We thank the reviewers for their insightful comments. We address their interrogations and comments below.

R1. *- Practical/concrete motivation for why/when online OT computation is needed.* Any applications that requires to estimate OT distances between large point clouds can benefit from online OT estimation, that accelerates training. An example is training generative models. In this case, the samples are renewed at each iteration of a training algorithm, and may be used to better evaluate the distance to minimize. We will better motivate our work.

- No experiments on high-dimensional settings, arguably those for which a streaming setting would be most compelling. This is indeed a weakness in our experiments, as we have measured performance up to $d = 10$. Real-world ML applications would typically consider points in latent spaces, with typical dimension $d = 128$. Similar convergence curves as those of Fig. 7 holds for 128-dimensional Gaussian. We will add these to the experiment section.

- In L62 it is claimed that the memory complexity increases lineraly on n_t. Should this be $O(n_t^2)$? The memory complexity is linear in n_t, as each potential is represented in memory by n_t points and weights. We will clarify.

- I'm not entirely convinced that using "number of computations" in the x-axis makes sense for Figs 1, 3 etc. We measure the number of computations needed to obtain a *first estimate* of the OT potentials, which is roughly proportional to wall-clock time (see answer to **R2**). It is of course higher for batch method than online method. Our intent in Fig. 3 is to show that online Sinhkorn efficiently warms up OT computation. We will clarify.

- I could not find a discussion or details on how the learning rate η_n is chosen in practice. We give practical recommendation regarding step-sizes and batch-sizes in Appendix B.3, and in particular Table 1. In experiments, we found that setting $n(t) \propto (1 + 0.1t)^{1/2}$, and $\eta_t = 1$ works best, although the range of usable exponents is rather wide. We will present Table 1 in the main text for clarification. See also App. C for details on hyper-parameters.

- Further references. We thank the reviewer for his insightful refrences on streaming method for EMD estimation, that we will discuss in the related work section. In the batch or online setting, regularization permits a faster estimation of OT distances, relying only on matrix-vector products. [39] fixes a spatial grid for the estimated barycenter, unrelated to observed samples, while we define potentials based on observed samples. We will discuss this in details.

R2. *- How is convergence affected by [...] the distributions involved.* We have tried to give more insight on this aspect in Appendix C. As predicted by the theoretical analysis, online Sinkhorn converges slowler for lower ϵ (or equivalently, less regular C, Fig. 5). For Gaussians distributions, online Sinkhorn outperforms batch Sinkhorn in all cases (Fig. 7).

- Could the authors comment on actual runtime? With proper GPU implementation of online Sinkhorn (using the *pyKeops* library), the C-transform wall-clock time is indeed roughly in $O(n_t^2)$. We have compared online Sinkhorn to batch Sinkhorn in term of wall-clock time, and found similar curves as reported in the paper, using batch-sizes larger than 1000. Batch Sinkhorn remains faster for small problems ($N < 10^4$), for which C can be precomputed and held in GPU memory. We will add wall-clock time experiments to the appendix.

- Confusion l.288-289. f and g are fit until C is formed, and we then run batch Sinkhorn. We will clarify.

- "It behaves like exp(1/eps)." Any intuition as to whether this can be improved? It is an inherent limitation of OT regularization (even in the batch setting), that cannot be improved: the sample complexity of unregularized OT is exponential in the dimension, while regularized OT enjoys fast rates. The constants before these rates must explode as the regularization disappear. This intuition is given in e.g. [18], and we will recall it.

R3. *- The per-iteration cost of the classic Sinkhorn algorithm can be reduced [...].* We have been too elusive on this aspect. Online Sinkhorn proves most useful in the case where the pairwise cost matrix must be computed on the fly due to memory constraints (and serves as a sound warmup otherwise). We will recall and discuss this observation.

- I find the Prop. 4 surprising. In Prop. 4, we assume that $\iota > 0$, and therefore that the batch-size goes to infinity. This is sufficient to ensure convergence, as the variance terms introduced by sampling are summable. For fixed batch-sizes, convergence cannot be guaranteed, due to the fact that $\sum_t \frac{1}{t}$ is not summable. The proof of Prop. 4 established a classical recursion between error terms, and requires $\iota > 0$ to conclude. We will discuss Prop. 4 more thoroughly.

R4. *- Experiments are performed in only simple cases.* This is a limitation of our work. The lack of gold-standard for estimating continuous OT distances makes it hard to evaluate our method on more complicated settings, as we are forced to approximate this gold-standard with very long runs of Sinkhorn algorithm. See also answer to **R1**.

- Soft C-transform. This term refers to Eq. (3). We will clarify.

- Related work. Bercu and Bigot's work is indeed relevant. It tackles the simpler problem of semi-discrete OT, that rewrites as a expected risk-minimization problem. A single finite dimensional potential must be estimated, which can be done through gradient descent. We will refer to Mena and Weed's refined sample complexities. The E-step of Sinkhorn-EM could be implemented using online Sinkhorn, with potential gain from warm-starting.