64th Adv. Beam Dyn. Workshop High-Intensity High-Brightness Hadron Beams HB2021, Batavia, IL, USA JACoW Publishing
ISBN: 978-3-95450-225-7 ISSN: 2673-5571 doi:10.18429/JACoW-HB2021-TUEC4

TEST OF MACHINE LEARNING AT THE CERN LINAC4

V. Kain[*], N. Bruchon, S. Hirlander, N. Madysa,
I. Vojskovic, P. Skowronski, CERN, Geneva, Switzerland
G. Valentino, University of Malta, Msida, Malta

Abstract

The CERN H$^-$ linear accelerator, LINAC4, served as a test bed for advanced algorithms during the CERN Long Shutdown 2 in the years 2019/20. One of the main goals was to show that reinforcement learning with all its benefits can be used as a replacement for numerical optimization and as a complement to classical control in the accelerator control context. Many of the algorithms used were prepared beforehand at the electron line of the AWAKE facility to make the best use of the limited time available at LINAC4. An overview of the algorithms and concepts tested at LINAC4 and AWAKE will be given and the results discussed.

INTRODUCTION AND MOTIVATION

The CERN accelerators generally use a modular control system to deal with the resulting complexity of hundreds or thousands of tuneable parameters. Low level hardware parameters are combined into higher level accelerator physics parameters defined by simulation results. For correction and tuning, low-level feedback systems are available, together with high-level physics algorithms to correct beam parameters based on observables from instrumentation. With this hierarchical approach large facilities like the LHC can be exploited efficiently.

There are still many processes at CERN's accelerators, however, that require additional control functionality. In the lower energy accelerators, models are often not available online or cannot be inverted to be used in algorithms. Sometimes instrumentation that could be used as input for model-based correction is simply lacking. Examples include optimisation of electron cooling without electron beam diagnostics, setting up of multi-turn injection in 6 phase-space dimensions and optimisation of longitudinal emittance blow-up with intensity effects. In recent years numerical optimisers, sometimes combined with machine learning techniques, have led to many improvements and successful implementations in some of these areas, from automated alignment of various devices with beam to optimising different parameters in FELs, see for example [1–6].

For a certain class of optimisation problems, the methods of Reinforcement Learning (RL) can bring further advantages. With RL the exploration time that numerical optimisers inevitably need at every deployment is reduced to a minimum - to one iteration in the best case. In 2019 most of the CERN accelerators were in shutdown to be upgraded as part of the LHC Injector Upgrade project [7]. The new linear accelerator LINAC4 and the proton-driven plasma wakefield test facility AWAKE [8] were, however, operated for part of the year. Taking advantage of recent rapid developments in deep machine learning and RL algorithms, the authors successfully implemented sample-efficient model-free RL for CERN accelerator parameter control and demonstrated its use online for trajectory correction both at the AWAKE facility and at LINAC4. The results were published in [9]. In 2020, trajectory correction with model-based RL could be demonstrated in addition.

This paper is organized as follows. A brief introduction is given to Reinforcement Learning in the domain of accelerator control following [9]. In the experimental section, the problem statements and results of the tests on trajectory correction both for the AWAKE 18 MeV electron beamline and the 160 MeV LINAC4 are given. A short summary of the main outcome of [9] is followed by new results using model-based reinforcement learning. The next steps and potential for wider application are covered in the discussion and conclusion part.

REINFORCEMENT LEARNING FOR ACCELERATOR CONTROL

The optimisation and control of particle accelerators is a sequential decision making problem, which can be solved using RL if meaningful state information is available. In the RL paradigm, Fig. 1, a software Agent interacts with an environment, acquiring the state and deciding actions to move from one state to another, in order to maximise a cumulative reward [10]. The Agent decides which action to take given the current state by following a policy. The goal is to learn the optimal policy for the problem.

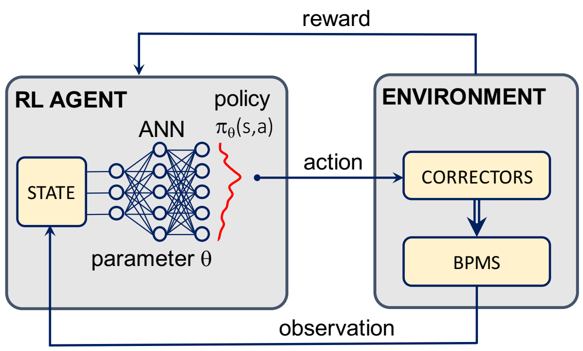

Figure 1: The RL paradigm as applied to particle accelerator control, showing the example of trajectory correction.

