Online Model-Based and Model-Free Reinforcement Learning in Accelerator Operation with Applications to FERMI FEL

Simon Hirlaender
Univertsity of Salzburg, Salzburg, Austria

Niky Bruchon
University of Trieste, Trieste, IT
(Dated: December 1, 2020)

In this paper, we compare a model-based reinforcement learning to a model-free reinforcement learning method applied to the FERMI FEL system. Both approaches hold tremendous promise, and the principal goal of this paper is to show how reinforcement learning can be utilised on an operational level in a reasonable training time on real accelerator physics problems. In terms of sample-complexity, the model-based approach is better, while the asymptotic performance of the model-free method is slightly superior. The model-based algorithm is done in a DYNA-style using an uncertainty aware model, and the model-free algorithm is based on tailored deep Q-learning. In both cases, the techniques were adapted to show increased noise resistivity as omnipresent in accelerator control problems. Code is released in `https://github.com/MathPhysSim/FERMI_RL_Paper`.

I. INTRODUCTION AND MOTIVATION

In particle accelerator operation, one main goal is to provide stable and reproducible performance. In order to achieve this, several control problems have to be considered simultaneously [1]. Especially if there is no way to model the physics a priori, one might use optimisation techniques as, e.g. derivative-free optimisers (DFOs) [2–6] or model-based optimisations as Gaussian processes [7, 8] to restore or maintain the performance. Another promising way is to apply reinforcement learning (RL) [1, 9–11] which unveils several advantages over optimisation methods:

- It covers a larger class of problems, as RL optimises a sequence of decisions.
- It memorises the problem and does not always begin from zero as a DFO.
- Already existing data can be used.
- Adaption to non-stationary systems.
- The underlying structure of the problem might be deduced.

One demanding aspect using RL is the number of iterations needed to train a controller, since RL methods are known to be data-hungry [12], and a second critical aspect is the robustness of the training in real world problems itself.
In this paper, we present the study carried out to solve the maximisation-problem of the radiation intensity generated by a seeded Free-Electron Laser (FEL) on the Free Electron laser Radiation for Multidisciplinary Investigations (FERMI) at Elettra Sincrotrone Trieste.
Two different algorithm classes were applied successfully to the FERMI FEL problem and reveal different advantages. The critical aspects are addressed, and tailored solutions, generally applicable to problems showing a sample complexity as in accelerator control problems are presented. The paper is organised as follows:

- Description of the problem set-up at FERMI
- Overview of RL
- Design decisions of the implementations used in these studies and theoretical concerns
- Results
- Summary and outlook

II. THE PHYSICAL SET-UP OF THE STUDIED PROBLEM

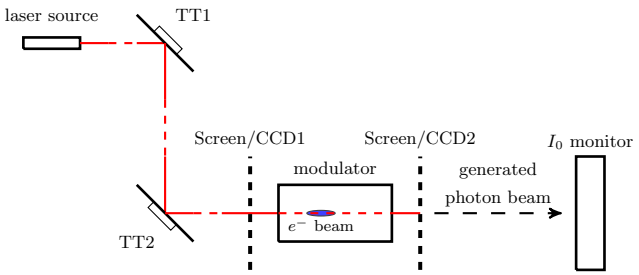

FIG. 1: A schematic view on the set-up of the FERMI FEL.

In a seeded free-electron laser one of the most critical parameters is the temporal and transverse overlap of the electron and laser beam in the magnetic section called modulator undulator.
At FERMI several beam-based feedback systems are deployed to control the beams trajectories shot to shot with the main purpose of guaranteeing a steady intensity of the light radiation. Nevertheless, in the last years, various approaches have been studied to investigate their

applicability in tuning operations.
A free-electron laser is a fourth-generation light source where the lasing medium consists of a very-high-speed electron moving freely through a magnetic structure. The FERMI peculiarity is given by the usage of an external seeding source that provides several advantages, as the increased stability in pulse and photon energy, reduced size of the device, improved longitudinal coherence, more control on the pulse structure, and improved temporal synchronisation with external timing systems.
The external optical laser signal provides is a contribution to the FEL process in the modulator where it interacts with the relativistic electron beam modulating it in energy. The modulation in energy is converted into a charge modulation in the dispersive section, and finally, the density modulated beams radiation is amplified in the section of the radiator. The importance of ensuring the best possible overlapping between the seed laser and the electron beam in the modulator is therefore evident. For this reason, the proposed study focuses on the control of the seed laser trajectory in the modulator, looking at FERMI performance as a reference.

A. The training environment

A schematic overview of the set-up is provided in fig. 1. Two mirrors, TT1 and TT2, are used to control the trajectory of the laser by tilting and inclining, which gives a total of four degrees of freedom (DOF). In turn, the laser overlaps with the electron beam between the two screens, CCD1 and CCD2. Lastly, the monitor measures the intensity, I_0.
The final problem faced consists of optimising the seed laser trajectory to match the electron beam and consequently increase the intensity of the FEL radiation. The RL algorithm accesses the problem via an *openai gym* environment [13], which interfaces the RL algorithm with the hardware control system.
The state $\mathbf{s}$ is a four dimensional vector holding the current voltage values applied to the piezo motors, where each component lies the normalzed interval [-1, 1]. Two values correspond to the two DOF of each mirror. The action $\mathbf{a}$ is four dimensional, is a delta step on the voltages at step t $\mathbf{s}_t$:

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{a}_t. \tag{1}$$

The training is done in episodes with a uniformly randomize initial state and a specific maximal length, called the horizon. For reason as discussed later, the horizon varies in the applications between either 10 or 50 steps. An early termination, when a specific threshold is obtained, lies in all cases at around 95% of the maximal achievable intensity of the system. During the training phase exploration and the training of the function approximator takes place.
After a defined number of training episodes, a verification of 50 episodes is performed. In the second phase, the RL only applies the learned policy.

