Uncertainty Aware Model-Based Reinforcement Learning in Accelerator Operation with Application to FERMI FEL

Simon Hirlaender

University of Salzburg, Salzburg, Austria

Niky Bruchon

University of Trieste, Trieste, Italy

(Dated: December 14, 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 representational power and sample-efficiency, the model-based approach is better, while the asymptotic performance of the model-free method is slightly superior. The model-based algorithm is implemented 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 robustness 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. Achieving this demands to consider several control problems simultaneously [1, 2]. Especially if there is no way to model the physics a priori, one might use optimisation techniques as, e.g. derivative-free optimisers (DFOs) [3–8] or model-based optimisations as Gaussian processes [9, 10] to restore or maintain the performance.
Recently, the community has begun to explore a novel approach to solve control problems. Reinforcement learning (RL) [1, 11–13] 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 is possible.

- The underlying structure of the problem might be deduced.

One demanding aspect using RL is the number of iterations needed to train a controller - the sample-efficiency, since RL methods are known to be data-hungry [14, 15], 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, a modification of the normalized advantage function *NAF* (section III A 2) and a novel anchored ensembling model-based algorithm, the *AE-DYNA* (section III B), 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.

A. An Overview of the Main Results

We demonstrate optimization techniques that simultaneously adjust 4 parameters, the tilt and incline of 2 mirrors that control the trajectory and overlap of the seed laser in the FERMI FEL relative to the beam in the undulator. We show that our algorithms require at most 1000 training sets, after which they are able to maximize the FERMI FEL's output intensity to $>$ that 95% of the maximum within only $\sim$ 3 to $\sim$ 5 steps depending on which algorithm is used. Also we show the operational applicability of purely model-based reinforcement learning in a accelerator control problem.
Furthermore, our algorithms can perform real-time learning based on data and therefore can be re-run and re-trained for time-varying systems. 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
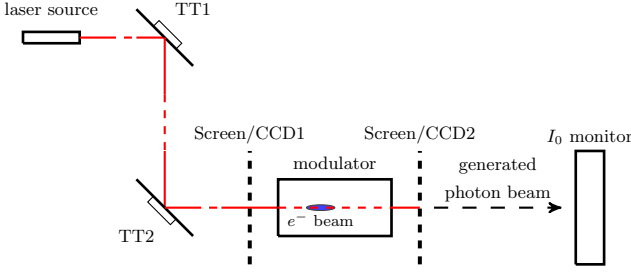
- Results of the experiments

- Summary and outlook

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

II. THE SET-UP OF THE STUDIED PROBLEM

A. The physical set-up

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.

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. By using an external seeding source, the FERMI FEL has several advantages relative to standard FEL approaches in terms of increased stability in pulse and photon energy, reduced size, improved longitudinal coherence, more control on the pulse structure, and improved temporal synchronization with external timing systems.

The external optical laser signal interacts with the relativistic electron beam in the undulator, introducing an energy modulation that aids in the FEL process. 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 overlap 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.

B. 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 [16], which interfaces the RL algorithm with the 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 within the normalized interval [0, 1]. Two values correspond to the two DOF of each mirror. The action $\mathbf{a}$ is four dimensional, namely a delta step on the voltages at time-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[17] randomized initial state and a specific maximal length of taken steps afterwards - the horizon. For reason discussed later, the horizon varies in the applications between either 10 or 500 steps. An early termination happens if a specific threshold is obtained, which lies in all cases at 95% of the maximal achievable intensity of the system. In our experiments the maximal intensity was determined by a manual expert optimization.

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

In what follows, we denote the set of all states $\mathbf{s}$ by $\mathcal{S}$ and the set of all actions $\mathbf{a}$ by $\mathcal{A}$. Relative to state and action vectors, $(\mathbf{s}, \mathbf{a})$, we define 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 $Q(\mathbf{s}, \mathbf{a}) \approx Q_\theta(\mathbf{s}, \mathbf{a})$ 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 [14, 18–22]. To train these methods, a large number of direct interactions with the system is required, because they belong to on-policy algorithms. In the latter one deals with approximate dynamic

programming, which we shortly treat 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 [23–25].

A. Model free reinforcement learning - MFRL

Why Q learning? We focus on the main principles of MBRL. An overview can be found e.g. in [14, 26]. In the following, we discuss the normalised advantage function (NAF) algorithm in some detail, as it has good characteristics for accelerator control problems as its high sample efficiency[1, 27].

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:

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

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 is, in terms of the expected return:

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

is expressed as $Q^\pi_\theta(\mathbf{s}, \mathbf{a})$, where θ denotes the parameters of the function approximator, such as the weights of a neural network. 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:

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

π can be calculated via:

$$\pi_\theta(\mathbf{s}) = \delta(\mathbf{a} - \arg\max_{\mathbf{a}} Q_\theta(\mathbf{s}, \mathbf{a})). \quad (6)$$

The Bellman operator $\mathcal{B}^*$ has a unique fixed point but is non-linear, due to the max - operator [14]:

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

The nature of this equation can cause overestimation and other complications, when using a function approximator and several attempts exist to overcome the difficulties [24, 28–31].

