TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Řízení pneumatické pružiny neuronovou sítí

Diplomová práce

Studijní program: N0715A270020 Aplikovaná mechanika
Autor práce: **Bc. Jiří Rágulík**
Vedoucí práce: Ing. Michal Sivčák, Ph.D.
Katedra mechaniky, pružnosti a pevnosti

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Zadání diplomové práce

Řízení pneumatické pružiny neuronovou sítí

Jméno a příjmení: **Bc. Jiří Rágulík**
Osobní číslo: S20000232
Studijní program: N0715A270020 Aplikovaná mechanika
Zadávající katedra: Katedra mechaniky, pružnosti a pevnosti
Akademický rok: **2020/2021**

Zásady pro vypracování:

Vytvořte matematický model pneumatické pružiny řízené pomocí neuronové sítě. Pokuste se realizovat praktický experiment s využitím vývojových modulů (Raspberry Pi, Arduino). Porovnejte výsledky s PID regulací.

Rozsah grafických prací:
Rozsah pracovní zprávy: 40
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

Slavík, J., Stejskal, V., Zeman, V. Základy dynamiky strojů, Praha, Vydavatelství ČVUT, 1997, ISBN 80-01-01622-6 Tůma, J. Zpracovaní signálů získaných z mechanických systémů, Praha, Sdělovací technika Praha, 1997, ISBN 80-901936-1-7

Vedoucí práce: Ing. Michal Sivčák, Ph.D.
Katedra mechaniky, pružnosti a pevnosti

Datum zadání práce: 31. října 2020
Předpokládaný termín odevzdání: 30. dubna 2022

L.S.

prof. Dr. Ing. Petr Lenfeld
děkan

doc. Ing. Iva Petríková, Ph.D.
vedoucí katedry

V Liberci dne 31. října 2020

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

10. ledna 2021 Bc. Jiří Rágulík

Abstrakt

Práce se zabývá nahrazením PID regulátoru pokročilejší metodou regulace určenou k řízení zdvihu pneumatické pružiny. K tomuto účelu zvolené algoritmy hlubokého posíleného učení jsou v této práci teoreticky popsány a následně aplikovány na matematický model pružiny. Dále se práce zabývá konstrukcí experimentálního zařízení určeného k řízení reálné pružiny.

Klíčová slova: pneumatická pružina, PID regulace, umělá inteligence, hluboké posílené učení, neuronové sítě, odměnová funkce, Deep Deterministic Policy Gradient

Abstract

The work deals with the replacement of the PID controller with a more advanced method of regulation designed to control the stroke of a pneumatic spring. The algorithms of deep reinforcement learning selected for this purpose are theoretically described in this work and subsequently applied to the mathematical model of the spring. Furthermore, the work deals with the construction of an experimental device designed to control a real spring.

Keywords: air spring, PID regulation, artificial inteligence, deep reinforcement learning, neural networks, reward function, Deep Deterministic Policy Gradient

Poděkování

Tímto bych chtěl poděkovat Ing. Michalu Sivčákovi, Ph.D. za jeho rady, věnovaný čas a ochotu při konzultacích. Mé poděkování patří též Bc. Lubomíru Sivčákovi, za pomoc při realizaci měření v laboratořích KMP.

Dále bych chtěl poděkovat své rodině za podporu, kterou mi poskytuje.

Tato práce byla podpořena Studentskou grantovou soutěží Technické univerzity v Liberci v rámci projektu č. SGS-2019-5072.

Vznik této práce byl podpořen projektem "Pokročilé evropské sedačky pro globální inovaci v automobilovém sektoru" (LTE12004) financovaným Ministerstvem školství, mládeže a tělovýchovy.

Obsah

Seznam zkratek a symbolů

a	akce
b	bias
c	reálná konstanta
d	ukazatel dosažení terminačního stavu
E	očekávaná hodnota
EB	překročení limitů
f	frekvence
L	ztrátová funkce
N	šum
p	přetlak
Q	Q-hodnota
R	návratová hodnota
r_t	odměna
r_{te}	odměna za minimalizaci odchylky
r_{tp}	penalizace za příliš rychlou změnu tlaku
r_{tt}	penalizace za terminaci
s	stav
S	aktivační funkce
V	hodnotová funkce
w	váhy umělých neuronových sítí
x	vstupy neuronových sítí
γ	discount factor
μ	deterministická strategie
π	stochastická strategie
ρ_0	úvodní distribuce
τ	trajektorie
φ	parametry sítě

1. Úvod

Tato diplomová práce se zabývá nahrazením PID regulátoru použitého v bakalářské práci, na kterou tato práce navazuje, který řídí zdvih vlnovcové pneumatické pružiny, pokročilejším a sofistikovanějším způsobem regulace. Pro tyto účely byl vybrán algoritmus umělé inteligence, konkrétně hlubokého posíleného učení.

V úvodních kapitolách je vysvětlena motivace k použití těchto poměrně komplikovaných algoritmů a je popsána současná úroveň poznání v oboru.

V následujících teoretických kapitolách jsou popsány umělé hluboké neuronové sítě, které představují jednu z nejpodstatnějších součástí algoritmů. V dalších kapitolách této práce je popsáno fungování algoritmů hlubokého posíleného učení a jsou vysvětleny jejich principy. Následuje popis fungování konkrétního vybraného algoritmu Deep Deterministic Policy Gradient.

V poslední části této diplomové práce je popsáno sepsání algoritmu v softwaru Matlab a jeho aplikace na matematický model vlnovcové pneumatické pružiny v softwaru Simulink [1], včetně porovnání s PID regulací. Je zde také nastíněna aplikace algoritmu na fyzickou pružinu. Pro tyto účely byl navržen a sestaven experimentální přípravek a byl navržen způsob komunikace algoritmu běžícím na počítači v softwaru Matlab s laserovým snímačem délky a tlakovým regulátorem.

2. Motivace

Hlavním důvodem pro použití těchto algoritmů je předpokládaný průnik umělé inteligence, konkrétně strojového učení, do naprosté většiny oborů lidské činnosti. Mezi nejzásadnější a nejpřínosnější aplikace algoritmů umělé inteligence patří aplikace ve zdravotnictví, kde algoritmy zajišťují práci s lékařskými záznamy, přípravou nových léčiv a vyhodnocování různých lékařských metod. Další aplikací jsou autonomní řízení vozidel silniční dopravy, robotika, finančnictví a kyber-bezpečnost. Dalším příkladem aplikace algoritmů umělé inteligence je například aplikace společnosti Disney, která algoritmy umělé inteligence využívá k předpovídání oblíbenosti vydávaných filmů.

Mezi dvě skupiny strojového učení, které se v současné době těší největší pozornosti, jsou zpracování přirozeného jazyka a posílené učení. Tato práce se zabývá algoritmy posíleného učení, zpracování přirozeného jazyka se věnovat nebude.

Mezi nejznámější aplikace posíleného učení patří algoritmy hrající deskové a počítačové hry. V roce 1997 porazil počítač DeepBlue společnosti IBM Garri Kasparova ve hře šachy. V roce 2016 potom počítač AlphaGo společnosti DeepMind porazil držitele 18 mezinárodních titulů Lee Se-dola ve hře go [2]. Hra go je považována za mnohem komplexnější hru, než jsou šachy a experti v dané oblasti předpovídali, že porazit nejlepšího hráče go na světě algoritmus nedokáže. Algoritmus porazil Lee Se-dola a prokázal značnou kreativitu hned v několika tazích. Ani jeden z těchto kreativních tahů nebyl součástí dat, ze kterých se algoritmus učil a nikdy je ani neprovedl ve hrách, které odehrál sám se sebou v rámci učení. O rok později potom porazil nejlepšího hráče go na světě, Kche Tie.

Velikost trhu s algoritmy umělé inteligence dosáhne v roce 2021 dle odhadů uvedených v článku [3] dosáhne přes 22 miliard amerických dolarů, přičemž software představuje přibližně třetinu trhu s umělou inteligencí, přibližně polovinu tvoří služby a zbytek hardware zaměřený na umělou inteligenci. Je rovněž v následujících letech očekáván růst o přibližně 50 % meziročně.

3. Současná úroveň poznání v oboru

Publikace [4] obsahuje kompletní teoretické poznatky o posíleném učení, od definování základního problému posíleného učení, jakožto markovského rozhodovacího procesu, dynamické programování, metody Monte-Carlo, aproximační gradientní metody, až po aplikace a souvislosti v psychologii a neurovědách. Aproximací funkcí se zabývá článek [5].

Kompletní shrnutí teoretických poznatků před implementací hlubokých neuronových sítí, která byla umožněna především nárůstem výpočetního výkonu, dává publikace [6]. S příchodem hlubokých neuronových sítí do posíleného učení vzniká hluboké posílené učení, jehož aplikace, konkrétně hluboké Q-učení, na jednoduchých počítačových hrách je představena v [7].

[8] pokládá teoretický základ všem algoritmům využívajícím deterministické strategie a popisuje výhody oproti strategiím stochastickým, především pak sníženou výpočetní náročnost. Právě snížení výpočetní náročnosti umožnilo aplikaci algoritmů na problémy se spojitým akčním prostorem.

[9] je publikace sepsaná lidmi z Google Deepmind a představuje algoritmus DDPG. Ten na základě úspěchů s hlubokým Q-učením a předchozího článku [8] dává dohromady zmíněný algoritmus, který je vhodný pro velmi komplexní problémy se spojitým akčním prostorem.

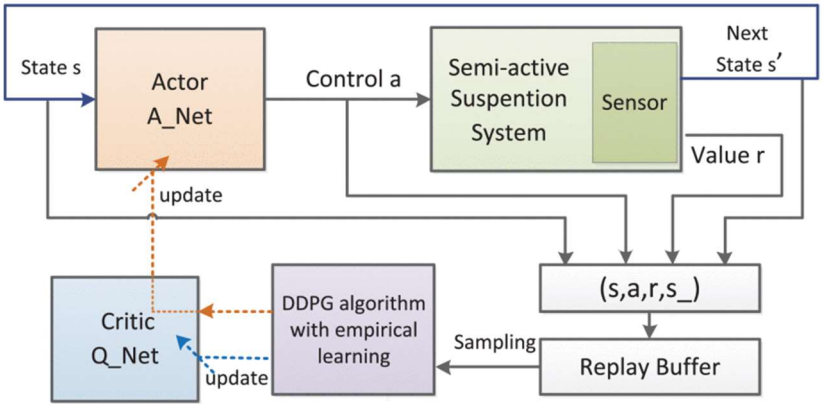
Pro řízení dynamických mechanických soustav by byla potřebná vysoká jemnost diskretizace akčního prostoru, což by vedlo k vysoké výpočtové náročnosti.

Publikace [10] pojednává o aplikaci různých algoritmů posíleného učení, včetně DDPG, na problémy se spojitým akčním prostorem. Rovněž se podrobně zabývá porovnáním dosažených výsledků jednotlivými algoritmy. V článku je též popsána aplikace algoritmů na 20 modelových prostředí.

Článek [11] řeší modelové situace, kdy algoritmus DDPG nekonverguje k optimálnímu řešení, což je autory řešeno úpravou odměnové funkce, která v algoritmu funguje jako zpětná vazba. Jsou zde popsány také další problémy s algoritmem, především pak jeho nestabilita, způsobená přílišnou citlivostí na hyper-parametry a ukazuje i modernější algoritmy a to, jakým způsobem příslušné problémy řeší.

Příspěvek [12] se zabývá aplikací hlubokého posíleného učení na mechanické systémy, zde konkrétně na dron. Je zde popsána konstrukce algoritmu upraveného pro dynamický mechanický systém. Dále ukazuje a porovnává výsledky dosažené algoritmy DDPG a Trust Region Policy Optimization (TRPO), o kterém se dá mluvit jako o nástupci DDPG.