Reinforcement Learning algorithms can be divided into two main classes: model-free and model-based. Model-free control assumes no *a priori* model of the environment, learning the system dynamics implicitly from interacting with

[*] verena.kain@cern.ch

the environment, and can be further sub-divided. Policy Optimisation methods consist of policy gradient or actor-critic methods which are more suitable for continuous action spaces. These methods attempt to learn the policy directly using gradient descent.Q-Learning methods seek to find the best action to take given the current state, i.e. the action that maximises the Q-function, and are specifically suitable for discrete action spaces.
The Q-function in RL is a measure of the overall expected (discounted) reward assuming the Agent in state s performs action a and then continues until the end of the episode following a stationary policy π. $\pi(a|s)$ assigns a probability to action a for a given state s. The Q-function and its parametric approximation $Q^{\pi}(s,a|\theta^{Q})$ (θ^{Q} are network parameters), are given as:

$$Q^{\pi}(s,a) = \mathbb{E}\Big[\sum_{i=0}^{N}\gamma^{i} r(s_i,a_i)|\pi, s_0 = s, a_0 = a\Big], \quad (1)$$

$$Q^{\pi}(s,a) \approx Q^{\pi}(s,a|\theta^{Q}). \quad (2)$$

N is the number of states from state $s = s_i$ till the terminal state, $\gamma \in [0,1]$ is a discount factor and $r(s,a) \in \mathbb{R}$ is the reward the agent receives after performing action a in state s.

Within the class of model-free algorithms, policy gradient algorithms are generally less sample-efficient than Q-learning ones and, as *on-policy* methods, cannot take advantage of experience replay [10].

For accelerator applications a continuous action space is usually required. This can be a serious limitation for Q-learning, due to the need to perform the non-trivial maximisation of the Q-function with respect to continuous a. It can however be overcome by assuming a specific algebraic form of the Q-function, such that $Q(a,s)$ is straightforward to optimise with respect to the action. This approach is applied in the Normalised Advantage Function (NAF) algorithm [11].

In model-based methods, the control method uses a predictive model of the environment dynamics to guide the choice of next action. Establishing a reliable enough model is clearly critical to success. Several sub-divisions of this class of RL algorithms exist, including Analytic Gradient Computation methods such as LQR [12], Sampling-Based Planning [13] and Model-Based Data Generation methods such as Sutton's original Dyna [14] and related algorithms using Bayesian methods (e.g. model ensemble techniques) to capture the uncertainty of the model and avoid model bias. An overview of the various algorithms is provided in [15].

For most real-world applications, where the state-space is large, the Q-function or the policy need to be approximated using e.g. neural networks. Advances in the use of deep learning to train such models in the RL paradigm have led to a range of algorithms such as Deep Q-Network (DQN) [16], Deep Deterministic Policy Gradient (DDPG) [17] and Normalized Advantage Function (NAF) [11].

The results in this paper were obtained with the very sample-efficient NAF algorithm, iLQR as well as the DDPG variant TD3 [18] as part of Dyna-style model-based Reinforcement Learning.

Operational Deployment

The CERN accelerator control system offers a python package (pyjapc [19]) to communicate directly with the hardware systems or with the high-level control system for parameters like deflection angle or tune.

For the problem description (environment), the generic OpenAI Gym framework [20] was chosen, to enforce standardisation and allow easy switching between different control problems or Agents. The OpenAI Gym environment python class provides an interface between RL agent and optimisation problem and is used by many available RL algorithm implementations. Gym environments contain all the control problem specific code - the interaction with the machine or simulation for setting the action, reading or calculating observation data as well as calculating or measuring reward.

RL AGENT FOR AWAKE TRAJECTORY CORRECTION

The first RL Agents were trained for trajectory correction on the AWAKE electron line with the goal that the trained Agents correct the line with a similar efficiency as the response matrix based SVD algorithm that is usually used in the control room, i.e. correction to a similar RMS as SVD within ideally 1 iteration.

The AWAKE electrons are generated in a 5 MV RF gun, accelerated to 18 MeV and then transported through a beam line of 12 m to the AWAKE plasma cell. A vertical step of 1 m and 60° bend bring the electron beam parallel to the proton beam shortly before the plasma cell. The trajectory is controlled with 11 horizontal and 11 vertical steering dipoles according to the measurements of 11 beam position monitors (BPMs). The BPM electronic read out is at 10 Hz and acquisition through the CERN middleware at 1 Hz.