III. DEEP REINFORCEMENT LEARNING

Assume states $\mathbf{s}$ the set of all states $\mathcal{S}$ and actions $\mathbf{a}$ all actions $\mathcal{A}$ and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a scalar $\gamma \in (0, 1]$ called discount factor and an initial state distribution d_0. $T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ characterises the probability to end in a state $\mathbf{s}_{t+1}$ if an action $\mathbf{a}_t$ is taken at state $\mathbf{s}_t$. The tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, d_0, r, \gamma)$ defines a Markov decision process.
Goal of reinforcement learning is to find a $\pi^* : \mathbf{s} \mapsto \mathbf{a}$, which is the solution of:

$$J(\pi^*) = \max J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{2}$$

where $p_\pi = d_0(\mathbf{s}_0) \prod_{t_0}^{H} \pi(\mathbf{a}_t, \mathbf{s}_t) T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is the distribution of all trajectories $\tau := (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \dots \mathbf{s}_H, \mathbf{a}_H)$ drawn by π with a horizon H. In the modern field of RL one distinguishes if the policy $\pi(\mathbf{a}) \approx \pi_\phi(\mathbf{s})$ or the state-action value function $Q(\mathbf{s}, \mathbf{a})$ is approximated using a high capacity function approximator, as e.g. a deep neural network.
In the first case π is optimized directly and one speaks about policy gradient methods [12, 14–18]. To train these methods, a large number of direct interactions with the system is required, because they are on-policy. In the latter one deals with approximate dynamic programming, which we discuss in section III A. One advantage of the latter approach is, that it can use previously stored data as it is an off-policy algorithm and generally needs less data to be trained. A combination of both is also used, known as actor-critic methods [19].

A. Model free reinforcement learning - MFRL

We focus on the main principles of MBRL. An overview can be found e.g. in [12, 20]. In the following, we discuss the normalised advantage function algorithm in some detail, which has good characteristics for accelerator control problems as shown in [1, 21].

1. *Approximate dynamic programming*

The state value function $V^\pi(\mathbf{s})$ tells us how good a state in terms of the expected return is following a specific π:

$$V^\pi(\mathbf{s}) := \mathbb{E}_{\tau \sim p_\pi(\tau|\mathbf{s}_t)} \left[\sum_{t=t'}^{H} \gamma^{(t'-t)} r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{3}$$

The state-action-value function $Q^\pi(\mathbf{s}, \mathbf{a})$, which tells us how good an action $\mathbf{a}$ in a state $\mathbf{s}$ following π in terms of

the expected return is:

$$Q^{\pi}(\mathbf{s},\mathbf{a}) := \mathbb{E}_{\tau\sim p_{\pi}(\tau|\mathbf{s}_t,\mathbf{a}_t)}\left[\sum_{t=t'}^{H}\gamma^{(t'-t)}r(\mathbf{s}_t,\mathbf{a}_t)\right], \quad (4)$$

is expressed as $Q^{\pi}_{\theta}(\mathbf{s},\mathbf{a})$, where θ denotes the parameters of the function approximator. By satisfying the Bellmann-optimality equation Q^{π}_{θ} can be trained towards the optimal $Q^{*}(\mathbf{s},\mathbf{a})$ by minimizing the Bellman error:

$$\min_{\theta}\left(\vec{Q}_{\theta} - \mathcal{B}^{*}\vec{Q}_{\theta}\right)^2. \quad (5)$$

And π can be calculated via:

$$\pi_{\theta}(\mathbf{s}) = (\delta(\mathbf{a}) - \operatorname{argmax}_{\mathbf{a}} Q_{\theta}(\mathbf{s},\mathbf{a})). \quad (6)$$

The Bellman operator $\mathcal{B}^{*}$ has a unique fixed point but is non-linear, due to the max - operator [12]:

$$\mathcal{B}^{*}\vec{Q}_{\theta}(\mathbf{s}_t,\mathbf{a}_t) := r(\mathbf{s}_t,\mathbf{a}_t) + \gamma\max_{\mathbf{a}}\left(Q_{\theta}(\mathbf{s}_{t+1},\mathbf{a}) - Q_{\theta}(\mathbf{s}_t,\mathbf{a}_t)\right). \quad (7)$$

The form of this equation can cause overestimation and other complications, when using a function approximator and several attempts exist to reduce the problems [22–26].

2. Normalized advantage function

One way to avoid the sensitivity to the mentioned complications is to take a simple analytical form with an explicitly calculable maximum of the Q-function. If a specific quadratic form of the Q function is assumed [25]:

$$Q_{\theta}(\mathbf{s},\mathbf{a}) = -\frac{1}{2}(\mathbf{a}-\mu_{\theta}(\mathbf{s}))P_{\theta}(\mathbf{s})(\mathbf{a}-\mu_{\theta}(\mathbf{s}))^{T} + V_{\theta}(\mathbf{s}). \quad (8)$$

$P_{\theta}(\mathbf{s})$ is a positive semi-infinite matrix. One modification, which is used in these tests is the use of a twin network (weights for network i denoted by θ^{i}), where only one is used to obtain the policy, while the other is used for the update rule to avoid over-estimation. It is motivated by double Q-learning [22, 27, 28]. The maximum is given analytically as $\max_{\mathbf{a}} Q(\mathbf{s},\mathbf{a}) = V(\mathbf{s})$, hence from eq. (5) the loss L can be formulated as:

$$L(\theta) = \left(\left(r(\mathbf{s}_t,\mathbf{a}_t) + \gamma\min_{1,2} V_{\theta^{i}_{\mathrm{targ}}}(\mathbf{s}_{t+1}) - (1+\gamma)Q_{\theta}(\mathbf{s}_t,\mathbf{a}_t)\right)\right)^2 \quad (9)$$

θ_{targ} are the weights of a target network, which is softly updated. To stabilize the network training a small clipped artificial noise is added to the actions called *smoothing* as described in appendix A 1.
This implementation has an extremely good sample-efficiency for suitable problems as can be found often in accelerators. It is used as the baseline for the considered control problem, as it achieves good results. An illustration and additional changes to the original proposal [25] and a previous implementation used for experiments using a prioritized replay buffer [21] are discussed in appendix A 1.

