tabela 4.1 Parametrat fizikë të Lavjerrësit të Vetëm të Përmbysur (SIP)

4.2.2.2. Modeli Linear i Hapsirës së Gjendjes

Modeli linear i hapsirës së gjendjes për sistemin e Lavjerrësit të Vetëm të Përmbysur (SIP) është paraqitur në Figurën 4.2. Bazuar në Ekuacionet e Lëvizjes, (4.37) dhe (4.38), përfaqësimi i hapsirës së gjendjes së sistemit tonë ka formën:

$$\frac{d}{dt}X(t) = f\big(X(t)\big) + B\big(X(t)\big)u(t) \tag{4.39}$$

ku X, vektori i gjendjes së sistemit, është dhënë nga $X^T(t) = [x(t), \alpha(t), \dot{x}(t), \dot{\alpha}(t)] = [x_1, x_2, x_3, x_4]$, dhe hyrja u është e barabartë me tensionin e hyrjes së motorit DC të karrocës, pra $u = V_m$. Funksioni jolinear $f(X)$ mund të shprehet si:

$$\boldsymbol{f}(X) = \frac{1}{J_T}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & a_{33} & a_{44} \\ 0 & 0 & a_{43} & a_{44} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{3M_p r_{mp}^2 g \cos(x_2)\sin(x_2)}{D(x_2)} \\ \dfrac{3\big(Mr_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\big) g \sin(x_2)}{l_p D(x_2)} \end{bmatrix} \tag{4.40}$$

ku:

$$a_{33} = -\frac{4\left(R_m r_{mp}^2 B_{eq} + K_g^2 K_t K_m\right)}{R_m D(x_2)} \qquad a_{34} = -\frac{3r_{mp}^2 B_p \cos(x_2) + 4M_p l_p^2 r_{mp}^2 \sin(x_2) x_4}{l_p D(x_2)}$$

$$a_{43} = -\frac{3\left(R_m r_{mp}^2 B_{eq} + K_g^2 K_t K_m\right)\cos(x_2)}{R_m l_p D(x_2)}$$

$$a_{44} = -\frac{3\left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\right)B_p + 3M_p^2 l_p^2 r_{mp}^2 \cos(x_2)\sin(x_2) x_4}{M_p l_p^2 D(x_2)}$$

dhe $D(x_2) = 4Mr_{mp}^2 + M_p r_{mp}^2 + 4J_m K_g^2 + 3r_{mp}^2 M_p \sin^2(x_2)$. Matrica e varur nga gjendja, $B(X(t))$ në Ekuacionin (4.39) është dhënë nga:

$$B(X(t) = \begin{bmatrix} 0 \\ 0 \\ \frac{4r_{mp} K_g K_t}{R_m D(x_2)} \\ \frac{3r_{mp} K_g K_t \cos(x_2)}{l_p R_m D(x_2)} \end{bmatrix} \tag{4.41}$$

Tani, konsiderojmë funksionin e kostos:

$$J(X_0, u) = \int_0^\infty (X^T Q X + R u^2)\, dt \tag{4.42}$$

ku Q është një matricë me vlerë konstante 4×4 simetrike pozitive-gjysmë të përcaktuar dhe R është një skalar pozitiv. Në rastin e fillimit dhe balancimit të lavjerrësit të përmbysur në pozicionin e drejtë, problemi optimal i kontrollit është gjetja e një kontrolli të riveprimit të gjendjes $u^*(X)$ i cili minimizon koston (4.42) për gjendjen fillestare $X_0^T = [0, 0, 0, 0]$.
Kontrolli optimal i riveprimit për sistemin në (4.39) me kosto (4.42) ka formën:

$$u^*(X) = -\frac{1}{2} R^{-1} B^T(X) S_X(X) \tag{4.43}$$

ku funksioni S është zgjidhja e ekuacionit Hamilton-Jacobi-Bellman (HJB):

$$S_X^T(X) f(X) - \frac{1}{4} S_X^T(X) B(X) R^{-1} B^T(X) S_X(X) + X^T Q X = 0 \tag{4.44}$$

dhe S_X përfaqëson jakobianin e S në lidhje me gjendjet.

4.2.3. Skema e rregullimit

Sistemi i lavjerrësit u modelua në Simulink duke përdorur komponentët Simscape Electrical dhe Simscape Multi-body. Për këtë sistem:

- $x \in [-0.43\ 0.43]$ është pozicioni i karrocës dhe $\alpha \in [-2\pi\ 2\pi]$ është këndi i lavjerrësit.
- Pozicioni i karrocës x është 0 $[m]$ kur karroca është në qendër të pistës.
- Këndi i pendulumit α është 0 $[rad]$ kur pendulumi është i orientuar vertikalisht poshtë.
- Inputi i kontrollit $u_t = a_t \in [-10\ 10]$ $[V]$ është sinjali i tensionit DC i aplikuar motorit.

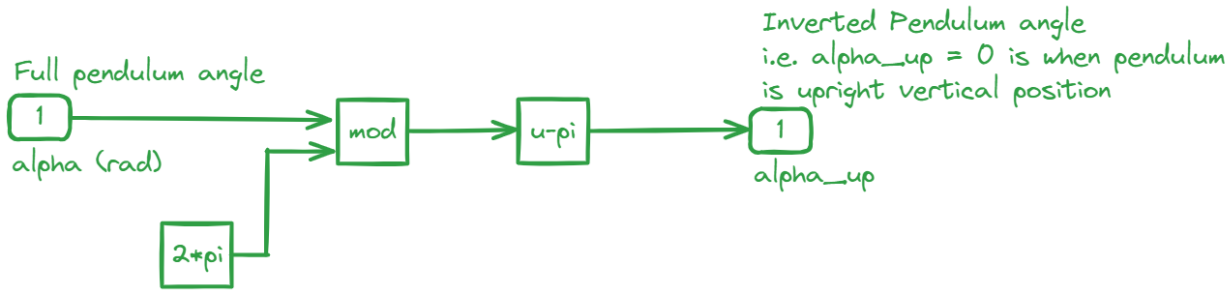


Figura 4.4 Konvertimi i sistemit referues të Lavjerrësit të Vetëm të Përmbysur

Kjo teknikë bën që pika e paqëndrueshme e ekuilibrit vertikal të jetë një vlerë e vetme ($\alpha = 0\ [rad]$) pa vendosur agjentin në një pozicion për të kuptuar nëse drejtimi i rrotullimit është kundërorës ($\phi = -\pi\ [rad]$) ose anës së orës ($\alpha = \pi\ [rad]$). Skema e përgjithshme e kontrollit është treguar në Figurën 4.5.

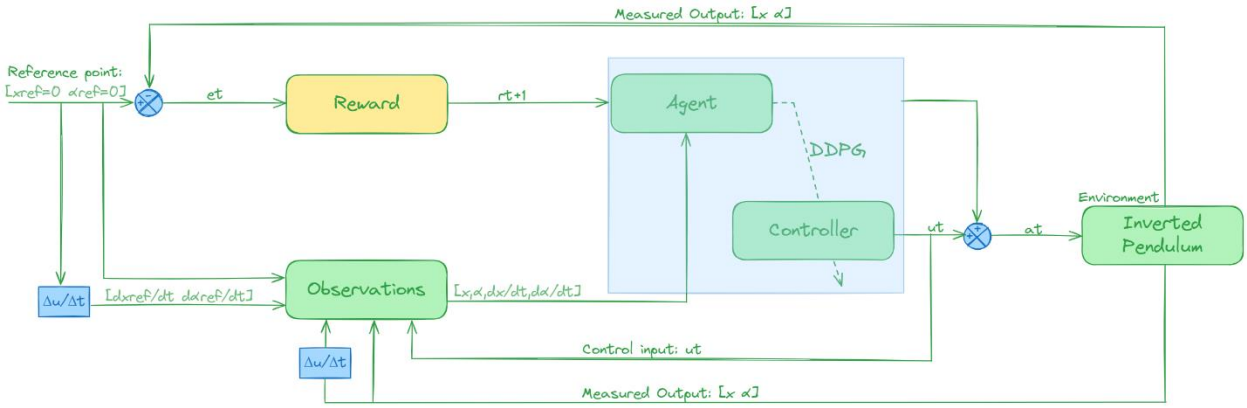


Figura 4.5 Skema e përgjithshme e kontrollit

Në modelin Simulink, u aplikua një ndryshim i sistemit referues në mënyrë që të kemi një vlerë këndi $0\ [rad]$ ($\alpha = 0\ [rad]$) në pozicionin e ekuilibrit vertikal për të trajtuar problemin si një problem kontrolli të rregullimit siç tregohet në skemën Simulink në Figurën 4.4.

4.2.4. Sinjalet e vëzhgimit dhe veprimit

Për sistemin e Lavjerrësit të Vetëm të Përmbysur Quanser Servo IP02, ekzistojnë katër sinjale të vazhdueshme të vëzhgimit: pozicioni i karrocës x, këndi i lavjerrësit α dhe normat e tyre të ndryshimit përkatësisht: $S \in [x, \alpha, \dot{x}, \dot{\alpha}]$. Megjithëse diapazoni i tensionit të aplikuar është ndërmjet $-12\ [V]$ dhe $12\ [V]$, seti i veprimit të vazhdueshëm është i kufizuar në $-10\ [V]$ deri në $10\ [V]$ për të përmirësuar qëndrueshmërinë në rastet e zbatimeve harduerike: $A \in [-10{,}10]\ [V]$.

Të dy gjendjet e vëzhgimit dhe veprimit janë normalizuar duke përdorur teknikën e normalizimit min-max siç përcaktohet në në ekuacionin 4.45 për të përmirësuar shpejtësinë dhe performancën e rrjeteve nervore gjatë trajnimit.

$$x_{k_{norm}} = \frac{x_{k_{meas}} - x_{min}}{x_{max} - x_{min}}(h - l) + l \tag{4.45}$$

ku:

- $x_{k_{meas}}$ janë të dhënat origjinale pa normalizim,
- $x_{k_{norm}}$ janë të dhënat e normalizuara,
- x_{max} dhe x_{min} janë respektivisht vlerat maksimale dhe minimale të sasisë që do të normalizohet,
- h dhe l janë respektivisht vlerat e sipërme dhe të poshtme të intervalit të ri për të dhënat e normalizuara.

4.2.5. Sinjali i shpërblimit

Sinjali i shpërblimit për sistemin tone u dizenjua si vijon:

$$R_{t+1} = -(Q_{11}x_t^2 + Q_{22}\alpha_t^2 + Q_{33}\dot{x}_t^2 + Q_{44}\dot{\alpha}_t^2 + R_{11}u_t^2) \quad (4.46)$$

Qëllimi i agjentit është të maksimizojë këtë funksion shpërblimi. Me fjalë të tjera, funksioni kuadratik i peshëzuar shpërblen agjentin kur pozicioni i karrocës x dhe këndi i lavjerrësit α janë afër pozicioneve të dëshiruara, zakonisht $x = 0$ dhe $\alpha = 0$. Termat që kanë të bëjnë me shpejtësitë ($\dot{x}$ dhe $\dot{\alpha}$) dhe përpjekjen e hyrjes së kontrollit (u) përfshihen për të ndëshkuar luhatjet e larta dhe përpjekjet e tepërta të kontrollit. Kjo ndihmon në ruajtjen e stabilitetit dhe siguron që tensioni i motorit të kontrollit të mos kalojë kufijtë e sigurt operacional.

Sinjalet e shpërblimit të bazuara në kuadrat janë veçanërisht të dobishme sepse janë më të lehta për t'u rregulluar dhe rrisin gjasat për të trajnuar një politikë të suksesshme me këtë sistem. Duke përzgjedhur në mënyrë të përshtatshme koeficientët e peshëzimit $(Q_{11}, Q_{22}, Q_{33}, Q_{44}, R)$, sistemi mund të drejtohet për të priorizuar operacionin e qetë dhe të qëndrueshëm, duke minimizuar lëvizjet e papritura dhe konsumimin e tepërt të energjisë.

4.2.6. Hiperparametrat dhe arkitekturat e rrjetit

Tabela më poshtë paraqet hiperparametrat e përdorur për trajnimin e agjentit pas disa iteracioneve të procesit të dizajnit të mësimit përforcues:

Hiperparametrat	
Peshat e funksionit të shpërblimit	Rasti 1: $Q_{11} = 0.75;\ Q_{22} = 4;\ Q_{33} = 0;\ Q_{44} = 0; R = 0.0003$ Rasti 2: $Q_{11} = 5;\ Q_{22} = 50;\ Q_{33} = 0;\ Q_{44} = 0; R = 0.002$ Rasti 3: $Q_{11} = 10;\ Q_{22} = 20;\ Q_{33} = 0;\ Q_{44} = 1; R = 0.01$ Rasti 4: $Q_{11} = 800;\ Q_{22} = 150;\ Q_{33} = 1;\ Q_{44} = 1; R = 0.1$
Rrjeti i aktorëve të lidhur plotësisht	**Arkitektura:** Shtresa hyrëse e veçorive: 4 neurone; Shtresat e Fshehura + Funksioni i Aktivizimit: 3 Shtresa të Fshehura me 200 neurone secila me Relu si funksion aktivizimi; Shtresa e daljes: 1 neuron që korrespondon me veprimin e zgjedhur me tanh si funksion aktivizimi për ta normalizuar atë midis -1 dhe 1 Learning rate: 1e-4;

	L2 Regularization Factor: 2e-4; Discount Factor: 0.995; Sample Time: 0.01; Mini Batch Size: 128; Experience Buffer Length: 1e6; Noise Variance: $1 * \frac{0.3}{\sqrt{T_s}}$ = 30% of Control Input Voltage
Rrjeti kritik i lidhur plotësisht	**Arkitektura:** Shtresa hyrëse e veçorive: 4 neurone (dimensioni i hapësirës së vëzhgimit); Shtresat e Fshehura + Funksioni i Aktivizimit: 4 Shtresa të Fshehura me 16 neurone secila dhe Relu që rrjedh me pjerrësi 0,5 si funksion aktivizimi. Zgjedhja e Relu që rrjedh është bërë për të marrë parasysh edhe vëzhgimet negative. Learning rate: 1e-4; L2 Regularization Factor: 1e-4; Discount Factor: 0.995; Sample Time: 0.01; Mini Batch Size: 128; Experience Buffer Lenght: 1e6;
Opsionet e trajnimit	Max Episodes: 10000; Max Steps Per Episode: 1000; Episode Duration: Sampling Time * Max Steps per Episode = 10 sec; Score Averaging Window Length: 5; Stop Training Criteria: Episode Reward; Stop Training Value: -10.

Tabela 4.2 Hiperparametrat dhe arkitekturat e rrjetit [33]

4.2.7. Performanca

Pasi një agjent i suksesshëm të jetë trajnuar, politika e tij për të balancuar saktësisht lavjerrësit testohet në simulim, në platformën Simulink duke përdorur bllokun Quanser Servo IP02. Trajnimi i një agjenti të mësimit me përforcim është një proces i gjatë që zgjat përafërsisht ndërmjet 25 min dhe 15 orë për lavjerrësin linear. Prandaj, ndryshe nga dizajni i bazuar në modele, me RL kërkohet kohë për të vërtetuar nëse agjenti mund të balancojë me sukses lavjerrësin dhe të vlerësojë përgjigjen e tij. Është e rëndësishme të dokumentohet se si ndryshimi i disa parametrave (p.sh., variablat e dizajnit) ndikon në përgjigjen e sistemit.
Përgjigjja e karrocës dhe lavjerrësit kur fillon afërsisht në pozicionin e ekuilibrit vertikal është treguar në grafikun e Lavjerrësit të Përmbysur (deg) në Fig. 4.6 dhe tensioni përkatës i aplikuar në motor është treguar në grafikun Tensioni në hyrje të motorit Vm (V) në Fig. 4.7.

