We are grateful to all the reviewers for taking the time, reading our paper, and providing many useful comments. We are particularly humbled by the fact that all reviewers were unanimously supportive of our work. Before addressing each reviewer's individual questions, we make some general remarks regarding comments that were shared by the reviewers.

Hyper-parameters: As reviewers noted, while DINGO converges for any choice of hyper-parameters, correctly choosing them will result in better performance. We have discussed this in Sections 3-4 and have provided theoretical guidelines in Lemmas 1 and 3. Sensitivity to θ and the consequential effect of Cases 2-3 is examined in Appendix B.1. We will move this result to the main paper, as the importance of such results was highlighted by reviewers.

Assumptions: Our assumptions (1, 2, 3, 5) are generalizations of those typically found in the literature. For example, if each f_i is smooth and strongly convex, then all our assumptions would be satisfied. As another example, if each f_i is an under-determined least squares objective, hence not strongly convex, Assumption 3 is still satisfied. In fact, Assumption 3 is weaker than requiring the pseudoinverse of each $\mathbf{H}_{t,i}$ is bounded in spectral norm.

Sub-problems: As noted in the paper, our analysis is limited to the exact solutions of the least-squares sub-problems, which can be done in $\mathcal{O}(d^3)$ time. This is the same for computing exact solutions to the sub-problems of GIANT. In practice, however, such least squares sub-problems can be solved inexactly using very efficient iterative methods. We are actively in the process of developing inexactness theory for our future work, and the main challenges lie in guaranteeing boundedness of the approximate solutions as well as ensuring sufficient descent using the average direction.

Line-search: Analogous to GIANT, backtracking line-search is performed distributively, and requires only 2 communication rounds. Each worker, in parallel, computes the gradient with each step-size from some predetermined list of k step-sizes, e.g., $1, 2^{-1}, 2^{-2}, \ldots, 2^{-k+1}$, and the driver node then aggregates and checks the line-search condition at each step-size. Alternatively, backtracking line-search can be done sequentially in which checking each step-size takes 2 communication rounds. The term τ in Corollary 1, which is a lower bound on the step-size under all cases, determines the maximum communication cost needed during line-search. We will elaborate on this in the paper.

Comparison to Related Work: On Page 2 we briefly compare the advantages and disadvantages of related methods. Then, we elaborate further on Page 3. We will add a table to the main paper that summarizes the discussion on pages 2 and 3 . Such aspects include: restrictions on data distribution and functional form of the objective, requirements on the degree of convexity/non-convexity, hyper-parameters and communication rounds per iteration. We will also include the number of communication rounds required to achieve a solution, and under what metric. For example, for DINGO to achieve $\|\mathbf{g}_t\| \leq \varepsilon$ then, by Corollary 1, it requires $\mathcal{O}\big(\log(\varepsilon)/(\tau\rho\theta)\big)$ communication rounds.

Below, we address each reviewer's specific comments:

Reviewer #1: (i) Indeed, as DINGO is minimizing the norm of the gradient, it may converge to a local maximum or saddle point in non-convex problems that are non-invex. We have mentioned this in Future Work on page 8. However, we agree and fully appreciate the need to highlight this disadvantage earlier on, e.g., in Contributions. We will do so in the revision. **(ii)** The factor 2, in Armijo condition, arises as we multiply both sides by 2 to remove the $1/2$ from equation (4). **(iii)** We will present the high level description (Section 2) before presenting Algorithm 1.

Reviewer #2: (i) Thank you for the reference. Indeed, it is relevant and very interesting, and we will reference it. However, it appears to be suited to decentralized settings; whereas, our focus is on centralized methods. We aim to compare extensively with it and similar methods, such as CoCoA, in future work. **(ii)** Each worker node uses its local Hessian information to transform the full-gradient. This is similar to the method GIANT. However, unlike GIANT, we don't impose restrictive assumptions that guarantee suitable descent automatically. Rather, we use three cases that are designed to ensure suitable descent in the norm of the gradient, despite $\mathbf{H}_t^\dagger \neq \sum \mathbf{H}_{t,i}^\dagger$. Please refer to Remark 1.

Reviewer #3: (i) As correctly pointed out, the ability to eventually converge using the step-size 1 is one of the most important aspects of Newton-type methods. To properly study this, we require local convergence analysis, which we are actively pursuing as an extension to our work. It seems like that this property is tightly intertwined with the number of workers and data distribution. **(ii)** Indeed, the issue of preconditioning is essential to the performance of iterative solvers. In future work, we aim to evaluate preconditioning ideas, proposed by DiSCO, in our context and evaluate convergence.

Reviewer #4: (i) Please note that Lemma 1 is not an assumption about DINGO, nor is it a statement about its overall performance. Rather, it illustrates the fact that if we were to make stronger assumptions, such as those in line with GIANT, e.g., if $\mathbf{H}_{t,i}$'s were to be invertible, then DINGO can be written in its simplest form, i.e., Case 1. Please see to Remark 1 for further discussion. **(ii)** Our implementation of DINGO uses Python with PyTorch on top of MPI and supports GPU and CPU. It can easily train an existing PyTorch Module and we will provide a link to the code in the final revision. **(iii)** We agree about the need to perform wall-clock time analysis, and we will do so extensively in future work as part of the extension of our current analysis to inexact sub-problem solutions. This is because such comparison, especially in distributed settings, is highly implementation dependent. **(iv)** All communication rounds, including those from line-search, were considered in Figure 1. **(v)** We will clarify what we mean by "worker" in Figure 1.