Supplementary Material

1 IMPLEMENTATION DETAILS

State-of-the-art Deep RL algorithms are difficult to implement from scratch due to their complexity. Normally, the code used to obtain the published results is provided by the authors. However, the code might not always be guaranteed to work as expected. RL algorithms started to become somewhat standardised with the introduction of OpenAI Spinning Up Achiam (2018). However, the RL implementations found in Spinning Up are designed for educational purposes. A more robust library is OpenAI Baselines, which offers tested implementations of the main algorithms, with the primary two purposes of comparing their performance and improving the algorithms Dhariwal *et al.* (2017).

Stable Baselines is a fork of OpenAI Baselines, and it provides an RL library of reliable algorithm implementations with an improved performance by using hardware acceleration Hill *et al.* (2018). Stable baselines was initially used to train PPO, TD3 and SAC agents on QFBEnv. However, a more stable implementation of SAC from Hirländer (2020) was used to obtain the best policy for SAC.

Stable Baselines supports Tensorflow, which allows the use of Graphics Processing Units (GPUs) for faster training Abadi *et al.* (2016). Furthermore, Tensorflow provides a visualisation toolkit called Tensorboard. This tool proved to be essential for logging purposes during RL training. Considering that RL algorithms have many moving parts, Tensorboard condensed training information and made it easier to compare different RL agents.

NAF2 and AE-DYNA were implemented using a fork of the original code used in Hirlaender and Bruchon (2020). NAF2 was upgraded to Tensorflow 2 for more efficient training and logging. Nevertheless, difficulties were found when using the original implementation of AE-DYNA. With a code upgrade provided in Hirländer (2020), a newer version of AE-DYNA was attempted, and the results shown were obtained from this version.

2 SUPPLEMENTARY RL ALGORITHMS

2.1 Training

The Twin-Delayed Deep Deterministic policy gradient (TD3) algorithm as introduced in td3 (2018) was attempted on QFBEnv. Regardless, the training was unsuccessful and the policy did not converge. As a result, a hyperparameter grid-search was performed. A permutation of the parameters shown in Table S1 was attempted. The agents which obtained an average success rate of 100% at least once during their training were recorded. The different agents were sorted by the best average episode length. Five configurations out of 48 obtained a 100% success rate at least once. Table S2 shows the hyperparameters of the successful agents.Figure S1 shows the training performance statistics of the five best agents trained in the hyperparameter search. It can be seen that TD3#4 obtained the highest return of -0.9. However, upon inspection of the policy at steps 75000 and 76000, it was found that both had

Table S1. Hyperparameter search of TD3.

Learning rate	Batch size	Buffer size	τ
3×10^{-4}	32	1000	0.05
3×10^{-4}	64	10000	0.005
	128	50000	
	512		

Table S2. Hyperparameters of the five successful agents and the corresponding best mean episode length. The best agent, TD3#0, obtained an average episode length of 41.3 steps after 15000 training steps.

Index	Learning rate	Batch size	Buffer size	τ	Ave. Ep. Length (@ train. step)
TD3#0	3×10^{-5}	64	50k	0.005	41.3 (15k)
TD3#1	3×10^{-4}	512	50k	0.05	45.92 (7k)
TD3#2	3×10^{-5}	128	10k	0.05	46.4 (23k)
TD3#3	3×10^{-5}	64	10k	0.005	49.38 (15k)
TD3#4	3×10^{-4}	32	1k	0.005	64.96 (59k)

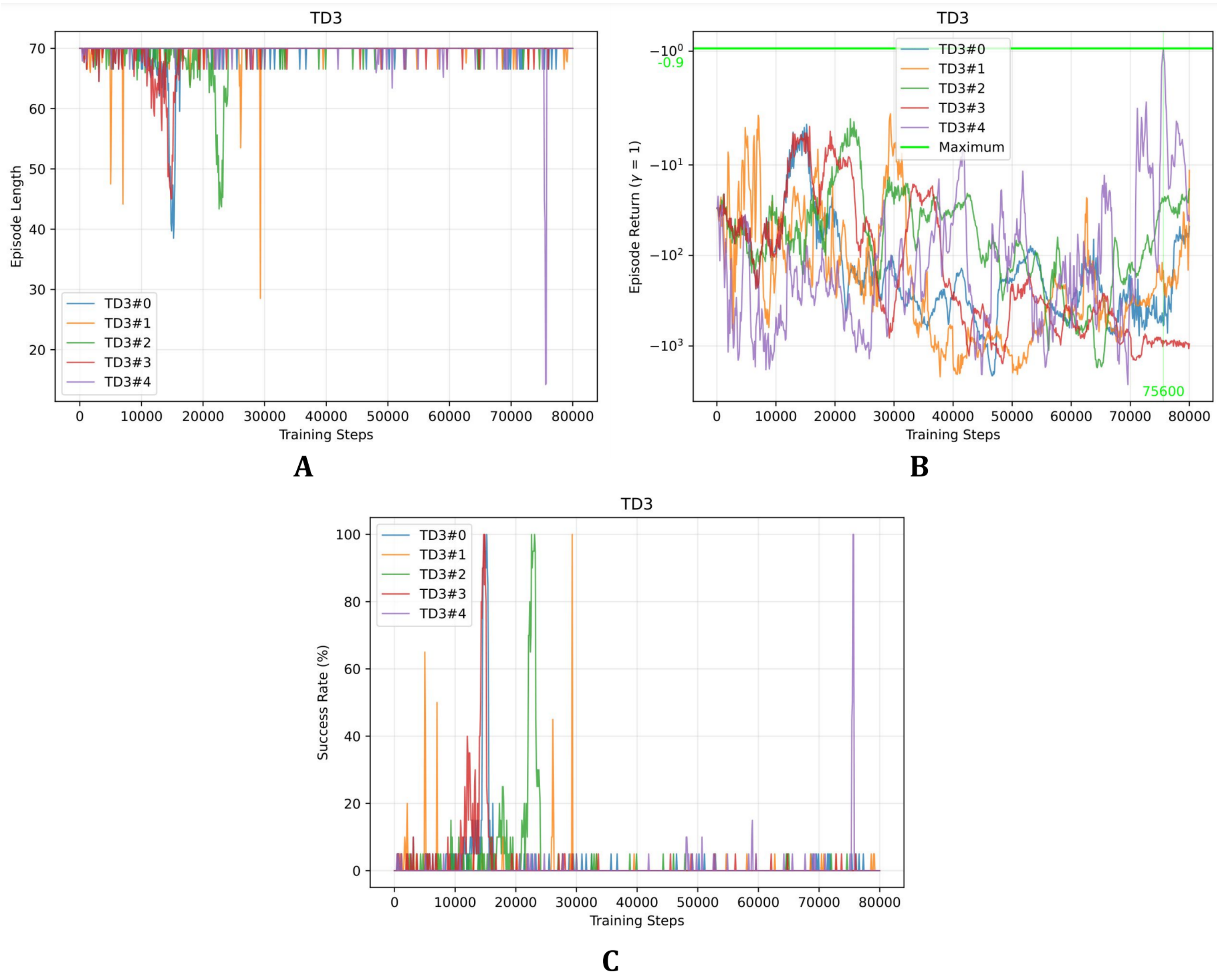

Figure S1. Performance statistics of the best TD3 agents during the hyperparameter search. (**A**) Episode length; (**B**) Undiscounted episode return and; (**C**) Success rate.

