Research Proposal

Rewarding of Strictly Completely Monotone (SCM) Trajectories

Nhu Huy Lê-Cao

October 01, 2025

1 Preliminaries

Before heading into the main theoretical aspects of the thesis, we first need to agree on common foundations to reach a mutual understanding. This includes terms and theorems on which the new hypotheses and propositions are grounded. To the accustomed reader, this section can be skipped.

1.1 Definitions

We denote a list of terms that are frequently used in this work. Most of them we borrow from control theory, graph theory and of course reinforcement learning, but a subset is necessarily defined this way for the first time.

Definition 1.1. Euclidean Space, Euclidean Distance, (Radial) Distance Space

Definition 1.2. State Space, Action Space, Transition, Path, Trajectory

Definition 1.3. Goal State, Goal-Distance, Goal-Direction

Definition 1.4. Dynamical System

Definition 1.5. Control, Policy, Optimality, Convergence, Decay, Oscillation

Definition 1.6. Stability, Global Stability, Asymptotic Stability, Unstability

Definition 1.7. Fix Point, Equilibrium, Attraction, Monotonicity, Ljapunov-Functional

Definition 1.8. Strictly Completely Monotonic (SCM) Trajectory, Strictly Absolutely Monotonic (SAM) Trajectory, Violation

Definition 1.9. Reachability, Goal-Convergence, Local Minima

Definition 1.10. Markov-Decision-Process (MDP), Reward

Definition 1.11. Reinforcement Learning (RL), RL Algorithm

Definition 1.12. Policy, Greedy Exploitation, Non-Greedy Exploration

Definition 1.13. Sample Efficiency, Generality

Definition 1.14. Goal-Directed Rewarding

1.2 Theorems

We denote a list of already established theorems from which we believe our results can be derived. Available proofs to these theorems are referenced at the end of this work, but they can also be found in the usual literature. We further do not claim the list to be exhaustive.

Theorem 1.1. Exponential Decay Limit

Theorem 1.2. Bernstein's Theorem on Monotone Functions

Theorem 1.3. Monotone Convergence Theorem

Theorem 1.4. Bellman Optimality

2 Introduction

Reinforcement Learning (RL) has been highly interesting to the control community for a long time, but struggles still to find widespread application where effectiveness and efficiency are **equally important**. RL has shown that it can produce trained behaviour to reach specific goals, but it often does so only after painfully large amounts of training cycles. This problem is one of the major stepstones for RL to find practical acceptance.

RL is sample-based and closed-loop optimization and therefore estimates the interactions in the world step-by-step with experience over time. The single most important condition for this is **meaningful feedback**. Implementations of this feedback in the form of *cost* or *reward* has been separated into two categories: Sparse feedback and dense feedback.

Sparse feedback occurs when a feedback signal is received close to reaching a goal. That means that the feedback is truly meaningful as it is sparse. No signal can be misinterpreted or even abused before the goal is even reached. But in consequence, feedback can be absent for a long time, possibly too long for the training to converge meaningfully and the goal is never reached in limited time.

Dense feedback on the other hand is received at all time steps. In dense feedback, a signal leads the training metaphorically *by the hand*. Every possible interaction in the world is answered with some cost or reward to the situation, but with the risk of over-abundant feedback. This can, and often does, lead to **abuse** of the feedback signal, as the training gets stuck in a local optimum to avoid or accumulate feedback. In that case,

the goal is also never reached without some extra sophistication to escape that dead-end.

A solution must lay somewhere in-between.

Experiments and modern computation have hinted that sparse feedback always leads to the goal if it is reachable and there is no time limit. In fact, theoretical work on Bellman optimality has proved this. What if we can substitute the objective of reaching (the proximity of) the final goal by a different, equally meaningful objective that is easier to achieve? A different objective that, if sparsely rewarded, is experienced frequently enough to reach the goal indirectly and in less training time? It might be enough for the overall task to reach an automatic *walkway* that, under circumstances, is guaranteed to lead to the goal.

