SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Ing. Michal Hlavatý

Autoreferát dizertačnej práce

Inteligentné metódy riadenia vybraných mechatronických systémov

na získanie akademického titulu
„doktor“ („philosophiae doctor“, v skratke „PhD.“)

v doktorandskom študijnom programe:	Mechatronické systémy
v študijnom odbore:	kybernetika
forma štúdia:	denná

Miesto a dátum: Bratislava, 21.5.2024

Dizertačná práca bola vypracovaná na:

Ústav automobilovej mechatroniky
Fakulta elektrotechniky a informatiky
Slovenská technická univerzita v Bratislave
Ilkovičova 2961, 841 04 Karlova Ves

Predkladateľ:

Ing. Michal Hlavatý
Ústav automobilovej mechatroniky
Fakulta elektrotechniky a informatiky
Slovenská technická univerzita v Bratislave
Ilkovičova 2961, 841 04 Karlova Ves

Školiteľ:

doc. Ing. Alena Kozáková, PhD.
Ústav automobilovej mechatroniky
Fakulta elektrotechniky a informatiky
Slovenská technická univerzita v Bratislave
Ilkovičova 2961, 841 04 Karlova Ves

Oponenti:

doc. Ing. Anna Jadlovská, PhD,
Katedra kybernetiky a umelej inteligencie
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach
Vysokoškolská 4, 042 00 Košice
anna.jadlovska@tuke.sk

prof. Ing. Aleš Janota, PhD.
Katedra riadiacich a informačných systémov
Fakulta elektrotechniky a informačných technológií
Žilinská univerzita v Žiline
Univerzitná 8215/1, 010 26 Žilina
ales.janota@uniza.sk

Autoreferát bol rozoslaný dňa: .....

Obhajoba dizertačnej práce sa bude konať dňa ..... o ..... hod. na Fakulte elektrotechniky a informatiky STU v Bratislave, Ilkovičova 2961, 841 04 Karlova Ves, v miestnosti ..... .

.....

prof. Ing. Vladimír Kutiš, PhD.

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný odbor:	kybernetika
Študijný program:	Mechatronické systémy
Autor:	Ing. Michal Hlavatý
Dizertačná práca:	Inteligentné metódy riadenia vybraných mechatronických systémov
Vedúci práce:	prof. Ing. Alena Kozáková, PhD.
Mesiac a rok odovzdania:	október 2024

Kľúčové slová: automatické riadenie, nelineárny systém, neurónové siete, proximal policy optimization, reinforcement learning, soft actor-critic, vzduchová levitácia

Predložená dizertačná práca sa zaoberá výskumom, návrhom a realizáciou inteligentného riadenia nelineárnych systémov na báze metodiky reinforcement learning. V práci sú hĺbkovo analyzované algoritmy na báze metodiky reinforcement learning vhodné na riadenie, je vytvorený simulačný model a komunikačné rozhranie medzi softwareom a hardwareom reálneho laboratórneho modelu. Pri riešení cieľov dizertačnej práce boli využité najmodernejšie technológie, a to OpenAI Gymnasium pre tvorbu modelu a Stable Baselines 3 pre návrh riadenia.

Možnosti riadenia reálnych dynamických nelineárnych systémov pomocou algoritmov na báze reinforcement learning sú overené na dvoch prípadových štúdiách:

1. Simulačné riadenie dvojitého kyvadla pomocou algoritmu Soft Actor-Critic
2. Riadenie unikátneho laboratórneho modelu vzduchovej levitácie pomocou algoritmov reinforcement learning simulačne

ako aj v reálnom čase. Ako najvhodnejší algoritmus na riadenie reálneho modelu vzduchovej levitácie sa ukázal algoritmus Soft Actor-Critic.

Postup návrhu a implementácie je zovšeobecnený pre možné využitie na riadenie levitačných systémov.

ABSTRACT

Slovak University of Technology in Bratislava

FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Branch: cybernetics

Study Programme: Applied mechatronics and electromobility

Author: Ing. Michal Hlavatý

Thesis: Intelligent control methods for selected mechatronic systems

Supervisor: prof. Ing. Alena Kozáková, PhD.

Submitted: october 2024

Keywords: nonlinear system, neural networks, reinforcement learning, proximal policy optimization, soft actor-critic, air levitation

The dissertation thesis deals with the research, design, and implementation of intelligent control strategies for nonlinear systems using reinforcement learning methodology. It provides a detailed analysis of RL algorithms suitable for control applications, alongside with the development of a simulation model and a communication interface between the software and hardware of a laboratory-scale model. State-of-the-art technologies, namely OpenAI Gymnasium for model building and Stable Baselines 3 for control design, have been used to address the objectives of the thesis.

The effectiveness of RL-based algorithms for controlling real-world dynamic nonlinear systems is demonstrated through two case studies:

1. Simulation control of a double pendulum using the Soft Actor-Critic algorithm

2. Control of a unique laboratory model of air levitation using reinforcement learning algorithms both in simulation and in real time. The Soft Actor-Critic algorithm proved to be the most effective for controlling the real air levitation model.

The design and implementation procedure is generalized for possible use for control of levitation systems.

OBSAH

Úvod

Celosvetové inovatívne trendy v oblasti komplexnej automatizácie a informatizácie procesov v posledných desaťročiach smerujú k potrebe výskumu, vývoja a implementácie nových systémových zmien, ktoré vyžadujú nové inteligentné, vysoko autonómne vnorené systémy riadenia, diagnostiky a komunikácie. Automatické systémy riadenia spolu s informačnými a komunikačnými technológiami tak predstavujú silný evolučný faktor výrazne ovplyvňujúci nielen priemyselné, informačné a komunikačné systémy ale i netechnické procesy ako sú bankovníctvo, zdravotníctvo a služby. Aplikáciou moderných komplexných systémov riadenia s využitím najnovších metód automatického riadenia v integrácii s numerickými metódami a inteligentnými algoritmami a informačnými technológiami je možné dosiahnuť vysokú kvalitu, spoľahlivosť a bezpečnosť prevádzok, systémov a zariadení.

Jedným z najdôležitejších faktorov, ktoré ovplyvňujú kvalitu automatických systémov riadenia, sú metódy automatického riadenia. Popri tradičných metódach (PID, LQ a LQG) sa aktuálne do popredia dostávajú pokročilé metódy na báze najnovších poznatkov numerickej matematiky, metód výpočtovej inteligencie ako aj informačných a komunikačných technológií.

Väčšina súčasných metód návrhu nelineárnych regulátorov vyžaduje úplnú znalosť modelu riadeného procesu, čo však často nie je reálne. Preto sa realizuje výskum a vývoj metód návrhu algoritmov riadenia, ktoré je možné využiť pri riadení nelineárnych a časovo premenlivých procesov bez znalosti ich matematického modelu (robustné a adaptívne prístupy). Rozvoj metód riadenia patriacich do skupiny tzv. soft technik umožnil ďalšie skvalitnenie riadiacich algoritmov spojitých procesov. Táto skupina zahŕňa napr. návrh a aplikáciu regulátorov na báze fuzzy logiky, umelých neurónových sietí a genetických algoritmov [1].

Umelé neurónové siete (Artificial Neural Networks, ANN) sú vďaka svojim univerzálnym aproximačným vlastnostiam silným modelovacím a riadiacim prostriedkom, ktorý si získal významné miesto v oblasti automatického riadenia zložitých nelineárnych procesov. Cieľom tréningu ANN je vytvoriť sieť, ktorá dokáže pochopiť zložité vzory v dátach, identifikovať relevantné charakteristiky a správne klasifikovať alebo predikovať stavy na základe vstupných informácií [2]. Nedávny pokrok umožnil ANN preniknúť do rôznych aplikáciách vrátane zdravotníctva, kde konvolučné neurónové siete dosahujú pozoruhodné úspechy pri diagnostike ochorení, ako sú nádory mozgu zo snímok magnetickej rezonancie, pričom dosahovali presnosť až 97 % [3]. Neurónové siete zohrávajú kľúčovú úlohu aj v riadení, pretože ponúkajú pokročilé riešenia pre komplexné systémy. V riadiacich aplikáciách dokážu neurónové siete efektívne identifikovať a riadiť nelineárne dynamické systémy. Na modelovanie systémov s komplexnou dynamiku je možné využiť nelineárne autoregresné modely s exogénnymi vstupmi (NARX) a následne ich trénovať pomocou vstupno-výstupných údajov z fyzikálnych zostáv s cieľom efektívne identifikovať neznáme parametre [4]. V regulačnom obvode teda môže neurónová sieť zastávať funkciu modelu, regulátora, predikčného člena a pod., a teda nahrádzať jeho konvenčné prvky [5].

Učenie s posilňovaním (Reinforcement learning, RL) sa využíva na riešenie problémov optimálneho riadenia vyjadrených ako Markovove rozhodovacie procesy (Markov Decision Processes - MDP) [6], ktoré pracujú diskrétne v čase: v každom časovom kroku dostáva regulátor spätnú väzbu od systému v podobe stavového signálu a v reakcii naň vykoná akciu. Rozhodovacím pravidlom je zákon riadenia so spätnou väzbou od stavu, ktorý sa v RL nazýva politika. Akcia stochasticky mení stav systému a posledný prechod sa vyhodnocuje prostredníctvom funkcie odmeny (záporné náklady). Cieľom optimálneho riadenia je maximalizovať z každého počiatočného stavu

(očakávanú) kumulatívnu odmenu, tzv. hodnotu. Ide teda o problém sekvenčného rozhodovania s cieľom optimalizovať dlhodobý výkon. Na rozdiel od tradičných metód učenia, kde model získava informácie z preddefinovaných dát, v RL agent neustále skúma prostredie a učí sa priamo z výsledkov svojho správania prostredníctvom vzoriek prechodov a odmien, a to buď offline (na dávke vzoriek získaných vopred zo systému) alebo online (zo vzoriek získaných priamo zo systému v uzavretej slučke, súčasne s učením optimálneho regulátora) [7].

Vďaka schopnosti ANN modelovať zložité vzťahy v dátach a RL, ktoré dokáže optimalizovať svoje správanie na základe interakcie s prostredím, vzniká silná kombinácia pre riešenie širokej škály problémov v oblastiach ako sú robotika [8]–[12], autonómne systémy [13]–[15], riadenie zložitých dynamických systémov[16], [17], finančné trhy [18], [19] a dokonca aj medicína [20]–[22].