2. Normalized advantage function

One way to avoid the sensitivity to the mentioned complications is to choose a simple analytical form with an explicitly calculable maximum of the Q-function. If a specific quadratic form of the Q function is assumed [30]:

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

$P_\theta(\mathbf{s})$ is a state-dependent, positive-definite square matrix, parametrised in a specific way [30]. One modification, which we made in our experiments is the application of a twin network (weights for network i denoted by θ^i). Only one network is used to obtain the policy, while the other is employed for the update rule to avoid over-estimation. It is motivated by double Q-learning [28, 32, 33]. The maximum is given analytically as $\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) = V(\mathbf{s})$, hence from eq. (5) the loss $\mathcal{L}$ can be formulated as:

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

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. We use it as the baseline for the FERMI-FEL control problem, as it yields good results. Additional changes to the original proposal [30] and a previous implementation used for experiments using a prioritized replay buffer [27] 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 [34]. Suitable methods are [35, 36], based on Gaussian processes and Bayesian Neural Networks and back-propagation in the dynamics model or [37, 38], using highly efficient sample based methods (e.g. the cross entropy method [39]). 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 in our case the reward is included in the model [40]:

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

Generally, DYNA style algorithms [41] 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 previously collected data. An uncertainty aware model of the dynamics is trained, using 'anchored ensembling' [42] 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 [34]. Alternatives as [37, 38, 43] also take these uncertainties into account using probabilistic ensembles. 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 model 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 of the ensemble 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' on a wrong model is avoided, as discussed in [44]. 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.
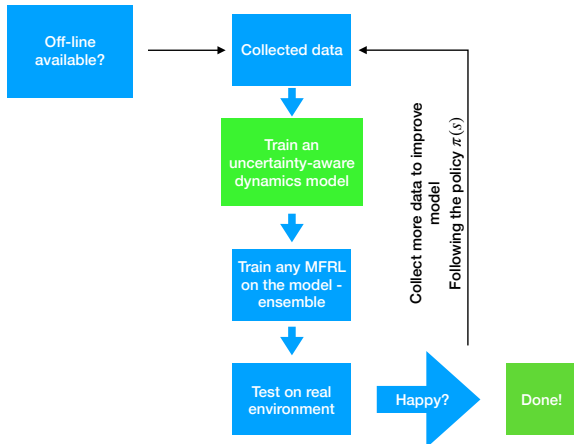

FIG. 2: A schematic overview of the *AE-DYNA* approach used in this paper.

C. Critical design decisions

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

The 'anchored ensembling' method, as treated in appendix A 2, usually yields good results already with a small number of models. Empirical results showed that already three models were satisfactory to see definite improvements over a single network approach (see fig. 15). The main goal is not to determine the exact posterior probability of the model, but, as already mentioned, providing the uncertainty to not over-train the controller at areas without sufficient data.

A small densely connected two-layer network with around 15-25 nodes each and *tanh* activation functions 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. For the moment, only homoscedastic Gaussian errors are considered with a standard deviation is σ_ϵ is also respected in a regularisation term appendix A 2 and improves the training in the presence of noise as can be seen in fig. 14. Several methods of network training were implemented. *Early stopping* [45], with a learning rate of 10^{-3} to 10^{-4} was employed with a waiting time of about 20 steps and a validation ratio of 20%. The number of training steps is increasing with more data in close-by areas. It is followed by a shrinking of the uncertainty at regions where no data is available, leading to better training performance. A fixed maximal loss 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 mean of the models μ_m and the standard deviation σ_m and sample from $\mathcal{N}(\mu_m, \sigma_m)$. Another strategy is to randomly select a model to provide at each training step, which demonstrated the best performance the original implementation of the *ME-TRPO* algorithm and was used in the experiments labelled *ME-TRPO* [44]. A pessimistic selection would only select the model resulting in the lowest predicted reward and showed constant but too slow improvement for our online-training. Good results were also obtained by following a randomly selected single model each full episode in combination with the soft-actor-critic (*SAC*) [33, 46], and were used in the experiments labelled *AE-DYNA-SAC*.

The number of data-points in a new batch to improve the model has to be chosen carefully. It is influenced by the number of randomly collected initial data-points (and π). The initialisation phase has to be selected not to small to minimise chances getting trapped in a local minimum for too long. Afterwards, from our experience, at least one full episode per epoch should be taken. We decided to use a short horizon to diminish the impact of the compound error [43], the accumulation error following a

wrong model, as well known in DYNA-style approaches. The maximal number of steps in our experimental runs was ten. During the data collected on the real system, the latest learned policy was taken with some small additional Gaussian noise to improve 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. 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 [47]. 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 [21] provides some theoretical improvement guarantees. As reported in [44] using proximal policy optimisation [22] 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 *AE-DYNA-SAC* the policy was reset after retraining the model, to profit from its exploration features.