One naive but often futile attempt is to reward the simple *approach* of the goal, namely any slope towards the goal in the form of the negative first derivative of the scalar distance, for example the Euclidean distance. That is, any action that decreases the distance *as the crow flies* is answered with a nice positive reward and the underlying *policy* is changed accordingly to eventually repeat the same action in the same situation or *state*. Given a situation where the agent is now not able to further decrease the distance, either at the goal, or at a local minimum other than the goal, the agent will eventually choose to retreat and increase the distance, but will then find itself in a previous situation where its experience recommends to simply approach the goal again. That is all well if the agent is already in goal proximity, but is disastrous at a local minimum. It is then stuck in an approach-retreat-cycle for the rest of the training. One major possibility is now to randomly *explore* actions that are not rec-

ommended by the current policy, that is, not greedingly taking the action with the highest probability under the policy. That mechanism has two downsides: First, explorative actions are by implementation much less probable, even less with later training progression. Second, to escape local minima with a large basin, explorative actions in *succession* are even less probable by the chain rule. In consequence, most of the training is spent stuck at local minima beside the occasional exploration venture. That is one if not the most important reason to sample-inefficiency in dense feedback.

Suppose we add to the rewarding of the naive simple approach the condition that *every* action in the past needs to be approaching the goal. That means that the only situation where a positive reward is received is when at all timesteps until now, the first derivative was negative, or in other words: There was never a sign flip from negative to positive. But since this additional condition would require information about the whole past to be stored, it is not memory-feasible as-is. The compromise is to limit the observed past to only a fixed length k and to hope for the rewarded actions to still reach the intended goal. That also makes escaping any local minimum possible *without* exploration, if it is doable in under k steps. Any escape route of length greater or equal k would be met with the choice of returning to the local minimum and collecting a cyclic reward, thus becoming less probable. But inside the k-lengthed basin, rewarding becomes sparse and actions are uniformly distributed at start. They represent a smaller sub-environment where escaping a local minimum and reaching a trajectory, a *walkway*, towards the goal is rewarded. Given no (other) local minimum is encountered afterwards, the goal will eventually be reached

since the agent is able to always take the approaching action with negative first derivative of the distance to the goal. Since the distance is non-negative, we have a Ljapunov argument.

We could add another condition to the rewarded trajectory and enforce not only approaching the goal, but also *slowing down* towards the goal, effectively seeking **convergence**. This is possible through the derivatives of higher order. To make sure that the agent slows down in the direction of the goal, the first derivative must not only be negative, but also strictly monotonically increasing until it hits zero at the goal. This change of the *first* derivative is formally represented by the *second* derivative and must also be strictly monotonic - in this case positive and strictly decreasing again. This change of the second derivative is enforced by a negative and strictly increasing *third* derivative, and so on in all higher orders. Any violation of that strict complete monotonicity (SCM) would eventually cascade into oscillation in the trajectory, which would stipulate correcting actions by the agent. Thus, an SCM trajectory can be desirable for its guarantee of non-oscillating convergence.

Furthermore, in discrete-time environments where the typical RL agent resides, derivatives up to order k are approximated by finite differences of the previous k timesteps. Since an SCM trajectory is also an approaching trajectory at all times, rewarding SCM trajectories limited to finite order k will eventually escape all local minima with basins of attraction up to size k and does so **without** exploration.

In summary, we hypothesize:

Hypothesis 2.1 (Fundamental Hypothesis)**.** In Reinforcement Learning, it can be more sample-efficient to reward strictly completely monotone trajectories towards the goal to achieve the goal.

3 The Original Experiment

Out of frustration for the shortcomings in Deep Reinforcement Learning, especially with limited computational resources, I first began experimenting with different reward functions. A lot of work in the community is mostly focused on changes in the algorithms and how the data(-buffers) are processed and fed to neural networks with custom architecture. Implementing the reward functions is regarded as likely tedious tweaking in the similar category as trying out hyperparameters is. To be precise, the programmer searches for viable choices in the wide space of functionals, or decides on sparse rewards and potentially replay buffers. This happens usually after the major design of the feedback loop has been already settled. Hence, for the reward function, the choice is typically between the one agnostic function for sparse rewarding and a possibly highly specialized function for dense rewarding. An agnostic dense function, however, is met more often that not with the learning agent getting stuck at local minima.

To my surprise, after experimenting with dense reward functions relative to some state history, I observed successful learning in multiple different three-dimensional simulations following the same reward function

Definition 3.1 (k-SCM Reward Function)**.**

$$R_{SCM}(d_t) = \begin{cases} 1, & \text{if } (-1)^i \nabla^i d_t > 0 \\ 0, & \text{otherwise} \end{cases} \qquad \text{for all } i = 1, \ldots, k,$$

where the binary reward depended solely on the finite differences

$$\nabla^k d_t = \sum_{i=0}^{k} (-1)^i \binom{k}{i} d_{t-i}$$

of the scalar Euclidean distance

$$d_t = \sqrt{(x_t - x_g)^2 + (y_t - y_g)^2 + (z_t - z_g)^2} \in \mathbb{R}$$

to the origin and goal state

$$s_{origin} = (x_g, y_g, z_g)$$

at time t and of order k (which can be seen as the amount of past achieved states to be regarded for the reward).

