Adaptive kernel density estimation proposal in gravitational wave data analysis

Mikel Falxa, Stanislav Babak, and Maude Le Jeune
Université Paris Cité, CNRS, Astroparticule et Cosmologie, F-75013 Paris, France

(Received 17 August 2022; accepted 22 December 2022; published 19 January 2023)

The Markov Chain Monte Carlo approach is frequently used within a Bayesian framework to sample the target posterior distribution. Its efficiency strongly depends on the proposal distribution used to build the chain. The best jump proposal is the one that closely resembles the unknown target distribution; therefore, we suggest an adaptive proposal distribution based on kernel density estimation (KDE). We group the model's parameters according to their correlation and build a KDE based on the already accepted points for each group. We update the KDE-based proposal until it stabilizes. We argue that such a proposal distribution could be efficient in applications where the data volume is increasing. We tested it on several astrophysical datasets (IPTA and LISA) and have shown that, in some cases, the KDE-based proposal also helps to reduce the chains' autocorrelation length. The efficiency of this proposal distribution is reduced in case of strong correlations between a large group of parameters.

DOI: 10.1103/PhysRevD.107.022008

I. INTRODUCTION

We live in the era of large physics and astrophysics projects and often have to deal with large and complex datasets. The data analysis usually requires large computing facilities, and a single computation could sometimes last for weeks. Optimizing the analysis techniques and pipelines is a key challenge of the data science associated with all large (astro)physical experiments.

Nowadays, it is quite common to use the Bayesian framework for analyzing the data when we have a parametrized data model (or several competing models) describing it. In this approach, we treat all parameters as random variables with some prior probability based either on some physical principles or informed from the previous independent experiments. We use the observations (measurements) to refine our prior knowledge and infer a posterior probability distribution function for the parameters of a model and/or perform a selection among several models. We often have to deal with a multidimensional parameter space with a nontrivial likelihood function that can be evaluated only numerically. One of the most used tools to perform numerical sampling from a target probability distribution is the Markov Chain Monte Carlo (MCMC). Building a Markov chain that represents the desired posterior distribution requires two key ingredients: (i) a jump proposal distribution suggesting how to choose point $\vec{X}_{i+1}$ given the last point in the chain $\vec{X}_i$; (ii) the detailed balance which ensures the reversibility of the chain. One of the most successful and frequently used jump proposals is parallel tempering (see, for example, [1] and Appendix C). Understanding the properties of the signal and the likelihood surface could be used to design a custom proposal distribution suitable for a particular problem. Custom-made proposals increase the efficiency (acceptance rate and exploration abilities) of MCMC.

Here we suggest a generic proposal distribution based on the kernel density estimation (KDE). The primary purpose of KDE is to build an analytic approximation of a probability density function represented by a set of samples. KDE consists of a collection of individual kernel functions (often normal distributions) with specific widths (bandwidth) attached to each sample point, interpolating the probability density between samples. The idea of using KDE as a proposal is not new and was already partially explored in [2,3]. The novelty of the work presented in this paper is in the particular implementation of the KDE and its embedding into a sampler. Even though the proposed method is very generic, we will discuss its application only in the gravitational waves (GWs) data analysis.

Let us summarize the key points of the KDE-based jump proposal:

(i) KDE is used together with other jump proposals to build a Markov chain. We assume an adaptive approach where we use the data accumulated in a chain to rebuild (update) the KDE regularly. We repeat the adjustments of KDE until the convergence criterium based on the Kulback-Leibler divergence is satisfied.

(ii) In order to build the KDE, we split all parameters into several groups, where parameters in each group show evidence of mutual correlation. The performance of the KDE-based jump proposal drops significantly if the dimensionality of a group is larger than 5. If possible, the KDE distribution should be built on the low-dimension parameter subspace.

(iii) We have built a KDE with an optimized bandwidth based on the distribution of the samples provided at the input (see Sec. II B).

Note that the updates of KDE based on the accumulated samples in the chain break the "Markovian" properties of the chain. The "memorylessness" of the Markov process assumes that the next point in the chain depends only on the previous one, whereas the regular updates of the KDE-based jump proposal bear memory of all the samples which went into its construction. One should either stop KDE updates at some point or identify the instance where KDE does not change appreciably and discard a part of the chain with evolving KDE. In the case of uninterrupted updates, the chain is asymptotically Markovian [4]. We give a detailed description of the implementation in Secs. II and III.

We have implemented the KDE-based jump proposal in a particular sampler.[1] We give a detailed description of this sampler in Appendix C. The main feature of this sampler is that it runs several chains either entirely independently or as parallel tempering. We compute the Gelman-Rubin ratio [5] in the multichain implementation to monitor the convergence of MCMC.

We assess the performance of the suggested jump proposal in two applications to GW data analysis. In the first one, we analyse the data combined by International Pulsar Timing Array (IPTA) Collaboration searching for a continuous GW signal in the nano-Hz band. As the second dataset, we use simulated LISA data publicly available through LISA Data Challenge (2a) portal. We use the KDE-based proposal distribution to infer the parameters of six Galactic white dwarf binaries. We present the performance of our jump proposal for those two data analysis problems in Sec. IV; in particular, we show that the KDE-based proposal distribution reduces the autocorrelation length while keeping a high acceptance rate. We conclude the paper with a discussion on the limitation and possible extension of our method in Sec. V.

II. KERNEL DENSITY ESTIMATION

This rather short section describes our particular way of building the KDE. We start with a short introduction to a KDE and then give details of its bandwidth optimization.

A. Brief introduction

KDE is a nonparametric method used to estimate a probability density function (PDF) based on a finite set of sample points [6,7]. It is a smooth alternative to a histogram. The advantage of KDE is that it uses no binning and gives a continuous function interpolating (and extrapolating) across the whole parameter space. For a D-dimensional dataset $\{\vec{X}\}$ of size N and kernel $K(\mathbf{x},\vec{h})$, we have our KDE $\hat{f}(\mathbf{x},\vec{h})$,

$$\hat{f}(\mathbf{x},\vec{h}) = \frac{1}{N}\sum_{a=0}^{N-1} K(\mathbf{x}-\vec{X}_a,\vec{h}), \tag{2.1}$$

with parameter $\vec{h}$ specifying the bandwidth of the kernel. We use Latin subscripts from the first half of the alphabet to enumerate the samples in the set. The main idea is to sum smooth kernel functions of $\mathbf{x}$ centred on each sample (input) data point $\vec{X}_a$. The overlaps between neighboring kernels will add up, shaping the PDF for the set of samples $\{\vec{X}\}$. The choice of the kernel is arbitrary, and we choose to work with a Gaussian kernel of the form

$$K_g(\mathbf{x}-\vec{X}_a,\vec{h}) = \prod_{i=1}^{d} \frac{\exp\{-\frac{1}{2}\frac{|\mathbf{x}-\vec{X}_a|_i^2}{h_i^2}\}}{\sqrt{2\pi}h_i}, \tag{2.2}$$

where the h_i is the local bandwidth corresponding to the ith parameter $|\mathbf{x}-\vec{X}_a|_i$, and d is the dimensionality of the parameter space. We use Latin letters from the second half of the alphabet to enumerate particular parameters, and the vector notation corresponds to a vector in the parameter space.

B. Optimal bandwidth

A KDE has one free parameter which we want to tune, the bandwidth $\vec{h}$. Its value should be adapted to the samples representing the probability distribution we want to approximate. There is no direct way of estimating it, and we use an optimization method [8] based on the minimisation of the mean squared error (MSE) ϵ^2 with respect to $\vec{h}$,

$$\epsilon^2 = \int d\mathbf{x}(\hat{f}(\mathbf{x},\vec{h}) - f(\mathbf{x}))^2, \tag{2.3}$$

$$\frac{\partial\epsilon^2}{\partial\vec{h}} = 0, \tag{2.4}$$

where $f(\mathbf{x})$ is the true PDF that we want to approximate with the KDE. The numerical evaluation of this integral is given in Appendix B.

