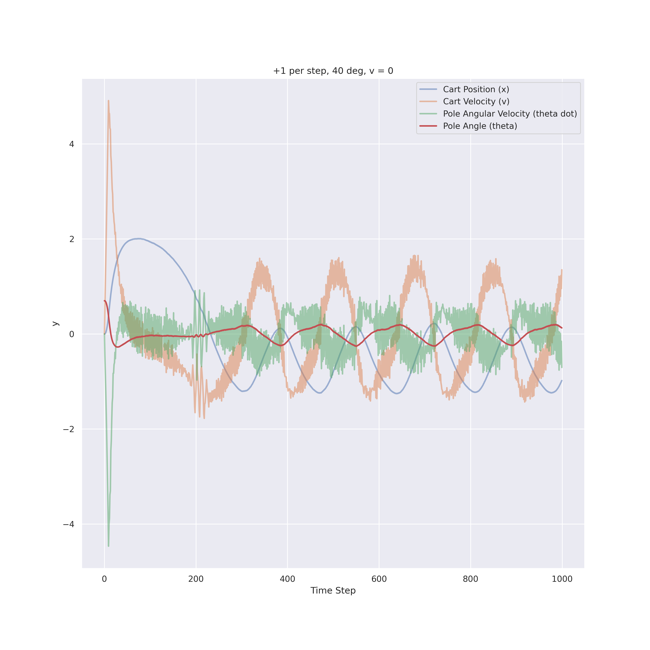
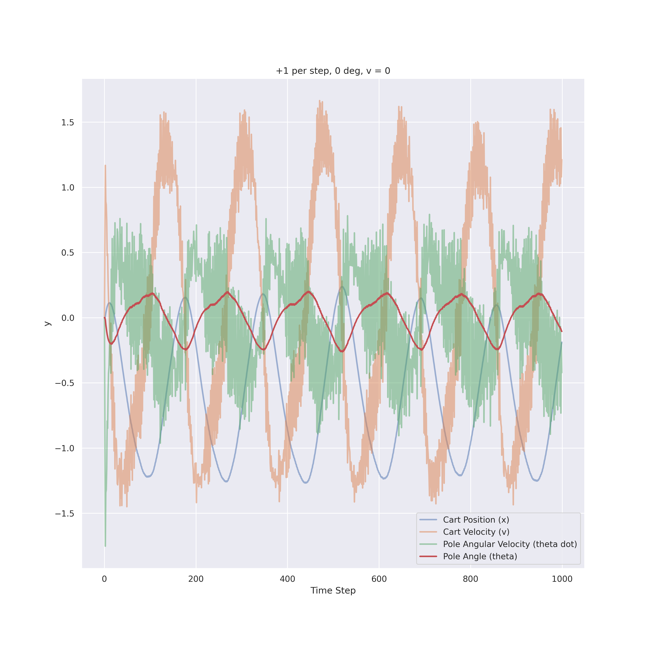
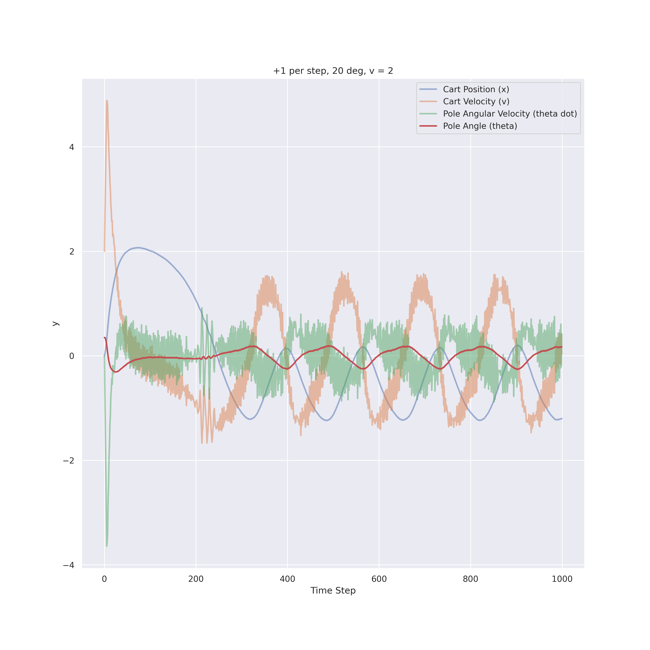
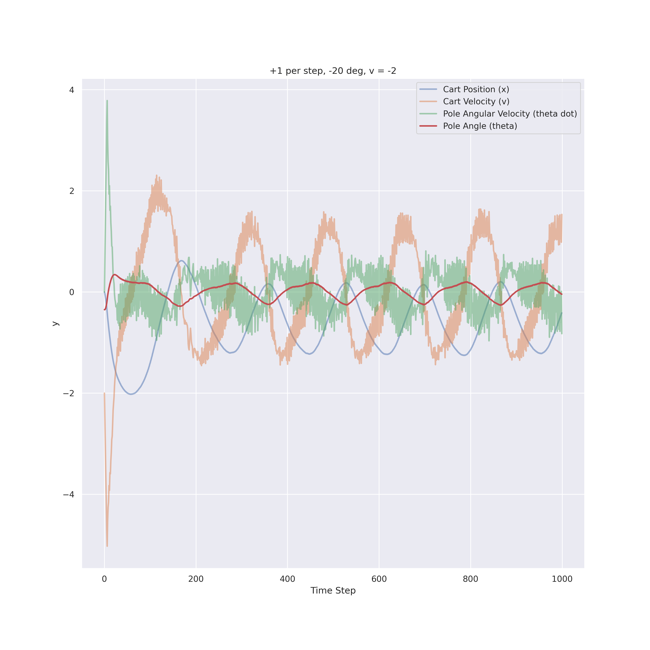
List of key rewards tested (in increasing order of complexity). Many other rewards were also tested for hyperparameter optimization, which will be introduced in the text. Note that these are rewards per time step, and episodic rewards are obtained by summing these up for the duration of the episode:

1. +1/step   
   [Basic Reward]
2. +1/step – energy   
   [Reward – Energy Penalty]
3. – energy   
   [Energy Penalty]
4. +1/step +   
   [Reward + CBF 1]
5. [CBF 1]
6. +1/step   
   [Relu CBF 1]
7. +1/step   
   [Relu CBF 2]
8. +1/step   
   [Weighted Relu CBF 2]
9. +1/step   
   [Weighted Relu CBF 3]

Building in order of complexity:

Basic reward:

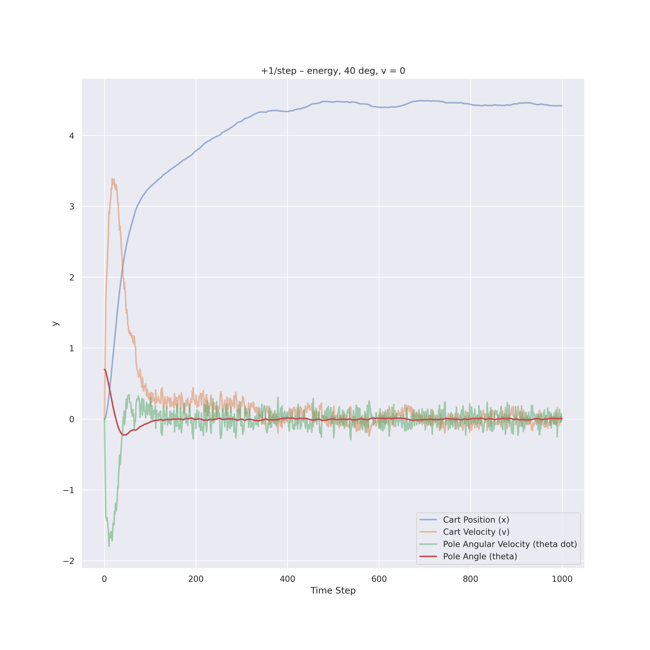
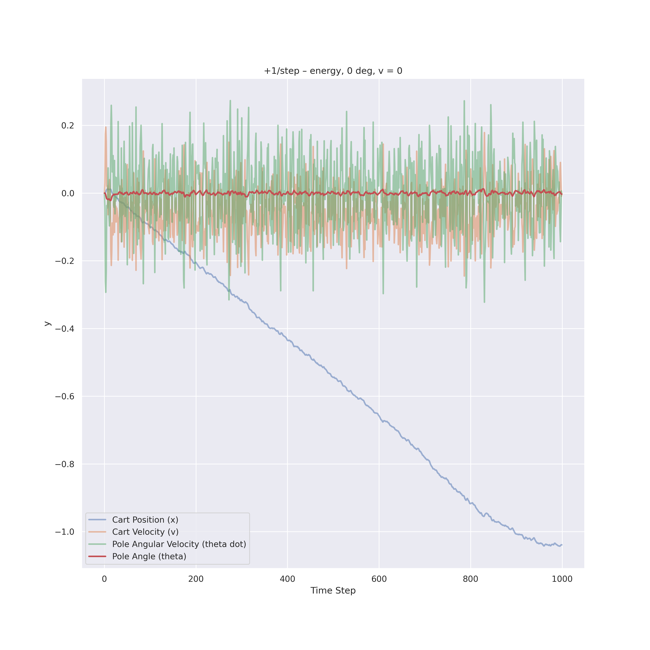


