Improved Spike-Sorting By Modeling Firing Statistics and Burst-Dependent Spike Amplitude Attenuation: A Markov Chain Monte Carlo Approach

Christophe Pouzat, Matthieu Delescluse, Pascal Viot, and Jean Diebolt

1Laboratoire de Physiologie Cérébrale, Centre National de la Recherche Scientifique (CNRS) Unité Mixte de Recherche (UMR) 8118, Université René Descartes, 75006, Paris; 2Laboratoire de Physique Théorique des Liquides, CNRS UMR 7600, Université Pierre et Marie Curie, 75252 Paris Cedex 05; and 3Laboratoire d’Analyse et de Mathématiques Appliquées, CNRS UMR 8050, Université de Marne la Vallée, Batiment Copernic, Cité Descartes, 77454 Marne la Vallée Cedex, France

Submitted 10 March 2003; accepted in final form 14 January 2004

Pouzat, Christophe, Matthieu Delescluse, Pascal Viot, and Jean Diebolt. Improved spike-sorting by modeling firing statistics and burst-dependent spike amplitude attenuation: a Markov chain Monte Carlo approach. J Neurophysiol 91: 2910–2928, 2004. First published January 28, 2004; 10.1152/jn.00227.2003. Spike-sorting techniques attempt to classify a series of noisy electrical waveforms according to the identity of the neurons that generated them. Existing techniques perform this classification ignoring several properties of actual neurons that can ultimately improve classification performance. In this study, we propose a more realistic spike train generation model. It incorporates both a description of “nontrivial” (i.e., non-Poisson) neuronal discharge statistics and a description of spike waveform dynamics (e.g., the events amplitude decays for short interspike intervals). We show that this spike train generation model is analogous to a one-dimensional Potts spin-glass model. We can therefore tailor to our particular case the computational methods that have been developed in fields where Potts models are extensively used, including statistical physics and image restoration. These methods are based on the construction of a Markov chain in the space of model parameters and spike train configurations, where a configuration is defined by specifying a neuron of origin for each spike. This Markov chain is built such that its unique stationary density is the posterior density of model parameters and configurations given the observed data. A Monte Carlo simulation of the Markov chain is then used to estimate the posterior density. We illustrate the way to build the transition matrix of the Markov chain with a simple, but realistic, model for data generation. We use simulated data to illustrate the performance of the method and to show that this approach can easily cope with neurons firing doublets of spikes and/or generating spikes with highly dynamic waveforms. The method cannot automatically find the “correct” number of neurons in the data. User input is required for this important problem and we illustrate how this can be done. We finally discuss further developments of the method.

INTRODUCTION

The study of neuronal populations activity is one of the main experimental challenges of contemporary neuroscience. Among the techniques that have been developed and used to monitor populations of neurons, the multielectrodes extracellular recordings, albeit one of the oldest, still stands as one of the methods of choice. It is relatively simple and inexpensive to implement (especially when compared to imaging techniques) and it can potentially give access to the activity of many neurons with a fine time resolution (below the ms). The full exploitation of this method nevertheless requires some basic problems to be solved, among which stands prominently the spike-sorting problem. The raw data collected by extracellular recordings are of course corrupted by some recording noise but, more important, are a mixture of activities from different neurons. This means that even in good situations where action potentials or spikes can be unambiguously distinguished from background noise, the experimenter is left with the intricate problem of finding out how many neurons are in fact contributing to the recorded data and, for each spike, which is the neuron of origin. This in essence is the spike-sorting problem. Extracellular recordings being an old method, the problem has been recognized as such for a long time and there is already a fairly long history of proposed solutions (Lewicki 1998). Nevertheless it seems to us that none of the available methods makes full use of the information present in the data. They consider mainly or exclusively the information provided by the waveform of the individual spikes and they neglect that provided by their occurrence time. Yet, the importance of the spikes occurrence times shows up in two ways.

1) First, the sequence of spike occurrence times emitted by a neuron has well-known features like the presence of a refractory period (e.g., after a spike has been emitted no other spike will be emitted by the same neuron for a period whose duration varies from 2 to 10 ms). Other features include an interspike interval (ISI) probability density, which is often unimodal and skewed (it rises “fast” and decays “slowly”). Although some methods make use of the presence of a refractory period (Fee et al. 1996a; Harris et al. 2000), none makes use of the full ISI density. This amounts to throwing useful information away, for if event 101 has been attributed to neuron 3 we not only know that no other spike can arise from neuron 3 for, say, the next 5 ms, but we also know that the next spike of this neuron is likely to occur with an ISI between 10 and 20 ms. As pointed out by Lewicki (1994), any spike-sorting method that would include an estimate of the ISI density of the different neurons should produce better classification performances.

2) Second, the spike waveform generated by a given neuron often depends on the time elapsed since its last spike (Quirk et al. 1999). This nonstationarity of spike waveform will worsen the classification reliability of methods that assume stationary waveform (Lewicki 1994; Pouzat et al. 2002). Other methods...
(e.g., Gray et al. 1995 for a “manual” method and Fee et al. 1996a for an automatic one) assume a continuity of the cluster of events originating from a given neuron when represented in a “feature space” [the peak amplitude, valley(s) amplitude(s), half-width are examples of such features] and make use of this property to classify nonstationary spikes. Although it is clear that the latter methods will in general outperform the former, we think that some cases could arise where none of them would work. In particular, doublet-firing cells could generate well-separated clusters in feature space and these clusters would not be put together by presently available methods.

These considerations motivated us to look for a new method that would have a built-in capability to take into account both the ISI density of the neurons and their spike waveform dynamics. However, there are two additional issues with spike-sorting that, we think, are worth considering.

1) “Hard” versus “soft” spike-sorting. Most spike-sorting methods, including the ones where users actually perform the clustering and some automatic methods like the one of Fee et al. (1996a), generate “hard” classification. That is, each recorded event is attributed “fully” to one of the K neurons contributing to the data. Methods based on probabilistic models (Lewicki 1994; Nguyen et al. 2003; Pouzat et al. 2002; Sahani 1999) deal with this issue differently: each event is given a probability to have been generated by each of the K neurons (e.g., one gets results like: event 100 was generated by neuron 1 with probability 0.75 and by neuron 2 with probability 0.25). We will refer to this kind of classification as “soft” classification. Users of the latter methods are often not aware of their soft classification aspect because what one does for subsequent processing like estimating the ISI density of a given neuron or computing the cross-correlogram of spike trains from 2 different neurons, is to force the soft classification into the most likely one (in the example above we would force the classification of event 100 into neuron 1). By doing so, however, the analyst introduces a bias in the estimates. It seems therefore interesting to find a way to keep the soft classification aspect of the model-based methods when computing estimates.

2) Confidence intervals on model parameters. Methods based on an explicit model for data generation could in principle generate confidence intervals for their model parameters, although this was never done. However, the values of model parameters do influence the classification and a source of bias could be reduced by including information about uncertainty of model parameters values.

Based on these considerations we have developed a semi-automatic method that takes spike waveform dynamics and ISI densities into account, which produces soft classification and allows the user to make full use of it, and which generates a full posterior density for the model parameters and confidence intervals. Our method uses a probabilistic model for data generation where the label (i.e., neuron of origin) of a given spike at a given time depends on the label of other spikes occurring just before or just after. We will call configuration the specification of a label for each spike in the train. It will become clear that with this model the spike-sorting problem is analogous to an image restoration problem where a picture (i.e., a set of pixel values) that has been generated by a real object and corrupted by some noise is given. The problem is to find out what the actual pixel value was, given the noise properties and the known correlation properties of noise free images. The case of spike sorting is equivalent to a one-dimensional “pixel sequence” where the pixel value is replaced by the label of the spike. More generally it is a special case of a Potts spin-glass model encountered in statistical physics (Newman and Barkema 1999; Wu 1982). Thanks to this analogy solutions developed to study Potts models (Landau and Binder 2000; Newman and Barkema 1999) and to solve the image restoration problem (Geman and Geman 1984) can be tailored to the spike-sorting problem. These solutions rely on a Markov chain that has the posterior density of model parameters and configurations given the observed data as its unique stationary density. This Markov Chain is stochastically simulated on the computer. This general class of methods has been used for 50 years by physicists (Metropolis et al. 1953) who call it Dynamic Monte Carlo and for 20 years by statisticians (Fishman 1996; Geman and Geman 1984; Liu 2001; Robert and Cassela 1999) who call it Markov Chain Monte Carlo (MCMC).

Our purpose in this paper is 2-fold: First to demonstrate that a particular MCMC method allows incorporation of more realistic data generation models and by doing so to perform reliable spike-sorting even in difficult cases (e.g., in the presence of neurons generating 2 well-separated clusters in feature space). Second, to explain how and why this method works, thereby allowing its users to judge the quality of the produced classification, to improve the algorithm implementing it, and to adapt it to their specific situations. Our method does not yet provide an automatic estimate of the number of neurons present in the data, although we illustrate how this critical problem can be addressed by the user.

METHODS

Data generation model

In the present manuscript we will make the following assumptions about the data generation.

1) The firing statistics of each neuron are fully described by a time-independent interspike interval density. That is, the sequence of spike times from a given neuron is a realization of a homogeneous renewal point process (Johnson 1986).

2) The spike amplitudes generated by each neuron depend on the elapsed time since the previous spike of this neuron.