B. Uncertainty aware DYNA-style reinforcement learning

A nice overview of model-based reinforcement learning (MBRL) methods is provided in [29]. Suitable methods are [30, 31], based on Gaussian processes and Bayesian Neural Networks and back-propagation in the dynamics model or [32, 33], using highly efficient sample based methods. We focus on model-based data generation.
In MFRL the dynamics T of an environment is learned implicitly, while in model-based reinforcement learning MBRL T is learned explicitly and approximated as $\hat{T}$. Additionally the reward is included in the model:

$$\hat{f}_{\theta}(\mathbf{s}_t,\mathbf{a}_t) := \{\hat{T}_{\theta}(\mathbf{s}_t,\mathbf{a}_t), r_{\theta}(\mathbf{s}_t,\mathbf{a}_t)\}. \quad (10)$$

Generally, DYNA style algorithms [34] denote algorithms, where an MFRL algorithm is trained on purely synthetic data from an approximate dynamics model $\hat{T}$ or a mixture of synthetic and real data. We use only synthetic data to reduce the interaction with the real environment to a minimum.
An overview of the used method is shown in fig. 2. In the beginning, the data is collected or read from a previously collected data. An uncertainty aware model of the dynamics is trained, using anchored ensembles [35] on the data, which allows taking the aleatoric (measurement errors) as well as the epistemic uncertainty (lack of data) into account. It was motivated by the fact that pure epistemic ensemble techniques as *ME-TRPO* were reported to be very sensitive to aleatoric noise [29]. Alternatives as [32, 33, 36] also take both uncertainties into account using probabilistic ensembles each model holding the parameters of the negative log likelihood of probability distribution function. Our implementation is simpler while showing similar performance.
Subsequently, an MFRL algorithm is deployed to learn the controller on the learned model by only using the synthetic data by taking the ensemble uncertainty into account, as explained in the next section.
After a defined number of training steps, called epoch, the controller is validated on each model individually. If there is no improvement in a number smaller than the total number of the models, the controller might be tested on the real environment for several episodes. In this way 'over-fitting' is avoided, as discussed in [37]. The training is stopped if the controller solves the problem on the real environment. Otherwise, a batch of new data is collected and the model is improved and consecutively a new epoch starts.

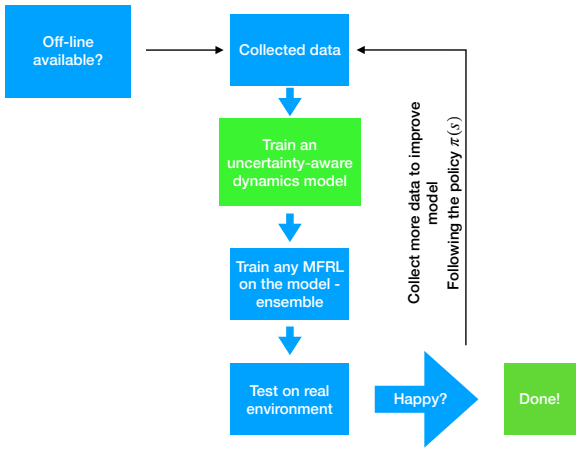

FIG. 2: A schematic overview of the *AE-DYNA* approach used in this paper.

C. Critical design decisions

In the following, the most important aspects of a successful application of model-based RL in accelerator problems are discussed, where we made some algorithmic design decisions which we think are beneficial for problems typically found in accelerator operation.

The anchoring ensemble methods usually yield good results already with a small number of different networks. The idea is to mimic the posterior probability of the dynamics model to capture the uncertainty due to the lack of data. Empirical results showed that already three models were satisfactory to see definite improvements over a single network approach (see fig. 13)). The main goal is not to determine the exact posterior probability of the model, but to provide the uncertainty to not overtrain the controller at areas without sufficient data.

A small two-layer network with around 15-25 nodes with a *tanh* activation and a learning rate of 10^{-3} were used. The last layer was linear, and the inputs $\{\mathbf{s}, \mathbf{a}\}$ were normalised to the interval [-1,1].

The prior is controlled by the initialisation of the weights of the network. It defines how large the variation of the approximated function is expected. The weights of the last layer were normalised by the number of its nodes. The noise level is also respected during the initialisation and has played a crucial role as can be seen in fig. 12. For the moment, only homoscedastic errors are considered.

Several methods of network training were implemented. Early stopping was employed successfully with a waiting time of about 20 steps since the number of training steps is increasing with more data in close-by areas. This increase is followed by a shrinking of the uncertainty at regions where no data is available, leading to better training performance. A fixed maximal loss value threshold was also beneficially tested. In the experiments, a combination of both was taken.

One of the most crucial points is how model uncertainties are taken into account by the MFRL controller. Several different approaches were tried. One way is to take the average of the models and add Gaussian noise levelled by the standard deviation of the models. Another straight forward way is to randomly select a model to provide at each training step, which was showed the best performance the original implementation of the *ME-TRPO* algorithm and was used in the experiments labelled *ME-TRPO*. A pessimistic selection only would select the model resulting in the lowest predicted reward and showed constant but too slow improvement for our tests.

Good results were also obtained by following a randomly selected single model each full episode in combination with the soft-actor-critic (*SAC*) [28, 38], and were used in the experiments labelled *AE-DYNA SAC*.

The number of data-points in a batch taken every time of a new epoch has to be chosen carefully. It is influenced by the number of initial data points, collected using a random policy. The initialisation phase has to be chosen not to small to minimise the chance of getting trapped in a local minimum for too long. Afterwards, from our experience, at least one full episode should be taken. We decided to use a short horizon to diminish the impact of the compound error, the accumulation error following a wrong model, as well known for DYNA-style approaches. The maximal number of steps was ten. During the data collected on the real system, the latest learned policy was taken with some small noise added to improve the exploration.