Na RL môžeme nazerať z rôznych pohľadov, a to najmä z pohľadu umelej inteligencie (artificial intelligence, AI), ktorá poskytuje najširšiu a najvšeobecnejšiu škálu algoritmov, a z pohľadu využitia v teórii riadenia, kde sa RL tiež nazýva aproximatívne alebo adaptívne dynamické programovanie (ADP). Kombináciu RL s hlbokými neurónovými sieťami (deep neural networks, DNN) označujeme pojmom deep reinforcement learning (DRL). Jedným z moderných spôsobov ako riadiť zložitý nelineárny systém je práve využitie neurónových sietí, či už ide o vylepšenie existujúceho regulátora alebo o nové, komplexné riadenie systému [23]–[25].

Levitačné systémy (vzduchové, magnetické) sú založené na jednoduchom princípe pôsobenia rovnovážnych síl, ktoré zabezpečujú vznášanie sa levitujúceho predmetu v požadovanej výške. V systémoch magnetickej levitácie pôsobí proti gravitačnej sile sila magnetického poľa, v systémoch vzduchovej levitácie (VL) je to sila pôsobiaca v dôsledku prúdiaceho vzduchu z ventilátora. Na riadenie polohy levitujúceho objektu v laboratórnom

zariadení VL sa upravuje prúdenie vzduchu na základe údajov zo snímačov polohy, ktoré sa spracúvajú pomocou riadiacich algoritmov, najčastejšie štandardných PID [26]–[29], ktoré však vzhľadom na silne nelineárnu a rýchlu dynamiku ako aj nízke tlmiace vlastnosti systému VL nedokážu zabezpečiť jeho stabilitu v celom pracovnom rozsahu. Preto sa implementujú metódy na báze pokročilých prístupov ako MPC [30], robustné riadenie a riadenie s prepínaním [31].

Predložená dizertačná práca sa zaoberá návrhom a implementáciou inteligentného riadenia s učením typu RL pre reálny vzduchový levitačný systém na báze laboratórneho zariadenia FloatShield [32]. Práca podrobne analyzuje rôzne architektúry neurónových sietí, pričom sa zameriava najmä na metodiku RL a príslušné algoritmy a možnosti ich aplikácie na riadenie systému vzduchovej levitácie, ktorý predstavuje náročný problém modelovania spočívajúci v opise nelineárnej závislosti rýchlosti prúdenia vzduchu a polohy guličky v trubici [26], [28], [32].

Práca je rozdelená do šiestich kapitol. Prvá kapitola poskytuje všeobecný teoretický prehľad o umelej inteligencii a neurónových sieťach so zameraním na princípy rôznych typov učenia a ich využitia. V druhej kapitole sa práca venuje opisu vybraných architektúr neurónových sietí s aplikáciami v oblasti riadenia. Tretia kapitola je venovaná metodike reinforcement learningu s dôrazom na relevantné algoritmy ako napríklad Actor-Critic (AC), Advantage Actor-Critic (A2C), Soft Actor-Critic (SAC) a ďalšie. Špecifikácia cieľa dizertačnej práce je obsahom štvrtej kapitoly. Ťažiskom práce je piata a šiesta kapitola. Piata kapitola opisuje simulačné modely nelineárneho laboratórneho systému FloatShield a ich tvorbu s využitím rôznych prostredí (Matlab Simulink, Ansys Fluent a Python), v šiestej kapitole sú prezentované dve prípadové štúdie aplikácie vybraných algoritmov RL na riadenie simulačných nelineárnych systémov a laboratórneho modelu vzduchovej levitácie.

1. Metodika Reinforcement Learning (RL)

Trénovanie neurónovej siete na rozlišovanie objektov, predikciu alebo identifikáciu chybových signálov sa stáva čoraz častejším riešením. Takéto problémy je možné optimálne riešiť za pomoci trénovania s učiteľom iba ak je cieľ diferencovateľný. Mnoho problémov však týmto spôsobom riešiteľných nie je, kvôli tomu, že neexistuje jednoznačné riešenie. Bežné riešenie je definovanie náhradných stratových funkcií (loss functions), ktoré však môže viesť k riešeniam, ktoré nie sú optimálne vzhľadom na reálny cieľ.

Reinforcement learning, RL („posilňované učenie“) sa v posledných rokoch ukázal ako sľubná alternatíva optimalizácie hlbokých neurónových sietí na maximalizáciu nediferencovaných cieľov. Medzi príklady použitia RL patrí napr. riešenie zložitých herných problémov, generovanie kódu, detekcia objektov alebo riadenie. RL sa od ostatných metód strojového učenia líši niekoľkými vlastnosťami, napr. tým, že údaje používané na trénovanie agenta sú zhromažďované prostredníctvom interakcií s prostredím samotným agentom (na rozdiel od učenia s učiteľom, kde je pevne stanovený súbor údajov).

RL metódy sú však časovo náročné a často prezentované len v teoretickej rovine [33]. V tejto kapitole hlbšie rozoberieme vybrané algoritmy založené na RL, postup ich použitia a možnosť implementácie v riadení zložitých, silno nelineárnych, ale aj jednoduchších mechatronických problémov.

V oblasti RL neurónových sietí a teórie riadenia sa používa rôzna terminológia na označenie konceptov, ktoré sú však vo svojej podstate príbuzné. Súvislosti medzi základnými pojmami sú stručne zhrnuté v tabuľke č.1 [34]:

Tabuľka 1: Základné pojmy RL

Teória riadenia	RL Neurónové siete
Regulátor	Politika (Policy)
Riadený systém, so všetkými signálmi ako napríklad šum, filtre, odchýlky, oneskorenie.	Prostredie (Environment)
Merateľné hodnoty, ktoré môže algoritmus využiť pri riadení.	Pozorovanie (Observation)
Akčný (riadiaci) zásah	Akcia (Action)
Funkcia merania, chybového signálu alebo inej metriky výkonnosti	Odmena (Reward)
Adaptačný mechanizmus adaptívneho regulátora	Algoritmus učenia (Learning algorithm)

1.1 Algoritmy na báze RL

V tejto kapitole hlbšie rozoberieme vybrané algoritmy založené na reinforcement learningu, postup ich použitia a možnosť implementácie v riadení zložitých, silno nelineárnych, ale aj jednoduchších mechatronických problémov.

Algoritmy RL zväčša obsahujú dve zložky: politiku a mechanizmus, ktorý politiku aktualizuje. Politika slúži na výber akcií, na základe pozorovaní minulých stavov prostredia. Mechanizmus aktualizácie politiky zase priebežne aktualizuje jej parametre na základe akcií, pozorovaní a odmien. Jeho cieľom je nájsť optimálnu politiku, ktorá maximalizuje kumulatívnu odmenu počas riešenia úlohy.

Základný algoritmus RL je Q-learning. Je založený na princípe hľadania optimálnej politiky a výbere akcií pre akýkoľvek konečný Markovov rozhodovací proces (MDP). Agent sa učí a maximalizovať celkovú odmenu v

čase prostredníctvom opakovaných interakcií s prostredím, aj keď model tohto prostredia nie je známy. Aktualizáciu stavu Q je možné zapísať:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big(r_t + \gamma\, max_a Q\big(s_{(t+1)}, a\big) - Q(s_t, a_t)\big), \quad (1.1)$$

kde $Q(s_t, a_t)$ je hodnota v tabuľke pre vykonanie akcie at v starom stave s_t, Q^{new} je nová vypočítaná hodnota, ktorou sa nahrádza stará hodnota $Q(s_t, a_t)$, $max_a Q\big(s_{(t+1)}, a\big)$ je najvyššia hodnota v novom stave $s_{(t+1)}$, α je rýchlosť učenia (*learning rate*), $0 < \alpha \leq 1$, ktorá určuje, do akej miery sa stará hodnota nahradí novou; r_t označuje odmenu získanú za prechod do nového stavu a γ je koeficient „zľavy“ (*discount rate*), $0 < \gamma \leq 1$, ktorá určuje dôležitosť celkových odmien dosiahnutých v budúcnosti. Pri hodnote $\gamma = 0$ bude agent brať do úvahy len aktuálne získanú odmenu bez ohľadu na budúce odmeny. Optimálna hodnota je γ je medzi hodnotami 0,9 a 0,99.

Rôzne algoritmy využívajú rôzne mechanizmy na maximalizáciu odmien a sú navrhované na prácu v spojitých, alebo diskrétnych akčných priestoroch pripadne v obidvoch. V diskrétnom akčnom priestore môže do riadeného systému vstupovať len konečná množina akcií (napr. zapni/vypni motor). V spojitom akčnom priestore môže do riadeného systému vstúpiť celá spojitá množina hodnôt (napr. $0 \leq a \leq 1$) [33].

V závislosti od typu použitého agenta si jeho politika a algoritmus učenia vyžadujú jednu alebo viac reprezentácií politiky a hodnotovej funkcie, implementovateľných pomocou hlbokých neurónových sietí. V nasledujúcich kapitolách budú použité aj tieto typy reprezentácií hodnotových funkcií a politík [34]:

$V(S \mid \theta_V)$ - Critic, ktorý odhaduje očakávanú kumulatívnu dlhodobú odmenu (hodnotovú funkciu) na základe daného pozorovania *S*.

$Q(S, A \mid \theta_Q)$ - Critic, ktorý odhaduje hodnotovú funkciu pre danú diskrétnu akciu *A* a dané pozorovanie *S*.

$Q_i(S, A_i \mid \theta_Q)$ - Critic s viacerými výstupmi, ktorý odhaduje hodnotovú funkciu pre všetky možné diskrétne akcie *Ai* a dané pozorovanie *S*.

$\mu(S \mid \theta_\mu)$ - Actor, ktorý vyberá akciu na základe daného pozorovania *S*. Actor môže vyberať akcie pomocou deterministických alebo stochastických metód.

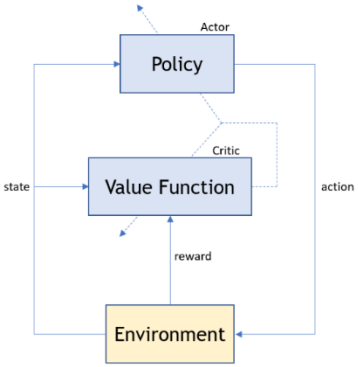
1.2 Actor-Critic (AC)

AC algoritmus pozostáva z dvoch neurónových sietí a kombinuje výhody sietí, ktoré pracujú výlučne len s neurónovými sieťami typu actor alebo critic (obr. 1). Zatiaľ čo parametrizovaný actor má výhodu v tom, že dokáže vypočítať spojité akcie bez potreby optimalizácie value (hodnotovej) funkcie a generuje akcie na základe aktuálneho stavu, critic má výhodu v tom, že poskytuje actorovi úzky okruh informácií o výkone a vyhodnocuje hodnotu aktuálnej dvojice stav-akcia generovanej actorom [35].