3) The measured spike amplitudes are corrupted by a Gaussian white noise, which sums linearly with the spikes and is statistically independent of them.

These assumptions have been chosen to show as simply as possible the implementation of our method and should not be taken as intrinsic limitations of this method. They constitute, moreover, a fairly good first approximation of real data in our experience. More sophisticated models where the next ISI of a neuron depends on the value of the former one (Johnson et al. 1986) could easily be included. In the same vein, models where the amplitude of a spike depends on several of the former ISIs could be considered. The Gaussian white noise assumption means that the analyst has “whitened” the noise before starting the spike-sorting procedure (Pouzat et al. 2002).

INTERSPIKE INTERVAL DENSITY. We will use in this paper a log-Normal density for the ISIs. This density looks like a good next guess, after the exponential density, when one tries to fit empirical ISI densities. It is unimodal, exhibits a refractory period, rises “fast,” and decays “slowly.” The version of the log-Normal density we are using depends on 2 parameters: a dimensionless shape parameter μ and a scale parameter σ measured in seconds. The probability density for a
realization of a log-Normal random variable $I$ with parameters $\sigma$ and $s$ to have the value $i$ is

$$\pi_{is}(I = i|\sigma, s) = \frac{1}{\sqrt{2\pi} \sigma} \exp \left[ -\frac{1}{2} \left( \frac{\log(i)}{\sigma} \right)^2 \right]$$

In what follows we will use a shorter notation

$$i \sim \log - \text{Norm}(\sigma, s)$$

meaning that $i$ is a realization of a log-Normal random variable with parameters $\sigma$ and $s$. A discussion of the properties of the log-Normal distribution can be found in the e-Handbook of Statistical Methods.

SPIKE AMPLITUDE DYNAMICS. We will consider events described by their occurrence time and their peak amplitude measured on one or several recording sites. This makes the equations more compact, but full waveforms can be accommodated if necessary. Following Fee et al. (1996b) we will describe the dependence of the amplitude on the ISI density and $2\text{nd}$ parameters (e.g., for $t_1$, $a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}$ for tetrode recording). We will use $Y$ to refer to the full data sample, that is

$$Y = (i_1, a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}, \ldots, b_5, a_{5,2}, a_{5,3}, a_{5,4})$$

Data augmentation, configurations, and likelihood function

DATA AUGMENTATION AND CONFIGURATIONS. Until now we have specified our model and our data representation but what we are really interested in is to find the neuron of origin of each individual spike, or more precisely, for a model with $K$ neurons, the probability for each spike to have been generated by each of the $K$ neurons (remember that we are doing soft classification). To do that we will associate with each spike $j$ a latent variable, $c_j \in \{1, 2, \ldots, K\}$. When $c_j$ has value 2, that means that spike $j$ has been “attributed” to neuron 2 of the model. We will use $C$ to refer to the set of $N$ variables $c$

$$C = (c_1, \ldots, c_N)$$

For a data sample of size $N$ and a model with $K$ neurons we have $K^N$ different $C$ values. We will call configuration a specific $C$ value. That is, a configuration is a $N$-dimensional vector whose components take value in $\{1, 2, \ldots, K\}$ (e.g., $(1, 1, \ldots, 1)$, $(K, \ldots, K)$, are $2$ configurations.) In the statistical literature, this procedure of “making the data more informative” by adding a latent variable is called data augmentation (Liu 2001; Robert and Casella 1999).

LIKELIHOOD FUNCTION. Because $C$ has been introduced it is pretty straightforward to compute the likelihood function of the “augmented” data, given specific values of $\theta$: $L(Y, C|\theta)$. We remind the reader here that the likelihood function is proportional to the probability density of the (augmented) data given the parameters. Figure 1 illustrates how this is done on a simple case with $2$ units in the model ($K = 2$) and a single recording site.

We first use the configuration specified by $C$ to split the sample into $K$ spike trains, one train from each neuron. (This is done in Fig. 1 when one goes from $A$ to $B$ and $C$.) Then because our model in its present form does not include interactions between neurons we just need to compute $K$ distinct likelihood functions, one for each neuron specific spike train, and multiply these $K$ terms to obtain the full likelihood

$$A$$

$$2$$

$$1$$

$$\theta$$

$$2$$

$$1$$

$$a_{11}$$

$$a_{12}$$

$$a_{13}$$

$$a_{14}$$

$$a_{15}$$

$$t_{11}$$

$$t_{12}$$

$$t_{13}$$

$$t_{14}$$

$$t_{15}$$

$$t_{15} - t_{12}$$

Train 1

$$B$$

$$C$$

Train 2

FIG. 1. Likelihood computation: $A$: snapshot of spikes $11$ to $15$ of a train. $c_j$ values are shown at the top. Spike amplitudes are given ($a_i$) as well as the occurrence times ($t_j$). $B$: spikes from units $1$ are shown alone. $C$: spikes from units $2$ are shown alone.
In the case illustrated on Fig. 1 we will have for the between 0 and working with normalized amplitudes. We will re that the joint prior density \( f_{\text{joint}}(\theta) \) on. For the model parameters and configu given the data using Bayes for- lation of \( f_{\text{post}}(\theta) \) is automatically taken into account. The reader can see that for any con- must, moreover, be smaller than \( \delta \), that is, the uncertainty on the former and next spins with \( \pm 1 \) \( \delta \) must be \( \leq 1 \) (the amplitude of a spike from a given neuron on a given recording site does not change sign) and \( \geq 0 \) (spikes do not become larger upon short ISI), so we used a uniform density between 0.1 and 0.9 for \( \delta \). An inspection of the effect of the shape parameter \( \sigma \) on the ISI density is enough to convince an experienced neurophysiologist that empirical unimodal ISI densities from well-isolated neurons will have \( \sigma \in [0.1, 0.2] \). We therefore took a prior density uniform between 0.1 and 2 for \( \sigma \). The same considerations led us to take a uniform prior density between 0.005 and 0.5 for \( s \). When one analyzes the following data chunks, say spikes 5,001 to 1,000, the best strategy is to take as a prior density for \( \theta \) the posterior density from the previous chunk.

\[
\pi_{\text{post}}(\theta, C|Y) = \sum_c \pi_{\text{post}}(\theta, C|Y) \tag{17}
\]

One has therefore a direct way to track parameters drifts during an experiment.

A COMBINATORIAL EXPLOSION. We have so far avoided considering the normalizing constant \( Z \) of Eq. 12, whose expression is given by \( \text{Eq. 13} \). A close examination of this equation shows that it involves a multidimensional integration in a rather highly dimensional space (e.g., for a model with 10 neurons and data from tetrode recordings, we have 80 components in \( \theta \) and a summation over every possible configuration, that is a sum of \( K^N \) terms. That is really a lot, to say the least, in any realistic case (say, \( N = 1,000 \) and \( K = 10 \)!) One can wonder how we will manage to compute \( Z \) and therefore implement our approach. Fortunately, we do not really need to compute it, as will now be explained.

The Markov Chain Monte Carlo method

AN ANALOGY WITH THE POTTS MODEL IN STATISTICAL PHYSICS. Let us define

\[
e(\theta, C|Y) = L(Y, C|\theta)\pi_{\text{post}}(\theta) \tag{15}
\]

that is, \( e(\theta, C|Y) \) is an unnormalized version of \( \pi_{\text{post}}(\theta, C|Y) \), the posterior density, and

\[
E(\theta, C) = -\log[e(\theta, C|Y)] \tag{16}
\]

Then Eq. 12 can be rewritten as

\[
\pi_{\text{post}}(\theta, C|Y) = \frac{\exp[-BE(\theta, C)]}{Z} \tag{17}
\]

with \( \beta = (1/kT) = 1 \). If one interprets \( E \) as an energy, \( \beta \) as an “inverse temperature,” \( Z \) as a partition function, one sees that the posterior density, \( \pi_{\text{post}}(\theta, C|Y) \), is the canonical density (or canonical distribution) used in statistical physics (Landau and Binder 2000; Newman and Barkema 1999). A closer examination of Eqs. 10 and 11 shows that, from a statistical physics viewpoint, the (log of the) likelihood function describes “interactions” between spikes attributed to the same neuron. If one goes a step further and replaces “spikes” by “atoms on crystal nodes” and “neuron of origin” by “spin orientation,” one sees that our problem is analogous to what physicists call a one-dimensional Potts model. The typical Potts model is a system of interacting spins where each spin interacts only with its nearest neighbors and only if they have the same spin value (Wu 1982). In our case, spins (that is, spikes) interact with the former and next spins with the same value regardless of the distance (time interval) at which they are located. Moreover the “interaction” strength is a random variable that makes our system analogous to a spin glass (Binder and Young 1986; Landau and Binder 2000; Newman and Barkema 1999) and the amplitude contribution (Eq. 3) gives rise to an energy term analogous to an interaction between a spin and a random magnetic field (New- man and Barkema 1999). To be complete we should say that physi-
exists consider that \( \theta \) is given, then Eq. 17 gives the probability to find the “crystal” in a specific configuration \( C \).

