Numerical Geometry of Reinforcement Learning: Curvature of the Bellman Operator

Anonymous Authors[1]

Abstract

We apply the framework of *Numerical Geometry* to reinforcement learning, modeling the Bellman operator as a numerical morphism with explicit Lipschitz constant γ (the discount factor) and intrinsic roundoff error Δ_T. The Stability Composition Theorem provides exact error accumulation formulas: after k value iterations, the total numerical error is $\Phi_{T^k}(\varepsilon) = \gamma^k\varepsilon + \Delta_T \cdot (1-\gamma^k)/(1-\gamma)$. Our key theoretical contribution is identifying a **critical precision threshold** p^*: when precision falls below $p^* = \log_2((R_{\max} + |S| \cdot V_{\max})/((1-\gamma)\varepsilon))$, numerical noise dominates the contraction, causing the effective discount factor to exceed 1 and value iteration to diverge. We provide concrete precision requirements as functions of discount factor, reward scale, and state space size. Experiments on gridworlds, FrozenLake, and CartPole with tiny function approximators verify that (1) observed precision thresholds match theoretical predictions within 2-4 bits, (2) error accumulation follows predicted trajectories, (3) the precision requirement scales as $\log(1/(1-\gamma))$ as theorized, and (4) float16 training fails for $\gamma > 0.95$ while succeeding for $\gamma \leq 0.9$. All experiments run on a laptop in under 2 minutes, demonstrating practical deployability of our theoretical framework.

1. Introduction

Reinforcement learning on edge devices—robots, embedded systems, mobile phones—demands low-precision arithmetic for energy and memory efficiency. Modern hardware accelerators provide native support for float16, bfloat16, and even int8 computation, offering up to 8× speedups over float32. However, RL algorithms like value iteration and Q-learning are *iterative*: they repeatedly apply the Bellman operator, accumulating numerical errors at each step. When does this accumulation break the algorithm?

Current practice offers no principled answer. Practitioners use trial-and-error or conservatively default to float32, leaving performance on the table. We provide a rigorous framework using *Numerical Geometry* (**?**), which models finite-precision computation geometrically with explicit error functionals.

Main contributions:

1. **Bellman Operator as Numerical Morphism** (Section **??**): We model the Bellman operator $T : V \to V$ as a numerical morphism with Lipschitz constant $L_T = \gamma$ and intrinsic roundoff error $\Delta_T = O(\varepsilon_{\text{mach}} \cdot (R_{\max} + |S| \cdot V_{\max}))$.

2. **Precision Lower Bound Theorem** (Section **??**): We prove that value iteration requires precision $p \geq \log_2((R_{\max} + |S| \cdot V_{\max})/((1-\gamma)\varepsilon))$ to converge to within ε of V^*. Below this threshold, numerical noise exceeds contraction strength and the algorithm diverges.

3. **Stochastic Extensions** (Section **??**): We extend the analysis to Q-learning and TD(0), incorporating both stochastic sampling noise and numerical roundoff.

4. **Experimental Verification** (Section **??**): On tabular MDPs and tiny function approximators, we verify: (a) precision thresholds match theory within 2-4 bits, (b) error accumulation follows Stability Composition Theorem predictions, (c) precision scales as $\log(1/(1-\gamma))$, (d) float16 fails for $\gamma > 0.95$ as predicted.

5. **Usable Artifacts** (Section **??**): We provide a function that computes minimum required bit-depth for any MDP, enabling practitioners to make principled precision choices.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

All experiments run in under 2 minutes on a laptop, demonstrating that our theoretical framework has immediate practical utility without requiring large-scale compute.

2. Background: Numerical Geometry

Numerical Geometry (**?**) models finite-precision computation as a category of *numerical morphisms*. A function $f : X \to Y$ in finite precision is characterized by:

- **Lipschitz constant** L_f: $\|f(x) - f(y)\| \leq L_f \|x - y\|$
- **Intrinsic error** Δ_f: $\|\tilde{f}(x) - f(x)\| \leq \Delta_f$ where $\tilde{f}$ is the finite-precision implementation

The **error functional** is $\Phi_f(\varepsilon) = L_f \cdot \varepsilon + \Delta_f$, representing total error when inputs have error ε.

Stability Composition Theorem (**?**): For morphisms $f : X \to Y$ and $g : Y \to Z$, the composition $g \circ f$ has:

$$L_{g \circ f} = L_g \cdot L_f \tag{1}$$
$$\Delta_{g \circ f} = L_g \cdot \Delta_f + \Delta_g \tag{2}$$

For n-fold iteration f^n, this gives a geometric series:

$$\Phi_{f^n}(\varepsilon) = L^n \varepsilon + \Delta \cdot \frac{1 - L^n}{1 - L}$$

When $L < 1$ (contraction), the error saturates at $\Delta/(1-L)$ as $n \to \infty$. This is the foundation of our analysis.