failed to converge. This suggests that the 100% successful policy obtained on step 75600 was forgotten after a few hundred steps. For comparison of the training with the other algorithms attempted in this work, Figure S2 shows the average training performance of five TD3 agents using the best hyperparameters and initialised using different random seeds.

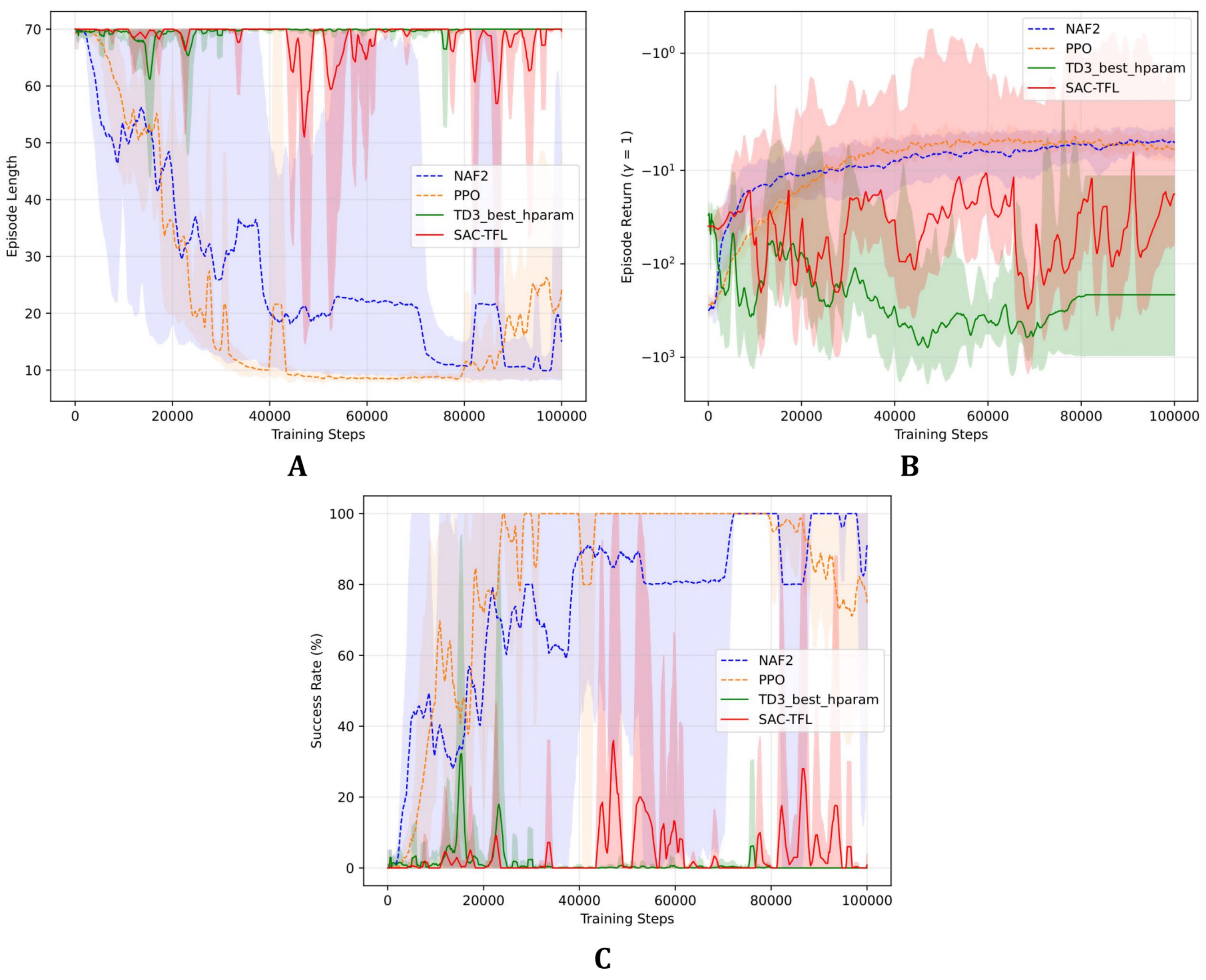

Figure S2. Performance statistics of NAF2, PPO, TD3 and SAC-TFL agents during training. Five agents per algorithm were initialised with different random seeds. (**A**) Average episode length; (**B**) Average episode return and; (**C**) Average success rate.

The SAC implementation which trained the best agents is based on the work in Hirländer (2020). Since this implementation uses TensorLayer, an open-source deep-learning and RL library extended from TensorFlow, it will be denoted by SAC-TFL. The hyperparameters shown in Table S3 are the parameters as implemented in Hirländer (2020). Figure S2 shows the average training performance statistics of five SAC-TFL agents initialised with different random seeds. Figure S2A to S2C show that some agents managed to obtain episode lengths smaller than 20 after approximately 45000 steps. A success rate of 100% was maintained for approximately 5000 steps, after which the policy maintains an episode return on the same order of magnitude. In real

Table S3. Hyperparameters used for SAC-TFL.