Instead of using a global bandwidth $\vec{h}$, we can define a local bandwidth $\vec{h}_a$ for each kernel $K(\mathbf{x}-\vec{X}_a,\vec{h}_a)$ [9]. In that case, our KDE $\hat{f}(\mathbf{x},\vec{h})$ is

$$\hat{f}(\mathbf{x},\vec{h}) = \frac{1}{N}\sum_{a=0}^{N-1} K(\mathbf{x}-\vec{X}_a,\vec{h}_a). \tag{2.5}$$

Intuitively we expect the local bandwidth $\vec{h}_a$ to be scaled according to the local density of points. Indeed, the bandwidth is chosen so that it is narrow in the regions of parameter space where the samples are most dense, and it is broad

[1] https://gitlab.in2p3.fr/lisa-apc/m3c2.

where we have fewer samples. This choice ensures good interpolation and overlap between kernels, particularly in high-dimensional problems with sparse sample points. The solution of Eq. (2.4) defines the optimal local bandwidth. The global bandwidth (if needed) could be defined as an average over all local values (by setting the input parameter `global bw = True`). In Appendix B, we have shown that each local bandwidth $\vec{h}_a = [h_{a,1}\, h_{a,2} \ldots]$ of the kernel centered on the point $\vec{X}_a$ can be approximately found as a solution of a system of linear equations using k nearest neighboring points $\vec{X}_b$

$$\mathbf{A}(\vec{X}_b)\begin{bmatrix} \frac{1}{h_{a,1}^2} \\ \frac{1}{h_{a,2}^2} \\ \vdots \end{bmatrix} = \vec{B}(k). \tag{2.6}$$

The matrix $\mathbf{A}$ and vector $\vec{B}$ are given explicitly in Appendix B [Eq. (B6)], where we provide a detailed description of the method. They depend on the position of nearest neighbors, which are the k_{near} points contained in a hypercube centered on the point $\vec{X}_a$ in the parameter space. The edge ΔX_i of the hypercube for each parameter X_i is defined as

$$\Delta X_i = (\max_{\{\vec{X}\}} X_i - \min_{\{\vec{X}\}} X_i)/s, \tag{2.7}$$

where maximization and minimization are performed over the set of input samples and s is a scaling parameter we call "`adapt scale`".

A hypercube might contain no points (besides the central). In that case, we cannot compute the local bandwidth and the point is discarded. Its bandwidth is later set to the global bandwidth as defined above. In case of high dimensionality and if the data points are very sparse, we change parameter s iteratively, decreasing it by a factor of 2 until we find nonempty hypercubes. However, if this happens, the bandwidth evaluation will probably be flawed, and there is not much we can do about it except use bigger datasets with more sample points. Often the number of additional points needed to cover all "holes" could be huge, incurring unmanageable computational costs. That is why good parameter grouping is essential; it reduces dimensionality without losing correlated features in the data. This will be the main subject of the next section. s is the scaling parameter "`adapt scale`".

III. METHOD

The main idea is to build a KDE for a given D-dimensional set of sample points $\{\vec{X}\}$. However, for a high-dimensional KDE, we are strongly affected by the "curse of dimensionality" because the sample sets are often limited in size, leaving undercovered regions of the parameter space. In addition, the efficiency of KDE degrades if there are too many points since we place the kernel on top of each sample. For that reason, we will split a D-dimensional parameter space into several low-dimensional subspaces, grouping the most correlated parameters. We assume that subgroups are not correlated and build the KDE for each of them $\hat{f}_\alpha(\mathbf{x}_\alpha, \vec{h}_\alpha)$, so the total KDE is the product of the low-dimensional KDEs,

$$\hat{F}(\mathbf{x}, \vec{h}) = \prod_\alpha \hat{f}_\alpha(\mathbf{x}_\alpha, \vec{h}_\alpha), \tag{3.1}$$

where the Greek indices enumerate the subgroups of parameters. Forming these subgroups relies on assessing the correlation between parameters based on the provided set of samples $\{\vec{X}\}$, which is the main subject for the following subsection.

A. Parameter grouping

We want to split the parameters into several subgroups for a D-dimensional dataset $\{\vec{X}\}$. Each subgroup will contain correlated parameters, while parameters from different subgroups will be uncorrelated. We could use a covariance matrix to identify correlations; however, it implicitly assumes Gaussian distribution and cannot account for complex 2D structures between pairs of parameters. Instead, we use a method based on the Jensen-Shannon divergence (JSD) [10]. JSD, similarly to the Kullback-Leibler (KL) divergence, measures the similarity between two distributions, but with the advantage of being symmetric and bound $0 \le \mathrm{JSD} \le \ln 2$. For each pair of parameters (X_i, X_j) with the joined probability distribution $p(X_i, X_j)$ we compute

$$0 \le \mathrm{JSD}(p(X_i, X_j) || p(X_i)p(X_j)) \le \ln 2. \tag{3.2}$$

The distributions $p(X_i), p(X_j)$ are obtained by using a provided set of samples and shuffling only parameters X_i (or X_j) while keeping other parameters fixed. This preserves a one-dimensional marginalized distribution for each parameter but destroys any correlations (see Fig. 1). If the JSD is low, the shuffling did not affect the dataset, and the parameters did not exhibit correlation (a product of marginalized 1D distributions well approximates the joint distribution). On the other hand, if the JSD is large, the shuffling did destroy the correlation, and the parameters should be grouped together.

We define a JSD threshold (usually chosen to be 0.1) for which we consider that two parameters are correlated. Starting from one parameter, we iterate the process to extract all correlated pairs.[2] Once no additional correlated

[2]In this paper, the dataset that we use to test correlation and build the KDEs are the chains obtained from previous MCMC runs carrying the correlations between parameters.

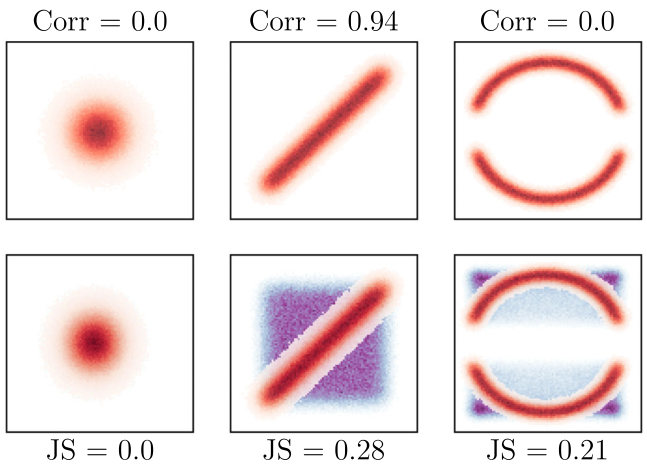


FIG. 1. We plot three examples of datasets where we have on the left no correlation, in the middle linear correlation and on the right more elaborate features. On the top panels, we have the corresponding values of the correlation coefficient based on a simple evaluation of the covariance matrix. We see that it excels at finding the linear correlation but completely fails with the right panel features. In the bottom panels, we have the same three datasets in red with their corresponding shuffled version that destroys correlations in blue. While the left panel remains unchanged, the others are affected, and the JS divergence captures it.

parameter is found, we take the union of all correlated pairs of parameters to form a sub-group. This process is illustrated in Fig. 2.

In the case of the multimodality of the PDF that we try to reproduce with the KDE, we implemented an additional (optional) feature; clustering samples before building KDE. This feature is especially welcome when the modes are separated by very low probability valleys. We cluster the samples (using the k-means method [11]) and apply the KDE building approach described above to each mode. This does not change the fundamental structure of the KDE, but it helps the bandwidth adaptation.