Critic je funkcia hodnoty stavu (*state value function*), ktorá sa aktualizuje po každej akcii minimalizáciou strednej kvadratickej chyby medzi odhadovanou a skutočnou hodnotou. Tento proces je vyjadrený pomocou časovej diferencie (*temporal difference*) δ_t [35]–[37]:

$$\delta_t = R_{t+1} + \gamma V_t(S_{t+1}) - V(S_t) \tag{1.2}$$

kde V_t je hodnotová funkcia získaná kritikom v čase t, Rt+1 je ďalšia očakávaná odmena, $S_{(t+1)}$ je očakávaný ďalší stav a γ je *discount-rate* koeficient. Ak je časová diferencia $\delta_t > 0$, pravdepodobnosť výberu akcie At do budúcnosti sa zvyšuje a naopak, ak je $\delta_t < 0$, pravdepodobnosť výberu akcie At sa znižuje. Na základe toho actor priebežne aktualizuje svoju politiku [36], [38].



Obrázok 1: Actor-Critic architektúra [35]

Pravidlá učenia pre sieť actor a sieť critic používajú rovnaký signál δ, ktorý však ovplyvňuje proces učenia v oboch sieťach odlišne. Chyba δ usmerňuje actora, ako aktualizovať pravdepodobnosti akcií tak, aby mohol dosiahnuť hodnotnejšie stavy. Učenie actora je podobné inštrumentálnemu podmieňovaniu pomocou pravidla učenia typu Law-of-Effect: actor sa snaží udržať δ v čo najvyšších pozitívnych hodnotách, pričom δ poskytuje sieti critic informácie o parametroch (smere a veľkosti) value funkcie, ktoré treba upraviť tak, aby sa zvýšila jej predpovedaná presnosť. Critic sa snaží minimalizovať veľkosť chyby na hodnotu čo najbližšiu k nule [38].

1.2 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) algoritmus využíva metodiku hlbokého RL a patrí do skupiny takzvaných model-free („bezmodelových“) algoritmov. Od pôvodného AC sa odlišuje entropickou regularizáciou, ktorej cieľom je maximalizovať očakávanú odmenu pri maximalizácii entropie. To znamená, že actor je podnecovaný vyberať nasledujúcu akciu z rozmanitejšej množiny akcií, čo vedie k efektívnejšiemu skúmaniu prostredia ako aj zlepšeniam

v dlhodobom horizonte a zabraňuje konvergencii do nesprávneho lokálneho optima.

SAC algoritmus sa učí pomerne efektívne, z hľadiska vzoriek je stabilnejší než AC. Vylepšenie fázy skúmania umožňuje sieti na báze SAC ľahko sa adaptovať aj na komplikované, rozsiahle problémy a znižuje možnú nestabilitu spojenú s aproximačným odvodzovaním skorších tzv. off-policy Q-learning algoritmov s maximalizáciou entropie [39].

Entropia je veličina, ktorá charakterizuje „náhodnosť náhodnej premennej“ [39]. Ak *x* je náhodná premenná s funkciou hustoty pravdepodobnosti *P* (opisuje pravdepodobnosť výskytu rôznych hodnôt náhodnej premennej v spojitých rozdeleniach), jej entropia *H* sa vypočíta nasledovne [40]:

$$\mathrm{H(P)} = \mathrm{E_{x\sim P}[-logP(x)]} \tag{1.3}$$

Základný princíp entropicky regularizovaného RL spočíva v tom, že v každom časovom kroku agent získava doplnkovú odmenu úmernú entropii politiky v danej vzorke [40]:

$$\pi = argmax_{\pi} E_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\left(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\circ\,|s_t)\right)\right)\right], \tag{1.4}$$

kde $\alpha > 0$ je *trade-off* koeficient („koeficient kompromisu“), γ je *discount rate* a R sú odmeny. Bellmanova rovnica pre Q^{π} je:

$$Q^{\pi}(s,a) = E_{s'\sim P}[R(s,a,s') + \gamma V\pi(s')] \tag{1.5}$$

Algoritmus SAC je efektívnejší ako jeho predchodcovia hlavne vďaka implementácií entropie, ktorá podnecuje sieť skúmať, a tým sa efektívnejšie učiť aj z menšieho počtu vzoriek [40].

1.3 Proximal Policy Optimization (PPO)

Existujú dva druhy PPO algoritmu, a to PPO-penalty a PPO-clip. V nasledujúcej kapitole sa zameriame na PPO-clip algoritmus, pretože ho budeme využívať neskôr v praktických častiach. PPO je robustný a výkonný algoritmus založený na Trust Region Policy Optimization (TRPO) metóde, ktorá aktualizuje politiku maximalizáciou výkonu, a zároveň zabezpečuje, aby sa nová politika príliš neodchyľovala od starej pomocou obmedzenia založeného na Kullback-Leibler-divergencii (KL divergencia), čo pomáha predchádzať jej veľkým destabilizujúcim aktualizáciám. Metóda TRPO je však veľmi komplikovaná, striktná a výpočtovo náročná, kým PPO algoritmus používa orezanú účelovú funkciu (objective function), ktorá len aproximuje TRPO obmedzenia.

Ak označíme $r_t(\theta) = \frac{\pi_\theta(a_t,s_t)}{\pi_{\theta old}(a_t,s_t)}$, $r_t(\theta_{old}) = 1$ pravdepodobnostný pomer, potom TRPO algoritmus maximalizuje „náhradný“ cieľ:

$$L^{CPI}(\theta) = \hat{E}_t\left[\frac{\pi_\theta(a_t,s_t)}{\pi_{\theta old}(a_t,s_t)}\hat{A}_t\right] = \hat{E}_t\left[r_t(\theta)\hat{A}_t\right], \tag{1.5}$$

kde π je funkcia politiky, ktorá mapuje stavy *s* na akcie *a* na základe parametra θ, ktorý predstavuje váhy neurónovej siete; $\hat{A}_t$ reprezentuje advantage function („funkciu výhod“) v čase t, ktorá hovorí, ako výhodná/nevýhodná je daná akcia v porovnaní s priemernou akciou v danom stave. Horný index CPI odkazuje na iteráciu konzervatívnej politiky (conservative policy iteration), kde bol cieľ navrhnutý. Bez obmedzenia by

maximalizácia L^{CPI} viedla k príliš veľkej politike, preto autori [41] upravili cieľ tak, aby sa penalizovali zmeny politiky, ktoré vzďaľujú $r_t(\theta)$ od 1. Hlavný cieľ je nasledujúci [41]:

$$L^{CLIP}(\theta) = \hat{E}_t\left[min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right], \qquad (1.6)$$

kde ε je hyperparameter ($\varepsilon = 0{,}2$[41]). Cieľová funkcia v PPO algoritme je navrhnutá tak, aby zabezpečila stabilné aktualizácie politiky. Rovnica 1.5 L^{CPI} predstavuje štandardný náhradný cieľ, zatiaľ čo rovnica 1.6 $clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t$, upravuje pravdepodobnostný pomer tak, aby neprekročil interval $[1-\varepsilon, 1+\varepsilon]$. Tým, že PPO algoritmus využíva minimum z orezaného aj neorezaného cieľa, účinne vytvára dolnú hranicu ("pesimistickú väzbu") na neorezaný cieľ. Taktiež v okolí θ_{old} platí $L^{CLIP}(\theta) = L^{CPI}(\theta)$ do prvého rádu (t. j. pre r = 1), avšak s rastúcou vzdialenosťou θ od θ_{old} sa ich hodnoty líšia [41].

Ak uvažujeme advantage funkciu $\hat{A}_t$ kladnú, potom:

$$L^{CLIP}(\theta) = min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}, 1+\varepsilon\right)\hat{A}_t \qquad (1.7)$$

Keďže $\hat{A}_t$ je kladné, cieľ sa zvýši, ak sa akcia stane pravdepodobnejšou, teda ak sa zvýši $\pi_\theta(a_t|s_t)$. Akonáhle $\pi_\theta(a_t|s_t) > 1+\varepsilon\,\pi_{\theta old}(a_t|s_t)$, začína platiť minimum a tento člen narazí na strop $(1+\varepsilon)\hat{A}_t$, čo znamená že nová politika nemá prospech z toho, že sa vzdiali od starej politiky. Rovnaká analógia platí aj pre výraz:

$$L^{CLIP}(\theta) = max\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}, 1-\varepsilon\right)\hat{A}_t \qquad (1.8)$$

Orezávanie slúži ako regularizátor tým, že odstraňuje podnety na výrazné zmeny politiky, hyperparameter ε zodpovedá tomu, ako ďaleko sa môže nová politika vzdialiť od starej, pri zachovaní efektivity vzhľadom na cieľ. Aj keď je tento druh orezávania účinný, nie je dokonalý a stále sa môže stať, že sa nová politika priveľmi vzdiali od starej. V [42] využili pomerne jednoduchú metódu skorého zastavenia: ak priemerná KL-odchýlka novej politiky narastie od starej nad zvolenú prahovú hodnotu, sieť prestane robiť gradientové kroky; tréning zastane.
PPO algoritmus trénuje stochastickú politiku on-policy spôsobom. To znamená, že definuje pravdepodobnostné rozdelenie možných akcií namiesto deterministického výberu jednej akcie. Ako prebieha tréning, PPO používa toto rozdelenie pravdepodobnosti na výber akcií podľa poslednej verzie svojej stochastickej politiky. Miera náhodnosti pri výbere akcií závisí od počiatočných podmienok aj od procesu trénovania. V priebehu tréningu sa politika zvyčajne stáva menej náhodnou, pretože je stimulovaná k využívaniu odmien v stavoch, ktoré už našla. To môže spôsobiť, že pri veľmi zložitých problémoch politika uviazne v lokálnom optime.

Každý z vyššie uvedených algoritmov je využiteľný pre určitý typ úloh. Pred implementáciou daného algoritmu na reálnom zariadení, je vhodné ho overiť na simulačnom modeli. Výhody simulačných modelov spočívajú hlavne v jednoduchosti testovania širokej škály algoritmov v pomerne krátkom čase. Virtuálny model je schopný vyhodnocovať niekoľko krokov či procesov paralelne a taktiež ním vieme overiť, či výstupy zo siete sú korektné a neuvedú reálny systém do stavov, ktoré budú viesť k jeho opotrebeniu alebo zničeniu.

V tejto kapitole sme uviedli iba niekoľko algoritmov, menovite SAC a PPO ktoré sa ukázali ako najvhodnejšie pre riadenie vzduchovej levitácie a AC,

z ktorého vychádza viacero algoritmov metodiky RL. Analýza širšieho spektra relevantných algoritmov je uvedená v 3. kapitole dizertačnej práce.

