We thank the reviewers for their insightful comments. As pointed out by R1, R2, and R4, our submission presents a distinctive framework for BO, with algorithms designed for modern hardware. Our work makes MC acquisition functions practical and easy-to-use by providing a modular design that carefully abstracts away a highly efficient computational system. Optimization within this framework is accomplished through a novel SAA approach backed by theoretical convergence results. We demonstrate the ease of use of our software—and the generality of our theory—with our parsimonious one-shot KG algorithm that achieves SoTA results. Our contributions have already enabled other BO researchers' novel methodological work in areas such as min-max optimization, optimization of risk measures, high-dimensional BO, transfer learning, and cost-efficient search. BoTorch has been applied in papers on material design, ML model reduction, drug discovery, trajectory optimization, and RL, to name a few. Our work is therefore a significant and timely contribution that expands the scope of problems that can be readily addressed through BO. We will use the extra page to address the review feedback and improve clarity, including adding details from the appendix.

We emphasize that the software and theoretical components in this case are highly *complementary*: to enable the widespread use of MC acquisition functions, BoTorch incorporates approaches such as fast predictive variance computations, parallelized posterior sampling, batch evaluation, and auto-differentiation of a large number of MC samples. Abstractions such as `fantasize` allow auto-differentiating through complex operations (in this case, batched low-rank updates of posterior variance caches and subsequent posterior predictions of conditioned GPs) in a transparent fashion. All of this functionality is specifically designed to exploit modern deep learning (DL) paradigms and hardware acceleration. Conversely, simply using a DL framework for conventional BO relying on Cholesky-based inference and analytic acquisition functions cannot effectively utilize the batch processing that DL frameworks are built around, and so will provide little benefit (this is a core aspect that distinguishes BoTorch from GPFlowOpt). Thus, our main contribution is the *joint* development of the MC+SAA methods together with a framework that renders their use practical, neither of which would, by itself, provide nearly as much value without the other.

Although submissions with different types of contributions can indeed be difficult to evaluate, they can also provide significant value that complements more traditional papers. There is a precedent at NeurIPS for domain-specific frameworks that leverage DL frameworks to simplify implementations and scale out computation (Gardner et al., 2018; Shazeer et al., 2018; Tran et al., 2018; Tran et al., 2019). For example, Tran et al. [2018] describe embedding probabilistic programming into DL frameworks, and Tran et al. [2019] discuss a collection of "layers" that intertwines DL components with probabilistic elements. Like our work, these papers provide a framework for research that abstracts away computational aspects to simplify and scale computation. We additionally provide theoretical results that form the basis of new approaches to BO.

R1: **(1)** Our motivation was to clearly lay out the abstractions and how they fit with the MC framework before illustrating their use. Thanks for your suggestions to improve the clarity of these sections, which we will incorporate. **(2)** The pending points strategy and the decorator are described on pg 5, in the paragraph directly before Sec 5.1. **(3)** We will be more specific in our discussion about GPFlowOpt, and hope the main text above clarifies this.

R2: **(1)** To provide better background, we will introduce aspects of the work such as fantasies and KG earlier on. **(2)** $f_D(\mathbf{x})$ and $y_D(\mathbf{x})$ are the posterior at the set of points $\mathbf{x}$, so under a GP prior these are indeed MVNs. The abstract posterior objects f_D and y_D are of course GPs. We will make this distinction more precise. **(3)** Wall times in Fig. 6 are end-to-end, including querying the GP. **(4)** The constrained experiments compare against random since (to our knowledge) other maintained packages do not support constraints. We used H6 to contrast with unconstrained results.

R3: **(1)** The goal of our work is to present a practical and efficient way of doing BO with MC acquisition functions. To this end, we demonstrate our methods compare favorably across a variety of well-established packages—to the satisfaction of other reviewers, and in line with the BO literature published at NeurIPS and ICML. **(2)** We do reference GPyFlowOpt in Sec 2. Given that GPFlowOpt (i) has never been rigorously evaluated (including within the GPFlowOpt tech report itself), (ii) has an algorithmic foundation that is not amenable to MC acquisitions, (iii) has not been maintained for 2 years, and (iv) won't build; we hope the reviewers will agree that comparison is not critical. **(4)** In Fig. 12, we included wall time comparison between KG implementations. Additionally, here are average wall time (sec) per iteration for CPUs across the synthetic problems (Figs 4, 10, 11, 13): MOE EI, 0.1; BoTorch EI, 1.1; BoTorch NEI 9.0; BoTorch OKG, 102.3; Dragonfly, 141.4; MOE KG, 160.6. We did not record per-iteration runtimes of GPyOpt (due to its full-loop implementation), but can include results in the camera-ready version. Note that MOE EI performed poorly on all benchmarks. **(5)** We will clarify the drawbacks of OKG. The primary drawback is that the dimensionality of the optimization problem grows with the number of fantasies N_f . In addition, memory complexity, due to batched GPs used for fantasization, also grows with N_f. In practical settings, only a moderate N_f can be used (we use 128).

R4: **(1)** While some BO algorithms do have simple structure, this is not universally true (NEI, KG). Further, the fact that BO can be implemented in a functional way using hardware acceleration does not mean that it will be particularly fast or easy to work with (see above). **(2)** We will use the additional page in the final version to move relevant parts of the appendix into the main paper, including the SAA results.