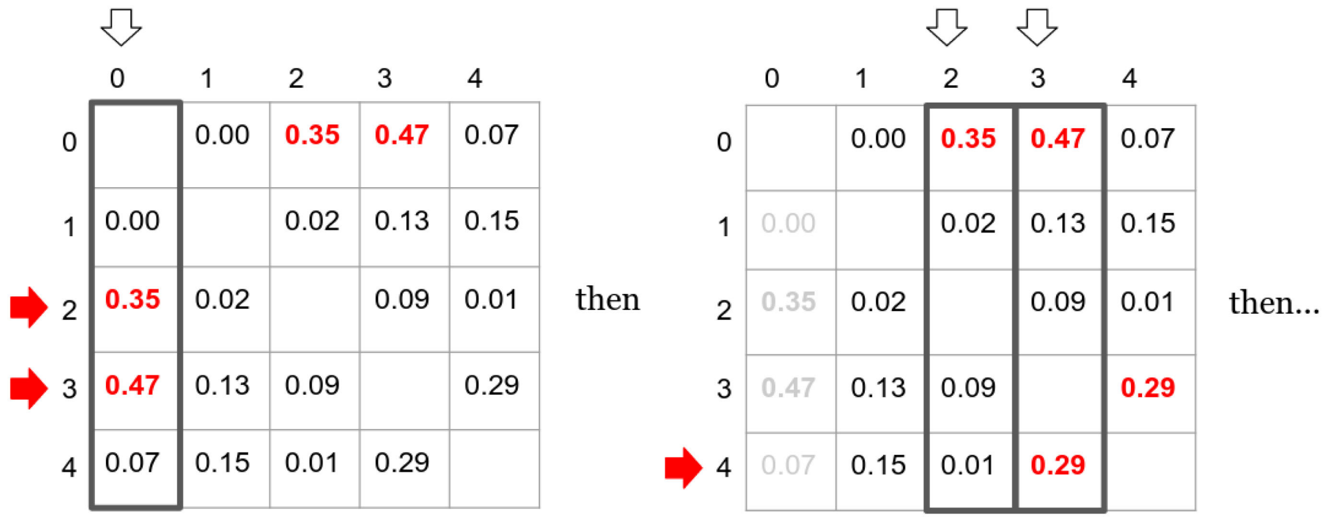
B. KDE: Turning posterior into a proposal distribution

It is desirable in several applications to use posterior points inferred for some parameters in another investigation. A KDE built from the posterior samples inferred from the data "A" can be used as a prior probability to investigate the data "B" or as a jump proposal probability distribution. Let us give several concrete examples of its use:

(i) The data inferred from electromagnetic observations in the form of samples is used as a prior for the GW experiment. In this case, we can either build a joint likelihood or, alternatively, build a KDE on the external posteriors and use it as a prior while analyzing the GW data.

(ii) In pulsar timing array (PTA) data analysis, we often first investigate the data acquired for each pulsar and try to build a noise model. Later this pulsar and associated noise model is plugged into the array of pulsars to search for a GW signal. It is proven to boost the efficiency of the GW search significantly if we use posteriors for the noise model inferred in the first step as a proposal distribution in the global fit later on.

(iii) Often, the data is taken continuously, and we want to analyze it "on the fly"; that is true for GW data



	0	1	2	3	4
0		0.00	0.35	0.47	0.07
1	0.00		0.02	0.13	0.15
2	0.35	0.02		0.09	0.01
3	0.47	0.13	0.09		0.29
4	0.07	0.15	0.01	0.29	

	0	1	2	3	4
0		0.00	0.35	0.47	0.07
1	0.00		0.02	0.13	0.15
2	0.35	0.02		0.09	0.01
3	0.47	0.13	0.09		0.29
4	0.07	0.15	0.01	0.29	

FIG. 2. Illustration of the parameter grouping process using a JSD matrix. For this example, the JSD threshold is set to 0.25; hence the parameter subgroups will be [0, 2, 3, 4] and [1]. Starting from parameter 0, we find that JSD for parameters 2 and 3 are above the threshold, so they are both correlated with parameter 0. Then we check for parameters 2 and 3 and find that 4 is correlated with 3 while 2 sees no additional correlation. The last step would have been to check for 4 and find no additional correlations. Therefore, 0, 2, 3, and 4 will form a subgroup of correlated parameters. The parameter "1" is the last parameter that does not correlate with others (according to the adopted threshold) and will be a subgroup on its own.

analysis. In this case, we want to increment the data with a certain cadence while using information about the sources acquired from the analysis of the past data. One possibility is to turn again posterior built from the analysis of, say, the first half a year of data into a prior in the analysis of the second half year of data.

(iv) The product-space sampling method (see Refs. [12,13]) gives a practical suggestion on how to compute the Bayes factor comparing several models without computing the evidence for each model. In this approach, we introduce a hyper-parameter indexing the models and jump in this parameter (say, within MCMC) which corresponds to jumping between the models. For this method to be robust, the exploration within each model must be very efficient, otherwise it will lead to very long and poorly converging runs or spurious results. If we have posteriors for each (or some) model, we can turn them into a proposal distribution and use them in the hypermodel exploration.

Grouping correlated parameters in building a KDE-based proposal allows us also to make jumps inside a particular (or several) subgroup(s) while keeping other parameters fixed (Gibbs-like sampling, see Ref. [14]). The subgroups for each iteration are chosen randomly with equal probability. In this way, we ensure that all subgroups are visited evenly (on average), and we explore the entire parameter space by doing low-dimensional jumps. Let us denote the number of sub-KDEs that are used for each jump $n_{\rm kde}$, then the proposal probability is

$$\hat{F}_{n_{\rm kde}}(\mathbf{x}, \vec{h}) = \prod_{\alpha}^{n_{\rm kde}} \hat{f}_{\alpha}(\mathbf{x}_{\alpha}, \vec{h}_{\alpha}), \tag{3.3}$$

where the subscript in $\mathbf{x}_\alpha$ implies that we vary only parameters that belong to that (α) subgroup. This probability is used to balance the chain in the Metropolis-Hastings step of the MCMC algorithm [15]. Choosing a point from a given sub-KDE $\hat{f}_{\alpha}(\mathbf{x}_{\alpha}, \vec{h}_{\alpha})$ is done by drawing a point from the randomly chosen kernel $K(\mathbf{x} - \vec{X}_a, h_a)$ of $\hat{f}_{\alpha}(\mathbf{x}_{\alpha}, \vec{h}_{\alpha})$ centered on $\vec{X}_a$,

$$\vec{X}^*_b \to \vec{X}_a + \mathcal{N}(\vec{0}, h_a), \tag{3.4}$$

where $\mathcal{N}(\vec{0}, h_a)$ is a normally distributed random variable with $\vec{0}$ mean and covariance matrix $\mathrm{diag}(h_a)$ that is the bandwidth of kernel $K(\mathbf{x} - \vec{X}_a, h_a)$. For a random set of $n_{\rm kde}$ sub-KDEs, the newly proposed point $\vec{X}^*$ is the union of parameters from each subgroup $\vec{X}^*_b$,

$$\vec{X}^* = \bigcup_b \vec{X}^*_b. \tag{3.5}$$

C. Adaptive proposal

If we do not have samples from the previous investigations, we can still build a KDE-based proposal using the points accepted by a running MCMC. There are several caveats which need to be considered: (i) during the burn-in and even sometime after, the distribution of the accepted points is quite unstable, and that will reflect on the KDE; (ii) we are breaking the rules of MCMC; the chain is only asymptotically Markovian [16]. The last point refers to the fact that the Markov chain we build should be memoryless as the next point in the chain depends only on the current one. Using the entire (or part of) chain to build a proposal distribution breaks that rule, so we should stop updates and discard the chain up to that point. The rest of this subsection details the practical implementation of the KDE adaptation.

We select only a subset of total N samples to build a KDE from the chain's current state. In particular, we take n_s uniformly spaced points, from burn-in to the last point, as illustrated in Fig. 3. The burn-in is defined as a fraction, q (always fixed to 0.25), of the total chain length N and those points are dismissed (hence, $n_s = (1-q)N$). As the chain evolves, the burn-in and post-burn-in increase in length and since we keep n_s fixed, the space Δn between the selected samples grows. An extensive burn-in period ensures the stationarity of the remaining points. We also expect that the KDE quality improves with an increase in Δn because of the reduced correlation in the samples taken for its building, and that is true until Δn reaches a typical autocorrelation length of the chain [17].

We rebuild a new KDE after each $N_{\rm adapt}$ iterations (jumps) of the chain. We want to track the evolution of the KDE and stop updating when it has reached stability (that could be another indicator of the burn-in phase). We compare the new (rebuilt) KDE $\hat{F}_1(\mathbf{x}, \vec{h}_1)$ with the old $\hat{F}_0(\mathbf{x}, \vec{h}_0)$ by computing the KL divergence [18],

$$\mathrm{KL}(\hat{F}_0 || \hat{F}_1) = \int d\mathbf{x} \hat{F}_0(\mathbf{x}) (\log \hat{F}_0(\mathbf{x}) - \log \hat{F}_1(\mathbf{x})). \tag{3.6}$$

This integral could be approximated as

$$\mathrm{KL}(\hat{F}_0 || \hat{F}_1) \simeq \frac{1}{n_s} \sum_a (\log \hat{F}_0(\vec{X}_{0a}, \vec{h_0}) - \log \hat{F}_1(\vec{X}_{0a}, \vec{h_1})), \tag{3.7}$$

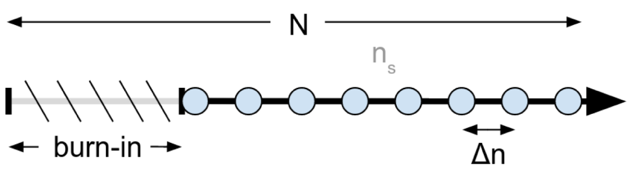


FIG. 3. For a chain of current length N, we get rid of the burn-in, then we extract n_s linearly-spaced samples from the remaining fraction of the chain. These n_s points are used to build the KDE.

where $\{\vec{X}_0\}$ is the set of n_s samples used to build $\hat{F}_0(\mathbf{x}, \vec{h}_0)$. This gives a measure of change in KDE between successive updates, ΔKL. We stop updating KDE in order to preserve the ergodicity of the process [19] as soon as the stability criterion

$$\frac{|\langle \Delta \mathrm{KL} \rangle|}{\sqrt{\langle \mathrm{KL}^2 \rangle}} < 5\% \tag{3.8}$$

is satisfied. The angular brackets denote the averaging over the last five updates and we demand that the average change in KL is small compared to the average KL values. ΔKL can be negative or positive, depending on the evolution of KL. If KL does not converge to a specific value, this condition ensures that it is at least oscillating around a mean value.