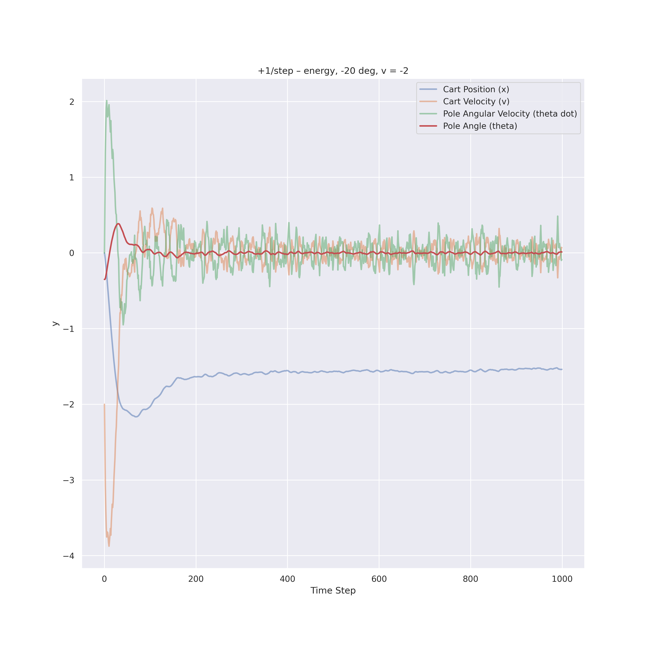


As shown in the plots for various initial conditions, this agent learns a policy that encodes oscillatory behavior as shown because there is no incentive to stabilize, and it is easier to learn a suboptimal policy. The major drawback is the amount of energy required to stabilize, which is the second worst among the reward formulations analyzed. This is mitigated by introducing various CBFs later in the text.

Reward – Energy Penalty:

This reward function is introduced to find the most optimal behavior in terms of energy required to stabilize by directly applying an energy penalty to the reward. Note that this is not considered a true CBF because it uses the action (force) in the reward formulation and is just included in the analysis for comparison purposes. Very stable and fine control is observed, as shown in the plots, and the non-essential oscillations are removed.

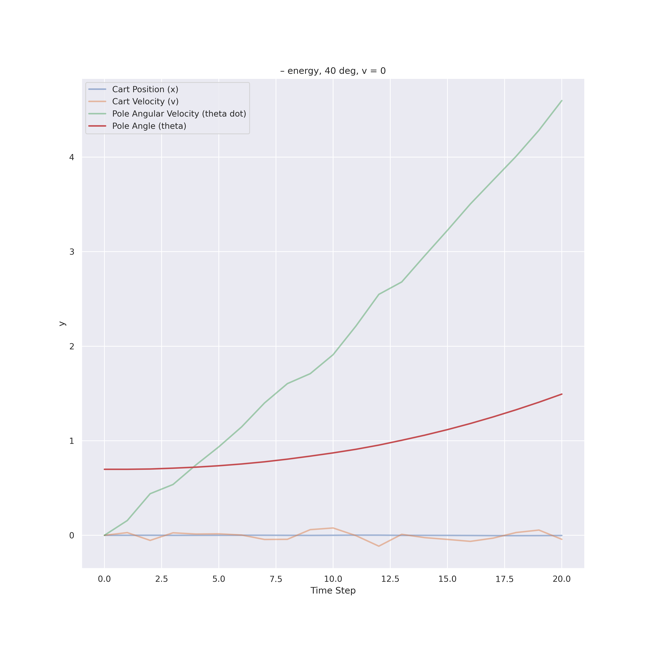
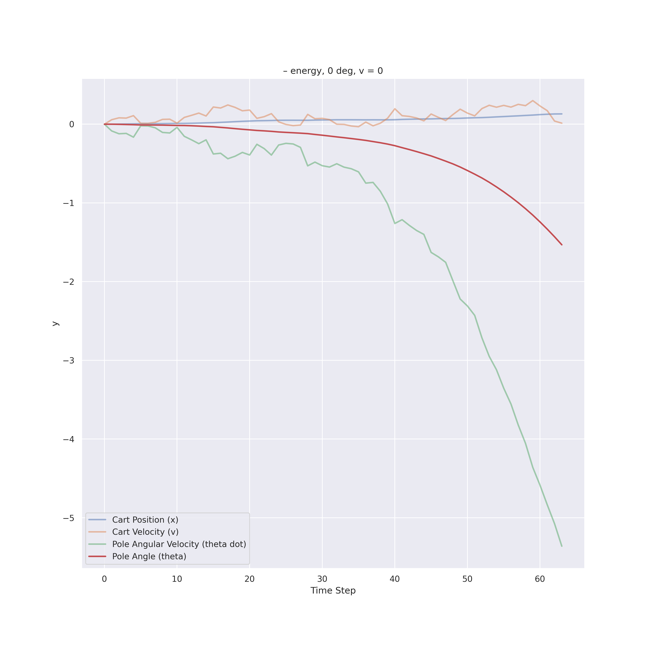


Chart, line chart

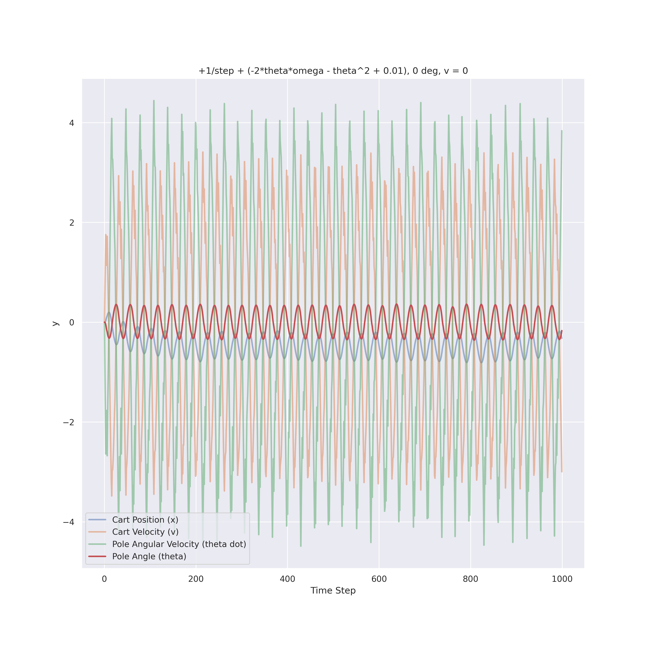
Description automatically generated

Energy penalty:

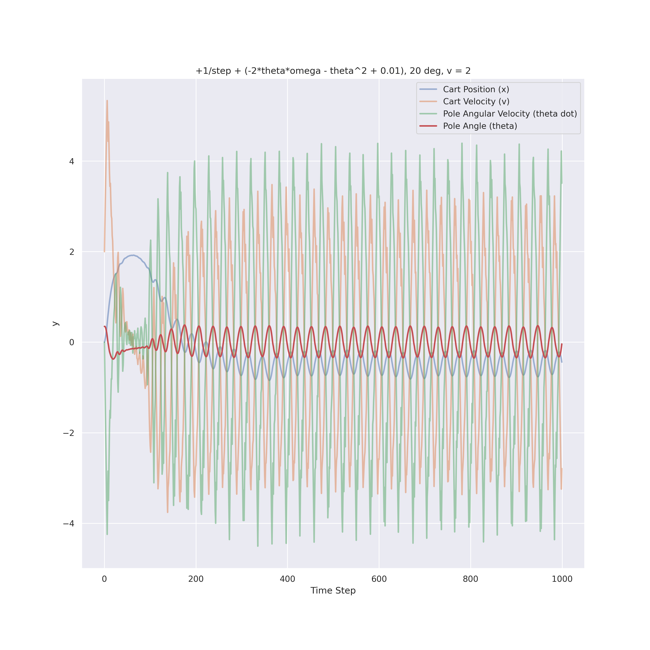
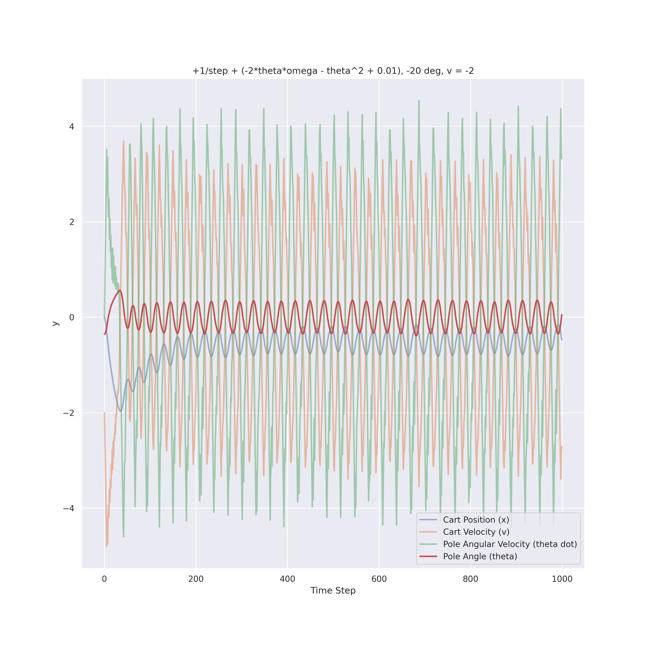
As we can see, this agent fails to stabilize for the simplest initial states. This is because the reward is always negative, as the energy implementation used is defined to be always positive. Hence, if the agent stays up for a larger number of time steps, it will accumulate more negative rewards, and this behavior will be discouraged. The agent learns to terminate the episodes in very few time steps to maximize reward. To avoid this behavior, we must include +1/step as in the earlier case to condition the agent to stay up for a long as it can accumulate more net positive rewards with time.



Reward + CBF 1:

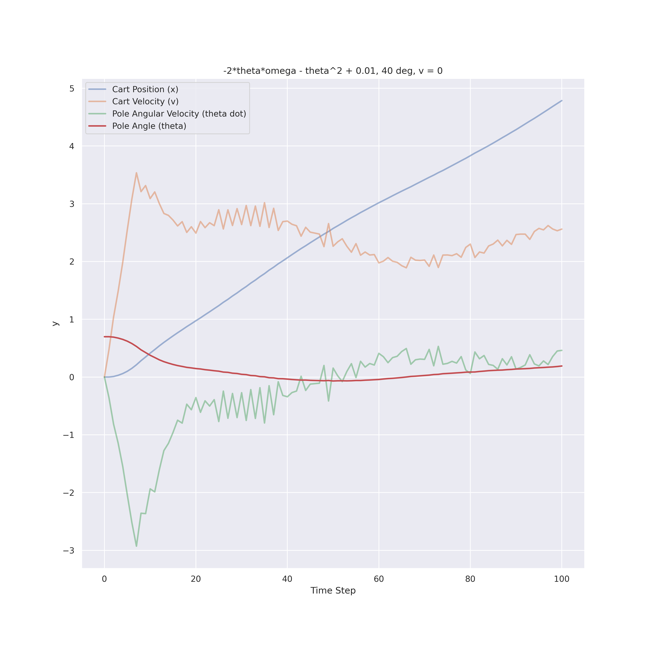
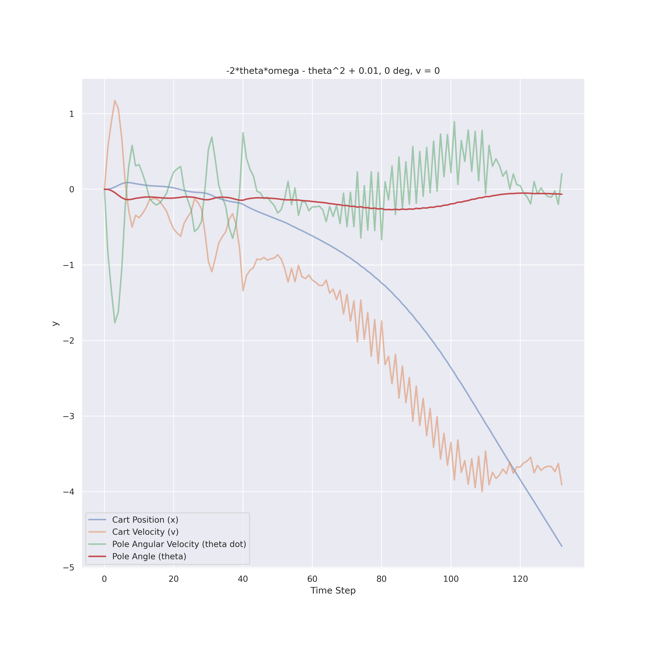
Chart

Description automatically generated



As inferred from the plots, this reward formulation consumes the most energy due to its oscillatory control style. We can argue this behavior arises because the CBF 1 “penalty” is not strictly positive, and the agent hyper-optimizes for the 2\*omega\*theta term (which can be positive and offset the penalty, leading to reward values of more than 1000). It does so by the distinct oscillation style observed in the safe region, but this behavior is the worst in terms of energy required to control. To mitigate this, the Relu function is introduced later in the text, which only penalizes and does not encourage oscillatory behavior.

CBF 1:

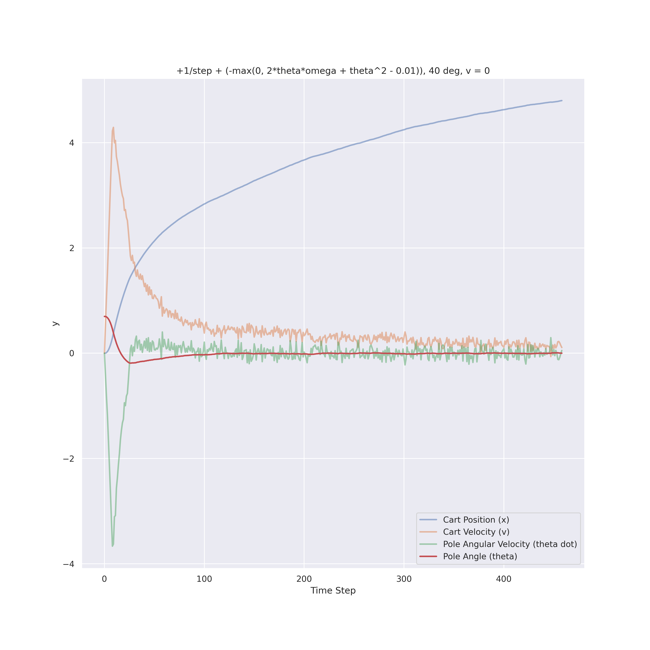


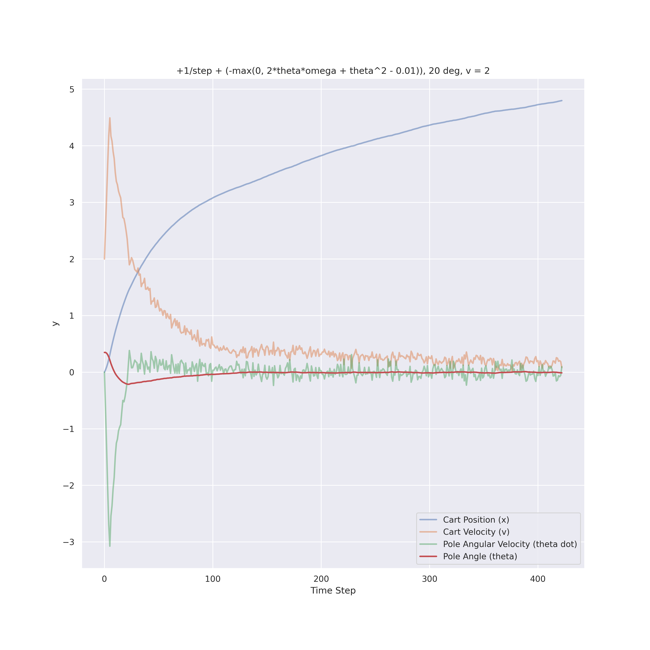
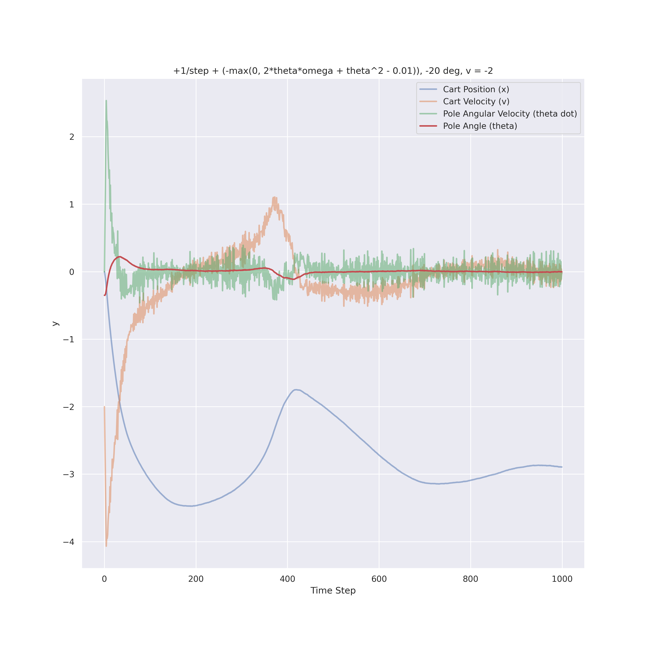
As we can see, this agent fails to stabilize for the simplest initial states. This is because the reward is mostly negative. The CBF 1 is a penalty for undesired behavior and is mostly negative (not strictly, it may be positive in case of exceptionally good control flows). Hence, during training, the agent initially encounters only negative rewards, and it observes that these negative rewards accumulate if they stay up longer. Hence this behavior is discouraged, and the agent learns to fall as quickly as possible. This is mitigated by introducing a +1/step reward so that staying up for longer is rewarded.

Relu CBF 1:

With the introduction of the Relu function, the subsequent oscillatory behavior is mitigated. However, the control flow is still not very fine and smooth, and the energy required is suboptimal compared to the Reward – Energy Penalty formulation (benchmark). A large variation in other parameters can be observed, especially the first-order derivative quantities (omega and v). To mitigate this and improve the control flow further, we also introduce CBFs on these quantities.

Chart

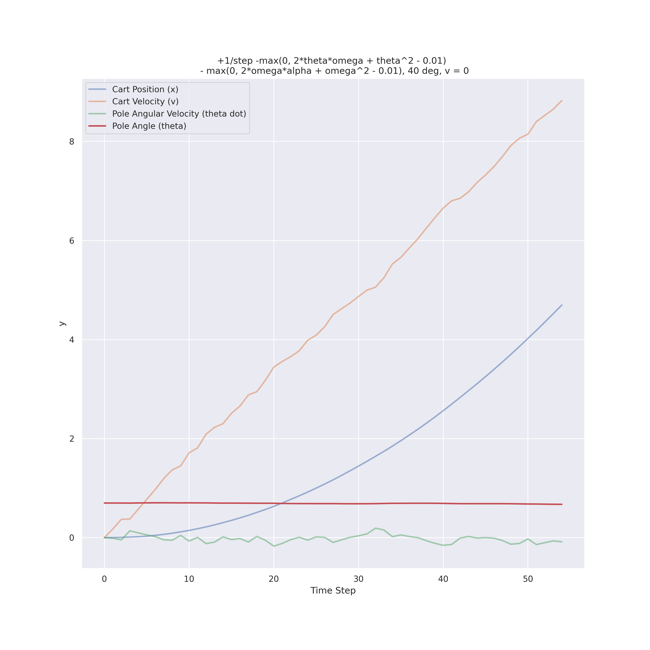
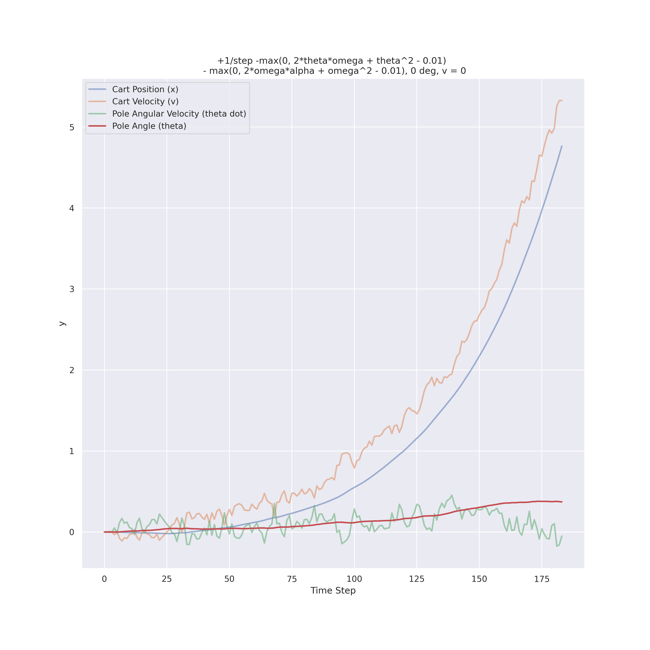
Description automatically generated