1. Overview of the used algorithms

Table I shows a comparison of the discussed algorithms. Details are provided in the appendix. All algorithms are suitable for many control tasks as common in accelerator operation. Their strength lies in the high sample-efficiency, but they differ in representational power and noise robustness.

In section IV the *NAF2*, *ME-TRPO* and *AE-DYNA* are tested and compared on the real environment.

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 to test the newly implemented algorithms on a real system to evaluate its operational feasibility. Because of the tight schedule at FERMI, only a few shifts of several hours could be reserved to carry out these experiments.

TABLE I: Overview of the algorithms.

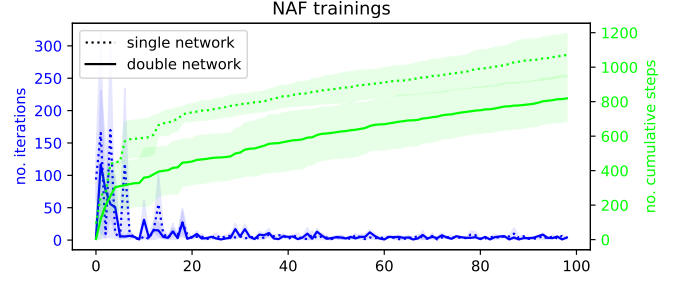| Algorithm	Class	Noise robustness	Represent. power	Sample efficiency
NAF	MFRL	low	low	high
NAF2	MFRL	high	low	high
ME-TRPO	MBRL	low	high	high
AE-DYNA	MBRL	high	high	high

FIG. 3: The training of the *NAF2* on the FERMI FEL using a single network (dashed) and a double network (solid).
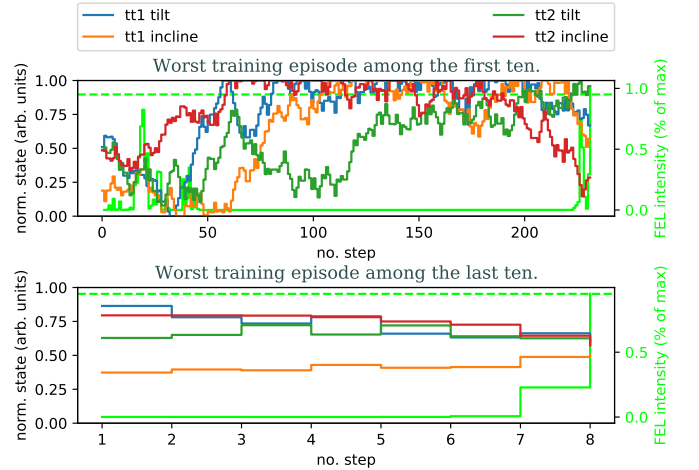

FIG. 4: The training of the *NAF2* on the FERMI FEL using a double network.

Since there was a positive outcome, it is evident, that the utilised methods are at an operational level.

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 given in appendix A 1 on a classical example from control theory, the *inverted pendulum*. In fig. 13 the role *smoothing* to increase the noise robustness can be seen. Therefore *smoothing* was applied in all experiments.

Figure 3 displays the results, averaged over the two tests [48]. A training of 100 episodes was accomplished. The number of iterations per episode is plotted, including the cumulative number of steps. In all episodes during the training the 95% intensity reward threshold was surpassed, and therewith successfully finished. The evolution of the states is provided in fig. 5 and fig. 4 for
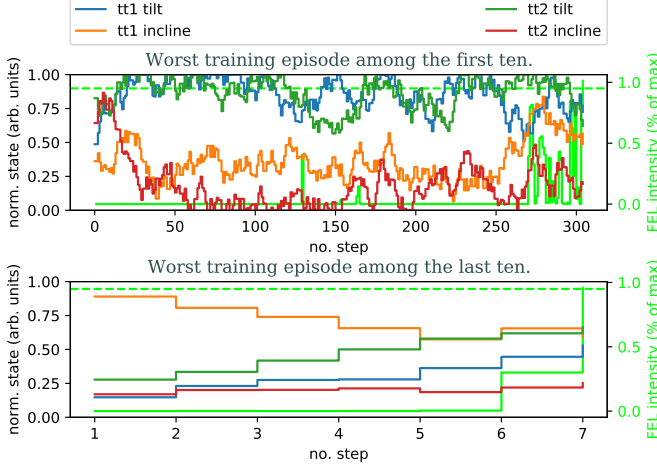
FIG. 5: The training of the *NAF2* on the FERMI FEL
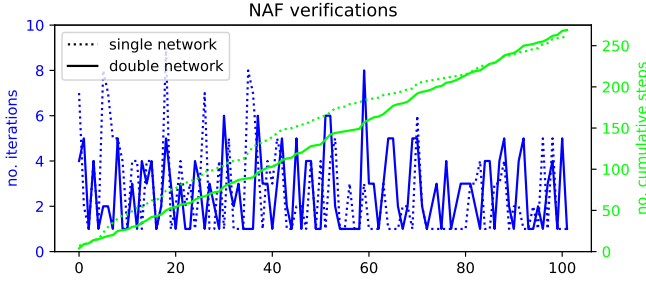using a single network.