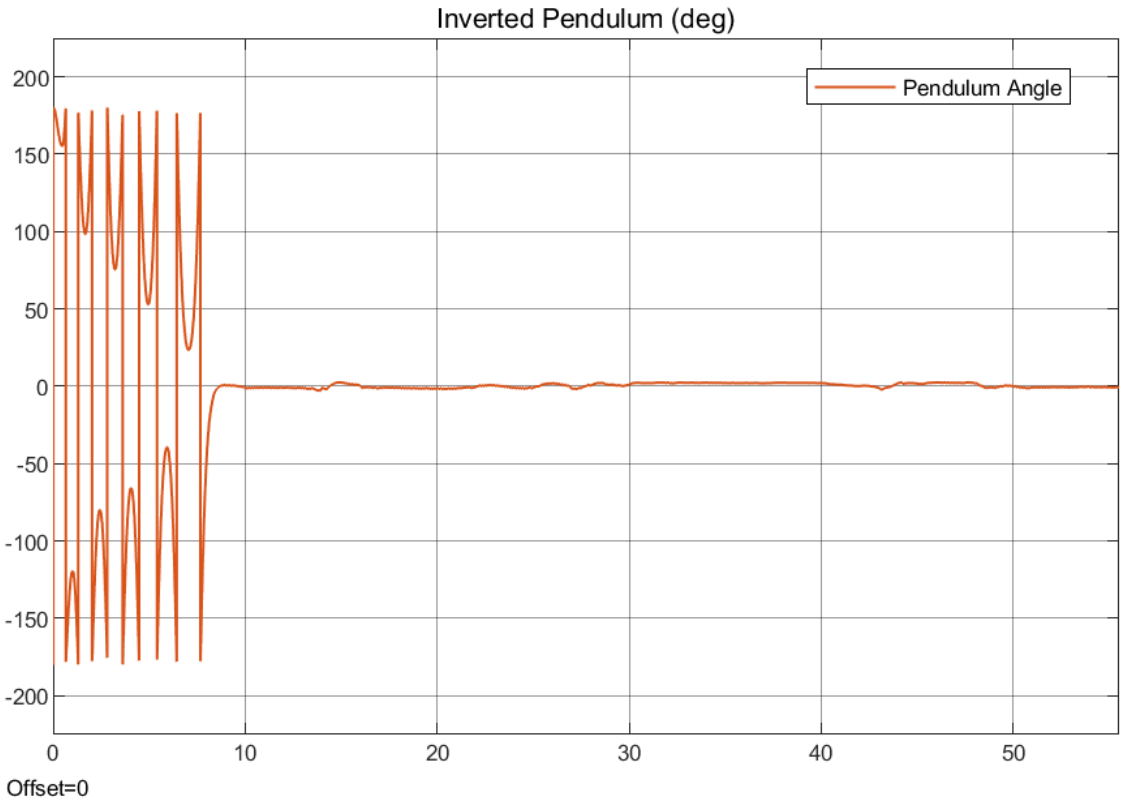


Figura 4.6 Këndi i Lavjerrësit të Përmbysur [deg]

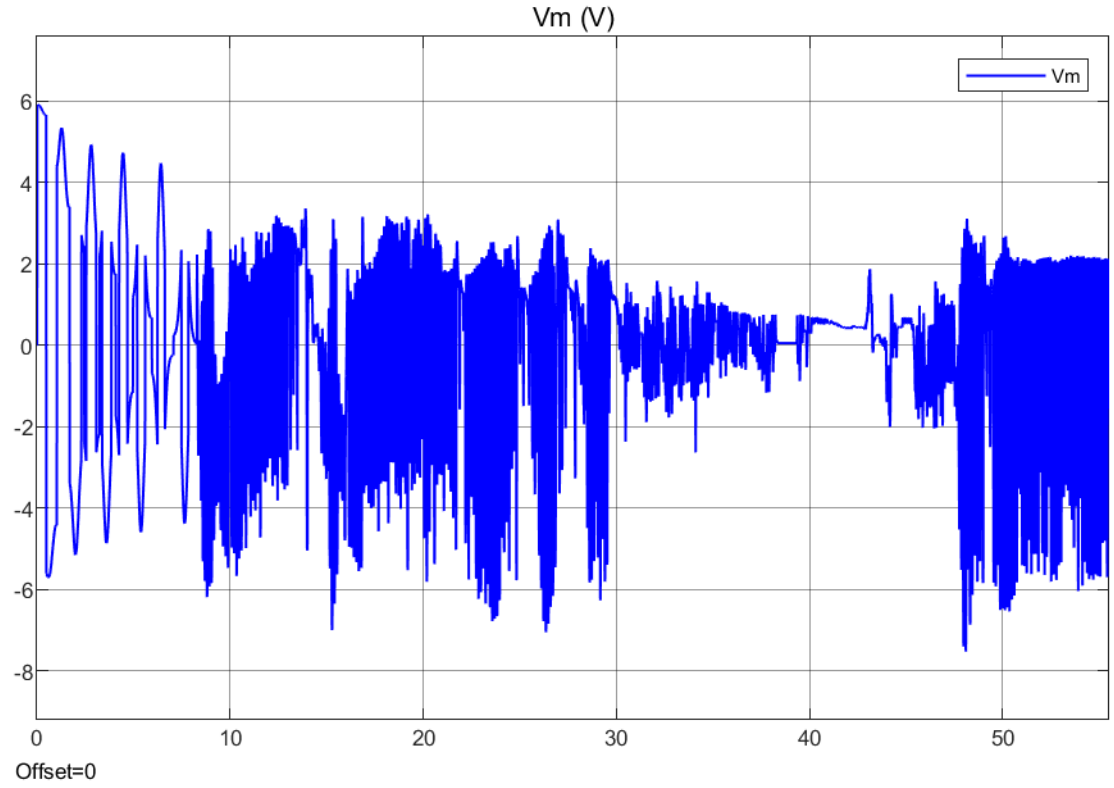


Figura 4.7 Tensioni në hyrje të motorit [V]

Siç mund të vërehet, lavjerrësi balancohet me një kohë vendosjeje $t_s = 2\ [sek]$. Kjo kohë mund të zvogëlohet duke zvogëluar numrin maksimal të hapave, gjë që do të zvogëlojë kohën e disponueshme për agjentin për t'u trajnuar në një episod të vetëm. Në këtë rast, numri maksimal i hapave për çdo episod u vendos në $s_{max} = 1000$, duke rezultuar në një kohëzgjatje episodi $t_{sim} = 10\ [sek]$:

$$t_{sim} = T_s * s_{max} \tag{4.47}$$

ku $T_s = 0.01$ është koha e kampionimit.

4.2.8. Përfundime

Ndërsa ka hapësirë për përmirësime të mëtejshme, ky studim tregon se mësimi përforcues (RL) mund të përdoret me sukses për detyra të avancuara të kontrollit në sisteme elektromekanike jo-lineare dhe mund të zbatohet në pajisje reale. Ekzistojnë shembuj të shumtë që ilustrojnë përdorimin e suksesshëm të RL për të balancuar dhe ngritur sistemet e lavjerrësit. Në këtë rast specifik, u përdor sistemi i Lavjerrësit të Vetëm të Përmbysur Quanser Servo IP02, ku pozicioni i karrocës x dhe këndi i lavjerrësit α ishin variablat kryesore. Qasja konfirmoi që RL jo vetëm që është e realizueshme, por edhe e dobishme për kontrollimin e këtyre sistemeve.

Potenciali i RL në aplikimet e sistemeve të kontrollit është demonstruar përmes aftësisë së tij për të menaxhuar dinamika komplekse dhe për t'u përshtatur me kushtet e ndryshme. Edhe pse RL për aplikimet e kontrollit është ende në zhvillim, trajektorja e tij e rritjes sugjeron që ajo do të bëhet një teknikë më e fuqishme dhe e lehtë për t'u përdorur në të ardhmen e afërt. Me përparimet që vazhdojnë, pritet që RL të ofrojë zgjidhje edhe më të sofistikuara për sistemet e kontrollit, duke e bërë atë një mjet thelbësor në fushën e mekatronikës.

Në përmbledhje, kjo punë me Lavjerrësin e Vetëm të Përmbysur Quanser Servo IP02 konfirmon realizueshmërinë dhe efektivitetin e RL për detyra të sofistikuara të kontrollit, duke vendosur një themel për hulumtime dhe zhvillime të mëtejshme në këtë fushë.

4.3. Diskutim mbi trajnimin RL

Trajnimi i agjentëve të mësimit përforcues (RL) për Lavjerrësin e Vetëm të Përmbysur Quanser Servo IP02 përfshin disa konsiderata kritike për të siguruar zhvillimin e suksesshëm të politikave dhe kontrollin e sistemit.

- **Vëzhgimi i Sjelljes së Agjentit**: Gjatë trajnimit, është thelbësore të monitorohet sjellja e agjentit duke përdorur skopet ose blloqet e vizualizimit brenda modelit Simulink. Kjo ndihmon në ndjekjen e evolucionit të politikave dhe shpërblimeve të episodeve. Pyetjet kryesore përfshijnë nëse agjenti është bllokuar në një politikë suboptimale ose po shfrytëzon mekanizmin e shpërblimit në mënyra të paqëllimta.
- **Kohëzgjatja e Trajnimit**: Trajnimi i RL është një proces që kërkon kohë, me agjentët që përjetojnë nivele të ndryshme të performancës ndërsa eksplorojnë politika të ndryshme. Nëse një agjent nuk ka arritur ende një performancë të kënaqshme por vazhdon të tregojë progres në të mësuar, zgjatni kohëzgjatjen e trajnimit duke rritur numrin e episodeve.
- **Rëndësia e Eksplorimit**: Eksplorimi i mjaftueshëm është thelbësor për të parandaluar agjentin që të vendoset në politika suboptimale. Eksperimentoni me parametrat e eksplorimit nëse agjenti ndalon së përmirësuari, duke e inkurajuar të eksplorojë mjedisin më efektivisht.

- **Dizajni i Funksionit të Shpërblimit**: Funksioni i shpërblimit drejton të mësuarin e agjentit. Sigurohuni që funksioni i shpërblimit të mos përmbajë boshllëqe që agjenti mund të shfrytëzojë për të fituar shpërblime të pamerituara. Formoni shpërblimin për të udhëhequr agjentin drejt arritjes së sjelljeve të dëshiruara. Shpërblimet e rralla, të cilat japin vetëm reagime kur detyrat përfundojnë me sukses, mund të pengojnë të mësuarin nëse agjenti rrallë i has këto shpërblime përmes eksplorimit të rastësishëm.
- **Rregullimet e Shkallës së Mësimit**: Shkalla e mësimit ndikon ndjeshëm në stabilitetin dhe kohëzgjatjen e trajnimit. Ndërsa një shkallë e ulët e mësimit mund të ngadalësojë përparimin, një shkallë e lartë mund të shkaktojë paqëndrueshmëri. Filloni me një shkallë më të lartë të mësimit por zvogëloni atë nëse politika e agjentit luhatet në mënyrë të rastësishme pa përmirësuar shpërblimin mesatar.
- **Kompleksiteti i Rrjetit Neural**: Kompleksiteti i rrjetit neural, i përcaktuar nga numri i neuroneve, ndikon në kohën dhe performancën e trajnimit. Filloni me një arkitekturë të thjeshtë të rrjetit, si konfigurimi i paracaktuar ose një nga një shembull i ngjashëm. Nëse të mësuarit ngec, konsideroni rritjen e numrit të neuroneve të shtresës së fshehur.
- **Normalizimi**: Zbatimi i normalizimit min-max në veprimet dhe gjendjet e vëzhgimit siguron që hyrjet në rrjetin neural janë brenda një shkalle të qëndrueshme, duke përmirësuar efikasitetin dhe performancën e trajnimit.

Për më tepër, efikasiteti i mbledhjes së të dhënave gjatë fazës së mësimit ndikon në shkallën e konvergjencës drejt politikës optimale dhe kohëzgjatjen e përgjithshme të trajnimit. Përmirësimi i efikasitetit të mostrave mund të përshpejtojë ndjeshëm procesin e të mësuarit dhe të përmirësojë performancën e agjentit [32] [33].

5. Kontrolli i Stabilizimit

5.1. Zbatimi në Kohë Reale

5.1.1. Aparatura

Ky kapitull diskuton aplikimin e teknikave të mësimit me përforcim (RL) për të kontrolluar një sistem lavjerrësi të përmbysur linear, veçanërisht sistemin e Lavjerrësit të Vetëm të Përmbysur (SIP) Quanser Servo IP02. Qëllimi është të stabilizohet lavjerrësi i përmbysur në kohë reale duke përdorur algoritmin Deep Deterministic Policy Gradient (DDPG) brenda mjedisit MATLAB/Simulink.

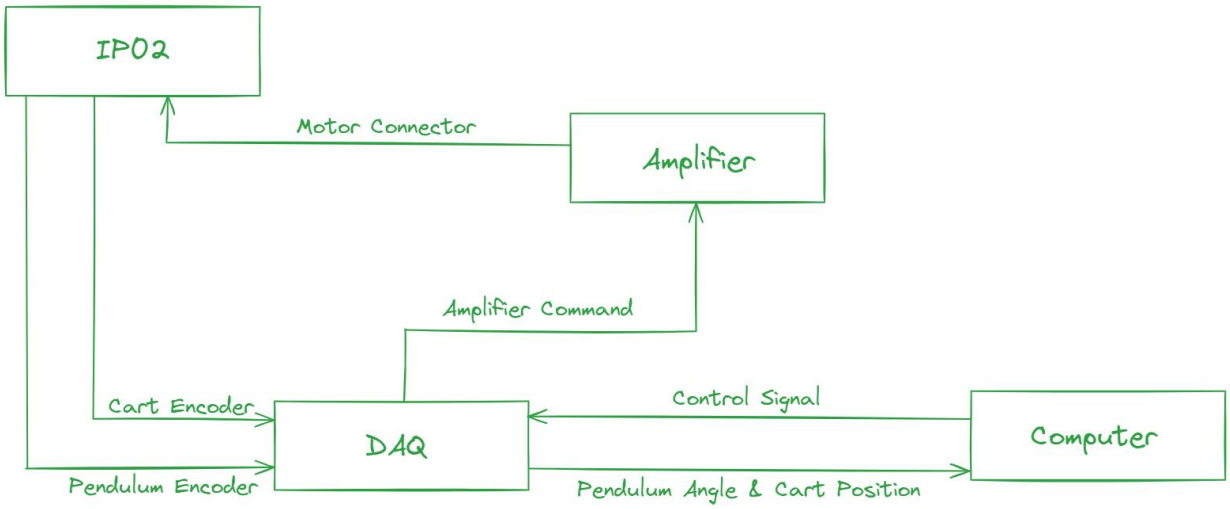


Figura 5.1 Diagrami i vendosjes eksperimentale

Për eksperimentet tona në kohë reale, ne përdorim sistemin Quanser Servo IP02, i cili përfshin komponentët e mëposhtëm:

- Lavjerrësi i Vetëm i Përmbysur (SIP) i montuar në një sistem servo IP02
- Përforcuesi VoltPAQ
- Bordi i kontrollit të përpunimit të të dhënave (DAQ) Q2-USB

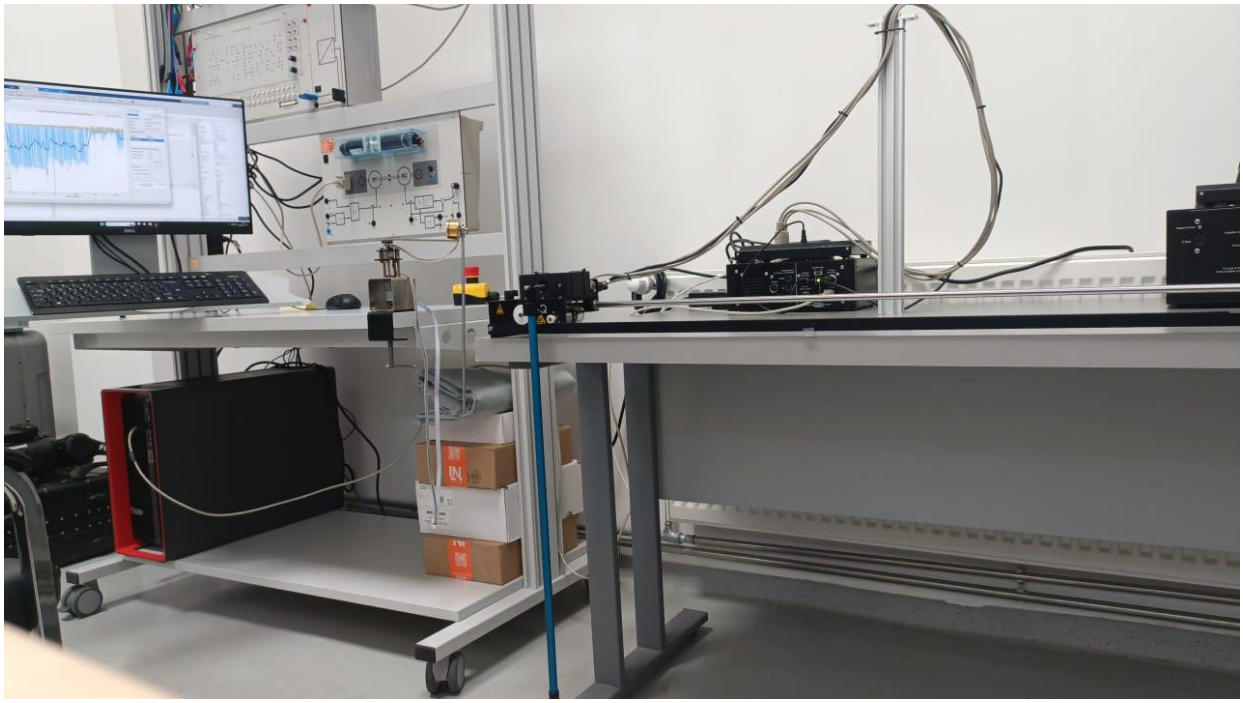


Figura 5.2 Lavjerrësi i vetëm i përmbysur i montuar në një sistem servo Quanser IP02

Karroca IP02 drejtohet nga një Motor DC Faulhaber Coreless (2338S006) i lidhur me një Faulhaber Planetary Gearhead Seria 23/1 dhe është e pajisur me një kodifikues optik boshtor të vetëm nga US Digital S1. Një diagram i detajuar i konfigurimit eksperimental është treguar në Figurën 5.2.

5.1.2. Specifikimet e Dizajnit

Qëllimi i eksperimentit tonë në kohë reale është të stabilizojmë lavjerrësin e përmbysur në pozicionin e drejtë me lëvizje minimale të karrocës dhe përpjekje të kontrollit. Peshat $Q \geq 0$ dhe $R > 0$ në funksionalin e kostos (4.42) duhet të zgjidhen në mënyrë që sistemi të përmbushë kërkesat e mëposhtme të performancës së projektimit të specifikuara:

1. Rregullohet këndi i lavjerrësit rreth pozicionit të drejtë dhe të mos tejkalohet një devijim prej ±1 shkallë nga ai, dmth. $|\alpha| \leq 1.0°$.
2. Minimizohet përpjekja e kontrollit të prodhuar, e cila është proporcionale me tensionin e hyrjes së motorit V_m. Amplifikatori i fuqisë nuk duhet të shkojë në ngopje në asnjë rast, dmth. $|V_m| \leq 10V$.