This procedure could also serve as an indicator of the stationarity of the chain. The sampled distribution could be considered stationary if: (i) the algorithm that groups the correlated parameters returns the same subgroups, and (ii) the KL divergence measuring changes in the KDE between successive updates is small.

IV. RESULTS

We consider two datasets and perform search/parameter estimation using MCMC with a KDE-based proposal distribution.

In the first application, we consider a dataset from the International PTA Collaboration and perform the noise analysis for each pulsar in the array [20]. The likelihood is expected to be quite broad and unimodal, but the dimensionality of the parameter space is large as well as its overall volume.

In the second application, we work with the simulated LISA data and search/characterize small bandwidths with several Galactic white-dwarf binaries. The likelihood in this case, has a more complex structure with a quite strong correlation between parameters.

We quantify the performance of the KDE-based proposal by evaluating the following quantities:

(i) Closeness between the KDE and the actual distribution using KL divergence.[3]
(ii) The acceptance rate when the KDE is used as a proposal distribution with MCMC sampler.
(iii) The autocorrelation length [17] of the MCMC chain when the KDE is used to propose jumps (mixing of the chain).

[3]Because we do not know the true distribution, we have to use histograms to evaluate KL. The formula for the binned KL is given in Appendix A.

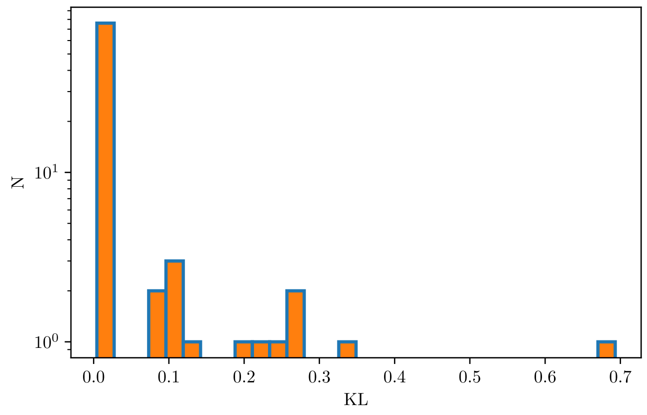


FIG. 4. Distribution of the KL divergence for all sub-KDEs. 1D sub-KDEs are of best quality.

A. IPTA dataset

We build the KDE using $n_s = 10000$ samples from the chains generated by previous MCMC runs. We adopt the following choice of parameters for generating KDE:

(i) `js threshold = 0.1`
(ii) `adapt scale = 10`
(iii) `use kmeans = False`
(iv) `global bw = True`
(v) `n kde = 1`

Within a total of 104 parameters, the grouping algorithm finds **76** one-dimensional subgroups, **12** two-dimensional subgroups and **1** four-dimensional subgroup. For each of these subgroups we calculate the KL divergence $\mathrm{KL}(p_\alpha(\mathbf{x}_\alpha)||\hat{f}_\alpha(\mathbf{x}_\alpha, \vec{h}_\alpha))$ of the true PDF of the subgroup $p_\alpha(\mathbf{x}_\alpha)$ against the corresponding sub-KDE $\hat{f}_\alpha(\mathbf{x}_\alpha, \vec{h}_\alpha)$. A good sub-KDE should give a $\mathrm{KL}(p_\alpha(\mathbf{x}_\alpha)||\hat{f}_\alpha(\mathbf{x}_\alpha, \vec{h}_\alpha))$ that is close to 0. Because the number of parameters is large, we show the histogram plot in Fig. 4 depicting KL for each subgroup.

The impact of the dimensionality of the sub-KDE on KL can be clearly seen in Fig. 4. Close to 0, we have 76 one-dimensional sub-KDEs, between 0.1 and 0.4, we have 12 two-dimensional sub-KDEs, and the four-dimensional sub-KDE has KL ≈ 0.7. As discussed in the previous section, increasing dimensionality implies sparse data samples, so we expect KL values to rise because the KDE might fail to interpolate the PDF correctly between neighboring points, producing holes in the distribution. Computing KL allows us to quantify this effect and assess the quality of the KDE that may not be obvious by just eyeballing (see Fig. 5).

Next, we analyzed the IPTA data using MCMC sampler [21], and ENTERPRISE [22] package for computing the likelihood function. We chose to use two jump proposals single component adaptive metropolis (SCAM) [23] and differential evolution (DE) [24,25] to compare to KDE. We search for a continuous gravitational wave signal while fitting for pulsar noise parameters [26]. The KDE for the noise parameters has been built using posterior samples

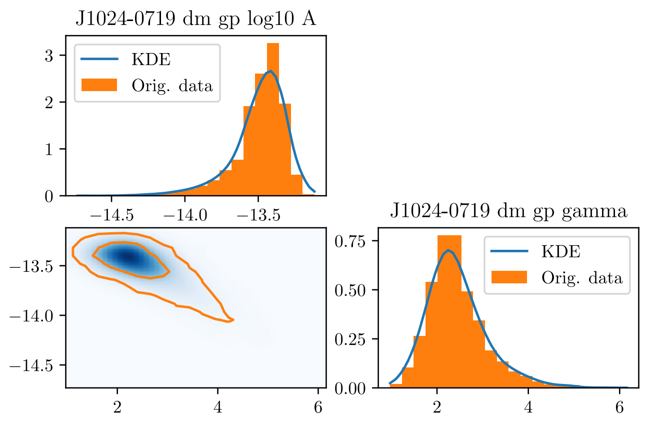


FIG. 5. Two-dimensional corner plot of binned original dataset against smooth KDE plot. KDE is blue, original is orange. The contour levels for the original dataset in the bottom left panel are [0.1, 0.95]. For this sub-KDE, KL $\simeq 0.11$.

obtained from the preceding single-pulsar analysis. We compare three different runs:

(i) using a KDE-based + default jump proposals SCAM and DE (labeled as "KDE");
(ii) using binned empirical distributions + default jump proposals SCAM and DE (labeled as "Binned");
(iii) using only default jump proposals SCAM and DE (labeled as "None").

The binned empirical distributions are essentially two-dimensional histograms based on the same posterior samples as used in building KDE [27]. Those carry a similar spirit to KDE, being a pairwise approximation to marginalized posterior, but at the same time, are fundamentally different from the KDE because the KDE is a continuous function in space that interpolates between the sample points using a smoothing kernel. In addition, the KDE-based jump proposal makes grouping based on the parameter correlation that could lead to more than two-dimensional group (see Fig. 4).

Tables I and II, compare the acceptance rate of different proposals in two independent runs. One can see that KDE has the highest acceptance rate due to intelligent parameter grouping and interpolation between the samples incorporated in the KDE-based proposal. We have tested jumps in only one or simultaneously in several (n_{kde}) subgroups of parameters in each iteration. Increasing n_{kde} leads to the higher dimensionality of the jumps and substantially impacts the acceptance rate, as shown in Fig. 6 and Table III. The best results are achieved if we perform jumps in one subgroup at a time; the acceptance rate decreases exponentially with n_{kde}.

