Exploration of Q-Learning using Normalized Advantage Functions and Model-Based Accelerations in Various Different Domains

Justin Lee

2023-12-17

1 Introduction

Q-learning is a major branch of reinforcement learning with a multitude of different algorithms and techniques. The fundamental idea behind Q-learning is to find a function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which takes in as input a state-action combination, and returns the reward-to-go [1]. Traditionally, this approach was used for discrete state-action spaces because it allows Q to be represented using a table where each cell represented a state-action combination. Continuous state-action spaces require a more explicit formulation for Q. Using the function approximation capabilities of neural networks, it is now possible to implement Q learning for continuous state-action spaces. Crucially, the Q-learning algorithm requires computing $\mu(x) = \text{argmax}_u Q(x, u)$ in the training loop which could be done quickly and efficiently in the tabular case, but would require solving an optimization problem as a subroutine. Several techniques have been developed in order to handle this bottleneck and make training feasible. This paper will explore the following methods: normalized advantage functions (NAF), deep deterministic policy gradient (DDPG), and twin delayed DDPG (TD3). Comparisons will be made on three different environments: Pendulum, Mountain Car Continuous, and Simglucose. This project will give a particular focus to NAF and Simglucose, especially because the existing literature on these Q-learning techniques are presented using a robotics domain. The goal is to explore whether these techniques are just as applicable across different types of domains.

2 Literature Review

Numerous works have explored Q-learning for continuous state-action domains using neural networks. Considerable results have been achieved with using neural networks to approximate Q for Atari games [2] [3], albeit for discrete action spaces, showing that deep Q networks (DQNs) were viable for reinforcement learning. Based on this success, Q-learning for continuous actions were explored with the introduction of the deterministic policy gradient [4]. Although this should work in theory, in practice convergence is difficult. This was followed

by the DDPG algorithm which introduced a slowly updated target network in order to improve numerical convergence [5]. DDPG was further improved with the TD3 algorithm [6]. TD3 improved DDPG by adding a Double Q-learning approach [7], updating the policy less frequently, and adding noise to the target action.

Although DQNs and the improvements take an actor-critic approach, they are not the only approach possible. NAF is an approch to analytically compute $\mu(x) = \text{argmin}_u Q(x, u)$ by restricting Q to take a quadratic form [8]. More specifically, Q is decomposed into a value function $V(x)$ and advantage function $A(x, u)$ [9] [10]. The advantage A is restricted to take a quadratic form such that whatever $\mu(x)$ is outputed by the network, it is by definition $\text{argmin}_u Q(x, u)$. The same work then continues to incorporate model-based accelerations in order to speed up training.

A particularly relevant way to leverage the benefits of having a dynamics model is by using model-guided exploration. Dyna Q is a method of Q-learning that learns a dynamics model $p(x_{t+1}|x_t, u_t)$ and uses those dynamics to generate synthetic rollouts originating at states visited by real world rollouts [11]. Another method is called the iterative linear quadratic gaussian controller (iLQG), and comes from trajectory optimization literature. Under local linear dynamics and quadratic rewards, optimized time-varying linear feedback controllers can be generated to maximize the locally quadratic Q [12] [13].

NAF implements model-based accelerations through what the authors call "imagination rollouts". The idea is to use an additional fictional replay buffer, filled by rollouts done in a Dyna Q fashion. These can be done using either actions outputted by the NAF network, or by actions outputted by the iLQG time-varying controllers. Training is done by sampling from both the real and fictional replay buffers. A key point to note is that in each training step, multiple samples are taken from the fictional replay buffer.

3 Outline of the Report

This report proceeds as follows:

1. Problem Statement and Formulations
2. Experiments and Benchmarks
3. Results
4. Conclusions

4 Problem Statement and Formulations

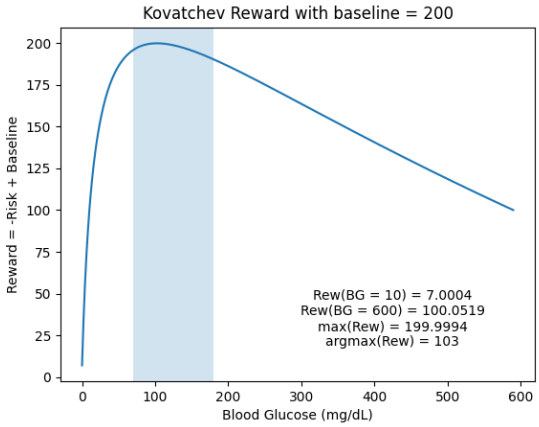
This project will consider how well four different Q-learning algorithms perform: DDPG, TD3, NAF, and NAF with model-based accelerations (NAF-MBA). These algorithms will be compared across three different environments: Pendulum, Mountain Car Continuous, and Simglucose. Pendulum and Mountain Car Continuous are part of the Gymnasium classic control environments [14]. Simglucose is a Python implementation of the FDA-approved UVA/Padova simulator type-1 diabetes simulator (2008 version) [15] [16]. The Simglucose

environment is slightly different from the usual glucose simulator in that the observation is just an hourly blood glucose reading. Further, the action only handles basal insulin and not bolus insulin.

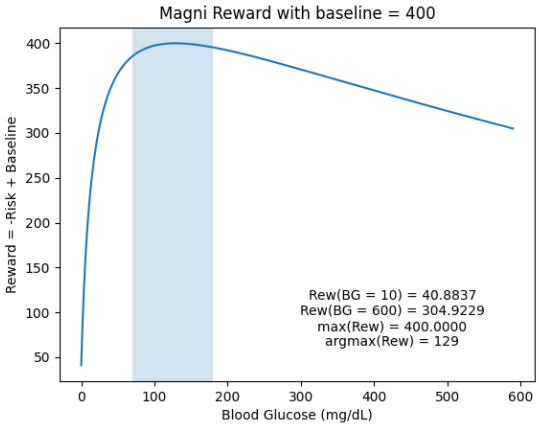
The state, action, and reward formulations are as follows:

- Pendulum:
  - **State**: Vector of length 3, describing the (x, y) coordinates and angular velocity of the pendulum's free end.
  - **Action**: Vector of length 1, representing the torque applied to the free end of the pendulum.
  - **Reward**: $r = -(\theta^2 + 0.1(\delta\theta)^2 + 0.001u)$ where θ is the angle of the pendulum, $\delta\theta$ is the change in angle, and u is the torque. The goal is to keep the pendulum upright where $\theta = 0$.
- Mountain Car Continuous:
  - **State**: Vector of length 2, describing the x-position of the car and the velocity of the car.
  - **Action**: Vector of length 1, representing the directional force applied on the car.
  - **Reward**: $r = -0.1u^2$ where u is the directional force. Additionally, a positive reward of 100 is added to that timestep if the car reaches the goal at x-position 0.45.
- Simglucose:
  - **State**: Vector of length 1, describing the CGM reading of blood glucose (BG) at that timestep, where each step represents 1 hour.
  - **Action**: Vector of length 1, representing the amount of basal insulin to administer.
  - **Reward**: Multiple are used:
    - *Default*: $r = risk[t-1] - risk[t]$ where $risk$ is defined as Kovatchev risk [17] [18].
    - *Simple*: $r = \begin{cases} -1 & BG > 180 \\ -4 & BG < 70 \\ 1 & otherwise \end{cases}$
    - *Magni*: $r = -risk[t]$ where $risk$ is defined as Magni risk [19].
    - *Kovatchev*: $r = -risk[t]$ where $risk$ is defined as Kovatchev risk [17] [18].

    The goal is to regulate BG between 70 and 180.



(a) Kovatchev reward baselined by 200 to keep it positive for $10 < BG < 600$



(b) Magni reward baselined by 400 to keep it positive for $10 < BG < 600$

Figure 1: Baselined Kovatchev and Magni reward functions

5 Experiments and Benchmarks

I ran six experiments for each algorithm:

1. Pendulum
2. Mountain Car Continuous
3. Simglucose with default reward
4. Simglucose with simple reward
5. Simglucose with Magni reward
6. Simglucose with Kovatchev reward

In order to maintain as much comparability between the experiments, I opted to keep as many hyperparameters consistent as possible across the experiments. Based on the paper introducing NAF, all the neural network stem architectures will have two hidden layers each with 256 nodes, both followed by a batch norm and ReLU activation function [8]. The output layer goes through a tanh activation, then an un-scaling operation to ensure that the range of the output spans the respective environment's action space. The experiments using the classic control environments are all trained on 40000 environment steps, while the experiments using the Simglucose are all trained on 200000 environment steps. The Simglucose experiments also all use the same meal schedule.

The following is a table of the hyperparameters shared across all experiments, regardless of choice of algorithm:

Learning Rate	τ (Polyak)	γ	σ (Exploration Noise)	Batch Size
0.001	0.005	0.99	0.3	100

Finally, for the NAF-MBA experiments, the time-varying linear dynamics were refit every 5 episodes, and the probability of selecting the NAF network for the imagination rollouts is set to 0.8. The quadratic reward matrix for iLQG was set to the identity matrix, and the linear reward vector was set to a vector of 1s.

As a sidenote, it is unclear what the original authors of the NAF work did to handle rollouts that do not extend all the way to the horizon when fitting the time-varying local linear models. This impacts both generating the iLQG controllers and doing the imagination rollouts. In the case of the former, I opted to return a "null controller" which would return the 0 vector as the output action to simulate "doing nothing". In the case of the latter, I opted to terminate the imagination rollout when null dynamics are encountered.

6 Results

Reward over Train Steps

For each experiment, I conducted periodic evaluations to check the average total reward and episode length over 5 rollouts. For the classic control environments, this was done every 1000 training steps, whereas for Simglucose this was done every 5000 training steps. Shown below are the plots for the Pendulum, Mountain Car Continuous, and Simglucose (Kovatchev reward). The average total reward is in red and the average episode length is in gray. For all the plots in more detail, check Appendix A.

(a) DDPG Pendulum (b) TD3 Pendulum (c) NAF Pendulum (d) NAF-MBA Pend.

Figure 2: Pendulum Training Evaluations

(a) DDPG MCC (b) TD3 MCC (c) NAF MCC (d) NAF-MBA MCC

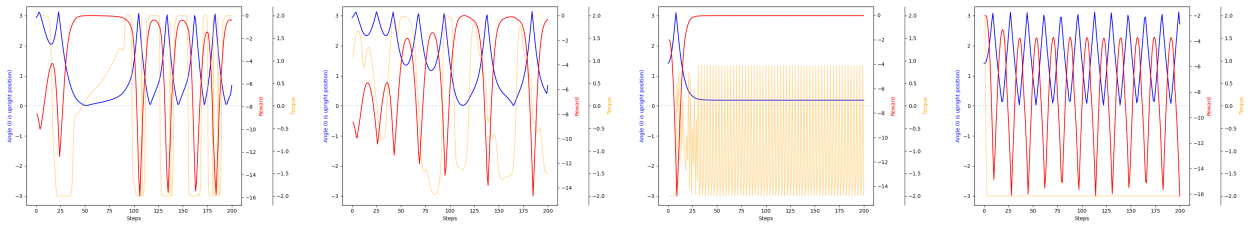
Figure 3: Mountain Car Continuous Training Evaluations

(a) DDPG SG:K (b) TD3 SG:K (c) NAF SG:K (d) NAF-MBA SG:K

Figure 4: Simglucose (Kovatchev Reward) Training Evaluations

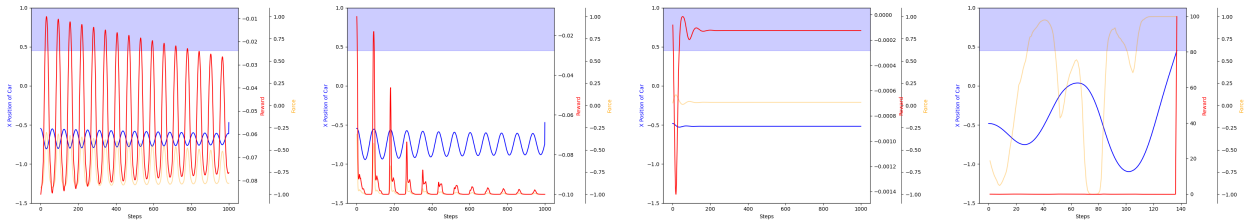
Final Evaluations

For each experiment, a final evaluation was conducted using the trained models. A total of 9 rollouts were performed, and for each individual rollout the observation is in blue, the action in orange, and the reward in red. Additionally, the dotted blue line indicates the observation goal for Pendulum, and the shaded blue regions indicate the observation goals for Mountain Car Continuous and Simglucose. Again, to see all the figures in more detail, check Appendix A.



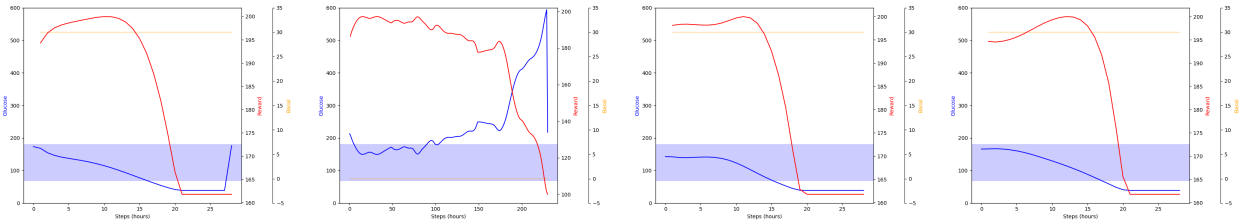
(a) DDPG Pendulum (b) TD3 Pendulum (c) NAF Pendulum (d) NAF-MBA Pend.

Figure 5: Pendulum Final Evaluation, Rollout 5



(a) DDPG MCC (b) TD3 MCC (c) NAF MCC (d) NAF-MBA MCC

Figure 6: Mountain Car Continuous Final Evaluation, Rollout 5



(a) DDPG SG:K (b) TD3 SG:K (c) NAF SG:K (d) NAF-MBA SG:K

Figure 7: Simglucose (Kovatchev Reward) Final Evaluation, Rollout 5

7 Conclusions

Pendulum

It is quite clear to see from both the training evaluations and the final rollout evaluation that NAF performed the best on Pendulum. Next best is TD3, closely followed by DDPG. Interestingly, last was NAF-MBA. I suspect this is the case because this is quite the simple environment where the local linear models may have only served to add unnecessary noise.

Mountain Car Continuous

In this environment, the training and final evaluations suggest that NAF-MBA performed the best, followed by DDPT and TD3 in no particular order, then NAF. NAF-MBA, at around 18000 environment steps of training, learns how to successfully drive the car to the flag (indicated by a consisted 100 reward and short episode length). DDPG and TD3, however, look to be less successful. There are a few instances of success, but it seems they are more coincidences of exploration rather than actual learning. For NAF, not once did it successfully guide the car to the flag.

I believe these models performed as they did because of the reward function for the MCC environment. The reward is very sparse, only giving a positive value when the car reaches the flag. Perhaps this meant that there was not enough exploration done by the DDPG, TD3, and NAF models to truly learn where the maximum reward comes from. On the other hand, the imagination rollouts in NAF-MBA provided enough exploration.

Simglucose

It is hard to come to any concrete conclusion on this environment because the results were not very successful, regardless of the algorithm or reward function. The reason for this may have more to do with the type of network chosen. All experiments used a MLP architecture, but a network which could handle past context such as RNNs may have been a better choice because unlike the classic control environments, doing an action (administering insulin) is not immediately reflected in the observation (BG levels). Yet, even with all this said, NAF-MBA using the simple reward function looks to have performed best on the final evaluation rollouts. There appears to be an attempt at regulating BG levels within the 70-180 range, unlike most of the other experiments. The only other experiment with remotely similar results is DDPG with Magni reward.

Summary and Future Work

Although the results using Simglucose were not completely satisfactory, NAF-MBA looks like a promising approach. It managed to handle the sparse rewards of the MCC environment, and was able to achieve some sort of learning on Simglucose using the simple reward function. This suggests that the NAF-MBA approach is still applicable to domains outside robotics, and perhaps generally applicable to even those domains which are quite different from robotics. Of course, one must bring the right tool for the right job because the results on the Pendulum environment suggest that the approach is too strong and unstable for overly simple environments.

One obvious avenue for future work is to study different neural net architectures depending on the task. For example, the results of the Simglucose experiments may have been greatly improved had a RNN been employed rather than a MLP. Another potential area of work is to better study the trajectory optimization subroutine. Exploring better reward matrix shaping for iLQG may yield better results. Moreover, while this project used iLQG, there is nothing inherent to iLQG that the NAF-MBA algorithm requires. Exploring other trajectory optimization algorithms may alter the learning behavior of model.