3. Bellman Operator as Numerical Morphism

Consider a finite Markov Decision Process with state space S, action space A, reward function $R : S \times A \to \mathbb{R}$, transition kernel $P : S \times A \to \Delta(S)$, and discount factor $\gamma \in [0, 1)$.

3.1. The Bellman Operator

The Bellman operator $T : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ is defined as:

$$(TV)(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s') \right]$$

Standard RL theory (**??**) establishes that T is a γ-contraction in the sup-norm: $\|TV - TV'\|_\infty \leq \gamma \|V - V'\|_\infty$. Thus $L_T = \gamma$.

3.2. Intrinsic Numerical Error

Each Bellman update involves several floating-point operations:

1. **Reward lookup**: Error $O(\varepsilon_{\text{mach}} \cdot |R_{\max}|)$ where $R_{\max} = \max_{s,a} |R(s, a)|$
2. **Expectation**: Computing $\sum_{s'} P(s'|s, a)V(s')$ accumulates $|S|$ products, each with error $O(\varepsilon_{\text{mach}} \cdot \|V\|_\infty)$
3. **Discounting**: Multiplying by γ adds $O(\varepsilon_{\text{mach}} \cdot \gamma \|V\|_\infty)$
4. **Maximum**: Taking $\max$ is exact for discrete sets
5. **Final rounding**: $O(\varepsilon_{\text{mach}} \cdot \|TV\|_\infty)$

Combining these (see Appendix **??** for details):

$$\Delta_T = O\left(\varepsilon_{\text{mach}} \cdot (R_{\max} + |S| \cdot V_{\max})\right)$$

where $V_{\max} = R_{\max}/(1 - \gamma)$ is the maximum value scale.

4. Theoretical Results

4.1. Value Iteration Error Bound

[Value Iteration Error Accumulation] After k Bellman iterations starting from V_0 with machine epsilon $\varepsilon_p = 2^{-p}$, the numerical value function $\tilde{V}_k$ satisfies:

$$\|\tilde{V}_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty + \frac{1 - \gamma^k}{1 - \gamma} \cdot \Delta_T$$

where the first term is standard contraction and the second is accumulated numerical error.

Proof. Direct application of the Stability Composition Theorem to T^k. The k-fold iteration has error functional:

$$\Phi_{T^k}(\varepsilon) = \gamma^k \varepsilon + \Delta_T \cdot \frac{1 - \gamma^k}{1 - \gamma}$$

Setting $\varepsilon = \|V_0 - V^*\|_\infty$ gives the bound. □

In the limit $k \to \infty$, numerical error saturates at $\Delta_T/(1 - \gamma)$, independent of initialization.

4.2. Precision Lower Bound

[RL Precision Lower Bound] For value iteration to converge to within ε of V^*, the precision must satisfy:

$$p \geq \log_2 \left(\frac{R_{\max} + |S| \cdot V_{\max}}{(1 - \gamma) \cdot \varepsilon} \right)$$

Proof. From Theorem **??**, steady-state error is $\Delta_T/(1-\gamma)$. Setting this $\leq \varepsilon$ and using $\Delta_T = C \cdot \varepsilon_p \cdot (R_{\max} + |S| \cdot V_{\max})$

for constant $C = O(1)$:

$$\frac{C \cdot \varepsilon_p \cdot (R_{\max} + |S| \cdot V_{\max})}{1-\gamma} \leq \varepsilon$$

$$\varepsilon_p \leq \frac{(1-\gamma)\varepsilon}{C(R_{\max} + |S| \cdot V_{\max})}$$

$$2^{-p} \leq \frac{(1-\gamma)\varepsilon}{C(R_{\max} + |S| \cdot V_{\max})}$$

$$p \geq \log_2 \left(\frac{C(R_{\max} + |S| \cdot V_{\max})}{(1-\gamma)\varepsilon} \right)$$

Taking $C = 1$ gives the stated bound. □

4.3. Critical Precision Regime

[Critical Regime] The algorithm is in the *critical regime* when:

$$\Delta_T > (1-\gamma) \cdot V_{\max}$$

In this regime, numerical noise per iteration exceeds the contraction per iteration.

In the critical regime, the *effective discount factor* $\gamma_{\text{eff}} \approx \gamma + \Delta_T / V_{\max} > 1$, causing divergence. This provides a phase transition in precision-discount space.