Článek [13] obsahuje přehled metod využívaných pro řízení systému odpružení tlumičem a pružinou, od PID a LQR regulátorů, až po řízení neuronovou sítí a hlubokým posíleným učením. Dále jsou zde shrnuty klady a zápory jednotlivých přístupů a je zde popsána vhodnost algoritmů hlubokého posíleného učení, především díky schopnosti generalizace komplexní situace, která ještě nikdy předtím nebyla algoritmem pozorována. Jsou popsány provedené simulace a je pojednáno o možnostech praktické aplikace. V článku vytvořený regulační obvod je vyobrazen na obrázku (Obr. 1)

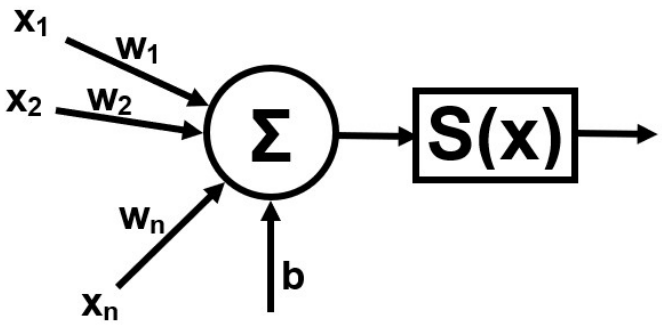


Obrázek 1: Schéma regulačního obvodu semi-aktivního odpružení

4. Neuronové sítě

V posledních několika letech zásadně vzrostl zájem o umělé neuronové sítě. Jedná se o velice zjednodušené matematické modely nervových systémů živých organismů. Jeden směr výzkumu v oblasti neuronových sítí se snaží modelovat lidský mozek a procesy v něm. Druhý směr, inspirovaný neurofyziologickými poznatky, tyto modely modifikují, aby se daly využít pro řešení úloh umělé inteligence. Umělé neuronové sítě při použití vykazují prvky podobné lidské inteligenci, především pak schopnost se učit a generalizovat předchozí zkušenosti.

Prvním takovým modelem je matematický model, představený v [14]. Jedná se o zjednodušený model neuronu (Obr. 2). Důležité je však empirické zjištění, že neuronové sítě složené z takovýchto neuronů, mohou počítat aritmetické a logické funkce.



Obrázek 2: Model umělého neuronu

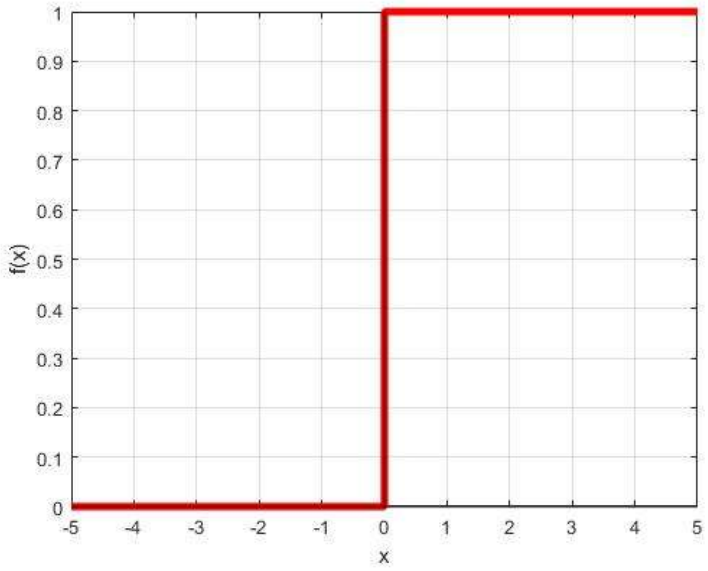
Matematický model neuronu má n vstupů x_n, kde každý z nich je vynásoben vahou w_n, čímž je dána propustnost vstupů. Tyto vynásobené vstupy jsou sečteny a je k nim přičtena ještě prahová hodnota b (bias). Nelineární nárůst výstupní hodnoty při dosažení prahové hodnoty potenciálu je dán aktivační funkcí $S(x)$.

4.1 Aktivační funkce

Výstup neuronu může nabývat hodnot od mínus nekonečna do nekonečna. Sám neuron nemá žádné omezení. Pro rozhodnutí, zda má být neuron aktivován či nikoli, je použita aktivační (někdy také přenosová) funkce. Aktivační funkce může být následujících typů:

4.1.1. Binární skoková funkce

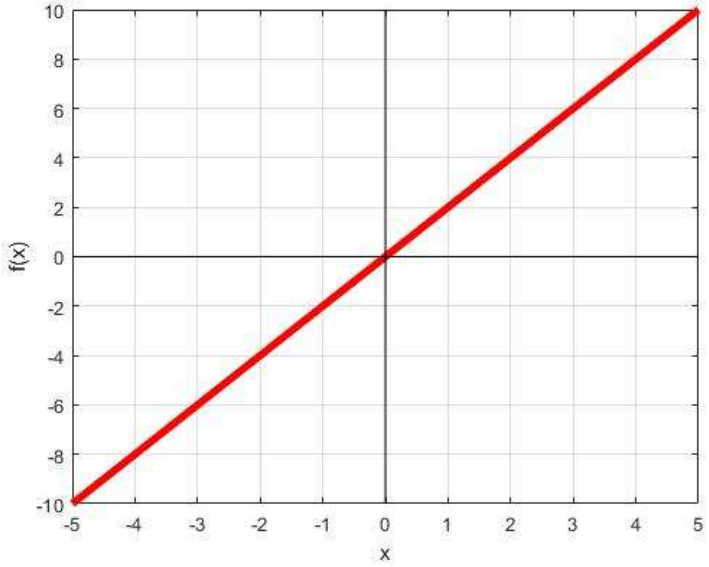
Nejjednodušší aktivační funkce, kde, pokud není překročen určitý práh, výstupní hodnota je 0, při překročení prahu je výstupem 1. Tato aktivační funkce je vhodná například pro binární klasifikátor. V případě, kdy je potřeba, aby výstupem byla určitá pravděpodobnost aktivace neuronu, již vhodná není. Takový příklad je například klasifikátor, který však vybírá z více možností. Pro takové případy je vhodnější například lineární aktivační funkce. Na (Obr. 3) je průběh této aktivační funkce.



Obrázek 3: Binární skoková aktivační funkce

4.1.2. Lineární funkce

Výstup lineární aktivační funkce je proporcionální vůči vstupu. Výhodou oproti skokové funkci je možnost nebinárního výstupu. Zásadním problémem této aktivační funkce je však nemožnost použít zpětnou propagaci, kterou se provádí učení sítě. Derivací lineární funkce je totiž konstanta, která není závislá na vstupu, není tedy možné určit, které provedené zásahy byly nejlepší. Další nevýhodou je splynutí všech vrstev neuronové sítě do jedné. Každá vrstva bude lineárně závislá na předchozích vrstvách, čímž prakticky přicházíme o možnost přidávání vrstev neuronové sítě. Neuronová síť s touto aktivační funkcí je v podstatě lineárně regresní model. Její průběh je na (Obr. 4) a její funkční předpis je dán vztahem (1).



Obrázek 4: Lineární aktivační funkce

$$f(x) = c \cdot x \tag{1}$$

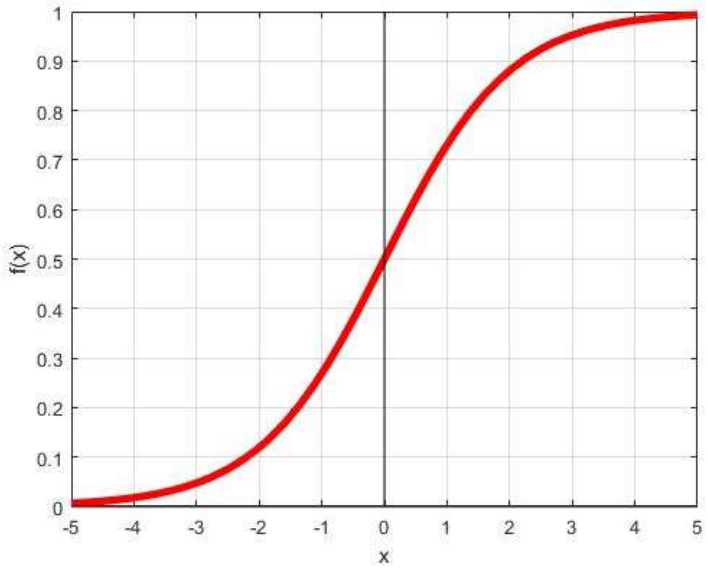
4.1.3. Nelineární funkce

Nelineární aktivační funkce umožňují komplexní mapování mezi vstupy a výstupy, proto pro komplexní složité úlohy téměř výhradně používáme nelineární aktivační funkce. Umožňují totiž využití zpětné propagace, protože jejich derivace je závislá na vstupu. S jejich požitím také můžeme používat vícevrstvé neuronové sítě, které jsou nutné pro komplexní úlohy, protože jednoduše nesplynou s ostatními vrstvami. Nejpoužívanější nelineární aktivační funkce jsou:

a) Sigmoidální (logistická) funkce

Sigmoidální aktivační funkce řeší hlavní problém funkcí lineárních a lze přidávat další vrstvy neuronů, protože jejich kombinace jsou určitě nelineární. Umožňují také non-binární aktivaci. Další nespornou výhodou je fakt, že výstup bude v rozsahu (0, 1), oproti rozsahu lineární aktivační funkce (-∞, ∞).

Z průběhu funkce na obrázku (Obr. 4) je patrný hlavní nedostatek a to tzv. mizející gradient (vanishing gradient problem), kdy pro x nižší než -2 a vyšší než 2 nenastává prakticky žádná změna. To může vést k velice dlouhým výpočetním časům. Průběh funkce je na (Obr. 5) a její funkční předpis je dán vztahem (2).

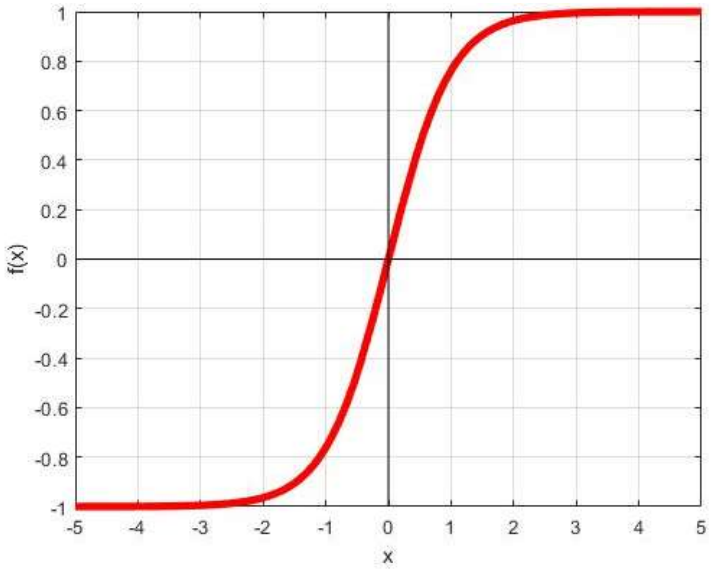


Obrázek 5: Sigmoidální aktivační funkce

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

b) Hyperbolický tangens (tanh)

Hyperbolický tangens a sigmoida jsou nápadně podobné funkce. Tanh funkce je však strmější, což má vliv na derivaci. Její hlavní výhodou oproti sigmoidální funkce je souměrnost podle $f(x) = 0$. Nevýhody tanh aktivační funkce jsou stejné, jako u funkce sigmoidální. Průběh funkce je na (Obr. 6) a její funkční předpis je dán vztahem (3).

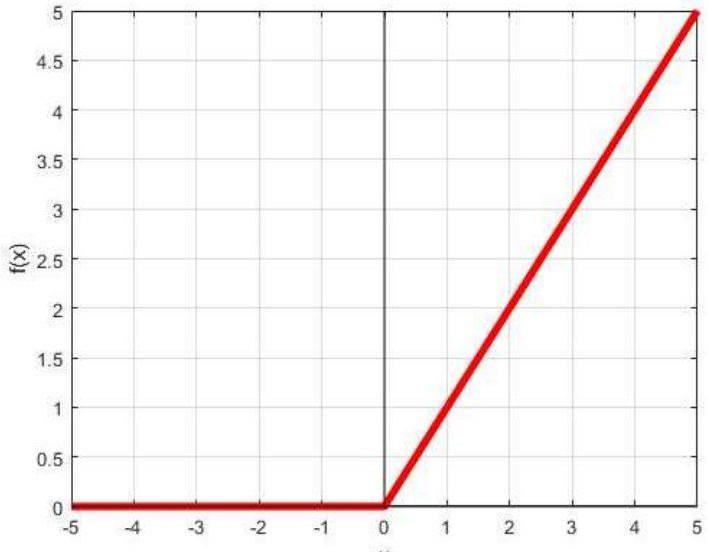


Obrázek 6: tanh aktivační funkce

$$f(x) = \tanh(x) \tag{3}$$

c) Rectified Linear Unit funkce (ReLU)

U této funkce nastává konstantní derivace, jako u lineární funkce. Její výhodou je fakt, že při záporném vstup je nulový výstup a neuron tedy není aktivován. Při neaktivování více neuronů tak dojde k úspoře výpočetního výkonu a proces učení probíhá rychleji. Každá funkce může být aproximována kombinací ReLU. Nevýhodou však je fakt, že v záporné oblasti je gradient nulový a váhy tak nebudou přenastaveny. Tento problém je v odborné literatuře popsán jako umírající ReLU (dying ReLU problem). Neurony, které projdou tímto stavem, přestávají reagovat a část sítě tak může zůstat pasivní. Tento problém řeší následující aktivační funkce. Průběh funkce je na (Obr. 7) a její funkční předpis je dán vztahem (4).