Name	Value
learning rate	0.001
γ	0.99
batch size	256
buffer size	500000
τ	0.005

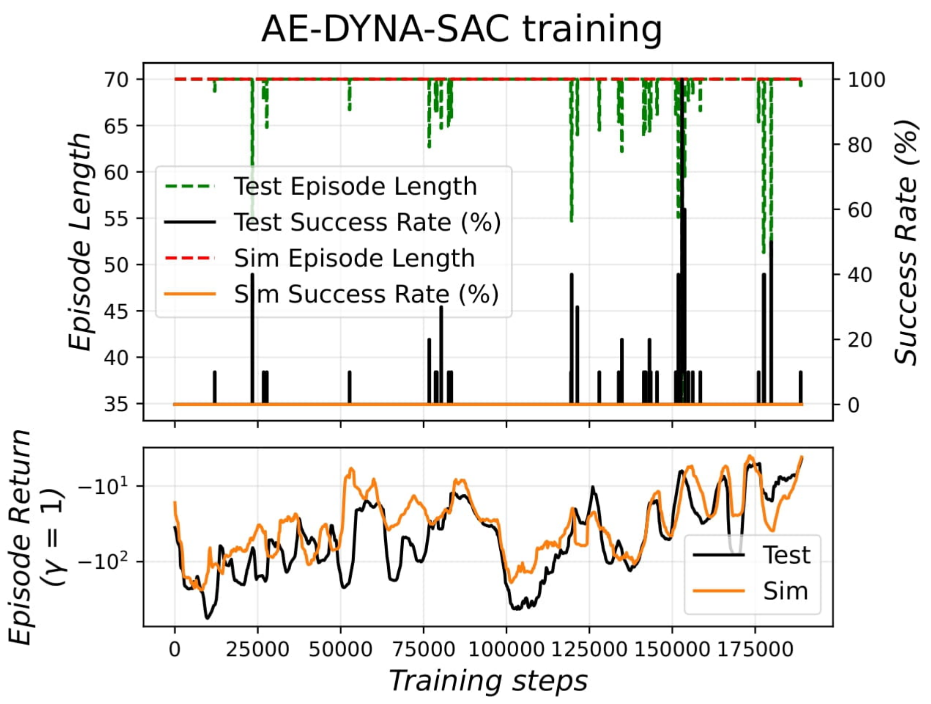

Figure S3. Snapshot of the training performance of AE-DYNA-SAC. *Sim* denotes information about the agent performance over the models. *Tests* denotes information about the agent performance over the real environment.

LHC operation on the QFB, the worst-case training time of 45,000 steps is estimated by:

$$\frac{45,000 \text{ steps}}{12.5\,\mathrm{Hz}} \times 2 \;=\; 2\,\mathrm{h}$$

Training AE-DYNA on QFBEnv consists of performing a sequence of three tasks iteratively. The tasks are: a) collecting a *batch* of data by interacting with the real environment; b) training a set of models from the collected data independently, where each model can accept a state-action pair and predict the next-state and reward; c) train a standard Model-Free (MF) RL algorithm by interacting with the trained models instead of the real environment.

Anchored-ensembles of neural networks were used to model both the aleatoric and epistemic uncertainty of the environment. This allowed the sampling of the next-state and reward from different, randomly initialised networks Pearce *et al.* (2018). This has been shown to approximate the stochasticity observed in a real environment and to capture the epistemic uncertainty of the system dynamics. In this work, the MF-RL algorithm chosen was SAC-TFL, which was trained from scratch on every epoch. An epoch is defined as the time between data collections from the real environment, i.e. time between two *batches*. The trained agent is consecutively used to obtain new batches of data from the real environment to diversify the variety of trajectories in the buffer. Consequently, the agent is re-initialised and the AE-DYNA training loop repeats.

The training performance of the best AE-DYNA-SAC on QFBEnv is shown in Figure S3. AE-DYNA-SAC was trained for six epochs which is equal to 4500 steps in QFBEnv. This is equivalent to $6\,\mathrm{min}$ of beam time and was the only data available for the models and the agent to successfully train a policy. Sim and Test denote information about the interactions of the RL agent with the models and the real environment, respectively. If the trained models were perfect representations of the real environment, the results of Sim would be equivalent to Test (training = evaluation). The top plot of Figure S3 shows that during evaluation on the real environment, a 100% success rate with an average episode length of 35 steps is observed at around 155000 training steps (black line). On the contrary, the predicted success rate of the agent by the models remains 0% throughout all the training (orange line). This indicates that the models cannot guide training and constant testing on the real environment is required to successfully train AE-DYNA-SAC.

To counteract the unsuccessful termination when using models to train the agent, evaluation episodes on the real environment are required to probe the performance of the latest AE-DYNA-SAC policy. If one evaluation episode is done on the real environment every 1000 training steps in the models, 155000 training steps become approximately: $\left(\frac{155000}{1000} * 70\right) * \frac{1}{12.5\,\mathrm{Hz}} \approx 14.5\,\mathrm{min}$. The total worst case training time for AE-DYNA while using these evaluation parameters becomes: $(\text{Batch time} + \text{Test time}) \times 2 \approx 41\,\mathrm{min}$

2.2 Evaluation

2.2.1 Effect of Gaussian noise

Figure S4A shows three episodes obtained by the best TD3 agent with deterministic actions in QFBEnv. The top plots show that the policy does not take the most optimal path to the terminal state. In particular, the state evolution of Episode #1 shows that the agent takes smaller steps towards the optimal point than the PI controller, and as a result, the episode length approximately doubles. The bottom plots show that similarly to PPO, the actions of TD3 start to converge back to zero towards the end of the episode. It can also be observed that the change in subsequent actions is gentler than both NAF2 and PPO. Figure S4B to S4D show a set of three evaluation episodes each obtained from the best TD3 agent under the effect of Gaussian action noise. It can be seen that the TD3 agent managed to satisfy the early termination criterion until 10% action noise, beyond which the episode length increases to 70 steps. The top plots of Figure S4C also show that the state excursions become longer for the agent, especially in Episode #3. This indicates that TD3 policy trained is not robust to stochasticity in QFBEnv.

Figure S5A shows three episodes obtained using the best SAC-TFL policy with deterministic actions. The top plots show that SAC-TFL has found an optimal policy which converges to the optimal point before the PI controller. However, the bottom plots illustrate that the deterministic actions selected by SAC-TFL are the most chaotic of all the agents discussed so far. Furthermore, the actions are not decaying to zero, which implies that SAC-TFL converged to a sub-optimal policy. Similarly to the other agents, Figure S5B to S5D show a set of three evaluation episodes each using different action noise. Until 25%, the agent obtains a shorter average episode length than the PI controller of approximately 15 steps. At 50% action noise, the SAC-TFL average episode length increases to approximately 30 steps. Notwithstanding the undesirable choice of actions, the agent managed to end each episode successfully. This result indicates that the SAC-TFL policy trained is very robust to stochasticity in QFBEnv.