The electron transfer line with all of its equipment is modelled in MAD-X [21]. The MAD-X model was used to prepare a simulated OpenAI Gym environment with the purpose to test various RL algorithms offline and define the hyper-parameters for the chosen algorithm for optimum sample efficiency.

Experiment Results from AWAKE RL Tests

The first successful online training of a NAF Agent on trajectory steering in the horizontal plane was obtained on November 22, 2019. The training for 11 degrees of freedom (DOF) took roughly 30 minutes, corresponding to about 350 iterations. At each start of an episode the correctors were reset to the initial setting before the training. A random Δ setting was then sampled from a Gaussian distribution with $\sigma = 300$ µrad for each corrector and added to the initial setting, leading to maximum 7 mm RMS (a factor 2-3 above the normal trajectory distortions caused by drifts and different initial conditions). The maximum step a corrector could do per iteration was set ±300 µrad.

The objective of the training was twofold: to maximise the reward from each initial condition, and to maximise the reward in the shortest possible time. Figure 2 shows the evolution of the 200 episode online training. The upper plot gives the length of the episodes in number of iterations as training evolves, while the lower plot shows the initial reward (i.e. negative RMS) at the beginning of the episode (green line) as well as the final reward achieved (blue line) at the end of each episode. For a successful termination of the episode, the final reward had to be above the target (dashed red line).

At the beginning of the training, the Agent could not correct the line to an RMS below 2 mm, despite many iterations. Instead, the trajectory deteriorated further. After about 15 episodes it had learned to successfully correct the trajectory within 1-2 iterations to mostly even below 1 mm RMS starting from any initial condition.

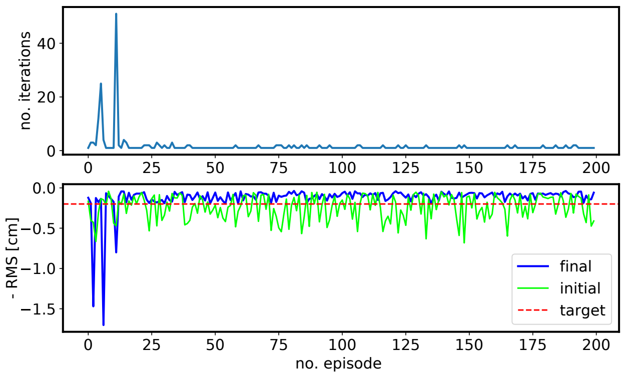

Figure 2: Online training of NAF Agent of AWAKE electron line trajectory steering in the horizontal plane. In the upper plot the number of iterations per episode is given. The lower plot shows the initial and final negative RMS value for each episode. The target negative RMS value is indicated in red.

RL AGENT FOR LINAC4 TRAJECTORY CORRECTION

Training the AWAKE RL Agent for trajectory correction was a test case for algorithm development, since classical optics model-based steering algorithms are available for the AWAKE 18 MeV beamline. The CERN LINACs, on the other hand, did not have online models at the moment of the first RL experiments. RL and numerical optimisation could be obvious and inexpensive solutions to many typical LINAC tuning problems. The 160 MeV LINAC4 provides H^- to the upgraded CERN proton chain through charge exchange injection into the PS Booster [22]. LINAC4 had its final commissioning run at the end of 2019, where some time was also allocated to test various advanced algorithms for different control problems. Also, an RL Agent using the NAF algorithm was trained for trajectory steering in the LINAC exploiting the experience with AWAKE.

LINAC4 accelerates H^- from 3 MeV after source and RFQ to 160 MeV. The Medium Energy Beam Transport (MEBT) after the RFQ is followed by a conventional Drift Tube Linac (DTL) of about 20 m that accelerates the ions to 50 MeV, then to 100 MeV in 23 m by a Cell-coupled Drift Tube LINAC (CCDTL) and finally to 160 MeV by a π-mode structure (PIMS). The total length of the LINAC up to the start of the transfer line to the PSB is roughly 75 m. The pulse repetition rate is 0.83 Hz. The trajectory in the MEBT is fine tuned for optimising chopping efficiency and should not be modified during general trajectory optimisation. In addition there are no BPMs available in the MEBT as observable for an RL Agent.