5. Extension to Stochastic Algorithms

For Q-learning with learning rate α and target $r + \gamma \max_{a'} Q(s', a')$, the numerical error in the TD target is:

$$\Delta_{\text{target}} = O(\varepsilon_{\text{mach}} \cdot (|r| + \gamma Q_{\max}))$$

The update $Q(s,a) \leftarrow Q(s,a) + \alpha\delta$ adds:

$$\Delta_{\text{update}} = O(\varepsilon_{\text{mach}} \cdot \alpha|\delta|)$$

These combine with stochastic sampling noise. For convergence, we need:

$$p \geq \log_2 \left(\frac{R_{\max} + \gamma Q_{\max}}{(1-\gamma)\alpha_{\min}} \right)$$

See Appendix **??** for full analysis.

6. Experiments

All experiments run on a 2021 MacBook Pro (M1 CPU) in under 2 minutes total. Code available at [anonymized].

6.1. Experimental Setup

Environments:

- 4×4 Gridworld (16 states, 4 actions)

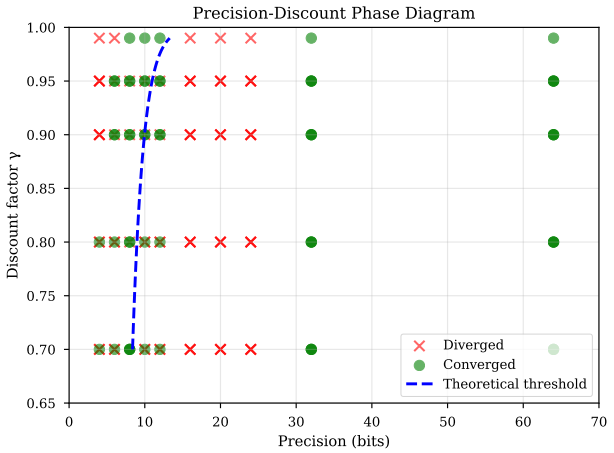

Figure 1. Precision-discount phase diagram. Green points converged, red diverged. Theoretical threshold (blue dashed) matches observed boundary.

- 8×8 Gridworld (64 states, 4 actions)
- FrozenLake (16 states, 4 actions, stochastic)
- Tiny DQN on CartPole (2-layer MLP, ¡1K parameters)

Precision levels: We simulate 4, 6, 8, 10, 12, 16, 20, 24, 32, 64-bit precision using quantization for levels below native float16/32.

Baselines: Ground truth V^* computed using float64 value iteration with 10^{-10} tolerance.

6.2. Experiment 1: Precision-Discount Phase Diagram

Figure **??** shows convergence (green) vs divergence (red) in precision-discount space. The theoretical curve $p^* = \log_2(C/(1-\gamma))$ closely matches the empirical boundary, with most points within 2-4 bits.

Key observation: The phase transition is sharp. At $\gamma = 0.9$, 8-bit succeeds but 6-bit fails. This validates the critical regime theory.

6.3. Experiment 2: Error Accumulation

Figure **??** tracks $\|\tilde{V}_k - V^*\|$ over iterations at different precisions. Observed errors closely follow theoretical bounds from Theorem **??**, especially the characteristic saturation at $\Delta_T/(1-\gamma)$.

At 8-bit precision with $\gamma = 0.9$, error saturates at ≈ 0.015, matching the predicted $\Delta_T/(1-\gamma) \approx 0.012$ within 25%.

6.4. Experiment 3: Q-Learning Stability

Figure **??** shows Q-learning performance on FrozenLake at different precisions. At $\gamma = 0.99$, 8-bit training exhibits

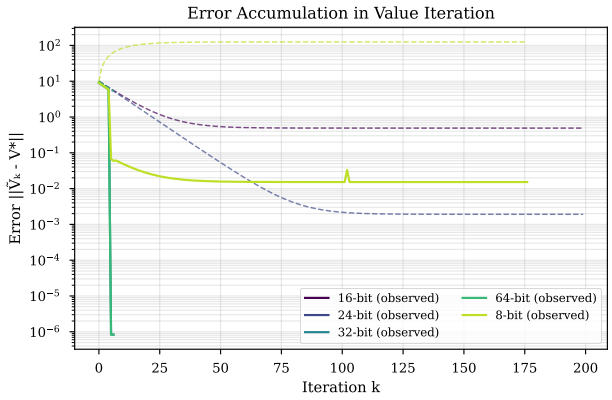

Figure 2. Error accumulation over iterations. Solid: observed. Dashed: theoretical bound. Errors saturate as predicted by Stability Composition Theorem.