The point of this analogy will now become clear. Physicists are interested in estimating expected values (which are what they can experimentally measure) of functions that can be easily computed for each configuration. For instance, they want to compute the expected energy of the “crystal”

\[
\langle E(\theta) \rangle = \sum_c E(\theta, C) \pi_{post}(\theta, C) Y
\]

To do that they obviously have to deal with the combinatorial explosion we alluded to. The idea of their solution is to draw configurations: \( C^{(1)}, C^{(2)}, \ldots, C^{(m)} \), from the “target” distribution: \( \pi_{post}(\theta, C) Y \). Then an estimate of \( \langle E(\theta) \rangle \) is the empirical average

\[
\hat{E}(\theta) = \frac{1}{m} \sum_{i=1}^{m} E[\theta, C^{(i)}]
\]

In other words, they use a Monte Carlo (MC) method (Landau and Binder 2000; Newman and Barkema 1999). The problem becomes correlated; we just need to be cautious when we compute the variance of a Markov chain. That means that in the general case, where we are interested in getting both expected values and errors on these estimates. The estimator of the probability for a given spike, say the 100th, to be correlated; we just need to be cautious when we compute the variance of the estimator (ACF), \( \rho_{normal}(l; \theta) \), defined by

\[
\rho(l; \theta) = \frac{1}{N_T - N_D - 1} \sum_{n=0}^{N_T} \rho(l; \theta_0)
\]

Then we compute the integrated autocorrelation time, \( \tau_{autocor}(\theta) \)

\[
\tau_{autocor}(\theta) = \frac{1}{\sum r \rho(l; \theta)}
\]

Our estimate of the variance, \( \text{Var} [\hat{\theta}] \), of \( \hat{\theta} \) becomes

\[
\text{Var} [\hat{\theta}] = \frac{2 \tau_{autocor}(\theta)}{N_T - N_D - 1} \sigma^2(\theta)
\]

The first consequence of the autocorrelation of the states of the chain is therefore to reduce the effective sample size by a factor of \( 2 \tau_{autocor}(\theta) \). This gives us a first quantitative element on which different algorithms can be compared (as explained in The Metropolis–Hastings algorithm of the APPENDIX, the MH algorithm does in fact give us a lot of freedom on the choice of proposal transition kernels). It is clear that the faster the autocorrelation functions of the parameters fall to zero, the greater the statistical efficiency of the algorithm. The other quantitative element we want to consider is the computational time, \( \tau_{cpu} \), required to perform one MC step of the algorithm. One could for instance imagine that a new sophisticated proposal transition kernel allows us to reduce the largest \( \tau_{autocor} \) of our standard algorithm by a factor of 10, but at the expense of an increase of \( \tau_{cpu} \) by a factor of 100. Globally the new algorithm would be 10 times less efficient than the original one. What we want to keep as small as possible is therefore the product \( \tau_{autocor} \tau_{cpu} \). With this efficiency criterion in mind, the replica exchange method described in Slow relaxation and REM becomes even more attractive.

Initial guess

MCMC methods are iterative methods that must be started from an initial guess. We chose randomly with a uniform probability \( 1/N \) as many actual events as neurons in the model (\( K \)). That gave us our initial guesses for the \( P_{ij} \). \( \delta \) was set to \( \delta_0 \) for each neuron. All the other parameters were randomly drawn from their prior distribution. The initial configuration was generated by labeling each individual...
spike with one of the \( K \) possible labels with a probability \( 1/K \) for each label (this is the \( \beta = 0 \) initial condition used in statistical physics).

**Data simulation**

We have tried to make the simulated tetrode data set used in this paper a challenging one. First the noise corruption was generated from a Student's \( t \) density with 4 degrees of freedom instead of a Gaussian density as assumed by our model (Eq. 3). Second, among the 6 simulated neurons only 3 had an actual log-Normal ISI density. Among the 3 others, one had a gamma density simulated neurons only 3 had an actual log-Normal ISI density.

\[
\sigma_{\text{simulated}}(\theta, s) = \frac{5^s}{\Gamma(s)} r^{s-1} \exp\left(-\frac{i}{r}\right)
\]

where \( s \) is a scale parameter and \( \sigma \) is a shape parameter. One neuron was a doublet-generating neuron and the third one had a truncated log-Normal ISI density. The truncation of the latter came from the fact that 18% of the events it generated were below the “detection threshold,” which would have been at 3 noise SD. The doublet-generating neuron was obtained with 2 log-Normal densities, one giving rise to short ISIs (10 ms) the other one to “long” ones (60 ms); the neuron was moreover switching from the short, respectively the long, ISI generating state to the long, respectively the short, ISI generating state after each spike, giving rise to a strongly bimodal ISI density (Fig. 9C). The mean ISI of these 6 neurons ranged from 10 to 210 ms (Table 1). A “noise” neuron was moreover added to mimic the collective effect of many neurons located far away from the recording sites. The data set has been made challenging for the algorithm by producing systematic deviations with respect to our model assumptions (Data simulation). This data set is supposed to come from 15 s of recording during which 5,058 events were detected. Wilson plots as they would appear to the user are shown on Fig. 2A. Figure 2B shows the same data with colors corresponding to the neuron of origin. The parameters used to simulate each individual neuron are given in Table 1. The reader can easily see that, whereas one cluster has very few points (purple cluster on Fig. 2B), others have a lot (brown, green, and blue clusters). Indeed, the smallest cluster contains 73 events, whereas the biggest contains 1,474 events. Some clusters are very elongated (e.g., green and red clusters) and one neuron, the doublet-generating neuron, even gives rise to 2 well-separated clusters (red). The “noise” neuron cluster (black) is much more spread out than the others. Two cluster pairs always exhibit an overlap (the yellow-black and the green-red pairs).

The deviation between the recording noise properties assumed by our model and the simulated noise is illustrated in Fig. 3. Figure 3A shows the peak amplitude versus the \( isi \) for the second (green) neuron on the second recording site together with the theoretical relation between these 2 parameters. The reader can therefore get an idea of what Eq. 2 means. Figure

---

**TABLE 1. Parameters used to simulate the neurons**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1, P_2, P_3, P_4 )</td>
<td>15, 10, 5, 0</td>
<td>10, 15, 5, 0</td>
<td>8, 12, 8, 3</td>
<td>9, 9, 9, 9</td>
<td>3, 8, 14, 19</td>
<td>3, 4, 5, 4</td>
<td>0, 0, 0, 0</td>
</tr>
<tr>
<td>( \delta )</td>
<td>0.7</td>
<td>0.8</td>
<td>0.6</td>
<td>0.5</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>100</td>
<td>100</td>
<td>50</td>
<td>67</td>
<td>20</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>( s )</td>
<td>20</td>
<td>5</td>
<td>10 and 50</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>0.3</td>
<td>3</td>
<td>0.2 and 0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>( \langle isi \rangle )</td>
<td>21</td>
<td>15</td>
<td>31</td>
<td>10</td>
<td>210</td>
<td>28</td>
<td>20</td>
</tr>
<tr>
<td>( n )</td>
<td>732</td>
<td>981</td>
<td>486</td>
<td>1,474</td>
<td>73</td>
<td>554</td>
<td>758</td>
</tr>
</tbody>
</table>

The maximal peak amplitude values (\( P_i \)) are given in units of noise SD. The scale parameters (\( s \)) and mean \( isi(\langle isi \rangle) \) are given in milliseconds. The neuron with a gamma ISI density is in column 2. For the doublet-generating neuron (column 3) the parameter values of the 2 underlying log-normal densities are given. The neuron with a truncated log-normal density is located in column 6. The bottom row indicates the number of events from each neuron. The correspondence between neuron number and color on Figs. 2B and 5A is: 1, blue; 2, green; 3, red; 4, brown; 5, purple; 6, yellow; 7, black.
3A2 shows the corresponding residuals (actual value \(\text{vs} \) theoretical one). By looking at the distribution of these points the reader can see that the cluster shape of this neuron, in the 4-dimensional space whose projections are shown on the Wilson plots of Fig. 2, will be significantly skewed. Algorithms assuming a multivariate Gaussian shape of the clusters would therefore not perform well on these data. Figure 3B1 shows the histogram of these residuals together with the Student's \(t\) density that was used to generate them (Data simulation & Random number generators for the simulated data) and the Gaussian density assumed by our model. The simulated recording noise generates at the same time more events very close to the ideal event and more events very far from it than a Gaussian noise would do.

**Early exploration and model choice**

We want to illustrate here some basic features of our algorithm dynamics as well as how model comparison can be done. In its present form, our approach requires the user to decide
what is the proper model (i.e., the proper number of neurons). This decision is based on the examination of “empirical” features associated with the different models. To obtain these empirical features, we have to perform the same computation on the different models, although the computation time increases with the number of events considered. It is therefore a good idea to start with a reduced sample that contains enough events for the model comparison to be done and little enough for the computation to run quickly. In this section we will work with the first 3 s of data (1,017 events), which represent approximately 20% of the total sample (5,058 events). We will focus here on a model with 7 neurons, but a similar study was performed with models containing from 4 to 10 neurons.