A high acceptance rate is not necessarily a sign of a good proposal, as it has to be paired with a low-autocorrelation length. For each run, we compute the autocorrelation lengths for all parameters and compare the maximum, minimum and mean autocorrelation lengths. The number of independent samples in the chain is defined as the accumulated number of samples divided by the autocorrelation length. Only independent samples tell us the actual length of the chain; therefore, having low autocorrelation implies the high efficiency of sampling the target distribution. Results are presented in Table IV. KDE performs very well for IPTA data. The minimum autocorrelation length does not seem to be affected much by the choice of the jump proposal, but the maximum is reduced by a factor of 2 when using the KDE. Reducing the maximum length is the most important because thinning the chain by this factor ensures that the samples are independent for all parameters. The mean autocorrelation length is just an indicator of the average performance of the proposal.

As for the acceptance rate, we check the influence of n_{kde} on the autocorrelation length. Results are given in Table V. For high values of n_{kde}, even though the acceptance rate decreases, the autocorrelation drops too and mixing improves. However, this result should be taken with

TABLE I. Acceptance for various proposals (KDE run).

	KDE	SCAM	DE
Acceptance rate	0.57	0.39	0.43

TABLE II. Acceptance for various proposals (binned proposal run).

	Binned	SCAM	DE
Acceptance rate	0.47	0.41	0.43

TABLE III. Acceptance for different n_{kde}.

n_{kde}	KDE	SCAM	DE
1	0.47	0.41	0.43
5	0.22	0.36	0.41
10	0.08	0.35	0.40

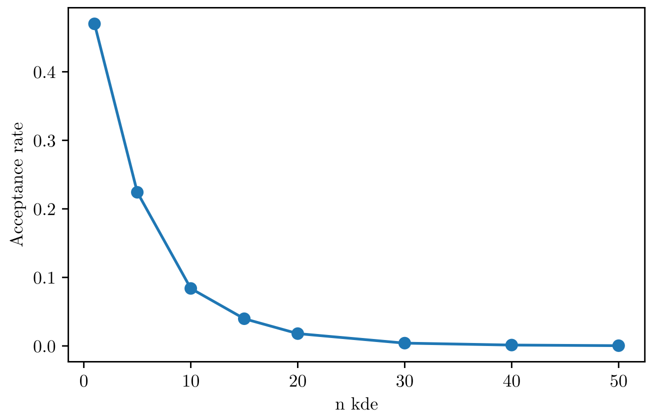


FIG. 6. Acceptance rate for several values of n_{kde}.

TABLE IV. The maximum, minimum and mean autocorrelation lengths for three runs.

	Max	Min	Mean
KDE	578	27	113
Binned	1386	29	160
None	1032	25	208

TABLE V. The maximum, minimum, and mean autocorrelation lengths for different n_{kde}.

n_{kde}	Max	Min	Mean
1	578	27	113
5	386	25	79
10	395	28	95

caution; using high n_{kde} could lead to a very low acceptance point, as shown in Fig. 6. In Fig. 7 we show the autocorrelation length as a function of n_{kde} and it indicates that the optimal number is around $n_{\mathrm{kde}} = 5$ with an acceptance rate of 0.22 that is very close to the expected optimal acceptance rate of 0.234 [28]. Based on these results, we recommended working with values of n_{kde} between 1 and 5, especially when the dimensionality of the parameter space is high, like in this IPTA example.

B. LISA dataset

Now we turn our attention to the simulated LISA data; in particular, we use the "Sangria" dataset. It is one year long and contains about a dozen merging massive black hole binaries and about 30 million Galactic binaries https://lisa-ldc.lal.in2p3.fr/. Here we are interested in Galactic binaries in the very narrow frequency range around 4 mHz, and we have removed all merging black holes. We have detected six sources in that frequency interval and we use KDE-based proposal together SCAM and DE.[4] This time we use a homemade sampler $M3C2$ (footnote 1), precisely a parallel tempering version of it. This sampler uses Metropolis-Hastings acceptance-rejection step, as well as slice sampling [29]. We describe this sampler in detail in Appendix C.

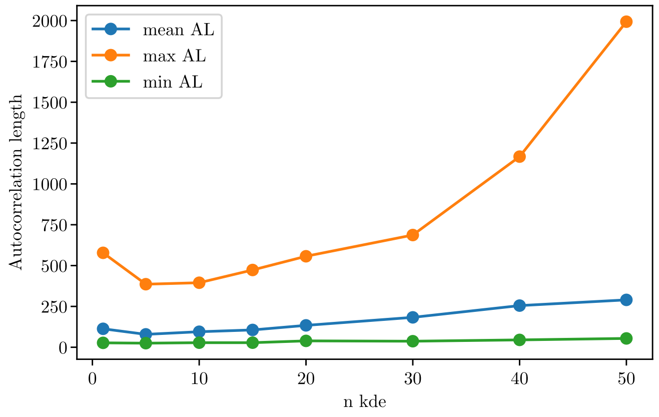


FIG. 7. Mean, maximum, and minimum autocorrelation lengths for several values of n_{kde}.

Each Galactic binary is characterized by eight parameters, so we have 48 parameters in total [30]. We expect some parameters (like amplitude and the orbital inclination angle) to correlate for each source, and, in addition, some parameters could correlate between the sources. Here we try to build a proposal on-the-fly. The likelihood surface for this problem is rather complex, having many well-separated maxima (the reason for using parallel tempering). We will build KDE as we accumulate samples by adapting the KDE proposal with the rate of every 5000 samples and using $n_s = 5000$ samples for each chain. Besides the KDE, we also use the SCAM jump proposal and slice sampling. Note that the SCAM and slice have a different nature. SCAM is just another jump proposal that uses the principal direction of the covariance matrix, and then the Metropolis-Hastings ratio is computed to decide on the acceptance/rejection. Slice is not based on the Metropolis-Hastings algorithm but an entirely different way of sampling the posterior. The combination of Metropolis-Hastings and slice samplers already significantly reduces the autocorrelation of the accepted points.

First, we consider the convergence of KDE adaptation. During the burn-in stage, the KDE changes quite violently. The correlation between parameters is quite unstable, leading to the fluctuation in how parameters are grouped and the number of subgroups. As burn-in proceeds, we keep track of the grouping and fix the splitting in subgroups as soon as it stabilizes (when the same grouping appears at least five times). Once we have fixed subgroups, we check the KL divergence between the subsequent updates of KDE (as described above). The results of KDE adaptation are presented in Fig. 8. The consistent grouping of parameters was reached after 26 updates, as indicated by a dashed red line. Then we compute KL after each KDE's update and fix it once the condition (3.8) is met (see the right panel of Fig. 8). Once KDE is fixed, we dismiss all accumulated points and perform the actual sampling.

As a next step, we want to check the acceptance rate of the KDE-based proposal and compare it to SCAM and slice. Note that the slice is not based on the Metropolis-Hastings acceptance/rejection algorithm; however, we can still introduce an effective acceptance rate as a ratio of the total number of calls to the total number of likelihood evaluations used by the slice.

[4]The DE introduced in [25] is using population MCMC. Here we choose to use it on a single chain in the spirit described in the snooker proposal in [24].

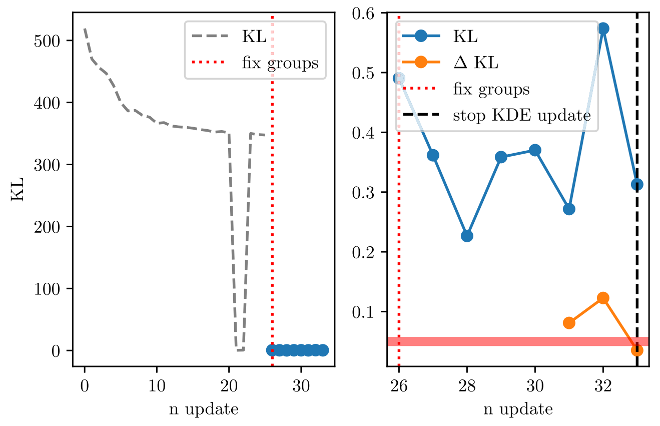


FIG. 8. Evolution of KL and ΔKL. Left panel, the evolution of KL before fixing parameter groups. Right panel, the evolution of KL after fixing parameter groups, we start computing ΔKL 5 updates after the grouping was fixed. The thick red line indicates the ΔKL threshold level of 5% below which we reach convergence.

We consider two cases for the subgroup jumps $n_{\mathrm{kde}} = 1$ and $n_{\mathrm{kde}} = 5$. Figure 9 compares acceptance rate for $n_{\mathrm{kde}} = 1$. One can see that it stabilizes around 0.31 very fast, and, despite that, it is lower than what we had for the PTA application as it is a very decent acceptance rate. SCAM has a similar acceptance (but usually longer autocorrelation length), while slice is worse by a factor of 10 (though it usually has low autocorrelation length).

