We thank the reviewers for their thorough feedback. We address each of their concerns as follows.

@R1, R3, R4 **Numerical experiments:** The main focus of this paper is to extend the "theory" of escaping from saddle points to the decentralized setting. However, following the reviewers' suggestion, we'll provide some experiments in the revised paper. Here, we briefly mention the setup and results and will include more details in the revised paper. Our goal is to show that PDGT is able to escape from saddle points quickly. We compare PDGT with D-GET which is a decentralized gradient tracking method that "doesn't use the perturbation idea" [37]. We focus on a matrix factorization problem for the MovieLens dataset, where the goal is to find a rank r approximation of a matrix $\mathbf{M} \in \mathcal{M}^{l\times n}$, representing the ratings from 943 users to 1682 movies. Each user has rated at least 20 movies for a total of 9990 known ratings. This problem can be written as: $(\mathbf{U}^*, \mathbf{V}^*) := \underset{\mathbf{U}\in\mathcal{M}^{l\times r},\mathbf{V}\in\mathcal{M}^{n\times r}}{\operatorname{argmin}} f(\mathbf{U},\mathbf{V}) = \underset{\mathbf{U}\in\mathcal{M}^{l\times r},\mathbf{V}\in\mathcal{M}^{n\times r}}{\operatorname{argmin}} \|\mathbf{M}-\mathbf{U}\mathbf{V}^\top\|_F^2$. We consider different values of target rank and number of nodes. Both methods are given the same randomly generated connected graph, mixing matrix, and step size. Further, they are initialized at the same point which lies in the neighborhood of a saddle point. Note that in this problem all saddles are escapable and each local min is a global min.

In Fig. 1 the experiment is run for 10 nodes, and the target rank is 20. Initially both algorithms are stuck close to a saddle point and make very little progress. However, since the theoretical criterion for PDGT is satisfied in the very first rounds (small average gradient and consensus error) we have injection of noise. This nudge is sufficient to accelerate substantially the escape of PDGT. As we see in the plot, D-GET remains close to the saddle point at least until iteration 1400 where we can see the gradient increasing somewhat faster. At the same time PDGT escapes the saddle point, decreases the loss and approaches a local minimum.

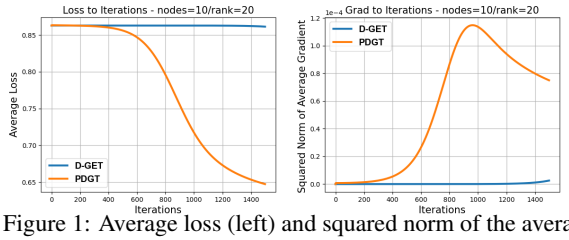

Figure 1: Average loss (left) and squared norm of the average gradient (right) vs. iteration (10 nodes and target rank 20).

In Fig. 2, the experiment is run for 30 nodes and the target rank is 30. Similarly, PDGT escapes from the saddle point much faster and decreases the loss substantially before it reaches the local minimum. We observe that D-GET also escapes the saddle point eventually following a similar trace to PDGT after spending a lot longer at the saddle. Interestingly, for this experiment, we observed that some parameters such as the stepsize of the first and the second phase, the injected noise and the threshold before we inject noise can afford to be substantially greater than the theoretical propositions casting PDGT useful for a series of practical applications.

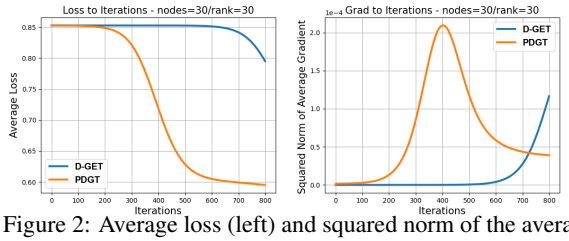

Figure 2: Average loss (left) and squared norm of the average gradient (right) vs. iteration (30 nodes and target rank 30).

@R1 **Importance of the results considering [15]:** On a high level the PGD method in [15] (designed for centralized optimization) is not applicable to decentralized settings as it requires coordination between all nodes at each iteration which has a prohibitive communication cost. Moreover, notice that PGD is a descent algorithm whose behavior is rather well understood, whereas our proposed PDGT method is a non-monotonically decreasing algorithm which requires the *careful construction and analysis of a potential function* in order to argue about its convergence. Specifically, when the PDG algorithm adds noise to the iterate, there exist a specific number of steps and function decrease threshold that characterize the space topology at the point that the noise was injected. When the same injection happens in a distributed algorithm as PDGT, the consensus error (which potentially increases exponentially fast) and more generally the fact that the algorithm is not strictly descent deems the escape from the saddle point non trivial and the existence of appropriate thresholds unclear. Diving into more technical details, one can observe that in comparison to Lemma 17 in [15], our corresponding Lemma 15 utilizes a new potential function and controls the consensus error among the nodes deriving a novel and more complex analysis. Specifically, the last terms in (148) and (149) correspond to the consensus error of the escaping sequences. To control these term we need to prove an induction with two parts as shown in (151), (152) and invoke novel Lemma 14. It follows that equations (159)-(163) should hold, and an interesting connection is derived between the potential function parameter α, the second phase step size η_2 and the dimension d. The aforementioned relation presents a trade-off between the first and second phase stepsizes, η_1 and η_2, through α and thus achieving the optimal overall convergence rate requires fine-tuning the stepsizes as well as coming up with the tightest bounds for quantities such as the target decrease of the potential function $\mathcal{F}$, the bound on the norm of the iterates P and the radius of the noise ball $\mathcal{R}$. All these values in our analysis are different from the ones in [15] and our main proofs are more complex and technical confronting new challenges presented in the distributed framework. Finally, the polynomial dependence on d appears to be crucial in controlling the consensus error after the injection of noise. To be more precise, notice that to prove the base of the induction in (151) we need to lower bound $\zeta\psi_0 = \zeta\mathcal{R}\mu$ where $\mu \in \left[\frac{\delta_2}{2\sqrt{d}}, 1\right]$, deriving the dependence on dimension. The importance of μ belonging in this interval becomes apparent in Lemma 16 and the corresponding Lemma 15 in [15], where the volume of the stuck region is bounded.