EVIDENCE FOR META-STABLE STATES. To explore a model with a given number of neurons we start by simulating 10 different and independent realizations of a Markov chain. The way random initial guesses for these different realizations are obtained is explained in Initial guess. For each realization, one Monte Carlo (MC) step consists in an update of the label of each of the 1,017 spikes (SPIKE LABEL TRANSITION MATRIX), an update of the amplitude parameters ($\mathbf{P}$, $\delta$, $\lambda$) (AMPLITUDE PARAMETERS TRANSITION KERNELS) and of the 2 parameters of the ISI density, $s$ and $\sigma$ (SCALE PARAMETER TRANSITION KERNEL & SHAPE PARAMETER TRANSITION KERNEL) for each neuron. As explained in AMPLITUDE PARAMETERS TRANSITION KERNELS, the amplitude parameters are generated with a piecewise linear approximation of the corresponding posterior conditional. This piecewise linear approximation requires a number of discrete sampling points to be specified. Because when we start our model exploration, we know little about the location of the corresponding posterior densities, we used during these first 2,000 MC steps, 100 sampling points regularly spaced on the corresponding parameter domain defined by the prior (PRIOR DENSITY). Figure 4A illustrates the energy evolution of the 10 different trials (realizations). The reader not familiar with our notion of energy (Eq. 16) can think of it as an indication of the quality of fit. The lower it is, the better the fit. Indeed, if we were working with a template-matching paradigm, the sum of the squared errors would be proportional to our energy. The striking feature on Fig. 4A is the presence of “meta-stable” states (a state is defined by a configuration and a value for each model parameter). The trials can spend many steps at an almost constant energy level before making discrete downward steps (like the one falling from 5,800 to 4,300 before the 500th step). If we look at the parameters values and configurations generated by the different trials (not shown), the ones that end above 5,000 “miss” the purple cluster on Fig. 2B. The time required to perform 2,000 MC steps with 7 neurons in the model was roughly 9 min, meaning that 1.5 h was required to obtain the data of Fig. 4A. This time (for 2,000 MC steps) grew from 8′ for a model with 4 neurons to 10′ for a model with 10 neurons.

THE REPLICA EXCHANGE METHOD SPEEDS UP CONVERGENCE IN THE PRESENCE OF META-STABLE STATES. The presence of meta-stable states is indeed a severe problem of our Markov chain dynamics, as further illustrated on the gray trace of Fig. 4B. Here the Markov chain was restarted from the state it had at the end of the 2,000 steps of the best among the 10 initial trials (gray trace on Fig. 4A). The last 1,000 steps of this trial were moreover used to locate 13 posterior conditional sampling points for each amplitude parameter of each neuron (AMPLITUDE PARAMETERS TRANSITION KERNELS). Then 5 different runs with 32,000 steps were performed. The best one is shown on the gray trace of Fig. 4B and the final mean energy of each of the 5 is shown on the right end of the graph (circles). What the reader sees here is a (very) slow convergence of a Markov chain to its stationary density. It is problematic because, if we stop the chain too early, our results could reflect more the properties of our initial guess than the properties of the target density we want to sample from. To circumvent this slow relaxation problem we have implemented a procedure commonly used in spin-glass (Hukushima and Nemoto 1996) and biopolymer structure (Hansmann 1997; Mitsutake et al. 2001) simulations: the replica exchange method (REM). The details of the methods are given in Slow relaxation and REM of the APPENDIX. The REM consists in simulations of “noninteracting” replicas of the same system (model) at different “inverse temperatures” ($\beta$) combined with exchanges between replicas states. These inverse temperatures are purely artificial and are defined by analogy with systems studied in statistical physics. The idea is that the system with a small $\beta$ value will experience little difficulty in crossing the energy barriers separating local minima (meta-stable states) and will therefore not suffer from the slow relaxation problem. The replica exchange dynamics will allow “cold” replicas (replicas with a larger $\beta$ value) to take profit of the “fast” state space exploration performed by the “hot” replica (small $\beta$). The efficiency of the method is demonstrated by the black trace on Fig. 4B. Here we again took the last state of the best trial of Fig. 4A to restart the Markov chain. We used 8 different $\beta$ values: 1, 0.8, 0.6, 0.5, 0.45, 0.4, 0.35, and 0.3. Five different runs with 4,000 steps were performed and the energy trace at $\beta = 1$ is shown in its integrity; this was the best of the 5. The final mean energy of the 5 trials is shown on the right side of the figure (triangles). The Markov chains simulated with the REM relax clearly faster than the ones simulated with the single replica method.
The number of MC steps has been chosen so that the same computational time (32') was required for both runs, the long one (32,000 steps) with a single replica at $\beta = 1$ (gray trace) and the short one (4,000 steps) with 8 replicas. The reader can remark that for these trials the time required to perform 2,000 MC steps for a single replica is 2', whereas it was previously much larger (9'). This is attributed to the reduction in the number of sampling points used for the posterior conditional, 13 instead of 100 (amplitude parameters transition kernels).

A significant amount of the computational time is therefore spent generating new amplitude parameters values.

**MODEL CHOICE.** Figure 5 illustrates what we meant by "empirical features associated with different models" at the beginning of this section. We have now performed the same series of runs with 7 different models having between 4 and 10 neurons. One feature we can look at in the output of these runs is the most likely classification produced (Empirical averages and SDs, Eq. 21). By "most likely" we mean that the label of each spike has been forced to its most likely value. Figure 5A shows one of the Wilson plots with different neuron numbers, where the events have been colored according to their most likely label. The good behavior of the algorithm appears clearly. When asked to account for the data with 6 neurons, it mainly lumps together the "small" neuron (yellow on Fig. 2B, neuron 6 in Table 1) with the "noise" neuron (black on Fig. 2B, neuron 7 in Table 1). With 7 neurons, the actual number of neurons, it splits the previously lumped clusters into 2 (although it attributes too many spikes to the small neuron and too few to the noise one; see Long runs with full data set). With 8 neurons, it further splits the noise cluster. In fact it creates a "noise" cluster (black on the figure) whose peak amplitude on the fourth recording site is roughly 3 times the recording noise SD and another one (clear blue) whose peak amplitude on this same site is smaller than 3. We remind the reader that our detection threshold is here set at 3 noise SD. The algorithm does not split the doublet-firing neuron (red) into 2 neurons, although this neuron gives rise to two well-separated clusters. With 9 neurons, it splits even further the previous clear blue "noise" neuron" into 2 neurons, one with a peak amplitude roughly equal to the detection threshold on the first recording site (pink cluster) and one whose peak amplitudes on sites 1 and 4 are smaller than 3. With 10 neurons (not shown) we end up with the actual noise neuron, resulting in 4 neurons with an average peak amplitude slightly above the detection threshold on one of the recording sites and very close to zero on the 3 others.

Another feature we can look at is the average values of the model parameters. Figure 5B shows the maximal peak amplitudes for 6 of the neurons and for the 4 different models considered in Fig. 5A. These maximal peak amplitudes are constrained by our priors to take values between 0 and 20. Each of the 4 semiaxes represents therefore the maximal peak amplitude on each of the 4 recording sites (see legend) and each frame represents one neuron in one model. For instance the blue frames correspond to neuron 1 in Table 1. Its maximal peak amplitudes on sites 1, 2, 3, and 4 are 15, 10, 5, and 0, respectively. The frames corresponding to this neuron in the different models considered overlap perfectly. The same holds for each neuron, except the yellow and the brown ones. The frame change for the "yellow" neuron is normal and is attributed to the fact that when we consider a model with a total of 6 neurons, the small neuron and the noise neuron get lumped together (Fig. 5A). It is only when we consider models with 7 neurons or more that the small neuron becomes (roughly) properly identified. That explains the variability in the yellow frames on Fig. 5B. The case of the brown neuron is a bit more subtle. This neuron fires very fast with ISIs between 8 and 15 ms (Fig. 9D) and does not generate ISIs large enough to properly explore its potential amplitude dynamic range. Stated differently, the amplitude parameters of this neuron are not well defined by the data (see as well Fig. 7). A more relevant feature for this neuron is the "typical" peak amplitudes produced. By that we mean that if we have estimates of the scale $s$ and shape $\sigma$ parameters of this neuron (10 ms and 0.2), we can compute the most likely ISI: $s \exp \left(-\sigma^2 \right) = 9.6$ ms. Then, the most likely peak amplitude is given by Eq. 2. It turns out that all the models considered gave the same and correct values for the 2 ISI density parameters (within error bars, not shown).

The models with 6, 7, and 9 neurons gave for the amplitude parameters ($P_1$, $\delta$, $\lambda$) of the brown neuron: 7.8, 0.45, 114 (the amplitudes on the different recording sites were the same within error bars and the parameters were the same, within error bars, across models, not shown), whereas the model with 8 neurons gave 9.9, 0.45, and 39. Then the most likely ampli-

**FIG. 5.** Model choice. A: most likely configurations generated by 4 of the 7 models studied are displayed on a Wilson plot where the amplitude on site 4 is plotted against the amplitude on site 1. Scale bars as in Fig. 2. B: maximal peak amplitude plots showing that neuron identification from model to model is easy. Each semiaxis corresponds to the maximal peak amplitude (estimated from the algorithm output) for each of the 6 “important neurons.” Vertical axis in the upper half-plan corresponds to the maximal peak amplitude on site 1. Horizontal axis in the right half-plan corresponds to the maximal peak amplitude on site 2. Vertical axis in the lower half-plan corresponds to the amplitude on site 3 and the horizontal axis in the left half-plan corresponds to the amplitude on site 4. For each semiaxis the amplitude value goes from 0 at the origin to 20 at the tip. Each neuron in each model corresponds to a frame. To make the figure more readable (by avoiding a too strong overlap of the frames) a random Gaussian noise ($\mu = 0$, $\sigma = 0.1$) has been added to the maximal peak amplitude values of each neuron.
tude for the models with 6, 7, or 9 neurons was 6.7, whereas it was 6.8 for the model with 8 neurons. That explains the larger frame of the brown neuron on Fig. 5B. In practice it turns out that the observation of the parameters alone is sufficient to find which neuron in a given model corresponds to which neuron in another model. Based on the considerations exposed in these last 2 paragraphs it seems reasonable to keep working with a model with 7 neurons, acknowledging the fact that the model cannot account very well for the noise neuron. In the sequel we will therefore proceed with the full data set (5,058 spikes) and consider only the model with 7 neurons.