5.2. Rezultatet eksperimentale

Ky punim tregon se si të dizajnohet një agjent përforcues duke përdorur MathWorks Reinforcement Learning Toolbox për të balancuar sistemin Quanser Servo IP02 Single Inverted Pendulum, i treguar në Figurën 5.2. Ky sistem ka një enkoder për të matur pozicionin e karrocës (d.m.th., pozicionin e karrocës) dhe lidhjen e lavjerrësit, dhe një motor DC që drejton karrocën përgjatë një shine lineare.

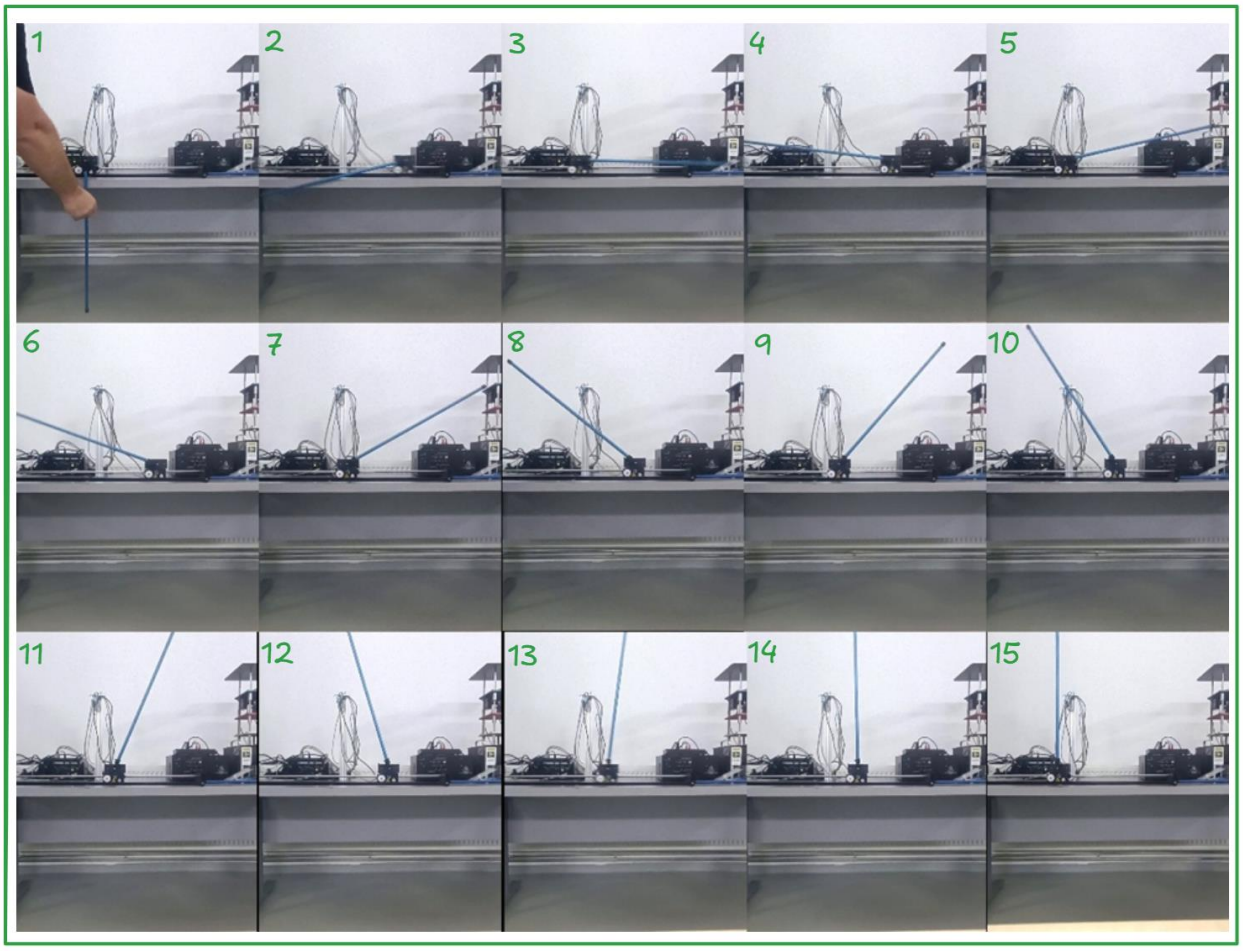


Figura 5.3 Sekuencia e Ngritjës dhe Stabilizimit të Lavjerrësit të Vetëm të Përmbysur (SIP) Quanser IP02

Figura 5.3 ilustron lëvizjet e Lavjerrësit të Vetëm të Përmbysur (SIP). Në këtë sekuencë, kornizat 1 deri në 3 paraqesin lëvizjen me rritje të amplitudës së lavjerrësit, kornizat 4 deri në 7 demonstrojnë gjendjen e ngritjes (swing-up), ndërsa kornizat 8 deri në 11 tregojnë teknikën e ngritjes përmes inercisë. Së fundi, kornizat 12 deri në 15 përshkruajnë procesin e stabilizimit. Ky seri imazhesh kap lëvizjet dinamike të SIP, nga lëvizja fillestare deri në stabilizimin e suksesshëm.

5.2.1. Matricat e Peshës së Qëndrueshme

Zgjedhja e matricave të peshës ka një ndikim të madh në performancën e kontrolluesit. Për të ndëshkuar fort pozicionet jo-zero, pesha e gjendjes Q duhet të zgjidhet me peshë të madhe në pozicionet dhe peshë të vogël në shpejtësitë. Vlera e R duhet të jetë mjaft e madhe për të siguruar që amplifikatori i fuqisë të mos hyjë në ngopje dhe për të parandaluar lëvizjen e tepërt të karrocës. Megjithatë, nëse është shumë e madhe, atëherë gjendjet mund të devijojnë shumë nga pozicioni zero.

Ne kemi testuar disa çifte të ndryshme të matricave të mundshme të peshës. Për të gjetur një kombinim të mirë të vlerave për Q dhe R, ne përdorim procedurën e mëposhtme të akordimit bazuar në procedurën e përshkruar nga Quanser:

1. Kryejm një simulim me një zgjedhje të veçantë për Q dhe R duke përdorur diagramet Simulink të ofruara. Studiojm rezultatet e gjendjes dhe përpjekjen e kërkuar për kontroll. Nëse rezultatet e gjendjes dhe përpjekjet e kontrollit janë brenda diapazonit të dëshiruar, atëherë kalojm në hapin tjetër. Përndryshe, rregullojmë vlerat në Q dhe R, dhe kryejm simulimin përsëri. Për të rregulluar vlerat, marrim parasysh si vijon:
    - Nëse pozicioni i karrocës devijon shumë nga qendra, atëherë përpiqemi të rrisim Q_{11} dhe/ose të ulim Q_{22}.
    - Nëse këndi i lavjerrësit devijon shumë nga pozicioni vertikal, atëherë përpiqemi të rrisim Q_{22} dhe/ose të ulim Q_{11}.
    - Nëse tensioni i hyrjes së motorit shkon në ngopje, përpiqemi të rrisim R dhe/ose të ulim Q_{11} së bashku me Q_{22}.
2. Nëse rezultatet e simulimit janë të kënaqshme, atëherë testojmë matricat Q dhe R në kohë reale duke përdorur diagramet Simulink të ofruara. Rregullojmë vlerat e Q dhe R deri sa përgjigjet e gjendjes dhe përpjekjet e kërkuara për kontroll të jenë të kënaqshme. Gjatë rregullimit të vlerave, përdorim konsideratat nga hapi i mëparshëm. Nëse karroca është shumë "hiperaktive" dhe vibrimet janë të tepruara, atëherë përpiqemi të rrisim R dhe/ose të ulim Q_{11} së bashku me Q_{22}.

Katër çifte të veçanta të matricave $Q = diag(Q_{11}, Q_{22}, Q_{33}, Q_{44})$ dhe R u zgjodhën duke përdorur procedurën e mësipërme, megjithatë, brenda këtyre konsideratave, këto zgjedhje janë disi arbitrare. Zgjedhja e parë është $Q = diag(0.75, 4, 0, 0)$ dhe $R = 0.0003$. Vini re që për gjendjet vetëm pozicioni jo-zero i karrocës dhe këndi i lavjerrësit janë ndëshkuar, dhe nuk ka peshë në shpejtësitë. Rezultatet e gjendjes dhe përpjekjet e kërkuara për kontroll janë paraqitur në Figura 5.4-5.7. Zgjedhja e dytë për këto matricat, $Q = diag(5, 50, 0, 0)$ dhe $R = 0.002$, rezulton në përmirësim të performancës. Rezultatet përkatëse të gjendjes dhe përpjekjet e kërkuara për kontroll janë paraqitur në Figura 5.8-5.11. Zgjedhja e tretë për këto matricat, $Q = diag(10, 20, 0, 1)$ dhe $R = 0.01$, rezulton në përmirësim të performancës. Rezultatet përkatëse të gjendjes dhe përpjekjet e kërkuara për kontroll janë paraqitur në Figura 5.12-5.15. Zgjedhja e katërt, $Q = diag(800, 150, 1, 1)$ dhe $R = 0.1$, gjithashtu përfshin ndëshkime në shpejtësitë, megjithatë, rezulton në performancë më të dobët. Rezultatet përkatëse të gjendjes dhe përpjekjet e kërkuara për kontroll janë paraqitur në Figura 5.16-5.19.