The different experiments consisted of reaching, holding or manipulation of states. For example, in a simulated episodic environment, a robotic hand with 24 joints was tasked to reach and hold varying goal coordinates with its fingertips under the new reward function R_{SCM}. The actor-critic-based agent achieved all varying goal states accurately in less total episodes than a baseline with sparse rewarding. In fact, in every different environment, the training reached a predefined conclusion with significantly less amount of samples needed. The details and results of the original experiments will be added to the appendix.

4 Under the Hood

To understand what is truly happening when an SCM trajectory in RL is rewarded and when it is more sample-efficient, we observe the optimal trajectory of the comparable continuous optimization problem

$$max_u \int_0^\infty \tilde{R}_{SCM}(d_t)\, dt, \tag{1}$$

and its optimal input solution

$$\tilde{\pi}^* = argmax_u \int_0^\infty \tilde{R}_{SCM}(d_t)\, dt,$$

where the distance $d_t \in C^\infty$ is smooth over time t and order k is at least 1. Additionally, the slightly different reward function $\tilde{R}_{SCM} \approx R_{SCM}$ has the actual higher-order derivatives

$$d_t^{(k)} = \frac{\mathrm{d}^{(k)} d_t}{\mathrm{d}t^{(k)}} \approx \nabla^k d_t$$

rather than their approximating finite differences.

The actual task that is *supposedly* represented by above *proxy* problem (and that is also not restricted to RL solutions) is to bring a dynamical system

$$\dot{d}_t = f(d_t, u_t)$$

to its reachable origin $d_t = 0$, although the reward itself cares only about the distance d_t and not the control input u_t. However, we can only propose:

Proposition 4.1 (Convergence towards a Minimum)**.** The optimal input policy $\tilde{\pi}^*$ that maximizes the accumulated reward from the continuous optimization problem (1) will eventually stabilize its dynamical system to a (local) minimum.

That means that the system is **not** necessarily brought to its origin, except if the origin is the only minimum. In other words, although the system is asymptotically stable, the system is *globally* asymptotically stable (GAS) at the origin if there is no other equilibrium.

Hence, for the next reasoning step, we observe the discrete variant to above optimization problem

$$max_u \sum_{t=0}^{\infty} R_{SCM}(d_t) \tag{2}$$

and its optimal input solution π^* with the original reward function R_{SCM} from definition 3.1. Here, an optimally solved controller to that problem finds finite *descending* segments where they can smoothly decrease the distance with as many consecutive k-SCM steps as possible:

Definition 4.1 (k-SCM Step)**.** A discrete state or step is k-SCM, if it yields a positive reward by the k-SCM reward function R_{SCM}.

(Note that a single k-SCM step needs at least k preceding steps, but the latter are not necessarily k-SCM.)

Similarly, we can define a segment, where all its steps are k-SCM:

Definition 4.2 (k-SCM Segment)**.** A segment of a trajectory is k-SCM, if all its steps are k-SCM.

Any interruption to that k-SCM segment, for example at a local minimum, is met with a traversal to the next closest possible k-SCM segment. If there are multiple in the same proximity, the **longer** one is taken, because it yields a greater expected return of uninterrupted rewards. If they all have the same length (and this can include already found and traversed segments), we are stuck again in a rewarding cycle. But we have found a segment that is closer to or even includes the origin, while inherently *escaping* all local minima when it was worth to do so. We propose:

Proposition 4.2 (Descent towards an isolated k-SCM Segment)**.** The optimal input policy π^* that maximizes the accumulated reward from the discrete optimization problem (2) will eventually stabilize its dynamical system to a limit cycle of an isolated k-SCM segment.

Hence, it is important to understand when a k-SCM segment is isolated:

Definition 4.3 (Isolated k-SCM Segment)**.** A k-SCM segment is *isolated*, if every other k-SCM segment of length l is more than $(2k + l)$ steps away.

Isolation seems to decide when it is worth to escape a segment for a better option and when it is rather worth to infinitely repeat a segment by its basin of attraction:

Proposition 4.3 (k-SCM Segments with a Basin of Attraction)**.** Under the optimal input policy π^*, a (local) minimum is attractive if and only if it is contained by an isolated k-SCM segment.