GitHub Repository

The codebase can be found at `https://github.com/JustUnoptimized/DeepRL-Project`. Notably, an implementation for NAF [20] has been modified to add model based accelerations. The Stable Baselines 3 [21] MLP policy implementation for DDPG and TD3 has also been modified to add a batch norm layer after each hidden layer.

References

[1] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (1992) 279–292.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602 (2013).

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (7540) (2015) 529–533.

[4] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International conference on machine learning, Pmlr, 2014, pp. 387–395.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971 (2015).

[6] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International conference on machine learning, PMLR, 2018, pp. 1587–1596.

[7] H. Hasselt, Double q-learning, Advances in neural information processing systems 23 (2010).

[8] S. Gu, T. Lillicrap, I. Sutskever, S. Levine, Continuous deep q-learning with model-based acceleration, in: International conference on machine learning, PMLR, 2016, pp. 2829–2838.

[9] M. E. Harmon, L. C. Baird III, Multi-player residual advantage learning with general function approximation, Wright Laboratory, WL/AACF, Wright-Patterson Air Force Base, OH (1996) 45433–7308.

[10] L. C. Baird, Advantage updating, Tech. rep., Technical report wl-tr-93-1146, Wright Patterson AFB OH (1993).

[11] R. S. Sutton, Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, in: Machine learning proceedings 1990, Elsevier, 1990, pp. 216–224.

[12] E. Todorov, W. Li, A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems, in: Proceedings of the 2005, American Control Conference, 2005., IEEE, 2005, pp. 300–306.

[13] Y. Tassa, T. Erez, E. Todorov, Synthesis and stabilization of complex behaviors through online trajectory optimization, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 4906–4913.

[14] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, O. G. Younis, Gymnasium (Mar 2023). doi:10.5281/zenodo.8127026.
URL https://zenodo.org/record/8127025

[15] J. Xie, Simglucose (2018).
URL https://github.com/jxx123/simglucose

[16] C. D. Man, F. Micheletto, D. Lv, M. Breton, B. Kovatchev, C. Cobelli, The uva/padova type 1 diabetes simulator: new features, Journal of diabetes science and technology 8 (1) (2014) 26–34.

[17] B. P. Kovatchev, M. Straume, D. J. Cox, L. S. Farhy, Risk analysis of blood glucose data: Az quantitative approach to optimizing the control of insulin dependent diabetes, Computational and Mathematical Methods in Medicine 3 (1) (2000) 1–10.

[18] B. P. Kovatchev, Metrics for glycaemic control—from hba1c to continuous glucose monitoring, Nature Reviews Endocrinology 13 (7) (2017) 425–436.

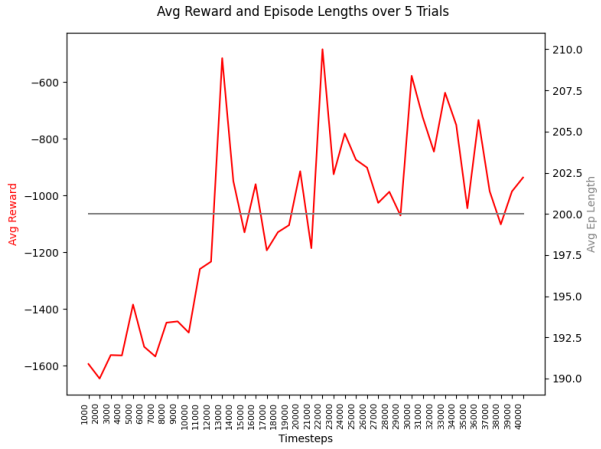
[19] L. Magni, D. M. Raimondo, L. Bossi, C. Dalla Man, G. De Nicolao, B. Kovatchev, C. Cobelli, Model predictive control of type 1 diabetes: an in silico trial (2007).

[20] S. Dittert, Pytorch implementation of normalized advantage function, `https://github.com/BY571/NAF` (2020).

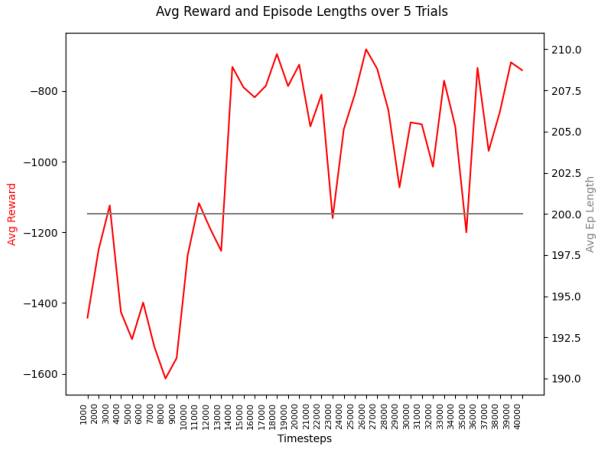
[21] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-baselines3: Reliable reinforcement learning implementations, Journal of Machine Learning Research 22 (268) (2021) 1–8.
URL `http://jmlr.org/papers/v22/20-1364.html`

Appendix

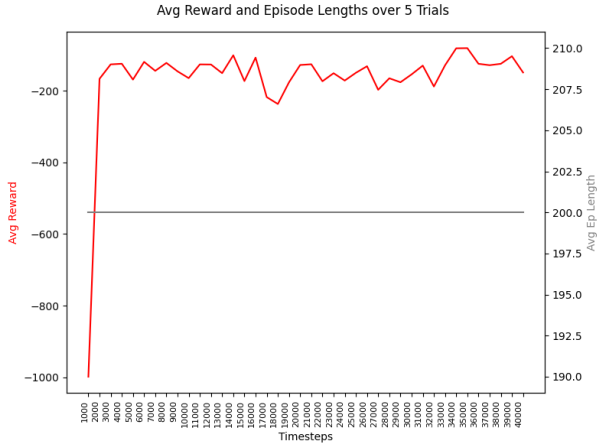
Training and Final Evaluation Plots



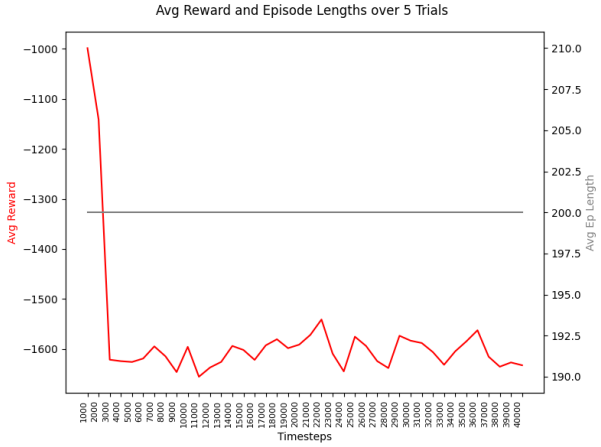
(a) DDPG Pendulum



(b) TD3 Pendulum

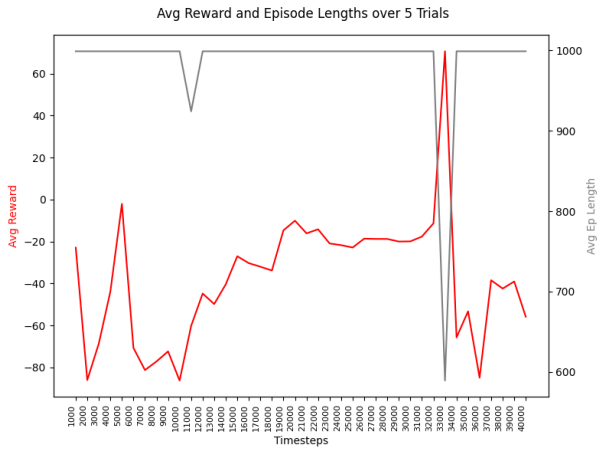


(c) NAF Pendulum

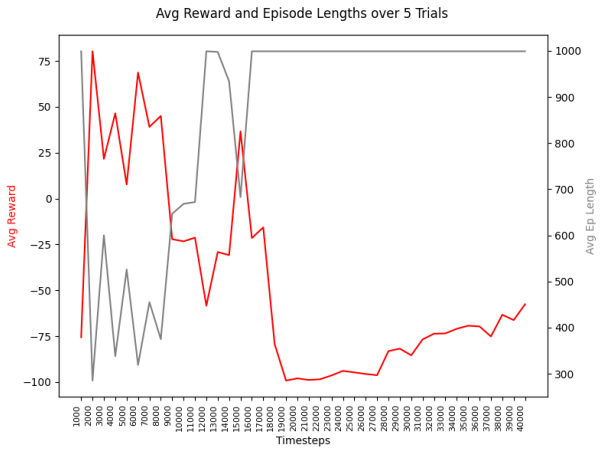


(d) NAF-MBA Pend.

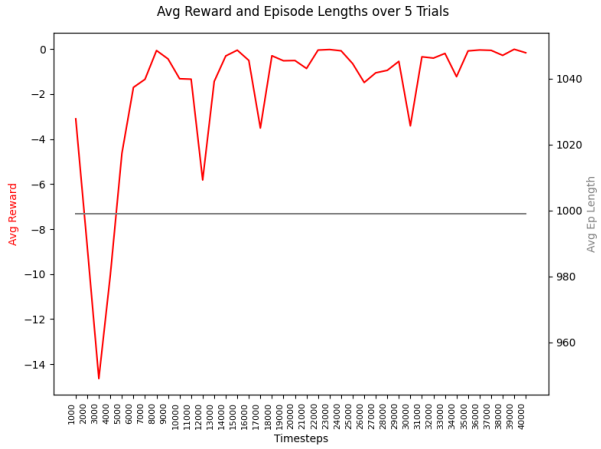
Figure 8: Pendulum Training Evaluations



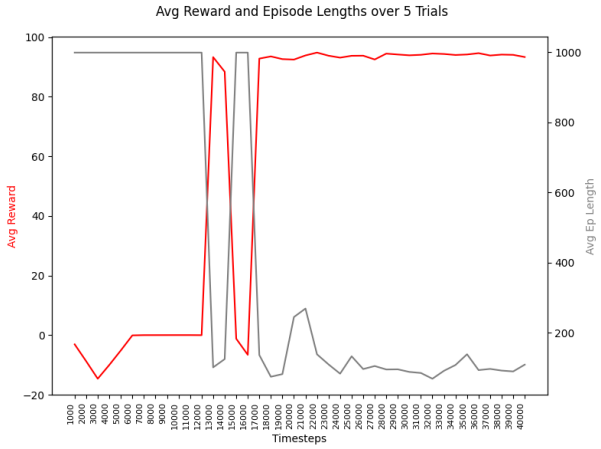
(a) DDPG MCC



(b) TD3 MCC

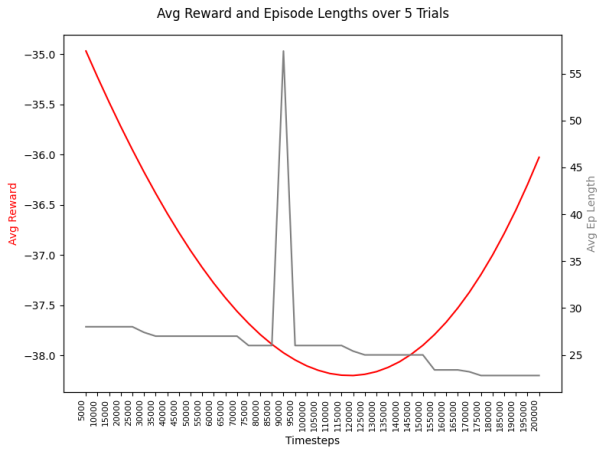


(c) NAF MCC

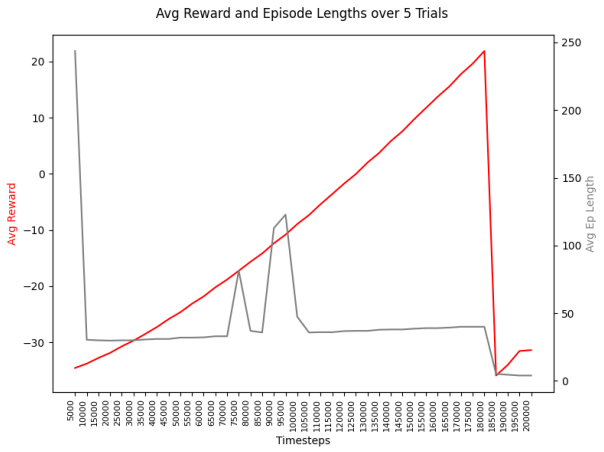


(d) NAF-MBA MCC

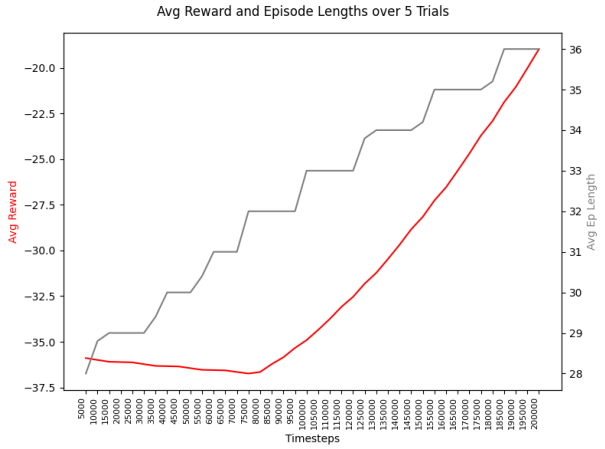
Figure 9: Mountain Car Continuous Training Evaluations



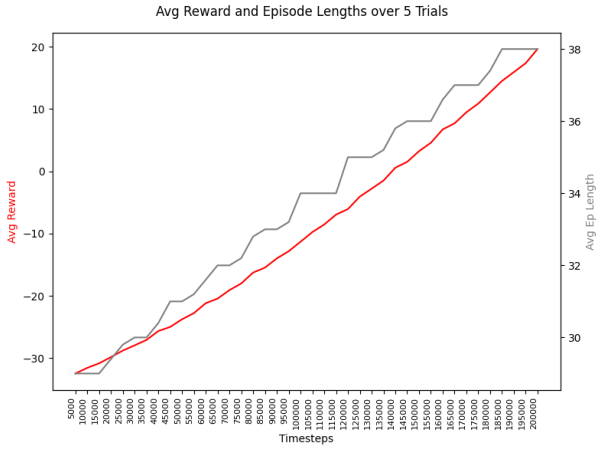
(a) DDPG SG:D



(b) TD3 SG:D

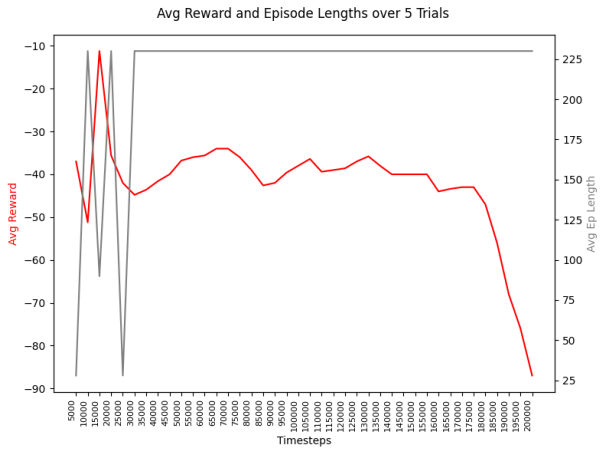


(c) NAF SG:D

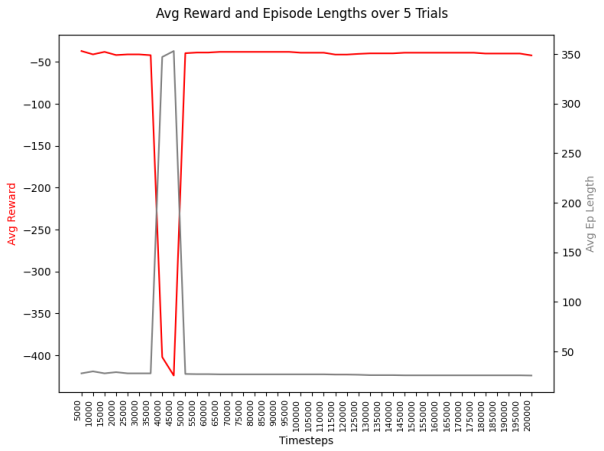


(d) NAF-MBA SG:D

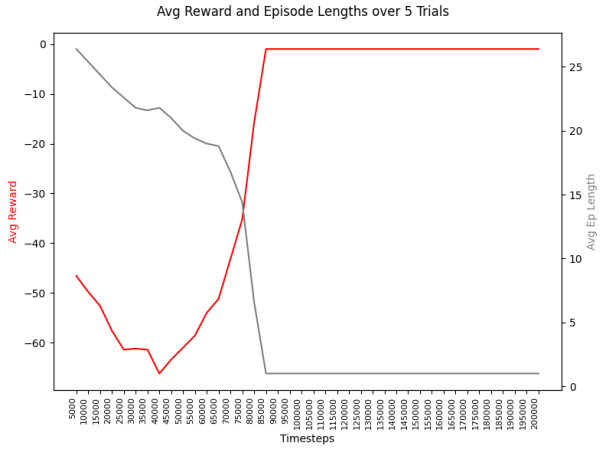
Figure 10: Simglucose (Default Reward) Training Evaluations