To decide which MFRL algorithm to use, two main algorithm classes have to be considered: the on and the off-policy algorithms. The online algorithm shows a more stable convergence in general, while off-policy algorithms have the advantage that the data buffer can be filled with real data as well. An attractive off-policy algorithm is *SAC*. The *SAC* not only tries to maximise J eq. (2) but also simultaneously is regularised to maximise the entropy in the action space. In this way, exploration is encouraged to avoid getting stuck in a local optimum and has a good trade-off between exploration and exploitation.

On the other hand, the on-policy algorithm TRPO provides some theoretical improvement guarantees. As reported in [37] using proximal policy optimisation [18] as an online MFRL algorithm does not show the same performance, which was confirmed in our tests. In the latter case, the policy was improved over the whole training. For the *SAC* the policy was reset after retraining the model, to profit from the exploration features.

IV. EXPERIMENTAL RESULTS FROM FERMI RL ONLINE TESTS

As discussed in the previous sections, several tests were performed on the FERMI FEL. The main purpose was

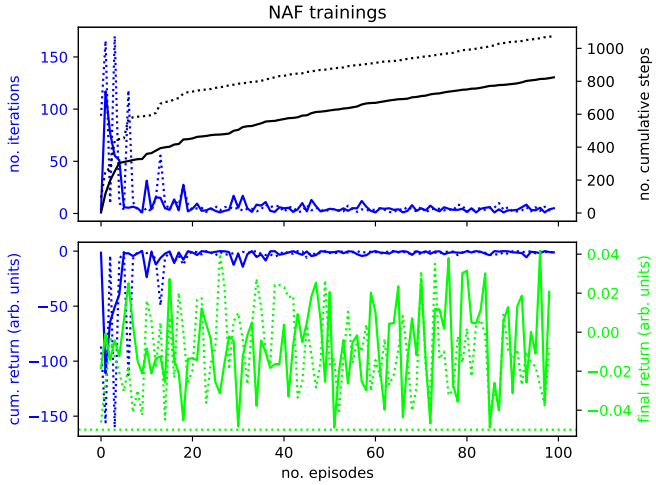

FIG. 3: The training of the *NAF2* on the FERMI FEL.

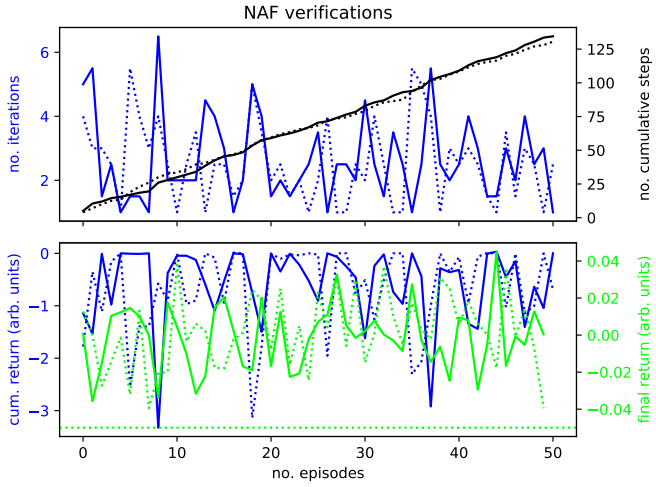

FIG. 4: The verification of the *NAF2* on the FERMI FEL.

to test the newly implemented algorithms on a real system to evaluate its operational feasibility. The *NAF2* algorithm, the representative for highly sample efficient MFRL algorithms, is discussed first.

A. MFRL tests

In total, four tests were carried out, two using a single network and two using the double network architecture. A motivation for the used algorithmic set-up is discussed in appendix A 1 on a classical example from control theory. In both cases *smoothing* to increase the noise robustness as discussed in appendix A 1) was applied.
Figure 3 displays the results, averaged over the two tests. A training of 100 episodes was accomplished. In the upper figure, the number of iterations per episode is plotted,

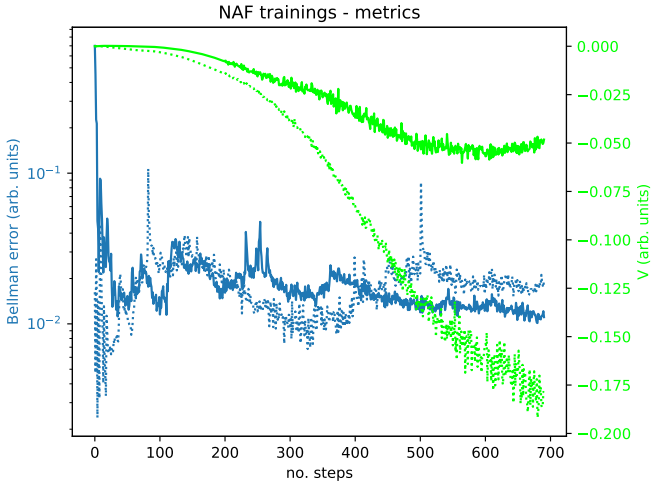

FIG. 5: The verification of the *AE-DYNA-SAC* on the FERMI FEL.

including the cumulative number of steps. The green line in the lower plot of fig. 3 indicates the 95% intensity reward threshold, to accept an episode to be successfully finished. In the verification of 50 episodes (fig. 4) both algorithms show similar performance, while the double network needed less training steps, and reveals a more stable overall performance.
Additionally, the convergence metrics of the two algorithms is plotted in fig. 5 against the number of training steps. The blue curves show the Bellmann error, which is comparable in both cases. The state-value function, which is a direct output of the neural net (eq. (8)), converges to a reasonable value for the double network within the shown 700 steps, whereas the single network seems to overestimate the value. In the single-network case, convergence is reached 1400 steps.