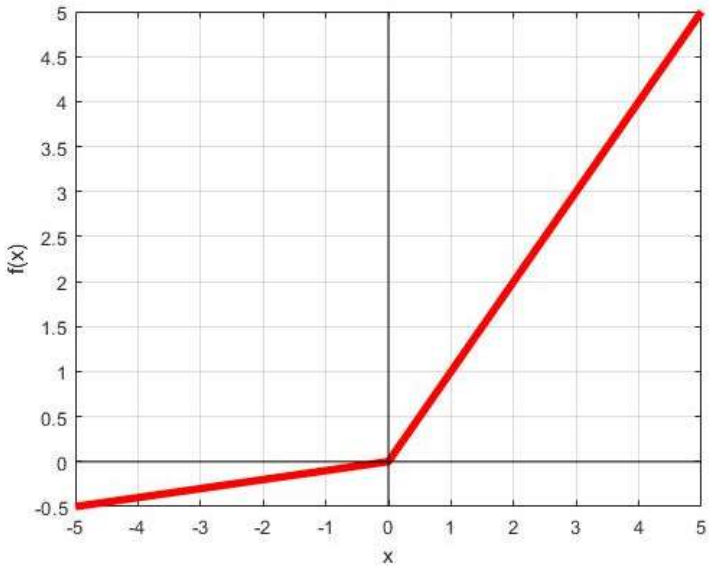


Obrázek 7:ReLU aktivační funkce

$$f(x) = \max(0, x) \tag{4}$$

d) Leaky Rectified Linear Unit funkce

Tato aktivační funkce řeší hlavní problém ReLU funkcí, a to přidáním sklonu pro záporné vstupy. Ostatní výhody a nevýhody jsou shodné s ReLU funkcí. Průběh funkce je na (Obr. 8) a její funkční předpis je dán vztahem (5).



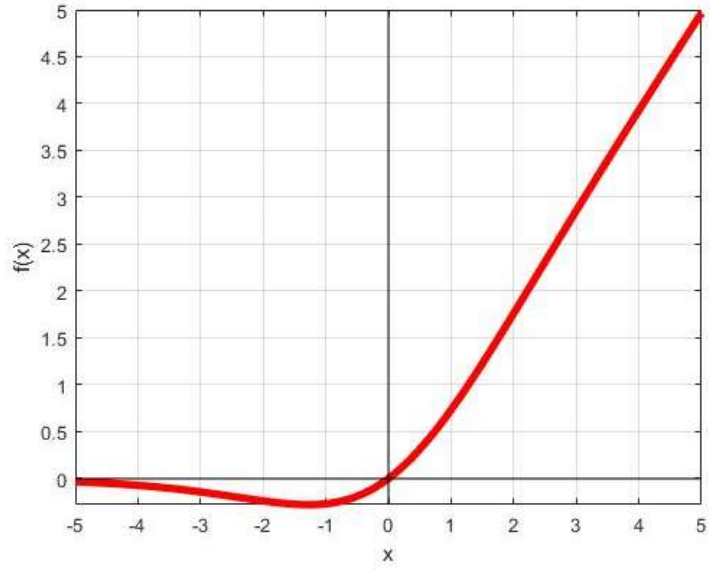
Obrázek 8: Leaky ReLU aktivační funkce

$$f(x) = \max(c \cdot x, x) \tag{5}$$

e) Swish

Swish je nejnovější aktivační funkce objevená vědci ze společnosti Google, kteří ve svém článku [15] uvádí, že Swish dosahuje lepších výkonů než ReLU funkce, při stejné výpočetní náročnosti. Průběh funkce je na (Obr. 9) a její funkční předpis je dán vztahem (6).

.



Obrázek 9: Swish aktivační funkce

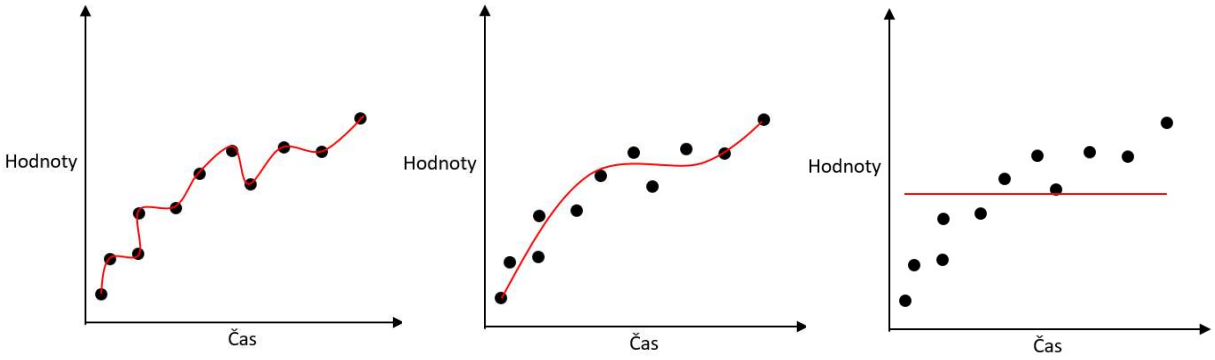
$$f(x) = \frac{x}{1 + e^{-x}} \tag{6}$$

Výběr aktivační funkce závisí na aproximované funkci. Obecně se dá tvrdit, že nejpoužívanější je ReLU aktivační funkce, kvůli rychlosti učení. Její největší nevýhoda se dá odstranit například nahrazením záporné části mírně sklopenou přímkou, což částečně řeší problém nulového gradientu. Problematikou výběru vhodné aktivační funkce se zabývá [16], kde je autory vyvíjena snaha o automatizaci jejího výběru, případně jejich kombinací.

4.2 Hluboké neuronové sítě

Neuronová síť se skládá z umělých neuronů, které jsou vzájemně propojené. Výstup neuronu je vstupem jiného neuronu. Počet neuronů a způsob jejich propojení v neuronové sítí, se nazývá topologie sítě. Neurony v neuronové síti dělíme podle vrstvy, ve které se nachází, na vstupní, skryté a výstupní. Hodnoty vah jednotlivých propojení mezi neurony se při procesu učení mění, to nazýváme konfigurace sítě. Šíření a zpracování informace v neuronové síti je umožněno změnou stavů neuronů.

Nejpoužívanější model neuronové sítě je vícevrstvá síť s učícím algoritmem zpětného šíření chyby (backpropagation). Mezi hlavní problémy, které je nutné řešit u tohoto modelu neuronové sítě se zpětným šířením chyby, je volba topologie dané sítě. Topologie sítě by měla odpovídat složitosti řešeného problému. Obecně se dá tvrdit, že čím složitější problém, tím je nutné použít větší síť. Při použití příliš malé sítě se algoritmus obvykle zastaví v lokálním minimu a následně je nutná zvýšení počtu neuronů ve skrytých vrstvách. S vyšším počtem vah však roste výpočetní náročnost. Použití příliš velké sítě může mít též za následek tzv. overfitting, kdy síť až příliš následuje tréninkové vzory a není schopná ani částečné generalizace. Na následujícím obrázku (Obr. 10) je znázorněn rozdíl mezi overfittingem (vlevo) a underfittingem (vpravo), což je jeho pravý opak, v porovnání s přibližně optimální aproximací.



Obrázek 10: Rozdíl mezi overfittingem, vhodnou aproximací a underfittingem

Neuronové sítě jsou velice důležitou součástí algoritmů, které budou popsány v následujících kapitolách této práce. Jsou zde využívány k aproximaci složitých funkcí. Schopnost neuronových sítí s jednou skrytou vrstvou obsahující až nekonečně neuronů aproximovat jakoukoli spojitou funkci n proměnných dokazuje Kolmogorova věta [17]. Důkaz, že se zvolenou libovolnou přesností lze aproximovat libovolnou funkci, za předpokladu omezené a nekonstantní aktivační funkce, která zároveň není polynomem, lze najít v [18].

4.3 Proces učení umělých neuronových sítí

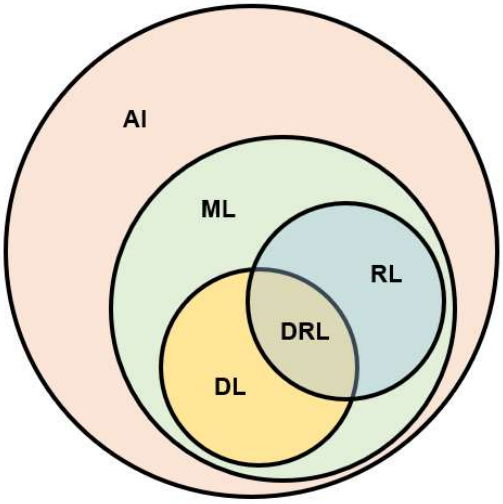
Obecně se dá tvrdit, že trénování umělých neuronových sítí probíhá v 5 krocích, které se opakují, dokud není dosaženo konvergence k optimálnímu řešení úlohy. Prvním krokem je inicializace vah v sítích. Obvykle jsou váhy inicializovány buď jako náhodná čísla nebo jako nuly. Druhým krokem je dopředný chod, kdy signál projde od vstupu, přes všechny neurony ve všech vrstvách (neplatí pro některé moderní architektury neuronových sítí) až po výstup. Výstup sítě je porovnán s optimálním řešením a je učiněn krok třetí, výpočet chybové funkce. Ta určuje, jak daleko je současný model od optimálního stavu. Tato chyba je ve čtvrtém kroku při zpětném průchodu neuronovou sítí parametrem pro krok pátý, aktualizaci vah v síti. Zpětný průchod chyby neuronovou sítí je algoritmus, kterým získáme po určitém počtu iterací optimální váhy, metodou gradientního klesání (gradient descent). Snaha je minimalizovat ztrátovou funkci s využitím co možná nejnižší výpočetní náročností.

Celková chyba, která je parametrem pro další optimalizaci, je dána součtem chybových funkcí pro všechny výstupy sítě. Algoritmus zpětného šíření chyby provádí derivace a s požitím Leibnizova řetězového pravidla určí iteračně optimální váhy.

5. Hluboké posílené učení

Hluboké posílené učení je obor kombinující poznání z oborů hlubokého učení a posíleného učení. Jedná se prakticky o využití hlubokých neuronových sítí pro potřeby aproximace složitých funkcí posíleným učením. Je vhodné v úlohách, kde nevíme, jaké akce by měl kontrolér vykonávat, aby bylo dosaženo optimálního chování systému. Bylo by velmi obtížné ho naprogramovat přímo. Hluboké zpětnovazební učení je technika založena na postupném prozkoumávání prostředí a hodnocení nejprve náhodných, později stále více optimalizovanými zásahy agenta (kontroléru). Získaná zpětná vazba, odměna, je zde mírou správnosti agentových zásahů.

Na následujícím vennově diagramu (Obr. 11) jsou uvedeny vztahy mezi umělou inteligencí (AI – Artificial Inteligence), strojovým učením (ML – Machine Learning), hlubokým učením (DL – Deep Learning), posíleným učením (RL – Reinforcement Learning) a hlubokým posíleným učením (DRL – Deep Reinforcement Learning).