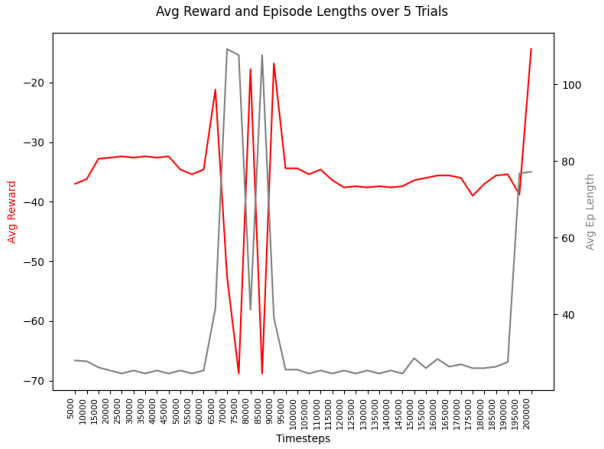
(a) DDPG SG:S



(b) TD3 SG:S

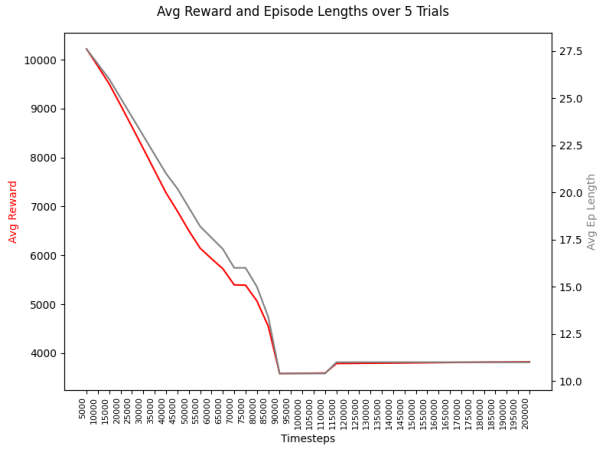


(c) NAF SG:S

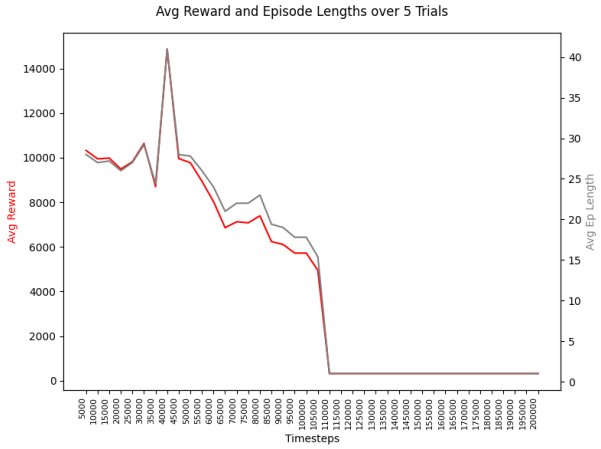


(d) NAF-MBA SG:S

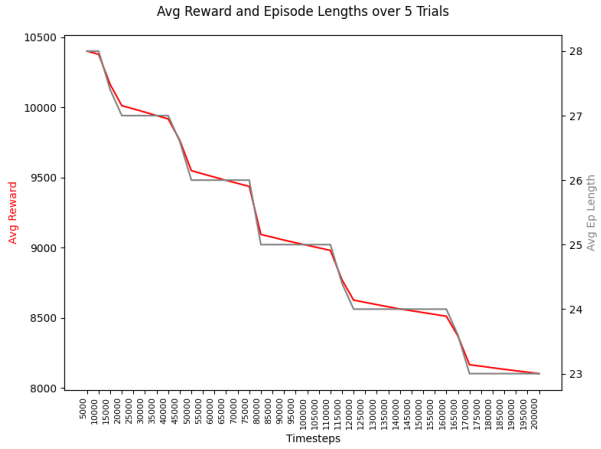
Figure 11: Simglucose (Simple Reward) Training Evaluations



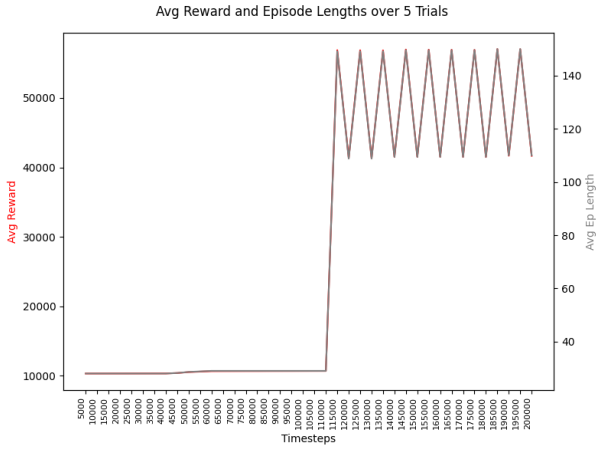
(a) DDPG SG:M



(b) TD3 SG:M

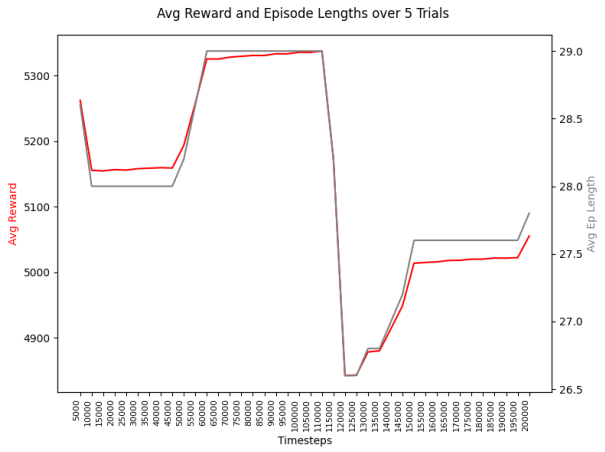


(c) NAF SG:M

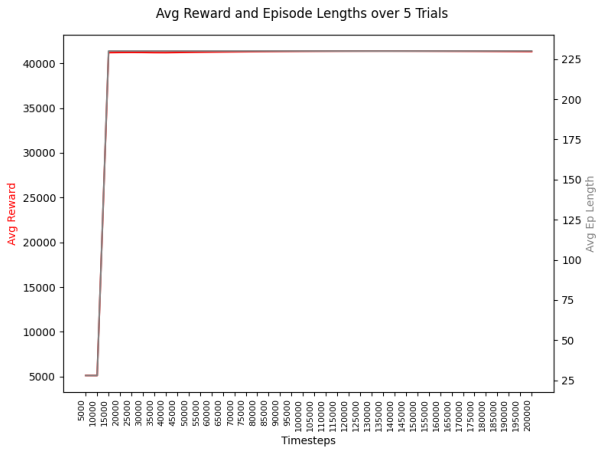


(d) NAF-MBA SG:M

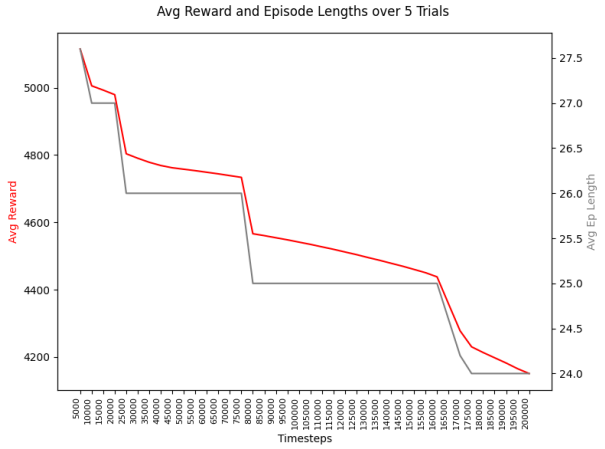
Figure 12: Simglucose (Magni Reward) Training Evaluations