Does that mean that we can increase our choice of order k for the reward function until the controller escapes every local minimum and eventually finds its way to the origin? Sadly no, because it is not guaranteed that the origin itself is contained by an isolated k-SCM segment with the single greatest order for a *unique* global basin of attraction.

As we finally apply RL to produce the optimal input policy π^* by methodic sampling (*trial-and-error*), the learning agent to the discrete optimization problem (2) already seeks the k-SCM segments during its *training*, where it has no model of the landscape (yet). Here, we also hope to traverse local minima inherently and especially without the need for exploration, that is, only depending on the greedy actions[1].

If the actions are initially uniformly distributed, every action to a state is greedy when that state is visited for the first time. In other words, the policy returns the same probability for every action before it is actually experienced and improved by an iterated value estimation. As a consequence, under the reward function R_{SCM}, its trajectories follow a uniform *Random Walk* until a non-zero reward is received and the policy is updated according to the (discounted) values of the states and decisions that led to the reward. But since rewards are sparse (but not

[1]Ties between actions are broken uniformly random, which does not fall under exploration in the RL sense.

as sparse as in typical Sparse Rewarding), trajectories spread out evenly enough to at least escape every local minimum with a basin of attraction up to size k (because no reward can be received inside), and at most until every trajectory leads to a (rewarded) k-SCM segment.

However, without a model, the agent initially does not know whether a k-SCM segment is isolated. That means that there might be a more profitable segment in proximity, but not in known *sight*, and the agent is not determined to reach it if a different segment has been found first. It is even worse when the found segment is an already visited segment and the trajectory gets practically *shorted*. In consequence, the agent presumes its isolation and will repeat the trajectory while recycling its rewards endlessly. Hence, we propose its behavior[2] as:

Proposition 4.4 (Greedy Sampling towards a occupied k-SCM Segment)**.** In greedy RL, the converging input policy π_θ that samples the accumulated reward from the discrete optimization problem (2) will eventually stabilize its dynamical system to a limit cycle of an occupied k-SCM segment as the number of samples N increases.

Definition 4.4 (Occupied k-SCM Segment)**.** A k-SCM segment is occupied, if it is visited at least twice during the same episode.

The converging input policy π_θ is the current policy improve-

[2]We can imagine a fit snowboarder who starts at the foggy mountain top and is searching for exciting slopes. What is their skill level k? What happens after they find a slope with a returning lift? And what changes if they have a map (model) of the area?

ment following the optimization of its parameters θ by dynamic programming[3]. As the number of samples N goes to infinity, the converging input policy π_θ should approach the optimal input policy π^* if *all* states are sampled infinitely often[4].

In RL without exploration, that is not certainly the case anymore. At the k-SCM segments, we are lacking model information about all states, and the sense of premature occupation takes over optimal isolation. Hence, the decisions made by the converging input policy π_θ during training will not necessarily bring the system to an optimum, that is, as close to the origin as profitable under the reward function R_{SCM}. However, in practice, we still get closer with less samples compared to other reward functions[5]. Therefore, the necessary training time for the agent to learn how to progressively approach a goal is shortened.

After all, this is what we aim to show in our experiments.

[3]See definition 1.11 of RL.

[4]See theorem 1.4 of Bellman Optimality by iterative sampling.

[5]We could also re-introduce exploration or even switch reward functions to cover for non-isolated occupation or the *last mile* to the goal.

5 Experiments

The following experiments are meant to show the propositions in a controlled, reproducible and quantifiable environment. Together, they should empirically support the fundamental hypothesis 2.1 of our new reward function R_{SCM} that is defined in 3.1.

Experiment 5.1 (**MPC of a k-SCM Double Integrator**)**.** This experiment shows propositions 4.1, 4.2 and 4.3 of an *optimal* input policy π^* under the new reward function R_{SCM}.

Experiment 5.2 (**RL of a k-SCM Double Integrator**)**.** This experiment shows proposition 4.4 of a *converging* input policy π_θ during RL training without exploration and local minima.

Experiment 5.3 (**RL of a 3-dim. k-SCM Robotic Manipulator**)**.** Similarly to experiment 5.2, this experiment shows proposition 4.4 during RL training without exploration, but with local minima in three dimensions. Additionally, there exist comparable SOTA benchmarks.