2. Návrh simulačného modelu v prostredí OpenAI Gymnasium

OpenAI Gymnasium je platforma vyvinutá na prácu s neurónovými sieťami. Platforma je založená na knižnici Pygame, ktorú podporujú mnohé knižnice s algoritmami pre reinforcement learning a dajú sa pomocou nej vytvárať prostredia, ktoré simulujú mechatronické problémy, ale aj rôzne hry. Na stránke [43] je k dispozícii niekoľko hotových prostredí z oblasti riadenia, napríklad kyvadlo, inverzné kyvadlo, inverzné dvojité kyvadlo a pod.

Nami navrhované prostredie, simulačný model, vychádza z laboratórneho zariadenia FloatShield (FS), ktoré predstavuje inovatívny systém vzduchovej levitácie určený na výučbu v oblasti riadenia a mechatroniky. Napriek svojej jednoduchej konštrukcii FS vykazuje niekoľko výziev v oblasti riadenia, a to nielen z dôvodu jeho nelineárnej dynamiky [26], [28], [32].

Prostredie v OpenAI Gymnasium je špecifikovaná trieda, ktorá musí spĺňať dané parametre. V prvom rade je nutné definovať akčný priestor (*action space*), čo je rozsah, v akom je prostredie schopné prijímať akcie.

$$a \in R \quad \text{a} \quad 0 \leq a \leq 100\ [\%] \tag{2.1}$$

Ďalším parametrom je priestor sledovaných hodnôt (*observation space*), čo je rozsah hodnôt, ktoré sú sledovateľné prostredím (výstupy). V prípade laboratórneho modelu FloatShield sme špecifikovali nasledovné intervaly:

$$h \in R \quad \text{and} \quad 0 \leq h \leq 1, \tag{2.2}$$

$$v_{ball} \in R \quad \text{and} \quad -2 \leq v_b \leq 2, \tag{2.3}$$

$$p_{fan} \in R \quad \text{and} \quad 30 \leq fp \leq 100, \tag{2.4}$$

$$v_{\text{air}} \in \text{R} \quad \text{and} \quad -10 \leq v_{\text{a}} \leq 10, \tag{2.5}$$

kde h je výška guličky v percentách rozsahu trubice, pričom 0 reprezentuje dno trubice a 1 jej horný okraj, v_{ball} je rýchlosť loptičky [*m/s*], P_{fan} je výkon ventilátora v rozsahu <30 - 100> [%], kde spodná hodnota je zvolená ako minimálna možná hodnota výkonu ventilátora, ktorá vybudí motor. Veličina v_{air} je rýchlosť prúdenia vzduchu v trubici [m/s].

Pre správnu funkcionalitu Gymnasium prostredia je nutné definovať aj metódy pre túto triedu, ktoré majú isté konvencie pomenovania: prostredie musí obsahovať metódy *step, reset, render* a *close*.

V metóde *step* sa vykonávajú výpočty a úkony, ktoré vykoná prostredie za jednotku času. Vstupným parametrom metódy *step* musí byť akcia, ktorú má agent v prostredí vykonať za jednotku času. Výstupom tejto metódy sú informácie o stave prostredia, odmena, ktorú agent získa za vykonanú akciu, informácia, či simulácia pokračuje a ďalšie nepovinné informácie, napríklad logy. V prípade nami vytvoreného prostredia táto metóda najskôr skontroluje, či je vstupná akcia valídna. Následne sa vypočíta rýchlosť guličky:

$$v_{ball}(t) = v_{0\,ball} \; + \; a \; dt \tag{2.6}$$

a analogicky poloha guličky:

$$h(t) = h_0 \; + \; v_{ball} \; dt \tag{2.7}$$

Okrajové podmienky pre guličku sa aplikujú tak, aby simulácia bola uvažovaná len pre oblasť trubice.

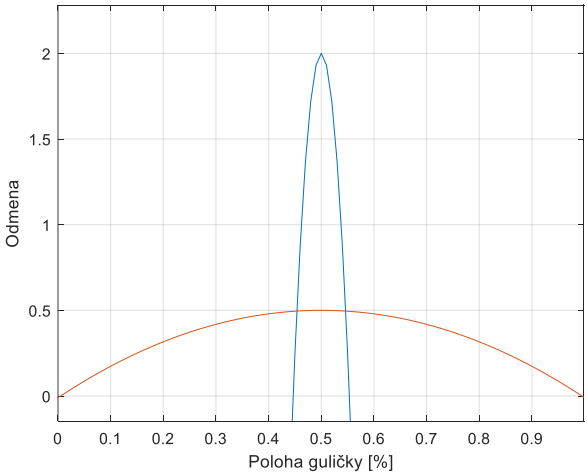
Posledným krokom funkcie *step* je udelenie odmeny za vykonanú akciu. Udeľovanie správnej odmeny je netriviálna úloha a pri návrhu prostredia bolo nutné vytvoriť funkciu odmeňovania iteračne.

Po viacerých experimentoch, ten najúspešnejší experiment využíval funkciu odmeňovania v tvare prieniku dvoch kvadratických funkcií r_1a r_2, a to „úzkej“, ktorá pokrýva odmeny v tesnej blízkosti žiadanej hodnoty, a „širokej“, ktorá ponúka čiastkové odmeny v širšej oblasti okolo žiadnej hodnoty a relevantnú negatívnu odmenu pre veľké odchýlky od žiadanej hodnoty obr.2.

$$r_1(e) = -7\left(\frac{e}{0.1}\right)^2 + 2 \tag{2.8}$$

$$r_2(e) = -\left(\frac{e}{0.7}\right)^2 + 0.5 \tag{2.9}$$

$$r(e) = \max\left(r_1(e), r_2(e)\right) \tag{2.10}$$



Obrázok 2: Prienik ostrej a plytkej kvadratickej funkcie odmeny

Aby neurónová sieť mala motiváciu zohľadňovať rôzne stavy, implementovali sme do funkcie odmeňovania funkciu skorého zastavenia, ktorá sa aktivuje vtedy, keď guličkа neopustila dolnú alebo vrchnú hranicu trubice počas niekoľkých po sebe idúcich krokov (5 pre simuláciu, 30 pre

experiment na laboratórnom modeli). Ak sa funkcia skorého zastavenia spustila, od nadobudnutej odmeny sa odčíta 50 bodov. Takto tvarovaná funkcia odmeňovania dokázala naviesť RL algoritmy k úspešnému riešeniu simulačného prostredia.

Metóda *reset* sa využíva na iniciálne spustenie prostredia do valídneho stavu, pričom musí byť zabezpečené, aby sa tento stav nachádzal v simulovateľnom intervale. Väčšinou sa požaduje, aby boli stavy náhodné a aby sa sieť nenaučila len jeden scenár riadenia, ale aby hľadala možné riešenia z rôznych počiatočných podmienok. Vstupnými parametrami tejto metódy sú aj *seed*, ktorý umožňuje reprodukovateľnosť výsledkov, a *options*, ktorý umožní modifikovať inicializáciu prostredia. Výstupom metódy *reset* je informácia o stave, v akom sa prostredie nachádza v počiatočných podmienkach a nepovinné logy.

Metóda *render* je čisto vizualizačná pomôcka a štandardne prebieha paralelne s behom systému. Dobre navrhnutá vizualizačná metóda pomáha pri opravovaní chýb v prostredí, ale aj pri prezentácií riešenia problémov. Neodporúča sa však nechať bežať simuláciu paralelne s trénovaním siete či už na reálnom zariadení alebo len vo virtuálnom prostredí, lebo táto metóda značne spomaľuje celkový beh programu. Vizualizuje sa každý jeden krok, ktorý prostredie vytvorí. Odporúča sa teda analyzovať a vizualizovať len vybrané behy z celej množiny realizovaných behov. Aktivácia a deaktivácia vizualizácie sa vykoná v momente tvorby prostredia pomocou parametra *render_mode*, ktorý má dva módy, a to „human“ a „rgb_array“. V móde „human“ nemá metóda render žiaden programový výstup iba okno, kde prebehne vizualizácia, a v móde „rgb_array“ je výstupom pole pixelov, ktoré reprezentujú vizualizáciu, neprebehne však jej zobrazenie.

Poslednou povinnou metódou je metóda *close*, ktorá ukončí beh prostredia a zavrie všetky okná spojené s vizualizáciou, ak sú nejaké otvorené.

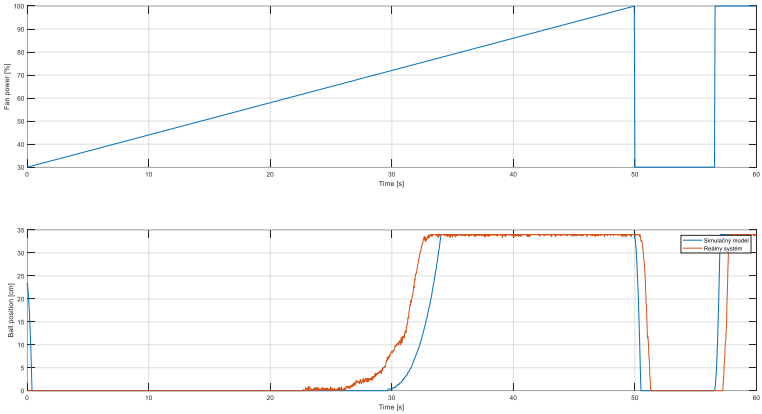
Takto namodelovaným prostredím sme získali ďalší simulačný model FS, ale aj napriek tomu sme narazili na rovnaký problém ako pri simulačnom modeli v programe MATLAB Simulink; simulačný model je citlivejší a začína reagovať o čosi neskôr ako reálny systém.

Presnosť modelu závisí od funkcie sily (2.11) F_f (akčného zásahu), ktorá závisí od výkonu ventilátora [%], preto je pre čo najpresnejšiu reprezentáciu reálneho systému FS nutná ďalšia modifikácia na základe nameraných dát a heuristické ladenie. Po heuristickom doladení je funkcia sily F_f vyjadrená v tvare:

$$F_f = 2.2745e^{-10}u^5 - 6.7203e^{-8}u^4 + 7.6882e^{-6}u^3 - 4.3034e^{-4}u^2 + 0.0123u - 0.1239 \quad (2.11)$$

Porovnanie výstupu zo simulačného prostredia OpenAI Gymnasium a z reálneho zariadenia FS je na obr. 3.

Takto navrhnuté prostredie lepšie zachytáva dynamiku reálneho systému FS a taktiež po drobných úpravách umožní rýchlu komunikáciu s reálnym systémom.