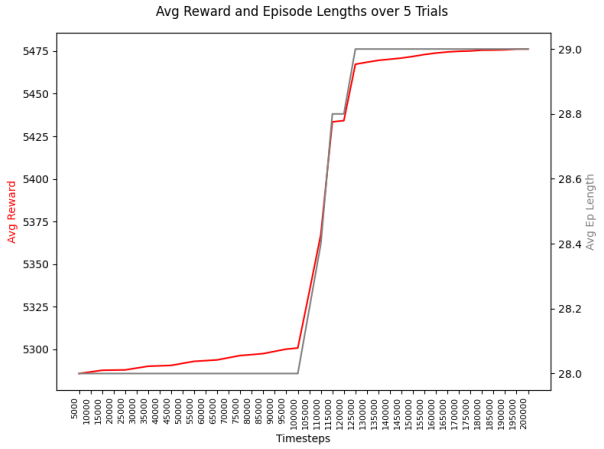
(a) DDPG SG:K



(b) TD3 SG:K



(c) NAF SG:K



(d) NAF-MBA SG:K

Figure 13: Simglucose (Kovatchev Reward) Training Evaluations

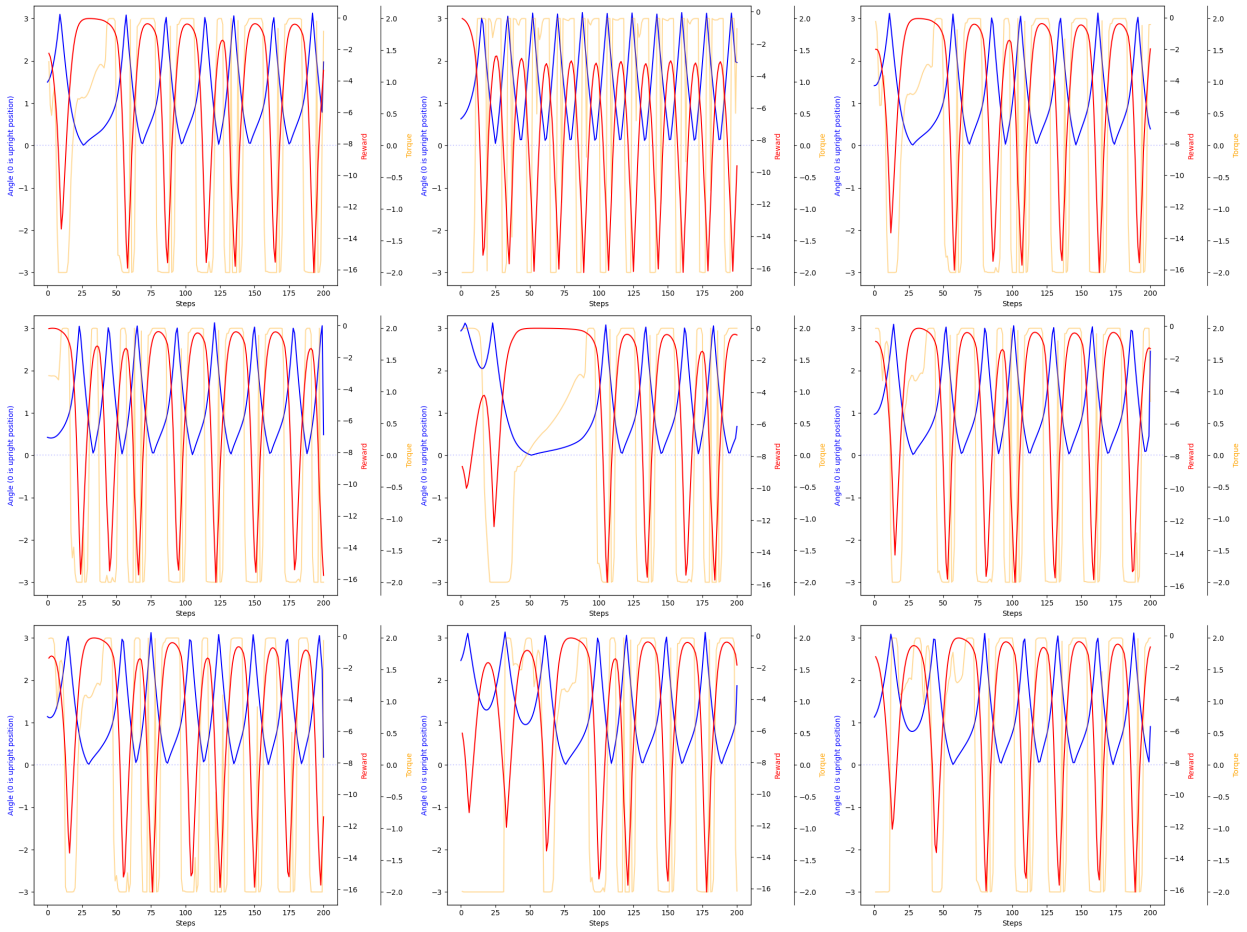


Figure 14: DDPG Pendulum Final Evaluations

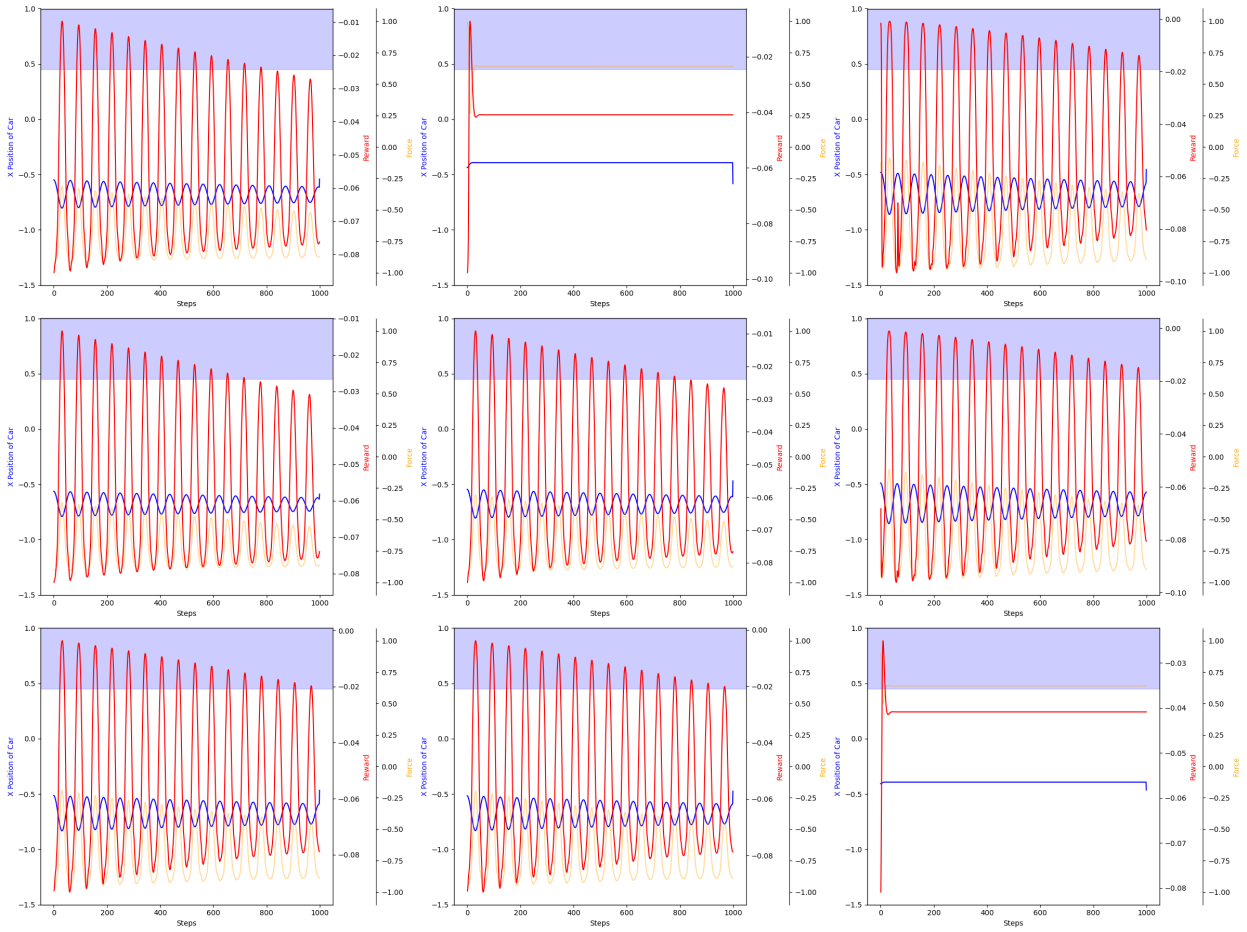


Figure 15: DDPG Mountain Car Continuous Final Evaluations

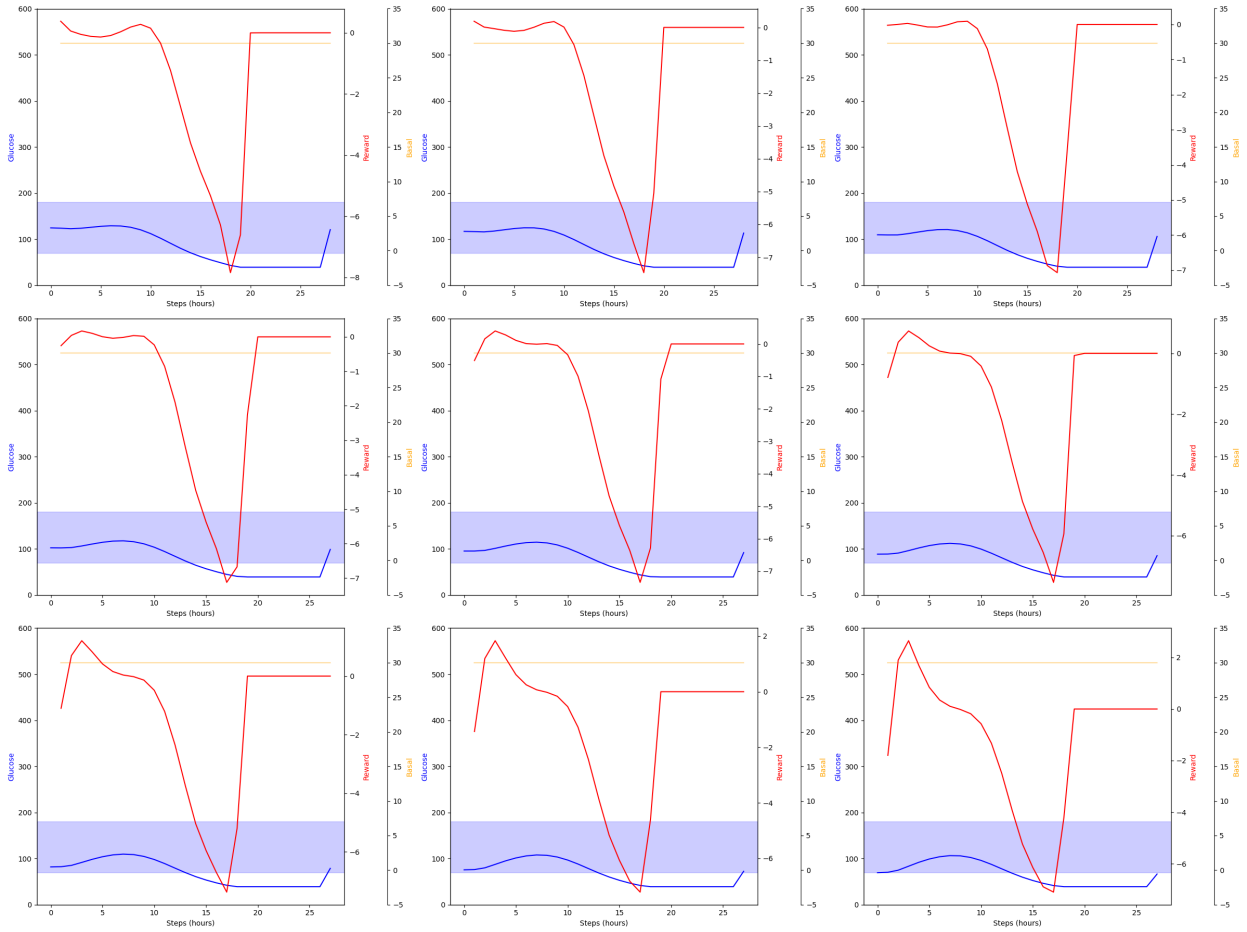


Figure 16: DDPG Simglucose (Default Reward) Final Evaluations

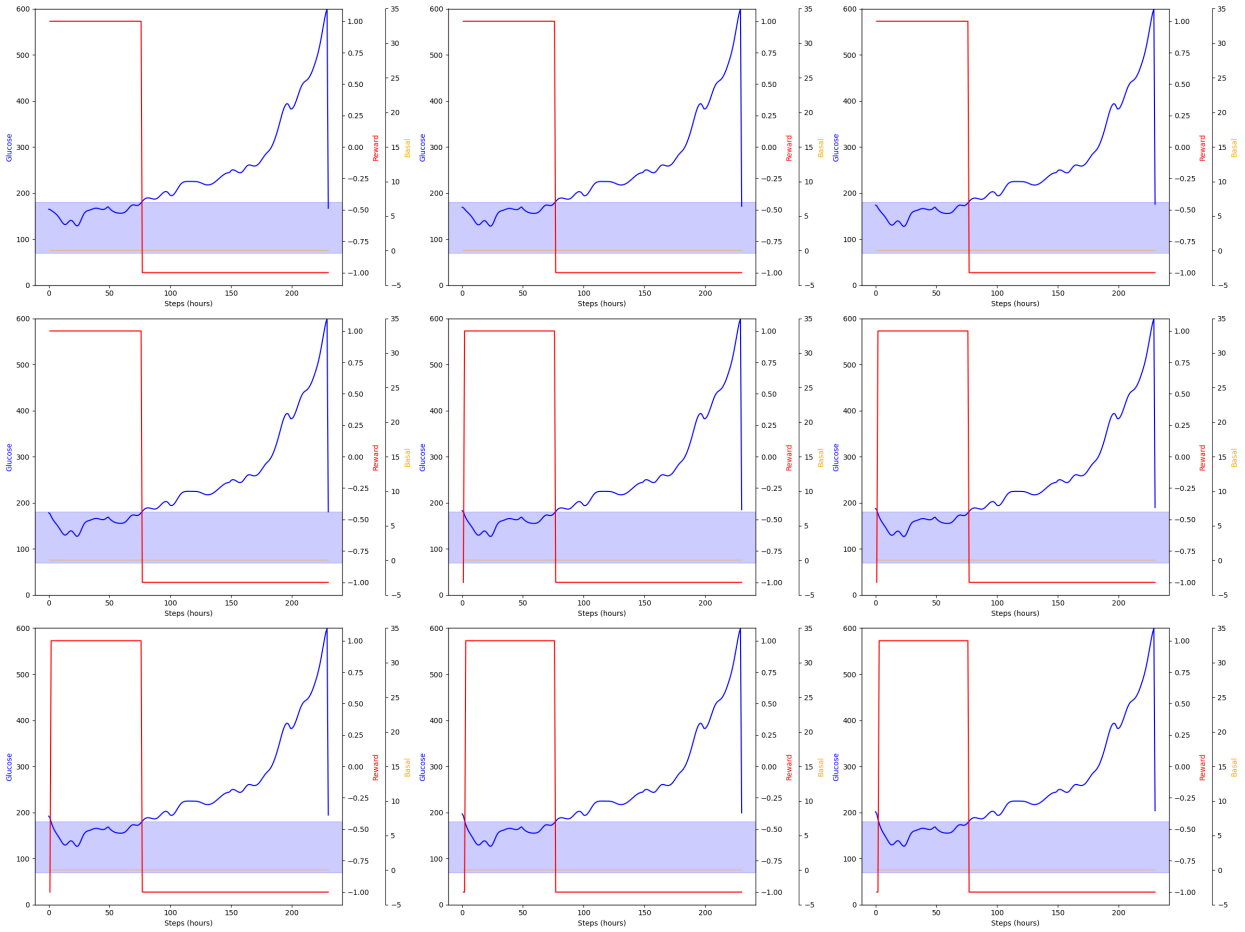