B. MBRL tests

The second test campaign was employing the *AE-DYNA* algorithm, as a representative for pure MBRL algorithms. Two variants were implemented: the *ME-TRPO* variant and the *AE-DYNA SAC* variant. The algorithmic design details are discussed in section III C. The first uses the trust region policy optimisation -*TRPO*- [17] to train the controller. The TRPO monotonically converges to better policy, and this property is exploited in training. The convergence property can be seen in fig. 8. In the upper figure, the total reward per batch and the number of data points are shown as used to train the dynamics model against the number of epochs. In the lower plot, the average cumulative reward as achieved by the TRPO on the individual models in the ensemble on ten episodes as in dependence of the epochs is drawn. During an epoch *TRPO* is trained for 10000 steps on the synthetic data. The shaded area

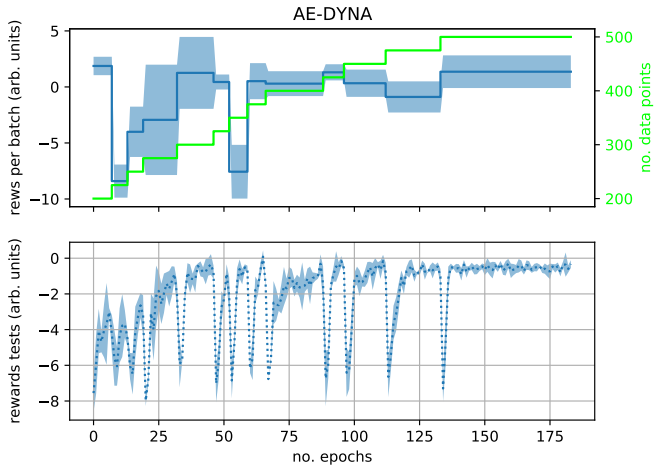

FIG. 6: The training of the *AE-DYNA-SAC* on the FERMI FEL.

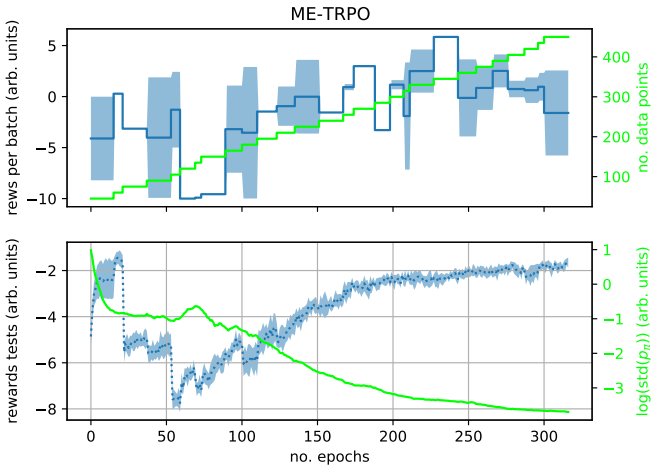

FIG. 8: The training of the ME-TRPO on the FERMI FEL.

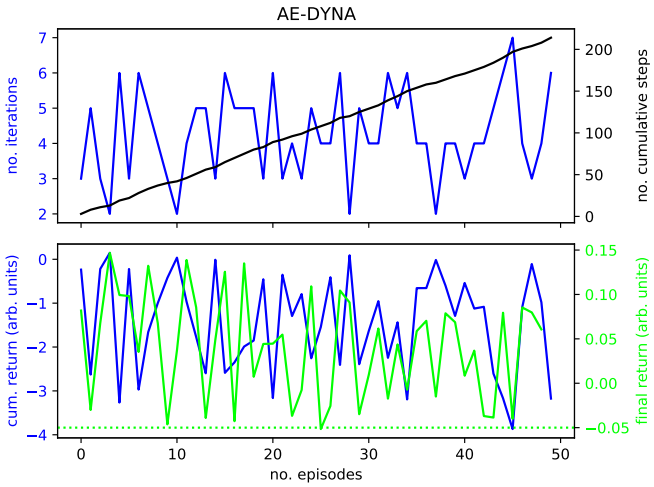

FIG. 7: The verification of the *AE-DYNA-SAC* on the FERMI FEL.

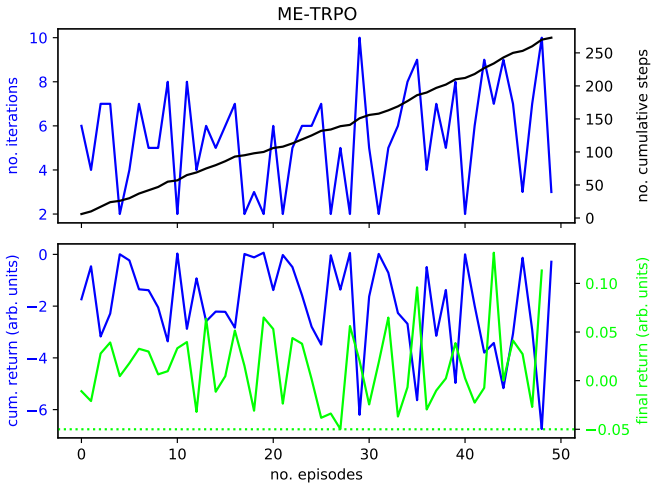

FIG. 9: The verification of the ME-TRPO on the FERMI FEL.

shows the corresponding standard deviation to indicate the uncertainty of the dynamics model. As a measure of convergence of the TRPO, the logarithm of the standard deviation of p_τ is visualised. The training was stopped after 450 steps collecting 25 steps each dynamics training. As shown in fig. 9, all of the 50 verification episodes were successfully finished after a few steps. To verify the impact of the ensemble technique, a test with a single network using the same hyper-parameters has been applied, where no convergence within 500 data points has been observed.

Secondly, the *AE-DYNA SAC* was tested, which uses the *SAC*- algorithm [39]. In this test, the controller is reset each time, when new data for the dynamics model training is acquired. Consequently, the performance drops each time the dynamics model is retrained, as shown in fig. 6. Chances to get trapped in a local optimum are littler by using this strategy of training.

