We thank reviewers ***R1***, ***R2*** and ***R3*** for their constructive comments. We give here answers to the main ones.

R2 *If there is one sampling scheme [...] it must be explained better [...] this part should be very clean and clear.*

R1 *I also expect more details on the methodological modification over BH for the proposed sampling method.*

We agree, and we have expanded Section 3.3 and Appendix A.4 to shed light on (i) sampling continuous DPPs, and (ii) our contributions that led to cutting sampling time by orders of magnitude for our specific DPPs. The following is a short preview of the improved presentation of our contributions on the sampling procedure.

The original sampling algorithm for projection DPPs by Hough et al. (2006, Algorithm 18) works as follows. Consider the projection $\operatorname{DPP}(\omega(x)\mathrm{d}x, K_N)$ as defined in our Section 2.2, with $K_N(x,y) = \Phi(x)^{\intercal}\Phi(y)$ where $\Phi(x) \triangleq (\phi_0(x), \dots, \phi_{N-1}(x))$. This DPP has exactly N points, μ-almost surely. To get a valid sample $\{X_1, \dots, X_N\}$, it is enough to apply the chain rule to the vector $(X_1, \dots, X_N)$ and forget about the order. The vector has density

$$\frac{\det[K_N(x_p,x_n)]_{p,n=1}^N}{N!} \prod_{n=1}^N \omega(x_n) \overset{(5)}{=} \frac{K_N(x_1,x_1)}{N}\omega(x_1) \prod_{n=2}^N \frac{K_N(x_n,x_n) - \mathbf{K}_{n-1}(x_n)^{\intercal}\mathbf{K}_{n-1}^{-1}\mathbf{K}_{n-1}(x_n)}{N-(n-1)}\omega(x_n),$$

where the RHS is precisely the chain rule. The challenge is twofold. First, one must use an efficient way to sample exactly from the conditionals in the chain rule. Second, one must efficiently evaluate the kernel K_N (6). In our submission, we followed BH and used rejection sampling to sample the conditionals, using always the same proposal distribution $\omega_{\text{eq}}(x)\,\mathrm{d}x$ (A.12) and rejection bound (A.14). But, unlike BH, we computed $K_N(x,y)$ more efficiently by coupling the slicing feature of the Python language with the dedicated `scipy.special.eval_jacobi`. This carefully avoided redundant evaluations of orthogonal polynomials (OPs) in evaluating the multivariate kernel, which were a bottleneck. Since the submission, we further applied tricks familiar to MLers, and stored the feature vectors $\Phi(x_{1:n-1})$ to exploit the Gram structure when computing $\mathbf{K}_{n-1}(x_n) = \Phi(x_{1:n-1})^{\intercal}\Phi(x_n)$. Besides, we found out that using the marginal $N^{-1}K_N(x,x)\omega(x)\,\mathrm{d}x$ as a rejection sampling proposal allowed us to reduce the number of (costly) evaluations of the quadratic form $\mathbf{K}_{n-1}(x_n)^{\intercal}\mathbf{K}_{n-1}^{-1}\mathbf{K}_{n-1}(x_n)$. This new proposal can be sampled without too many OP evaluations since it is a mixture, where each mixture component involves only one OP and can be sampled using rejection sampling again, this time using the original proposal $\omega_{\text{eq}}(x)\,\mathrm{d}x$ of BH and the rejection bound (A.13). After making implementation improvements, getting one sample of a multivariate Jacobi ensemble with $N = 1000$ points in dimension $d = 2$ now takes less than a minute, compared to hours with the original implementation of BH (2016).

All these improvements hold for all continuous DPPs, and together resulted in the dramatic speedups that we observed. A more specific improvement for OP-based kernels and $d = 1$, has been to implement Theorem 2 of Killip & Nenciu (2004), which surprisingly reduces DPP sampling to diagonalizing a simple tridiagonal random matrix. Finally, as noted by ***R2***, the code we provided substantially helps; this code will be made public as a fully documented Python toolbox.

R1 *Theoretically, the authors shed light on the classical EZ method that utilizes DPP for numerical integration.*

Indeed, one of our contributions is to formally link EZ and DPPs. This needed to be done because (i) DPPs were formalized 15 years after EZ (1960), and (ii) while the theory of DPPs has been thoroughly investigated in the 2000s, the work of EZ had been mostly forgotten by then, probably because sampling looked an insurmountable obstacle. All recent DPP-based numerical integration works were seemingly unaware of EZ. Now that the link is made and the methods are "aligned", as ***R1*** writes, we can imagine several lines for future work, like borrowing DPP machinery to prove a central limit theorem for EZ, or further connecting EZ to Bayesian quadrature.

R1 *the method still does not seem to be applicable to practical tasks with high dimension or large sample size.*

As remarked by ***R3***, DPP-based Monte Carlo algorithms are new and further work is certainly needed to make them practical. However, we believe that the EZ estimator combined with fast DPP samplers is already of practical interest for high-dimensional integration of functions that are known to be sparse in some basis of L^2.

R1 *I cannot tell whether (a) Theorem 1 has new results or is a known result listed for a modern proof (b) Section 3.1 contains novel results (c) it is a novel idea to use orthonormal polynomials in EZ.*

(a) Theorem 1 is indeed a known result and we bring a modern formulation and a modern proof. As discussed above, our reformulation highlights the unknown fact that EZ is actually based on sampling from a DPP. We provide an efficient exact sampling procedure for the multivariate Jacobi ensemble with ideas that benefit the general case (b) Section 3.1 only recalls known results from BH (2016), to ease the later comparison with EZ. Comparing nonasymptotic and asymptotic variances brings insight, for instance. (c) It's not novel: orthogonal polynomials are quite common in approximation theory, where the EZ method comes from.

R1 *I expect (a) one experimental result for Sections 4.2-3. [...] (b) comparisons with i.i.d. Monte Carlo or MCMC [...]*

(a) Sections 4.2 and 4.3 indeed each contain an experiment. We designed these experiments to go beyond the toy illustration of BH (2016) and showcase the differences between EZ and BH. Extensive plots are to be found in Sections B2–6 of the appendix. (b) We will add the i.i.d. baseline, which will help visualize the faster rates of BH and EZ.

R1 *Is it possible to apply DPP to dynamics-based MCMC and particle-based variational inference methods?*

In principle yes, DPPs have the potential to yield generic variance reduction in importance sampling. The kernel needs to be carefully chosen, though, if one wants faster-than-Monte-Carlo rates like with BH or EZ.