FIG. 6: The verification of the *NAF2* on the FERMI
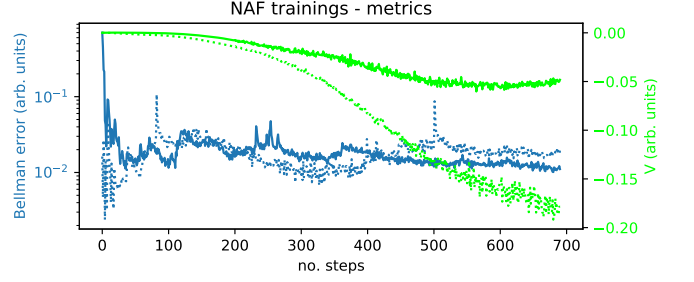FEL using a single network (dashed) and a double
network (solid).

FIG. 7: The training metrics of the *AE-DYNA-SAC* on
the FERMI FEL using a single network (dashed) and a
double network (solid). The Bellman error (eq. (5)) and
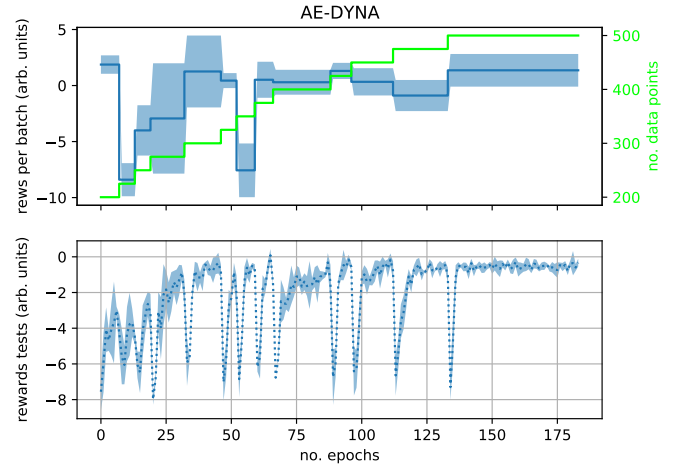the state-value function (eq. (3)) are shown.

FIG. 8: The training of the *AE-DYNA-SAC* on the
FERMI FEL.

the single and double network. At the boundary of the
domain, it takes time to learn, since many actions are
mapped onto the same state[**?**]. The behaviour indi-
cates that the double network converges faster spending
less time at the boundary.

In the verification of 100 episodes (fig. 6) both algo-
rithms show similar performance, while the double net-
work needed less training steps, and reveals a more stable
overall performance.

Additionally, the convergence metrics of the two algo-
rithms is plotted in fig. 7 against the number of training
steps. The blue curves show the Bellmann error eq. (5),
which is comparable in both cases. The state-value func-
tion V (eq. (3)), 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 after around 1400
steps. In the plot V was averaged over 200 randomly
selected states.

B. MBRL tests

The second test campaign was employing the *AE-
DYNA* algorithm, as a representative for pure MBRL al-
gorithms. As discussed, two variants were implemented:
the *ME-TRPO* variant and the *AE-DYNA-SAC* vari-
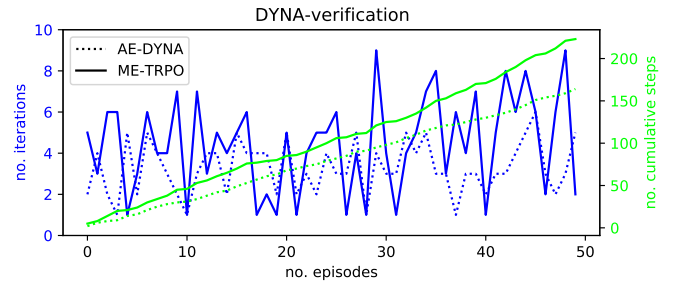