In contrast to the first MBRL test, the new data batches consist of 50 data points with an initial random walk of 200 steps. The number of initial steps was chosen high enough because the convergence is slowed down strongly, so that the training becomes unfeasible on a real machine as discussed in section III C. Each epoch consists of 2500 steps of controller training on the model. The training was stopped after the acquisition of 500 data points. The verification was executed as in the first test. Again the success is 100%, but the number of needed iterations per episodes is less than for the ME-TRPO. It might be a result of the higher number of data points (50), but in general, this method exhibited better asymptotic performance than the $ME-TRPO$ variant.

V. DISCUSSION AND OUTLOOK

In this paper, the applicability of the method of deep reinforcement learning on the optimization of the FERMI FEL intensity was presented. Two different approaches were tested: a model free and a model based method.
Both experiments yielded satisfactory results showing that this non-linear and noisy problem could be solved in a feasibly number of training steps.
Usage of the proposed methods in an operational way is attractive and would replace the current optimization method using the destructive measurement of the screens and additionally would save valuable time.
The model-free methods were slightly better in the final performance but also more sample were taken. The model-based methods had to be stopped due to operational reasons, hence additional studies regarding a long time performance would be interesting. One big issue when applying model-based methods is the computational effort. In our tests the data acquisition time was only a fraction of the time needed to train the controller on the learned model. In case, the used methods could be parallelized to decrease the computational time.

VI. CONCLUSIONS

The used methods hold tremendous promise for automatizing setups common in accelerators. One purpose of this work was to provide some suggestions how to make advanced deep reinforcement learning techniques work on a real set-up by adjusting available methods.
Generally control problems in accelerators have a short horizon, which makes the usage of *DYNA-style* algorithms attractive as the *Me-TRPO* or our *AE-DYNA* attractive choices for highly sample efficient algorithms.
Complementary, the *NAF* algorithms provides a good alternative, revealing as a model-free method good asymptotic performance. As most cases in accelerator control could be covered using the assumption of the quadratic dependence of the actions of the state-action value function, there is a wide range of possible applications.

To provide the possibility for other labs to use the proposed methods the code was released in `https://github.com/MathPhysSim/FERMI_RL_Paper` and [40].

Appendix A: A non-linear standard control problem

To provide some transparency of these studies for other labs, we provide results on a famous classical standard control problem [41], the *inverted pendulum*.
It is a non-linear low dimensional unsolved continuous control problem. Unsolved means there is no threshold for the reward to terminate an episode. The episode length, the horizon, is set to 200 steps. In the following several tests were carried out on the *inverted pendulum* to demonstrate the improvements of the selected algorithms, mainly in terms of noise handling. It is of importance when dealing with measurements on real systems. For statistical significance, all shown results were performed with five different seeds. The average value and the standard deviation (shaded) are plotted. For this study, we assume that the problem is successfully solved, if the cumulative reward surpasses a threshold of -200. The threshold is indicated by a dashed green line in the corresponding figures.

1. NAF2 details

We compare the different *NAF* variants: *Clipping*, *No-clipping-smoothing*, *No-clipping-no-smoothing*, where *clipping* indicates the use of the double network and in all other cases a single network is used. The term *smoothing* indicates that a small clipped noise is added on the actions to stabilize the network training as:

$$\mathbf{a}(\mathbf{s}) = \text{clip}\left(\mu_{\theta_{\text{targ}}}(\mathbf{s}) + \text{clip}(\epsilon, -c, c), \mathbf{a}_{Low}, \mathbf{a}_{High}\right), \tag{A1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$. $c > 0$ denotes the clipping coefficient and $\mathbf{a}_{Low}, \mathbf{a}_{High}$ the minimal and maximal possible action. This method was used already in [28] to improve the deterministic policy gradient [42]. The double network was used in [28, 39] and is done in the following way:

$$y(r_t, \mathbf{s}_{t+1}, d_t) = r_t + \gamma(1 - d_t) \min_{i=1,2} V_{\theta_{i,\text{targ}}}(\mathbf{s}_{t+1}), \tag{A2}$$

with d is one if the episode is finished and 0 otherwise. Then both are learned by regressing to this target (using the tuples from the data buffer $\mathcal{D}$):

$$L(\theta_i, \mathcal{D}) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}, d_t) \sim \mathcal{D}} \left(Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t) - y(r_t, \mathbf{s}_{t+1}, d_t) \right)^2, \quad i \in \{1, 2\} \tag{A3}$$

and the policy is obtained via $\max_{\mathbf{a}} Q_{\theta_1}(\mathbf{s}, \mathbf{a}) = \mu_{\theta_1}$.
The results are shown in fig. 10. One sees the cumulative reward per episode for a training of a total of 100 episodes. As mentioned, the curve labelled *clipping* corresponds to the double network including *smoothing* and shows the best overall stability during the training

yielding quickly a high reward. Also the *smoothed* single network, labelled *No-clipping-smoothing*, shows good and comparable performance, except for the sightly decreased stability. The worst performance is achieved without smoothing and a single network (*No-clipping-no-smoothing*), nevertheless the result is competing with state of the art model free methods as [43] as the benchmark in the *leaderboard* of openai gym [13].

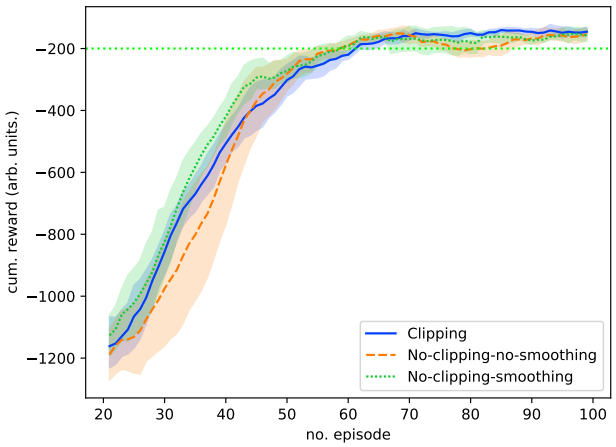

FIG. 10: Cumulative reward of different *NAF* implementations as discussed in the text on the *inverted pendulum* without noise.