The LINAC4 Gym environment comprised state information from 17 BPMs and actions possible on 16 correctors, through DTL, CCDTL, PIMS and start of the transfer line in the horizontal plane (the allocated accelerator development time was not sufficient to also train for the vertical plane). The LINAC4 trajectory steering OpenAI Gym environment had to respect the machine protection constraints and finalise episodes in case of violation, reset to safe settings as well as to deal with various hardware limitations (e.g. the power supplies of the steering dipoles cannot regulate for $|I| < 0.1$ A).

Experimental Results from LINAC4 RL Tests

LINAC4 had 8 weeks of final commissioning run in 2019. On November 27, half a day was allocated to training and testing the NAF Agent. The training is shown in Fig. 3. The total number of episodes was set to 90 (taking in total about 300 iterations).

After about 25 episodes (or the equivalent of about 125 iterations), the Agent had learned to correct the trajectory to below 1 mm RMS within a maximum of 3 iterations each time.

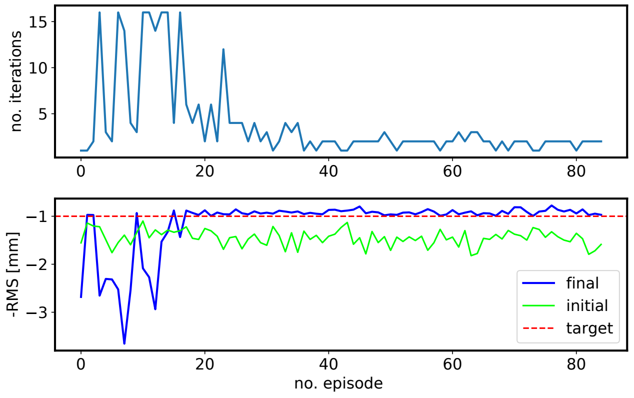

Figure 3: Online training of NAF Agent on LINAC4 trajectory steering in horizontal plane. The maximum allowable RMS was limited to 3 mm due to machine protection reasons. The target for the training was set to reach 1 mm RMS.

MORE SAMPLE-EFFICIENCY WITH RL

Another important application of RL will result from training RL Agents on simulation and then exploiting the Agent with or without additional short training on the accelerator. The obvious advantage of this approach, if possible, is that in this case the algorithm does not have to be restricted to be a very sample-efficient one, as accelerator time for training is either zero or limited. To test this principle of offline training, another NAF Agent was trained - this time on the simulated AWAKE trajectory correction environment. This Agent was then used on the accelerator in operational configuration as trajectory correction algorithm. As expected, the results - maximum 2 iterations for correction to well below 2 mm RMS for each episode - were as good as with the online trained Agent. Figure 4 shows the results.

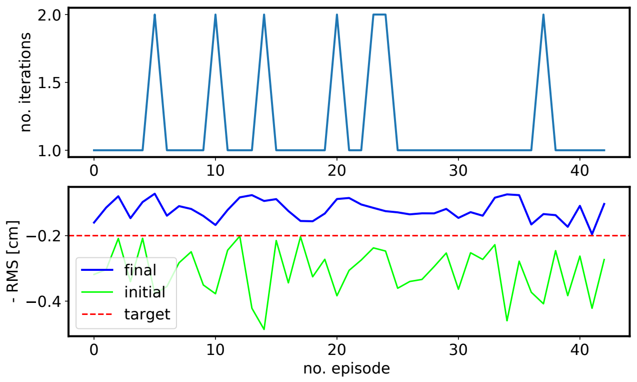

Figure 4: Validation on accelerator of Agent that was trained on simulation. The Agent corrects the trajectory to better than 2 mm RMS within 1-2 iterations.

Experiments with Model-based Reinforcement Learning

Model-based RL is a promising alternative to overcome the sample efficiency limitation. Sutton's Dyna-style algorithm uses supervised learning for explicitly learning the dynamics model and then trains a model-free RL Agent (TD3 in our case) on the learned model instead of the real environment. Improving the model and agent training are interleaved until reaching a termination criterion. The Dyna algorithm deployed for AWAKE trajectory correction (see Algorithm 1) uses a simple fully connected artificial neural network $s_{t+1} = f(s_t, a_t)$ for the dynamics model, avoiding the additional complexity with uncertainty-aware models as used in other model-based RL algorithms (e.g. [23]). This approach was sufficient due to the short horizon and the beforehand reduced stochasticity of the problem.