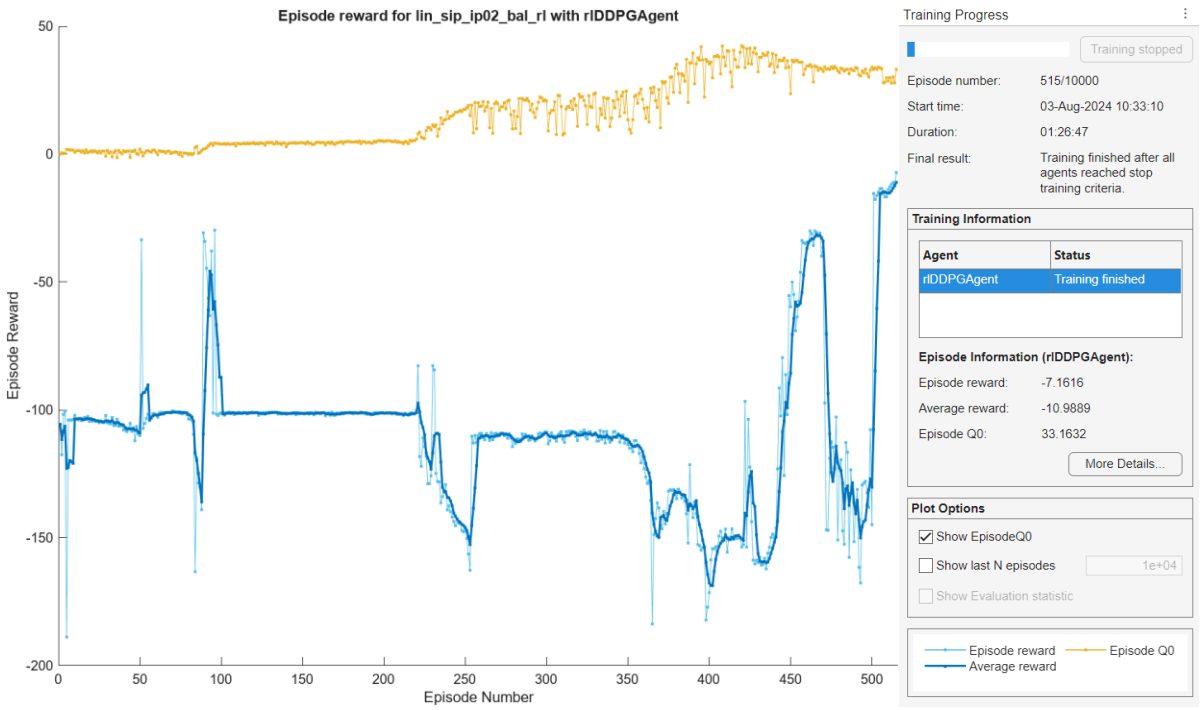


Figura 5.4 Përparimi i trajnimit për peshat Q=diag(0.75,4,0,0), R=0.0003

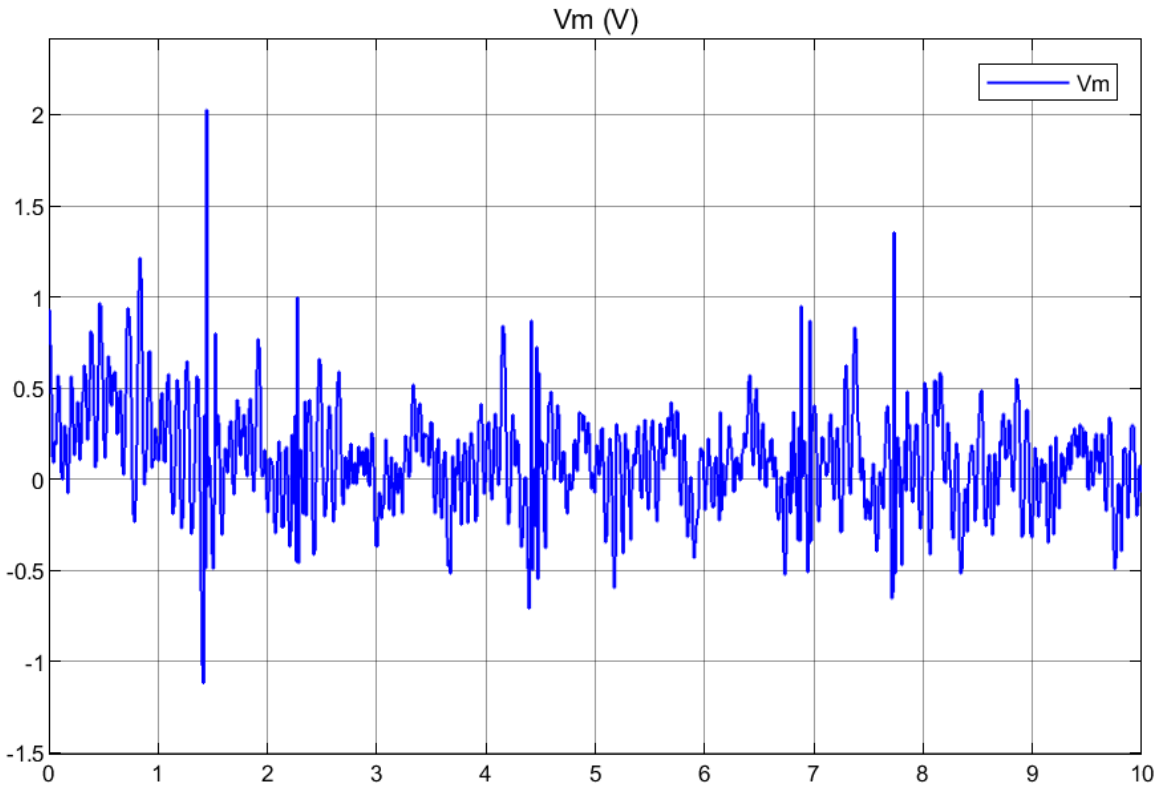


Figura 5.5 Tensioni në hyrje të motorit për peshat Q=diag(0.75,4,0,0), R=0.0003

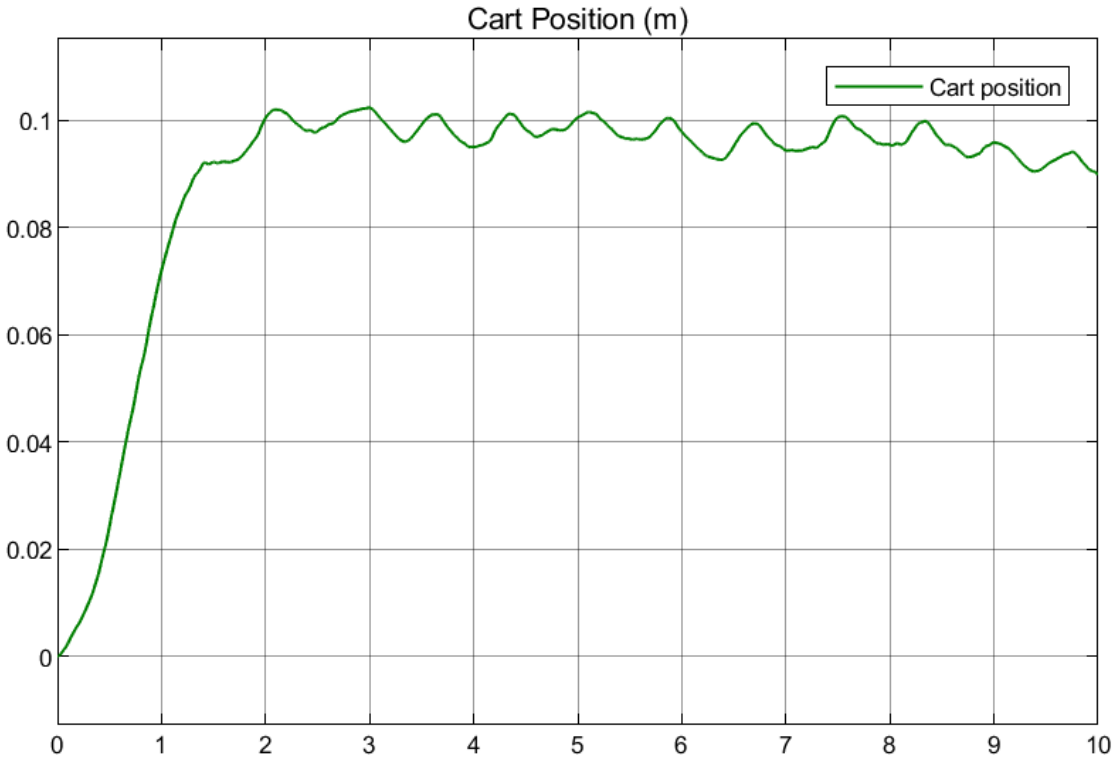


Figura 5.6 Pozicioni i karrocës (m) për peshat Q=diag(0.75,4,0,0), R=0.0003

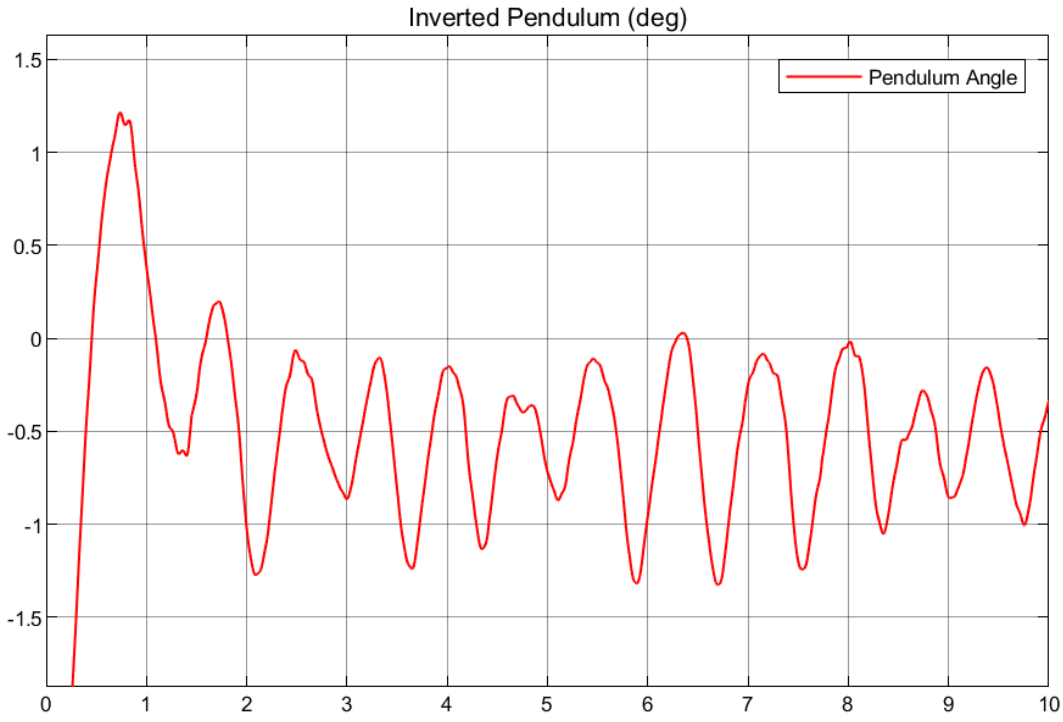


Figura 5.7 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(0.75,4,0,0), R=0.0003

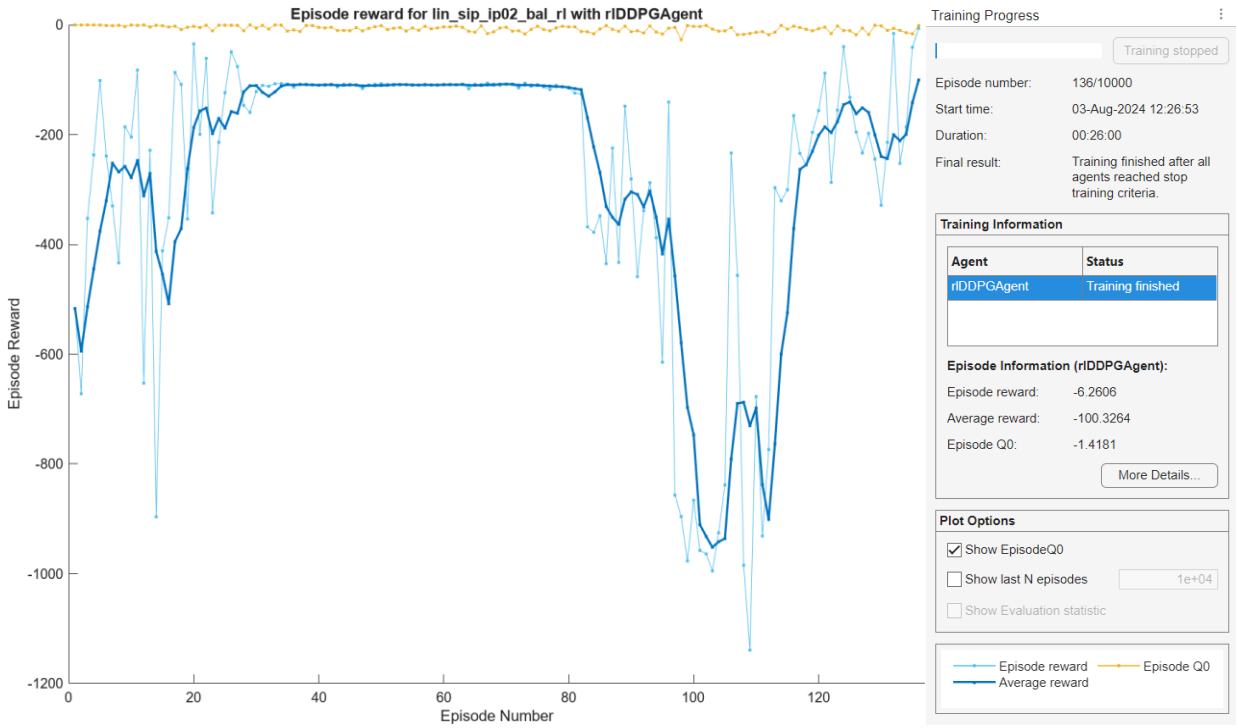


Figura 5.8 Përparimi i trajnimit për peshat Q=diag(5,50,0,0), R=0.002

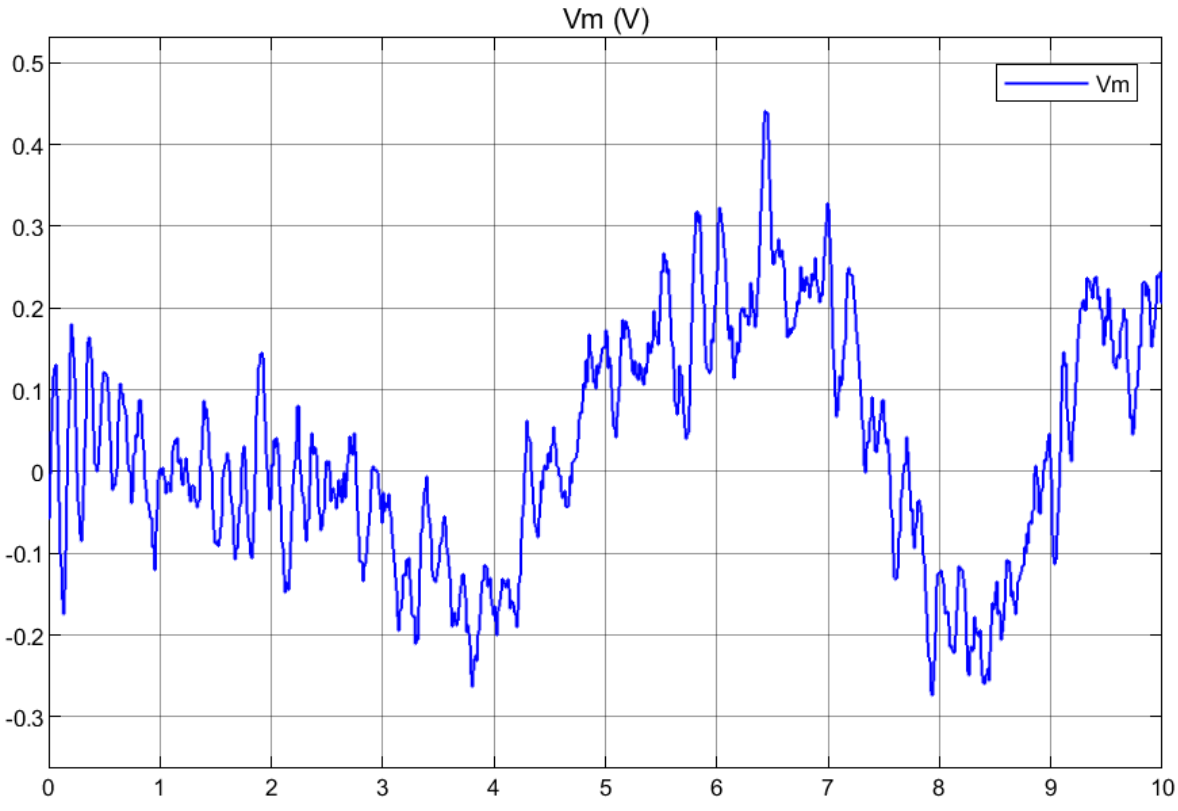


Figura 5.9 Tensioni në hyrje të motorit për peshat Q=diag(5,50,0,0), R=0.002

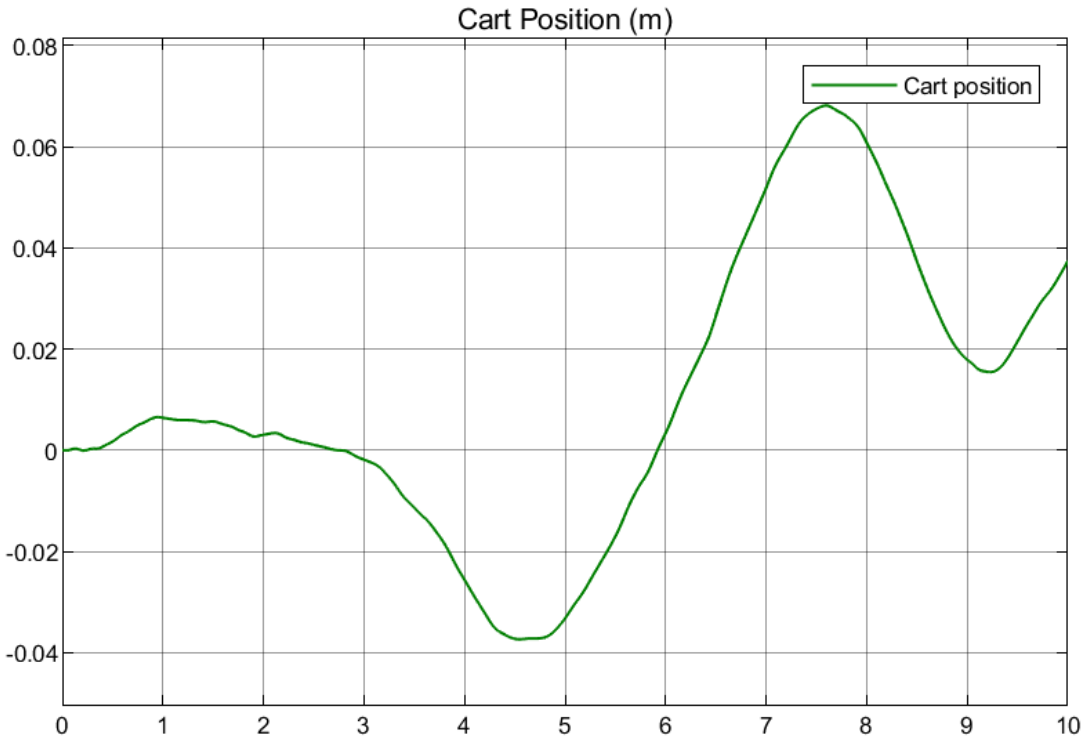


Figura 5.10 Pozicioni i karrocës (m) për peshat Q=diag(5,50,0,0), R=0.002

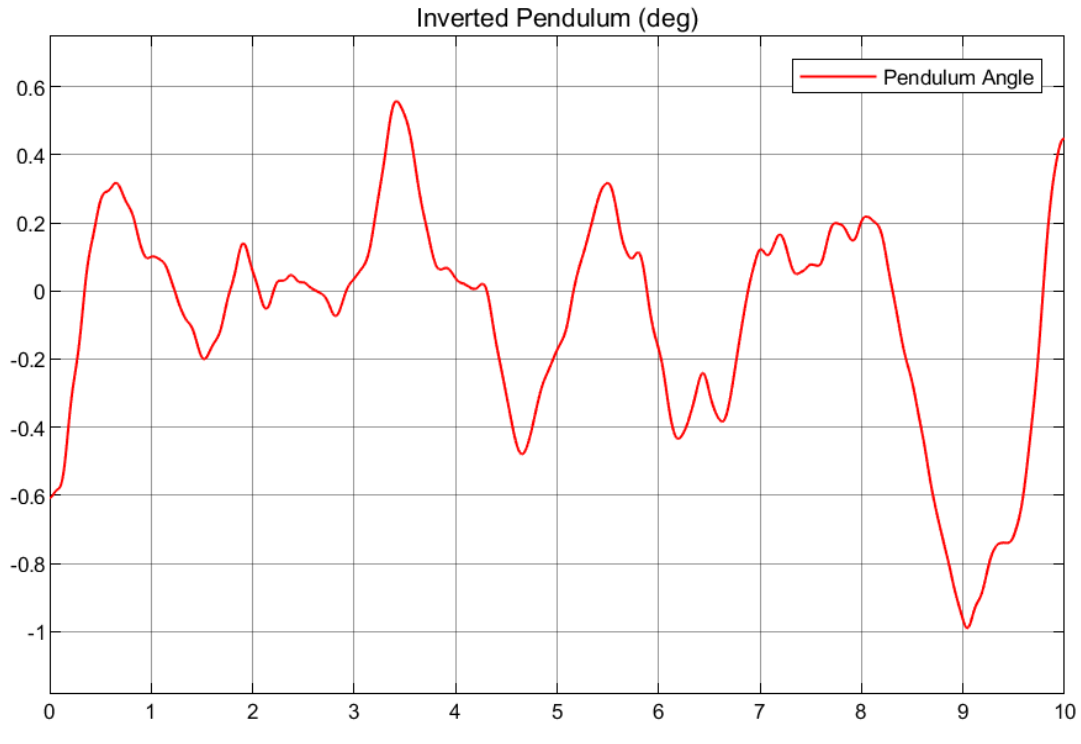


Figura 5.11 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(5,50,0,0), R=0.002