FIG. 9: The verification of the *ME-TRPO* and the
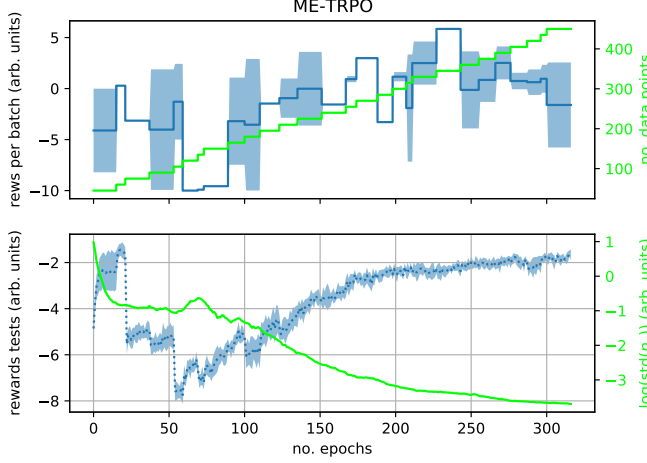*AE-DYNA-SAC* on the FERMI FEL.

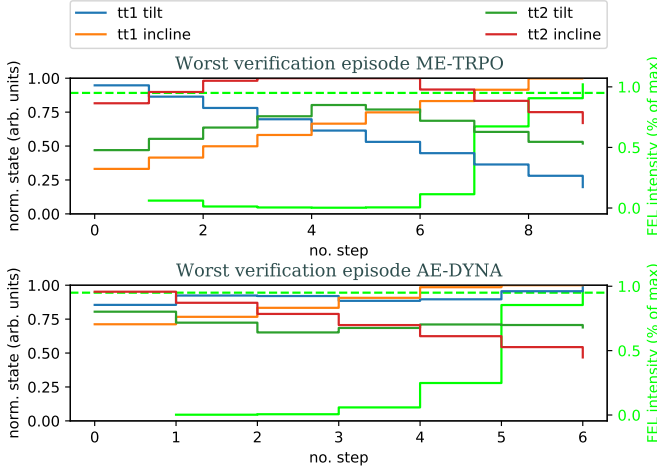FIG. 10: The training of the ME-TRPO on the FERMI FEL.

FIG. 11: The verification of the *ME-TRPO* and the *AE-DYNA-SAC* on the FERMI FEL.

ant. The algorithmic design details are discussed in section III C.

The first uses the trust region policy optimisation - *TRPO*- [21] 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. 10. In the upper figure, the total reward per batch gathered from the real environment 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 the *TRPO* is trained for 10000 steps on the synthetic data. The shaded area 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 [47]. 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. 8.

In contrast to the *ME-TRPO* training, the data batches consisted 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 of all 50 episodes is 100%, but the number of needed iterations per episodes is less than for the ME-TRPO, wich can be seen in fig. 9. The somewhat better performance 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.

In both experiments the training was using a small number of data-points while still successfully solving the problem. in fig. 11 the worst verification episode of both experiments is plotted. Different experiment days influence the boundary conditions and hence the performance. However, a longer training would increase the performance in both cases, especially for the *ME-TRPO*. Figure 11 shows that the tt2 incline and tilt parameters for the *ME-TRPO* move up and then down towards the optimal position, which would be cured by a better dynamics model.

TABLE II: An overview over the verification results of the different algorithms.

	data	nr	cum	final
AE-DYNA	500	3.28 ± 1.26	-1.44 ± 1.10	0.04 ± 0.06
ME-TRPO	450	4.46 ± 2.32	-1.95 ± 1.76	0.01 ± 0.04
NAF	1 074	2.56 ± 1.96	-0.66 ± 1.22	0.00 ± 0.02
NAF2	824	2.64 ± 1.65	-0.57 ± 0.92	0.00 ± 0.03

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 a non-linear and noisy problem could be solved in a feasibly number of training steps. The results of the verification is summarised in table II. In the MFRL, around 800-1000 and in the MBRL 450-500 data-points were used. In all experiments the 50 verification episodes were finished successfully. *NAF* performs better in terms of episode length while the average reward in the *DYNA*-style algorithms is higher. The experiments were done on different days, hence under slightly different conditions.

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 MFRL methods were slightly better in the final performance but also more samples have been collected during the training. The MBRL methods had to be stopped due to lack of available experimental time, hence additional studies regarding a long time performance would be interesting.

One big issue in applying MBRL methods can be the computational time to train the controller on the model. In our tests the data acquisition time was only a fraction of the time needed for the controller-training. In case, the used methods could be parallelized to decrease the computational time.

VI. CONCLUSIONS

The presented methods hold tremendous promise for automatizing set-ups typical in accelerators. One objective of this work was to provide some suggestions on how to make advanced deep reinforcement learning techniques work on a real set-up by adapting available methods.

Regularly, control problems in accelerators have short horizons, which makes *DYNA-style* algorithms as the *ME-TRPO* or our *AE-DYNA* with their high representational power and excellent sample efficiency attractive choices.

Complementary, the *NAF* algorithm presents a good alternative, revealing, as a model-free method, good asymptotic performance. As numerous cases in accelerator control can be treated with the assumption of the quadratic dependence of the actions of the state-action value function, there is a broad spectrum of potential applications.