**Long runs with full data set**

The analysis of the full data set (5,058 spikes or 15 s of data) was done in 2 stages. During the first 4,000 MC steps we fixed the model parameters at their mean values computed from the last 1,000 MC steps of the REM run with the reduced data set (Fig. 4B, black trace) and we updated only the configuration. The initial configuration was as usual, randomly set. We used the last 1,000 MC steps of this first stage to take “measurements” and compare the algorithm’s output with the actual values of different data statistics (Figs. 8 and 9). The “complete” algorithm was used during the 21,000 MC steps of the second stage. By complete algorithm we mean that at each MC step a new configuration was drawn as well as new values for the model parameters. The 13 sampling points of the posterior conditionals for the amplitude parameters between steps 4,001 and 6,000 were set using the last 1,000 MC steps of the REM run with the reduced data set. The last 2,000 MC steps of the second stage were used to take measurements. We use here a very long run to illustrate the behavior of our algorithm. Such long, and time-consuming, runs are not required for a daily use of our method. The time required to run the first stage (4,000 MC steps) was 3 h, whereas 26 h were required for the second (21,000 MC steps).

**THE REM REQUIRES MORE INVERSE TEMPERATURES WHEN THE SAMPLE SIZE INCREASES.** Figure 6A shows the evolution of the energies at the 15 inverse temperatures used. The energy overlap at adjacent temperatures is clear and is confirmed by the energy histograms of Fig. 6B. One of the shortcomings of the REM is that it requires more β to be used (for a given range) as the number of spikes in the data set increases because the width of the energy histograms (Fig. 6B) is inversely proportional to the square root of the number N, of events in the sample (Hansmann 1997; Hukushima and Nemoto 1996; Iba 2001). The necessary number of β grows therefore as $\sqrt{N}$ (if we had used the same 8 inverse temperatures as during our model exploration with a reduced data set, we would have only every second histogram). The computation time per replica grows, moreover, linearly with N. We therefore end up with a computation time of the REM growing like $N^{1.5}$. This justifies keeping the sample size small during model exploration. Figure 6C shows that the autocorrelation function of the energy at each temperature falls to zero rapidly (within 10 MC steps) which is an indication of the good performance of the algorithm from a statistical view point (Empirical averages and SDs). A pictorial way to think of the REM is to imagine that several “boxes” at different preset inverse temperatures are used and that there is one and only one replica per box. After each MC step, the algorithm proposes to exchange the replicas located in neighboring boxes (neighboring in the sense of their inverse temperatures) and this proposition can be accepted or rejected (Eq. A28). Then if the exchange dynamics works properly one should see each replica visit all the boxes during the MC run. More precisely, each replica should perform a random walk on the available boxes. Figure 6D shows the random walk performed by the replica which starts at $\beta = 1$. The ordinate corresponds to the box index (see legend). Between steps 1 and 25,000, the replica travels several times through the entire inverse temperature range.

**MODEL PARAMETERS EVOLUTION AND POSTERIOR ESTIMATES.** We have until now mainly shown parameters, like the energy of the first replica path, which are likely to be unfamiliar to our reader but our algorithm does generate as well outputs that should be more directly meaningful. We can for instance look at the evolution of the 8 parameters associated with each given neuron. The fluctuations of the parameters values around their means gives us the uncertainty of the corresponding estimates. As explained in Empirical averages and SDs, the accuracy of our estimates for a given number of MC steps will be better if the successive values of the corresponding parameters exhibit short autocorrelation times. We can as well use the algorithm output to build the posterior marginal density of each model parameter as shown on Fig. 7. It can be seen that most posterior densities are fairly close to a Gaussian except for 3 neurons whose amplitude parameters are not well defined by the data: neurons 4, 5, and 7. The case of neuron 4, the brown neuron on Fig. 2B, has already been discussed in Early exploration and model choice. The one of neuron 5 is a bit analogous, except that instead of not generating long enough ISIs it does not generate short enough ones (Fig. 9E) to fully explore the
amplitude dynamic range. Neuron 7 is the noise neuron and suffers basically from the same problem as the 2 others in addition to the more fundamental one that it is not well described by our model.

A more compact way to present the information of Fig. 7 is to summarize the distributions that are approximately Gaussian by their means and autocorrelation corrected SDs. The other distributions can be summarized by a 95% confidence interval, the left boundary being such that 2.5% of the generated parameters values were below it and the right boundary such that 2.5% of the generated values were above it, as shown in Table 2 (compare with the actual values of Table 1).

SPIKE CLASSIFICATION PERFORMANCES. We can now fully exploit the fact that we are working with simulated data and compare the classification produced by our algorithm with the actual one. To do that easily we first forced the soft classification generated by the algorithm, which gives for each spike the posterior probability to originate from each of the neurons in the model (Eq. 21) into the “most likely” classification (see MODEL CHOICE). Then 2 kinds of errors can be made: false negatives and false positives. A false negative (Fig. 8A) is a spike actually generated by neuron $j$, which ends up with the label $i \neq j$. A false positive (Fig. 8B) is the reverse situation, a spike that was generated by neuron $i \neq j$ and which ends up with label $j$. Two false positive and false negative values corresponding to the 2 measurements periods defined in Fig. 6A are given for each neuron on Fig. 8. Three features appear already clearly. First, the errors are smaller for the neurons that correspond to the model (neurons 1, 4, and 5) than for the others. Second, the difference between the 2 measurements periods is surprisingly small (given the significant difference in computation times). Third, most of the errors are attributed to noise events (from neuron 7) wrongly attributed to the small neuron (neuron 6), as expected from the Wilson plots. Figure 8C shows for each neuron, the sum of false positive and negative divided by the number of spikes actually generated by

![Fig. 7. Marginal posterior density estimate for each parameter of the model computed from the last 2,000 MC steps performed.](image)

### Table 2. Values of estimated parameters from the second measurement period

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Neuron 1</th>
<th>Neuron 2</th>
<th>Neuron 3</th>
<th>Neuron 4</th>
<th>Neuron 5</th>
<th>Neuron 6</th>
<th>Neuron 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$14.7 \pm 0.1$</td>
<td>$9.9 \pm 0.2$</td>
<td>$8.1 \pm 0.1$</td>
<td>$[7.43, 9.53]$</td>
<td>$2.9 \pm 0.1$</td>
<td>$3.1 \pm 0.7$</td>
<td>$0.8 \pm 0.2$</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$9.88 \pm 0.07$</td>
<td>$14.8 \pm 0.3$</td>
<td>$12.2 \pm 0.2$</td>
<td>$[7.42, 9.46]$</td>
<td>$8.2 \pm 0.1$</td>
<td>$4.1 \pm 0.4$</td>
<td>$0.6 \pm 0.4$</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$4.94 \pm 0.05$</td>
<td>$4.98 \pm 0.09$</td>
<td>$8.1 \pm 0.1$</td>
<td>$[7.44, 9.51]$</td>
<td>$14.0 \pm 0.3$</td>
<td>$5.6 \pm 0.7$</td>
<td>$0.4 \pm 0.4$</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$0.02 \pm 0.02$</td>
<td>$0.06 \pm 0.04$</td>
<td>$3.09 \pm 0.07$</td>
<td>$[7.42, 9.46]$</td>
<td>$18.9 \pm 0.1$</td>
<td>$4.3 \pm 0.7$</td>
<td>$0.7 \pm 0.3$</td>
</tr>
<tr>
<td>$\delta$</td>
<td>$[0.72, 0.89]$</td>
<td>$0.84 \pm 0.03$</td>
<td>$0.57 \pm 0.01$</td>
<td>$[0.46, 0.69]$</td>
<td>$[0.12, 0.88]$</td>
<td>$0.56 \pm 0.02$</td>
<td>$[0.1, 0.46]$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$[103, 128]$</td>
<td>$109 \pm 10$</td>
<td>$[36.8, 51.5]$</td>
<td>$[52, 192]$</td>
<td>$[174, 190]$</td>
<td>$[16.3, 41]$</td>
<td>$[10.5, 192]$</td>
</tr>
<tr>
<td>$s$</td>
<td>$19.3 \pm 0.2$</td>
<td>$12.5 \pm 0.2$</td>
<td>$21 \pm 2$</td>
<td>$0.94 \pm 0.08$</td>
<td>$190 \pm 8$</td>
<td>$18 \pm 2$</td>
<td>$14.4 \pm 0.9$</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>$0.31 \pm 0.01$</td>
<td>$0.65 \pm 0.02$</td>
<td>$0.90 \pm 0.05$</td>
<td>$0.198 \pm 0.006$</td>
<td>$0.34 \pm 0.03$</td>
<td>$0.7 \pm 0.1$</td>
<td>$1.21 \pm 0.06$</td>
</tr>
</tbody>
</table>