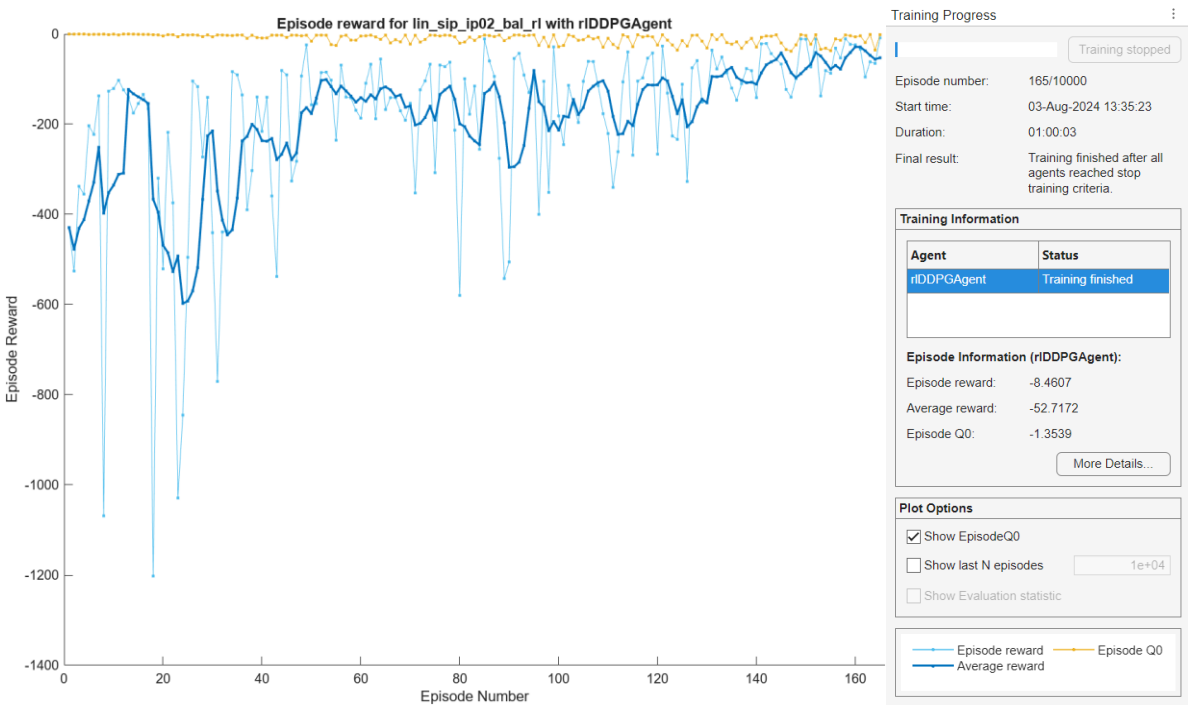


Figura 5.12 Përparimi i trajnimit për peshat Q=diag(10,20,0,1), R=0.01

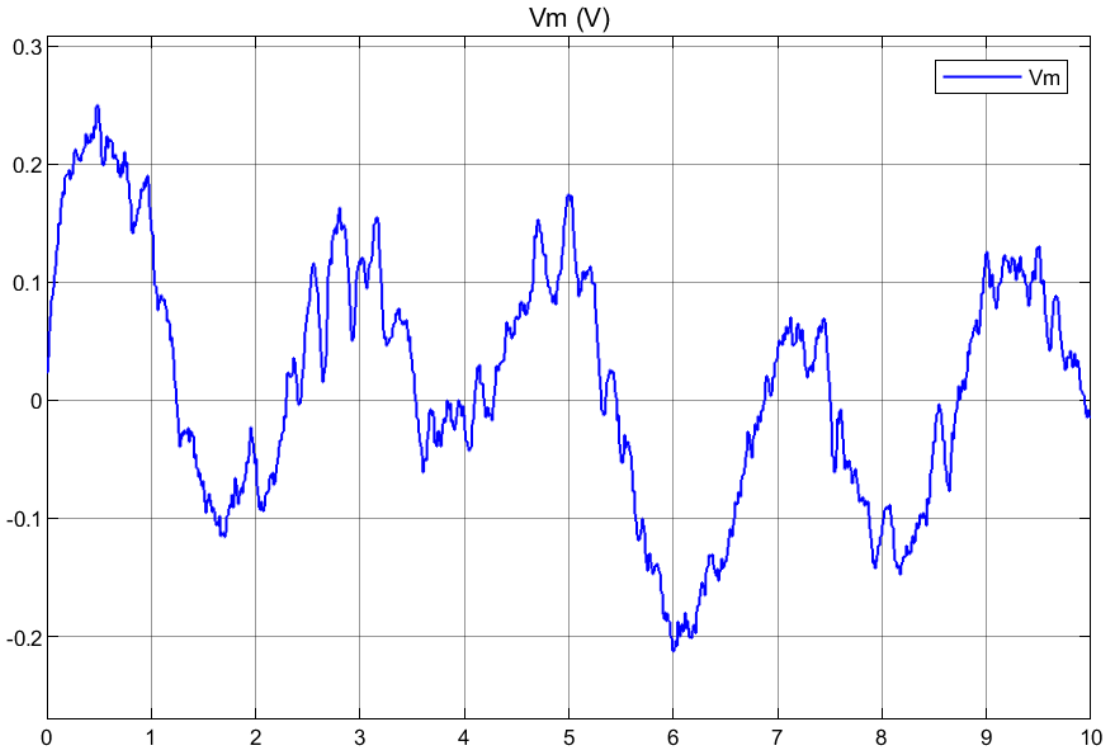


Figura 5.13 Tensioni në hyrje të motorit për peshat Q=diag(10,20,0,1), R=0.01

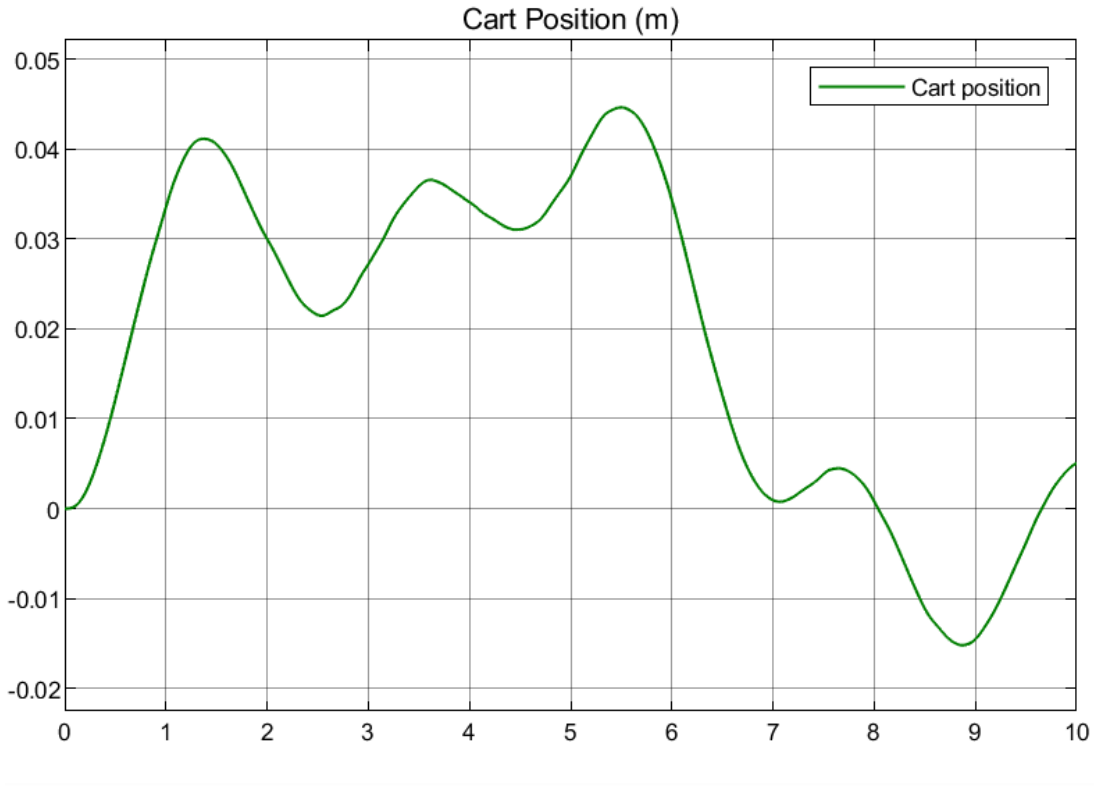


Figura 5.14 Pozicioni i karrocës (m) për peshat Q=diag(10,20,0,1), R=0.01

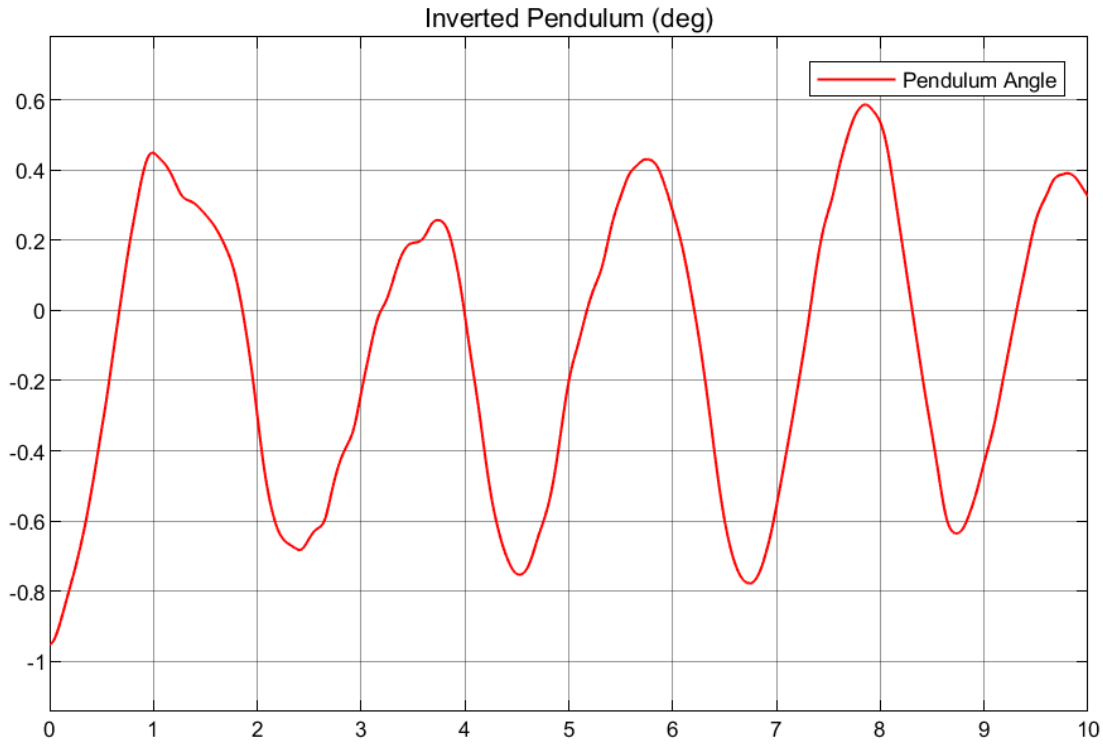


Figura 5.15 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(10,20,0,1), R=0.01

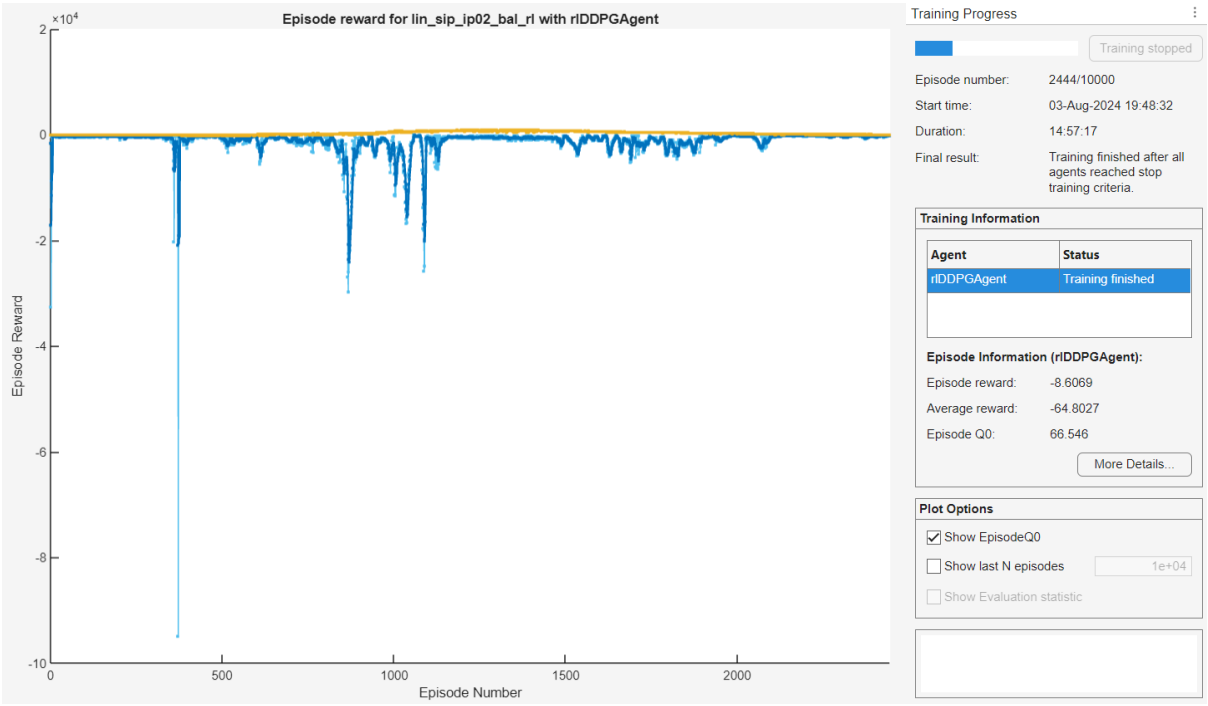


Figura 5.16 Përparimi i trajnimit për peshat Q=diag(800,150,1,1), R=0.1

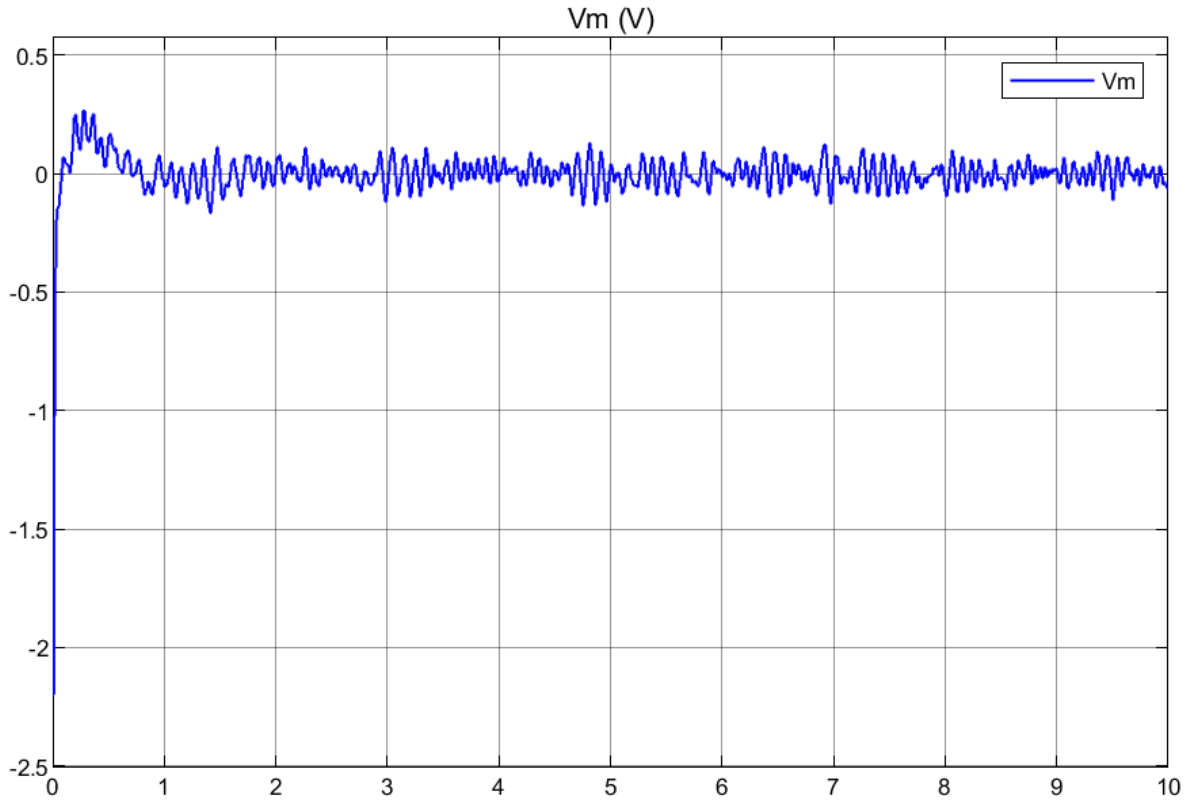


Figura 5.17 Tensioni në hyrje të motorit për peshat Q=diag(800,150,1,1), R=0.1

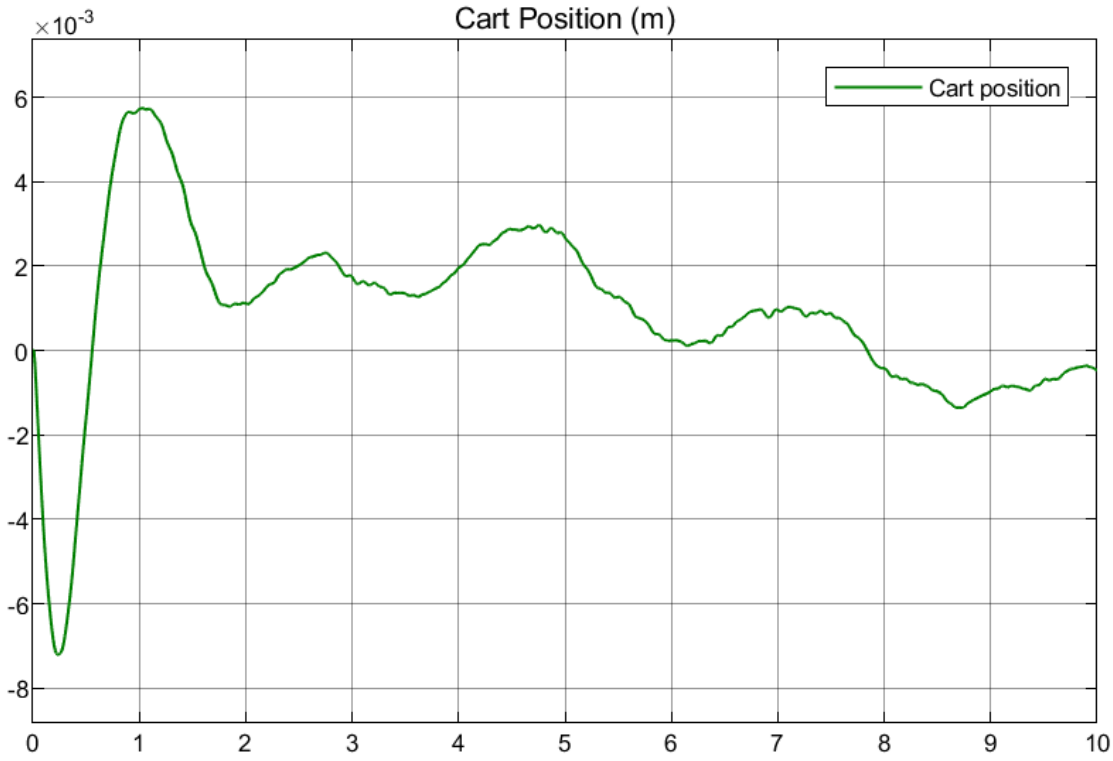


Figura 5.18 Pozicioni i karrocës (m) për peshat Q=diag(800,150,1,1), R=0.1

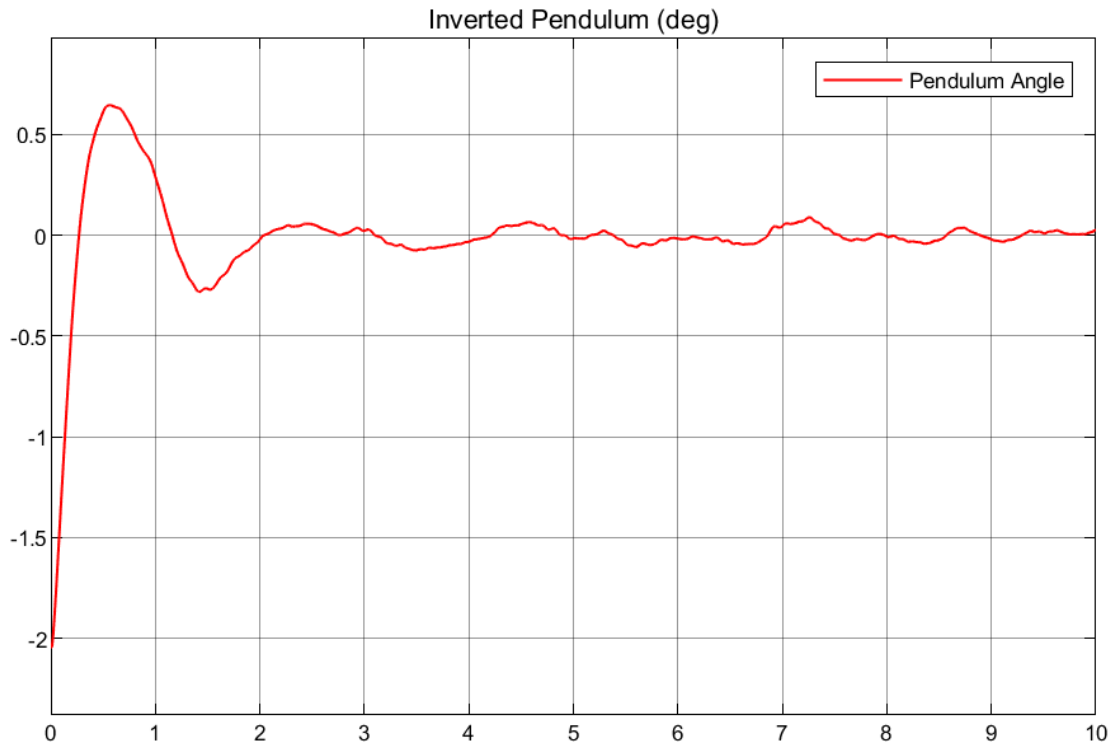


Figura 5.19 Këndi i lavjerrësit të përmbysur (deg) për peshat Q=diag(800,150,1,1), R=0.1

6. Përfundimet

Kjo tezë ka analizuar dhe përmirësuar në mënyrë të thellë algoritmin Deep Deterministic Policy Gradient (DDPG), duke filluar nga baza e tij në procesin e vendimmarrjes Markov dhe kornizën Aktor-Kritik. Duke integruar algoritmin DQN, u përftua procesi i përditësimit të gradientit të DDPG-së, dhe u propozua një metodë e përmirësuar për të adresuar problemin e mbivlerësimit të Q-vlerës. Efektiviteti i algoritmit të përmirësuar DDPG u verifikua përmes zhvillimit dhe testimit të një sistemi kontrolli për lavjerrësin e vetëm të përmbysur (SIP), fillimisht në një mjedis simulimi dhe më pas në një ambient laboratorik.