Figure S6A shows three episodes obtained by the best AE-DYNA-SAC policy with deterministic actions. Similarly to the other agents, Figure S6B toS6D correspond to an action noise of 10%, 25% and 50%. In the deterministic case, the policy converges to the optimal point with an average episode length approximately equal to that of the PI controller. As the action noise is increased, the state excursions become larger; e.g. Episode #2 of Figure S6C; and the policy can also converge to a bad optimum, e.g. Episode #1 of Figure S6C.

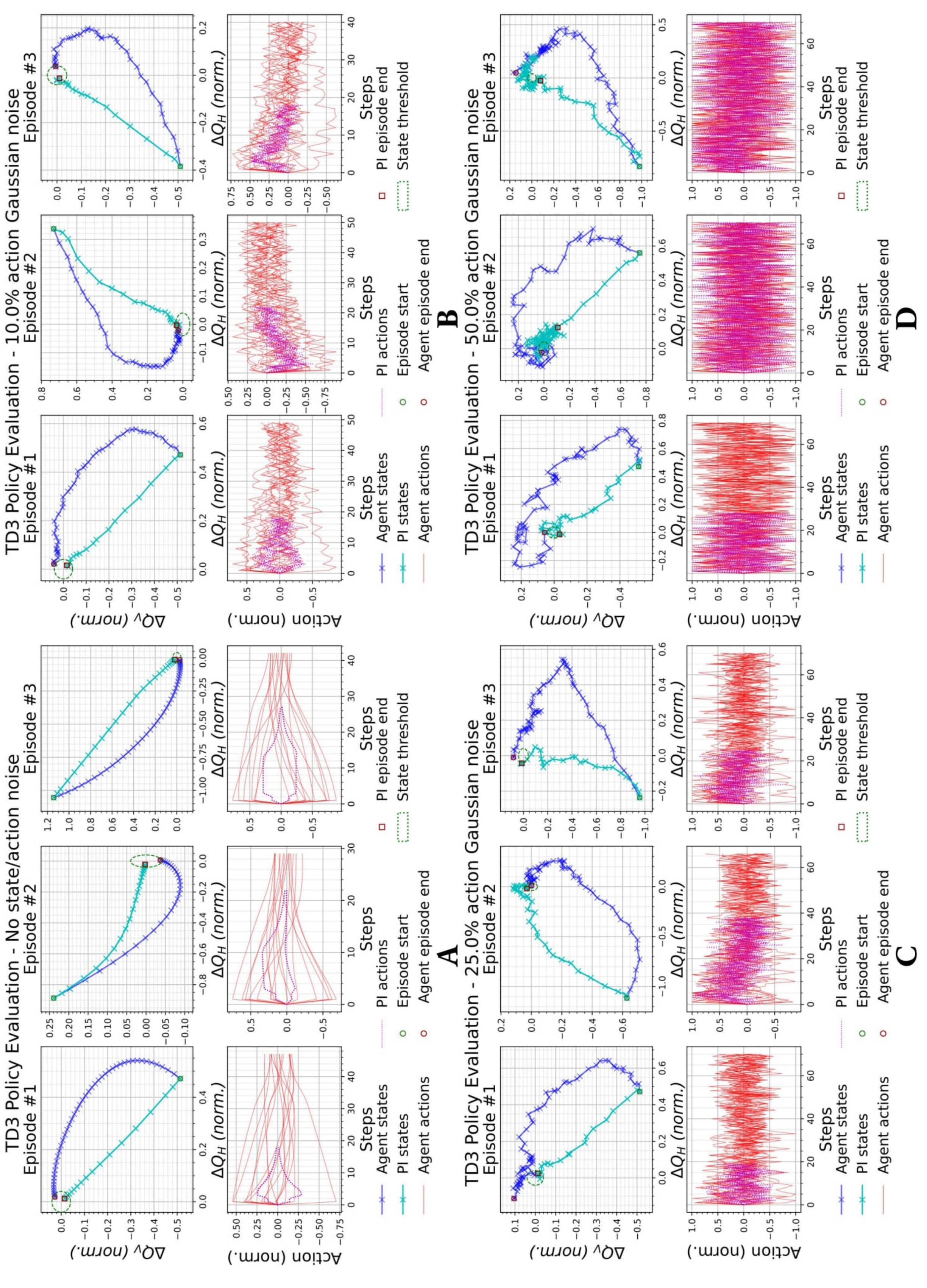

Figure S4. Episodes from the best TD3 agent and the PI controller with the same initial states and with a varying additive Gaussian action noise with zero mean and standard deviation as a percentage of the half action space [0, 1]. (**A**) 0%, (**B**) 10%, (**C**) 25%, and (**D**) 50% Gaussian action noise.