Each estimated value is given with its SD (Eq. 26) when the corresponding posterior density (Fig. 7) is close to a Gaussian. When such is not the case, a 95% confidence interval is given. The maximal peak amplitude values ($P_i$) are given in units of noise SD. The scale parameters ($s$) are given in milliseconds. The ISI density parameters ($\delta$ and $\sigma$) are those of the corresponding single log-normal density used by the model.
this neuron. Here again that gain in accuracy provided by the model does use a unimodal gram for the doublet-particular it generates the proper strongly bimodal generates very good estimates for the 5 good neurons. In actual histograms are shown on Fig. 9. Again, our algorithm neuron during the 2 periods. The results together with the bias in the estimate. We have done this computation for each gram for each neuron because this latter method introduces a

FIG. 8. Comparison between the actual configuration and the most likely configuration generated by the algorithm. Two measurements periods are considered, the first one between steps 3,001 and 4,000 (Fig. 6A) represented here in black and the second period (from step 23001 to step 25000) represented in gray. A: number of false negative for each neuron. B: number of false positive for each neuron. C: fraction of wrongly classified spikes for each neuron.

de the neuron. The luxury simulation performed with the complete algorithm can be seen to decrease the fraction of misclassified spikes between the small neuron (6) and the noise neuron (7), as well as the fraction of misclassified spikes for the doublet-firing neuron (3). Overall, if we consider the full data set, we end up with 10.4% of the spikes misclassified from the first measurements period and 8.7% from the second one. If we (wisely) decide that neurons 6 and 7 cannot be safely distinguished and decide to keep only the first 5 neurons we misclassify 3.9% of the spikes during the first period and 3.5% during the second.

ISI HISTOGRAMS ESTIMATES. The last statistics shown are the ISI histograms. Our algorithm generates at each MC step a new configuration, that is, a new set of labels for the spikes. An ISI histogram can therefore be computed for each neuron after each algorithm step. Then the best estimate we can get for an actual histogram (i.e., the histogram obtained using the true labels) is the average of the histograms computed at each step. This is better than getting first the most likely configuration (as we did in the previous paragraph) and then the histogram for each neuron because this latter method introduces a bias in the estimate. We have done this computation for each neuron during the 2 periods. The results together with the actual histograms are shown on Fig. 9. Again, our algorithm generates very good estimates for the 5 good neurons. In particular it generates the proper strongly bimodal ISI histogram for the doublet-firing neuron (Fig. 9C), whereas the model does use a unimodal log-Normal ISI density to describe this neuron. Here again that gain in accuracy provided by the very long run with the complete algorithm is rather modest.

DISCUSSION

We have described an application of the MCMC methodology to the spike-sorting problem. This approach was recently applied to the same problem (Nguyen et al. 2003) but with a
ersimpler data generation model, which led to a simpler algorithm because successive spikes were considered as independent. We have shown here that a MCMC algorithm allows the neurophysiologist to use much richer data generation models than he/she could previously afford, thereby allowing him/her to include a description of the ISI density as well as a description of the spike waveform dynamics. For the sake of illustration we have used in this paper simple but nontrivial models of ISI densities and waveform dynamics. It should nevertheless be clear that the MCMC approach is very flexible (perhaps too flexible). It is for instance very easy to include more sophisticated models for the neuronal discharge based on hidden Markov chains, like the one underlying the doublet-firing neuron of the present paper (Delescluse et al. unpublished observations). We have shown as well for the first time as far as we know, how to estimate both a probability density for the spike train configuration (i.e., the soft classification) and a probability density for the model parameters. The way to use the soft classification to compute statistics of interest (e.g., ISI histograms) was moreover illustrated.

We have shown the presence of meta-stable states, which are, we think, an intrinsic property of models with “interacting” spikes. We have illustrated the potential problems resulting from these meta-stable states, which could introduce a bias in our algorithm’s output. However, using a powerful methodology recently introduced in statistics and computational physics, the replica exchange method, we were able to circumvent the slow relaxation resulting from the meta-stable states.

We have deliberately used simulated data that did not correspond to our model assumptions, but which were hopefully realistic enough. Our algorithm with its “simple” data generation model performed very well for the neurons that generated events larger than the detection threshold, even for a neuron which
generated two separated clusters on the Wilson plots. As far as we know, no other algorithm would automatically properly cluster the events of such a neuron. Clearly, such a performance is possible only if both the spike occurrence times and amplitude dynamics are taken into account. The robustness of the algorithm’s output with respect to overclustering was moreover demonstrated. That being said, if a recording noise model based on a Student’s $t$ distribution turns out to be better than a Gaussian model, our algorithm can be trivially modified to work with the former. It can be easily modified as well to include an explicit “noise neuron” (i.e., a neuron with an exponential ISI density and a maximal amplitude below the detection threshold on all recording sites). We permit reader to download our implementation of the algorithm and check that an almost perfect classification (and model parameter estimation) can be obtained for data that correspond to the model, even when strong overlap between clusters are observed on the Wilson plots.

We have considered spikes described by their peak amplitude(s) instead of their waveforms, although it would not require a tremendous algorithmic change to deal with waveforms (see for instance Pouzat et al. 2002). Caution would nevertheless be required to deal with superpositions of spikes, but as long as spikes do not occur exactly at the same time, superpositions will show up as “interaction” terms. That is, Eq. 2 would have to be modified to include a baseline change because of neighboring spikes.

The major drawback of our algorithm is the rather long computational time it requires. It should nevertheless be clear that the user can choose between different implementations of our approach. A fast one, where the model parameters are set during a prerun with a reduced data set, leads to an already very accurate classification of spikes from “good” neurons (a good neuron generates events clearly above the detection threshold). We think that this use of our method should satisfy most practitioners. For preliminary tests on a Beowulf cluster consuming 10.220.32.247 on June 22, 2017 http://jn.physiology.org/ Downloaded from
\[ A[\theta_{\text{propose}}, C_{\text{propose}}]\theta^{(i-1)}, C^{(i-1)} = \min \left\{ 1, \frac{\pi_{\text{post}}(\theta_{\text{propose}}, C_{\text{propose}}) \mathbb{P}(\theta^{(i-1)}, C^{(i-1)} | \theta_{\text{propose}}, C_{\text{propose}})}{\pi_{\text{post}}(\theta^{(i-1)}, C^{(i-1)})} \right\} \] 

(A.3)

It is not hard to show that the transition kernel induced by this algorithm has the desired properties (Liu 2001; Robert and Casella 1999). The important feature of this procedure is that its implementation requires only a knowledge of \( \pi_{\text{post}} \) up to a normalizing constant because Eq. A.3 involves a ratio of \( \pi_{\text{post}} \) values in 2 states. In other words it requires only the capability to compute the energy of any state. An initial version of this algorithm was given by Metropolis et al. (1953) and the present version is attributed to Hastings (1970).

The reader can see that the Metropolis–Hastings (MH) algorithm is indeed a very large family of algorithms because there exists a priori an infinite number of transitions \( g \) from which a specific procedure can be built. We therefore have to find at least one \( g \) and if we find several we would like to have a way to compare the resulting MH transition kernels \( T \). A common way to proceed for complicated models, like the one we are presently dealing with, is to split the transition kernel \( T \) in a series of parameter and label specific transition kernels (Besag 1981; Fishman 1996). Let us define

\[ \theta_{\text{par}} = (\theta_1, \ldots, \theta_i, \ldots, \theta_p) \tag{A4} \]

where \( n_p \) is the number of parameters in the model and

\[ C_{\text{par}} = (c_1, \ldots, c_i, c_{i+1}, \ldots, c_d) \tag{A5} \]

Then the parameter-specific transition kernels are objects like

\[ T_\theta(\theta_{\text{par}}, \theta = b, C_{\text{par}}, \theta = a, C) \tag{A6} \]

and the label specific transition kernels

\[ T_{C_{\text{par}}}(C_{\text{par}}, C_i = b, C, C_{\text{par}}, C_i = a) \tag{A7} \]

We will often use the following short notation for Eqs. A6 and A7

\[ T_{\theta\gamma}(\theta = b, \gamma = a) = T_\theta(\theta_{\text{par}}, \theta = b, C_{\text{par}}, \theta = a, C) \tag{A8} \]

and

\[ T_{C_i}(C_i = b, C) = T_{C_i}(C_{\text{par}}, C_i = b, C_{\text{par}}, C_i = a) \tag{A9} \]

A “complete” transition kernel can be constructed with, for instance, a sequential application of every label and parameter specific transition

\[ T = T_{C_{\text{par}}} \cdot T_{C_{\text{par}}} \cdot T_{\theta_{\text{par}}} \cdot T_{\theta_{\text{par}}} \tag{A10} \]

In the sequel we will say that one MC time step has been performed every time a complete transition kernel has been applied to the “present” state of the Markov chain. Sufficient conditions the parameter and label specific transitions must exhibit for the complete transition \( T \) to be ergodic and for \( \pi_{\text{post}} \) to be stationary with respect to \( T \) are as follows:

1. Each parameter and label specific transition \( T_{\theta_{\text{par}}}, T_{C_{\text{par}}} \) is such that

\[ T_\theta(\theta = b, \gamma = a) \neq 0 \tag{A11} \]

for all pairs of states \((\theta_{\text{par}}, \gamma = a, C)\) and \((\theta_{\text{par}}, \gamma = b, C)\) such that

\[ \pi_{\text{post}}(\theta_{\text{par}}, \gamma = a, C, Y) > 0 \quad \text{and} \quad \pi_{\text{post}}(\theta_{\text{par}}, \gamma = b, C, Y) > 0 \tag{A12} \]