Algoritmi i përmirësuar DDPG tregoi performancë superiore krahasuar me kontrolluesit tradicionalë, duke ofruar kohë reagimi më të shpejtë, stabilitet më të madh dhe konvergjencë të përmirësuar. Duke modifikuar strukturën e bazës së përvojës dhe rrjetin Kritik, algoritmi shmangu në mënyrë efektive optimat lokale dhe arriti një trajnim më efikas. Algoritmi i përmirësuar tregoi gjithashtu aftësi të forta kundër ndërhyrjeve, duke e bërë atë një zgjidhje të vlefshme për problemet komplekse të kontrollit.

Megjithatë, ndonëse algoritmi DDPG tregon premtime të mëdha, ka sfida të natyrshme që lidhen me qasjet e thella të mësimit përforcues (DRL), veçanërisht në aplikimet reale. Çështje si paefikasiteti i mostrave, qëndrueshmëria dhe stabiliteti mbeten shqetësime të rëndësishme. Megjithëse metodat DRL kanë arritur rezultate mbresëlënëse në mjediset e kontrolluara, aplikimi i tyre praktik në sistemet fizike kërkon rafinim të mëtejshëm, veçanërisht për të siguruar siguri dhe besueshmëri. Punimet e ardhshme duhet të fokusohen në rregullimin e hipërparametrave, përmirësimin e qëndrueshmërisë së algoritmeve dhe eksplorimin e integrimit të DRL me metodat klasike të kontrollit për të zhvilluar sisteme kontrolli më të besueshme dhe efikase.

Në përfundim, kjo tezë demonstron potencialin e kombinimit të mësimit përforcues me teorinë tradicionale të kontrollit për të adresuar problemet komplekse të kontrollit. Algoritmi i përmirësuar DDPG ofron një drejtim premtues për hulumtime të mëtejshme, veçanërisht në përmirësimin e qëndrueshmërisë dhe stabilitetit të tij për aplikime reale. Ndërsa qasjet tërësisht të bazuara në të dhëna si DRL mund të mos jenë ende zgjidhja kryesore për të gjitha skenarët reale, ato kanë arritur suksese të jashtëzakonshme dhe përfaqësojnë një fushë emocionuese të kërkimeve në vazhdim.

7. Conclusions

This thesis has carefully studied and improved the Deep Deterministic Policy Gradient (DDPG) algorithm, starting from its basic principles in the Markov decision process and the Actor-Critic framework. By adding the DQN algorithm, the update process for DDPG was adjusted, and a better method was suggested to solve the problem of Q-value overestimation. The effectiveness of the improved DDPG algorithm was tested by developing and evaluating a control system for the single inverted pendulum (SIP), first in a simulated environment and then in a lab setting.

The improved DDPG algorithm showed better performance compared to traditional controllers, with faster response times, more stability, and better convergence. By changing the experience replay structure and the Critic network, the algorithm effectively avoided being stuck in local optima and achieved training that is more efficient. The improved algorithm also proved to be strong against disturbances, making it a valuable solution for complex control problems.

However, even though the DDPG algorithm is very promising, there are challenges related to deep reinforcement learning (DRL) methods, especially when used in real-world situations. Issues like sample inefficiency, consistency, and stability are still important concerns. Although DRL methods have achieved great results in controlled environments, applying them in real physical systems needs more refinement, especially to ensure safety and reliability. Future work should focus on tuning hyperparameters, making the algorithms more robust, and exploring how to combine DRL with traditional control methods to create more reliable and efficient control systems.

In conclusion, this thesis shows the potential of combining reinforcement learning with traditional control theory to solve complex control problems. The improved DDPG algorithm offers a promising direction for further research, especially in improving its robustness and stability for real-world applications. While fully data-driven approaches like DRL may not yet be the main solution for all real-world scenarios, they have achieved significant successes and represent an exciting area of ongoing research.

8. Referencat

[1] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. 1st ed. Boca Raton: CRC Press, 2010. eBook Published January 31, 2017. ISBN: 9781439821091 (Print), 9781315217932 (eBook). doi: 10.1201/9781439821091. Taylor & Francis (visited on 07/29/2024).

[2] Jannes Quer, Enric Ribera Borrell, (2022). "Connecting Stochastic Optimal Control and Reinforcement Learning". https://doi.org/10.48550/arXiv.2211.02474. ArXiv (visited on 07/29/2024).

[3] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N.M., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *CoRR, abs/1509.02971*. ArXiv (visited on 07/29/2024).

[4] Mnih, V., Kavukcuoglu, K., Silver, D. et al. "Human-level control through deep reinforcement learning". en. In: Nature 518.7540 (Feb. 2015), pp. 529–533. issn: 0028-0836, 1476-4687. doi: 10.1038/nature14236. nature (visited on 07/29/2024).

[5] Stuart J. Russell, Peter Norvig, and Ernest Davis. Artificial intelligence: a modern approach. 3rd ed. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010. Isbn-13: 978-0-13-604259-4.

[6] Silver, D., Lever, G., Heess, N.M., Degris, T., Wierstra, D., & Riedmiller, M.A. et al.. Deterministic Policy Gradient Algorithms. ICML, Jun 2014, Beijing, China (visited on 07/29/2024).

[7] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: an introduction. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. isbn: 9780262039246.

[8] Marco Wiering and Martijn van Otterlo, eds., *Reinforcement Learning: State-of-the-Art*, Adaptation, Learning, and Optimization, vol. 12, 1st ed., Springer Berlin, Heidelberg, 2012. DOI: https://doi.org/10.1007/978-3-642-27645-3. Hardcover ISBN: 978-3-642-27644-6, Softcover ISBN: 978-3-642-44685-6, eBook ISBN: 978-3-642-27645-3.

[9] van Otterlo, M., Wiering, M. (2012). Reinforcement Learning and Markov Decision Processes. In: Wiering, M., van Otterlo, M. (eds) Reinforcement Learning. Adaptation, Learning, and Optimization, vol 12. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27645-3_1. SpringerLink (visited on 07/29/2024).

[10] Andrew G. Barto and Sridhar Mahadevan. "Recent Advances in Hierarchical Reinforcement Learning." *Discrete Event Dynamic Systems* 13, 341–379 (2003). doi: 10.1023/A:1025696116075. SpringerLink (visited on 07/29/2024).

[11] Richard S. Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: Proceedings of the 12th International Conference on Neural Information Processing Systems. NIPS'99. Cambridge, MA, USA: MIT Press, Nov. 1999, pp. 1057–1063. (visited on 07/29/2024).

[12] Thanh Long Vu et al. Barrier Function-based Safe Reinforcement Learning for Emergency Control of Power Systems. 2021. doi: 10.48550/ARXIV.2103.14186. ArXiv (visited on 07/29/2024).

[13] Quanser Consulting, Inc. Linear Motion Servo Plants: IP01 or IP02 - Single Inverted Pendulum (SIP) User Manual, 3 ed.

[14] Ronald J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229-256, May 1992. DOI: https://doi.org/10.1007/BF00992696. SpringerLink (visited on 07/29/2024).

[15] Prakash Mallick, Zhiyong Chen, “Dynamic Programming-based Approximate Optimal Control for Model-Based Reinforcement Learning”. 2023. https://doi.org/10.48550/arXiv.2312.14463. ArXiv (visited on 07/29/2024).

[16] Li, Y. (2017). Deep Reinforcement Learning: An Overview. *ArXiv, abs/1701.07274*. ArXiv (visited on 07/29/2024).

[17] Ben J.A. Kröse, "Learning from delayed rewards," *Robotics and Autonomous Systems*, vol. 15, no. 4, 1995, pp. 233-235, ISSN 0921-8890, https://doi.org/10.1016/0921-8890(95)00026-C, ScienceDirect (visited on 07/29/2024).

[18] Hao Dong, Zihan Ding, and Shanghang Zhang, eds. *Deep Reinforcement Learning: Fundamentals, Research and Applications*. 1st ed. Singapore: Springer Singapore Pte. Ltd., 2020. ISBN: 9789811540950 (eBook), 9789811540943 (Hardcover), 9789811540974 (Softcover). doi: 10.1007/978-981-15-4095-0. SpringerLink (visited on 07/29/2024).

[19] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. 2nd ed. Cham: Springer International Publishing, 2023. ISBN: 978-3-031-29641-3 (Hardcover), 978-3-031-29644-4 (Softcover), 978-3-031-29642-0 (eBook). doi: 10.1007/978-3-031-29642-0. SpringerLink (visited on 07/29/2024).

[20] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.A. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv, abs/1312.5602*. ArXiv (visited on 07/29/2024).

[21] Lawrence, N.P., Forbes, M.G., Loewen, P.D., McClement, D.G., Backstrom, J.U., & Gopaluni, R.B. (2021). Deep Reinforcement Learning with Shallow Controllers: An Experimental Application to PID Tuning. *ArXiv, abs/2111.07171*. ArXiv (visited on 07/29/2024).

[22] Nagabandi, A., Kahn, G., Fearing, R.S., & Levine, S. (2017). Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. 2018 IEEE International Conference on Robotics and Automation (ICRA), 7559-7566. ArXiv (visited on 07/29/2024).

[23] Chen, P.; He, Z.; Chen, C.; Xu, J. "Control Strategy of Speed Servo Systems Based on Deep Reinforcement Learning." *Algorithms* 2018, 11(5), 65. doi: 10.3390/a11050065. MDPI (visited on 07/29/2024).

[24] MinKu Kang, Kee-Eung Kim. (2020). Analysis of Reward Functions in Deep Reinforcement Learning for Continuous State Space Control. Journal of KIISE, 47(1), 78-87, 10.5626/JOK.2020.47.1.78. DBpia (visited on 07/29/2024).

[25] Hu H, Chen Y, Wang T, Feng F, Chen W. Research on the Deep Deterministic Policy Algorithm Based on the First-Order Inverted Pendulum. *Applied Sciences*. 2023; 13(13):7594. https://doi.org/10.3390/app13137594. MDPI (visited on 07/29/2024).

[26] Long-Ji Lin. “Reinforcement Learning for Robots Using Neural Networks”. PhD Thesis. Pittsburgh, PA, USA, 1992. UMI Order No. GAX93-22750.

[27] Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R.Y., Chen, X., Asfour, T., Abbeel, P., & Andrychowicz, M. (2017). Parameter Space Noise for Exploration. *ArXiv, abs/1706.01905*. ArXiv (visited on 07/29/2024).

[28] Mukhopadhyay, R., Sutradhar, A. & Chattopadhyay, P. A novel investigation on the effects of state and reward structure in designing deep reinforcement learning-based controller for nonlinear dynamical systems. *Int. J. Dynam. Control* (2024). https://doi.org/10.1007/s40435-024-01407-6. SpringerLink (visited on 07/29/2024).

[29] Quanser Consulting, Inc. Linear Motion Servo Plants: IP01 or IP02 - IP01 and IP02 User Manual, 5.0 ed.

[30] Ho, T., Tat, T., Ngo, H., Nguyen, T., Bui, D.A., Le, T., Le, V., & Huynh, L. (2023). Applying DDPG Algorithm to Swing-Up and Balance Control for a Double Inverted Pendulum on a Cart. *Robotica & Management, Vol. 28, No. 2, pp. 14-20.* DOI: https://doi.org/10.24193/rm.2023.2.3. url (visited on 07/29/2024).

[31] Kennedy, E.A., & Tran, H. (2016). Swing-up of an Inverted Pendulum on a Cart Using a Modified Energy Based Approach. Proceedings of the International MultiConference of Engineers and Computer Scientists 2016 Vol I, IMECS 2016, March 16 - 18, 2016, Hong Kong. ISBN: 978-988-19253-8-1.

[32] Quanser (2024). Quanser QUBE-Servo 2 Pendulum Control Reinforcement Learning MathWorks, MATLAB Central File Exchange. Retrieved August 4, 2024.

[33] Toshinobu Shintai (2024). Reinforcement-Learning-Inverted-Pendulum-with-QUBE-Servo2 GitHub, GitHub. Retrieved August 4, 2024.

Appendix A: MATLAB Code and Simulink Diagrams

A.1 Code to Setup Parameters for the Simulink Diagrams

Setup Environment

The environment includes the plant, any disturbances, the observation signals and action signals, and anything else that is outside the agent.

```
% load IP02+SIP parameters
run("IP02_linsip_param.m");
% initial pendulum angle (rad)
IC_ALPHA0 = 0; % set to pi to start pendulum in inverted position, 0 for down
position
% Maximum voltage (V)
Vmax = 10;
% max displacement cart +/- 0.43 m
X_MAX = 0.43;
X_MIN = - X_MAX;
% inverted pendulum angle balance threshold (rad)
alpha_bal_threshold = 10.0 * pi / 180;
% reward QR weights
Q11 = 10; Q22 = 20; Q33 = 0; Q44 = 1; R = 0.01; B = -100; % agent 9


% Code to setup the SIP system's model parameters.
% The below code is a modification of the setup code provided by Quanser
%% Motor
% Motor Armature Resistance in Ohms
Rm = 2.6;
% Motor Armature Inductance (H)
Lm = 180e-006;
% Motor ElectroMechanical Efficiency [ = Tm * w / ( Vm * Im ) ]
eta_m = 1;
% Rotor Moment of Inertia in kg.m^2
Jm = 3.90e-007;
% Planetary Gearbox Gear Ratio
Kg = 3.71;
% Planetary Gearbox Efficiency
eta_g = 1;
% Motor Pinion Radius in meters
rmp = 6.35e-003;
% Motor Torque Constant in N.m/A
Kt = 7.68e-3;
% Back-ElectroMotive-Force Constant in V.s/rad
Km = Kt;
%
%% Cart
% Cart Weight Mass in kg
Mw = 0.37;
```

```
% IP02 Cart Mass, with 3 cable connectors (kg)
Mc = 0.57;
% Total mass of the cart
M = 0.57 + Mw;
% Equivalent Viscous Damping Coefficient in N.m.s/rad
Beq = 5.4;
% Cart Travel (m)
Tc = 0.814;
% Specifications of a second-order low-pass filter
wcf = 2 * pi * 10.0;  % filter cutting frequency
zetaf = 0.9;        % filter damping ratio
%
%% Pendulum Link
% Pendulum Mass in kg
Mp = 0.230;
% Pendulum Length from Pivot to COG in meters
lp = 0.3302;
% Pendulum center of mass (m)
l = lp/2;
% Pendulum Moment of Inertia about its COG in kg.m^2
Jp = 7.88E-003;
% Viscous Damping Coefficient of the pendulum in N.m.s/rad
Bp = 0.0024;
% Gravitational Constant in m/s^2
g = 9.81;

% Rack Pitch (m/teeth)
Pr = 1e-2 / 6.01; % = 0.0017
% Cart Position Pinion number of teeth
N_pp = 56;

global K_EC K_EP
   % Cart Encoder Resolution (m/count)
    K_EC = Pr * N_pp / ( 4 * 1024 ); % = 22.7485 um/count
    % Pendulum Encoder Resolution (rad/count)
    % K_EP is positive, since CCW is the positive sense of rotation
    K_EP = 2 * pi / ( 4 * 1024 ); % = 0.0015

% Turn on or off the safety watchdog on the cart position: set it to 1 , or 0
X_LIM_ENABLE = 1;       % safety watchdog turned ON
% X_LIM_ENABLE = 0;      % safety watchdog turned OFF

% Cable Gain used: set to 1
K_AMP = 1;

% Amplifier Type: set to 'VoltPAQ' or 'Q3'
AMP_TYPE = 'VoltPAQ';
% AMP_TYPE = 'Q3';
% Digital-to-Analog Maximum Voltage (V); for MultiQ cards set to 10
VMAX_DAC = 10;

% Reference Energy (J)
Er = 2*Mp*lp*9.81;
```