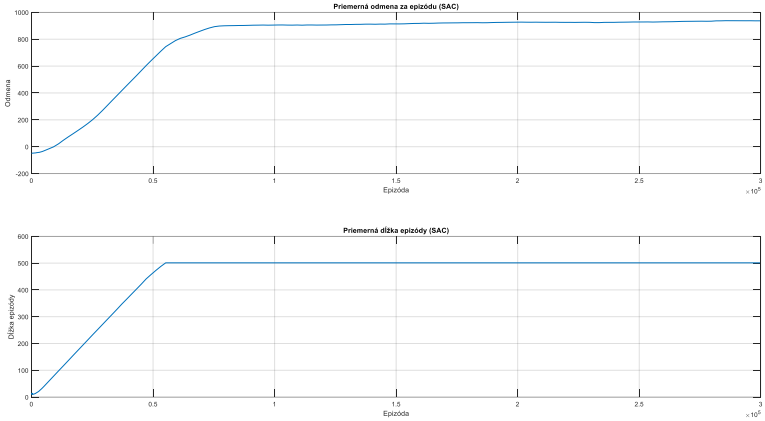
Obrázok 3: Porovnanie výstupu zo simulačného prostredia OpenAI Gymnasium a z reálneho zariadenia FS

V nasledujúcej kapitole sa zameriame na implementáciu riadenia pomocou algoritmov RL na rôznych simulačných modeloch, so zameraním predovšetkým na modely vytvorené v kapitole 5.

3. Simulačná analýza algoritmov

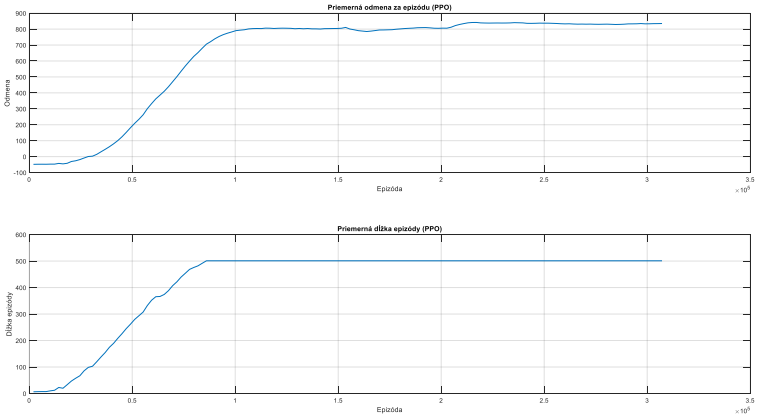
V tejto časti autoreferátu sú algoritmy analyzované na základe vhodne zvoleného prostredia, v ktorom sa testuje ich funkčnosť. Vybrané algoritmy SB3 boli verifikované a analyzované na virtuálnom prostredí z predošlej kapitoly a to nasledovne: tréningový proces každého algoritmu bol spustený desaťkrát, pričom každé spustenie učenia pozostávalo z 300 000 behov s variabilnou dĺžkou. Dĺžka behu závisela od toho, kedy a či vôbec RL algoritmus narazil na podmienku skorého zastavenia. Pre tento experiment boli využité nezmenené hodnoty algoritmov prednastavené v SB3. V autoreferáte spomenieme iba algoritmy SAC a PPO, pretože budú relevantné aj v testoch na reálnom zariadení.

Algoritmus SAC je výkonný algoritmus a v simulácii preukázal veľmi dobré výsledky. Maximálnu priemernú dĺžku epizódy dosiahol už v 55,2k. iterácii a priemerná hodnota odmeny, ktorá najprv rástla prudko, neprestala pomaly narastať ani do konca simulácie, kde sa zastavila na hodnote 937. Aj keď je na grafoch obr. 4 znázornený len najlepší tréningový proces, z dát v prílohe B môžeme vyhodnotiť, že tento algoritmus dosahoval konzistentne dobré výsledky, ktoré sú využiteľné pre riadenie reálneho systému.



Obrázok 4: Trénovací proces algoritmu SAC

Posledným testovaným algoritmom bol PPO. Keďže sa jedná o najmodernejší z testovaných algoritmov, predpokladali sme, že bude aj najefektívnejší. Naprieč všetkými tréningovými cyklami dosahoval dobré hodnoty a maximálnu priemernú dĺžku epizódy 500 krokov, ktorú dosiahol v 86,02k. kroku. Maximálna priemerná odmena bola 841,5 dosiahnutá v 215k. kroku, ktorá sa ku koncu mierne prepadla a skončila na hodnote 835,1 (obr.5). Obe hodnoty sú pomerne vysoké a vďaka konzistencii dosahovania dobrých výsledkov tento algoritmus môžeme považovať za vhodný na riadenie reálneho systému.



Obrázok 5: Trénovací proces algoritmu PPO

V dizertačnej práci, na konci kapitoly 5, sa nachádza prehľadná tabuľka analýzy výsledkov simulácií. Na základe výsledkov týchto simulácií boli pre riadenie laboratórneho modelu zvolené algoritmy SAC a PPO.

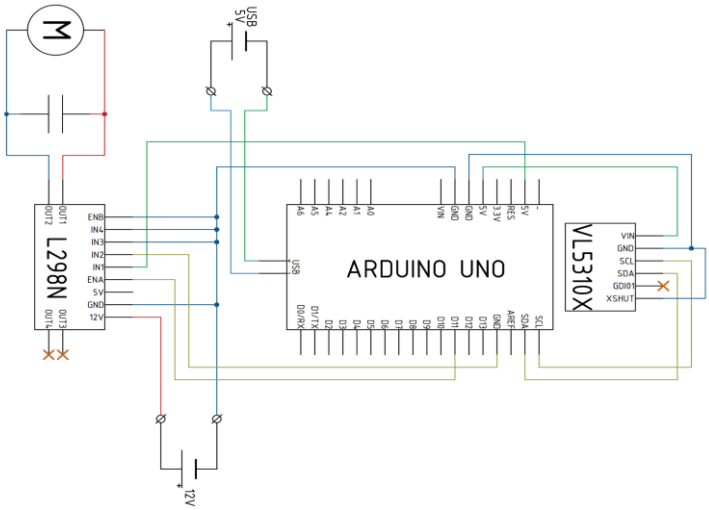
4. Riadenie laboratórneho modelu vzduchovej levitácie

Pre komunikáciu s Arduinom bolo nutné mierne upraviť prostredie v OpenAI Gymnasium. Pôvodné prostredie pracovalo s fixnou vzorkovacou frekvenciou a všetky údaje o sile pôsobiacej na guličku boli získavané z rovnice *Ff* č. 5.23. Tento celý krok bolo pre reálny systém možné vypustiť, pretože údaje o výške guličky boli priamo merateľné. Tieto údaje bolo nutné normalizovať, pretože navrhnutá sieť pracuje dobre s údajmi v intervale <0-1>, ale hodnoty na výstupe senzora sú intervale <65-395>, kde hodnota 65 je moment, kedy je gulička najbližšie k senzoru a naopak. Z týchto normalizovaných dát o polohe guličky je následne počítaná jej približná rýchlosť dosiahnutá medzi dvomi vzorkami:

$$v = \frac{\Delta p}{\Delta t} \tag{3.1}$$

Perióda vzorkovania zberu dát Δt nie je konštantná, pretože úkony, ktoré robí sieť, majú často premenlivú dĺžku a keby sa dĺžka jedného časového kroku určovala fixne, bolo by nutné ju zvoliť podľa najpomalšieho kroku siete, čo by neúmerne zdržovalo celú komunikáciu, ktorá prebieha pomocou Arduino rozhrania.

FS, z ktorého pri návrhu modelu vychádzame, obsahuje množstvo komponentov, ktoré nie sú nevyhnutné pre jeho prevádzku riadenú neurónovou sieťou, ako sú napríklad filtre a digitálny zosilňovač. Navrhnutá adaptácia systému FS, je zjednodušená a plne postačujúca pre riadenie pomocou metód RL. Z pôvodného systému FS bol využitý senzor vzdialenosti, ventilátor, 3D-tlačené komponenty a trubica. Miesto zložitého digitálneho zosilňovača, ktorý sa nachádzal na shielde, bol využitý H-mostík L298N. Nová schéma zapojenia laboratórneho modelu je na obr.6.



Obrázok 6: Schéma zapojenia laboratórneho modelu vzduchovej levitácie

Systém riadenia pomocou spätnej väzby je navrhnutý tak, že komunikácia medzi Arduinom a Pythonom prebieha cez USB kábel. Spätná väzba z Arduina je odosielaná cez USB rozhranie do Pythonu, konkrétne do simulačného prostredia OpenAI Gymnasium. Na druhej strane je akčný zásah generovaný RL ANN odosielaný prostredím rovnakého USB kábla späť do Arduina, kde sa tento signál aplikuje vo forme PWM (Pulse Width Modulation) vstupu do zosilňovača, čím sa riadi výkon ventilátora. Arduino nepretržite číta údaje zo senzora a opäť ich odosiela do prostredia na vyžiadanie. Riadiaci cyklus v Arduine beží na maximálnej možnej frekvencii a prostredie si údaje žiada vtedy, keď ich potrebuje. Prostredie z celého toku dát využíva len tie najaktuálnejšie, pričom tieto údaje predstavujú priemer zo 16 posledných vzoriek vytvorený pomocou bitového posunu. Tento jednoduchý typ filtrácie pomáha eliminovať extrémne hodnoty a stabilizovať kvalitu dát, čím sa znižuje vplyv náhodných šumov alebo výkyvov v meraniach.

Po vytvorení rozhrania bolo nutné mierne prepísať OpenAI Gymnasium prostredie, pretože údaje o pozícii guličky už nebolo potrebné počítať nakoľko boli merané senzorom. V prvom kroku bol na laboratórny model implementovaný agent trénovaný na simulačnom prostredí. Riadenie nebolo úspešné ani pre PPO, ani pre SAC agenta. Počas testovania sa agentom nepodarilo dosiahnuť ani kladné odmeny. Dôvodov môže byť viacero, a to napr.: na rozdiel od reálnej aplikácie, má simulačná schéma presné numerické, nezašumené hodnoty, presnú vzorkovaciu frekvenciu a simulácia vždy čaká, kým sa nedokončia výpočty v sieti a pod.