(1) It is straightforward to write the equivalent condition for \( T_{C_{\text{par}}} \)

2. Each parameter and label specific transition \( T_{\theta_{\text{par}}}, T_{C_{\text{par}}} \) is such that the detailed balance condition is met

\[ \pi_{\text{post}}(\theta_{\text{par}}, \gamma = a, C)Y T_{\theta_{\text{par}}}(\theta = b|\theta = a) \]

\[ = \pi_{\text{post}}(\theta_{\text{par}}, \gamma = b, C)Y T_{\theta_{\text{par}}}(\theta = a|\theta = b) \tag{A13} \]

Moreover, we have

\[ \pi_{\text{post}}(\theta_{\text{par}}, \gamma = a, C, Y) = \pi_{\text{post}}(\theta_{\text{par}} = a|\theta_{\text{par}}, C, Y) \pi_{\text{post}}(\theta_{\text{par}}, C, Y) \]

and Eq. A12 implies that

\[ \pi_{\text{post}}(\theta_{\text{par}} = a|\theta_{\text{par}}, C, Y) > 0 \quad \text{and} \quad \pi_{\text{post}}(\theta_{\text{par}}, C, Y) > 0 \]

We can therefore rewrite the detailed balance condition as follows

\[ \pi_{\text{post}}(\theta_{\text{par}} = a|\theta_{\text{par}}, C, Y) Y T_{\theta_{\text{par}}}(\theta = b|\theta = a) \]

\[ = \pi_{\text{post}}(\theta_{\text{par}} = b|\theta_{\text{par}}, C, Y) Y T_{\theta_{\text{par}}}(\theta = a|\theta = b) \tag{A14} \]

where \( \pi_{\text{post}}(\theta_{\text{par}}|\theta_{\text{par}}, C, Y) \) is the posterior conditional density of \( \theta \). There are, of course, equivalent equations for the label-specific transition kernels \( T_{C_{\text{par}}} \).

A straightforward way to obtain \( T_{\theta_{\text{par}}} \) and \( T_{C_{\text{par}}} \) with the desired properties is to find parameter- and label-specific proposal transition kernels \( (g_{\theta_{\text{par}}}, g_{C_{\text{par}}}) \) respecting Eq. A11 in combination with an acceptance probability like Eq. A3, that is, to have an MH rule for each parameter of the model and for each label of the configuration. It is indeed easy to show that such a construct gives rise to \( T_{\theta_{\text{par}}} \) and \( T_{C_{\text{par}}} \), respecting the detailed balance condition because, assuming \( a \neq b \)

\[ \pi_{\text{post}}(\theta_{\text{par}} = a, C, Y) Y T_{\theta_{\text{par}}}(\theta = b|\theta = a) \]

\[ = \pi_{\text{post}}(\theta_{\text{par}} = b, C, Y) Y T_{\theta_{\text{par}}}(\theta = a|\theta = b) \tag{A15} \]

\[ = \min \left[ \pi_{\text{post}}(\theta_{\text{par}} = a, C, Y)Y g_{\theta_{\text{par}}}(\theta = b|\theta = a) \right] \]

\[ = \pi_{\text{post}}(\theta_{\text{par}} = b, C, Y)Y g_{\theta_{\text{par}}}(\theta = a|\theta = b) \]

\[ = \pi_{\text{post}}(\theta_{\text{par}} = a, C, Y)Y T_{\theta_{\text{par}}}(\theta = b|\theta = a) \]

\[ = \pi_{\text{post}}(\theta_{\text{par}} = b, C, Y)Y T_{\theta_{\text{par}}}(\theta = a|\theta = b) \tag{A16} \]

To complete this justification of this parameter and label specific decomposition the reader needs to realize that the detailed balance condition is stronger than the stationarity condition (Eq. A1) and in fact implies it as shown in the following paragraph.

The detailed balance condition imposed on the parameter specific transition kernels implies the stationarity condition on the resulting “complete” transition kernel. We present here the justification for a simple model with 2 parameters \((\theta_1, \theta_2)\) and without latent variable; the extension to the full model considered in this paper being straightforward. So we have

\[ T = T_{\theta_{\text{par}}} \cdot T_{\theta_{\text{par}}} \]
The acceptance probability requires the calculation of the ratio of the posterior conditionals: \( \pi_{post}(c_i = 1|\theta, C_{-i}, Y) \) and \( \pi_{post}(c_i = 2|\theta, C_{-i}, Y) \). From Eq. 15 we have

\[
\pi_{post}(c_i = 1|\theta, C_{-i}, Y) \propto L(Y, C_{-i}, c_i = 1|\theta) \pi_{prior}(\theta)
\]

and

\[
\pi_{post}(c_i = 2|\theta, C_{-i}, Y) \propto L(Y, C_{-i}, c_i = 2|\theta) \pi_{prior}(\theta)
\]

Therefore

\[
A_{C_i}(c_i = 2|c_i = 1) = \min \left\{ 1, \frac{L(Y, C_{-i}, c_i = 2|\theta)}{L(Y, C_{-i}, c_i = 1|\theta)} \right\}
\]

We will illustrate the computation on the simple case presented in likelihood function and Fig. 1, with one recording site and 2 neurons in the model (\( K = 2 \)). Because \( c_i \) can only have 2 values, \( c_i = 1 \) and \( c_i = 2 \), the matrix \( G_{C_i} \), is straightforward to write

\[
G_{C_i} = \begin{pmatrix} 0 & 1 \\
1 & 0 \end{pmatrix}
\]

The acceptance probability requires the calculation of the ratio of the posterior conditionals: \( \pi_{post}(c_i = 1|\theta, C_{-i}, Y) \) and \( \pi_{post}(c_i = 2|\theta, C_{-i}, Y) \). From Eq. 15 we have

\[
\pi_{post}(c_i = 1|\theta, C_{-i}, Y) \propto L(Y, C_{-i}, c_i = 1|\theta) \pi_{prior}(\theta)
\]

and

\[
\pi_{post}(c_i = 2|\theta, C_{-i}, Y) \propto L(Y, C_{-i}, c_i = 2|\theta) \pi_{prior}(\theta)
\]

Therefore

\[
A_{C_i}(c_i = 2|c_i = 1) = \min \left\{ 1, \frac{L(Y, C_{-i}, c_i = 2|\theta)}{L(Y, C_{-i}, c_i = 1|\theta)} \right\}
\]

We then assume for instance that

\[
\frac{L(Y, C_{-i}, c_i = 2|\theta)}{L(Y, C_{-i}, c_i = 1|\theta)} = 0.75
\]

we get the following transition matrix

\[
T_{C_i} = \begin{pmatrix} 0.25 & 0.75 \\
0.75 & 1 \end{pmatrix}
\]

In the case of Fig. 1, if we consider \( i = 13 \), we get, using Eq. 5

\[
L(Y, C_{-i}, c_i = 1|\theta) = M(t_{i1} - t_{i2}, a_{i1}|\sigma_i, \delta_i, P_i, \delta_i, \lambda_i) \times \pi(t_{i1} - t_{i2}, a_{i1}|\sigma_i, \delta_i, P_i, \delta_i, \lambda_i)
\]

and

\[
L(Y, C_{-i}, c_i = 2|\theta) = M(t_{i1} - t_{i2}, a_{i1}|\sigma_i, \delta_i, P_i, \delta_i, \lambda_i) \times \pi(t_{i1} - t_{i2}, a_{i1}|\sigma_i, \delta_i, P_i, \delta_i, \lambda_i)
\]
\[ L(Y, C_{-i}, c_i) = 2 \theta \]

where \( M \) represents all the terms that are identical in the two equations.

A remark on numerical implementation. To avoid rounding errors and keep numerical precision we work with the log-likelihoods

\[ l_b = \log L(Y, C_{-i}, c_i) = \beta \theta \]

look for \( l_{\text{max}} \), the maximum (with respect to \( b \)) of \( l_b \) and obtain \( \pi_{\text{post}}(c_i = b|\theta, C_{-i}, Y) \)

\[ \pi_{\text{post}}(c_i = b|\theta, C_{-i}, Y) = \frac{\exp(l_b - l_{\text{max}})}{\sum_{c_i} \exp(l_b - l_{\text{max}})} \]

A proposed value \( b \in \{1, \ldots, K\}\) drawn from the probability mass \( p_{c_i}(c_i = b|c_i = a) \) using the inverse cumulative distribution function (Fishman 1996; Gelman et al. 1995; Robert and Casella 1999).

Boundary treatment. From the statistician's viewpoint the spike trains are being dealt with are censored data. They mean by that that there are missing data without which we cannot, strictly speaking, compute our likelihood function. The missing data here are the last spikes from each neuron that occurred before our recording started and the next spikes after our recording stopped. There are proper approaches to deal with such censored data (Gelman et al. 1995; Robert and Casella 1999) and we plan to implement them in a near future. In the present study we used a simple and quick way to deal with the end effects in our data. We used periodic boundary conditions; that is, the last spike from each neuron was considered as the spike preceding the first spike of the same neuron.

