We thank the reviewers for their comments and insightful reviews.

=== R1 ===

Best approximator in max-norm. Prior literature (e.g., Remi Munos, *"Error Bounds for Approximate Value Iteration",AAAI-05*, Remi Munos, Csaba Szepesvari, *Finite-Time Bounds for Fitted Value Iteration*, JMLR 2008) studied errors measured in weighted p-norm. This approach is often preferable when using "standard" parametric regression algorithms, as the weighted p-norm can be directly minimized at learning time (e.g., in least-squares regression, the ℓ_2-norm is minimized). Nonetheless, this gives rise to the so-called concentrability coefficients, which may be very large (even unbounded). In our case, the interpolation algorithm directly controls the max-norm (see e.g., Lemma 1) and this allows us to derive the analysis directly in max-norm and avoid concentrability coefficients.

=== R2 ===

Trade-off amplification and anchor points. In the worst case even the best trade-off between $\bar{C}$ and K may indeed lead to an exponential complexity. This is not surprising as in the worst case the inherent Bellman error may be unbounded and standard AVI tends to diverge. Recent work (Jinglin Chen, Nan Jiang, *Information-Theoretic Considerations in Batch Reinforcement Learning*, ICML 2019, Conjecture 8) has even conjectured an exponential lower bound in case of unbounded Bellman error. As a consequence, there might exist a fundamental barrier to obtaining polynomial sample complexity in the worst case. Nonetheless, in many other cases a good trade-off between amplification and anchor points may correspond to a much smaller sample complexity (e.g., the condition in Prop.1 could be achieved by a polynomial number of anchor points). Indeed in our experiments even in the case of unbounded Bellman error, the extrapolation can be controlled, and thus we can obtain satisfactory solutions by slightly increasing the number of anchor points without requiring an exponential number of them (see e.g., the experiments on the Tsitsiklis and Van Roy domain and the linear bandit).

Construction of anchor points. In the paper we propose a first heuristic algorithm to automatically construct a set of anchor points (see beginning of page 6). While we do not have any guarantee for the method, in our preliminary experiments it seems effective in building a compact set with small amplification factor.

Incremental construction from H down to 1. A good choice for the anchor points depends only on the linear architecture, i.e., it is computed exclusively on the basis of the feature map $\phi_t(\cdot,\cdot)$. Thus if one already knows a good set of anchor points at the final timestep $t = H$ and the feature map $\phi_t(\cdot,\cdot)$ does not vary a lot for $t = 1,\dots,H$ then the set of anchor points for the timestep $t = H$ is also a good choice for prior timesteps $t = 1,\dots,H-1$.

Infinite horizon. The algorithm does not need any modification (other than the addition of the discount factor) to deal with the infinite horizon case, with the additional benefit that the identification of the support points can be done once at the beginning as opposed to every timestep (as it only depends on the feature map $\phi(\cdot,\cdot)$). However, one would need to change the analysis. Note that Yang and Wang's analysis operates in the much easier setting of zero Bellman error.

Continuous state space. The main algorithm can be applied to the continuous state space case without any modification, and the choice of discrete MDPs in the experiments is for illustrative purposes.

=== R3 ===

Computational complexity of the heuristic. In its most naive form and without further structure, one would need to loop through the state action pairs; for large or continuous state-actions spaces one should sample the state-action pairs to reduce the complexity. This should suffice as we only require approximate convex hulls. For computing the θ's, this is a linear program and the best known computational complexity can be found in the Arxiv paper "Solving Linear Programs in the Current Matrix Multiplication Time"; other methods, like interior point methods, may be used.

Experiments. We can report more thorough statistics for the anchor points (i.e., number and positions, and resulting extrapolation coefficients) as those are computed inside the program. Some of these are reported in figure 3 for that example, but we can add them for the other examples as well.

Real Life Experiments As the reviewers points out, we have chosen examples where realizability holds and the choice was deliberate to reduce the number of confounding factors: the approximation error may in general affect the comparison between methods, and might obscure the key underlying processes. We agree with the reviewer that a comparison with averagers like k-NN or kernel based would be practically interesting, but it raises questions of how to best define the function class (for example, the position of the points in a nearest neighbour procedure, the type of radial basis function, etc.) encoded by those settings as a fair comparison to our setting. Therefore in our work we focus on fixing the function approximation class (general linear value functions) and evaluate the impact of the algorithm used to fit the function class. A key benefit of our approach is that it does not modify the underlying (linear) feature representation, allowing the user to use the linear representation with, for example, approximate value iteration, and should this fail, the user can switch to our algorithm and progressively increase the number of support points while keeping the same feature representation. Finally, we can easily add more variations of the chain experiments with varying horizon (for example H = 50, 100, 200) and feature representation as noted by the reviewer.