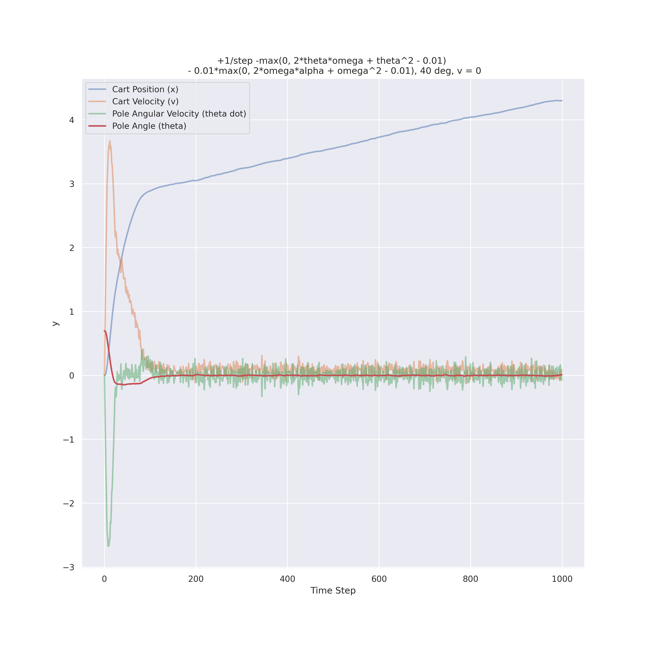
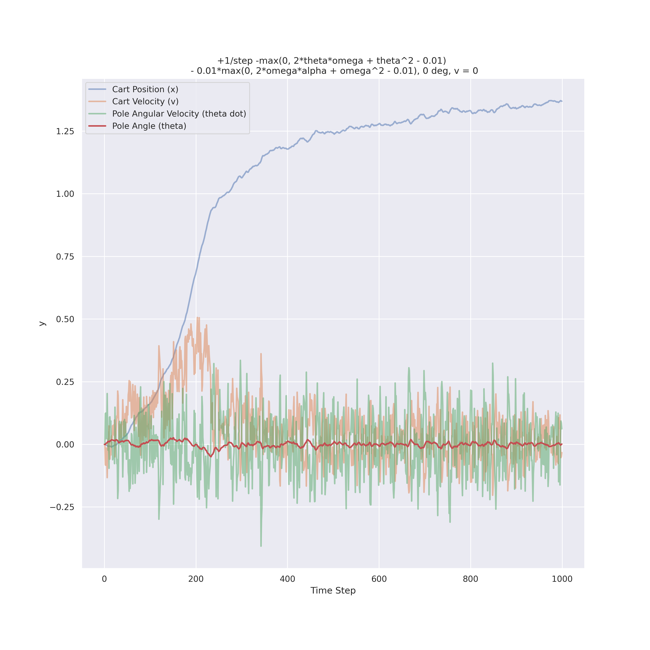
Relu CBF 2:

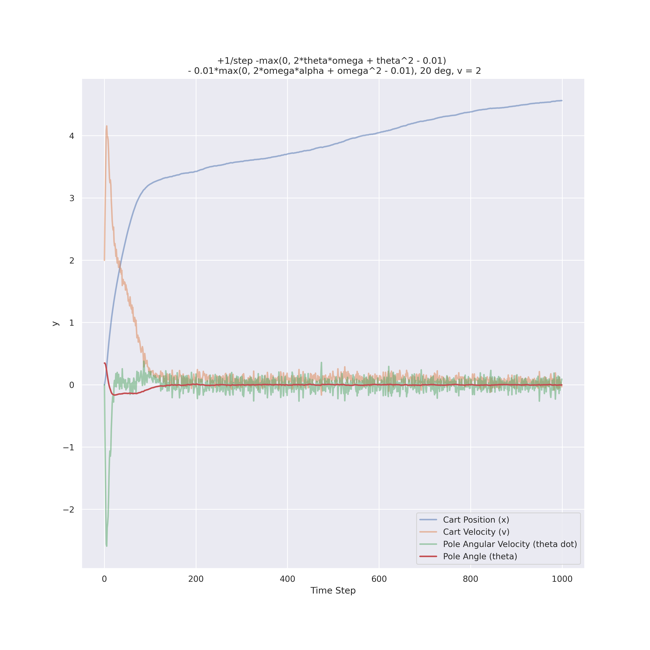
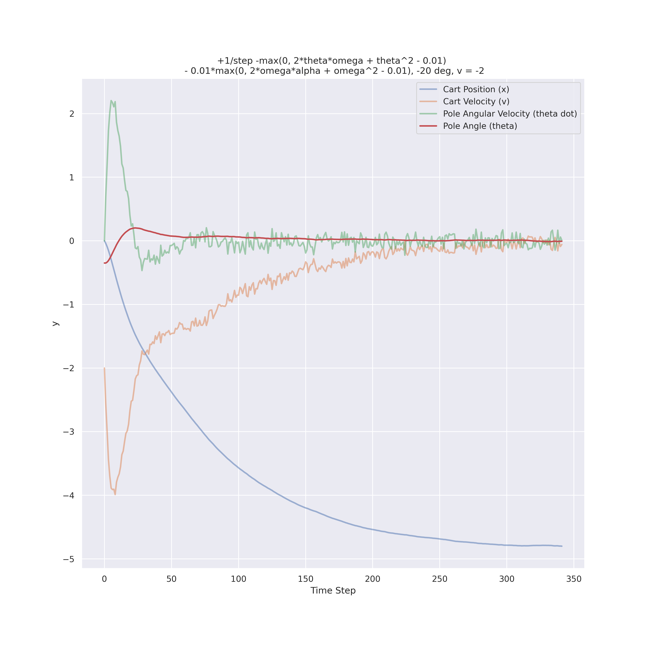
This reward is considered to ensure the control is even finer and less energy is required by rejecting actions promoting sudden changes in the first-order differentials. We see that the agent trained on this reward cannot control even the most basic states. This is simply because the alpha penalty (second term of CBF) is too large for most cases, and the agent accumulates a negative reward as earlier if it stays up too long. Hence it learns to terminate quickly. This behavior is mitigated by introducing a weight for the second CBF term.