Figure 17: DDPG Simglucose (Simple Reward) Final Evaluations

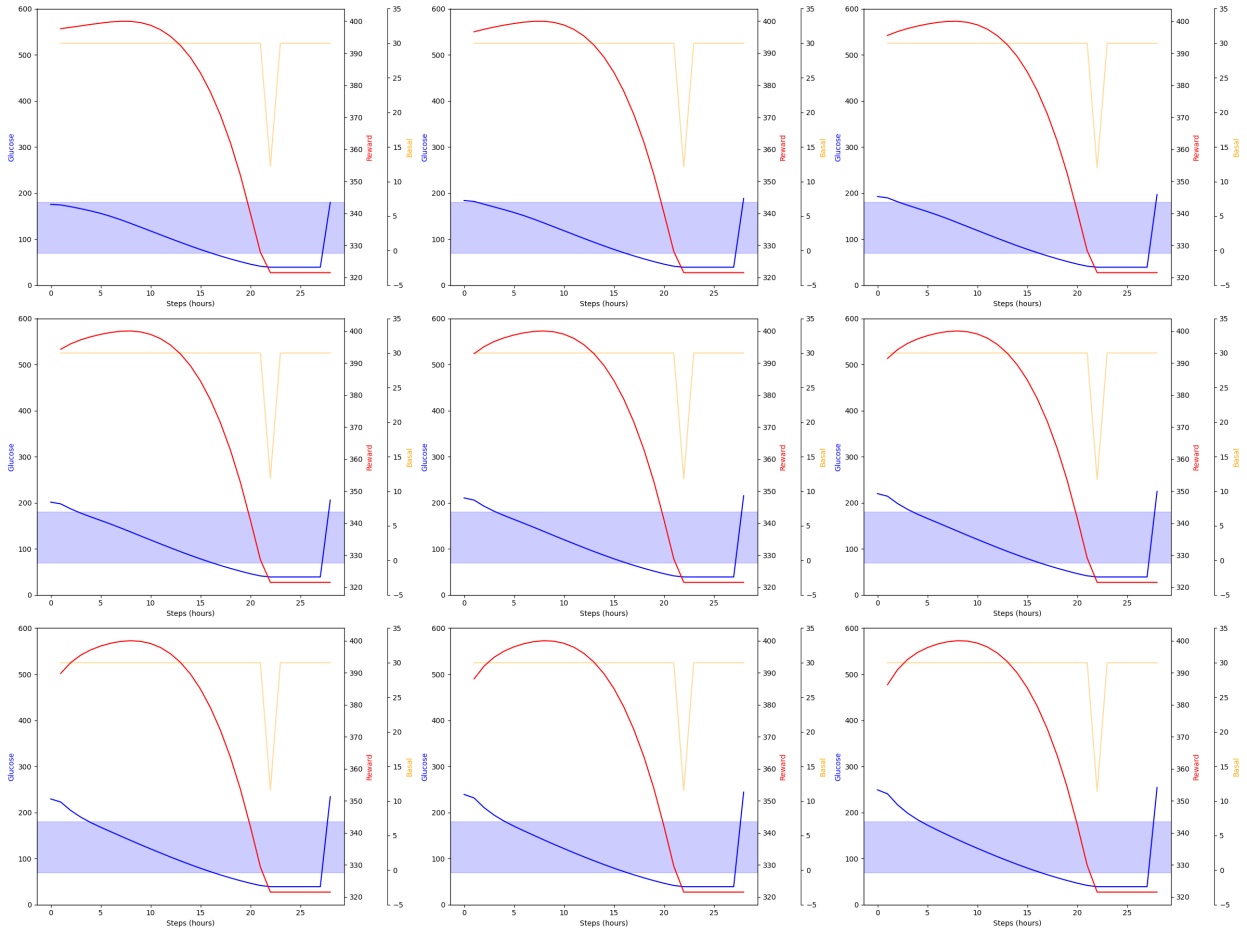


Figure 18: DDPG Simglucose (Magni Reward) Final Evaluations

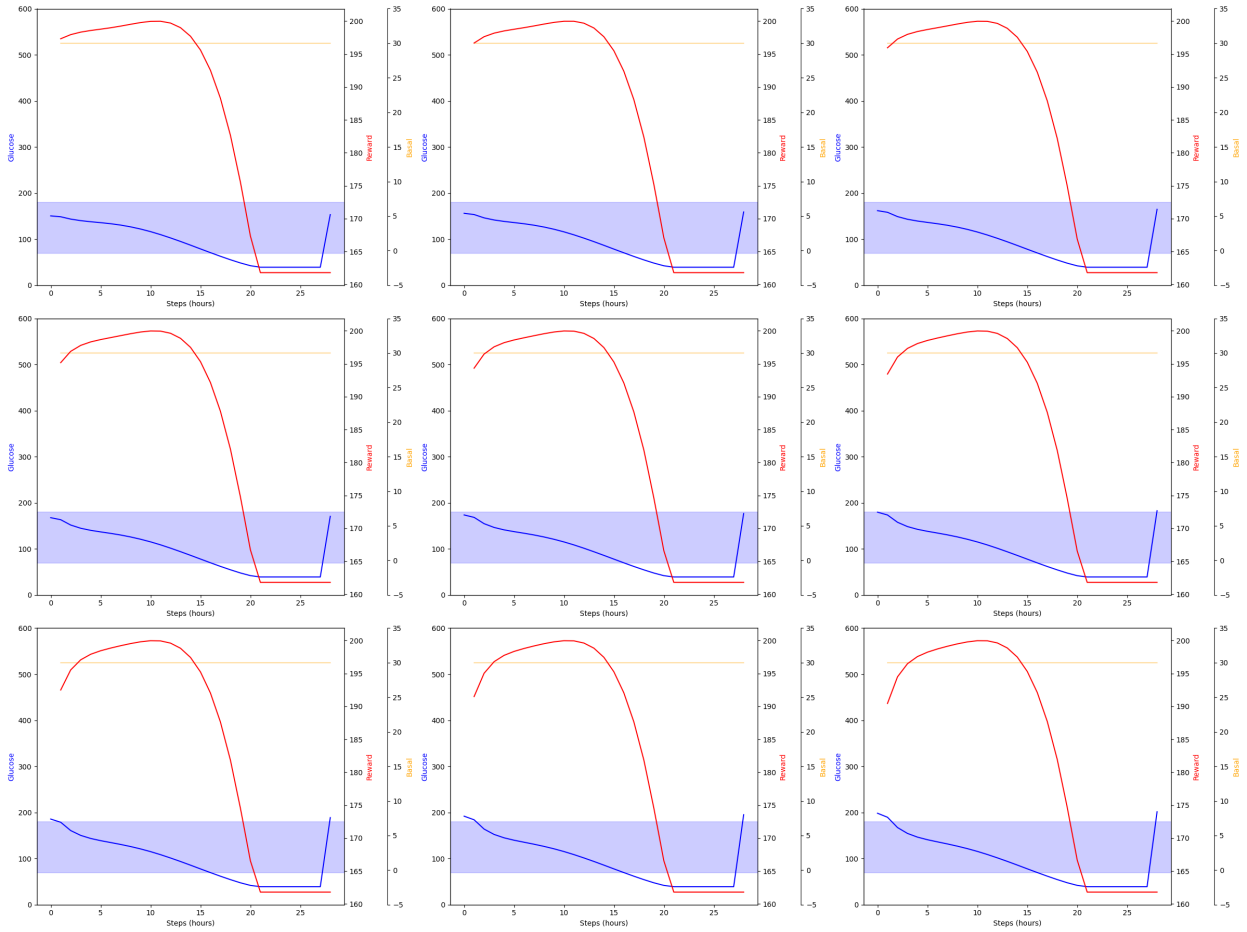


Figure 19: DDPG Simglucose (Kovatchev Reward) Final Evaluations

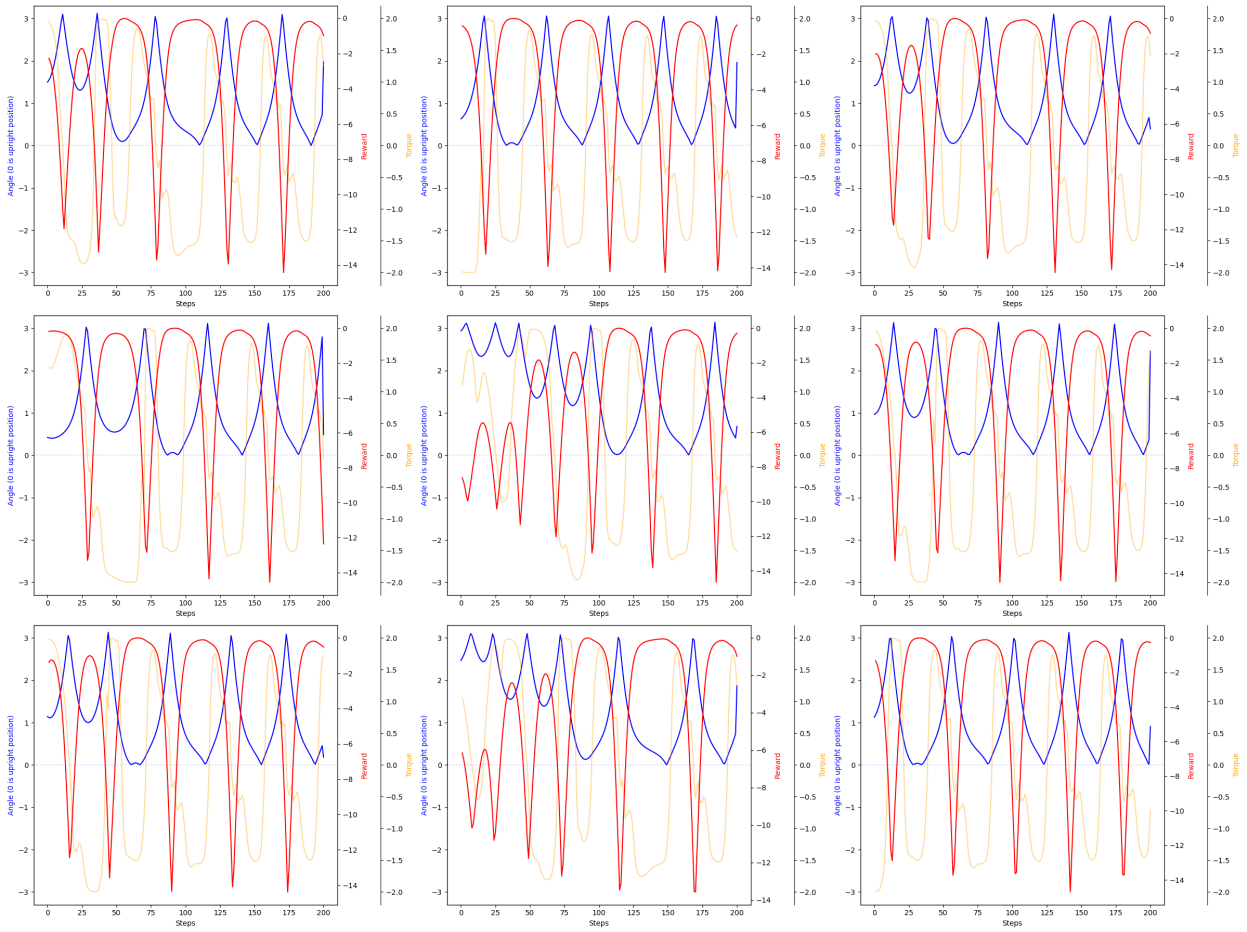


Figure 20: TD3 Pendulum Final Evaluations

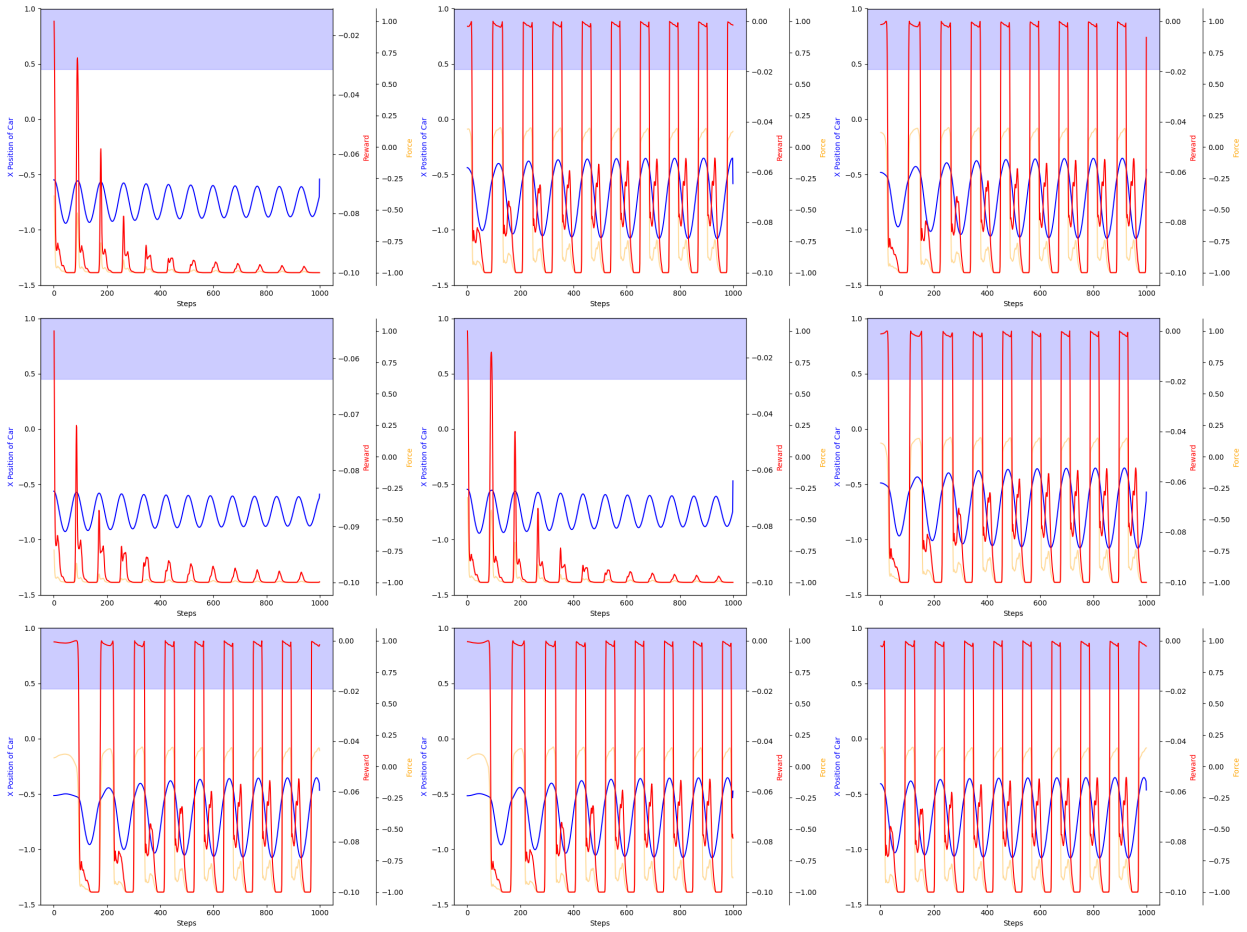


Figure 21: TD3 Mountain Car Continuous Final Evaluations

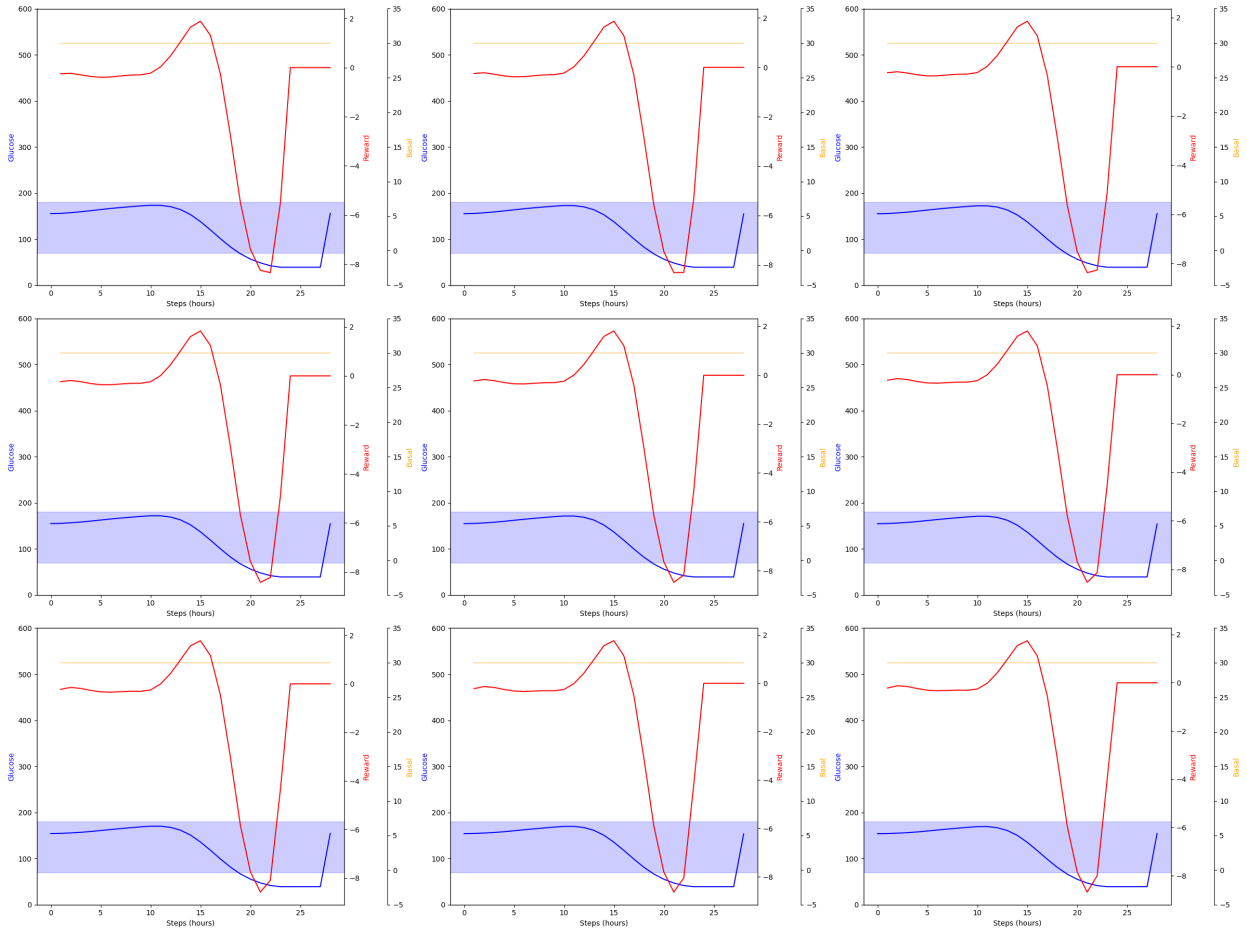


Figure 22: TD3 Simglucose (Default Reward) Final Evaluations