2. The impact of noise

A test adding artificial Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma)$ with $\sigma = 0.05$ in the normalized observation space on the states is presented in fig. 11. There the difference of the three methods becomes even more evident.The results are shown in fig. 11. After around 65 episodes the single network without *smoothing* (*No-clipping-no-smoothing*) decreases before reaching the final performance at around 95 episodes, while *smoothing* prevents this performance drop in the other cases.

Using the anchors (the noise level is homeoscetastic) and known in the dynamics model stabilizes the training of the *AE-DYNA* as illustrated in fig. 12. It the upper plot the mean cumulative reward on 10 test episodes on the real environment is shown during the training. It should indicate the result if the training is stopped at this training epoch. One cannot observe this during a real training, unless one does costly performance measurements during the training. Respecting the aleatoric (*Noise-on-aleatoric*) helps to reach the target much quicker exhibiting less variation compared to the standard use of an ensemble (*Noise-on-non-aleatoric*). An epoch consists of 3000 iterations of the *SAC*.

The lower plot of fig. 12 shows the batch rewards a measured during the data collection, which is observable dur-

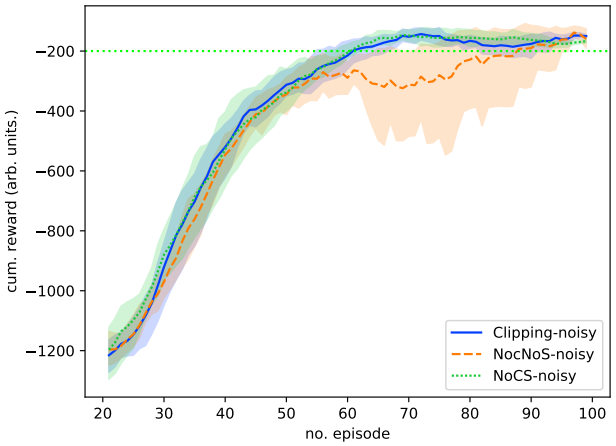

FIG. 11: Cumulative reward of different *NAF* implementations on the *inverted pendulum* with artificial noise as discussed in the text.

ing the training. Figure 13 shows the impact of the num-

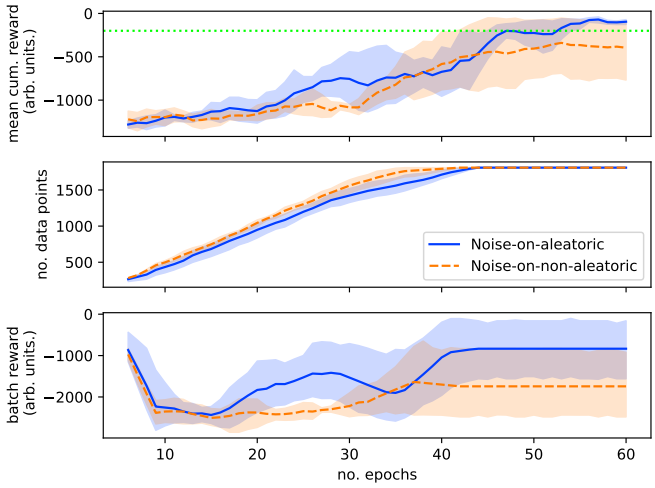

FIG. 12: Cumulative reward of *AE-DYNA* on the *inverted pendulum* with artificial noise using the anchor.

ber of models onto the performance on the *inverted pendulum*. The maximum cumulative reward averaged over five different runs tested on the real environment during the training is visualized in dependence of the number of data points. A number of three models (label *Three*) shows a good trade-off between performance and training time. A single network might not converge or too slowly (label *Single*) and a model of ten shown fast and stable performance in this case (label *Ten*).

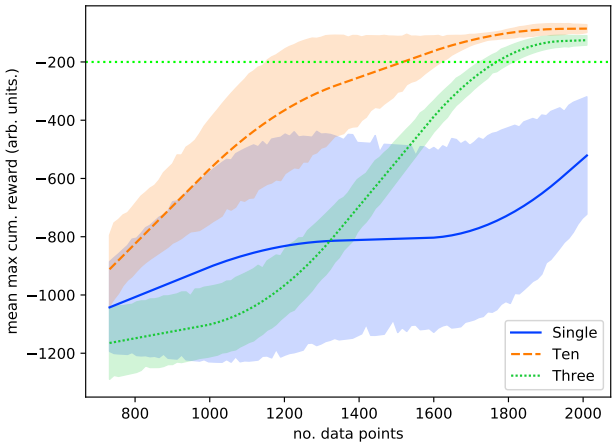

FIG. 13: Varying the number of models in the ensemble of the *AE-DYNA* on the *inverted pendulum*.

3. NAF versus AE-DYNA

Finally, fig. 14 should demonstrate the sample efficiency of the *AE-DYNA-SAC* and the *NAF* algorithm on the noisy *inverted pendulum*. *AE-DYNA* converges below 2000 data points (without noise even below 800) and the *NAF2* starts to perform equally 10000 data points surpassing the -200 reward threshold. One clearly sees the increased sample efficiency on this problem using the *AE-DYNA* in contrast to the *NAF2*.

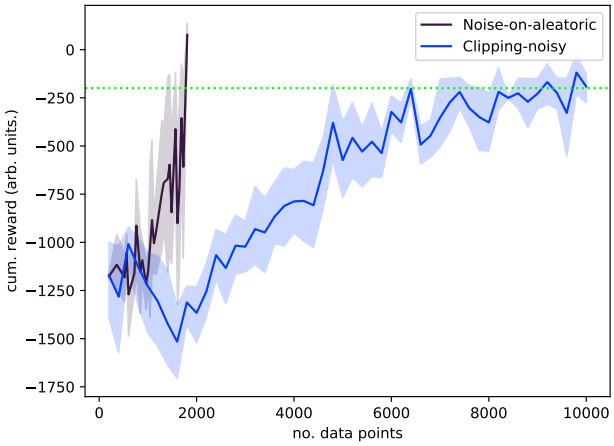

FIG. 14: The comparison of the NAF2 and the *AE-DYNA* on the noisy *inverted pendulum*.