Figure 10 compares acceptance rate for $n_{\mathrm{kde}} = 5$. An increase in the dimensionality of the jumps has a drastic effect on the acceptance rate, its value drops to about 0.015. It is also interesting to compare the behavior of SCAM and slice for two runs. Slice shows very stable/consistent results, while SCAM has significant fluctuations while preserving the trend. The two-dimensional KDE jumps seem to be the best option in this application.

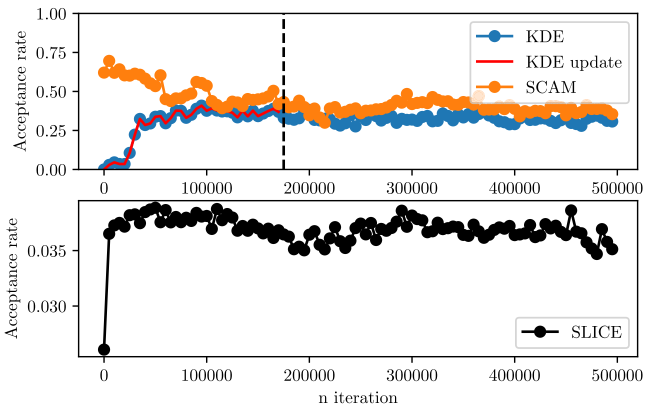


FIG. 9. Acceptance rate of all proposals for $n_{\mathrm{kde}} = 1$. KDE and SCAM are on top panel, SLICE on bottom panel. Red plot and black dashed line shows where KDE finally converged and stopped updating.

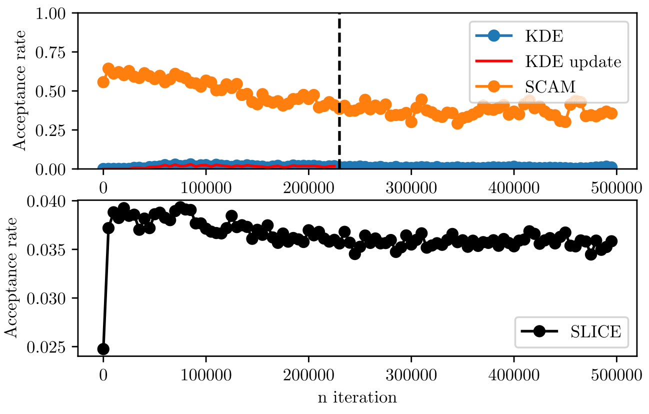


FIG. 10. Acceptance rate of all proposals for $n_{\mathrm{kde}} = 5$. KDE and SCAM are on top panel, SLICE on bottom panel. Red plot and black-dashed line shows where KDE finally converged and stopped updating.

Next, we check the autocorrelation length when we run with and without a KDE-based jump proposal. We restrict ourselves with the case $n_{\mathrm{kde}} = 1$ since $n_{\mathrm{kde}} = 5$ has a very poor acceptance rate. Figure 11 compares the maximum and mean autocorrelation of two runs. We observe that the mean value is slightly (about 17%) lower when we include the KDE-based jump proposal and the maximum length remains the same. As we have already mentioned, mixing SCAM with Metropolis-Hastings and slice steps does reduce the autocorrelation already (compared to the PTA example where we did not use slice sampling). In addition, we use a parallel tempering algorithm, where the hot chains could be seen as yet another jump proposal. Overall, KDE does not add much to the already reduced autocorrelation run in the current analysis.

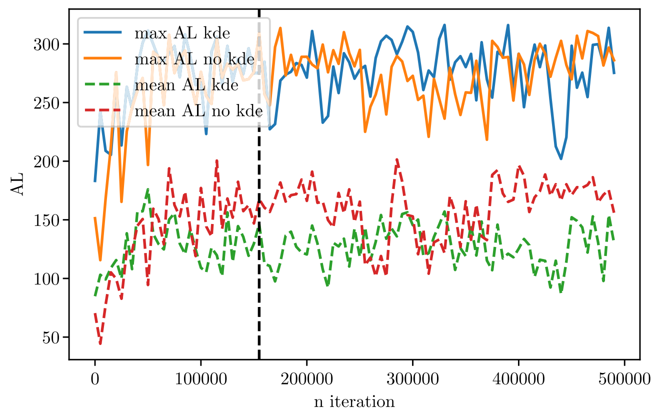


FIG. 11. Maximum and mean autocorrelation lengths for $n_{\mathrm{kde}} = 1$, comparing a run with KDE against a run without KDE. Black dashed line shows where KDE finally converged and stopped updating.

V. CONCLUSION

Bayesian formalism is a common approach in current gravitational wave data analysis. The inference of the parameters' posterior distribution is often done using MCMC, and the efficiency strongly depends on the jump proposal it uses. In this article, we have presented a KDE-based proposal which can be either built on-the-fly during the extended burn-in stage or constructed using posterior points from another run.

The suggested KDE-based jump proposal has several extended features: (i) the adaptive bandwidth based on the local density of points (small bandwidth in the densely sampled regions of the parameter space); (ii) splitting the parameter space into subgroups of the correlated parameters and applying KDE on each subgroup, we identify correlations using JSD; (iii) the possibility of building KDE adaptively.

We tested this jump proposal by running MCMC on IPTA data using a KDE that we have built from previous MCMC runs (i.e., nonadaptive case). The advantage of a KDE-based jump proposal was clearly seen in the high acceptance rate with low-autocorrelation length. We found that using a relatively low value $n_{\mathrm{kde}} = 1$–5 (number of subgroups used in the jump simultaneously) seems to be optimal.

Another application of the KDE-based jump proposal was running PTMCMC on the simulated LISA data searching for Galactic white dwarf binaries in a narrow frequency band. In this case, we have built the KDE adaptively during an extended burn-in stage. The addition of the KDE jump proposal to the sampling had only a moderate impact: it shows a decent acceptance rate (about 31%) with only a small improvement in the autocorrelation length. Moreover, we have shown that the low-dimensional jumps are strongly preferred.

A few things could be improved, most notably in the adaptation. The threshold for grouping parameters was chosen somewhat *ad hoc*, and the correlation of some parameters could be close to the threshold. We did observe the fluctuation in choosing the subgroups during the adaptation. One possibility could be to choose not one but two plausible groupings, build KDE for each and use two KDE proposal distributions in a probabilistic manner. The criteria for stopping adaptation was also chosen somewhat arbitrarily and might benefit from further tuning. Finally, we should implement an adaptive tuning for optimal n_{kde} based on the acceptance rate.

ACKNOWLEDGMENTS

We acknowledge financial support from Agence Nationale de la Recherche (ANR-18-CE31-0015), and from CNES in the framework of the LISA mission.

APPENDIX A: KULLBACK-LEIBLER DIVERGENCE

We test the quality of our KDE by computing the KL divergence for each subgroup $\hat{f}_\alpha(\mathbf{x}_\alpha, \vec{h}_\alpha)$. Because we do not know the actual PDF for a set of samples $\{\vec{X}\}$, we have to resort to binning. From sub-KDE $\hat{f}_\alpha(\mathbf{x}_\alpha, \vec{h}_\alpha)$ built on subgroup of parameter samples $\{\vec{X}_\alpha\}$, we draw a new set of samples $\{\vec{X}^*_\alpha\}$. We estimate the corresponding normalized histogram distributions P_α and P^*_α using same grid of N bins to compute the KL divergence as [18]

$$\mathrm{KL}(\mathrm{P}_\alpha \| \mathrm{P}^*_\alpha) = \sum_{i=0}^{N} \mathrm{P}_{\alpha,i} (\ln \mathrm{P}_{\alpha,i} - \ln \mathrm{P}^*_{\alpha,i}). \tag{A1}$$

To avoid divergence of the logarithms, we set every $\mathrm{P}_{\alpha,i}$ and $\mathrm{P}^*_{\alpha,i}$ that are equal to 0 to the minimum found value in $\mathrm{P}_\alpha \cup \mathrm{P}^*_\alpha$ that is not 0.

APPENDIX B: OPTIMAL BANDWIDTH OF KDE

We start this appendix with defining few useful expressions that will be used in our derivations later.