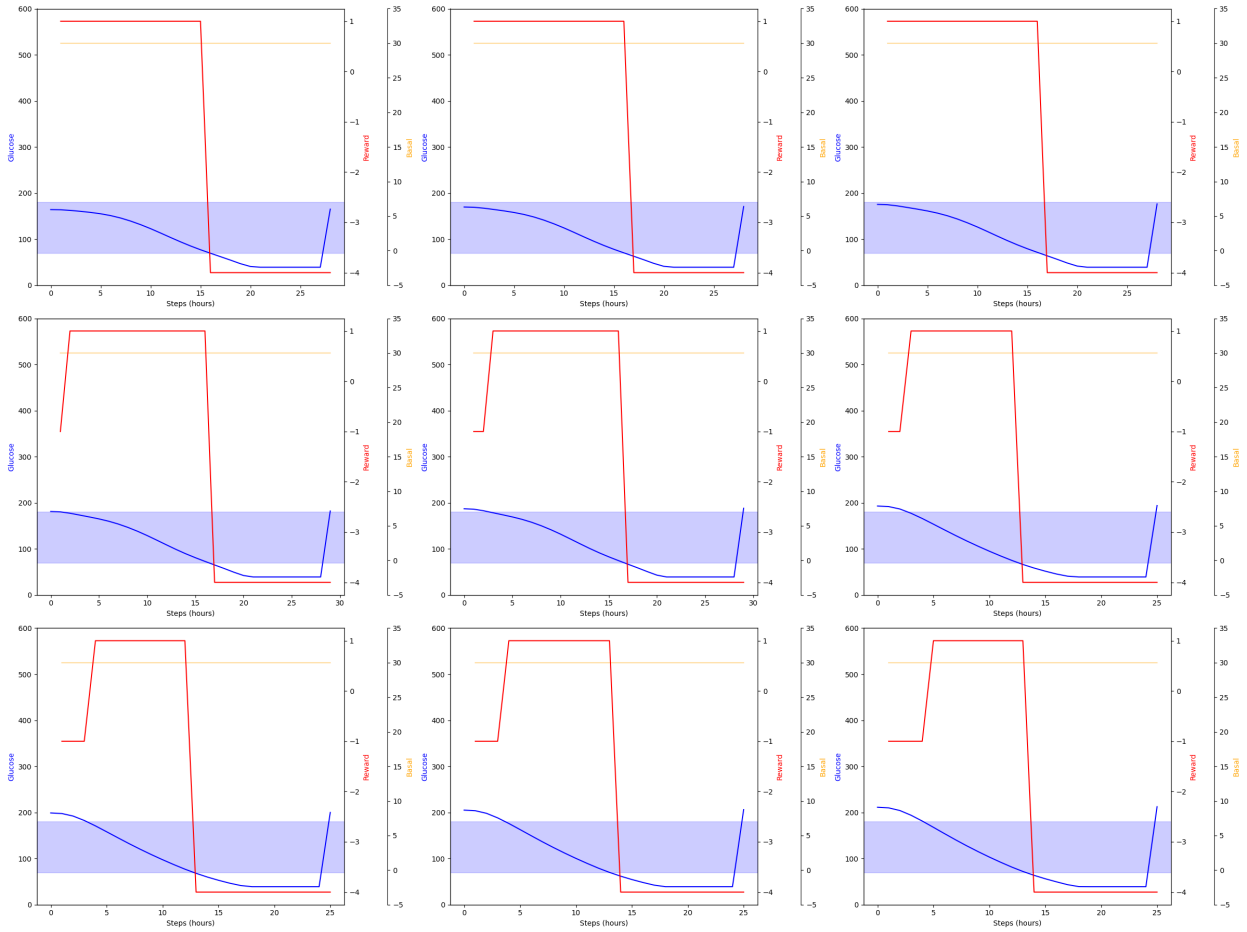


Figure 23: TD3 Simglucose (Simple Reward) Final Evaluations

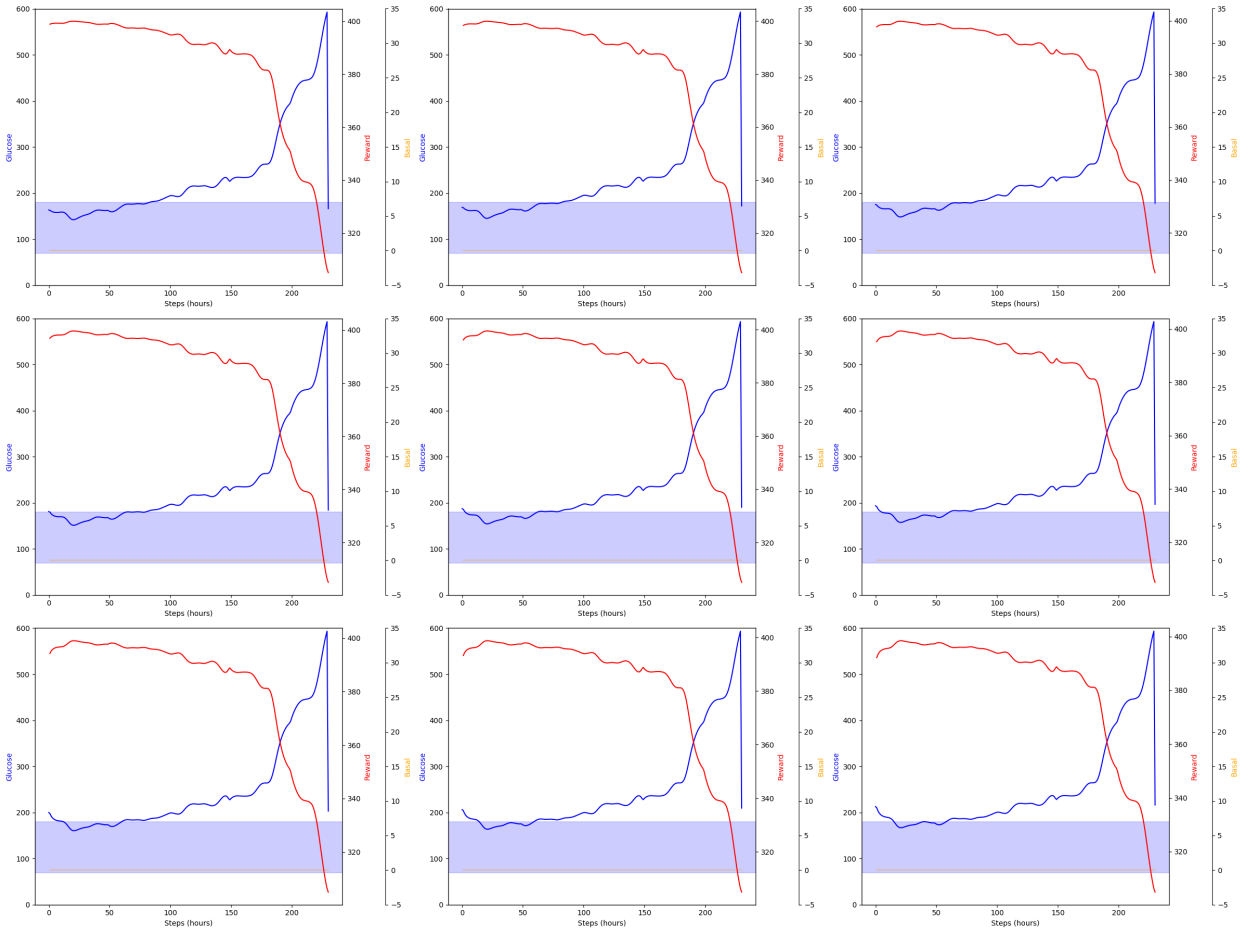


Figure 24: TD3 Simglucose (Magni Reward) Final Evaluations

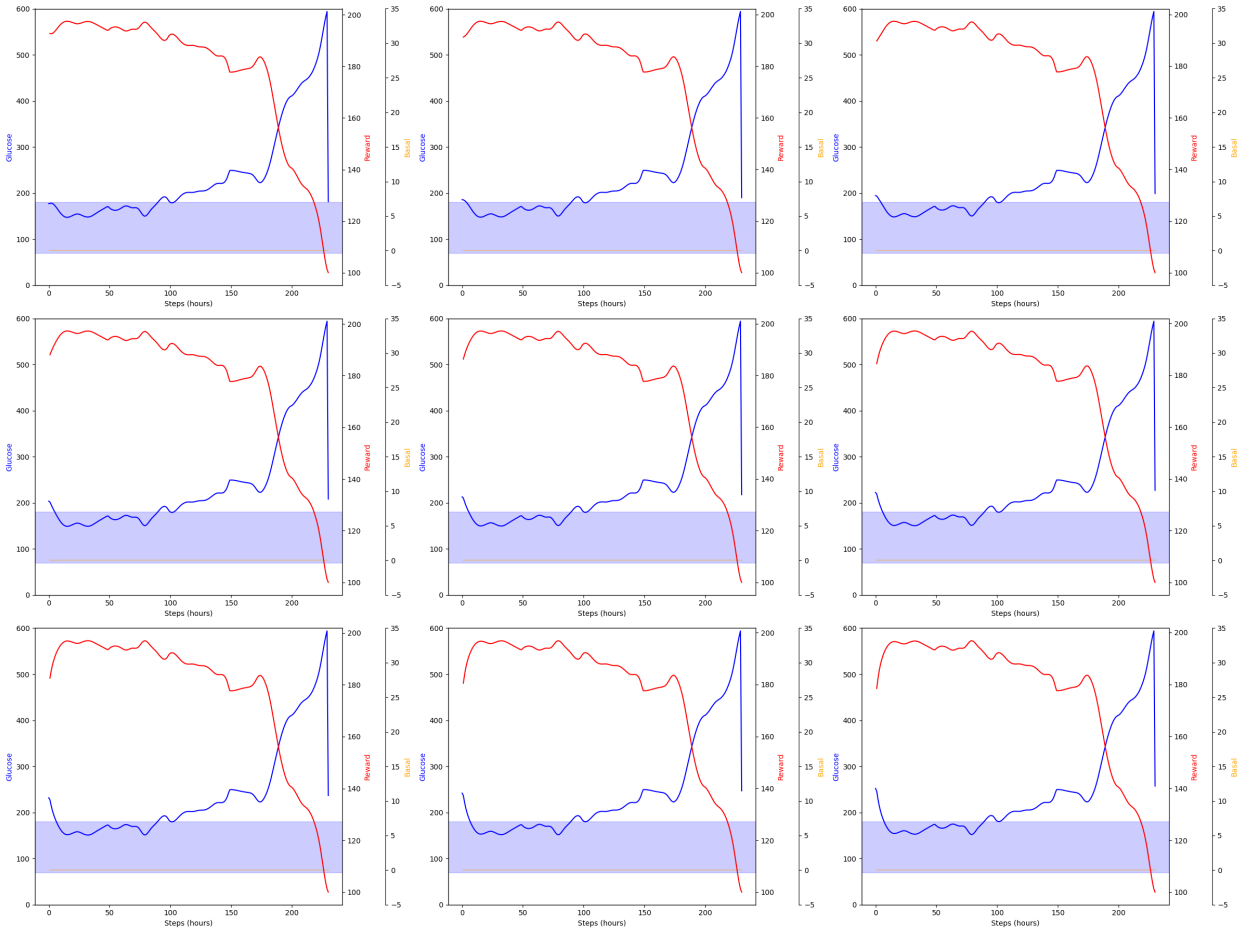


Figure 25: TD3 Simglucose (Kovatchev Reward) Final Evaluations

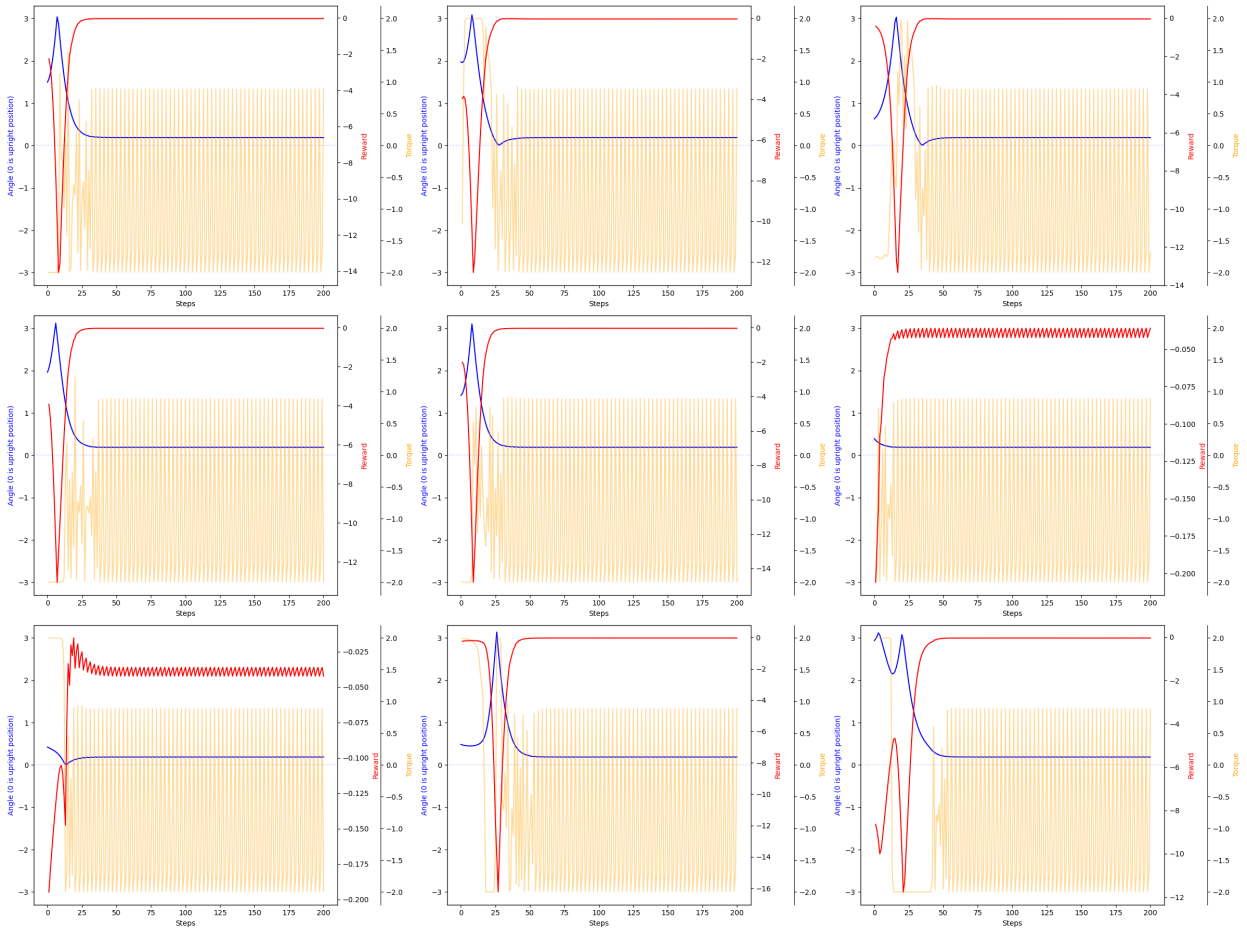


Figure 26: NAF Pendulum Final Evaluations

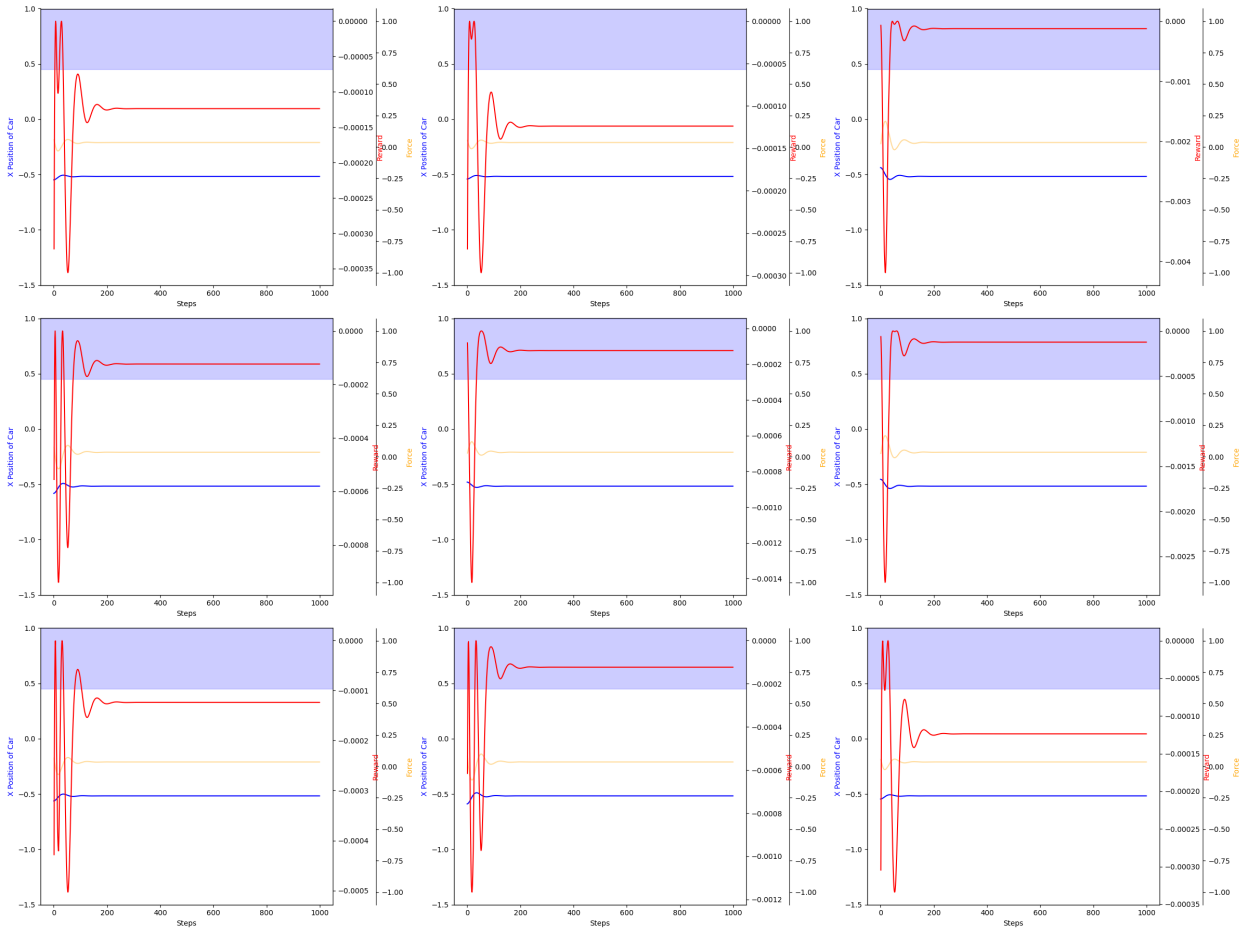


Figure 27: NAF Mountain Car Continuous Final Evaluations