Obrázek 11: Vennův diagram vztahů mezi obory umělé inteligence

5.1 Hluboké učení

Publikace [19] uvádí pět definic hlubokého učení:

a) Definice 1

Třída technik strojového učení, která využívá mnohavrstvého nelineárního zpracování informací pro extrakci a transformaci funkcí se supervizí nebo bez ní a pro analýzu a klasifikaci vzorů.

b) Definice 2

Hluboké učení je odvětví strojového učení, jež je založeno na algoritmech pokrývající víceúrovňovou reprezentaci vzorů tak, aby se daly modelovat složité vztahy mezi jednotlivými daty. Funkce vyšší úrovně jsou tak definovány s ohledem na funkce nižší úrovně. Takové hierarchie funkcí se nazývají hluboké architektury. Většina těchto modelů je založena na nesupervizovaném učení, tedy na učení bez učitele.

c) Definice 3

Hluboké učení je odvětví strojového učení, které je založeno na učení několika úrovní reprezentace, což odpovídá hierarchii vlastností, faktů nebo konceptů, kde jsou jednotlivé závislosti na vysokých úrovních definovány pomocí závislosti na úrovních nižších. Obdobně mohou závislosti na nižších úrovních pomoct definovat závislosti na vyšších úrovních. Metody hlubokého učení konkrétně spadají do části strojového učení zabývající se učením reprezentace dat. Jednotlivá data můžeme reprezentovat mnoha způsoby (u obrázku např. samotnými pixely nebo různými důležitými rysy). Některé reprezentace dat nám v závislosti na požadované funkci systému usnadňují jeho naučení. Výzkum v této oblasti strojového učení se snaží definovat, jaká je ideální reprezentace dat a jak daný systém naučit požadované funkce.

d) Definice 4

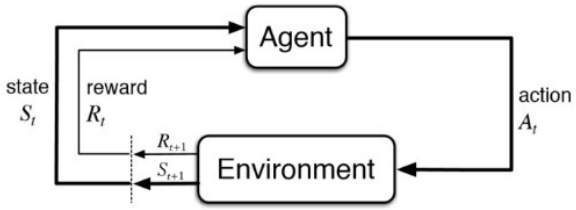
Hluboké učení je sada algoritmů ve strojovém učení, které se pokouší učit na více úrovních, odpovídajících různým úrovním abstrakce. Obvykle používá umělé neuronové sítě. Úrovně v těchto naučených statistických modelech odpovídají odlišným úrovním konceptů, kde jsou koncepty vyšší úrovně definovány z konceptů nižší úrovně a stejné koncepty nižší úrovně mohou pomoci definovat mnoho konceptů vyšší úrovně.

e) Definice 5

Hluboké učení je nová oblast strojového učení, která vznikla za účelem přiblížit strojové učení k jejímu prvotnímu cíli, a to k umělé inteligenci. Dává si za cíl zachytit data na několika úrovních abstrakce a reprezentace, což je velmi důležité u dat jako jsou obrázky, zvuk a text.

5.2 Posílené učení

Posílené učení je oblast strojového učení, zabývající se učením softwarových agentů, kteří na základě pozorování S_t následně vykonávají akce A_t v prostředí, za které získávají odpovídající odměnu R_t, případně penalizaci, což je znázorněno na následujícím obrázku (Obr. 12).



Obrázek 12: Schéma posíleného učení [4]

5.2.1 Definice základních pojmů

Stav prostředí s je kompletní popis stavu prostředí. Pozorování o je částečný popis stavu prostředí. Akční prostor je množina všech v prostředí proveditelných akcí, mohou být spojité nebo diskrétní.

Strategie je soubor pravidel, podle kterých se agent rozhoduje, kterou akci provede. Může být deterministická (7) nebo stochastická (8)

$$a_t = \mu(s_t) \tag{7}$$

$$a_t \sim \pi(\cdot\,|s_t). \tag{8}$$

Trajektorie je sekvence nastalých stavů a provedených akcí (9). Rozdíly v prostředí v čase *t* a *t* plus jedna jsou dány zákonitostmi prostředí a záleží pouze na poslední provedené akci

$$\tau = (s_0, a_0, s_1, a_1, \dots). \tag{9}$$

První stav s_0 je náhodně vybrán z úvodní distribuce ρ_0, dle vztahu (10)

$$s_0 \sim \rho_0(\cdot). \tag{10}$$

Odměna r_t záleží na současném stavu prostředí, právě provedené akci a stavu, který nastane po provedení akce, dle vztahu (11)

$$r_t = R(s_t, a_t, s_{t+1}). \tag{11}$$

Často však dochází ke zjednodušení na závislost pouze na současném stavu, dle vztahu (12)

$$r_t = R(s_t) \tag{12}$$

nebo na stavově-akčním páru (s_t,a_t) dle vztahu (13)

$$r_t = R(s_t, a_t). \tag{13}$$

Cílem agenta je maximalizace kumulativní odměny. Návratová hodnota *R* bývá nediskontovaná časově konečná suma daná vztahem (14)

$$R_t = \sum_{t=0}^{T} r_t \tag{14}$$

nebo diskontovaná nekonečná suma odměn daná vztahem (15)

$$R_t = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{15}$$

Discount factor γ nám určuje důležitost odměny v současnosti a v dlouhodobém horizontu. Dále zaručuje, že návratová suma bude konvergovat ke konečné hodnotě.

5.2.2 Hodnotové funkce

Je velice užitečné znát hodnotu stavu nebo stavově-akčního páru (*a*, *s*). Takovou hodnotou je myšlena očekávaná návratová hodnota, při začátku v daném stavově-akčním páru a následném chování se podle strategie. Existují čtyři v algoritmech posíleného učení nejčastěji používané hodnotové funkce.

a) On-policy hodnotová funkce

Vrací očekávanou návratovou hodnotu, při začátku ve stavu s, za předpokladu, že vždy postupujeme podle strategie π a je dána vztahem (16)

$$V^{\pi}(s) = E_{\tau\sim\pi}[R(\tau)|s_0 = s]. \tag{16}$$

b) On-policy akčně-hodnotová funkce

Vrací očekávanou návratovou funkci, při začátku ve stavu *s*, provedení akce *a* a následném následování strategie π a je dána vztahem (17)

$$Q^{\pi}(s,a) = E_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a\ ]. \tag{17}$$

c) Optimální hodnotová funkce

Vrací očekávanou návratovou hodnotu, při začátku ve stavu *s*, pokud postupujeme vždy podle optimální strategie, dle vztahu (18)

$$V^{*}(s) = \max\left(E_{\tau\sim\pi}[R(\tau)|s_0 = s]\right). \tag{18}$$

d) Optimální akčně-hodnotová funkce

Vrací očekávanou návratovou hodnotu, při začátku ve stavu *s*, provedení akce *a* a následném postupování podle optimální strategie, dle vztahu (19)

$$Q^*(s,a) = \max\left(E_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a\,]\right). \tag{19}$$

Dvě hlavní souvislosti mezi stavově-akčním párem a hodnotovou funkcí se tedy dají vyjádřit následujícími vztahy (20) a (21)

$$V^{\pi}(s) = E_{a\sim\pi}[Q^{\pi}(s,a)] \tag{20}$$

a

$$V^*(s) = \max\left(Q^*(s,a)\right). \tag{21}$$

5.2.3 Optimální Q-funkce a optimální akce

Důležitá je souvislost mezi optimální akčně-hodnotovou funkcí $Q^*(s, a)$ a akcí vybranou optimální strategií. Dle definice optimální akčně-hodnotová funkce vrací očekávanou návratovou hodnotu, při prostředí ve stavu s a následném provedením akce a. Optimální strategie vybere ve stavu s akci, která povede k maximalizaci hodnotové funkce. Pokud je nalezena Q^*, lze přímo získat optimální akci, dle vztahu (22)

$$a^* = argmax_a Q^*(s,a). \tag{22}$$

5.2.4 Bellmanovy rovnice

Všechny uvedené hodnotové funkce se řídí speciálními rovnicemi konzistence zvanými Bellmanovy rovnice. Základní myšlenka Bellmanových rovnic je, že hodnota v počátku je součet očekávané odměny a odměny získané v následujícím kroku.

Bellmanovy rovnice pro on-policy hodnotové funkce jsou dány vztahem (23)

$$V^{\pi}(s) = E[r(s,a) + \gamma \cdot V^{\pi}(s')] \tag{23}$$

a (24)

$$Q^{\pi}(s,a) = E\big[r(s,a) + \gamma \cdot E[Q^{\pi}(s',a')]\big]. \tag{24}$$

Bellmanovy rovnice pro optimální hodnotové funkce jsou dány vztahy (25):

$$V^*(s) = \max\left(E[r(s,a) + \gamma \cdot V^*(s')]\right) \tag{25}$$

a (26)

$$Q^*(s,a) = E[r(s,a) + \gamma \cdot \max\left(Q^*(s',a')\right)]. \tag{26}$$

5.2.5 Výběr algoritmu

Prvním algoritmem schopným pracovat se spojitými akčními prostory byl v článku [20] představený algoritmus Deep Deterministic Policy Gradient (DDPG), který využívá off-policy data. To znamená že aktualizuje Q-hodnoty s využitím Q-hodnot v následujícím stavu a v něm provedené akce, která je provedena bez použití explorace. Algoritmy využívající on-policy data aktualizují Q-hodnoty na základě Q-hodnot v následujícím stavu a provedené akci, která je provedena na základě současné strategie.

Jeho hlavní výhodou je, že jeho implementace je jednoduší než u konkurenčních algoritmů. Další výhodou je fakt, že tento algoritmus byl již úspěšně využit v několika publikacích pro řízení mechanických soustav. Kombinace snazší implementace a možnost inspirace v odborných publikacích, byla při výběru algoritmu zásadní, protože se jedná o velice složité algoritmy.

6. Deep Deterministic Policy Gradient

DDPG je algoritmus posíleného učení, který kombinuje přístup optimalizaci strategie a Q-učení. Za použití off-policy dat a Bellmanovy rovnice je vypočtena Q-funkce, která je následně využita ke zjištění optimální strategie. Jedná se o actor-critic techniku, operující se dvěma modely, actorem a criticem. Actor je síť aproximující strategii, jejímž vstupem je stav prostředí a výstupem provedená akce ze spojitého akčního prostoru. Critic je aproximován Q-hodnotovou sítí, jejímž vstupem je akce a stav prostředí a výstupem Q hodnota. DDPG je off-policy algoritmus. Deterministic v názvu algoritmu značí, že actor počítá akci přímo a jeho výstupem není pravděpodobnostní distribuce, jako je tomu u jiných algoritmů [20].

6.1 Q-učení

Optimální akčně hodnotová funkce $Q^*(s, a)$ je popsána Bellmanovou rovnicí (27)

$$Q^*(s,a) = E[r(s,a) + \gamma \cdot max_{a'}(s', a')]. \quad (27)$$

Q^* je aproximováno neuronovou sítí Q s parametry φ, kterou budeme značit Q_φ. Zde je využita Mean-Squared Bellamn Error (MSBE) funkce, která určuje, jak dobře Q_φ splňuje Bellmanovu rovnici (28)

$$L(\varphi, D) = E\left[\left(Q_\varphi(s,a) - (r + \gamma \cdot (1-d) \cdot max_{a'} Q_\varphi(s', a')\right)^2\right], \quad (28)$$

kde d je boolesovská hodnota, tedy pravda (d=1), pokud nastal terminální stav nebo nepravda (d=0), pokud ještě nenastal.

Algoritmus Q-učení pro aproximaci funkcí, jako je například DQN, jsou založeny na minimalizaci MSBE ztrátové funkce. Jsou zde využity především tyto dva tzv. triky:

a) První trik – replay buffer

Replay buffer funguje jako zásobník předchozích zkušeností algoritmu. Pro stabilitu algoritmu je nutné, aby byl replay buffer dostatečně velký, aby v něm mohlo být uloženo dostatečné množství zkušeností. Při použití pouze několika posledních zkušeností, nastává overfitting. Použití všech zkušeností by zase podstatně zpomalilo proces učení. Je velice vhodné uchovávat starší zkušenosti, získané s horší strategií.

b) Druhý trik – target sítě

Když minimalizujeme MSBE, snažíme se o přiblížení Q-funkce těmto sítím. Problémem je závislost na stejných parametrech fí, které se snažíme optimalizovat, což by činilo minimalizaci MSBE značně nestabilní. Řešením je použití parametrů blízkých fí, s časovým odstupem. Target sítě jsou tedy opožděnými verzemi původních actor a critic sítí. V některých algoritmech hlubokého posíleného učení jsou tyto target sítě aktualizovány fixně každý krok. V algoritmu DDPG je prováděna aktualizace dle následujícího vztahu (29)

$$\tau_{target} \leftarrow \rho \cdot \varphi_{tarhet} + (1 + \rho) \cdot \varphi, \tag{29}$$

kde ρ je parametr dosahující hodnot mezi 0 a 1, obvykle je velmi blízký 1.