Appendix B: Still open

- reference clean up.

[1] V. Kain, S. Hirlander, B. Goddard, F. M. Velotti, G. Z. D. Porta, N. Bruchon, and G. Valentino, Sample-efficient reinforcement learning for CERN accelerator control, Phys. Rev. Accel. Beams (Accepted 9th November 2020).

[2] X. Huang, J. Corbett, J. Safranek, and J. Wu, An algorithm for online optimization of accelerators, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **726**, 77 (2013).

[3] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. A. Pellegrino, and L. Saule, Free-electron laser spectrum evaluation and automatic optimization, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **871**, 20 (2017).

[4] A. Scheinker, S. Hirlaender, F. M. Velotti, S. Gessner, G. Z. D. Port, V. Kain, and R. Ramjiawan, Online multi-objective particle accelerator optimization of the awake electron beam line for simultaneous emittance and orbit control, 2003.11155.

[5] S. Hirlaender, M. Fraser, B. Goddard, V. Kain, J. Prieto, L. Stoel, M. Szakaly, and F. Velotti, Automatisation of the SPS ElectroStatic Septa Alignment, Proceedings of the 10th Int. Particle Accelerator Conf. **IPAC2019**, Australia (2019).

[6] C. Welsch, Numerical optimization of accelerators within opac, Proceedings of the 6th Int. Particle Accelerator Conf. **IPAC2015**, USA (2015).

[7] A. Hanuka, X. Huang, J. Shtalenkova, D. Kennedy, A. Edelen, V. R. Lalchand, D. Ratner, and J. Duris, Physics-informed gaussian process for online optimization of particle accelerators, 2009.03566.

[8] R. Roussel, A. Hanuka, and A. Edelen, Multi-objective bayesian optimization for accelerator tuning, 2010.09824.

[9] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. H. O'Shea, F. A. Pellegrino, and E. Salvato, Basic reinforcement learning techniques to control the intensity of a seeded free-electron laser, Electronics **9**, 781 (2020).

[10] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. A. Pellegrino, and E. Salvato, Toward the application of reinforcement learning to the intensity control of a seeded free-electron laser, in *2019 23rd International Conference on Mechatronics Technology (ICMT)* (IEEE, 2019).

[11] X. Pang, S. Thulasidasan, and L. Rybarcyk, Autonomous control of a particle accelerator using deep reinforcement learning, 2010.08141.

[12] R. Sutton and A. Barto, *Reinforcement Learning* (MIT Press Ltd, 2018).

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym, 1606.01540.

[14] R. J. Williams, Simple statistical gradient-following algo-

rithms for connectionist reinforcement learning, Machine Learning **8**, 229 (1992).
[15] J. Baxter and P. L. Bartlett, Infinite-horizon policy-gradient estimation 10.1613/jair.806, 1106.0665.
[16] S. Levine and V. Koltun, Guided Policy Search, in *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 28, edited by S. Dasgupta and D. McAllester (PMLR, Atlanta, Georgia, USA, 2013) pp. 1–9.
[17] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, Trust region policy optimization, 1502.05477.
[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, CoRR **abs/1707.06347** (2017), arXiv:1707.06347.
[19] C. Szepesvári, Algorithms for reinforcement learning, Synthesis Lectures on Artificial Intelligence and Machine Learning **4**, 1 (2010).
[20] S. Levine, A. Kumar, G. Tucker, and J. Fu, Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, 2005.01643.
[21] S. Hirlaender, Mathphyssim/per-naf: Initial release (2020).
[22] H. van Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double q-learning, 1509.06461.
[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning, 1312.5602.
[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, 1509.02971.
[25] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, Continuous deep q-learning with model-based acceleration, 1603.00748.
[26] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, Dueling network architectures for deep reinforcement learning, (), 1511.06581.
[27] H. Hasselt, Double q-learning, in *Advances in Neural Information Processing Systems*, Vol. 23, edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Curran Associates, Inc., 2010) pp. 2613–2621.
[28] S. Fujimoto, H. van Hoof, and D. Meger, Addressing Function Approximation Error in Actor-Critic Methods (2018), arXiv:1802.09477 [cs.AI].
[29] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, Benchmarking model-based reinforcement learning, (), 1907.02057v1.
[30] Y. Gal, R. McAllister, and C. E. Rasmussen, Improving PILCO with Bayesian neural network dynamics models, in *Data-Efficient Machine Learning workshop, ICML*, Vol. 4 (2016) p. 34.
[31] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, Gaussian processes for data-efficient learning in robotics and control, IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 408 (2015).
[32] K. Chua, R. Calandra, R. McAllister, and S. Levine, Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 1805.12114.
[33] T. Wang and J. Ba, Exploring model-based planning with policy networks, 1906.08649.
[34] R. S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting, ACM SIGART Bulletin **2**, 160 (1991).
[35] T. Pearce, F. Leibfried, A. Brintrup, M. Zaki, and A. Neely, Uncertainty in neural networks: Approximately bayesian ensembling, 1810.05546.
[36] M. Janner, J. Fu, M. Zhang, and S. Levine, When to trust your model: Model-based policy optimization, 1906.08253.
[37] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, Model-ensemble trust-region policy optimization, 1802.10592.
[38] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, Stable baselines, `https://github.com/hill-a/stable-baselines` (2018).
[39] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, Soft actor-critic algorithms and applications, 1812.05905.
[40] S. Hirlaender and N. Bruchon, .
[41] K. Furuta, M. Yamakita, and S. Kobayashi, Swing up control of inverted pendulum, in *Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation* (IEEE).
[42] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, Deterministic policy gradient algorithms, in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14 (JMLR.org, 2014) p. I–387–I–395.
[43] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap, Distributed distributional deterministic policy gradients, 1804.08617.