We thank all the reviewers for their time and effort during these tough times. Much of the feedback we received was inspiring, and we have worked day and night to respond. Details and additional numerical results are included below. However, please keep in mind that this is a theory paper. We doubt anyone before now thought it is possible to establish stability of Q-learning with nonlinear function approximation under the general conditions of our paper. The simple examples are included to illustrate the theory, and the stunning reliability of the new algorithms.

Assumptions (R#1 & R#4):

1. Smoothness of the network and ζ (R#1): The Lipschitz condition is only required for the ODE approximation—the Newton-Raphson flow is stable without this condition. Hence more sophisticated numerical methods will likely extend the theory beyond Lipschitz functions. Alternatively, we can project the estimates to a bounded convex region.

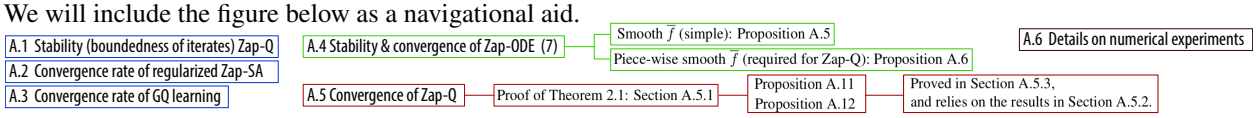
2. On non-singularity in (A4) (R#4): Indeed, we find that $A(\theta^*)$ is often singular in neural network architectures. We conjecture that the symmetry of the NN leads to both lack of uniqueness of θ^*, as well as the potential singularity. However: this condition is imposed to obtain a CLT. *It is not crucial for our main results.*

3. Roots of $\overline{f}$ (R#4): We have included fresh experiments below to show that $\overline{f}(\theta_n) \to 0$, and reward plots in the paper are consistent with this conclusion. We have not observed 'bad roots' from 'dead units' in our experiments, but will consider how the algorithm can be modified to avoid this potential threat.

4. On stability in Section A.1 (R#1): *This assumption is not required.* In Theorem 2.1 we *assume* that the parameter sequence is bounded. $A2_\infty$ provides a sufficient condition for boundedness (avoiding projection of parameter estimates).

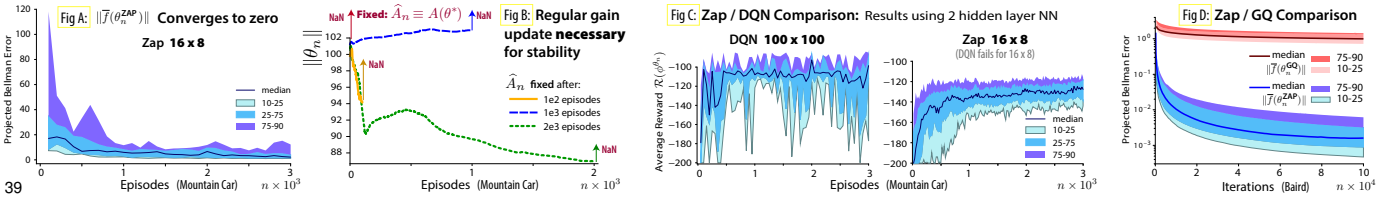
5. On ε-greedy policies (R#1): Incorporating exploration in these algorithms, while maintaining stability guarantees, is an important topic for future research. We will expand on the discussion currently found below (A1): there is hope that stability can be established, provided we adapt the policy on a slow time scale. This works well in experiments.

Reorganization & Discussions (R#1 & R#4): The high-level style was chosen to reach a broad audience, and motivate readers to read the supplementary material for technical details. Nevertheless, we welcome the reviewers criticisms: we will include more technical insights in a revision, making use of the additional page available for the final version. Parts of Section A.5.1 will be moved to the main body, highlighting the big challenge dealing with discontinuous dynamics. We will include the figure below as a navigational aid.



Connection between Zap-Q and Newton-Raphson flow (R#1): The Zap-SA algorithm is designed to approximate the Newton-Raphson flow: see lines 51-55, much like an Euler-discretization. Stochasticity presents a minor challenge. The complexity of the paper arises mainly because the vector field (7) is not continuous in applications to Q-learning.

Experiments (R#1 & R#4): We have conducted many more experiments over the past week in response to the reviewers. The figure below is the basis of the summary that follows



1. Convergence of $\overline{f}(\theta_n)$ (R#4): Fig A shows that $\|\overline{f}(\theta_n)\|$ converges to zero for the mountain car example.

2. Fixing $\widehat{A}_n$ (R#1 R#3 & R#4): We update $\widehat{A}_n$ every 50 samples in these experiments (see line 254 of the paper). In experiments this week, we found that performance degrades and is eventually unstable with increasing update interval. Fig B shows that the algorithm is not stable if the matrix is fixed, and the worst option is to take $\widehat{A}_n = A(\theta^*)$ for all n.

3. Comparisons (R#1): This week we tried the off-the-shelf implementation of DQN from bsuite (Osband et. al., ICLR, 2020, online). Using their default hyperparameters and the same environment setup used in our paper, we tested DQN with two layer NN, 100×100. The average rewards $\mathcal{R}(\phi^{\theta_n})$ estimated based on eq (31) are shown in Fig C. DQN with a big network is able to learn better policies, but it has high variance. Zap-Q is reliable and consistent in every example considered. The smooth behavior shown in Fig C is typical of Zap-Q, while DQN failed for the 16×8 NN.

4. Comparison with GQ-learning (R#4): GQ-learning [30] aims to solve (18) with linear function approximation. We applied this algorithm to mountain car using tile coding as basis vectors, but were unable to obtain meaningful results in this one week period. Fig D summarizes the behavior of the two algorithms for Baird's example [3].

To R#3 on complexity of inverting $\widehat{A}_n$: We are eager to explore more efficient techniques to approximate the Newton-Raphson flow. This may require advice from experts in the numerical analysis and optimization communities.

To R#2 on the complexity of the paper: We will add a short tutorial on SA and RL to the supplementary material.