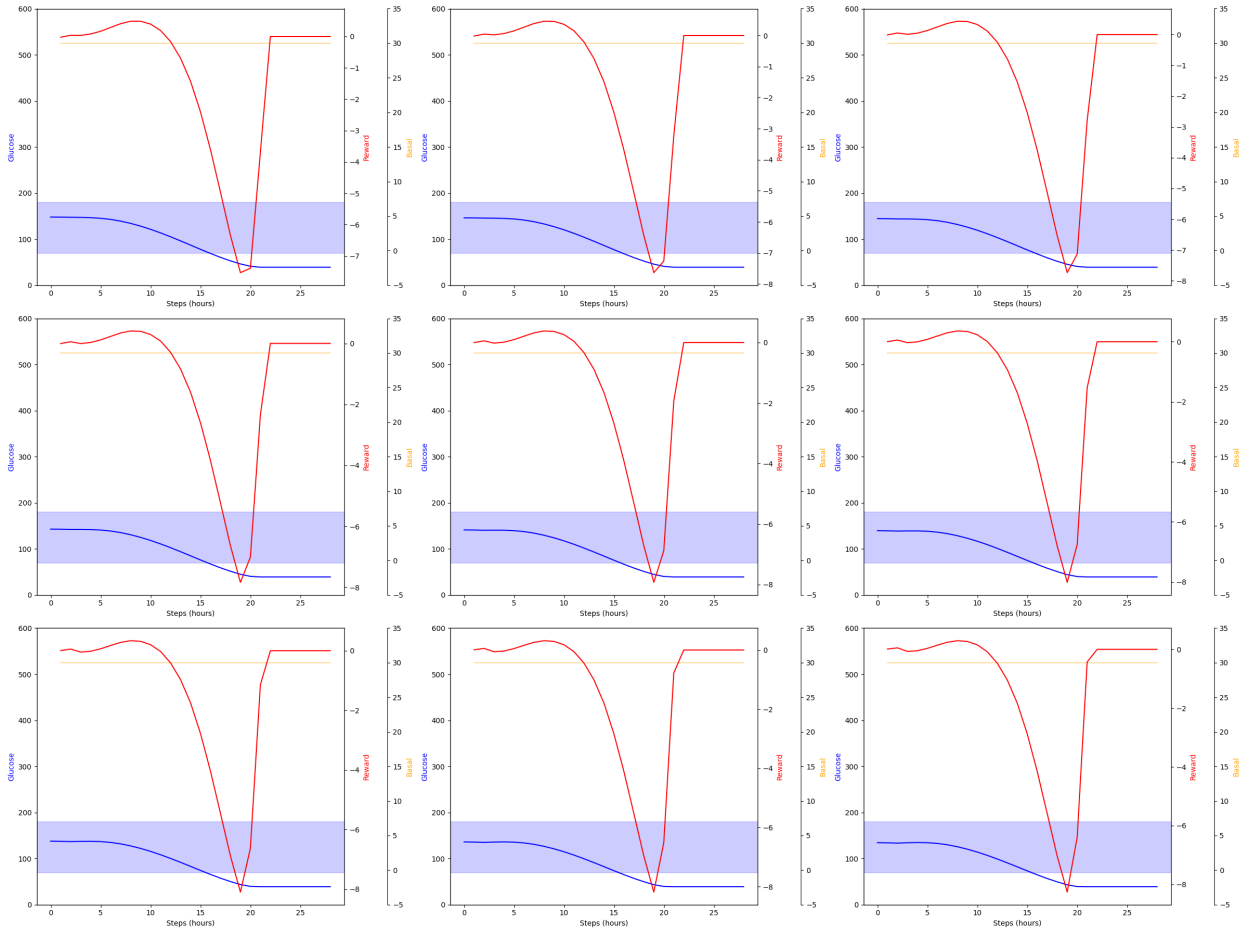


Figure 28: NAF Simglucose (Default Reward) Final Evaluations

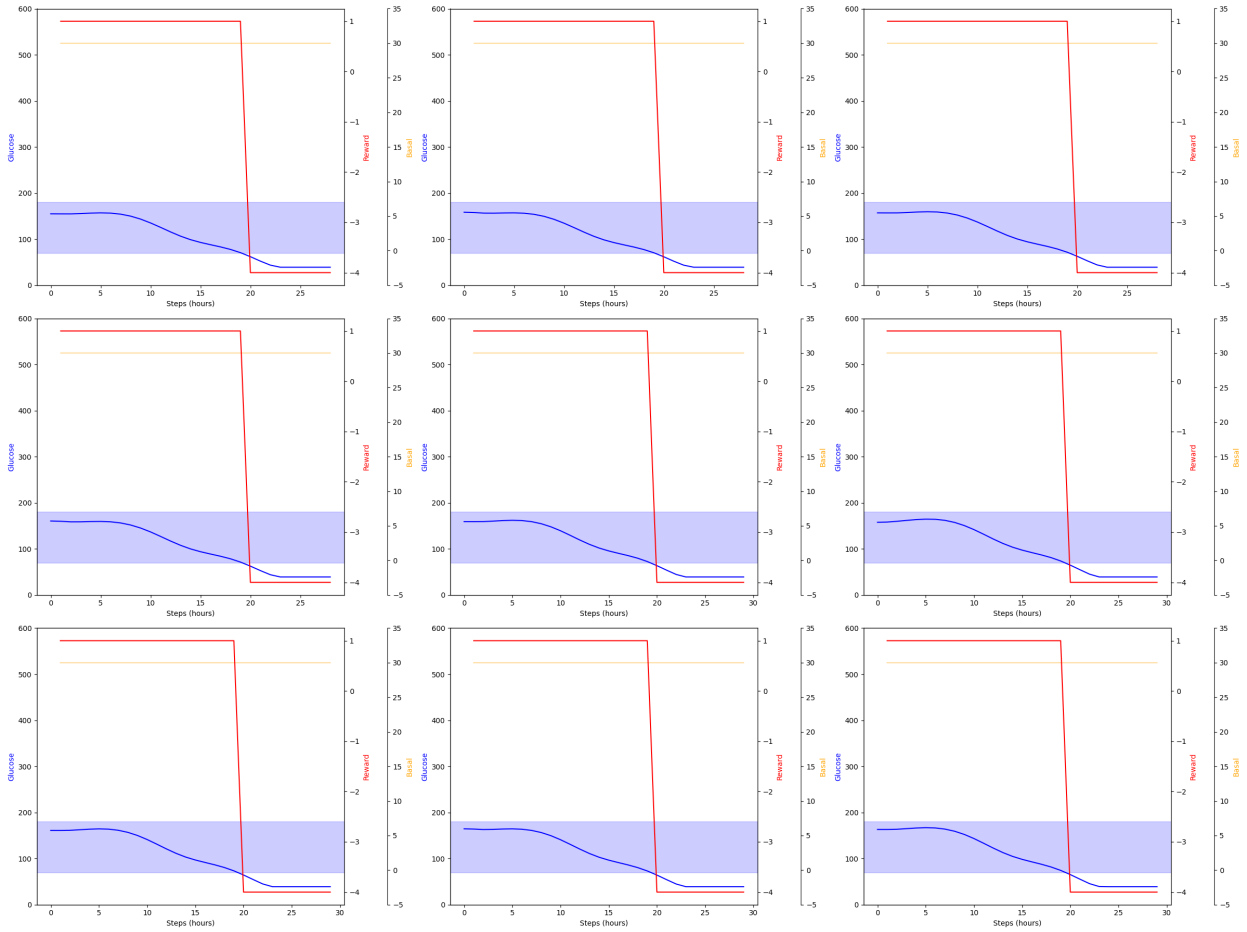


Figure 29: NAF Simglucose (Simple Reward) Final Evaluations

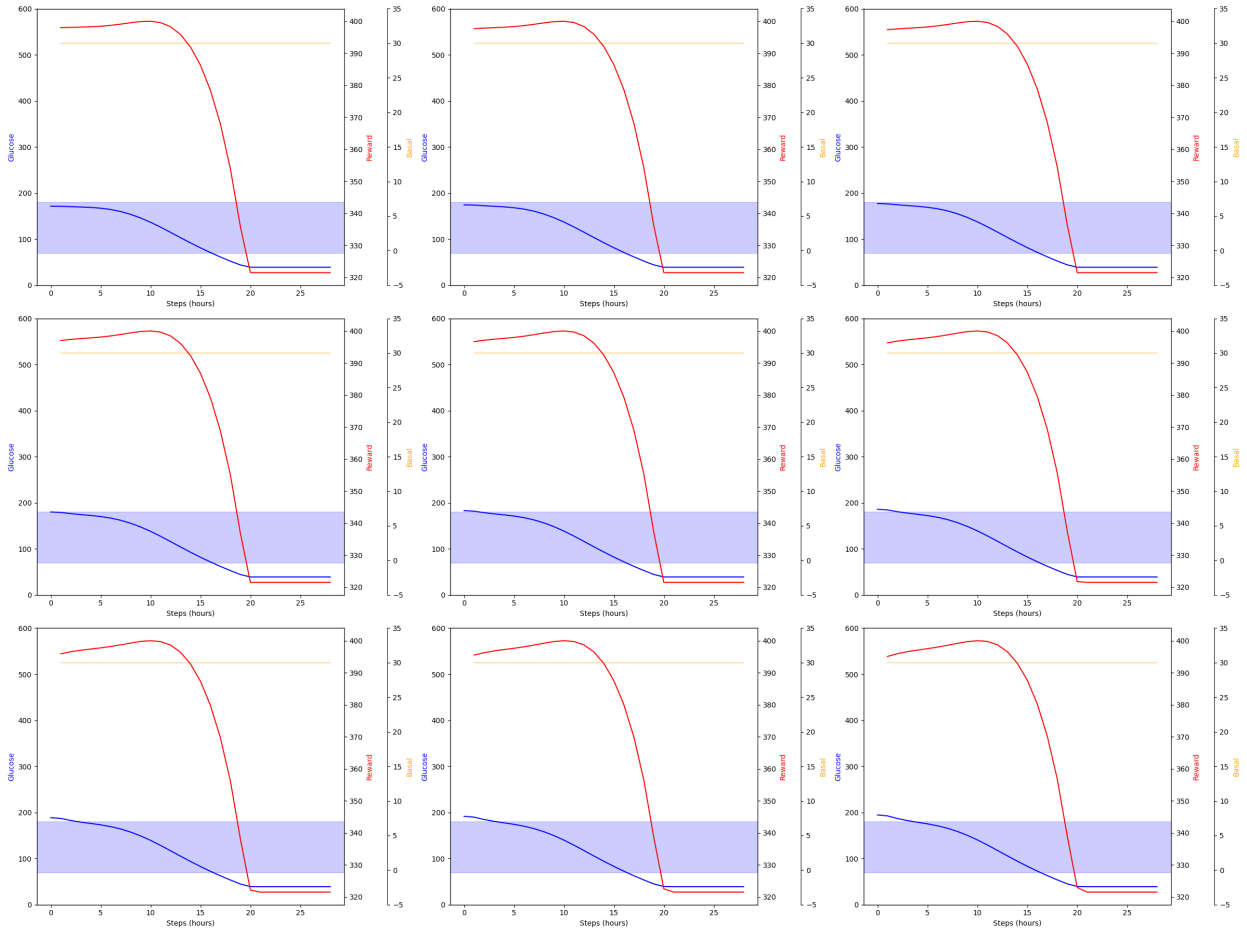


Figure 30: NAF Simglucose (Magni Reward) Final Evaluations

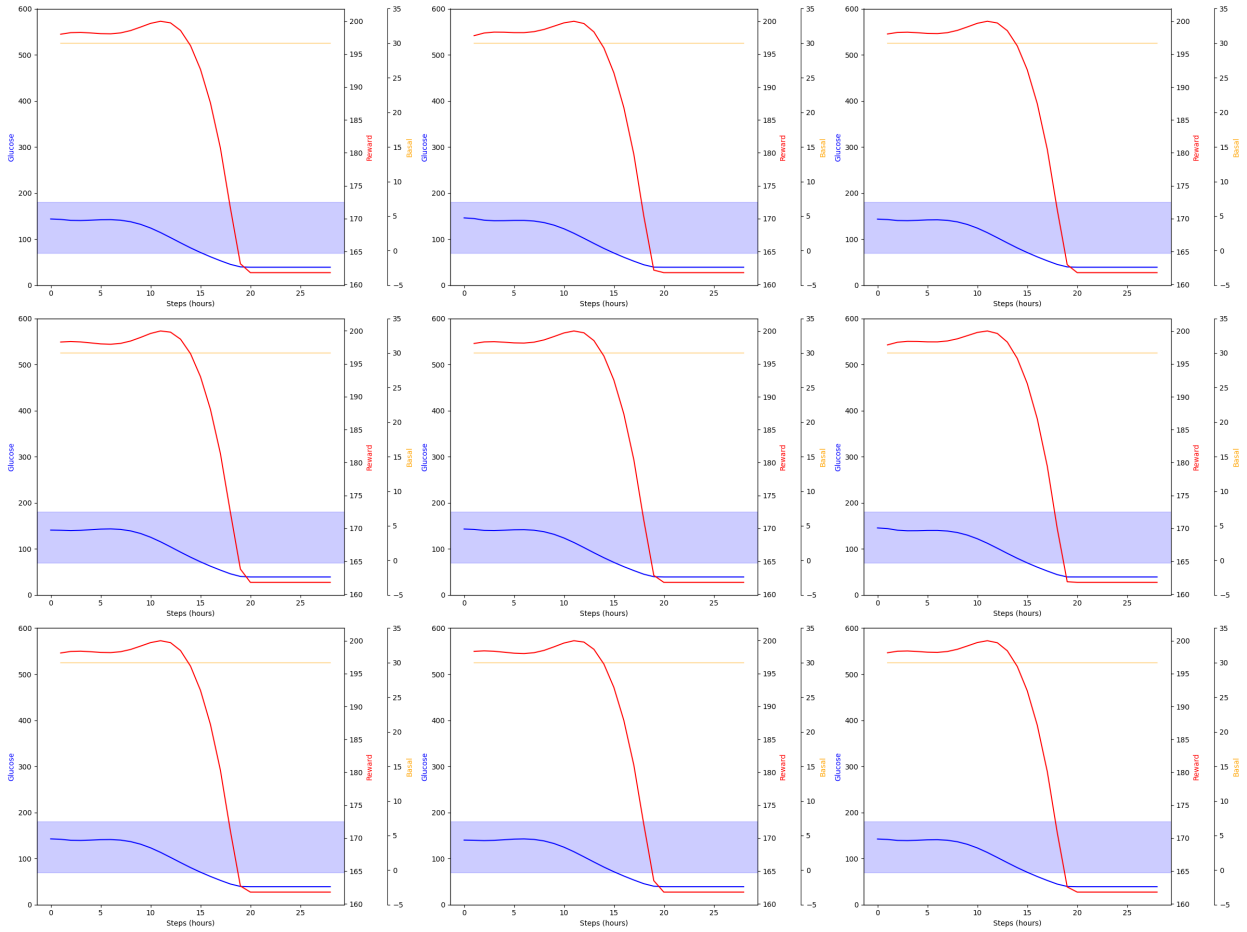


Figure 31: NAF Simglucose (Kovatchev Reward) Final Evaluations

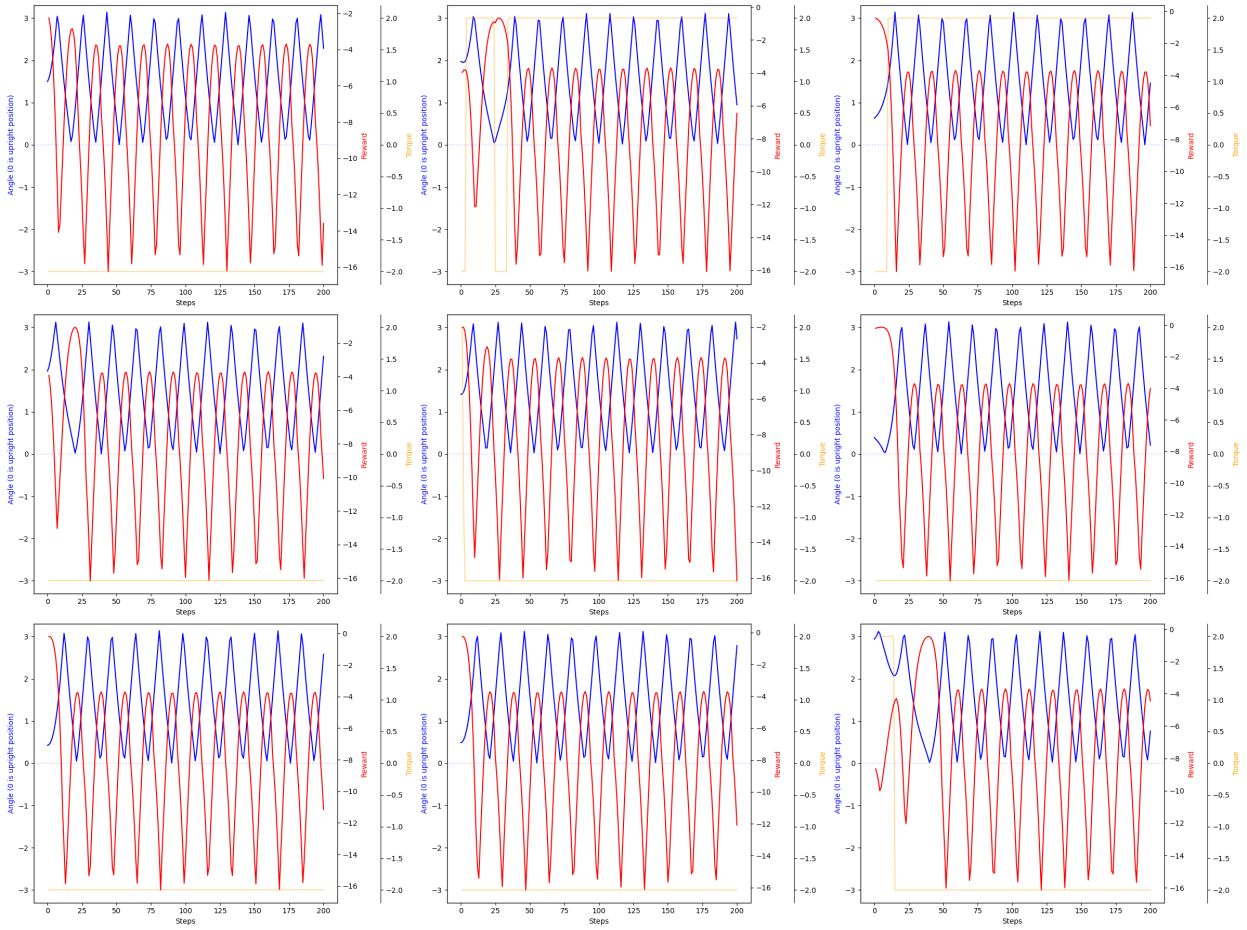


Figure 32: NAF-MBA Pendulum Final Evaluations