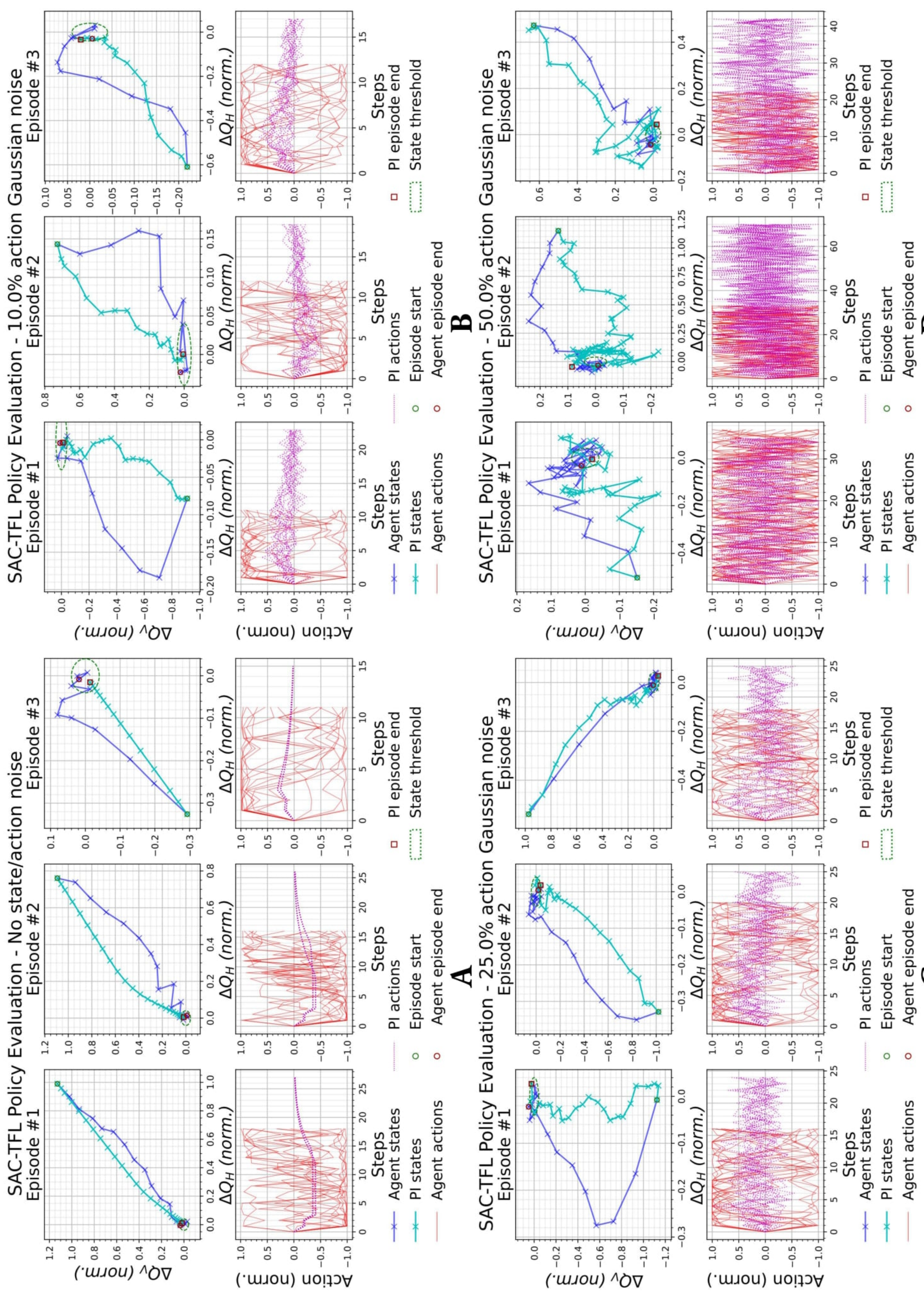

Figure S5. Episodes from the best SAC-TFL agent and the PI controller with the same initial states and with a varying additive Gaussian action noise with zero mean and standard deviation as a percentage of the half action space [0, 1]. (**A**) 0%, (**B**) 10%, (**C**) 25%, and (**D**) 50% Gaussian action noise.

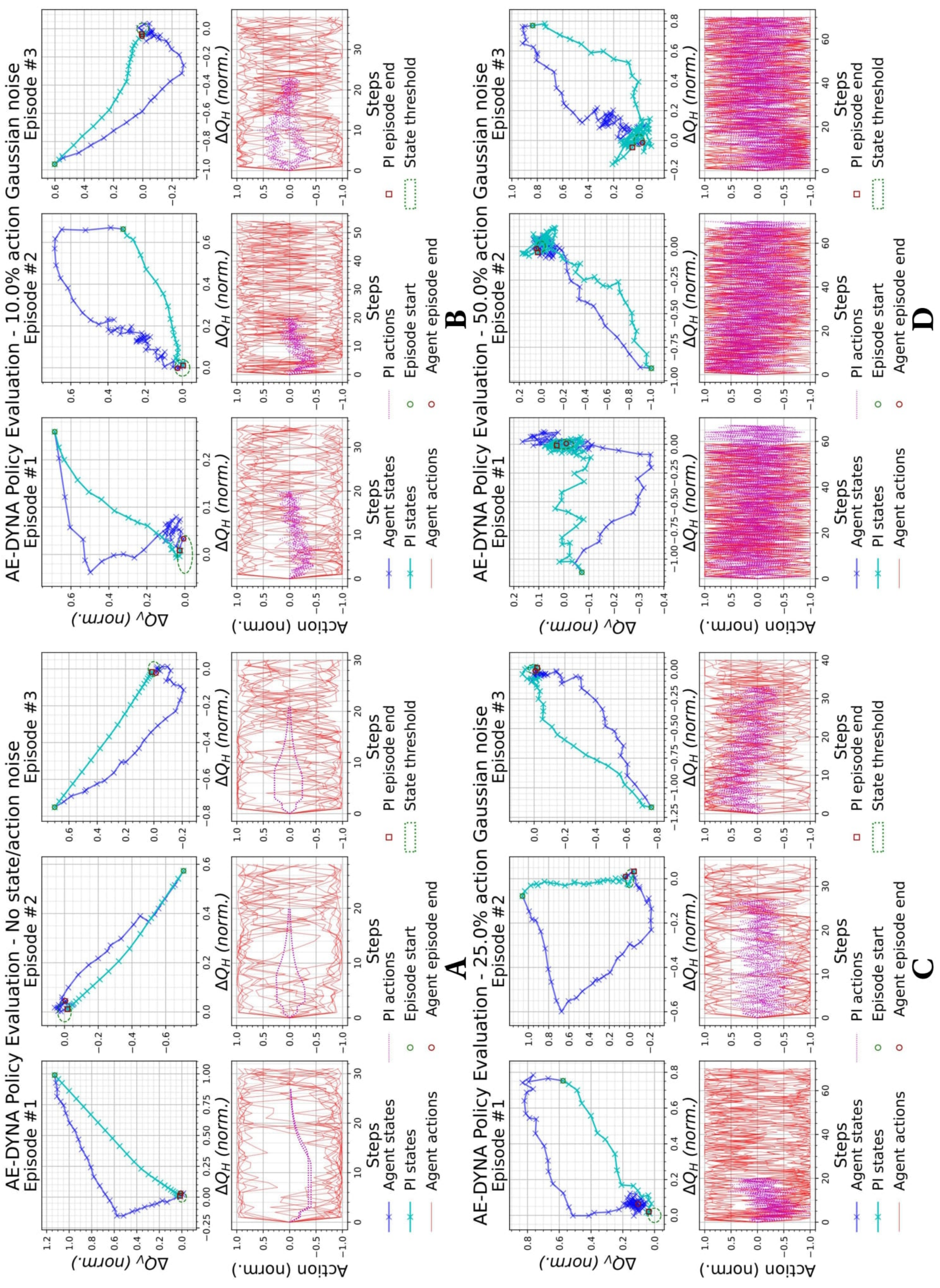

Figure S6. Episodes from the best AE-DYNA-SAC agent and the PI controller with the same initial states and with a varying additive Gaussian action noise with zero mean and standard deviation as a percentage of the half action space [0, 1]. (**A**) 0%, (**B**) 10%, (**C**) 25%, and (**D**) 50% Gaussian action noise.