Q-učení v algoritmu DDPG je prováděno minimalizací následující MSBE ztrátové funkce metodou stochastického gradientního sestupu (SGD) dle vztahu (30)

$$L(\varphi, D) = E\left[\left(Q_{\varphi}(s,a) - (r + \gamma \cdot (1-d) \cdot Q_{target}(s', \mu_{\varphi_{target}}(s'))\right)^2\right], \tag{30}$$

kde $\mu_{\varphi_{target}}$ je target strategie.

6.2 Optimalizace strategie

Optimalizace strategie je u algoritmu na rozdíl od jiných algoritmů velice jednoduché. Snaha je naučit deterministickou strategii, která bude maximalizovat $Q_{\varphi}(s, a)$. Kvůli spojitosti akčního prostoru můžeme učinit předpoklad, že je Q-funkce diferencovatelná podle akcí. Stačí tedy provézt gradientní vzestup vzhledem k parametrům strategie, dle vztahu (31)

$$\max\left(E\left[Q_{\varphi}\left(s, \mu_{\theta}(s)\right)\right]\right). \tag{31}$$

Parametry Q-funkce jsou brány jako konstanty.

6.3 Aktualizace sítí

Aktualizace critic sítě probíhá minimalizací následující ztrátové funkce, která je dána vztahem (32)

$$L = \frac{1}{N}\sum_{i=1}^{N}((r_i + \gamma \cdot Q(s_{i+1}, \mu(s_{i+1}|\theta^{\mu})|\theta^{Q})) - Q(s_i, a_i|\theta^{Q}))^2 \tag{32}$$

Protože cílem je maximalizace Q-hodnot ve vybrané dávce dat, určíme gradient ze vztahu (33)

$$\nabla_{\theta^\mu} J \approx E[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|s = s_t, a = \mu(s_t|\theta^\mu)], \tag{33}$$

kde J značí očekávanou odměnu. Aplikací řetízkového pravidla potom dostaneme vztah (34)

$$E\big[\big(\nabla Q(s, a|\theta^Q)\big|s = s_t, a = \mu(s_t|\theta^\mu)\big) \cdot (\nabla_{\theta^\mu} \mu(s|\theta^\mu)|s = s_t)\big]. \tag{34}$$

V článku [7] bylo dokázáno, že při použití tohoto vztahu dostaneme maximální očekávanou odměnu.

Na následujícím obrázku (Obr. 13) je tzv. pseudokód algoritmu DDPG [20].

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1 - \tau)\theta^{\mu'}$$

  end for
end for

Obrázek 13: Pseudokód algoritmu DDPG

6.4 Explorace a exploatace

Exploatace znamená, že agent našel řešení problému, jehož aplikací získá určitou odměnu. Nalezené strategie se bude držet, protože o jiné zatím nenashromáždil žádné informace. Algoritmus tak zůstane u nalezené strategie a nikdy se nepokusí vyzkoušet jinou. Setrvá tedy se suboptimální

strategií. Explorace je pravý opak exploatace. Agent provádí exploraci, když neustále zkouší nové strategie. Pokud však nalezne kvalitní strategii, nezůstane u ní a začne zkoušet další.

Ani jeden z popsaných přístupů není vhodný. Využívá se zde kompromisu, kdy nejprve agent provádí exploraci, získá představu o vhodné strategii a tu následně exploatuje. Snaha o exploraci v čase klesá, což přispívá ke konvergenci k optimální strategii.

Algoritmus DDPG trénuje deterministickou strategii off-policy způsobem. Deterministická strategie způsobuje, že agent v úvodních epizodách učení nemá šanci prozkoumat metodou pokus-omyl dostatečné množství akcí k nalezení pro učení užitečných signálů. Tento problém je autory článku [20] řešen přičtením šumu k prováděné akci. Konkrétně autoři doporučují použití časově korelovaného Ornstein-Uhlenbeckova šumu, dle vztahu (35)

$$\mu'(s) = \mu(s|\theta^{\mu}) + N, \tag{35}$$

kde N představuje šum přičítaný k výstupu actor sítě. Výběr šumu záleží na prostředí. Autoři článku [20] použili Ornstein-Uhlenbeckův proces [21]. Jedná se o stochastický proces používaný ve finanční matematice, fyzice a evoluční biologii. Původně byl autory aplikován na fyzikální model pro rychlost Brownovské částice se třením.

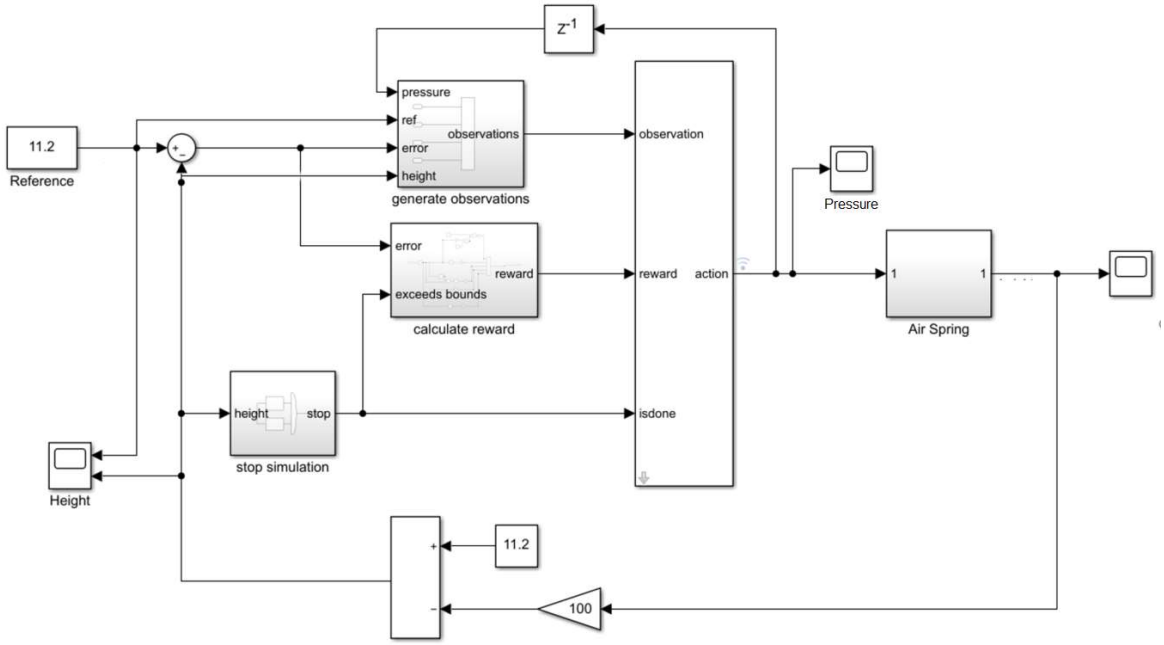
U algoritmů se spojitým akčním prostorem je důležité vhodně nastavit rozptyl šumu, aby byla podpořena explorace. Součin rozptylu a odmocniny ze vzorkovacího času odpovídá 1 až 10 % akčního prostoru. Při rychlé konvergenci agenta k lokálnímu minimu, je obecně doporučováno zvýšit rozptyl šumu. Ornstein-Uhlenbeckův šum je určen stochastickou diferenciální rovnicí (36)

$$dx_t = \theta_{ou} \cdot x_t \cdot dt + \sigma \cdot dW_t, \tag{36}$$

kde θ_{ou} a σ jsou konstanty a platí že $\theta_{ou} > 0$ a $\sigma > 0$. W_t potom představuje Wienerův proces, jehož derivací je bílý šum. Autoři [20] doporučují hodnoty $\theta_{ou} = 0{,}15$ a $\sigma = 0{,}2$.

7. Tvorba algoritmu a proces učení

Algoritmus DDPG představený v minulé kapitole byl následně sepsán v softwaru Matlab, kde interaguje s matematickým modelem pneumatické pružiny (Obr. 14), vytvořeném v Simulinku. Agent zde má 4 vstupy a 1 výstup. Vstupy jsou pozorování z prostředí, konkrétně akce vykonaná v předchozím kroku, požadovaná výška, skutečná výška pružiny a rozdíl skutečné a naměřené výšky. Výstupem agenta je přetlak v měchu pružiny. Agent zde také dostává zpětnou vazbu prostřednictvím odměnové funkce, které bude věnována celá jedna z následujících podkapitol. Vyskytuje se zde také terminační blok, který při překročení maximální nebo minimální přípustné výšky pružiny ukončí právě probíhající episodu učení.



Obrázek 14: Model vytvořený v Simulinku

7.1 Nastavení hyper-parametrů

Hyper-parametry se v nazývají parametry algoritmů posíleného učení. Patří mezi ně learning rate, discount factor, mini-batch size a další. V následujících odstavcích bude popsán vliv vybraných hyper-parametrů, jejichž volba nejzásadněji ovlivnila chování vytvořeného algoritmu. Nastavování naprosté většiny těchto hyper-parametrů je prováděno empiricky. Pro některé z nich existují obecně platné poučky. Neexistují žádná pevně daná pravidla pro jejich volbu.

Learning rate je hyper-parametr, který určuje rozsah změny modelu v časovém kroku dané epizody učení. Pro actor síť byla zvolena hodnota 0,0001 a pro critic síť 0,001. Jedná se o hodnoty používané ve většině aplikací těchto algoritmů. Obecně může learning rate nabývat hodnot

v intervalu (0, 1). Discount factor může nabývat hodnot v intervalu (0, 1). Udává, jak moc má agentovi záležet na odměnách získaných v nejbližší budoucnosti a v budoucnosti vzdálené. Pokud je roven nule, agenta zajímá pouze následující odměna. Pokud se však rovná 1, agent se bude plně soustředit na odměny v budoucnu. Obvykle se používají hodnoty tohoto hyper-parametru od 0,9 do1. V případě této práce byl empiricky zvolen discount factor roven 0,97. Mini-batch size představuje velikost dávek náhodně vybraných dat, na kterých algoritmus provádí učení. V této práci byla zvolena hodnota 128. Vyšší hodnoty způsobují zásadní zvýšení výpočetní náročnosti, nižší naopak neschopnost algoritmu nalézt optimální řešení.

Mezi hyper-parametry se řadí také parametry hlubokých neuronových sítí. Těm bude věnována podkapitola 7.3.

7.2 Odměnová funkce

Odměnová funkce r_t (40) slouží agentům posíleného učení jako zpětná vazba o kvalitě a vhodnosti provedených akcí [11]. Navržená odměnová funkce je rozdělena na část závislou na odchylce mezi požadovanou a naměřenou výškou pružiny r_{te} (37), dále na část penalizující algoritmus za příliš rychlou změnu tlaku mezi předchozím a současným krokem v dané epizodě r_{tp} (38) a poslední část penalizuje algoritmus za překročení vyhrazených mezí r_{tt} (39), kde *EB* značí logickou hodnotu rovnou jedné právě při jejich překročení. Část odměnového signálu závislá na změně přetlaku je zde z toho důvodu, že algoritmus před jejím přidáním rychle konvergoval k řešení, kdy s určitou frekvencí aplikoval maximální a minimální možný přetlak. Dále v průběhu učení již agent pouze měnil frekvenci, se kterou osciloval mezi limitními přetlaky. Algoritmus je z těchto důvodů penalizován za příliš rychlé změny tlaku.