Pre úspešné riadenie je teda nutné sieť trénovať na reálnom systéme nanovo. Najskôr sme začali experimentovať so SAC algoritmom, keďže bol pri simulačnom modeli úspešnejší. Politika odmeňovania a hyperparametre ostali rovnaké ako pri trénovaní na simulačnom prostredí. Výsledkom bol agent, ktorý dosahoval kumulatívnu odmenu približne 300 (± 50), čo je neúmerne

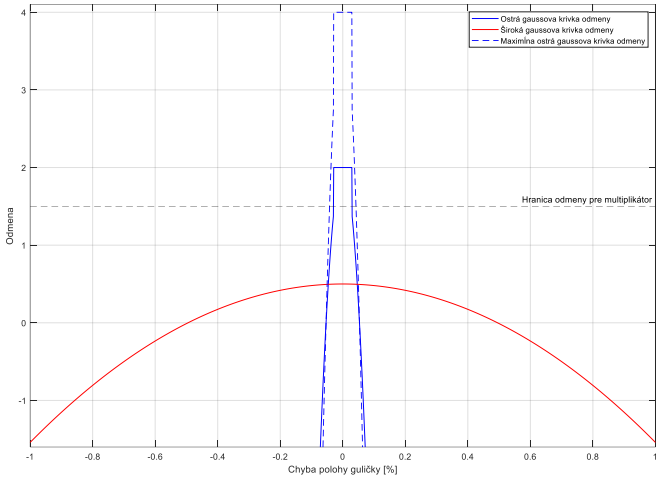
nízke v porovnaní so simulačným modelom (simulačný model dosahoval hodnoty okolo 900). Pri testovaní agenta na širšom spektre žiadaných hodnôt v rozmedzí od 30% výšky do 90% bolo možné pozorovať veľkú amplitúdu oscilácií guličky (± 30% od žiadanej hodnoty polohy). Z hľadiska kvality riadenia sa takéto riešenie nedalo považovať za úspešné.

Zmena správania sa agenta sa dá dosiahnuť najúčinnejšie zmenou odmeňovacej politiky a zmenami hyperparametrov. Pri pozorovaní oscilácií guličky sme predpokladali, že problémy môžu byť práve v odmeňovacej politike. V pôvodnej politike odmeňovania agent dostáva pomerne veľkú negatívnu odmenu za predčasné ukončenie behu. Tento prístup môže spôsobiť jeho priveľkú „opatrnosť" pri explorácii a dokonca si agent nemusí spojiť veľkú negatívnu odmenu s jej reálnou príčinou. Namiesto veľkej jednorazovej zápornej odmeny v nežiadúcom konečnom stave sme v okrajových stavoch guličky od odmeny odrátali hodnotu 1. Takto sa agentovi priamo zvýraznili nežiadúce stavy bez toho, aby spôsobili averziu voči skúmaniu nových stavov. Ďalšie vylepšenie oproti pôvodnej politike odmien bolo zavedenie koeficientu za každý žiadaný stav. Keď opakovane dosiahla odmena hodnotu $r > 1,5$, nasledujúca odmena bola o 10% väčšia. Tento koeficient sa zvyšoval až po 100%, teda maximálna dosiahnuteľná odmena bola 4 (obr. 7). Implementácia koeficientu podporovala stabilitu a motivovala agenta zotrvávať v stavoch vysokej odmeny čo najdlhšie bez prerušenia, čím sa znížila amplitúda oscilácií. Po testovaní zmien v politike odmien sa ukázalo, že agent je schopný riadiť stavy v nižšej polovici spektra žiadaných hodnôt ($u < 0,6$), pri vyšších hodnotách stále dochádzalo k nežiadúcim osciláciám.

$$r_1(e) = \begin{cases} 2, & if\ |e| < 0.03 \\ -7 * \left(\frac{e}{0.1}\right)^2 + 2, & if\ |e| \geq 0.03 \end{cases} \tag{3.2}$$

$$r_1(e) = -\left(\frac{e}{0.7}\right)^2 + 0.5 \tag{3.3}$$

$$r(e) = \max\left(r_1(e), r_2(e)\right) \tag{3.4}$$



Obrázok 7: Funkcia politiky odmeňovania

Zmenami hyperparametrov sa dá dosiahnuť zmena správania sa agenta, otestovali sme preto niekoľko variácií hyperparametrov, popíšeme však len finálnu a doteraz najlepšiu množinu hyperparametrov (hĺbka siete, počet neurónov vo vrstve, rýchlosť učenia, veľkosť vyrovnávacej pamäte, learning starts, batch size, koeficient gamma, SDE vzorkovacia frekvencia).

Siete agenta a kritika boli prehĺbené z dvoch počiatočných vrstiev na tri a zvýšený bol počet neurónov na jednotlivých vrstvách zo 64 na 254. Toto vylepšenie umožňuje sieti učiť sa komplexnejšie príznaky za cenu zvýšenia výpočtového výkonu. Ak je akčný alebo stavový priestor veľmi rozsiahly, širšie vrstvy môžu lepšie zachytiť zložitosť možných stavov alebo akcií.

Hyperparameter, ktorý priamo ovplyvňuje spôsob a mieru učenia agenta, je *learning_rate*, ktorý je pôvodne 0,0003. V našom prípade bol tento parameter zmenšený na 0,0002, čo zabezpečí stabilnejšie učenie, ale ho aj mierne spomalí.

Na to, aby trénovací algoritmus mal dostatočne rôznorodú sadu údajov, slúži parameter *learning_starts*, ktorý sme nastavili na 1000 z pôvodných 0.

Ďalej bol zmenený hyperparameter „*batch_size*“, ktorý reprezentuje počet vzoriek použitých na každú aktualizáciu gradientu. Z pôvodných 256 vzoriek bol zvýšený na 512, čím sa síce zvýšila výpočtová náročnosť učenia, ale toto zvýšenie zabezpečilo stabilitu učenia.

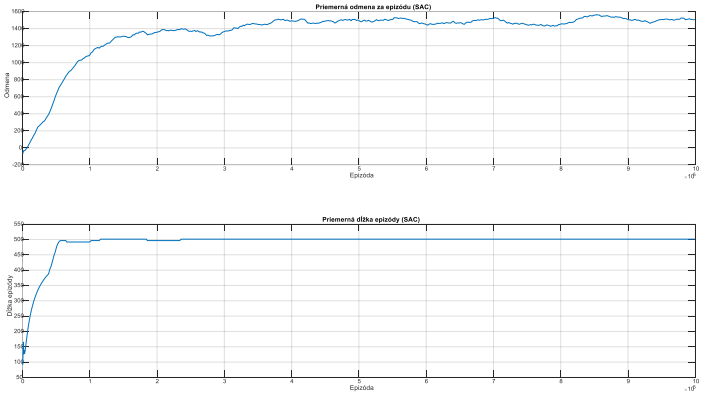
Hyperparameter *gamma* je odporúčané voliť z rozsahu 0.95 – 0.999. Čím väčšia je gamma, tým viac sa bude agent zameriavať na dlhodobý horizont odmien. V našom prípade sme jemne zvýšili tento hyperparameter z 0.99 na 0.992.

Veľmi dôležitý hyperparameter pre trénovanie na reálnom zariadení je *use_sde*, a s ním spätý *sde_sample_freq*. Tieto parametre aktivujú „zovšeobecnený prieskum v závislosti od stavu“ (gSDE - Generalized State-Dependent Exploration), čo je pokročilá stratégia explorácie, ktorá sa používa na zvýšenie efektivity trénovania. Nahrádza totiž klasický, gaussovský vzorkovací šum funkciou šumu závislú od stavu. Tento prístup zabezpečuje, že prechody medzi jednotlivými stavmi sú plynulé a výrazne sa redukujú nepravidelné, trhavé zmeny akcií, čo pri reálnom zariadení môže spôsobiť poškodenie [44].

Agent s takto nastavenou politikou odmeňovania a hyperparametrami bol spoľahlivý v nižších častiach trubice (do 50%), avšak ak žiadaná hodnota dosiahla vyšších hodnôt, gulička začala oscilovať. Amplitúda oscilácií bola úmerná žiadanej hodnote polohy. Tieto problémy pretrvávali aj v prípade, keď bol interval žiadaných hodnôt obmedzený na hodnoty <0,6 – 0,9>. Takéto nestabilné správanie je typické pre systémy s dopravným oneskorením, preto

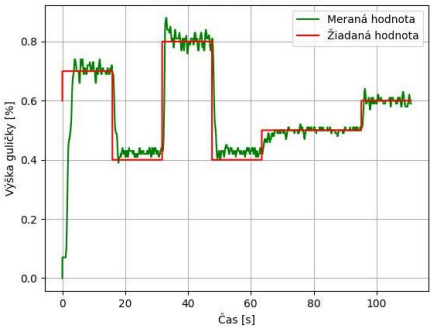
sme aplikovali funkciu z SB3 tzv. *FrameStacking*, ktorý dovoľuje posielať do siete niekoľko minulých stavov prostredia bez toho aby bolo nutné vykonať zmeny v kóde prostredia ako takého. *FrameStacking* teda umožňuje agentovi vidieť predošlé stavy, čo je veľmi cenná informácia, hlavne v nelineárnom systéme s oneskorením, ako je ten náš. Doteraz najlepšie výsledky sme dosiahli pri hodnote *FrameStacking = 30*, čo znamená, že do siete posielame ako pozorovanie aktuálny stav, a tridsať minulých stavov, čo je približne 0,6-0,8s. Najskôr boli vykonané testy na hornom intervale <0,6 – 0,9>, aby sa ušetril čas. Ak by sieť nebola schopná úspešne riadiť najproblematickejší interval žiadaných hodnôt, nemá zmysel ju trénovať na celom intervale. Agent úspešne riadil systém v danom intervale a pomerne dobre reagoval aj na vstupy mimo neho, avšak keďže počas tréningu nikdy za tieto vstupy nedostal odmenu, nedokázal spoľahlivo udržať žiadanú hodnotu $u < 0,6$.

Následne prebehol tréning v celom rozsahu trubice. Takto nastavená sieť dokázala spoľahlivo regulovať výšku guličky dokonca aj pri impulze externej poruchy, ktorú sme zaviedli krátkym obmedzením prívodu vzduchu k ventilátoru. Agent veľmi rýchlo dosiahol maximálnu dĺžku behu a priemerná odmena pomaly stúpala počas celého tréningu obr. 8.



Obrázok 8: Trénovací proces algoritmu SAC na laboratórnom modeli

Na grafe obr. 9 je stále vidno jemné oscilácie okolo žiadanej hodnoty. Tento efekt je spôsobený nerovnosťami povrchu korkovej guličky a tým, že sa gulička odráža od stien trubice a pohybuje sa v žiadanej výške aj vodorovne. Pri súčasnom konštrukčnom vyhotovení laboratórneho modelu sa takejto chybe nebude možné vyhnúť.



Obrázok 9: Odozva na zmeny žiadanej hodnoty (agent na báze SAC)

Po úspešnej implementácii algoritmu SAC sme sa analogicky pokúsili implementovať aj algoritmus PPO. Pre riadenie systému pomocou algoritmu PPO sme využili sme už vylepšenú politiku odmien a znalosti funkcie *FrameStacking*, no ani po experimentoch s niekoľkými sadami hyperparametrov sme nedosiahli uspokojivé riadenie, takže sa potvrdila hypotéza založená na experimente so simulačným modelom, že algoritmus SAC je vhodnejší pre reálny systém. Riadenie pomocou algoritmu PPO bude predmetom ďalšieho výskumu a experimentov.

