Model based and model free reinforcement learning at FERMI FEL

Simon Hirlaender
Univertsity of Salzburg, Salzburg, Austria

Niky Bruchon
University of Trieste, Trieste, IT
(Dated: November 24, 2020)

In this paper we discuss a model based reinforcement learning approach in comparison to a model free reinforcement learning approach applied at the FERMI FEL system. Both algorithms and approaches are new in this context and the main purpose of this paper is to show how reinforcement learning can be applied on an operational level in a feasible training time on real accelerator physics problems. In terms of sample-complexity the model-based approach is faster, while the final performance of the model free method is superior - the so called asymptotic performance. 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 using some tricks to increase the sample efficiency.

- Noise benchmarks with pendulum
- Benchmark NAF2 with pendulum
- The first time model based on accelerator problem compared to a novel state of the art MFRL: NAF2
- Tailored to accelerators - short horizons are standard?
- Improvements of the algorithms:
  - Adding the noise feature
  - Jumping feature using SAC
  - Noise in the policy
  - discuss different approaches to e.g. dynamic waiting time

I. INTRODUCTION AND MOTIVATION

In particle accelerators one main goal is to provide stable and reproducible performance. In order to achieve this, a number of control problems have to be considered simultaneously. Especially if there is no way to model the physics, one might use optimisation techniques as e.g. derivative free optimisers (DFOs) or model-based optimisations as Gaussian processes to restore or maintain the performance. Another promising way is to apply reinforcement learning which shows several advantages over optimisation methods:

- It covers a larger class of problems, RL optimises a sequence of decisions.
- It memorizes the problem and does not start always from scratch as an DFO.
- Existing data can be used.
- The underlying structure of the problem might be deduced.

On critical aspect using RL is the number of iteration to train a controller. 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. Three algorithms were applied successfully showing different advantages, which will e discussed in the following.

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.

II. THE PHYSICAL SET-UP

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. The paper is organised as follows.

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

III. DEEP REINFORCEMENT LEARNING

Assume states $\mathbf{s}$ the set of all states $\mathcal{S}$ and actions $\mathbf{a}$ all actions $\mathcal{A}$ and rewards r and the set of all rewards

$\mathcal{R}$ an initial state distribution d_0. Goal of reinforcement learning is to find a $\pi : \mathbf{s} \mapsto \mathbf{a}$, which is the solution of:

$$\max J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{1}$$

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 $\tau := (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots \mathbf{s}_H, \mathbf{a}_H)$ 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 one speaks about policy gradient methods in the latter about approximate dynamic programming, which we now discuss.

A. Approximate dynamic programming

The state-value function $Q(\mathbf{s}, \mathbf{a})$ is expressed as $Q_\theta(\mathbf{s}, \mathbf{a})$, where θ denotes the parameters of the function approximator. By satisfying the Bellmann-optimality equation Q_θ can be trained towards the optimal $Q^*(\mathbf{s}, \mathbf{a})$:

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

And π can be calculated via:

$$\pi_\theta(\mathbf{s}) = (\delta(\mathbf{a}) - \operatorname{argmax}_{\mathbf{a}} Q_\theta(\mathbf{s}, \mathbf{a})). \tag{3}$$

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

$$\mathcal{B}^* \vec{Q}_\theta(\mathbf{s}_t, \mathbf{a}_t) := r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} \left(Q_\theta(\mathbf{s}_{t+1}, \mathbf{a}) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \right). \tag{4}$$

The form of this equation can cause overestimation and other complications, when using a function approximator. Several methods exist which try to mitigate the problems []. One way to reduce the effect is to take a simple analytical form of the Q-function.

B. Normalized advantage function

If a specific quadratic form of the Q function is assumed:

$$Q_\theta(\mathbf{s}, \mathbf{a}) = -\frac{1}{2} (\mathbf{a} - \mu_\theta(\mathbf{s})) P_\theta (\mathbf{a} - \mu_\theta(\mathbf{s}))^T + V_\theta(\mathbf{s}). \tag{5}$$

One modification, which is used in these tests is 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 [2]. The maximum is given analytically as $\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) = V(\mathbf{s})$, hence from eq. (2):

$$\min_\theta \left((r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \min_{1,2} V_{\theta^i_{\text{targ}}}(\mathbf{s}_{t+1}) - (1+\gamma) Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \tag{6}$$

θ_{targ} are the weights of a target network, which is softly updated. To stabilize the network training a small artificial noise is added to the actions in the update. This algorithm has an extremely good sample-efficiency for suitable problems as can be found often in accelerators and is used as the baseline for the considered control problem, as it shows very good results.