To provide the possibility for other laboratories to use the stated methods, the code was released in `https://github.com/MathPhysSim/FERMI_RL_Paper` and [49].

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 [50], 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, as introduced in section III A 2 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 [33] to improve the deterministic policy gradient [51]. The double network was used in [33, 47] 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 $\mathcal{D}$):

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

and the policy is obtained via $\max_{\mathbf{a}} Q_{\theta_1}(\mathbf{s}, \mathbf{a}) = \mu_{\theta_1}(\mathbf{s})$. The results are shown in fig. 12. 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 [52] as the benchmark in the *leaderboard* of openai gym [16].
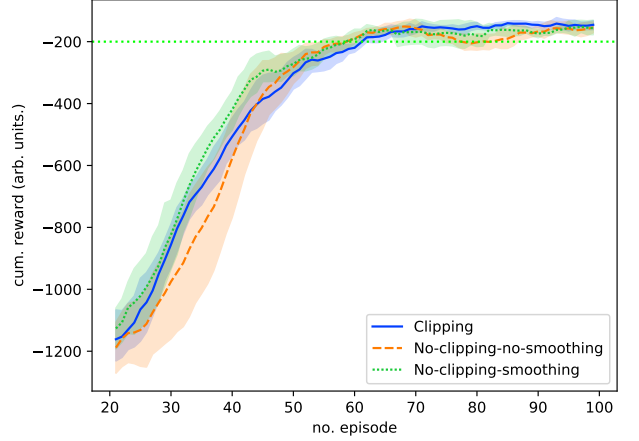

FIG. 12: Cumulative reward of different *NAF* implementations as discussed in the text on the *inverted pendulum* without noise.

2. The impact of noise

A test adding large artificial Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$ with $\sigma_\epsilon = 0.05$ in the normalized observation space on the states is presented in fig. 13. There the difference of the three methods becomes even more evident. The results are shown in fig. 13. 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.

a. Regression assuming homoskedastic Gaussian noise using 'Anchored ensembling'

The Bayesian community and the RL community is spending more and more attention to new approaches to Bayesian inference. In the 'Anchored ensembling' technique a regularisation term is added to the loss function, which returns a point estimate of the Bayesian posterior the maximum a posteriori (MAP) parameter estimate. Using an ensemble of networks and adding noise on either the targets of the regularisation term a distribution
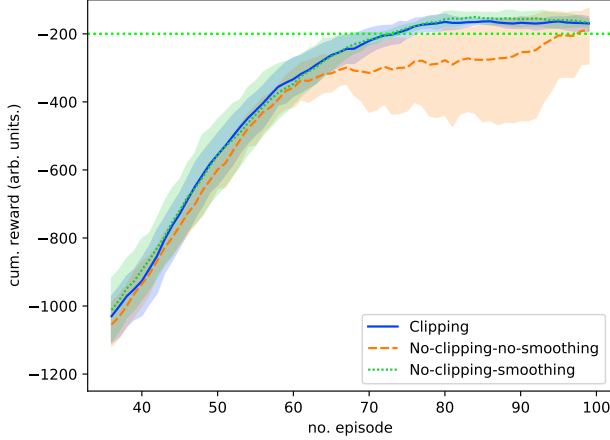
FIG. 13: Cumulative reward of different *NAF* implementations on the *inverted pendulum* with artificial noise as discussed in the text.

of MAP solutions [42, 53–55] is obtained, which mimics true posterior.

Consider an artificial neural net containing parameters, θ, making predictions for the dynamics and the reward eq. (10), $\hat{f}_\theta$ (f denotes the true unknown function), with N data-points. If the prior is given by $P(\theta) = \mathcal{N}(\boldsymbol{\mu}_{prior}, \boldsymbol{\Sigma}_{prior})$, maximising the following returns MAP parameter estimates (details see [42]):

$$\theta_{MAP} = \mathrm{argmax}_\theta \log(P_\mathcal{D}(\mathcal{D}|\theta)) - \frac{1}{2}\|\boldsymbol{\Sigma}_{prior}^{-1/2} \cdot (\theta - \boldsymbol{\mu}_{prior})\|_2^2 \tag{A4}$$

We replace $\boldsymbol{\mu}_{prior}$ with some random variable θ_{anc}. $\theta_{anc} \sim \mathcal{N}(\boldsymbol{\mu}_{prior}, \boldsymbol{\Sigma}_{prior})$, to make it practical and assuming homoskedastic Gaussian noise of variance σ_ϵ^2, we obtain a parametric form of the data likelihood. Taking M models, where each model is indexed by $j \in \{1...M\}$, the MAP estimates are found by minimising ($\mathcal{L}_j$ denotes the loss of the jth model),

$$\mathcal{L}_j = \frac{1}{N}\|f - \hat{f}_{\theta,j}\|_2^2 + \frac{1}{N}\|\boldsymbol{\Gamma}^{1/2} \cdot (\theta_j - \theta_{anc,j})\|_2^2. \tag{A5}$$