Záver

Dizertačná práca sa zaoberá návrhom, vývojom a overením moderných inteligentných metód automatického riadenia s využitím algoritmov hlbokého RL učenia pre vybrané mechatronické systémy. Práca rieši kompletný návrh SW a HW prvkov a systémov inteligentného automatického riadenia a real-time implementáciu inteligentných algoritmov pre komplexný nelineárny dynamický systém vzduchovej levitácie.

Dizertačná práca poskytuje prehľad teoretických základov z oblasti umelej inteligencie a neurónových sietí, venuje sa popisu ich rôznych architektúr, spôsobom učenia ako aj aplikáciám v oblasti riadenia dynamických systémov. Osobitný dôraz sa kladie na použitie algoritmov RL, ako napríklad Actor-Critic (AC), Advantage Actor-Critic (A2C) a Soft Actor-Critic (SAC), ktoré sú podrobne analyzované a implementované pre adaptívne riadenie nelineárnych systémov. Aby sa predišlo možnému opotrebovaniu alebo poškodeniu laboratórneho zariadenia, boli na trénovanie ANN regulátora vytvorené jeho simulačné modely, a to dvoma spôsobmi: v prostredí Matlab-Simulink a následne, pomocou freameworku OpenaAI Gymnasium v programovacom jazyku Python. Druhý spôsob sa ukázal ako efektívnejší a rýchlejší, pretože umožňuje rýchlu komunikáciu s neurónovou sieťou a paralelné trénovanie niekoľkých sietí naraz.

Na základe analýzy boli vyšpecifikované vhodné algoritmy RL, ktoré boli trénované na simulačnom modeli. Výsledky simulácií a následných experimentov na reálnom modeli preukázali, že navrhované inteligentné pokročilé metódy dokážu efektívne riešiť aj iné všeobecné náročné úlohy riadenia so spojitým akčným priestorom, ako je napríklad stabilizácia polohy obráteného dvojitého kyvadla alebo riadenie polohy guličky vo vytvorenom unikátnom systéme vzduchovej levitácie. Prvý navrhovaný algoritmus SAC po optimalizácií vynikal rýchlou konvergenciou k stabilným riešeniam, poskytoval robustnú a efektívnu politiku. Druhý algoritmus PPO

však nedosahoval uspokojivé výsledky aj napriek snahe o optimalizáciu odmeňovacej politiky či hyperparametrov.

Na základe simulácií a experimentov je možné konštatovať, že použité algoritmy RL (SAC a PPO) sú efektívne a dokážu zabezpečiť vysokú kvalitu riadenia a stabilitu a rýchlu konvergenciu riešenia v komplexných úlohách automatického riadenia zložitých nelineárnych systémov v reálnom čase pre širokú škálu aplikácií riadenia mechatronických procesov so zložitou nelineárnou dynamikou.

Dosiahnuté výsledky experimentov na laboratórnom systéme vzduchovej levitácie naznačujú vysokú úspešnosť navrhovanej metodiky na báze algoritmov AI a potvrdzujú tak trendy využívania moderných metód strojového učenia v širokom spektre mechatronických aplikácií.

Predložené riešenie umožňuje spustiť riadenie s deterministickým agentom, ktorý sa snaží vždy dosiahnuť maximálnu možnú odmenu v každom stave, čím zabezpečuje kvalitné riadenie, avšak na úkor jeho adaptability. Naopak, nedeterministický agent umožňuje neustále učenie sa a adaptáciu na zmeny prostredia, no za cenu zníženej kvality riadenia, keďže skúmaním prostredia nemusí vždy preferovať najväčšiu odmenu.

Dizertačná práca svojou štruktúrou, komplexnosťou a dosiahnutými výsledkami predstavuje významný prínos v porovnaní s existujúcimi riešeniami vo svete v oblasti vzduchových levitačných systémov. Práca prináša inovatívne riešenia tak v oblasti algoritmických postupov ako aj v oblasti vývoja unikátneho nízkonákladového riadiaceho systému, vrátane návrhu inteligentných senzorov a real-time komunikačných systémov.

Na záver možno konštatovať, že predložená dizertačná práca prispieva k vývoju metód umelej inteligencie a jej uplatneniu v reálnych zložitých nelineárnych systémoch riadenia. Výsledky dizertačnej práce sú súčasťou riešenia úloh v rámci výskumných projektov APVV a VEGA.

Hlavným prínosom dizertačnej práce je vytvorené inovatívne riešenie unikátneho nízkonákladového riadiaceho systému na báze algoritmov RL vrátane návrhu inteligentných senzorov a real-time komunikačných systémov a algoritmických postupov.

Čiastkové prínosy zahŕňajú:

- Vytvorenie simulačného modelu vzduchovej levitácie v prostredí OpenAI Gymnasium. Simulačný model veľmi presne opisuje správanie sa nelineárneho reálneho laboratórneho systému FloatShield a umožňuje rýchlu a priamu komunikáciu s komponentmi ANN a SB3 a taktiež jednoduché ladenie (debugovanie) na rozdiel od simulačného modelu vytvoreného v prostredí Matlab Simulink.
- Vytvorenie jednoduchého rozhrania medzi prostredím Python a Arduino, ktoré umožňuje ich rýchlu komunikáciu.
- Vytvorenie prostredia na báze OpenAI Gymnasium, ktoré dokáže komunikovať s ANN regulátorom a reálnym zariadením.
  V rámci jeho adaptácie sa vynechá výpočet veličín, ktoré je možné merať. Pri HW riešení je potrebné vykonať normalizáciu a filtráciu vstupných dát.
- Testovanie viacerých relevantných, moderných algoritmov a analýza výsledkov.
- Vytvorenie nového laboratórneho modelu vzduchovej levitácie, pripraveného na riadenie pomocou ANN, ktorý je schopný komunikovať s programovacím prostredím Python.
- Vytvorenie metodiky postupu návrhu pre riadenie levitačných systémov pomocou RL algoritmov

Literatúra

[1] Š. Kozák, "State-of-the-art in control engineering," *J. Electr. Syst. Inf. Technol.*, vol. 1, no. 1, 2014, doi: 10.1016/j.jesit.2014.03.002.

[2] Maruti, "AI and ML made simple," 2016. Accessed: Feb. 11, 2024. [Online]. Available: https://www.marutitech.com/artificial-intelligence-and-machine-learning/

[3] S. Benkrama and N. E. H. Hemdani, "Deep Learning with EfficientNetB1 for detecting brain tumors in MRI images," 2023. doi: 10.1109/ICAECCS56710.2023.10104761.

[4] H. A. Taha, M. K. Othman, N. E. Abbas, Y. K. Sayed, H. H. Ammr, and R. Shalaby, "Modeling of Nonlinear Enhanced Air Levitation System using NARX Neural Networks," 2021. doi: 10.1109/NILES53778.2021.9600486.

[5] A. Perrusquía and W. Yu, "Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview," *Neurocomputing*, vol. 438, 2021, doi: 10.1016/j.neucom.2021.01.096.

[6] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46. 2018. doi: 10.1016/j.arcontrol.2018.09.005.

[7] H. nan Wang *et al.*, "Deep reinforcement learning: a survey," *Frontiers of Information Technology and Electronic Engineering*, vol. 21, no. 12. 2020. doi: 10.1631/FITEE.1900533.

[8] S. Cao, L. Sun, J. Jiang, and Z. Zuo, "Reinforcement Learning-Based Fixed-Time Trajectory Tracking Control for Uncertain Robotic Manipulators with Input Saturation," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 8, 2023, doi: 10.1109/TNNLS.2021.3116713.

[9] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, 2023, doi: 10.1038/s41586-023-06419-4.

[10] Z. Bing, D. Lerch, K. Huang, and A. Knoll, "Meta-Reinforcement Learning in Non-Stationary and Dynamic Environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, 2023, doi: 10.1109/TPAMI.2022.3185549.

[11] Q. Shen, P. Shi, J. Zhu, S. Wang, and Y. Shi, "Neural Networks-Based Distributed Adaptive Control of Nonlinear Multiagent Systems," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 3, 2020, doi: 10.1109/TNNLS.2019.2915376.

[12] L. Liu, Y. J. Liu, and S. Tong, "Neural networks-based adaptive finite-time fault-tolerant control for a class of strict-feedback switched nonlinear systems," *IEEE Trans. Cybern.*, vol. 49, no. 7, 2019, doi: 10.1109/TCYB.2018.2828308.

[13] M. Hillebrand, M. Lakhani, and R. Dumitrescu, "A design methodology for deep reinforcement learning in autonomous systems," in *Procedia Manufacturing*, 2020, vol. 52. doi: 10.1016/j.promfg.2020.11.044.

[14] N. Ragheb and M. M. A. Mahmoud, "Implementing Deep Reinforcement Learning in Autonomous Control Systems," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 41, no. 1, 2024, doi: 10.37934/araset.41.1.168178.

[15] Z. Zhang, Z. Mo, Y. Chen, and J. Huang, "Reinforcement Learning Behavioral Control for Nonlinear Autonomous System," *IEEE/CAA J. Autom. Sin.*, vol. 9, no. 9, 2022, doi: 10.1109/JAS.2022.105797.

[16] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan, "Reinforcement learning control of constrained dynamic systems with uniformly

ultimate boundedness stability guarantee," *Automatica*, vol. 129, 2021, doi: 10.1016/j.automatica.2021.109689.

[17] C. Park, C. Jeong, J. Yoo, and C. M. Kang, "Efficient Reinforcement Learning Method for Dynamic System Control," *Trans. Korean Inst. Electr. Eng.*, vol. 71, no. 9, 2022, doi: 10.5370/KIEE.2022.71.9.1293.

[18] B. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," *Math. Financ.*, vol. 33, no. 3, 2023, doi: 10.1111/mafi.12382.

[19] A. Charpentier, R. Élie, and C. Remlinger, "Reinforcement Learning in Economics and Finance," *Comput. Econ.*, vol. 62, no. 1, 2023, doi: 10.1007/s10614-021-10119-4.

[20] S. H. Oh, S. J. Lee, and J. Park, "Precision Medicine for Hypertension Patients with Type 2 Diabetes via Reinforcement Learning," *J. Pers. Med.*, vol. 12, no. 1, 2022, doi: 10.3390/jpm12010087.

[21] S. H. Oh, S. J. Lee, and J. Park, "Effective data-driven precision medicine by cluster-applied deep reinforcement learning," *Knowledge-Based Syst.*, vol. 256, 2022, doi: 10.1016/j.knosys.2022.109877.