SIP Nonlinear EOMs - DOWN

```
function [d2xdt2, d2aldt2, Fc] = pen_nonlin_eoms(Vm, dxdt, daldt, Jm, lp, Mp,
Beq, Bp, Rm, al, g, Kg, Kt, Km, rmp, M)

Fc = - (Kg * Kt * Km * dxdt) / (Rm * rmp^2) + (Kg * Kt * Vm) / (Rm * rmp);

% Denominator
D = 4 * M * rmp^2 + Mp * rmp^2 + 4 * Jm * Kg^2 + 3 * Mp * rmp^2 * sin(al)^2;

% Acceleration of the cart
d2xdt2 = (1 / D) * (-(3 * rmp^2 * Bp * cos(al) * daldt) / lp - 4 * Mp * lp *
rmp^2 * sin(al) * daldt^2 - 4 * rmp^2 * Beq * dxdt + 3 * Mp * rmp^2 * g * cos(al)
* sin(al) + 4 * rmp^2 * Fc);

% Angular acceleration of the pendulum
d2aldt2 = (1 / D) * (- 3 * (M * rmp^2 + Mp * rmp^2 + Jm * Kg^2) * Bp * daldt /
(Mp * lp^2) - 3 * Mp * rmp^2 * cos(al) * sin(al) * daldt^2 - 3 * rmp^2 * Beq *
cos(al) *dxdt / lp + 3 * (M * rmp^2 + Mp * rmp^2 + Jm * Kg^2) * g * sin(al) / lp
+ 3 * rmp^2 * cos(al) * Fc / lp);
```

Observation and Action Signals

The observation and action signals for the environment are defined based on the Quanser Single Inverted Pendulum (SIP) system mounted on a Linear Cart IP02. The interface is then configured to connect with the Simulink model/environment, which includes the nonlinear dynamics of the pendulum. For this linear inverted pendulum system, there are four key observation signals: the cart's position, the angular position, and the velocities of both the cart and the pendulum link, represented as $[x \quad \alpha \quad \dot{x} \quad \dot{\alpha}]$. There is also a single action signal, which is the maximum motor voltage, V_{max}. Additionally, a reset function is implemented to adjust the initial position of the inverted pendulum within a range of $+/-10\ deg$ degrees from the vertical.

```
% observation signals "rlNumericSpec" creates action/observative data of a given
dimension and signal limits
obsInfo = rlNumericSpec([4 1],'LowerLimit',[-inf -inf -inf -
inf]','UpperLimit',[inf inf inf inf]');
obsInfo.Name = 'observations';
obsInfo.Description = 'x, alpha, x_dot, alpha_dot';
numObs = obsInfo.Dimension(1);
% action signals - constrained to motor limit
actInfo = rlNumericSpec([1 1],'LowerLimit',-Vmax,'UpperLimit',Vmax);
actInfo.Name = 'Motor Voltage';
% numActions = actInfo.Dimension(1);
% create environment object
mdl = 'lin_sip_ip02_bal_rl';
agentBlk = [mdl '/RL Agent'];
env = rlSimulinkEnv(mdl,agentBlk,obsInfo,actInfo);
% reset function used to randomize initial position of pendulum
env.ResetFcn = @(in)localResetFcn(in);
% Sampling rate
Ts = 0.01;
% Simulation duration
```

```
Tf = 10;
% Fix the random generator seed for reproducibility.
rng(0);
```

Creating a DDPG Agent

The MathWorks® Reinforcement Learning Toolbox™ offers a variety of agent types that can be employed to generate policies. Among these is the Deep Deterministic Policy Gradient (DDPG) algorithm, a model-free, online, and off-policy reinforcement learning approach. The DDPG agent operates within an actor-critic framework, enabling it to determine an optimal policy that maximizes long-term rewards.

```
statePath = [
    imageInputLayer([numObs 1 1],'Normalization','none','Name','observation')
    fullyConnectedLayer(128,'Name','CriticStateFC1')
    reluLayer('Name', 'CriticRelu1')
    fullyConnectedLayer(200,'Name','CriticStateFC2')];
actionPath = [
    imageInputLayer([1 1 1],'Normalization','none','Name','action')
    fullyConnectedLayer(200,'Name','CriticActionFC1','BiasLearnRateFactor',0)];
commonPath = [
    additionLayer(2,'Name','add')
    reluLayer('Name','CriticCommonRelu')
    fullyConnectedLayer(1,'Name','CriticOutput')];
%
criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);
%
criticNetwork = connectLayers(criticNetwork,'CriticStateFC2','add/in1');
criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');
% plot network
plot(criticNetwork)
%
% Specify options for the critic representation .
criticOpts = rlRepresentationOptions('LearnRate',1e-4,'GradientThreshold',1);
%
% Create critic representation using the specified deep neural network
obsInfo = getObservationInfo(env);
actInfo = getActionInfo(env);
critic =
rlQValueRepresentation(criticNetwork,obsInfo,actInfo,'Observation',{'observation'
},'Action',{'action'},criticOpts);
%
% Create the actor, first create a deep neural network with one input,
% the observation, and one output, the action.
% Construct actor similarly to the critic.
actorNetwork = [
    imageInputLayer([numObs 1 1],'Normalization','none','Name','observation')
```

```
    fullyConnectedLayer(128,'Name','ActorFC1')
    reluLayer('Name','ActorRelu1')
    fullyConnectedLayer(200,'Name','ActorFC2')
    reluLayer('Name','ActorRelu2')
    fullyConnectedLayer(1,'Name','ActorFC3')
    tanhLayer('Name','ActorTanh')
    scalingLayer('Name','ActorScaling','Scale',max(actInfo.UpperLimit))];
%
actorOpts = rlRepresentationOptions('LearnRate',1e-4,'GradientThreshold',1);
%
actor =
rlDeterministicActorRepresentation(actorNetwork,obsInfo,actInfo,'Observation',{'o
bservation'},'Action',{'ActorScaling'},actorOpts);
% specify agent options
agentOpts = rlDDPGAgentOptions(...
    'SampleTime',Ts,...
    'TargetSmoothFactor',1e-3,...
    'ExperienceBufferLength',1e6,...
    'DiscountFactor',0.995,...
    'MiniBatchSize',128);
% For continuous action signals, it is important to set the noise variance
% appropriately to encourage exploration. It is common to have
% Variance*sqrt(Ts) be between 1% and 10% of your action range
agentOpts.NoiseOptions.Variance = 0.3;
agentOpts.NoiseOptions.VarianceDecayRate = 1e-5;
%
% create the DDPG agent using the specified actor representation, critic
% representation and agent options.
agent = rlDDPGAgent(actor,critic,agentOpts);
```

Train or Load Agent

This process involves configuring the necessary parameters and initiating the reinforcement learning training session, either to create a new agent or to load an existing one. Users have the option to specify the MAT file of the agent to be loaded or to define the file name for storing the newly trained agent.

```
% Load pre-defined agent (false) or train new agent (true)
% doTraining = false;
doTraining = true;

% Ensure Tf and Ts are defined somewhere above this line
maxsteps = ceil(Tf/Ts);  % Calculate maximum number of steps for training

% Setup training parameters
maxEpisodes = 10000;
maxSteps = floor(Tf/Ts);
trainOpts = rlTrainingOptions(...
    'MaxEpisodes',maxEpisodes, ...
    'MaxStepsPerEpisode',maxSteps, ...
    'ScoreAveragingWindowLength',5, ...
```

```
    'Verbose',false, ...
    'Plots','training-progress',...
    'StopTrainingCriteria','EpisodeReward',...
    'StopTrainingValue', -10);

% Ensure doTraining is defined as true or false
if doTraining
    % Train the agent.
    trainingStats = train(agent,env,trainOpts);
    % Save trained agent
    save('SIPIP02BalDDPG17.mat','agent');
else
    % Load pretrained agent. Choose one to load based on your requirements.
    load('SIPIP02BalDDPG16.mat','agent');
    % load('SIPIP02BalDDPGNominalParam.mat','agent'); % Uncomment if needed
end
```

Simulating Reinforcement Learning with the Quanser Single Inverted Pendulum (SIP) System Mounted on a Linear Cart IP02

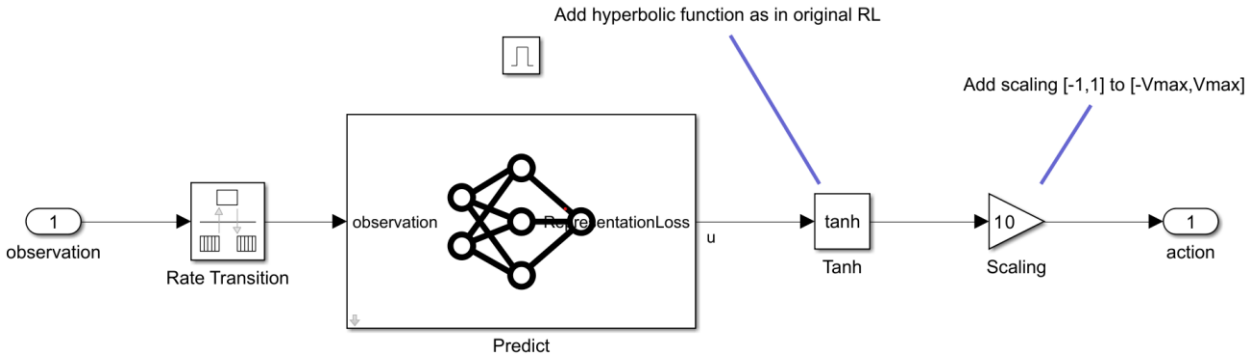
The Simulink model used for training can also be utilized to simulate the balance control of the inverted pendulum using the single inverted pendulum model. To begin, open the lin_sip_ip02_bal_rl Simulink model as demonstrated below.

Run the Simulink model and experiment with adjusting the initial angle of the inverted pendulum within a range of +/- 10 degrees. For instance, set the IC0 block to 0.96*ic_alpha0 to start the pendulum at an initial angle of 172.8 degrees, which is 7.2 degrees away from the perfectly upright position. Note that the variable ic_alpha0 is initially set to 180 degrees, representing the balanced and upright angle of the inverted pendulum.

```
% open Simulink model to simulate IP02+SIP RL balance control
open("lin_sip_ip02_bal_rl.slx");
```

Running RL using the Quanser SIP Cart IP02 Hardware

The Quanser QUARC Real-Time Control Software is required to interface the Quanser Single Inverted Pendulum (SIP) System mounted on a Linear Cart IP02 with Simulink. QUARC creates an executable for 64-bit Windows using code generated from Simulink Coder and MATLAB code. This executable is then run through the Simulink interface in full External mode. As of MATLAB R2023b, the RL Agent block does not support code generation. Instead, the trained agent is deployed on the 64-bit Windows target using QUARC by employing the Predict block from the Deep Learning Toolbox, as demonstrated below.



Generating a Policy

Generate the policy for the trained agent to be used with the Deep Learning Predict block, or load an existing policy if one has already been created.

```
% Load pre-defined policy (false) or generate new policy for RT code gen (true)
doPolicy = false;
% doPolicy = true;
%
if doPolicy
    % Generate policy for deployment if it doesn't exist
    policyFile = 'SIPIP02BalRLPolicy.mat';
    if ~exist(policyFile, 'file')
        generatePolicyFunction(agent, 'MATFileName', policyFile);
    end
    load(policyFile, 'policy');
    policy = cut_unnecessary_layers_for_SAC_policy(policy);
    save(policyFile, 'policy');
else
    % Load previously saved policy
    load('SIPIP02BalRLPolicy.mat', 'policy');
end
```

Run Simulink/QUARC

```
% load the Simulink model that uses QUARC to run the RL balance control using the
IP02+SIP
open("q_sipip02_bal_rl_hw.slx");
```

Reset Function

The reset function initializes the angle of the inverted pendulum within a range of +/- 10 degrees from the vertical position at the start of each training episode.

```
function in = localResetFcn(in)
    % randomize initial inverted position angle
    blk = sprintf('lin_sip_ip02_bal_rl/IC0');
    % initialize angle of pendulum +/- 10 deg about vertical upright position
    ic0 = ( 20 * ( rand()-0.5) + 180 ) * pi/180;
    in = setBlockParameter(in,blk,'Value',num2str(ic0));
end
```

A.2 Simulink Model for Training the Reinforcement Learning Agent

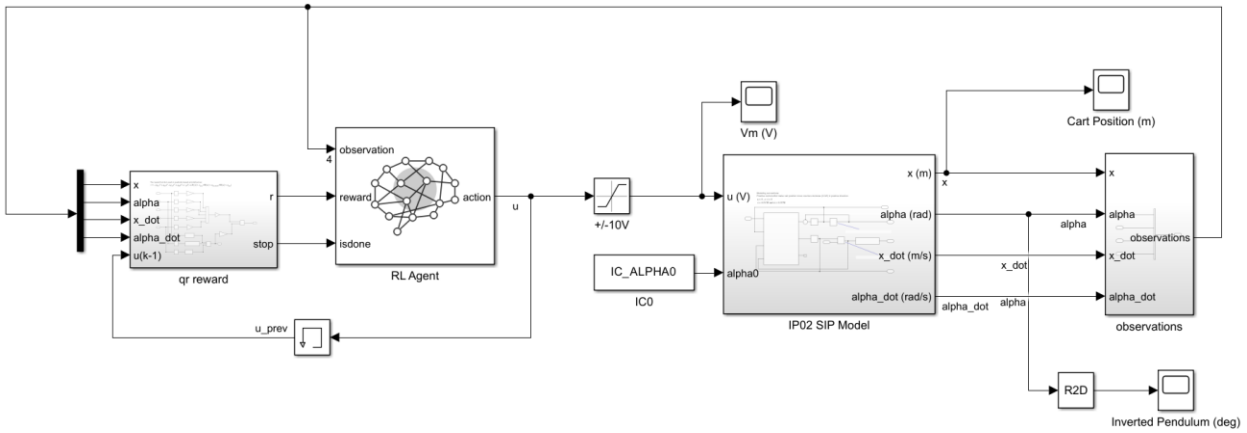


Figure A.2.1: The Simulink model utilized for training the reinforcement learning agent is based on a nonlinear model of the Quanser Single Inverted Pendulum (SIP) System mounted on a Linear Cart IP02. In this setup, the Simulink model serves as the RL environment, incorporating the RL Agent block to implement the agent. The model processes observation and reward signals, which guide the agent's decision-making. The agent, in turn, generates an action—specifically, the motor voltage—to maintain the balance of the pendulum. This configuration allows for effective training of the agent to achieve stable control of the inverted pendulum.

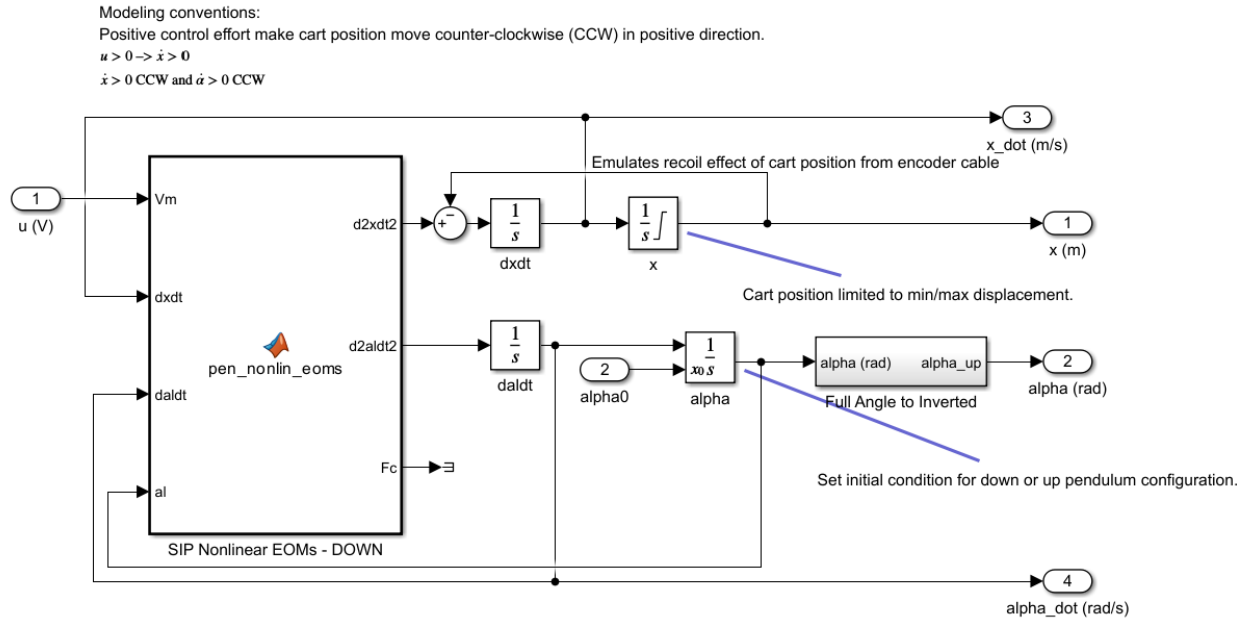


Figure A.2.2: This diagram represents the nonlinear equations of motion for the Single Inverted Pendulum (SIP) system mounted on a Linear Cart IP02. The model simulates the dynamics of the pendulum when it starts from the downward position. The system inputs a control voltage u(V) which influences the cart's position x(m) and velocity x_dot(m/s), as well as the pendulum's angle alpha(rad) and angular velocity alpha_dot(rad/s). The model includes an emulation of the recoil

effect due to the cart's position, constraints to limit the cart's displacement, and conditions to set the initial pendulum angle either in the down or inverted (up) position. The nonlinear equations of motion (EOMs) are essential for accurately simulating the physical behavior of the system under different control inputs.