Algorithm 1: On-policy policy gradient with Monte Carlo estimator

initialise θ_0 **for** *iteration* $k \in [0, \ldots, K]$ **do**

  sample trajectories $\{\tau_i\}$ by running $\pi_{\theta_k}(\mathbf{a}_t|\mathbf{s}_t)$

▷ each τ_i consists of $\mathbf{s}_{i,0}, \mathbf{a}_{i,0}, \ldots, \mathbf{s}_{i,H}, \mathbf{a}_{i,H}$

IV. UNCERTAINTY AWARE DYNA-STYLE REINFORCEMENT LEARNING

The original Dyna algorithm [3] is modified here in several aspects. Generally, Dyna style algorithms [4] denote algorithms, where a MFRL algorithm is trained on purely synthetic data from an approximate dynamics model or on 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. 1. At the beginning the data is collected or read in from a pre-collection. An uncertainty aware model is trained, using anchored ensembles [5] on the data, which allows to take the allegorical (measurement errors) as well as the epistemic uncertainty (lack of data) into account. Subsequently a MFRL algorithm is deployed to learn the controller on the learned model by only using the synthetic data, by taking the uncertainty into account. After a defined number of training steps the controller is testes on each model individually. If there is no improvement in a specific ratio< 1 of models, the controller is tested on the real environment for a number of episodes. The training is stopped if the controller solves the problem on the real environment.

A. Critical design decisions

In the following the most important aspects of a successful application of a Dyna-AE algorithm are listed:

- The ANN - including the prior - number of models - noise level - early stopping
- The number of data points and the policy.

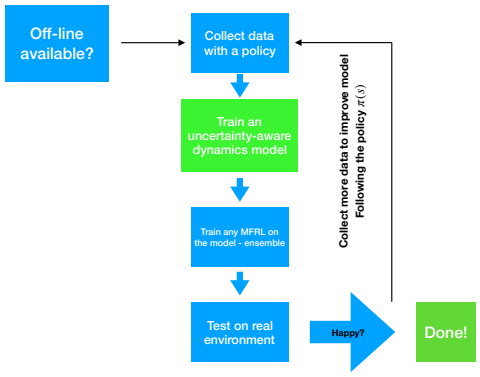

FIG. 1: A schematic overview of the AE-Dyna approach used in this paper.

- The uncertainty - how is this included?
- The episodic design - avoid long trajectories.
- The MFRL agent as e.g. the TRPO and PPO or the SAC and its training
- Tuning the algorithm on a model.

We try to discuss all items in some detail in the following. The anchoring ensemble methods usually yields good results already with a small number of different networks. The idea behind it, is to mimic the posterior probability of the dynamics model to capture first of all its own uncertainty due to the lack of data. Empirical results show that three to five models were sufficient to see clear advantages over a single network approach and the main goal is not to calculate the exact posterior of the model. A small two-layer network with around 15-25 nodes and the *tanh* activation were used. The last layer was linear and the inputs were normalized to the interval [-1,1]. The prior is controlled by the initialization of the weights, which were normalized by the number of nodes in each layer. The noise level is also respected during the initialization. For the moment only homeoscetastic errors are used. Several methods to set the training method were implemented. Early stopping, a standard techniques showed good results, mainly to the fact, that the number of training steps is increasing with more data. The advantage is that the uncertainty at regions without data shrink in this way leading to a better training performance. A fixed loss value threshold was also tested, as also used in the original anchore-ensemble technique. In the experiments a combination of both was taken.
One of the most crucial points is how this uncertainties are taken into account by the RL algorithms. Several different approaches were tried. It is possible to take the average of the models and add Gaussian noise leveled by the standard deviation of the models. Another straight forward way is to randomly select a model to provide at each training step, which is the original implementation of the ME-TRPO algorithm and was used in the experiments labeled *ME-TRPO*. A pessimistic setting only would select the model resulting in the lowest predicted reward. Good results were obtained by following a randomly selected single model each full episode and were used in the experiments labeled *AE-Dyna*.
The number of data-points taken every time the model as improved was firstly determined by the number of initial data points, collect using a random policy. The initialization phase has to be chosen not to small to minimize the chance of getting trapped in a local minimum for too long. Afterwards at least on full episode should be taken. We decided to use a short total episode length to reduce 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 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. Online algorithm show 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. A very attractive off-policy algorithm is the Soft-actor-critic. This algorithm tries to maximize the entropy of the actions to find a good trade-off between exploration and exploitation. On the other hand the on-policy algorithm TRPO provides some theoretical improvement guarantees. In the latter case the policy was improved over the whole training, while for the SAC the policy was reset, to take advantage of the exploration features.