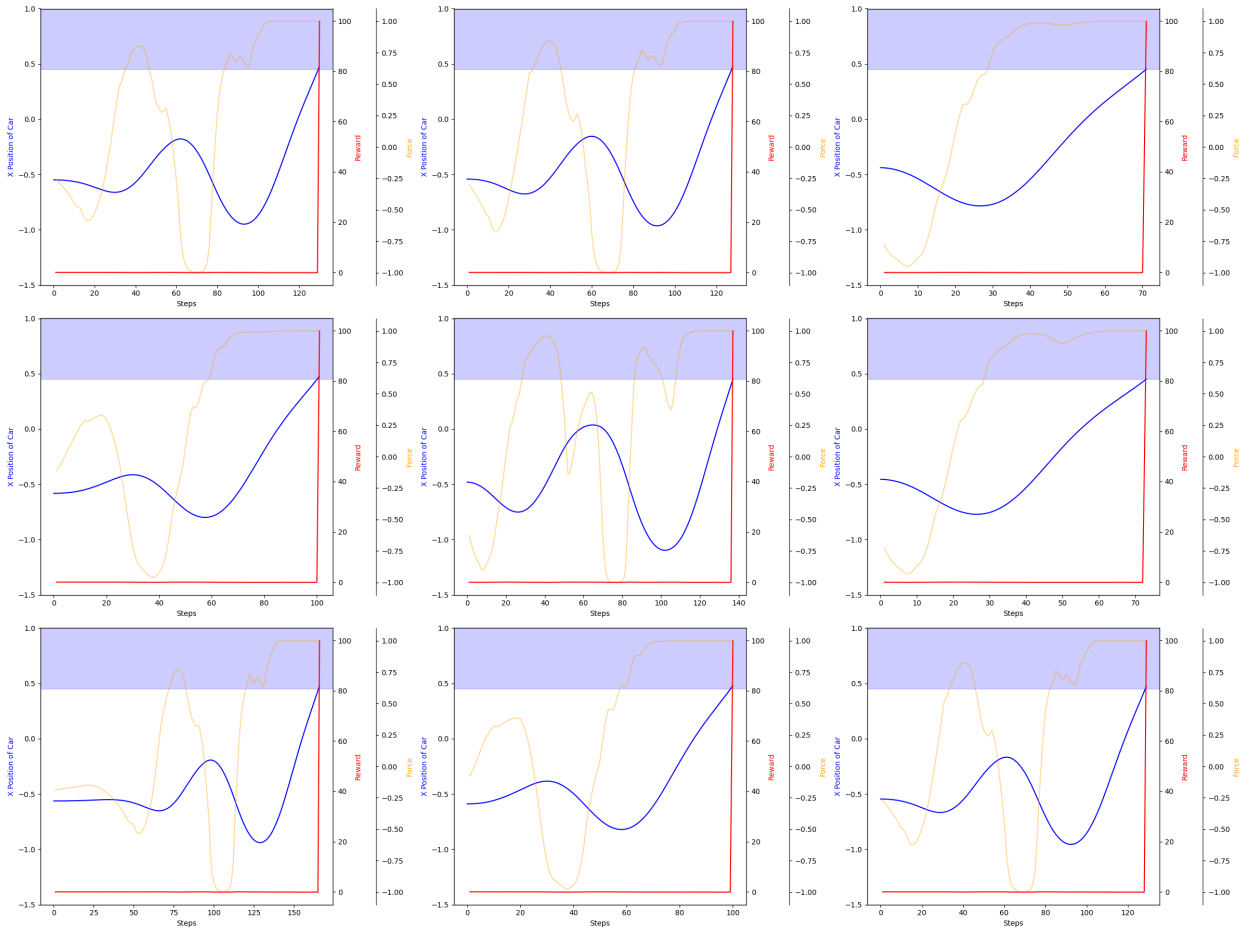


Figure 33: NAF-MBA Mountain Car Continuous Final Evaluations

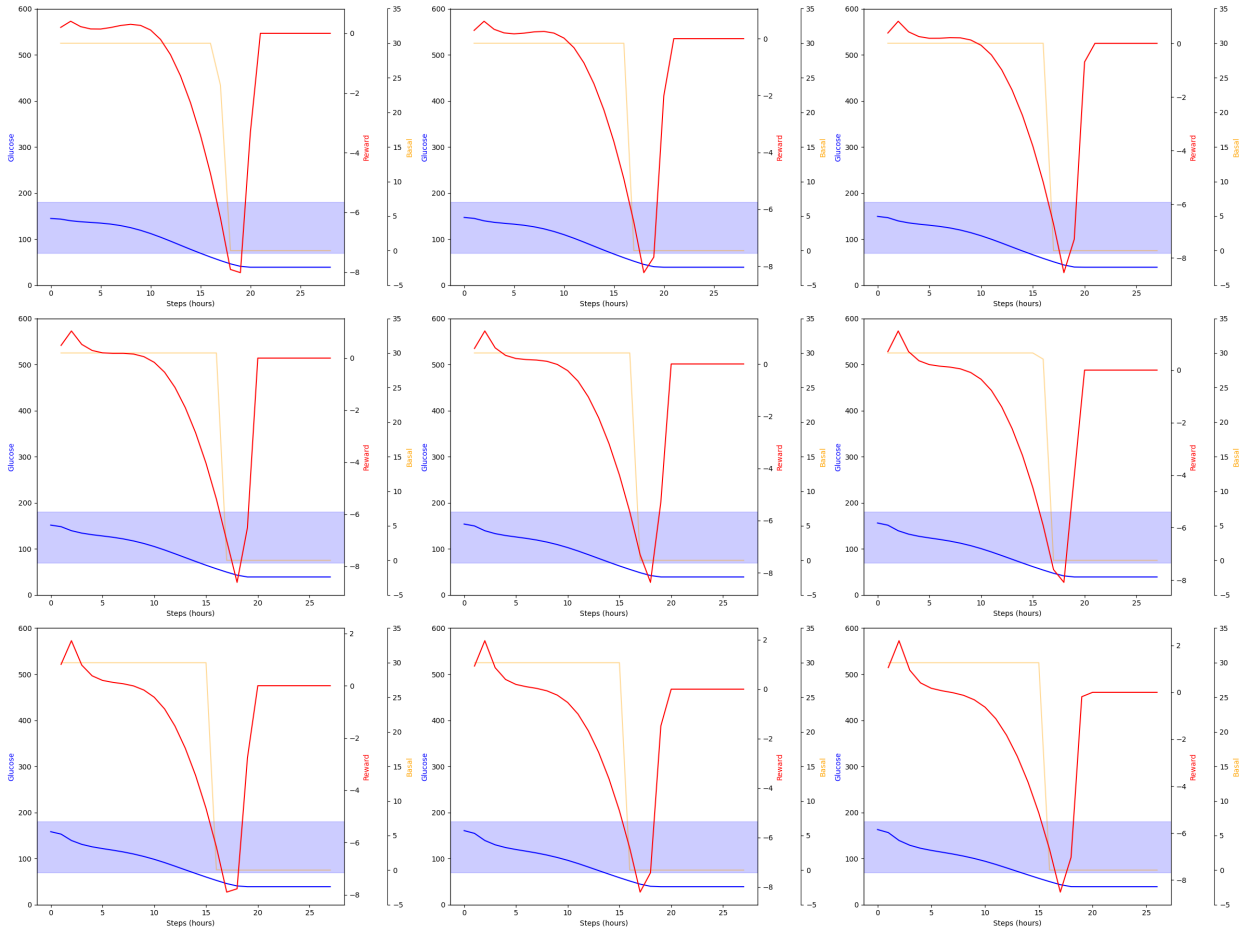


Figure 34: NAF-MBA Simglucose (Default Reward) Final Evaluations

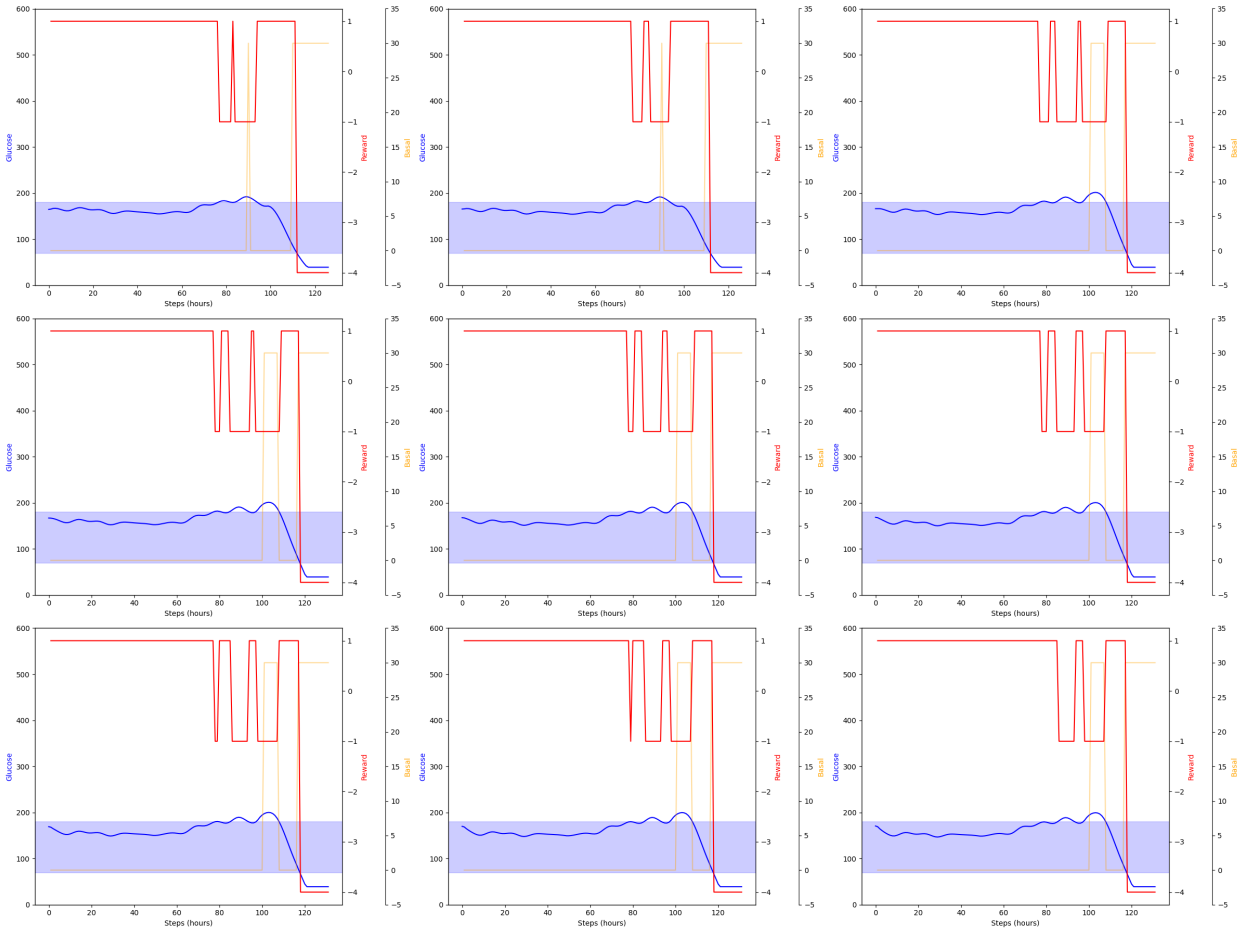


Figure 35: NAF-MBA Simglucose (Simple Reward) Final Evaluations

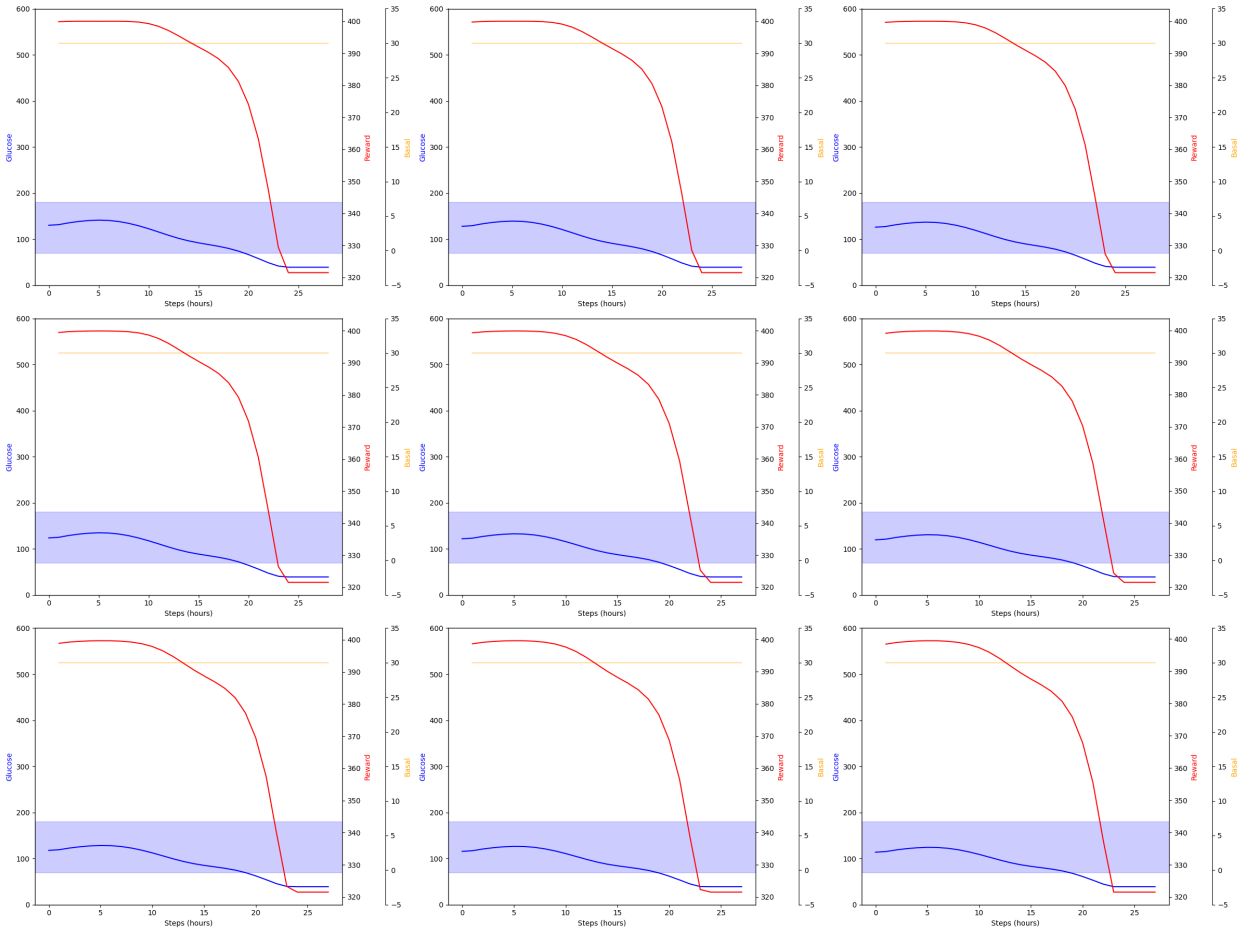


Figure 36: NAF-MBA Simglucose (Magni Reward) Final Evaluations

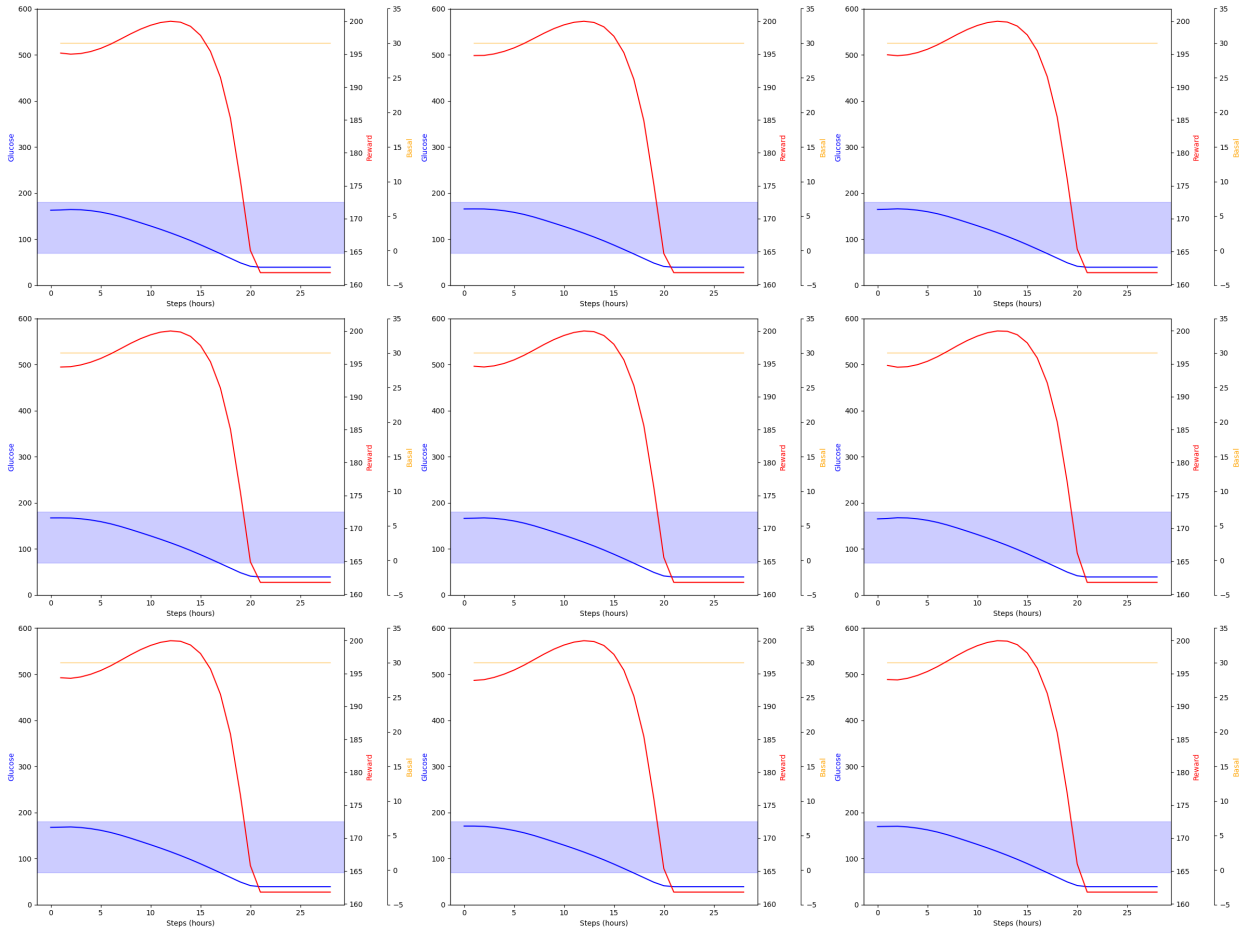


Figure 37: NAF-MBA Simglucose (Kovatchev Reward) Final Evaluations