Algorithm 1 was used with the AWAKE trajectory steering online environment and led indeed to a significant reduction of required data samples to train the Agent. Figure 5 shows the number of data samples for 6 consecutive trainings. The median number of required data samples was $N_{data} \approx 80$ compared to $N_{data} \approx 300$ for the model-free case.

Algorithm 1 The OpenAi Gym environment env$_{model}$ contains the neural network for the data-driven dynamics model and the methods to train this model. N_{init} data samples are collected with random policy at the beginning. Agent training and dynamics model learning is repeated maximum N_{dyna} times. After each agent training, the agent is tested on the real environment according to some performance criterion. In case of successful test, the training is stopped.

```
env$_{model}$ = env$_{model}$(env$_{real}$, s$_0$)
td3 = TD3(env$_{model}$)
fill initial buffer $\mathscr{D}$ with $N_{init}$ samples
env$_{model}$.train_model()
for $N_{dyna}$ do
    td3.learn(N$_{td3}$)
    test on env$_{real}$ with td3.predict($s$)
    add validation data to $\mathscr{D}$;
    if test OK then
        break
    end if
    env$_{model}$.train_model()
end for
```

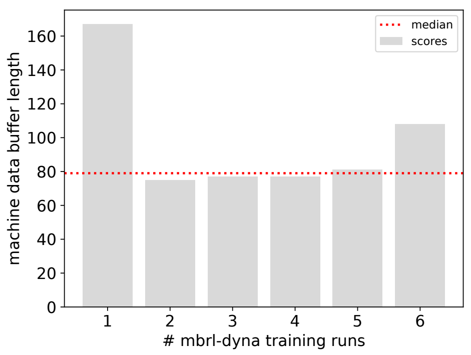

Figure 5: Required number of data samples before obtaining a sufficiently well trained agent using Dyna-style model-based RL following algorithm 1 on the online AWAKE trajectory correction environment.

Another approach of using explicit dynamics models involves model-predictive control algorithms. Instead of training a model-free Agent on the data-driven dynamics model and iterating between model learning and Agent training, the surrogate dynamics model could be used as input to a model-predictive control algorithm like iLQR [12]. The successful deployment of this approach is shown in Fig. 6.

DISCUSSION AND CONCLUSION

The experience with the AWAKE and LINAC4 RL Agent deployment has proved that the question of sample efficiency for RL can be addressed for real accelerator control problems. The training with algorithms such as NAF, TD3 and in particular model-based RL is sample-efficient enough to allow for deployment in the control room. It still requires more iterations than a numerical optimisation algorithm,

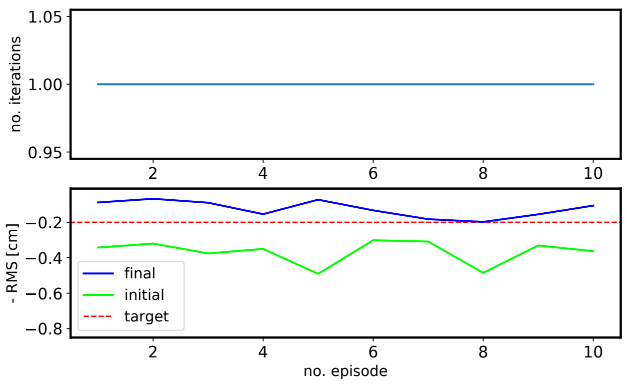

Figure 6: Running iLQR to correct the horizontal trajectory of the AWAKE electron line on a dynamics model trained with 200 data points for various initial trajectories. The problem is solved within 1 step each time.

but after training it out-performs numerical optimisers. The resulting product is a control algorithm like SVD, but in the case of RL without the requirement of a linear response.

The standardisation of the environment description using OpenAI Gym proved a big advantage, allowing rapid switching between simulated and online training, and between Agents.

In addition to studying new algorithms, infrastructure and frameworks will have to be deployed in the control system to easily make use of advanced algorithms and machine learning. At CERN, a generic optimisation framework for the control room was provided, including centrally stored neural networks as well as registering environments and algorithms for common use.