B. Operational deployment

Niky

C. short FERMI introduction

The Free Electron laser Radiation for Multidisciplinary Investigation better known as the FERMI, is the seeded free-electron laser facility next to the third-generation synchrotron facility ELETTRA at the Elettra Sinchrotrone Trieste laboratories. A free-electron laser is a fourth-generation light source where the lasing medium consists of 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 synchronization with external timing systems.

The external optical laser signal provide is 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 the converted into a charge modulation in the dispersive section, and finally the density modulated beams radiation is amplified in the radiators section. 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.

D. our system: the modulator and the seed laser

A more detailed description of the alignment process in the modulator is here provided. The most critical parameters in a seeded free-electron laser are the temporal and transverse overlap of the electron and laser beams in the modulator magnetic section. The problem is simplified by keeping constant the trajectory of the electron beam and the mechanical delay line that controls the temporal alignment. The final problem faced consists in optimising the seed laser trajectory to match the electron beam and consequently increase the intensity of the FEL radiation.

V. PROOF-OF-PRINCIPLE APPLICATION OF RL AGENT AT FERMI TRAJECTORY CORRECTION

Niky

A. Experiment results from FERMI RL tests

As discussed in the previous sections, several tests were performed on the FERMI XFEL. The main purpose was to test the newly implemented algorithms on a real system to evaluate their operational feasibility. The *NAF2* algorithm, as a representative for highly sample efficient MFRL algorithms, was tested first.

B. MFRL tests

In total four tests were carried out, two using a single network and two using the double network architecture. In both cases *smoothing* was applied, as described in appendix A 1.
Figure 2 displays the results, averaged over the two test. A training of 100 episodes (shades area) is followed by a verification of 50 steps. in the upper figure the number of iterations per episode is plotted including the cumulative number of steps. In the verification phase both algorithms show a similar performance, while the double network needs less training steps, and reveals a more stable overall performance.
Additionally the convergence metrics of the two algorithms is plotted in fig. 3 against the number of training steps. The blue curves shows the Bellmann error, which is comparable. The state-value function, which is a direct output of the neural net (eq. (5)), converges to a reasonable value for the double network within the shown 700 steps, whereas the single network tends to overestimate the value. In this case convergence is reached 1400 steps.

C. MBRL tests

The second test campaign was employing the *AE-DYNA* algorithm, as representative for pure a MBRL algorithms. Two variants were implemented: the *ME-TRPO* variant and the *AE-DYNA SAC* variant. In both cases an ensemble of three network was used and the epistemic and aleatoric uncertainty is measured via the anchor-ensemble technique as discussed in section IV A. On top early stopping was used.
The first uses the trust region policy optimization -*TRPO*- [6] to train the controller. The TRPO monotonically converges to better policy and this property is exploited in the training. The convergence property can be seen in fig. 7. In the upper figure the total reward per batch is shown, as well as the number of data points used in the training of the dynamics model. In the lower plot the average cumulative reward as achieved by the TRPO on the individual models in the ensemble on a number of 10 episodes is drawn. The shade area shows the corresponding standard deviation to indicate the uncertainty of the dynamics model. To measure the convergence of the TRPO the logarithm of the standard deviation of the distribution drawn trajectories p_τ is also visualized. Here the training was stopped after 450 steps acquiring 25 steps each dynamics training. In the verification, as can be seen in fig. 8, all of the 50 episodes where successfully finished after a few steps.
Secondly, the *AE-DYNA SAC* was tested, which uses the soft actor critic -SAC- algorithm [7]. The SAC not only tries to maximise J eq. (1), but also simultaneously but weights on the maximisation of the entropy in the action space. In this way exploration is encouraged to avoid getting stuck in a local optimum. 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 was retrained, as shown in fig. 5. Chances to get trapped in a local optimum are smaller by using this strategy of training. In difference to the first MBRL test, the batches of when acquiring new data are 50 with an initial random walk of 100 steps. The number of initial steps was chosen carefully large enough, because the convergence is slowed down and there is the risk that the training becomes un-

feasibly. The training was stopped after the acquisition of 500 data points and a verification was done as in the first test. Again the success is 100%, but the number of iterations per step is smaller. It might be a result of the bigger number of data points, but in general, this method showed a better asymptotic performance than the $ME-TRPO$ variant.