(i) The overlap between two neighbouring kernels of same bandwidth is given by

$$\int d\mathbf{x} K(\mathbf{x} - \vec{X}_a, \vec{h}) K(\mathbf{x} - \vec{X}_b, \vec{h}) = \prod_{i=1}^{d} \frac{\exp - \frac{1}{4} \frac{|\vec{X}_a - \vec{X}_b|_i^2}{h_i^2}}{\sqrt{\pi} 2 h_i}. \tag{B1}$$

Let us remind you that d is the dimensionality.

(ii) If a set of samples $\{\vec{X}\}$ of size N drawn from the probability density function $f(x)$, then we can approximate the averaging integral as

$$\int dx f(x) g(x) \simeq \frac{1}{N} \sum_a g(X_a), \tag{B2}$$

where the function g is evaluated at the sample points $\vec{X}_a$.

The main objective of this appendix is to derive the optimal local bandwidth which is defined through the minimization of the mean square error,

$$\begin{aligned} \epsilon^2 &= \int dx (\hat{f}(x,h) - f(x))^2 \\ &= \int dx \hat{f}^2(x,h) - 2 \int dx \hat{f}(x,h) f(x) + \int dx f^2(x) \\ &= \int dx \hat{f}^2(x,h) - 2 \int dx \hat{f}(x,h) f(x) + \text{const}, \end{aligned} \tag{B3}$$

where $f(x)$ is the true PDF, $(\hat{f}(x,h)$ is its KDE approximation and const is the term independent of the bandwidth h. We introduce local bandwidth h_a attached to each sample point X_a. Next, we assume that all points in the vicinity of each point have similar bandwidths, in other words, $h_b \approx h_a$ for k_{near} local points X_b. Using these assumptions we can approximate the first term,

$$N^2 \int dx \hat{f}(x,h_a)\hat{f}x, h_b$$

$$= \sum_a \prod_{i=1}^{d} \frac{1}{2\sqrt{\pi}h_{a,i}} + \sum_a \sum_{b\neq a} \prod_{i=1}^{d} \frac{e^{-\frac{1}{4}\frac{\Delta X^2_{ab,i}}{h^2_{a,i}}}}{2\sqrt{\pi}h_{a,i}}, \qquad (B4)$$

where $h_{a,i}$ is ith component of bandwidth attached to Gaussian kernel at $\vec{X}_a$ and $\Delta X_{ab,i} = (\vec{X}_a - \vec{X}_b)_i$ is ith component of the a vector connecting two samples in the parameter space.

Using now second bullet equation and excluding the actual sample from the sum (for improving stability and removing the possible bias, see "leave one out estimator" [8]) we obtain for the second term

$$2\int dx \hat{f}(x,h_a)\hat{f}x \approx \frac{2}{N^2} \sum_a \sum_{b\neq a} \prod_{i=1}^{d} \frac{e^{-\frac{1}{2}\frac{\Delta X^2_{ab,i}}{h^2_{a,i}}}}{\sqrt{2\pi}h_{a,i}}, \qquad (B5)$$

where we have assumed $N \gg 1$, $N(N-1) \approx N^2$. Combining these terms together gives us

$$N^2(\epsilon^2 - \mathrm{const}) = \sum_a \left\{ \prod_{i=1}^{d} \frac{1}{2\sqrt{\pi}h_{a,i}} + \sum_{b\neq a} \prod_{i=1}^{d} \frac{e^{-\frac{1}{4}\frac{\Delta X^2_{ab,i}}{h^2_{a,i}}}}{2\sqrt{\pi}h_{a,i}} - 2\sum_{b\neq a} \prod_{i=1}^{d} \frac{e^{-\frac{1}{2}\frac{\Delta X^2_{ab,i}}{h^2_{a,i}}}}{\sqrt{2\pi}h_{a,i}} \right\}.$$

Find the minimum of this expression by differentiating with respect to $h_{c,j}$ and equating it to zero,

$$0 = -\frac{1}{h_{c,j}} \prod_{i=1}^{d} \frac{1}{2\sqrt{\pi}h_{c,i}} \left\{ 1 + \sum_{b\neq c} \left[1 - \frac{1}{2}\frac{\Delta X^2_{bc,j}}{h^2_{c,j}} \right] \prod_{i=1}^{d} e^{-\frac{1}{4}\frac{\Delta X^2_{bc,i}}{h^2_{c,i}}} - 2(\sqrt{2})^d \left[1 - \frac{\Delta X^2_{bc,j}}{h^2_{c,j}} \right] \prod_{i=1}^{d} e^{-\frac{1}{2}\frac{\Delta X^2_{bc,i}}{h^2_{c,i}}} \right\}.$$

Next we assume a quite conservative approximation; $\frac{\Delta X^2_{bc,j}}{h^2_{c,j}} \ll 1$ for all points $\vec{X}_b$ in vicinity of $\vec{X}_c$ and all components j. This assumption overestimates the bandwidth and is therefore conservative; this is what is used in this paper. Expanding in these small parameters and retaining only the quadratic terms in this small ratio we obtain the system of linear equations for $1/h^2_{c,j}$,

$$\frac{k_{\mathrm{near}}(2^{d/2+1} - 1) - 1}{(2^{d/2} - 1)} = \sum_{b\neq c} \left[3\frac{\Delta X^2_{bc,j}}{h^2_{c,j}} + \sum_{i\neq j} \frac{\Delta X^2_{bc,i}}{h^2_{c,i}} \right]. \qquad (B6)$$

Solving this system at each point $\vec{X}_c$ for each direction in the parameter space (j) gives us the desired local bandwidth $\vec{h}_c$.

As an alternative approach we can assume that the bandwidth is comparable to the distance to the neighbors and define $h^2_{c,j} = \overline{\Delta X^2}_{c,j}(1 + \varepsilon_{c,j})$, where $\overline{\Delta X^2}_{c,j} = 1/k_{\mathrm{near}} \sum_b \Delta X^2_{bc,j}$ is the average square distance (ith component) to the points in vicinity of $\vec{X}_c$ and assume that $\Delta X^2_{bc,j}/\overline{\Delta X^2}_{c,j} \sim 1$ and $\varepsilon_{c,j} \ll 1$ for all b, c, j. This yields

$$\prod_{i=1}^{d} e^{-\frac{1}{4}\frac{\Delta X^2_{bc,i}}{h^2_{c,i}}} \approx \left(1 + \frac{1}{4}\sum_i^d \frac{\Delta X^2_{bc,i}}{\overline{\Delta X^2}_{c,i}} \right) \prod_{i=1}^{d} e^{-\frac{1}{4}\frac{\Delta X^2_{bc,i}}{\overline{\Delta X^2}_{c,i}}}. \qquad (B7)$$

Using this approximation we arrive at the system of linear equations for $1/h^2_{c,j}$,

$$-1 + \sum_{b\neq c} 2(\sqrt{2})^d \prod_{i=1}^{d} e^{-\frac{1}{2}\frac{\Delta X^2_{bc,i}}{\overline{\Delta X^2}_{c,i}}} - P_{bc}\left(1 - \frac{1}{2}\sum_i^d \frac{\Delta X^2_{bc,i}}{\overline{\Delta X^2}_{c,i}} \right) = \sum_{b\neq c} P_{bc} \left(3\frac{\Delta X^2_{bc,j}}{\overline{\Delta X^2}_{c,j}}\varepsilon_{c,j} + \sum_{i\neq j} \frac{\Delta X^2_{bc,i}}{\overline{\Delta X^2}_{c,i}}\varepsilon_{c,i} \right),$$

where

$$P_{bc} \equiv \frac{1}{4}\left[\prod_{i=1}^{d} e^{-\frac{1}{4}\frac{\Delta X^2_{bc,i}}{\overline{\Delta X^2}_{c,i}}} - 4(\sqrt{2})^d \prod_{i=1}^{d} e^{-\frac{1}{2}\frac{\Delta X^2_{bc,i}}{\overline{\Delta X^2}_{c,i}}} \right].$$

Solving this system at each point $\vec{X}_c$ for each direction in the parameter space (j) gives us the desired local bandwidth $\vec{h}_c$.