Table S4. The statistics (mean±std.) for the episode length obtained by the best RL agents and PI controller with respect to the amplitude of Gaussian action noise.

Action noise	**0%**	**10%**	**25%**	**50%**
TD3	39.06 ± 12.08	40.34 ± 12.43	47.06 ± 15.24	63.36 ± 13.21
SAC-TFL	13.34 ± 3.19	13.64 ± 3.43	15.60 ± 4.53	24.81 ± 12.27
AE-DYNA-SAC	36.25 ± 17.50	37.22 ± 17.78	41.98 ± 18.24	54.51 ± 17.84
PI controller	20.32 ± 3.60	20.66 ± 3.82	25.20 ± 7.33	53.12 ± 18.92

Table S5. Episode length statistics (mean±std) under the effect of actuator failures.

Algorithm	**Nb. of actuator failures**			
	0	**1**	**3**	**5**
NAF2	9.10 ± 1.39	33.01 ± 29.24	69.14 ± 7.01	70.00 ± 0.00
PPO	8.80 ± 1.28	9.14 ±1.50	41.60 ± 29.85	70.00 ± 0.00
TD3	39.06 ± 12.08	69.05 ± 6.21	70.00 ± 0.00	70.00 ± 0.00
SAC	43.09 ± 25.49	62.97 ± 17.77	69.72 ± 3.93	70.00 ± 0.00
SAC-TFL	12.97 ± 3.03	29.44 ± 24.69	63.45 ± 17.40	69.90 ± 2.21
AE-DYNA-SAC	36.79 ± 17.55	61.58 ± 15.97	69.82 ± 2.72	70.00 ± 0.00
PI controller	20.09 ± 3.70	70.00 ± 0.00	70.00 ± 0.00	70.00 ± 0.00

2.2.2 Effect of actuator failure

It can be seen from Figure S7A that the best TD3 agent trained in this work, already failed to successfully terminate episodes with one actuator failure, i.e. fails to converge $max(\Delta Q_H, \Delta Q_V) <$ Goal threshold. This trend worsens as the number of actuator failures increases, as can be seen in Figure S7B. At one actuator failure in Figure S7C, the SAC-TFL policy optimally converges to the optimal point. However, at three actuator failures in Figure S7D, SAC-TFL approaches the optimal point but fails to converge under the threshold boundary. Figure S8 shows that the AE-DYNA-SAC policy can, at times, converge close to the optimal point. The performance continues to degrade as more actuator failures are introduced. Nonetheless, the policy degradation is comparable to the performance degradation of the PI controller during actuator failures.

The same procedure used to create the evaluation plots discussed above was used to obtain more statistics on the performance of the trained RL agents and the PI controller in the presence of actuator failures. The maximum episode length was set back to 70 steps. The episode length and the final distance to the optimal point (DTO) were used as performance metrics. Three scenarios with one, three and five actuator failures were attempted using the same procedure described above, with a total of 1000 episodes per scenario per algorithm used to obtain the performance statistics.

Tables S5 and S6 tabulate the statistics for the episode length and DTO under the effect of actuator failures for all algorithms attempted in this work. The best PPO agent manages to maintain the smallest average episode length and a DTO which is close to the Goal threshold of QFBEnv at up to three actuator failures. NAF2 and SAC-TFL show a similar performance up to one actuator failure, however, it can be seen that SAC-TFL was the only agent which obtained successful termination at five actuator failures. The performances of TD3, SAC and AE-DYNA-SAC with respect to the average episode length are similar, however, AE-DYNA-SAC obtained slightly lower DTO. Ultimately, all the RL agents behaved better than the PI controller, which obtained no successful termination during actuator failures.