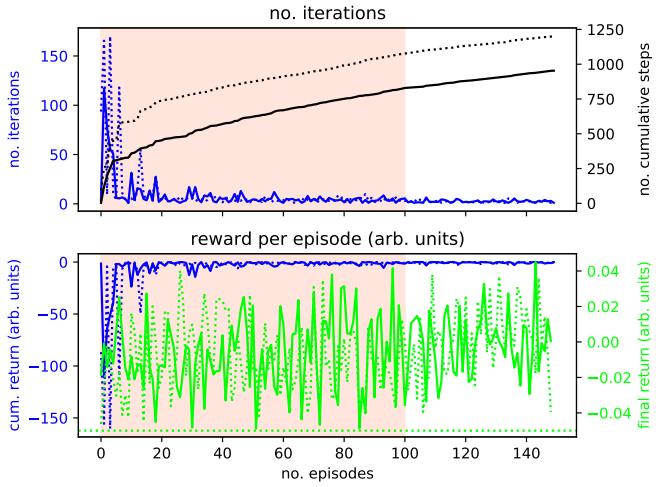

FIG. 2: The training on the *NAF2* in the FERMI FEL.

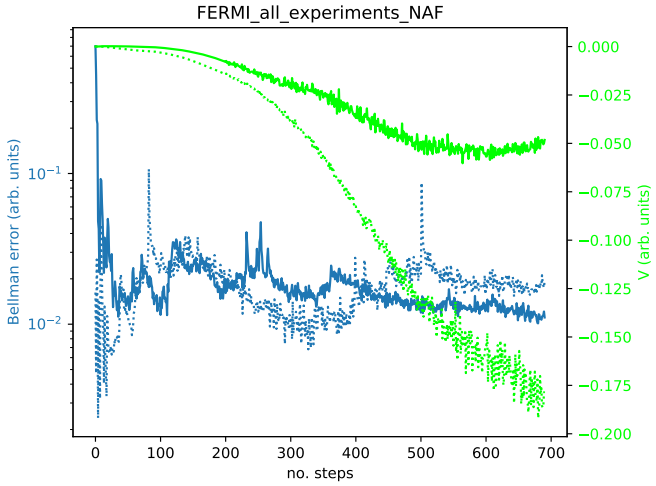

FIG. 3: The training .

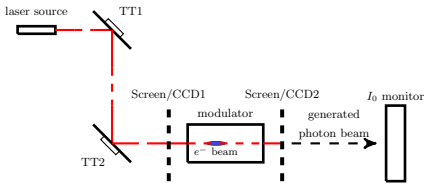

FIG. 4: The training .

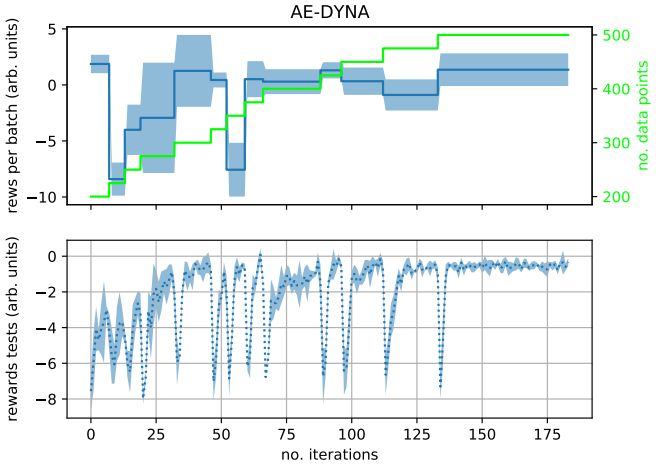

FIG. 5: The training .

