Higher-order derivatives of likelihood. As proposed by Reviewer 1, we will comment that, unlike full HMC, the embedded Laplace approximation (ELA) requires the Hessian w.r.t θ, and furthermore, the third-order derivatives (line 6 in algorithm 1). It is possible to obtain these derivatives using automatic differentiation but only at an important cost, which is why we stick to analytical derivatives for the log likelihood. Our intuition is that we can bypass this calculation but this result is not in the present work. This is why we have not, as noted by Reviewer 3, run computer experiments on general likelihoods, but we agree that this is exactly the right direction for future research.

Choice of benchmark. We use adaptive HMC as our benchmark because for our main motivating problems, in Sections 5 and 6, this is the method used in references. The authors of these models choose MCMC in order to accommodate a high-dimensional and intricate prior, built as a mixture of Cauchy distributions. We will, as suggested by Reviewer 3, detail which tuning parameters of HMC get adapted, likely in the Supplementary. As Reviewers 3 and 4's comments imply, the reader will ask: "why use the Laplace approximation and not variational inference?" There already exist several papers that compare the ELA to variational inference in various settings, for example Nickish and Rasmussen [2008], Vanhatalo and Vehtari [2010]. Their analysis finds that marginalization with ELA yields comparable, and at times better performance, for several problems of interest. Section 1.1 will mention variational inference (notably its implementation in gppytorch) and point the reader to the above references. With that said, our paper's focus remains on an improvement to the Laplace approximation and to HMC.

ARD. Reviewer 4 correctly indicates automatic relevance determination regression (ARD) is relevant to the problems we study. The regularized horseshoe model and the SKIM are in fact ARD models with a special choice of sparsity inducing prior. We avoided the term ARD because it is a misnomer, as in most used cases it assesses non-linearity and not relevance, see e.g. Paananen et al. [2019]. Nevertheless we will draw the connection to ARD for the benefit of the readers.

Impact and applications. Reviewer 4 points out that typically the choice of K comes from a small subset and that the dimension of the hyperparameter is small. We agree, and our method is not meant to supersede existing methods for classic GPs. Our goal is to empower users to fit non-classic, but in our view nevertheless important, GPs, such as those presented in Sections 5 and 6; notably the SKIM which has a non-trivial K. Furthermore, many popular hierarchical models can be cast as latent Gaussian models – as done for example in Section 5 and Section E.2 of the Supplement – which makes for a relatively broad range of applications.

Naming convention. To justify the name "latent Gaussian model", we will state that we follow the convention in Rue and Chopin [2009] and emphasize the term is used differently across the literature. Per Reviewer 2's suggestion, we will state that K is the covariance matrix.

Including user time in figures. We appreciate Reviewer 1's comments on figures 4 and 5. Papers often compare the computational run times of (hopefully) well-tuned algorithms, with less consideration for user time to tune said algorithms. We are not sure how to best measure this latter factor and include it in the figures. Our plan is to mention in the captions how many times *we* had to fit each model to tune it. We will tinker with better alternatives.

Related work on adaptive HMC. Based on comments by Reviewers 1 and 3, it seems readers would appreciate a brief discussion on adaptive HMC methods; notably semi-separable HMC by Zhang and Sutton [2014] and RHMC. These methods are indeed designed to overcome the geometric difficulties our paper is concerned with. We did not use them as benchmarks because they are either not commonly used, computationally expensive, or because building an efficient implementation in C++ – to make comparisons equitable – is not straightforward. We will however mention these contributions either in Section 1.1 or in Section D of the supplement.

Related work on implicit differentiation. Reviewer 3 refers us to the work by Lorraine et al. [2019], which uses the implicit function theorem (IFT), as one would when differentiating a black box optimizer (notably mentioned in Section 2 of our submission). The IFT requires an inverse Hessian term which must be fully computed and a Hessian term that can be contracted with a co-tangent vector. Because you have to compute the full inverse Hessian, a good approximation is desirable. We considered using IFT but after some experimentation chose to not treat the optimizer as a black box and use an "open" Newton solver instead. This allowed us to bypass the computation of the inverse Hessian altogether. Hence IFT is a valid choice but we have found our approach to work better for our problem. The comparison is interesting and may be worth describing in the Supplement.

References. (Nickis and Rasmussen, 2008) *Approximations for binary Gaussian process classification*, JMLR; (Vanhatalo and Vehtari, 2010) *Speeding up the binary Gaussian process classification*, Uncertainty in Artificial Intelligence; (Rue and Chopin, 2009) *Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations*, Journal of Royal Statistics B; (Lorraine et al, 2020) *Optimizing millions of hyperparameters by implicit differentiation*, Artificial Intelligence and Statistics; (Zhang and Sutton, 2014) *Semi-Separable Hamiltonian Monte Carlo for Inference in Bayesian Hierarchical Models*, Neurips; (Paanen et al, 2019) *Variable selection for Gaussian processes via sensitivity analysis of the posterior predictive distribution*, PMLR.