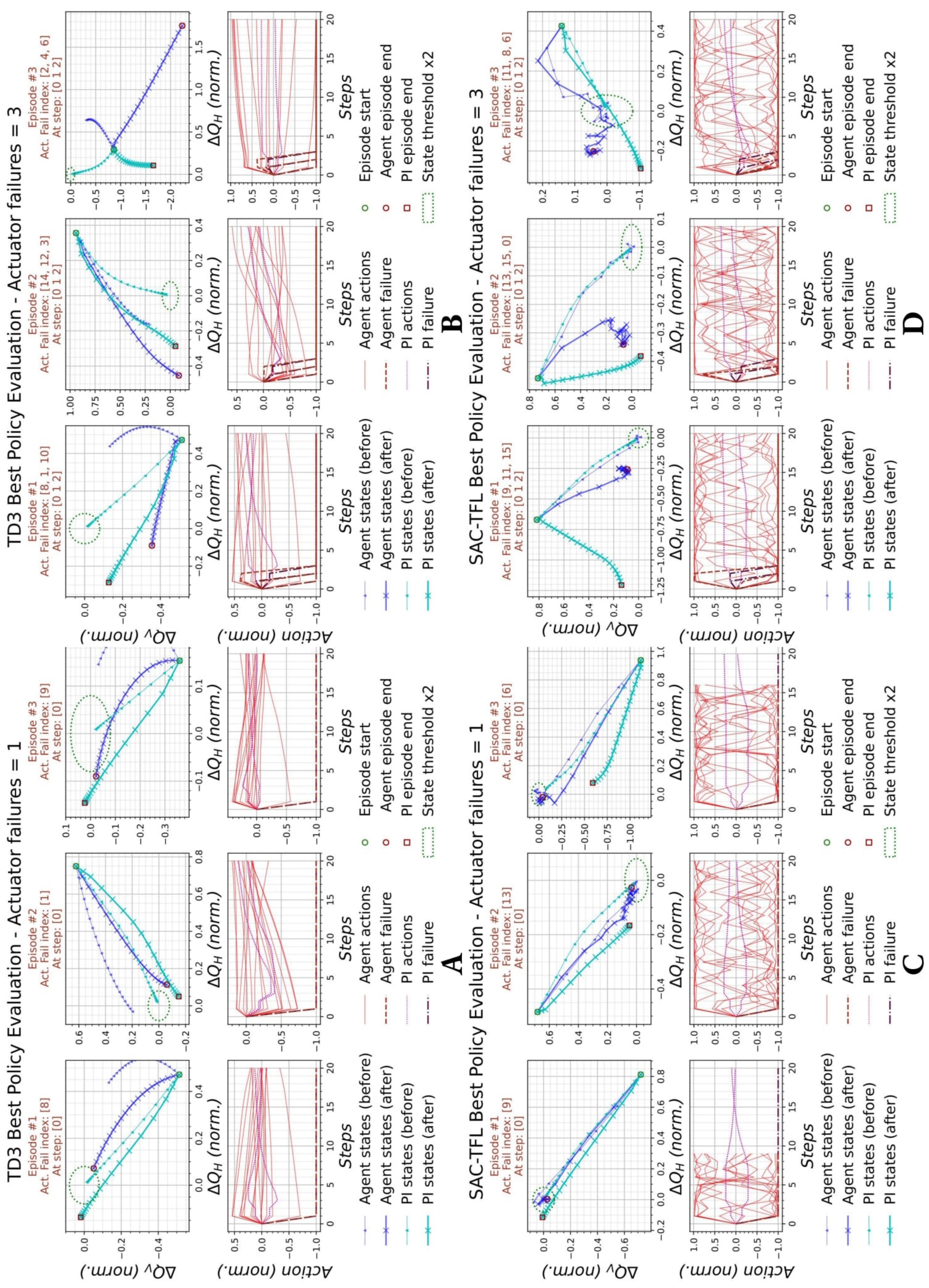

Figure S7. Episodes from the best RL agents and the PI controller under the effect of different number of actuator failures.(**A**) TD3 agent with one actuator failure, (**B**) TD3 with three actuator failures, (**C**) SAC-TFL agent with one actuator failure, (**D**) SAC-TFL agent with three actuator failures.

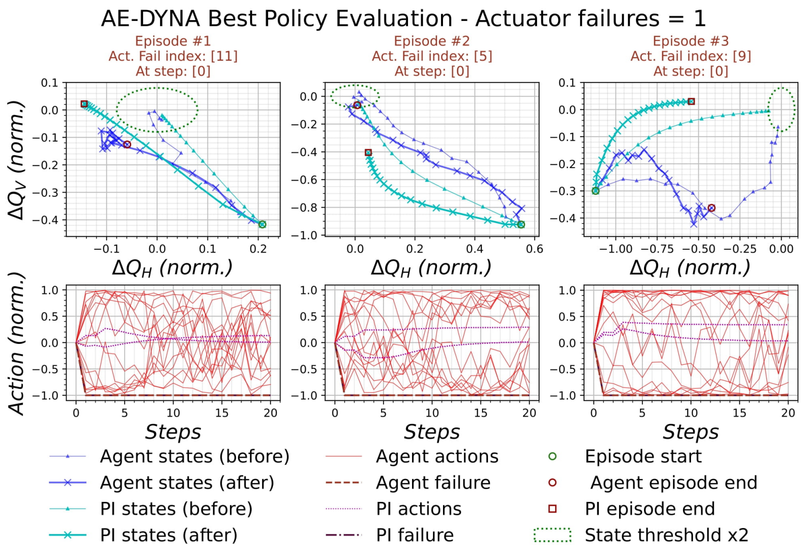

Figure S8. Best AE-DYNA-SAC agent and PI controller with 1 actuator failure.

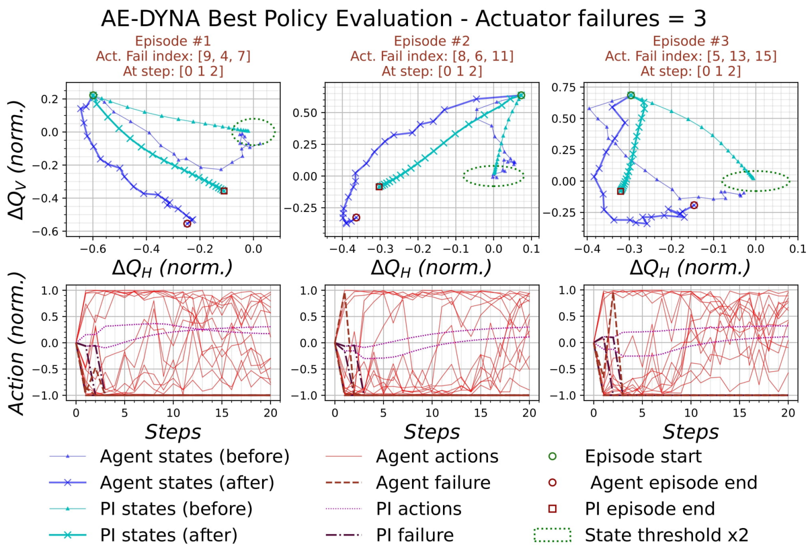

Figure S9. Best AE-DYNA-SAC agent and PI controller with 3 actuator failures.

Table S6. Distance to Optimal point (DTO) statistics (mean±std) under the effect of actuator failures. Underlined cells show where it is likely to observe successful termination.

Algorithm	Nb. of actuator failures			
	0	1	3	5
NAF2	<u>0.02 ± 0.00</u>	<u>0.05 ± 0.02</u>	0.13 ± 0.05	0.56 ± 0.87
PPO	<u>0.01 ± 0.00</u>	<u>0.02 ± 0.01</u>	<u>0.07 ± 0.02</u>	0.52 ± 1.02
TD3	<u>0.03 ± 0.00</u>	0.84 ± 0.76	1.96 ± 1.78	3.43 ± 2.03
SAC	<u>0.04 ± 0.02</u>	0.25 ± 0.16	1.92 ± 1.04	5.02 ± 1.82
SAC-TFL	<u>0.02 ± 0.01</u>	<u>0.07 ± 0.07</u>	0.26 ± 0.17	0.94 ± 0.75
AE-DYNA-SAC	<u>0.04 ± 0.04</u>	<u>0.18 ± 0.14</u>	1.03 ± 0.87	2.18 ± 1.46
PI controller	<u>0.02 ± 0.00</u>	0.17 ± 0.01	0.98 ± 0.97	3.10 ± 1.34
Goal threshold	0.057			

2.2.3 Effect of incorrect tune estimation

Figure S10A shows the best TD3 policy under the effect of $50\,\mathrm{Hz}$ harmonics. It can be observed that the policy behaves similarly to the response of the PI controller and extends up to the second harmonic away from the optimal point. Figure S10B shows the results for the best SAC-TFL policy. It can be seen that the state was maintained around the closest $50\,\mathrm{Hz}$ harmonic intersection to the optimal point. This performance is similar to the best-performing agents, NAF2 and PPO. Figure S10C are the results corresponding to the best AE-DYNA-SAC policy and show that the policy obtained a performance similar to the PI controller.