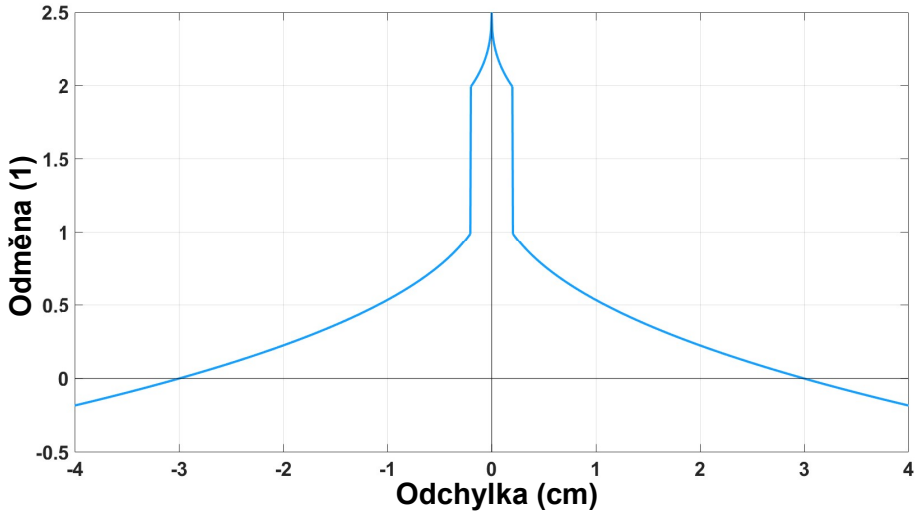
$$r_{te} = \frac{3 \cdot \left(1 - \left(\frac{|e_t|}{3}\right)^{0.4}\right)}{2} + (|e_t| < 0.2) + (|e_t| = 0) \tag{37}$$

$$r_{tp} = -\frac{\left((|p_{t-1}| - |p_t|) \geq 0.03\right)}{2} \tag{38}$$

$$r_{tt} = -(EB) \cdot 5 \tag{39}$$

$$r_t = r_{te} + r_{tp} + r_{tt} \tag{40}$$

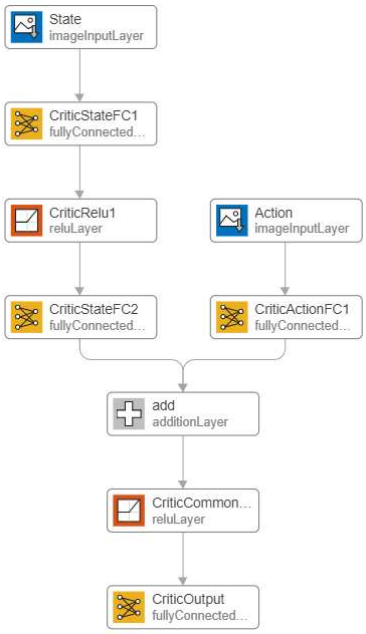
Na následujícím obrázku (Obr. 15) je zobrazen průběh části odměnové funkce závislé na odchylce mezi požadovanou a skutečnou hodnotou. Maxima funkce dosahuje při nulové odchylce a je symetrická podle osy y, protože při výpočtu odměny je vždy uvažována absolutní hodnota z této odchylky.



Obrázek 15: Odměnová funkce

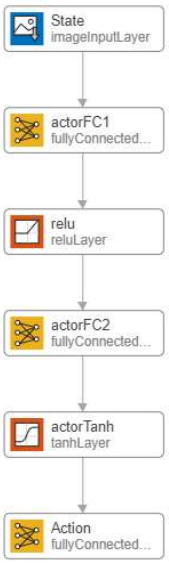
7.3 Použité hluboké neuronové sítě

Na následujícím obrázku (Obr. 16) je znázorněna hluboká umělá neuronová síť reprezentující critic síť. Má dvě různé vstupní, dvě skryté a jednu výstupní vrstvu. Rozměr skrytých sítí je 100 neuronů.



Obrázek 16: Schéma actor sítě

Na následujícím obrázku (Obr. 17) je znázorněna hluboká umělá neuronová síť reprezentující actor síť. Má jednu vstupní, dvě skryté a jednu výstupní vrstvu. Rozměr skrytých sítí je 100 neuronů. Výstupní síť nemá na rozdíl od ostatních vrstev použitých u sítí v této práci ReLU aktivační funkci, ale tanh aktivační funkci, aby byl její výstup v intervalu (-1, 1). To je běžná praktika u algoritmů hlubokého posíleného učení a je zaváděna kvůli podobnosti řádů hodnot na vstupu, uvnitř skrytých vrstev a na výstupu, což vede ke stabilitě algoritmu [20].



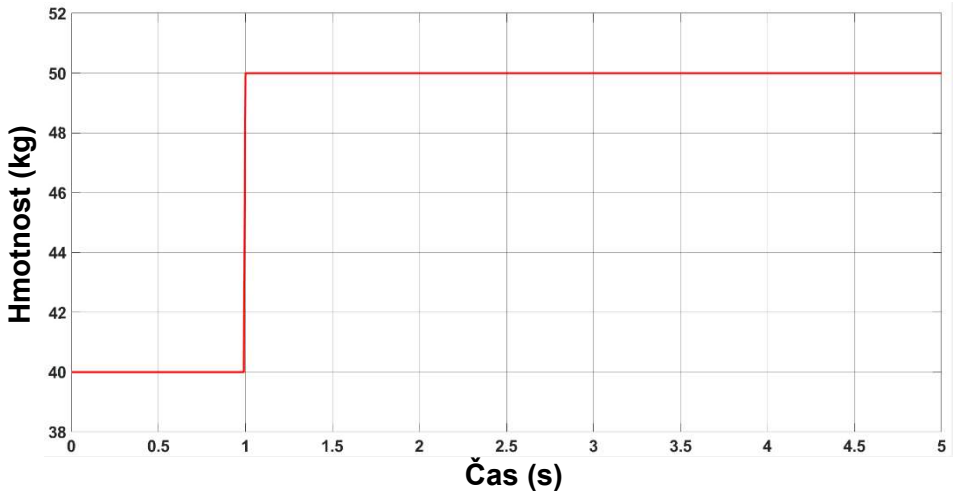
Obrázek 17: Schéma critic sítě

V obou neuronových sítích byla použita ReLU aktivační funkce. Důvodem k jejímu výběru byl fakt, že se jedná o v praxi nejčastěji používanou funkci a je vysoce stabilní [20]. Před samotným procesem učení byly váhy v sítí ortogonálně inicializovány [22]. Bez provedení této inicializace nebyl algoritmus schopen konvergovat k řešení úlohy.

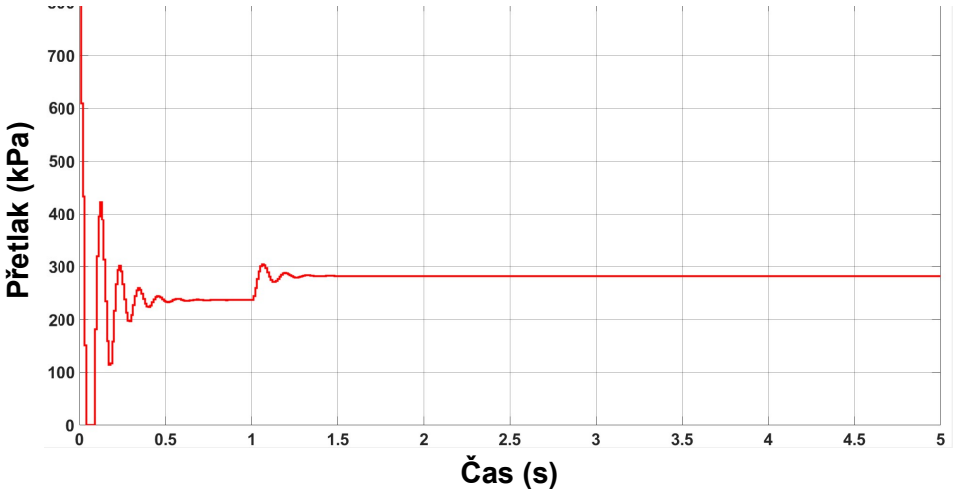
7.4 Výsledky simulací

V této podkapitole budou představeny výsledky simulací pro jednotlivá externí buzení. K buzení systému pružiny a pevně upevněného závaží o hmotnosti 40 kg byly použity signály skokové funkce, sinusový a pilový průběh. V průběhu jednotlivých epizod bylo toto externí buzení přičteno ke statickému závaží připočteno v čase 1 s ve formě změny hmotnosti. Typ zatěžovacího signálu a frekvence a amplituda u sinusového a pilového průběhu byly voleny náhodně. Výsledky jsou na obrázcích (Obr. 18 až 26).

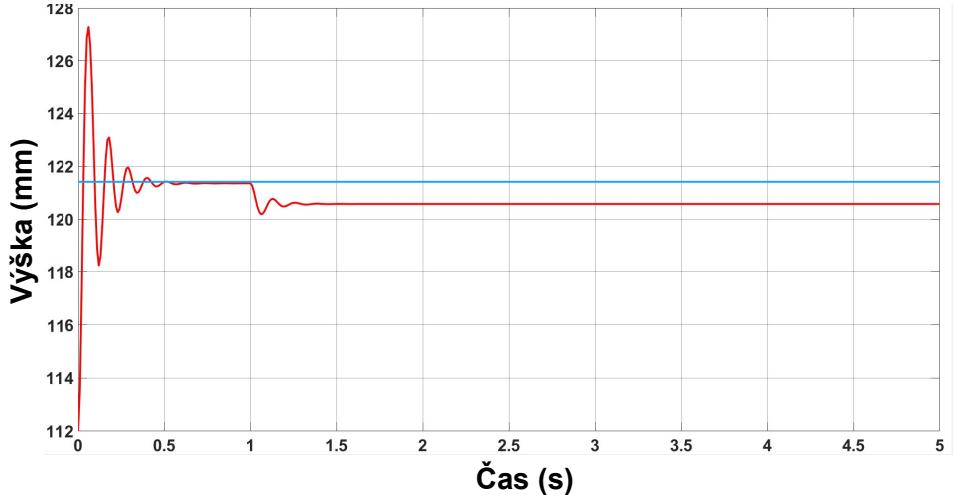
a) Skoková funkce – 10 kg



Obrázek 18: Průběh zatěžování skokovou změnou hmotnosti

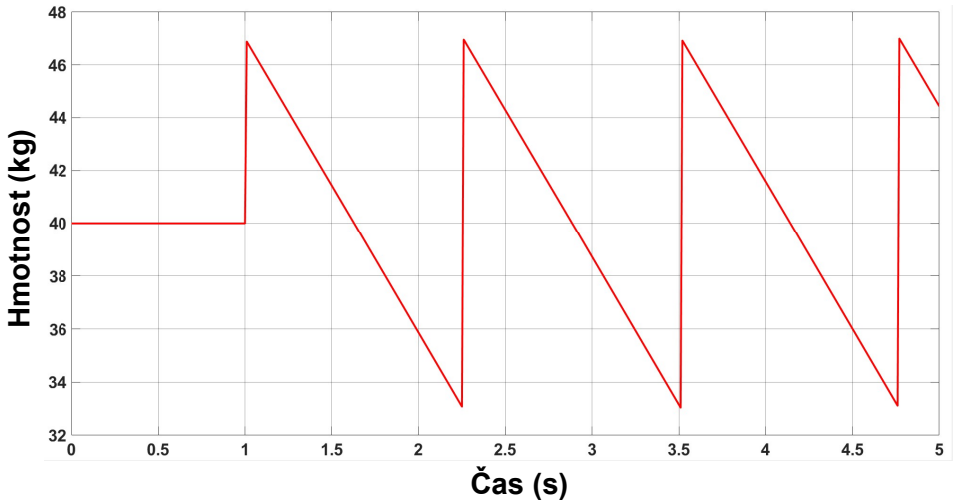


Obrázek 19: Průběh přetlaku při buzení skokovou funkcí

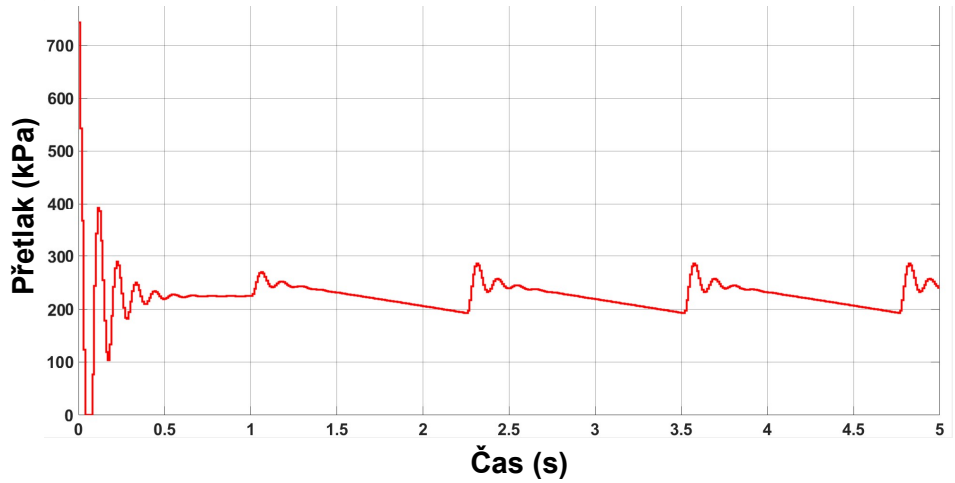


Obrázek 20: Průběh naměřené (červeně) a požadované (modře) výšky pružiny při skokovém zatížení