$\boldsymbol{\Gamma}$ is diagonal regularisation matrix. The ith diagonal element is the ratio of data noise of the target variable to prior variance for parameter θ_i:

$$\mathrm{diag}(\boldsymbol{\Gamma})_i = \frac{\sigma_\epsilon^2}{\sigma_{prior_i}^2}. \tag{A6}$$

Using the anchors and $\sigma_\epsilon = 0.05$ in the dynamics model stabilizes the training of the *AE-DYNA* as illustrated in fig. 14. In 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. 14 shows the batch rewards a measured during the data collection, which is observable during the training. Figure 15 shows the impact of the num-
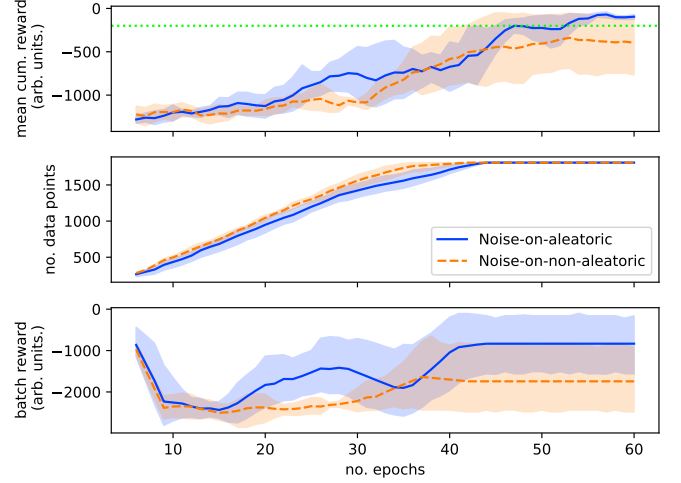

FIG. 14: Cumulative reward of *AE-DYNA-SAC* on the *inverted pendulum* with artificial noise using the anchor.

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*).

3. NAF versus AE-DYNA

Finally, fig. 16 demonstrates the sample-efficiency of the *AE-DYNA-SAC* and the *NAF* algorithm on the noisy *inverted pendulum*. *AE-DYNA-SAC* 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*.

Appendix B: Still open

- Reference and code clean up.
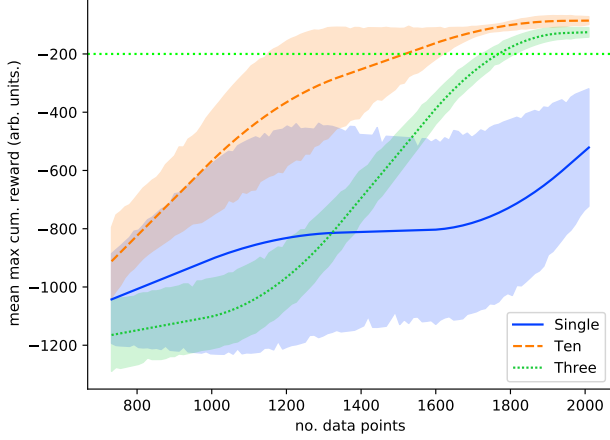
- Conclusion and outlook...

FIG. 15: Varying number of models in the ensemble of the *AE-DYNA-SAC* on the *inverted pendulum*.
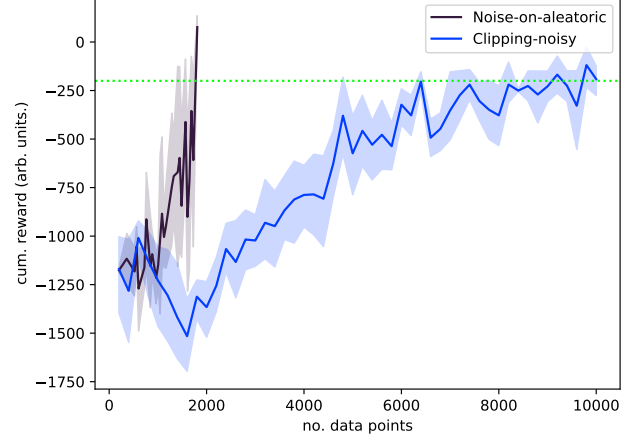

FIG. 16: The comparison of the *NAF2* and the *AE-DYNA-SAC* on the noisy *inverted pendulum*.

- Clean discussion of the results. What is superior and why? A table with numbers ...

- Add the noise policy of NAF

- Check the plots, which might be already okay!

- Acknowledgements

[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, Physical Review Accelerators and Beams **23**, 10.1103/physrevaccelbeams.23.124801 (2020).