REFERENCES

[1] A. Edelen, N. Neveu, M. Frey, Y. Huber, C. Mayes, and A. Adelmann, "Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems", *Phys. Rev. Accel. Beams*, vol 23, p. 044601, 2020. doi:10.1103/PhysRevAccelBeams.23.044601

[2] G. Azzopardi, A. Muscat, G. Valentino, S. Redaelli, B. Salvachua, "Operational results of LHC collimator alignment using machine learning", Proc. IPAC'19, Melbourne, Australia, pp. 1208-1211, 2019. "Operational Results of LHC Collimator Alignment Using Machine Learning", in *Proc. IPAC'19*, Melbourne, Australia, May 2019, pp. 1208–1211. doi:10.18429/JACoW-IPAC2019-TUZZPLM1

[3] S. Hirlaender, M. Fraser, B. Goddard , V. Kain, J. Prieto, L. Stoel, M. Szakaly, F. Velotti, "Automatisation of the SPS ElectroStatic Septa Alignment", in *Proc. IPAC'19*, Melbourne, Australia, May 2019, pp. 4001–4004. doi:10.18429/JACoW-IPAC2019-THPRB080

[4] J. Duris *et al.*, "Bayesian Optimization of a Free-Electron Laser", *Phys. Rev. Lett.*, vol. 124, p. 124801. doi:10.1103/PhysRevLett.124.124801

[5] A. Hanuka *et al.*, "Online tuning and light source control using a physics-informed Gaussian process Adi", https://arxiv.org/abs/1911.01538/.

[6] M. McIntire *et al.*, 'Sparse Gaussian Processes for Bayesian Optimization", https://www-cs.stanford.edu/~ermon/papers/sparse-gp-uai.pdf/.

[7] E. Shaposhnikova *et al.*, "LHC Injectors Upgrade (LIU) Project at CERN", in *Proc. IPAC'16*, Busan, Korea, May 2016, pp. 992–995. doi:10.18429/JACoW-IPAC2016-MOPOY059

[8] E. Adli, A. Ahuja, O. Apsimon, R. Apsimon, A.-M. Bachmann, D. Barrientos, F. Batsch, J. Bauche, V. B. Olsen, M. Bernardini *et al.*, "Acceleration of electrons in the plasma wakefield of a proton bunch," *Nature*, vol 561, pp. 363–367, 2018. doi:10.1038/s41586-018-0485-4)

[9] V. Kain, S. Hirlander, B. Goddard, F. Velotti, G. Zevi Della Porta, N. Bruchon, G. Valentino, "Sample-efficient reinforcement learning for CERN accelerator control", *Phys. Rev. Accel. Beams*, vol 23, p. 124801, 2020. doi:10.1103/PhysRevAccelBeams.23.124801

[10] R. Sutton, A. Barto, "Introduction to Reinforcement Learning", MIT Press, Cambridge, MA, USA, 2018.

[11] S. Gu, T. Lillicrap, I. Sutskever, S. Levine, "Continuous Deep Q-Learning with Model-based Acceleration", in *Proc. 33rd International Conference on Machine Learning*, New York, NY, USA, 2016, pp. 2829–2838.

[12] H. Kwakernaak, R. Sivan, "Linear Optimal Control Systems", Wiley-Interscience, 1972.

[13] A. Nagabandi, G. Kahn, R. S. Fearing, S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning", arXiv:1708.02596, 2017.

[14] R. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting", *AAAI Spring Symposium*, pp. 151-155, 1991. doi:10.1145/122344.122377

[15] T. Wang *et al.*, "Benchmarking Model-Based Reinforcement Learning", https://arxiv.org/abs/1907.02057v1, 2019.

[16] V. Mnih *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, pp. 529–533, 2015. doi:10.1038/nature14236

[17] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning", in *Proc. ICLR*, 2016. https://arxiv.org/abs/1509.02971/.

[18] S. Fujimoto, H. van Hoof, D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods", arXiv:1802.09477, 2018.

[19] "pyjapc", available at https://pypi.org/project/pyjapc/.

[20] http://gym.openai.com/.

[21] MAD-X documentation and source code available at https://mad.web.cern.ch/mad/.

[22] G. Bellodi,"Linac4 Commissioning Status and Challenges to Nominal Operation", in *Proc. HB'18*, Daejeon, Korea, Jun. 2018, pp. 14–19. doi:10.18429/JACoW-HB2018-MOA1PL03

[23] S. Hirlaender and N. Bruchon, "Model-free and Bayesian Ensembling Model-based Deep Reinforcement Learning for Particle Accelerator Control Demonstrated on the FERMI FEL", https://arxiv.org/abs/2012.09737/.