[22] R. K. Tan, Y. Liu, and L. Xie, "Reinforcement learning for systems pharmacology-oriented and personalized drug design," *Expert Opinion on Drug Discovery*, vol. 17, no. 8. 2022. doi: 10.1080/17460441.2022.2072288.

[23] M. T. Hagan and H. B. Demuth, "Neural Networks for Control," 1999. doi: 10.1109/ACC.1999.786109.

[24] K. Cheon, J. Kim, M. Hamadache, and D. Lee, "On Replacing PID Controller with Deep Learning Controller for DC Motor System," *J. Autom. Control Eng.*, vol. 3, no. 6, 2015, doi:

10.12720/joace.3.6.452-456.

[25] K. El Hamidi, M. Mjahed, A. El Kari, and H. Ayad, “Adaptive control using neural networks and approximate models for nonlinear dynamic systems,” *Model. Simul. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/8642915.

[26] P. Chmurčiak, “FloatShield: An Educational Device for Air Levitation Experiments.,” Slovak University of Technology in Bratislava, 2020.

[27] T. Tkáčik, M. Tkáčik, S. Jadlovská, and A. Jadlovská, “Design of aerodynamic ball levitation laboratory plant,” *Processes*, vol. 9, no. 11, 2021, doi: 10.3390/pr9111950.

[28] D. Rosinova and R. Schwarz, “Teaching State-Space Control Methods Using a Floatshield Laboratory Plant,” 2023. doi: 10.1109/PC58330.2023.10217680.

[29] A. Tootchi, S. Amirkhani, and A. Chaibakhsh, “Modeling and Control of an Air Levitation Ball and Pipe Laboratory Setup,” 2019. doi: 10.1109/ICRoM48714.2019.9071827.

[30] W. Hu, Y. Zhou, Z. Zhang, and H. Fujita, “Model Predictive Control for Hybrid Levitation Systems of Maglev Trains with State Constraints,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, 2021, doi: 10.1109/TVT.2021.3110133.

[31] D. Chaos, J. Chacón, E. Aranda-Escolástico, and S. Dormido, “Robust switched control of an air levitation system with minimum sensing,” *ISA Trans.*, vol. 96, 2020, doi: 10.1016/j.isatra.2019.06.020.

[32] G. Takács *et al.*, “FloatShield: An Open Source Air Levitation Device for Control Engineering Education,” in *IFAC-PapersOnLine*, 2020, vol. 53, no. 2. doi: 10.1016/j.ifacol.2020.12.1807.

[33] B. Jaeger and A. Geiger, “An invitation to deep reinforcement

learning.," 2023.

[34] I. The MathWorks, "Reinforcement Learning for Control Systems Applications." Accessed: Apr. 1, 2024. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/reinforcement-learning-for-control-systems-applications.html

[35] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," *IEEE Trans. Neural Networks*, vol. 9, no. 5, 1998, doi: 10.1109/tnn.1998.712192.

[36] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuška, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 6. 2012. doi: 10.1109/TSMCC.2012.2218595.

[37] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, 1988, doi: 10.1007/bf00115009.

[38] R. S. Sutton and A. G. Barto, "Reinforcement learning : an introduction 2nd (19 June, 2017)," *Neural Networks IEEE Trans.*, vol. 9, no. 5, 2017.

[39] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *35th International Conference on Machine Learning, ICML 2018*, 2018, vol. 5.

[40] J. Achiam, "Spinning Up in Deep RL," *OpenAI Spinning Up*, 2018.

[41] S. John, W. Filip, D. Prafulla, R. Alec, and K. Oleg, "Proximal Policy Optimization Algorithms," 2017.

[42] "Proximal Policy Optimization," 2021. Accessed: Apr. 7, 2024. [Online]. Available:

https://spinningup.openai.com/en/latest/algorithms/ppo.html

[43] "OpenAI Gymnasium homepage", [Online]. Available: Accessed: Apr. 11, 2024. [Online]. Available: https://gymnasium.farama.org/index.html

[44] A. Raffin, J. Kober, and F. Stulp, "Smooth Exploration for Robotic Reinforcement Learning," in *Proceedings of Machine Learning Research*, 2021, vol. 164.

Vlastná publikačná činnosť

V2 Vedecký výstup publikačnej činnosti ako časť editovanej knihy alebo zborníka

V2_01 HAFFNER, Oto [42 %] - KUČERA, Erik [42 %] - URMINSKÁ, Barbora [6 %] - HLAVATÝ, Michal [5 %] - DRAHOŠ, Peter [5 %]. Lemna minor bioassay evaluation using computer vision. In QUAERE 2018 [elektronický zdroj] : Roč. VIII : Interdisciplinární mezinárodní vědecká konference doktorandů a odborných asistentů. Hradec Králové, Česká republika. 27.-29. června 2018. Hradec Králové : Magnanimitas, 2018, CD-ROM, S. 561-567. ISBN 978-80-87952-26-9.

Kategória publikácie do 2021: AFC

V2_02 HLAVATÝ, Michal [45 %] - KOZÁKOVÁ, Alena [45 %] - ROSINOVÁ, Danica [10 %]. Efficient fuzzy control of a laboratory ABS. In 2018 Cybernetics & Informatics (K&I) [elektronický zdroj] : 29th International Conference. Lazy pod Makytou, Slovakia. January 31-February 3, 2018. 1. vyd. Bratislava : Slovak Chemical Library, 2018, USB, [7] s. ISBN 978-1-5386-4420-1. V databáze:

IEEE: 8337558 ; DOI: 10.1109/CYBERI.2018.8337558 ; WOS: 000454633500029 ; SCOPUS: 2-s2.0-85050916948.

Kategória publikácie do 2021: AFD

Ohlasy:

1. [1] JAIN, Himani - MAITREYA, Shreyas - PALIWAL, Priyanka. Cooperative Control of Regenerative and Anti-lock Braking Systems in Electric Vehicles Using Fuzzy Logic. In 2021 4th International Conference on Recent Developments in Control, Automation and Power Engineering, RDCAPE 2021, 2021, pp. 62-66. ISBN 978-166541429-6., Registrované v: SCOPUS

Ohlas: zahraničný

V2_03 HLAVATÝ, Michal [55 %] - KOZÁKOVÁ, Alena [5 %] - HAFFNER, Oto [40 %]. Application for python programming language education developed by unity engine. In 2022 Cybernetics & Informatics (K&I) [elektronický zdroj] : Proceedings ; 31st International Conference; 11-14 September 2022 Visegrád, Maďarsko. 1. vydanie. Danvers, Massachusetts, USA : IEEE, 2022, [6] s. ISBN 978-1-6654-8775-7. V databáze: DOI: 10.1109/KI55792.2022.9925955 ; SCOPUS: 2-s2.0-85142052861 ; IEEE: 9925955.

Typ výstupu: príspevok z podujatia; Výstup: zahraničný; Kategória publikácie do 2021: AFC

Ohlasy:

1. [1] SHAHBAZ, Muhammad - ALTAF, Ayesha - IQBAL, Faiza - SHOAIB, Shazia. Smart and Advanced E-learning Methodology with IoT Device Integration. In: Proceedings 2023 6th International Conference of Women in Data Science at Prince Sultan University,

WiDS-PSU 2023, 2023, pp. 217-222. ISBN 978-166547723-9., Registrované v: SCOPUS

Ohlas: zahraničný

2. [1] GROSU, Mihai Alexandru - BRUMAR, Raul - NICOLA, Stelian - CIOCÂRLIE, Horia. Trigger Systems: Reusability and applicability in Serious Games. In: 2023 27th International Conference on System Theory, Control and Computing, ICSTCC 2023 Proceedings, 2023, pp. 106-110. ISBN 979-835033798-3., Registrované v: SCOPUS

Ohlas: zahraničný

V2_04 HLAVATÝ, Michal [90 %] - KOZÁKOVÁ, Alena [10 %]. Development of Advanced Control Strategy Based on Soft Actor-Critic Algorithm. In COMSCI 2023 : 11th International Scientific Conference on Computer Science. Sozopol, Bulgaria. September 18-20, 2023. Danvers : IEEE, 2023, [5] s. ISBN 979-8-3503-2525-6. V databáze: DOI: 10.1109/COMSCI59259.2023.10315824 ; IEEE: 10315824 ; SCOPUS: 2-s2.0-85186317880.

Typ výstupu: príspevok z podujatia; Výstup: zahraničný; Kategória publikácie do 2021: AFC

V2_05 MINÁR, Martin [25 %] - HLAVATÝ, Michal [25 %] - ROSINOVÁ, Danica [25 %] - KOZÁKOVÁ, Alena [25 %]. Design, realization and modeling of a laboratory heat exchanger. In ELITECH´18 [elektronický zdroj] : 20th Conference of doctoral students. Bratislava, Slovakia. May 23, 2018. 1. ed. Bratislava : Vydavateľstvo Spektrum STU, 2018, CD-ROM, [7] p. ISBN 978-80-227-4794-3.

Kategória publikácie do 2021: AFD

V3 Vedecký výstup publikačnej činnosti z časopisu,

V3_01 HLAVATÝ, Michal - KOZÁKOVÁ, Alena - HAFFNER, Oto. Application for python programming language education developed by unity engine. In Information Technology Applications : 31st International Conference on Cybernetics & Informatics (K&I). Visegrád, Maďarsko. 11-14 September 2022. Vol. 11, No. 1 (2022), s. 3-9. ISSN 1338-6468. Typ výstupu: článok; Výstup: zahraničný; Kategória publikácie do 2021: ADF

O2 Odborný výstup publikačnej činnosti ako časť knižnej publikácie alebo zborníka,

O2_01 HLAVATÝ, Michal [95 %] - KOZÁKOVÁ, Alena [5 %]. Reinforcement learning and possibilities of its use in mechatronics. In ELITECH´22 [elektronický zdroj] : 24th Conference of Doctoral Students. Bratislava, Slovakia. June 1, 2022. 1. ed. Bratislava : Vydavateľstvo Spektrum STU, 2022, [4] s. ISBN 978-80-227-5192-6. Typ výstupu: príspevok z podujatia; Výstup: domáci; Kategória publikácie do 2021: BEF

Štatistika: kategória publikačnej činnosti od 2022

V2	Vedecký výstup publikačnej činnosti ako časť editovanej knihy alebo zborníka	5
V3	Vedecký výstup publikačnej činnosti z časopisu	1

O2	Odborný výstup publikačnej činnosti ako časť knižnej publikácie alebo zborníka	1
Súčet		**7**

Štatistika: kategória ohlasov od 2022

1 Citácia v publikácii registrovaná v citačných indexoch				**3**
	Zahraničné		3	
Súčet				**3**

Zdroje metrík: Clarivate, Scopus a Journalmetrics