[2] A. Scheinker, A. Edelen, D. Bohler, C. Emma, and A. Lutman, Demonstration of model-independent control of the longitudinal phase space of electron beams in the linac-coherent light source with femtosecond resolution, Physical Review Letters **121**, 10.1103/physrevlett.121.044801 (2018).

[3] 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).

[4] 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).

[5] A. Scheinker, S. Hirlaender, F. M. Velotti, S. Gessner, G. Z. D. Porta, V. Kain, B. Goddard, and R. Ramjiawan, Online multi-objective particle accelerator optimization of the AWAKE electron beam line for simultaneous emittance and orbit control, AIP Advances **10**, 055320 (2020).

[6] 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).

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

[8] S. Albright, R. Alemany Fernandez, M. E. Angoletta, H. Bartosik, A. Beaumont, G. Bellodi, N. Biancacci, M. Bozzolan, M. Buzio, F. Di Lorenzo, A. Frassier, D. Gamba, S. Hirlander, A. Huschauer, V. Kain, G. Kotzian, D. Kuchler, A. Latina, T. Levens, E. Mahner, E. Manosperti, O. Marqversen, D. Moreno Garcia, D. Nicosia, M. O'Neil, E. Ozturk, A. Saa Hernandez, R. Scrivens, S. Jensen, G. A. Tranquille, C. Wetton, and M. Zampetakis, Review of LEIR operation in 2018, (2019).

[9] 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.

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

[11] 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).

[12] 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).

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

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

[15] G. Dulac-Arnold, D. Mankowitz, and T. Hester, Challenges of real-world reinforcement learning, 1904.12901.

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

[17] In the normalized state-space.

[18] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning **8**, 229 (1992).

[19] J. Baxter and P. L. Bartlett, Infinite-horizon policy-gradient estimation 10.1613/jair.806, 1106.0665.

[20] 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.

[21] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, Trust region policy optimization, 1502.05477.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, CoRR **abs/1707.06347** (2017), arXiv:1707.06347.

[23] C. Szepesvári, Algorithms for reinforcement learning, Synthesis Lectures on Artificial Intelligence and Machine Learning **4**, 1 (2010).

[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] 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 Machine Learning*, Proceedings of Machine Learning Research, Vol. 32, edited by E. P. Xing and T. Jebara (PMLR, Bejing, China, 2014) pp. 387–395.

[26] S. Levine, A. Kumar, G. Tucker, and J. Fu, Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, 2005.01643.

[27] S. Hirlaender, Mathphyssim/per-naf: Initial release (2020).

[28] H. van Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double q-learning, 1509.06461.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning, 1312.5602.

[30] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, Continuous deep q-learning with model-based acceleration, 1603.00748.

[31] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, Dueling network architectures for deep reinforcement learning, (), 1511.06581.

[32] 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. Cu-

lotta (Curran Associates, Inc., 2010) pp. 2613–2621.

[33] S. Fujimoto, H. van Hoof, and D. Meger, Addressing Function Approximation Error in Actor-Critic Methods (2018), arXiv:1802.09477 [cs.AI].

[34] 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.

[35] 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.

[36] 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).

[37] K. Chua, R. Calandra, R. McAllister, and S. Levine, Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 1805.12114.

[38] T. Wang and J. Ba, Exploring model-based planning with policy networks, 1906.08649.

[39] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, A tutorial on the cross-entropy method, Annals of Operations Research **134**, 19 (2005).

[40] In some applications r can be deduced from the current state or the state change. In such a situation the modelling can be simplified to model directly $\hat{T}$.

[41] R. S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting, ACM SIGART Bulletin **2**, 160 (1991).

[42] T. Pearce, F. Leibfried, A. Brintrup, M. Zaki, and A. Neely, Uncertainty in neural networks: Approximately bayesian ensembling, 1810.05546.

[43] M. Janner, J. Fu, M. Zhang, and S. Levine, When to trust your model: Model-based policy optimization, 1906.08253.

[44] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, Model-ensemble trust-region policy optimization, 1802.10592.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) http://www.deeplearningbook.org.

[46] 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).

[47] 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.

[48] Considering the training is stochastic with random initial conditions, by chance, the training of the double network variant started above the threshold in both cases.

[49] S. Hirlaender and N. Bruchon, Mathphyssim/fermi_rl_paper: Initial release (2020).

[50] 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).

[51] 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.

[52] 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.

[53] Y. Gu and D. S. Oliver, An iterative ensemble kalman filter for multiphase fluid flow data assimilation, SPE Journal **12**, 438 (2007).

[54] Y. Chen and D. S. Oliver, Ensemble randomized maximum likelihood method as an iterative ensemble smoother, Mathematical Geosciences **44**, 1 (2011).

[55] J. M. Bardsley, MCMC-based image reconstruction with uncertainty quantification, SIAM Journal on Scientific Computing **34**, A1316 (2012).