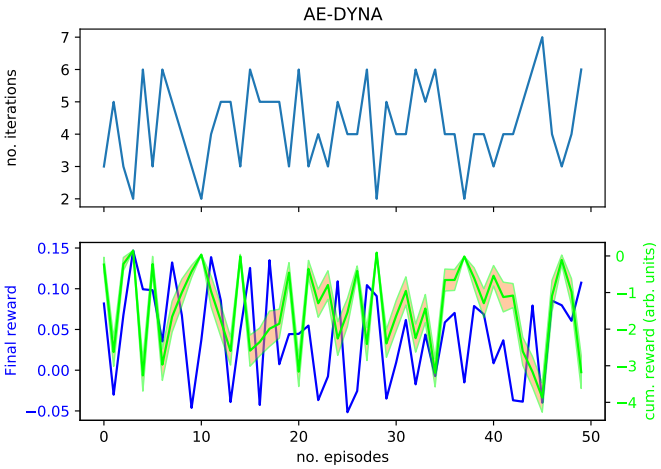

FIG. 6: The training .

VI. DISCUSSION AND OUTLOOK

VII. CONCLUSIONS

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 [8], 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 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. This is of importance when dealing with measurements on real systems.

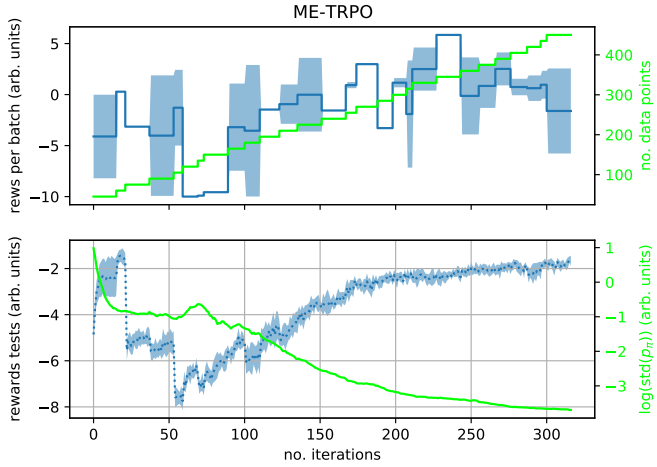

FIG. 7: The training .

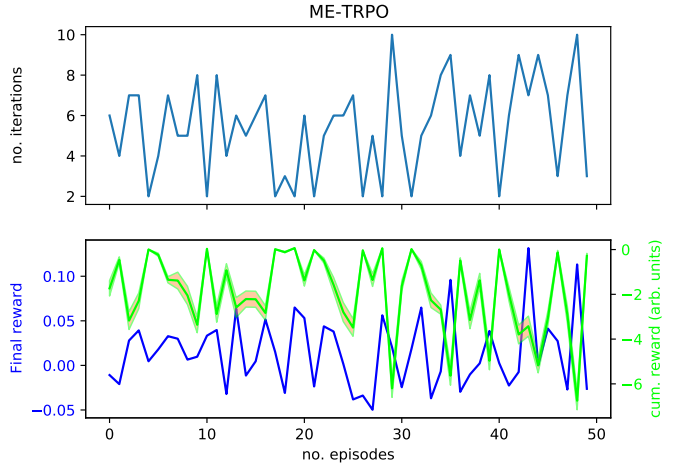

FIG. 8: The training .

1. NAF2 details

We compare the different *NAF* variants: *smoothing-double*, *smoothing single*, *standart*. The smoothing adds a small clipped noise on the actions to stabilize the network training as:

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

where $\epsilon \sim \mathcal{N}(0, \sigma)$. This method was used already in [9] to improve the deterministic policy gradient [10]. The double network was used in [7, 9] and is done in the following way:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')) \tag{A2}$$

and then both are learned by regressing to this target:

$$L(\phi_1, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left(Q_{\phi_1}(s, a) - y(r, s', d)\right)^2 \tag{A3}$$

and the policy is obtained via:

$$\max_{\theta} \mathop{\mathrm{E}}_{s\sim\mathcal{D}}\left[Q_{\phi_1}(s, \mu_\theta(s))\right]. \tag{A4}$$

The results are shown in fig. 9. One sees the cumulative reward per episode for a training of a total of 50 episodes. The curve labelled *clipping* corresponds to the double network including the smoothing method and shows the best stability during the training yielding quickly a high reward. Also the single network shows good and comparable performance, except for the stability. The worst performance is achieved without smoothing and a single network, nevertheless the result is competing with state of the art model free methods as [11] as the benchmark in the *leaderboard* of openai gym [12].