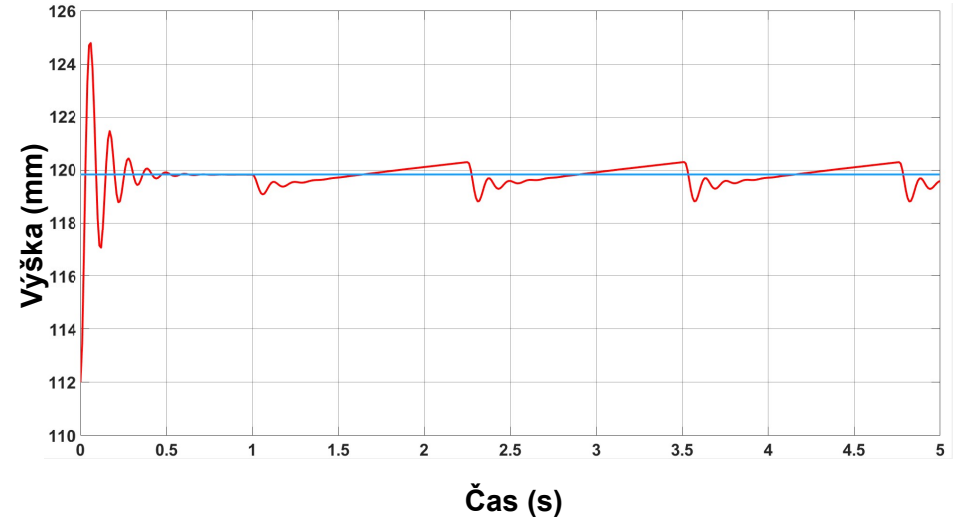
b) Pilový průběh – 7 kg, f = 2,25 Hz



Obrázek 21: Průběh zatěžování změnou hmotnosti s pilovým průběhem

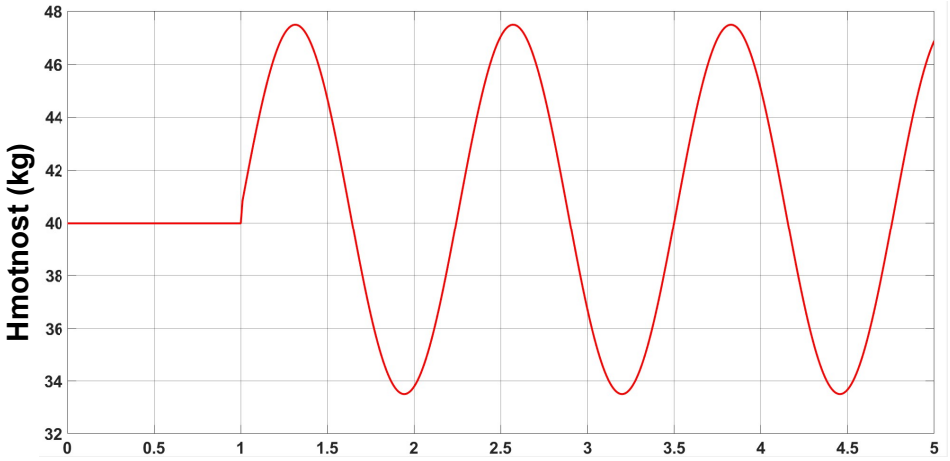


Obrázek 22: Průběh přetlaku při buzení pilovou funkcí

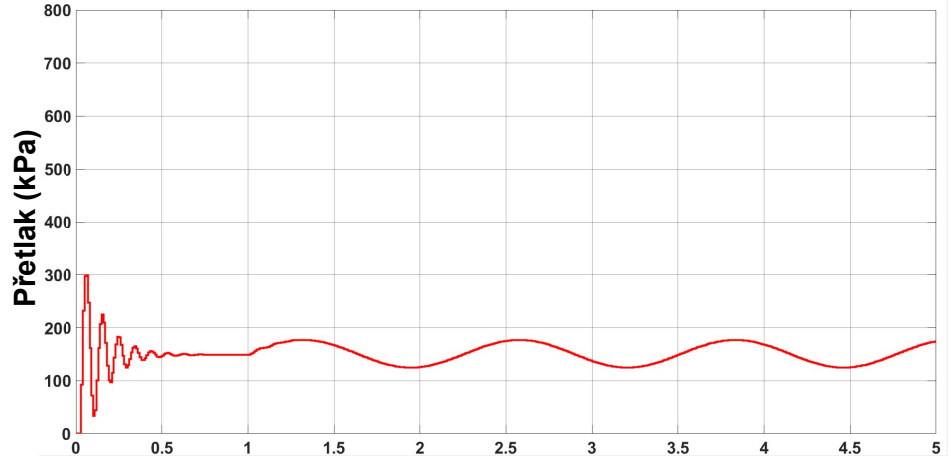


Obrázek 23: Průběh naměřené (červeně) a požadované (modře) výšky pružiny při zatížení s pilovým průběhem

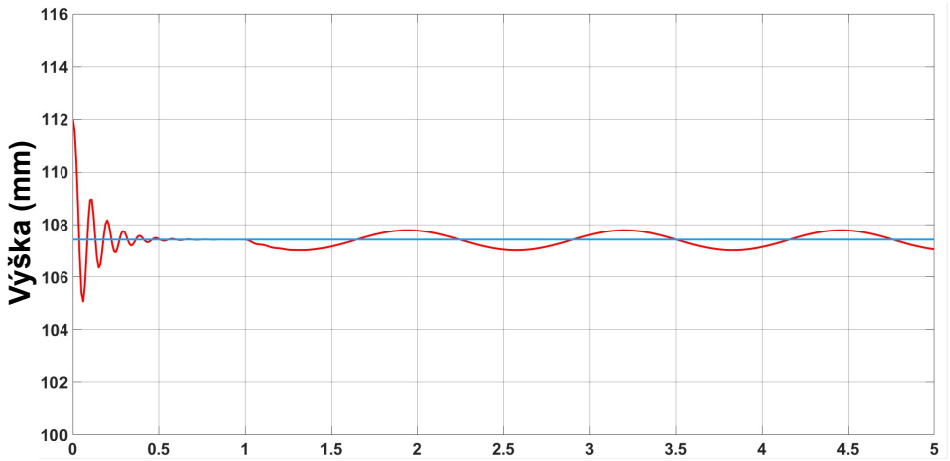
c) Sinový průběh – 7 kg, f = 2,25 Hz



Obrázek 24: Průběh zatěžování změnou hmotnosti se sinovým průběhem



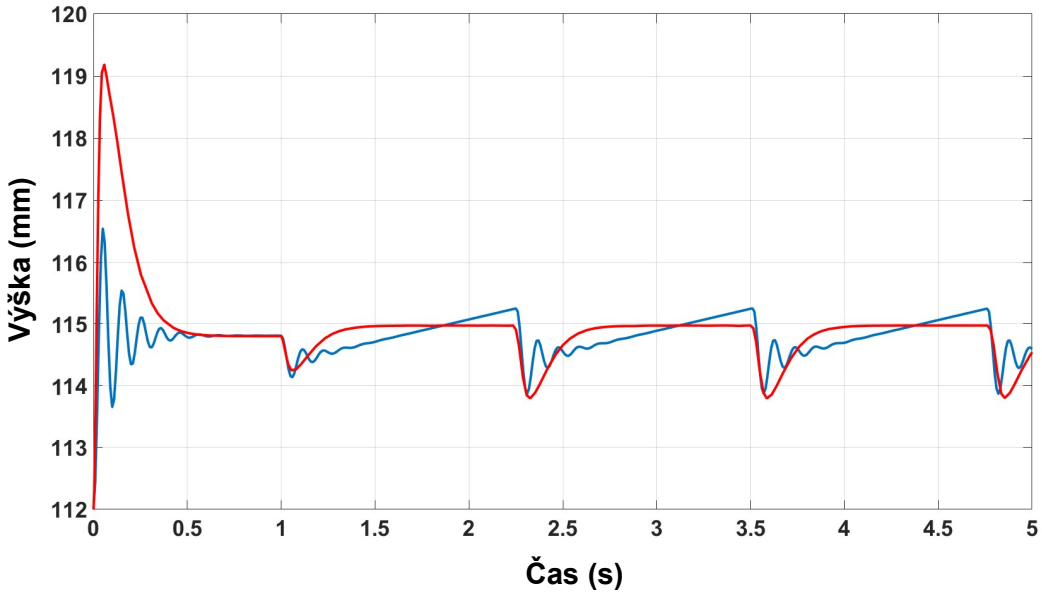
Obrázek 25 Průběh přetlaku při buzení sinovou funkcí



Obrázek 26: Průběh naměřené (červeně) a požadované (modře) výšky pružiny při zatížení se sinovým průběhem

7.5 Porovnání s PID regulací

Na následujícím obrázku (Obr. 27) je vidět porovnání výsledků získaných regulací algoritmem DDPG a PID regulátorem pro pilový průběh zatěžovací funkce. Po odeznění přechodových jevů na začátku simulace je patrné, že PID regulace rychle odstraní regulační odchylku, zatímco DDPG algoritmus dosáhne nulové odchylky až s podstatnou prodlevou a následně překročí hodnotu požadované výšky o přibližně 0,3 mm.

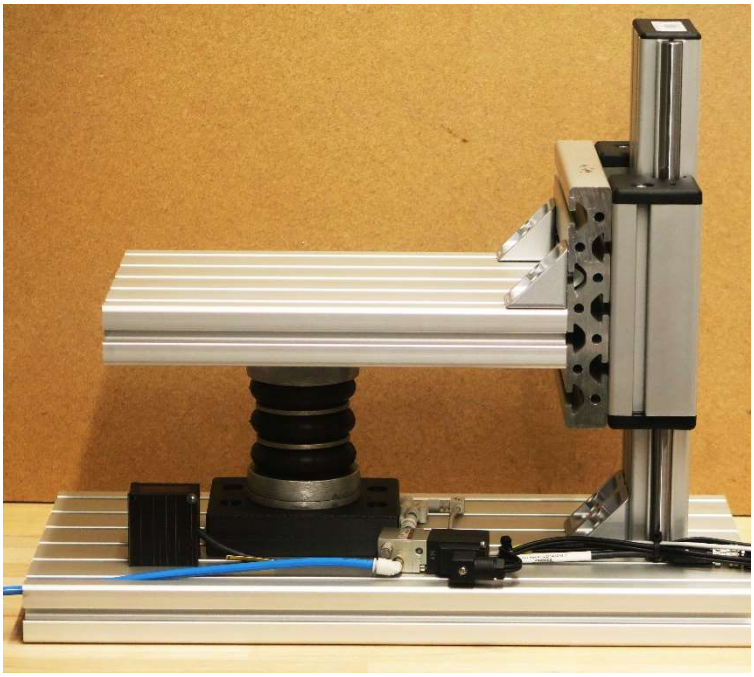


Obrázek 27: Porovnání PID regulace a výsledků algoritmu DDPG pro zatěžování s pilovým průběhem

Z těchto průběhů je patrné, že vyšší kvality regulace je dosaženo PID regulací. Nedostatky regulace použitým algoritmem se nepodařilo odstranit kvůli jeho nízké stabilitě, způsobené vysokou citlivostí na některé klíčové hyper-parametry. Především potom na parametry šumu, pomocí kterého je prováděna explorace. Tento problém by mělo vyřešit použití modernějších algoritmů, které z DDPG vycházejí a řeší jeho problémy se stabilitou.

