We are grateful to all the reviewers for their feedback. Below we provide responses to the main comments.

**Reviewer 1:**
"I don't feel NeurIPS is the appropriate venue to publish it. My main concern about the paper is with respect to its appeal for the ML community at large. My impression is that its scope is rather limited. The presented examples are of reduced dimension" Although we respect reviewer's opinion we disagree and we believe that our work is very suitable for NeurIPS which is an interdisciplinary conference and MCMC is one of its subject areas. Many MCMC papers and related Monte Carlo methods have been previously published in NeurIPS. Also not all inference problems in statistics and ML are large scale or high-dimensional and certainly our method does not exclude applicability to high dimensions.

**Reviewer 2:**
"It was not clear how the Adaptive MCMC (AM) described in the results works i.e. what is the objective function of the adaptation. A reference is made to the supplement, but I didn't quite find it there (lines 226-227)." The underlying objective of AM is the minimisation of the KL divergence between the target distribution and the proposal distribution, as described earlier in section 4.1 in the tutorial paper of Andrieu and Thoms. Of course this optimisation is challenging because we only observe correlated samples which at the early adaptation stages are really far from the target.

"Line 266 and Figure 2 top panel. This should really be the auto-correlation plot." Thank you, we will follow your suggestion.

"The choice of $\rho_t$ on lines 136, 137 is not well motivated. In similar situations a Robbins-Monro sequence is typically used." We agree that the Robbins-Monro sequence is the one that ensures convergence in the limit. The motivation behind the RMSprop sequence we used (that is very popular in Deep Learning together with similar adaptive learning rate schemes such as Adam) is that in practise when you run for a fixed (relatively small) budget of stochastic optimisation/adaptation iterations it tends to provide more effective optimisation. Note that in our experiemtns we adapt only during burn-in, while at the collection of samples phase we keep the proposal fixed.

"The authors could give more theoretical justification for their MALA approximation to avoid computing the Hessian on lines 172-174." So far we only empirically observe that the fast MALA scheme tends to provide stostastic gradients with smaller variance leading to faster optimisation. We will try to analyse this theoretically by finding expressions of the variance (at least for simple targets) for the exact Hessian and the fast scheme.

" I would have like to see the results for Stan, which has a good implementation of NUTS, on the presented data sets. That would help highlight the advantage of the proposed method over the current standard practice." Currently all experiments are based on a MATLAB implementation and the NUTS version is precisely from the published 2014 article and follows Hoffman's implemenetation. We are going to provide code that reproduces all our results.

**Reviewer 3:**
"The specific algorithm (and simplifications) for MALA are clearly the "highlight", performance-wise, but may be slightly lower impact simply due to implementation headaches (it can be done easily enough by people familiar with deep learning or AD software, but that leaves out many practicing statisticians)." Thanks for the comment. For the final version we plan to release a non automatic differentiation (AD) based MATLAB implementation for all proposed algorithms. We also plan to provide pseudo-code showing how this fast MALA scheme can be implemented with the minimum number of vector operations (few details about this are already part of the supplement) and with AD possibly used only to compute the gradient of the log target.

"the one thing I find unsatisfying is the need to fall back on existing results for "optimal" acceptance rates when tuning $\beta$. For RW and MALA proposals, maybe this is even reasonable (there's some theory about optimal preconditioners, proposal covariances, mass matrices anyway), but it would be nice to address this more generally, or at least discuss the appropriateness of falling back on these recommendations." We agree that this is a limitation. The ideal will be to automaticlaly "learn" what is the optimal average acceptance rate for a specific target. Hoewever, the standard heuristics for adapting $\beta$ worked well in the all our experiments.

"For MALA and NUTS, what sort of preconditioning is done as a baseline? For NUTS as implemented in STAN and PyMC3, I know a standard practice is to run a short pre-run which is used to estimate the mass matrix; this is then used fixed with only the step size adapted online. Is something like this done here? What about for the non-adaptive MALA?" For NUTs we use the algorithm as defined in the initial 2014 paper and it is based on the corresponding implementation of Hoffman, which does not use preconditioners. Also the non-adaptive MALA is not using a preconditioner. Notice our method regarding MALA essentially allows to obtain a full covariane preconditioner using gradient-based adaptation.

"line 140, L is described as a "positive definite lower triangular matrix" … presumably a typo (it is correct later for MALA), but LL' is positive definite, while L is lower triangular." Thank you, we will clarify this.