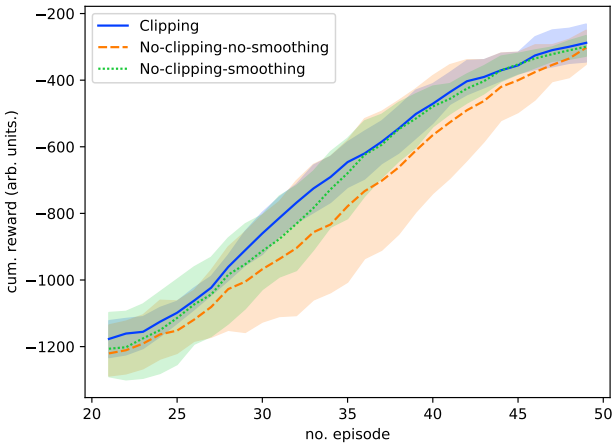

FIG. 9: Cumulative reward of different NAF implementations as discussed in the text.

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. 10. There the difference of the three methods becomes even more evident.

Appendix B: Theoretical aspects on the AE-DYNA on FERMI FEL

In this section we want to discuss some theoretical aspects concerning the AE-DYNA approach. A simulation

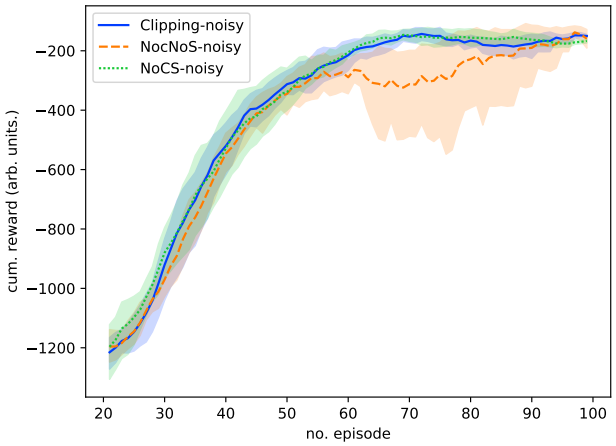

FIG. 10: Cumulative reward of different NAF implementations with artificial noise as discussed in the text.

was used to obtain the presented analysis.

One issue of MBRL methods is the asymptotic performance. Altough good results were obtained using the AE-DYNA, there is a difference between the MFRL and MBRL. We call it the *reality gap*. There are ideas to attack the problem e.g. by learning a meta-policy [13] followed by a fine tuning on the residual physics, which minimizes the *reality gap* ot training on a simulator and closing than with a small amount of training iterations on the real system the gap [14].

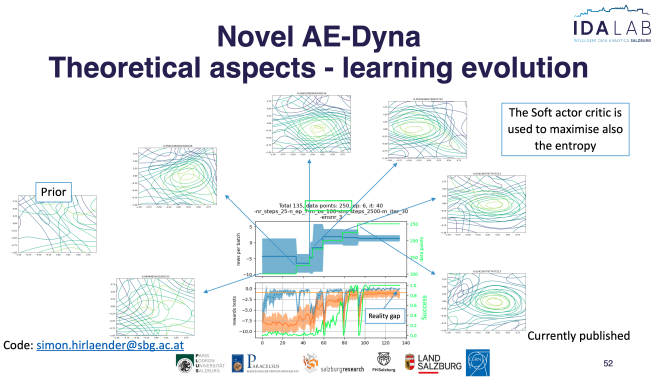

FIG. 11: The training .

[1] S. Levine, A. Kumar, G. Tucker, and J. Fu, Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2005.01643.

[2] 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.

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

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

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

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

[7] 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.

[8] 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).

[9] S. Fujimoto, H. van Hoof, and D. Meger, Addressing function approximation error in actor-critic methods (2018), arXiv:1802.09477 [cs.AI].

[10] 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.

[11] 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.

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

[13] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, Model-based reinforcement learning via meta-policy optimization, 1809.05214.

[14] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, Tossingbot: Learning to throw arbitrary objects with residual physics, 1903.11239.