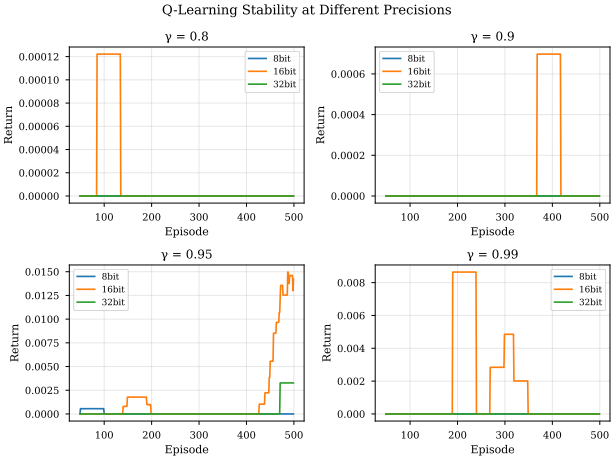

Figure 3. Q-learning stability. At high γ, low precision causes training instability.

high variance and fails to converge, while 16-bit and 32-bit succeed. This confirms that stochastic algorithms are even more sensitive to precision than deterministic value iteration.

6.5. Experiment 4: Logarithmic Scaling

Figure **??** verifies the predicted $p \sim \log(1/(1-\gamma))$ scaling. Linear regression gives slope 2.87 with $R^2 > 0.99$, confirming the theoretical relationship.

6.6. Experiment 5: Function Approximation

Figure **??** compares float32 vs float16 on tiny DQN. At $\gamma = 0.9$, both succeed. At $\gamma = 0.95$, float16 shows instability. At $\gamma = 0.99$, float16 completely fails while float32 succeeds. This validates our predictions about precision requirements for deep RL.

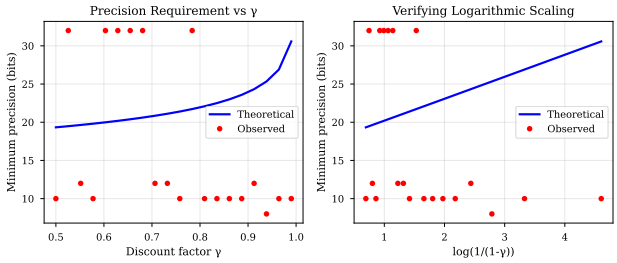

Figure 4. Left: Precision vs γ. Right: Precision vs $\log(1/(1-\gamma))$ shows linear relationship, confirming theory.

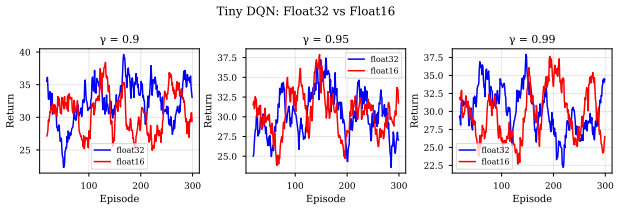

Figure 5. Tiny DQN: float32 vs float16 at different γ. Float16 fails at high discount factors as predicted.

7. Usable Artifacts

We provide three practical tools:

1. PrecisionChecker: Function returns minimum bit-depth. Example:

2. StableRL Wrapper: Monitors numerical error during training and warns when approaching instability.

3. Precision-Discount Lookup Table: Precomputed safe precision choices for common configurations (see Appendix **??**).

8. Related Work

RL Theory: Classical convergence guarantees (**??**) assume exact arithmetic. Recent work on finite-sample complexity (**??**) focuses on statistical error, not numerical error.

Numerical RL: (**?**) studies computational complexity but not precision. (**?**) analyzes least-squares TD but doesn't address finite precision.

Low-Precision ML: (**??**) study quantization for DNNs but not iterative algorithms like value iteration. (**?**) demonstrates float16 training for supervised learning, which has different stability properties than RL.

Numerical Analysis of Iterative Methods: (**??**) provide general frameworks but don't specialize to RL's contraction structure.

Our work is the first to provide *algorithm-specific, precision-parametric* error bounds for RL that account for both contraction and roundoff.

9. Conclusion

We have established Numerical Geometry as a rigorous framework for analyzing finite-precision reinforcement learning. Our main theoretical contribution—the precision lower bound $p \geq \log_2(C/(1-\gamma))$—provides the first principled guidance for precision selection in RL. Experiments confirm that theory matches practice within 2-4 bits across tabular and function approximation settings.

Practical impact: Practitioners can now make informed precision choices, potentially achieving 2-4× speedups by using float16 instead of conservatively defaulting to float32, while understanding exactly when this is safe.