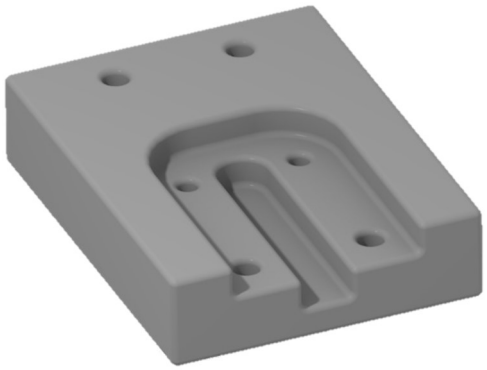
8. Konstrukce experimentálního přípravku

Pro potřeby aplikace algoritmu na reálnou pružinu byl navržen a sestaven experimentální přípravek (Obr. 28). Přípravek musí umožňovat zatěžování pružiny, a to statické zatížení v podobě závaží, ale také dynamické zatěžování vyvolané vnějším buzením. Dále musí umožňovat měření délky pružiny laserovým snímačem, přívod stlačeného vzduchu do pružiny a regulaci tlaku vzduchu uvnitř měchu pružiny tlakovým regulátorem.



Obrázek 28: Sestavené experimentální zařízení

Kvůli potřebě upnutí pružiny k základně přípravku byla vytvořena podložka pod pneumatickou pružinu (Obr. 29).



Obrázek 29: CAD model podložky pod pružinu

Byla vytvořena technologií 3D tisku, konkrétně metodou FDM (Fused Deposition Modeling), na tiskárně Creality Ender 3. Hodnoty nejdůležitějších parametrů 3D tisku, které se nastavují v softwaru, tzv. sliceru, který na základě CAD modelu generuje G-kód, jsou uvedeny v následující tabulce (Tab. 1).

Tabulka 1: Parametry tisku

Parametr	Hodnota
Teplota tisku	205 °C
Teplota tiskové podložky	60 °C
Hustota výplně	60 %
Rychlost tisku	50 mm/s
Tloušťka stěny	3 mm
Výška vrstvy	0.16
Vzor výplně	trojúhelníkový

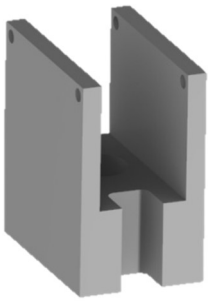
Byl použit materiál tiskové struny PLA neboli kyseliny polymléčné taktéž od firmy Creality. Tento termoplastický materiál se vyrábí z kukuřičného nebo bramborového škrobu či z cukrové třtiny. Jedná se spolu s ABS o nejčastěji používané materiály k 3D tisku, kvůli jejich univerzálnosti a dobrým mechanickým vlastnostem. Výrobky z ABS jsou výrazně tepelně odolnější, ale PLA je snadněji zpracovatelný a je odolnější proti deformacím a vadám vlivem chladnutí vytištěného materiálu. Některé vlastnosti použitého PLA jsou uvedeny v následující tabulce (Tab. 2).

Tabulka 2: Vlastnosti PLA

Parametr	Hodnota
Hustota	1250 kg/m³
Teplota tisku	185 až 235 ºC
Teplota podložky	0 až 60 ºC
Youngův modul	3500 MPa
Modul pružnosti	4000 MPa
Pevnost v ohybu	80 MPa
Pevnost v tahu	110 MPa
Teplota skelného přechodu	60 °C

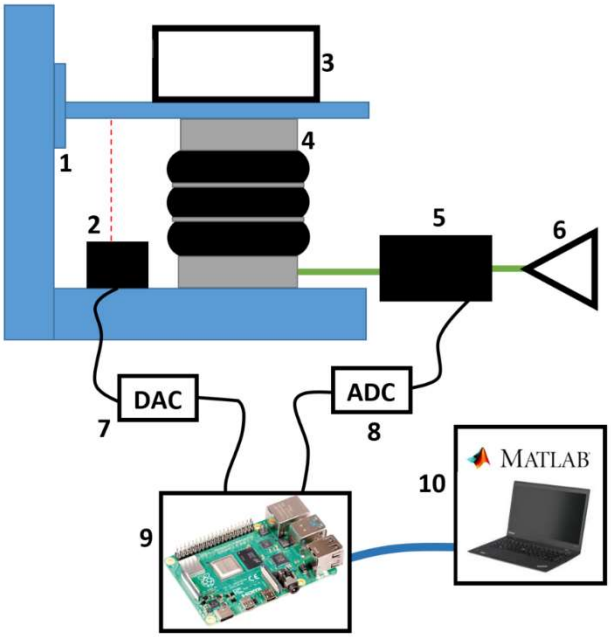
Pro potřeby upevnění laserového snímače vzdálenosti k základně testovacího přípravku byl vytvořen plastový díl, zobrazený na následujícím obrázku (Obr. 30). Tento díl byl stejně jako podložka pod pneumatickou pružinu vyroben technologií 3D tisku z materiálu PLA. Jelikož zde nenastává

žádné působení vnějších sil, byla zvolena hustota výplně pouze 10 %, kvůli úspoře materiálu a času potřebného k vytištění dílu.



Obrázek 29: CAD model držáku laserového sníamče

Schéma kompletního experimentálního zařízení je na obrázku (Obr. 31). Byla provedena montáž přípravku a byla ověřena funkčnost elektronického a pneumatického zapojení.



1. Rám
2. Laserový snímač délky
3. Závaží
4. Pneumatická pružina
5. Tlakový ventil
6. Zásobník stlačeného vzduchu
7. DA převodník
8. AD převodník
9. Raspberry PI 4B
10. PC s Matlabem

Obrázek 30: Funkční schéma zapojení experimentálního přípravku

9. Závěr

V úvodu této diplomové práce je popsána motivace k jejímu vytvoření a současná úroveň poznání v oblasti algoritmů hlubokého posíleného učení.

V následné teoretické části práce je popsáno fungování hlubokých neuronových sítí a jejich aplikace pro použité algoritmy. Následuje popis základních principů hlubokého posíleného učení a je odůvodněn výběr algoritmu Deep Deterministic Policy Gradient, který byl aplikován na regulaci výšky vlnovcové pneumatické pružiny.

V další části práce je popsána tvorba algoritmu v softwaru Matlab a matematický model pneumatické pružiny v softwaru Simulink, především potom jejich vzájemná interakce při procesu učení. Následuje popis asi nejdůležitější části algoritmů hlubokého posíleného učení, odměnové funkce, která algoritmu poskytuje zpětnou vazbu o kvalitě provedených řídících zásahů.

Na získaných výsledcích simulací je patrné, že regulace není ideální, odchylky od požadované hodnoty však dosahují pro sinusový a pilový průběh zatěžovacího signálu maximálně 0.3 mm, což je vzhledem k rozsahu pracovní oblasti pružiny, od 90 do 130 mm, dostatečná přesnost. Poměrně špatné kvality regulace je však dosaženo při skokovém navýšení hmotnosti, kdy nedojde k dostatečnému zvýšení přetlaku uvnitř měchu pružiny a nastalá trvalá regulační odchylka dosahuje 1 mm.

Z porovnání s PID regulací vyplývá, že řízení systému algoritmem DDPG nedosahuje kvalit PID regulace. To by mohlo být vyřešeno použitím modernějšího a stabilnějšího algoritmu, případně lepším nastavením současného algoritmu. Při optimalizaci fungování současného algoritmu by se bylo potřeba zaměřit především na parametry šumu, kterým je prováděna explorace prostředí a dále na inicializaci vah v hlubokých neuronových sítích ještě před začátkem procesu učení. To by však vyžadovalo paralelizaci výpočtů, protože proces učení zabere na stolním PC čas v řádu desítek hodin. S takto dlouhým časem učení je poměrně náročné měnit jeho parametry a sledovat změny v chování algoritmu.

V poslední části je popsána konstrukce experimentálního zařízení, sloužícího k upnutí reálné pružiny, na které měla být provedena aplikace algoritmů, které byly popsány v této práci. Za tímto účelem byl přípravek sestaven, byly rozmístěny snímače a připojen pneumatický obvod přivádějící stlačený vzduch do pružiny. Bylo též provedeno připojení algoritmu běžícího na PC s mikropočítačem Raspberry Pi 4, které slouží jako prostředník komunikace mezi algoritmem, laserovým snímačem délky a tlakovým ventilem. I přes dokončení všech příprav nebyla nakonec aplikace algoritmu na reálnou pružinu provedena, kvůli pandemické situaci v zemi v souvislosti s onemocněním COVID-19.

Další možnosti rozvoje této práce je aplikace naučeného algoritmu na reálnou pružinu v sestaveném experimentálním přípravku a následné učení se na této pružině. Doposud provedené učení algoritmu by tak posloužilo jako výchozí bod dalšímu učení, protože by bylo časově velice náročné, začít reálnou pružinu učit úplně od začátku. Tímto způsobem již bude mít algoritmus dobrou představu o fungování řízeného systému a mělo by tak docházet k jeho zdokonalování. Algoritmus by při tomto učení byl rovněž nucen zachytit reálné chování pružiny, především pak jevy, nezachycené v matematickém modelu.

Reference

[1] Rágulík, Jiří & Sivčák, Michal. (2019). Modeling of the Controlled Air Spring. Strojnícky časopis – Journal of Mechanical Engineering. 69. 107-112. 10.2478/scjme-2019-0037.

[2] Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore and Hassabis, Demis. "Mastering the Game of Go with Deep Neural Networks and Tree Search." Nature 529, no. 7587 (2016): 484--489.

[3] Shanhong, Artifial intelligence software market growth forecat worldwide 2019-2025, 2020

[4] Sutton, R. and Barto, A. (2004) Reinforcement Learning: an Introduction. MIT Press

[5] Sutton et al,Policy Gradient Methods for Reinforcement Learning with Function Approximation, 2000

[6] Szepesvari, Algorithms for Reinforcement Learning, 2009

[7] Mnih et al, Playing Atari with Deep Reinforcement Learning, 2013

[8] Silver et al, Deterministic Policy Gradient Algorithms, 2014

[9] Lillicrap et al, Continuous Control With Deep Reinforcement Learning, 2015

[10] Duan et al, Benchmarking Deep Reinforcement Learning for Continuous Control, 2016

[11] Matheron et al, The problem with DDPG: understanding failures in deterministic environments with sparse rewards, 2019

[12] Jemin Hwangbo et al, Control of a Quadrotor with Reinforcement Learning, 2017

[13] Ming et al, Semi-Active Suspension Control Based on Deep Reinforcement Learning, 2020

[14] W. S. McCulloch, W. Pitts. A logical calculus of the ideasimmanent in nervous aktivity. Bulletin of Mathematical Biophysics, 5:115-133, 1943

[15] Ramachandran, Prajit & Zoph, Barret & Le, Quoc. (2017). Swish: a Self-Gated Activation Function.

[16] Manessi, Franco & Rozza, Alessandro. (2018). Learning Combinations of Activation Functions. 61-66. 10.1109/ICPR.2018.8545362.

[17] Šíma, Jiří a Roman Neruda. Teoretické otázky neuronových sítí. Vyd. 1. Praha: Matfyzpress, 1996. 390 s. ISBN 80-85863-18-9.

[18] Kůrková Věra Kolmogorov's theorem and multilayer neural networks, Neural Networks, Volume 5, Issue 3, 1992, Pages 501-506, ISSN 0893-6080

[19] Li Deng and Dong Yu (2014), Deep Learning: Methods and Applications, Foundations and Trends® in Signal Processing: Vol. 7: No. 3–4, pp 197-387. http://dx.doi.org/10.1561/2000000039

[20] Lillicrap, Timothy & Hunt, Jonathan & Pritzel, Alexander & Heess, Nicolas & Erez, Tom & Tassa, Yuval & Silver, David & Wierstra, Daan. (2015). Continuous control with deep reinforcement learning. CoRR.

[21] Uhlenbeck, G. E, Ornstein, L. S. (1930), On the theory of Brownian Motion". Phys. Rev. 36 (5): 823–841.

[22] Hu, Wei & Xiao, Lechao & Pennington, Jeffrey. (2020). Provable Benefit of Orthogonal Initialization in Optimizing Deep Linear Networks.

Příloha A – obsah přiloženého DVD

a) Matlab skript – algoritmus DDPG

b) Model v Simulinku – matematický model pneumatické pružiny