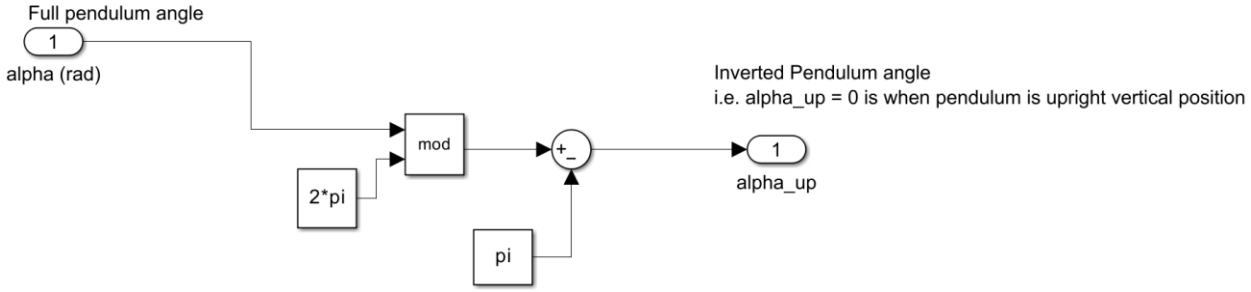


Figure A.2.3: This diagram illustrates the process for calculating the inverted pendulum angle, referred to as alpha_up. The input to the system is the full pendulum angle alpha (rad), which represents the pendulum's current angular position. The model then uses the modulo operation to normalize this angle within a 2*pi range, ensuring that the angle is properly adjusted for continuous rotation. The resulting value is then offset by subtracting pi, which shifts the reference point so that alpha_up equals 0 when the pendulum is in the upright vertical position. This adjusted angle, alpha_up, is crucial for accurately determining the pendulum's position relative to its upright, balanced state.

The reward function used in quadratic-based and defined as:

$$r = -(q_{11}x^2 + q_{22}\alpha^2 + q_{33}\dot{x}^2 + q_{44}\dot{\alpha}^2 + r_{11}u^2) + B(|x| > x_{max}\ OR\ |\alpha| > \alpha_{threshold}\ OR\ |u| > u_{max})$$

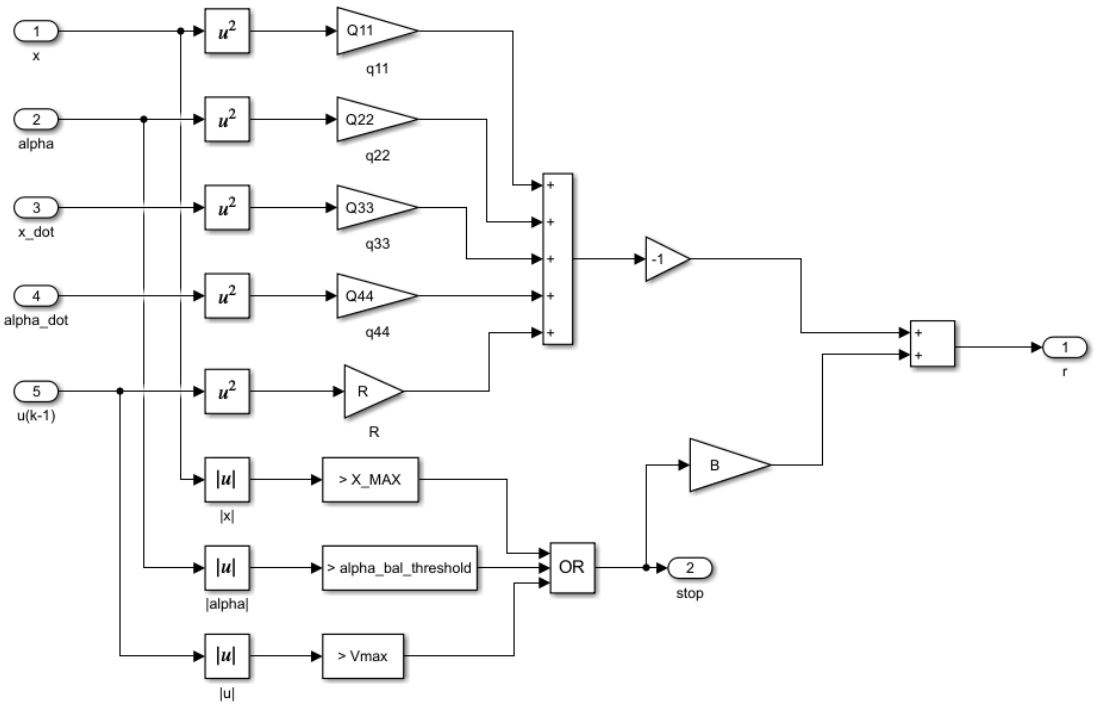


Figure A.2.4: This diagram illustrates the quadratic-based reward function used in the reinforcement learning model for an inverted pendulum system. The function penalizes deviations in cart position, pendulum angle, and their respective velocities, while also considering the control effort. Additional penalties are applied if the system exceeds predefined limits for cart displacement, pendulum angle, or control input, which helps guide the agent towards achieving stable and efficient control.

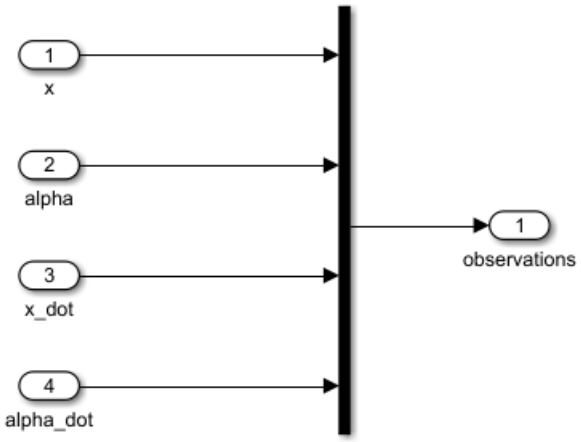


Figure A.2.5: This diagram shows the observation signals used in the control model for an inverted pendulum system. The inputs include the cart position (x), pendulum angle (alpha), cart velocity (x_dot), and angular velocity of the pendulum (alpha_dot). These signals are combined to form the complete observation vector, which is then used by the reinforcement learning agent to make decisions and control the system effectively.

A.3 Simulink Diagram for Running RL Balance Control on the Quanser Single Inverted Pendulum (SIP) System Mounted on a Linear Cart IP02

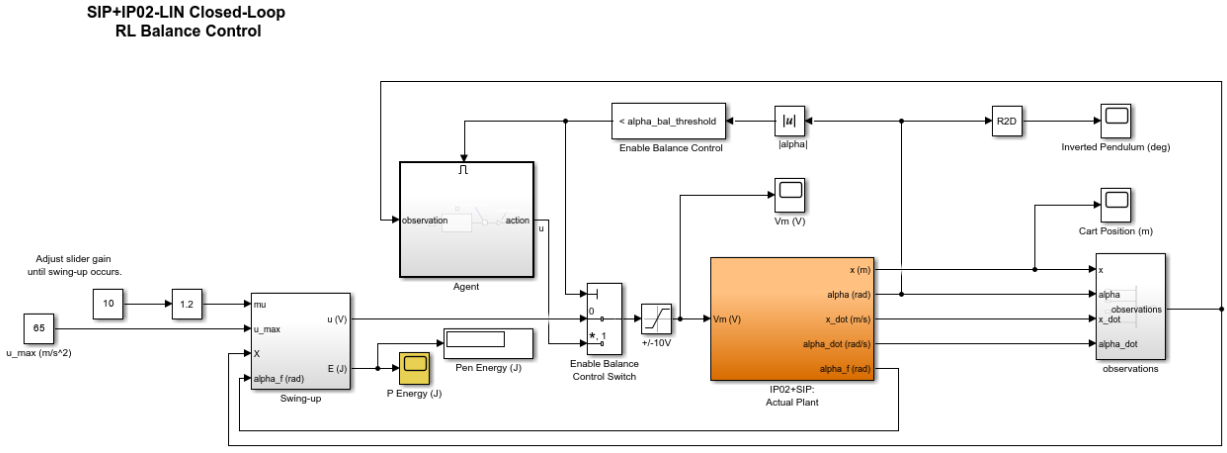


Figure A.3.1: This figure illustrates the Simulink diagram integrated with QUARC Real-Time Control Software, used for implementing both swing-up and reinforcement learning (RL) balance

control on the Quanser Single Inverted Pendulum (SIP) System mounted on a Linear Cart IP02. The diagram showcases the control architecture, where the RL Agent interacts with the physical hardware through QUARC, processing observation signals from the system and outputting control actions, such as motor voltage, to first swing up the pendulum and then maintain its balance. This setup facilitates real-time control of the inverted pendulum, demonstrating the effective application of reinforcement learning in managing complex, nonlinear dynamics.

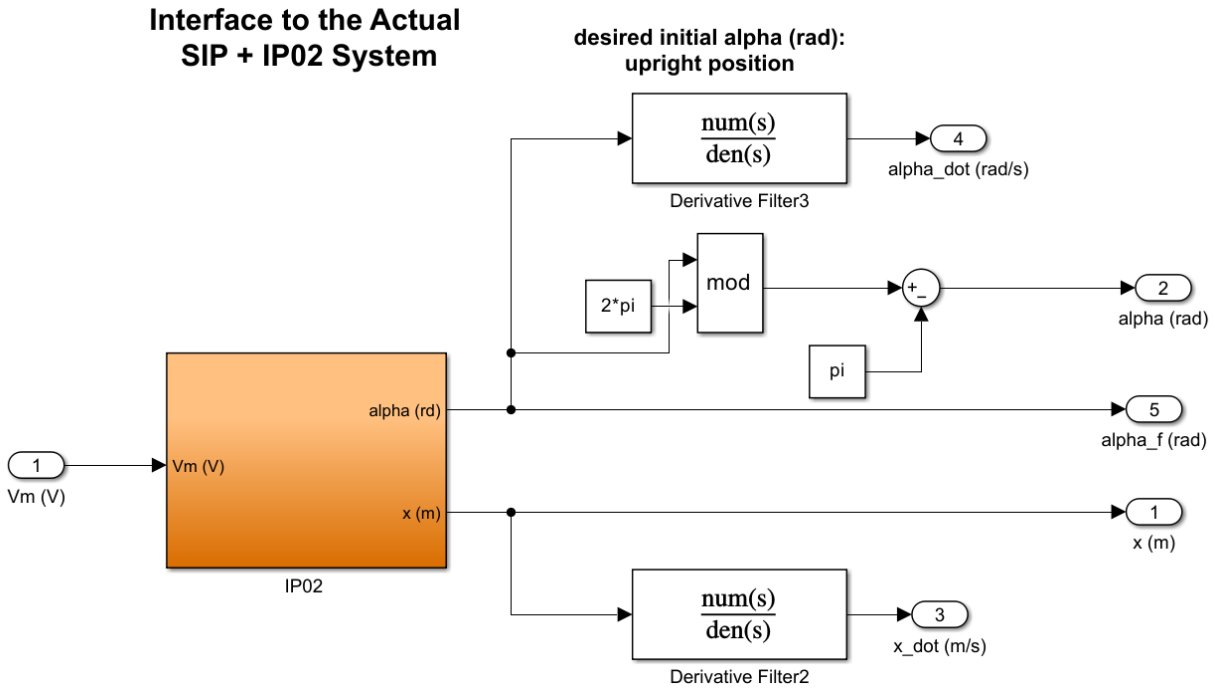


Figure A.3.2: This figure shows the interface between the control system and the Quanser Single Inverted Pendulum (SIP) on a Linear Cart IP02. The input voltage Vm (V) controls the pendulum, while the outputs alpha (rad) and x (m) represent the pendulum's angle and cart's position. These outputs are processed to compute velocities (alpha_dot and x_dot), enabling the control algorithm to maintain the pendulum's upright position.

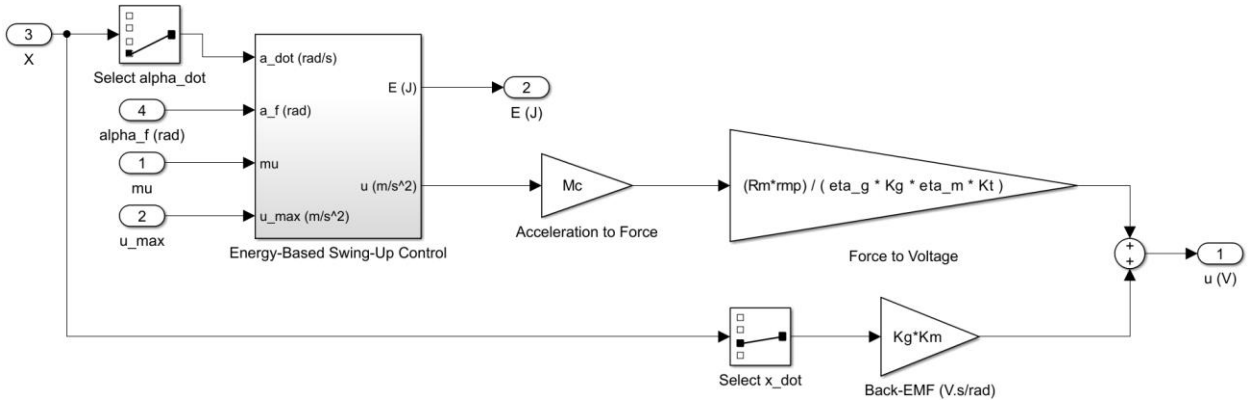


Figure A.3.3: The diagram illustrates the energy-based swing-up control system designed for a Quanser Single Inverted Pendulum (SIP) mounted on a Linear Cart IP02. This Simulink model calculates and applies the necessary control inputs to effectively swing the pendulum from its

initial downward position to an upright, balanced position, and subsequently maintain its stability. The system integrates various components to observe the pendulum's dynamics, compute control actions, and apply them to achieve and sustain the desired balance.

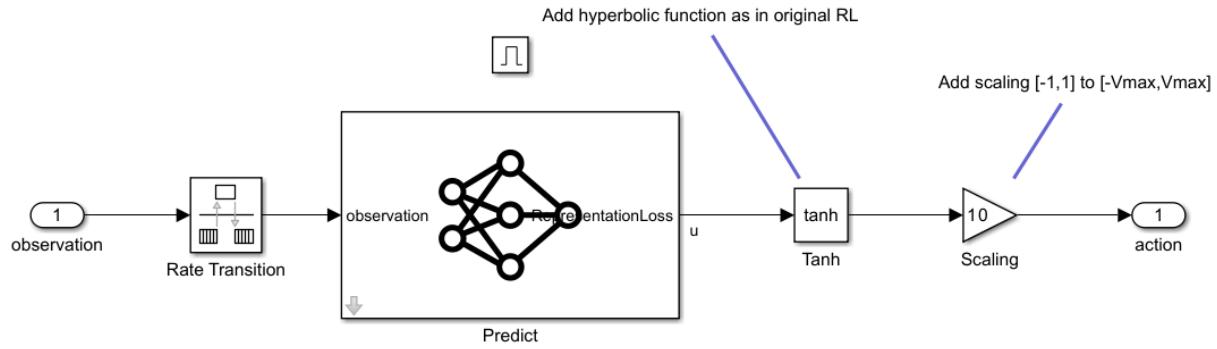


Figure A.3.4: This diagram illustrates the process of predicting actions using a reinforcement learning agent in a Simulink model. The observation input is processed through a Rate Transition Block and then fed into the Predict Block, which contains the neural network of the RL agent. The predicted action is passed through a Tanh Activation Function to constrain the output, followed by a Scaling Block that maps the action to the appropriate control range. This final output is the control signal used to drive the system, such as balancing the Quanser Single Inverted Pendulum (SIP).

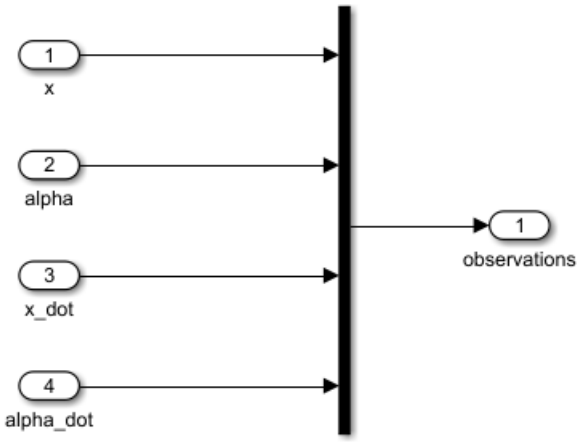


Figure A.3.5: This diagram illustrates how the observation signals are structured in the Quanser Single Inverted Pendulum (SIP) on a Linear Cart IP02 system. The system collects four key observation signals: cart position x, pendulum angle alpha, cart velocity x_dot, and pendulum angular velocity alpha_dot. These signals are then combined into a single observation vector, which is used by the control algorithm to make decisions about the system's behavior.