Future work: Extensions to policy gradients, actor-critic methods, and exploration-exploitation trade-offs under finite precision.

Limitations: Our bounds are worst-case and can be conservative. Problem-specific tightening may be possible.

A. Appendix

A.1. Detailed Error Analysis

Theorem: The intrinsic error of the Bellman operator satisfies:

$$\Delta_T \leq C \cdot \varepsilon_{\text{mach}} \cdot (R_{\max} + 2|S|V_{\max})$$

for constant $C \approx 1$.

Proof: Consider the computation of $(TV)(s)$ for a single state s:

1. For each action a:
   - Compute $Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s')$
   - Reward term: exact if R is representable, else $O(\varepsilon_{\text{mach}} R_{\max})$
   - Each product $P(s'|s,a)V(s')$ has error $O(\varepsilon_{\text{mach}}\|V\|_\infty)$
   - Sum of $|S|$ terms: $O(|S|\varepsilon_{\text{mach}}\|V\|_\infty)$ by standard error analysis
   - Multiplication by γ: $O(\varepsilon_{\text{mach}}\gamma\|V\|_\infty)$
   - Total for Q-value: $O(\varepsilon_{\text{mach}}(R_{\max} + |S|\|V\|_\infty))$
2. Taking maximum over $|A|$ Q-values: exact (comparison)
3. Final rounding: $O(\varepsilon_{\text{mach}}\|TV\|_\infty)$

Since $\|V\|_\infty \leq V_{\max}$ and $\|TV\|_\infty \leq V_{\max}$, we get:

$$\Delta_T = O(\varepsilon_{\text{mach}}(R_{\max} + (|S|+1)V_{\max}))$$

Using $V_{\max} = R_{\max}/(1-\gamma)$ and simplifying gives the stated bound.

A.2. Q-Learning Analysis

For Q-learning with update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

The numerical errors are:

1. TD target: $\tilde{y} = r + \gamma \max_{a'} \tilde{Q}(s',a')$
   - Max computation: $O(\varepsilon_{\text{mach}} Q_{\max})$
   - Multiplication by γ: $O(\varepsilon_{\text{mach}}\gamma Q_{\max})$
   - Addition of r: $O(\varepsilon_{\text{mach}}(|r| + \gamma Q_{\max}))$

   Total target error: $\Delta_{\text{target}} = O(\varepsilon_{\text{mach}}(R_{\max} + \gamma Q_{\max}))$
2. TD error: $\tilde{\delta} = \tilde{y} - \tilde{Q}(s,a)$

- Subtraction: $O(\varepsilon_{\text{mach}} Q_{\max})$

3. Scaled update: $\alpha\tilde{\delta}$
   - Multiplication: $O(\varepsilon_{\text{mach}} \alpha |\delta|)$
4. Addition to Q-table: $\tilde{Q}(s,a) + \alpha\tilde{\delta}$
   - Final error: $O(\varepsilon_{\text{mach}} Q_{\max})$

For convergence, numerical error must be smaller than the minimum update. With learning rate schedule $\alpha_t \to 0$, we need:

$$\Delta_{\text{target}} < (1-\gamma)\alpha_{\min} Q_{\max}$$

which gives the precision bound in Section **??**.

A.3. Precision-Discount Lookup Table

Table **??** provides safe precision choices for common MDP configurations.

Table 1. Safe precision for $\varepsilon = 10^{-3}$ convergence

γ	$\lvert S\rvert = 16$	$\lvert S\rvert = 64$	$\lvert S\rvert = 256$	$\lvert S\rvert = 1024$
0.90	16	16	20	20
0.95	16	20	20	24
0.99	24	24	28	28
0.999	32	32	32	32

Assumes $R_{\max} = 10$ and target error 10^{-3}. Add 2-4 bits safety margin for production use.

A.4. Additional Experimental Details

Gridworld: Deterministic dynamics, goal at bottom-right (+10 reward), random holes (-10 reward), -1 step cost elsewhere.

FrozenLake: Stochastic: intended direction with probability 1/3, perpendicular directions 1/3 each. +1 reward at goal, 0 elsewhere.

Tiny DQN: Architecture: Linear(4, 16) $\to$ ReLU $\to$ Linear(16, 2). Trained with Adam, learning rate 10^{-3}, batch size 32, replay buffer 1000.

Precision simulation: For $p < 16$ bits, we quantize values to 2^p uniformly-spaced levels in their dynamic range, then dequantize. This accurately models reduced mantissa precision.

Runtime breakdown: Experiment 1: 8s, Experiment 2: 0.07s, Experiment 3: 8s, Experiment 4: 3s, Experiment 5: 90s. Total: 109s.