REFERENCES

R. J. Steinhagen, *LHC Beam Stability and Feedback Control-Orbit and Energy*, Ph.D. thesis, RWTH Aachen U. (2007).

L. Grech, G. Valentino, D. Alves, A. Calia, M. Hostettler, J. Wenninger, and S. Jackson, in *Proceedings of the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021)* (2021).

M. Gasior and R. Jones, in *Proceedings of 7th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC 2005)* (2005) p. 4.

MAD - methodical accelerator design, `http://mad.web.cern.ch/mad/` (2019), accessed: 2019-6-26.

R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction*, Adaptive Computation and Machine Learning (The MIT Press, 2018).

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, Nature **529**, 484 (2016).

T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, Model-Ensemble Trust-Region policy optimization (2018), arXiv:1802.10592 [cs.LG] .

S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, in *International Conference on Machine Learning* (2016) pp. 2829–2838.

S. Hirlaender and N. Bruchon, Model-free and bayesian ensembling model-based deep reinforcement learning for particle accelerator control demonstrated on the FERMI FEL (2020), arXiv:2012.09737 [cs.LG] .

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

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms (2017), arXiv:1707.06347 [cs.LG] .

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, Soft Actor-Critic: Off-Policy maximum entropy deep reinforcement learning with a stochastic actor (2018), arXiv:1801.01290 [cs.LG] .

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym (2016), 1606.01540 .

J. G. Ziegler and N. B. Nichols, J. Dyn. Syst. Meas. Control **115**, 220 (1942).

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (USENIX Association, Savannah, GA, 2016) pp. 265–283.

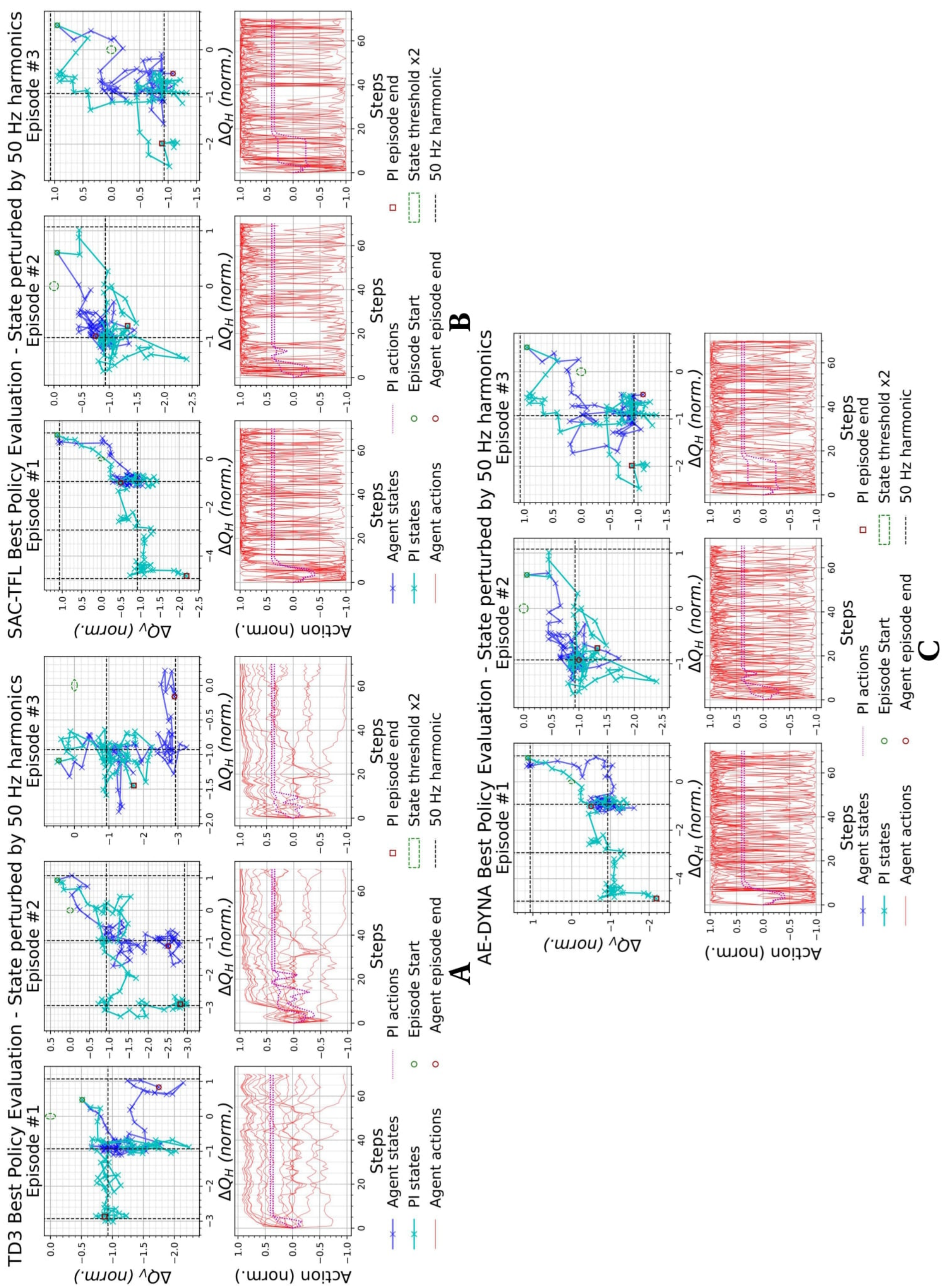

Figure S10. Effect of 50 Hz harmonics on the best RL agents and a PI controller. (**A**) Best TD3 agent, (**B**) best SAC-TFL agent and (**C**) best AE-DYNA-SAC agent.

L. Grech, G. Valentino, D. Alves, M. Gasior, S. Jackson, R. Jones, T. Levens, and J. Wenninger, in *Proceedings of the 9th International Beam Instrumentation Conference (IBIC 2020)* (2020).

L. Grech, G. Valentino, and D. Alves, Information **12**, 197 (2021b).

M. Janner, J. Fu, M. Zhang, and S. Levine, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019).

J. Achiam, Spinning up in deep reinforcement learning (2018).

P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, Openai baselines (2017).

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).

S. Hirländer, MathPhysSim/FERMI_RL_Paper: Preprint release (2020).

Addressing Function Approximation Error in Actor-Critic Methods (2018) arXiv:1802.09477 [cs.AI] .

T. Pearce, N. Anastassacos, M. Zaki, and A. Neely, Bayesian inference with anchored ensembles of neural networks, and application to exploration in reinforcement learning (2018), arXiv:1805.11324 [stat.ML] .