APPENDIX C: THE M3C2 SAMPLER

The M3C2 (footnote 1) (multiple parallel Markov Chain Monte Carlo) is a python implementation of MCMC sampler. This tool aims to improve the sampling robustness of complex posterior distribution by running multiple chains in parallel. The cross-check of the chain performance informs us about convergence (using the Gelman-Rubin ratio [5]). We have implemented two mechanisms of building the chain: (1) using the slice sampling (slice) and (2) Metropolis-Hastings algorithm (MH) which could be used separately or together to improve the mixture of the chains and reducing the autocorrelation length. Even though the sampler is very generic, we primarily use it within the context of GW data analysis. For the Metropolis-Hastings method, we have implemented a set of proposal jumps:

(i) `SCAM` (single component adaptive Metropolis), jumps along one randomly chosen direction given by the eigenvectors of the covariance matrix [16,23].

(ii) `DE` (differential evolution), jumps along the direction given by the difference of two randomly chosen samples of the chains, or (as in the classic implementation [24,25]) by using the state of different chains running in parallel.

(iii) `ReMHA` (regional Metropolis-Hastings algorithm), the proposal represented by a mixture of several Gaussians distributions [16,31].

For `SCAM` we build the covariance matrix adaptively based on the accumulated samples. The use of the accumulated samples breaks the Markov property of the chain, making it asymptotically Markovian. The stability of the covariance matrix is yet another sign of the converged chain. One can stop adaptation after the burn-in run. This proposal is suggested in [32].

`ReMHA` is similarly used adaptively. We use accumulated samples during the burn-in to estimate the number of clusters using variational Bayesian-Gaussian mixture (`skikit-learn` package) and use this Gaussian mixture probability as a proposal. This proposal is similar to the one suggested in [31].

`DE` could be used as a proposal (snooker) described in [24] using multiple chains running in parallel or using the accumulated samples for each chain to propose the jump. In the case of well-converged chains, there is not much difference between those two ways. However, the behavior and efficiency of those two implementations is very different during the burn-in stage.

In `slice` sampling, we use slicing of the parameter space either randomly or along the eigendirections of the covariance matrix; the choice is made with a probability set by the user. In the case of a mixture of `slice` and MH, the frequency of each method is defined by a user-specified weight. In addition to preset proposals available in `M3C2`, the user can add custom jump-proposals using a standard interface. The weights and proposals can be set individually for each running chain.

Besides running parallel independent chains, `M3C2` sampler also has parallel tempering implementation with an adaptive temperature ladder following [33]. In this method, we modify the likelihood by introducing the "temperature" [34], which makes it lower and broader, allowing "hot chains" to explore parameter space at a larger scale. The "melted" hot chains play the role of a proposal there, and its efficiency depends on the interplay of the number (and distribution) of hot chains (more chains is better) and computational demands (increase with the number of chains). The adaptation aims at increasing the acceptance rate between the chains.

The multichain scheme of `M3C2` can be easily deployed on the multicore CPU infrastructure. In the case of parallel tempering, data exchange between chains is restricted to its minimum level (pairwise communication between the chains) to ensure good scalability.

[1] D. J. Earl and M. W. Deem, Parallel tempering: Theory, applications, and new perspectives, Phys. Chem. Chem. Phys. **7**, 3910 (2005).

[2] G. Ashton and C. Talbot, BILBY-MCMC: An MCMC sampler for gravitational-wave inference, Mon. Not. R. Astron. Soc. **507**, 2037 (2021).

[3] B. Farr, W. Farr, D. Rudd, A. Price-Whelan, and D. Macleod, kombine: Kernel-density-based parallel ensemble sampler, Astrophysics Source Code Library, record ascl:2004.010, 2020, https://github.com/bfarr/kombine.

[4] G. O. Roberts and J. S. Rosenthal, Coupling and ergodicity of adaptive Markov Chain Monte Carlo algorithms, J. Appl. Probab. **44**, 458 (2007).

[5] A. Gelman and D. B. Rubin, Inference from iterative simulation using multiple sequences, Stat. Sci. **7**, 457 (1992).

[6] E. Parzen, On estimation of a probability density function and mode, Ann. Math. Stat. **33**, 1065 (1962).

[7] M. Rosenblatt, Remarks on some nonparametric estimates of a density function, Ann. Math. Stat. **27**, 832 (1956).

[8] Y. Jin, Y. He, and D. Huang, An improved variable kernel density estimator based on L2 regularization, Mathematics **9**, 16 (2021).

[9] G. R. Terrell and D. W. Scott, Variable Kernel density estimation, Ann. Stat. **20**, 1236 (1992).

[10] F. Nielsen, On the Jensen-Shannon symmetrization of distances relying on abstract means, Entropy **21**, 485 (2019).

[11] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proceeding of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, edited by L. M. Le Cam and J. Neyman (University of California Press, Berkeley, California, 1967), Vol. 1, pp. 281–297.

[12] B. P. Carlin and S. Chib, Bayesian model choice via Markov Chain Monte Carlo methods, J. R. Stat. Soc. **57**, 473 (1995).

[13] S. Hee, W. J. Handley, M. P. Hobson, and A. N. Lasenby, Bayesian model selection without evidences: Application to the dark energy equation-of-state, Mon. Not. R. Astron. Soc. **455**, 2461 (2016).

[14] G. Casella and E. I. George, Explaining the Gibbs sampler, Am. Stat. **46**, 167 (1992).

[15] W. K. Hastings, Monte Carlo sampling methods using Markov Chains and their applications, Biometrika **57**, 97 (1970).

[16] G. Roberts and J. Rosenthal, Examples of adaptive MCMC, J. Comput. Graph. Stat. **18**, 349 (2009).

[17] D. W. Hogg and D. Foreman-Mackey, Data analysis recipes: Using Markov Chain Monte Carlo, Astrophys. J. Suppl. Ser. **236**, 11 (2018).

[18] S. Kullback and R. A. Leibler, On information and sufficiency, Ann. Math. Stat. **22**, 79 (1951).

[19] Y. F. Atchadé and J. S. Rosenthal, On adaptive Markov Chain Monte Carlo algorithms, Bernoulli **11**, 815 (2005).

[20] B. B. P. Perera *et al.*, The International Pulsar Timing Array: Second data release, Mon. Not. R. Astron. Soc. **490**, 4666 (2019).

[21] J. Ellis and R. van Haasteren, jellis18/ptmcmcsampler: Official release, 2017, https://github.com/jellis18/PTMCMCSampler.

[22] J. A. Ellis, M. Vallisneri, S. R. Taylor, and P. T. Baker, ENTERPRISE: Enhanced numerical toolbox enabling a robust pulsar inference suite, Zenodo, 2020, 10.5281/zenodo.4059815.

[23] H. Haario, E. Saksman, and J. Tamminen, An adaptive metropolis algorithm, Bernoulli **7**, 223 (2001).

[24] Cajo J. F. Ter Braak and J. A. Vrugt, Differential evolution Markov Chain with snooker updater and fewer chains, Stat. Comput. **18**, 435 (2008).

[25] Cajo J. F. Ter Braak, A Markov Chain Monte Carlo version of the genetic algorithm differential evolution: Easy Bayesian computing for real parameter spaces, Stat. Comput. **16**, 239 (2006).

[26] K. Aggarwal *et al.*, The NANOGrav 11 yr data set: Limits on gravitational waves from individual supermassive black hole binaries, Astrophys. J. **880**, 116 (2019).

[27] S. R. Taylor, P. T. Baker, J. S. Hazboun, J. Simon, and S. J. Vigeland, Enterprise extensions, 2021, v2.3.3, https://github.com/nanograv/enterprise_extensions.

[28] A. Gelman, W. R. Gilks, and G. O. Roberts, Weak convergence and optimal scaling of random walk metropolis algorithms, Ann. Appl. Probab. **7**, 110 (1997).

[29] R. M. Neal, Slice sampling, Ann. Stat. **31**, 705 (2003).

[30] T. B. Littenberg, A detection pipeline for galactic binaries in LISA data, Phys. Rev. D **84**, 063009 (2011).

[31] R. Craiu, J. Rosenthal, and C. Yang, Learn from thy neighbor: Parallel-chain and regional adaptive MCMC, J. Am. Stat. Assoc. **104**, 1454 (2009).

[32] J. A. Ellis, A Bayesian analysis pipeline for continuous GW sources in the PTA band, Classical Quantum Gravity **30**, 224004 (2013).

[33] W. D. Vousden, W. M. Farr, and I. Mandel, Dynamic temperature selection for parallel tempering in Markov Chain Monte Carlo simulations, Mon. Not. R. Astron. Soc. **455**, 1919 (2016).

[34] R. Swendsen and J.-S. Wang, Replica Monte Carlo Simulation of Spin-Glasses, Phys. Rev. Lett. **57**, 2607 (1986).