Note: A CBF parameter for omega is added here. Note that we encounter pole angular acceleration (denoted alpha) in its derivative. This can be determined by using mechanics and the action values, but this would mean we are considering a function of action as the CBF, which is not desired. Hence, we build a first-order secant approximation using the observables as a proxy for alpha.



Weighted Relu CBF 2:





A weight for the CBF of omega (0.01) is added to this reward to ensure that the penalties aren’t too large and there is enough net positive reward to condition the agent to stay up longer. However, we are not constraining motion along the x-direction here. As a result, the x-coordinate exhibits monotonic increasing/decreasing behavior and crosses the thresholds in some cases, due to which this agent fails to stabilize many initial states. However, it stabilizes the ones it can in a very smooth, controlled and energy-efficient way. As a final measure, we add a CBF constraint on x-velocity (referred to as v hereafter) to mitigate this and ensure it controls some more tricky initial states.

We also note that this CBF will perform better and might be able to learn not to cross the x-threshold if allowed to train for longer (a greater number of episodes). Training is difficult on this CBF because the reward is very complex, and learning to optimize for it by mere trial and error is hard. It also does not explicitly optimize for keeping x and v low.

Weighted Relu CBF 3:

Chart

Description automatically generatedGraphical user interface, chart

Description automatically generated

Chart, line chart

Description automatically generatedGraphical user interface, chart, line chart

Description automatically generated

This is the most optimal (in terms of energy), reliable (in terms of % of initial states stabilized) and smooth (in terms of control flow) reward formulation among the ones investigated. As we shall see soon in the energy heatmaps, the amount of energy it takes to stabilize the initial states is like the Reward – Energy Penalty case (benchmark). Both the CBFs for the first-order derivatives (omega and v) are weighted as discussed earlier with weights of 0.01. The exact value of weights was determined by a series of trial and error experiments.

Note: A CBF parameter for v is added here. Note that we encounter cart acceleration (acc) in its derivative. This can be determined by using mechanics and the action values, but this would mean we are considering a function of action as the CBF, which is not desired. Hence, we build a first-order secant approximation using the observables as a proxy for acc.

Stabilization heat maps

For v=0:



For v=-2:



For v=2:



Inference from the stabilization energy heatmaps:

* Basic Reward (1) and Reward + CBF 1 (4) consume a large amount of energy. This is because of the oscillatory behavior seen in them that arises due to different reasons. In Basic Reward (1), there is no incentive to stabilize at theta = 0 and the agent is just asked to stay up for longer. As a result, it enters an oscillatory cycle which is easier to learn. In Reward + CBF 1 (4), because the CBF reward is not strictly negative, the agent exploits the -2\*theta\*omega term to get positive rewards by continuously oscillating.
* Reward – Energy Penalty (2) is the gold standard for minimal energy and optimal control because it is directly trained to minimize energy penalty. It is not a CBF under consideration because it has action (force) in the reward formulation (in the energy equation).
* Weighted Relu CBF 2 and 3 are the closest to reward–energy penalty (2) in terms of energy required to control, which indicates their superior performance. Note that Weighted Relu CBF 3 (9) stabilizes for most angles with the most optimum energy; hence, it is the best model among the ones studied in this work.
* A lot of reward formulations are missing from this table. Namely: Energy Penalty (3), CBF 1 (5), Relu CBF 2 (7). This is because they could not stabilize for 1000 timesteps for 57 initial conditions tested. This behavior can be attributed to these rewards being mostly negative. If the agent stays up for a larger number of time steps, it will accumulate more negative rewards, and this behavior is discouraged by these reward formulations. Measures and improvements are noted later to mitigate this to ensure that a positive reward is given for desirable behavior so that the agent learns to stay up for longer.
* All the initial theta conditions were repeated for three values of cart velocity: v = 0, -2 and 2. Largely symmetric behavior is observed in the case of v = 0, as can be inferred from the above heatmaps. It becomes difficult to stabilize the larger positive angles when v = 2 because applying force to the right will cause the cart to cross the x threshold and applying force to the left will cause the pole to fall. Hence asymmetry is introduced when the cart is given a non-zero starting velocity. Similarly, it becomes easier to stabilize negative angles in the case of v = 2. An exact opposite argument can be made to explain the v = -2 results.