Scale parameter transition kernel. We consider here cases where \( \theta_i \) of Eq. A14 corresponds to one of the \( K \) scale parameters, \( s_{iq} \) where \( q \) stands for the neuron index. We will first show that the corresponding posterior conditional: \( \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}, Y|C) \) can be written in a closed form. From Eq. 1S we have

\[ \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}, Y|C) \propto L(Y, C|\theta = s_{[\theta_{-i}, \theta_i]} \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}) \]

Going back to the structure of the likelihood function (Eq. 9), one sees, using Eqs. 5 and 10, that the likelihood function with the labels set and all model parameters set except the scale parameter of one neuron \( q \) depends only on the events that are labeled as belonging to neuron \( q \), and on the shape parameter of \( \sigma_q \). If we write \( \{i_{q,1}, \ldots, i_{q,n_q}\} \), the ISI computed from the spike train of neuron \( q \) in configuration \( C \) we get

\[ \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}, Y|C) \propto \exp \left[ -\frac{1}{2} \sum_{j=1}^{n_q} \left( \frac{\log i_{q,j} - \log s_{iq}}{\sigma_q} \right)^2 \right] \]

If we introduce: \( \log i_{q,j} = \ln \sum_{j=1}^{n_q} \log i_{q,j} \); a short manipulation shows that

\[ \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}, Y|C) \propto \exp \left[ -\frac{1}{2} \sum_{j=1}^{n_q} \left( \frac{\log i_{q,j} - \log s_{iq}}{\sigma_q} \right)^2 \right] \]

In words: the posterior density of the log of the scale parameter \( s_{iq} \) is Normal with mean: \( \log i_{q,j} \) and variance, \( \sigma_q^2/n_q \). Our prior on \( s_{iq} \) is a uniform distribution between \( s_{iq} \) and \( s_{iq} \) (PRIOR DENSITY); we can therefore, using the Gibbs algorithm, generate \( s_{iq} \) as follows: 1.) Given \( C \) and \( Y \) compute \( n_q \) and \( \log i_{q,j} \) 2.) Generate \( u \sim \text{Norm} [\log i_{q,j}, (\sigma_q^2/n_q)] \) 3.) Compute \( s_{iq} = \exp(u) \) 4.) If \( s_{iq} \leq s_{iq} \text{ or } s_{iq} \), update \( s_{iq} = s_{iq} \); otherwise, go back to 2

Shape parameter transition kernel. Here \( \theta_i \) is the shape parameter of neuron \( q \); \( \theta_i = \sigma_q \). We again have to show that the posterior conditional can be written in a closed form. Using the approach of the previous section we quickly get

\[ \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}, Y|C) \propto \exp \left[ -\frac{1}{2} \sum_{j=1}^{n_q} \left( \frac{\log i_{q,j} - \log s_{iq}}{\sigma_q} \right)^2 \right] \]

Then an identification with an inverse-Gamma distribution gives us the answer

\[ \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}, Y|C) \propto s_{iq}^{n_q} \exp \left[ -\frac{s_{iq}}{\sigma_q} \right] \]

and it is not hard to show that if \( s_{iq} \) is Gamma with parameters \( \alpha \) and \( \gamma \), then \( 1/s_{iq} \) is inverse-Gamma with the same parameters. To generate \( s_{iq} \) using the Gibbs algorithm, we therefore perform

1.) Given \( s_{iq} \) and \( Y \) compute \( n_q \) and \( \log i_{q,j} \) 2.) Generate \( \sigma_q \sim \text{Gamma} [n_q/2 - 1, n_q/2(\log i_{q,j} - \log s_{iq})^2] \)

and \( \sigma_q \) again as follows: 1.) Given \( C \) and \( Y \) compute \( n_q \) and \( \log i_{q,j} \) 2.) Generate \( \sigma_q \sim \text{Gamma} [n_q/2 - 1, n_q/2(\log i_{q,j} - \log s_{iq})^2] \)

where we are assuming we are dealing with a tetrode recording. These un normalized posterior densities are then normalised such that the piecewise linear function:

\[ s_{iq}(P) = \pi_{\text{post}}(\theta_{[\theta_{-i}, \theta_i]}, Y|C) \]

\[ + \pi_{\text{post}}(\theta_{[\theta_{-i}, \theta_i]}, Y|C) \]

\[ \pi_{\text{post}}(\theta = s_{[\theta_{-i}, \theta_i]}, Y|C) \]

where \( P \in \{P_{iq}, P_{iq+1}\} \) sums to 1. Of course when we start the algorithm we do not know what is the best location for the discrete points \( P_{iq} \). We therefore used during a first run 100 regularly spaced points, although it turns out that computing at each MC step 100 values of the posterior conditional (for each amplitude parameter of the model) takes time. It is therefore very interesting, to speed up the sorting procedure, to use fewer points, which basically means forgetting about the points located where the posterior conditional is extremely small. To achieve this goal, at the end of each run, the last 1000 generated values of the parameter of interest \( P_{iq} \) were used to get an estimate of the posterior or marginal cumulative distribution of the parameter. Then 10 quantiles were located on this cumulative distribution, one discrete point was taken slightly smaller than the smallest generated value for that parameter, and the two extreme values al-
lowed by the priors were used as discrete points as well. The new piecewise proposal kernel was therefore based on a sparser approximation.

Slow relaxation and REM

One of the nice features of the MCMC approach, and in fact the key theoretical result justifying its growing popularity, is that the user can be sure that empirical averages (Eq. 19) computed from the samples it generates do converge to the corresponding expected values (Eq. 18). The not-so-nice feature, forgetting about the actual implementation details, which can be tedious, is that the user can run his/her MCMC algorithm only for a finite amount of time. Then the fact that empirical averages are approximations cannot be ignored. There are 3 sources of errors in such estimates. The first that comes to mind is the one resulting from the finite sample size, that is, the one we would have even if our draws where directly generated from \( \pi_{\text{post}} \). The difference between the MCMC-based estimates and estimates based on direct samples is that the variance of the estimators of the former have to be corrected to take into account the correlation of the states generated by the Markov chain, as explained in Empirical averages and SDs. The second source of error is a bias induced by the state \([\theta^{(0)}, C^{(0)}] \) with which the chain is initialized (Fishman 1996; Sokal 1989). The bad news concerning this source of error is that there is no general theoretical result providing guidance on the way to handle it, but the booming activity in the Markov chain field already produced encouraging results in particular cases (Liu 2001). The common wisdom in the field is to monitor parameters (and labels) evolution, and/or their functions like the energy (Eq. 16). Based on examination of evolution plots (e.g., Fig. 4, A and B, Fig. 6a) and/or application of time-series analysis tools, the user will decide that “equilibrium” has been reached and discard the parameters values before equilibrium. More sophisticated tests do exist (Robert and Casella 1999) but they were not used in this paper. These first 2 sources of error are common to all MCMC approaches. The third one appears only when the energy function exhibits several local minima. In the latter case, the Markov chain realization can get trapped in a local minimum, which could be a poor representation of the whole energy function. This sensitivity to local minima arises from the local nature of the transitions generated by the MH algorithm. That is, in our case, at each MC time step, we first attempt to change the label of spike 1, then the one of spike 2, and so forth, then the one of spike \( N \), then we try to change the first component of \( \theta^{(1)} \), and so on until the last component \( \theta^{(n)} \). That implies that if we start in a local minimum and if we need to change, say, the labels of 10 spikes to reach another lower local minimum, we could have a situation in which the first 3 label changes are energetically unfavorable (giving, for instance, an acceptance probability, Eq. A3, of 0.1 per label change), which would make the probability to accept the succession of changes very low (0.1), meaning that our Markov chain would spend a long time in the initial local minimum before “escaping” to the neighboring one. Stated more quantitatively, the average time the chain will take to escape from a local minimum with energy \( E_{\text{min}} \) grows as the exponential of the energy difference between the energy \( E^* \) of the highest energy state the chain has to go through to escape and \( E_{\text{min}} \):

\[
\tau_{\text{escape}} \propto \exp[\beta(E^* - E_{\text{min}})]
\]

Our chains will therefore exhibit an Arrhenius behavior. To sample more efficiently such state spaces, the replica exchange method (REM) (Hukushima and Nemoto 1996; Mitsutake et al. 2001), also known as the parallel tempering method (Falcioni and Deem 1999; Hansmann 1997; Yan and de Pablo 1999), considers \( R \) replicas of the system with an increasing sequence of temperatures (or a decreasing sequence of \( \beta \)) and a dynamic defined by 2 types of transitions: usual MH transitions performed independently on each replica according to the rule defined by Eq. A10 and a replica exchange transition. The key idea is that the high temperature (low \( \beta \)) replicas will be able to easily cross the energy barriers separating local minima (in the example above, if we had a probability of 0.1 to accept a sequence of labels switch for the replica at \( \beta = 1 \), the replica at \( \beta = 0.2 \) will have a probability 0.1 \( \times 0.2 = 0.2 \) to accept the same sequence). What is needed is a way to generate replica exchange transitions such that the replica at \( \beta = 1 \) generates a sample from \( \pi_{\text{post}} \) defined by Eq. 17. Formally the REM consists in simulating, on an “extended ensemble,” a Markov chain whose unique stationary density is given by

\[
\pi_r(\theta, C_1, \ldots, \theta, C_R) = \pi_{\text{post}}(\theta_1, C_1) \cdots \pi_{\text{post}}(\theta_R, C_R)
\]

where “ee” in \( \pi_r \) stands for “extended ensemble” (Iba 2001). \( R \) is the number of simulated replicas, \( \beta_1 > \cdots > \beta_R \) for convenience, and

\[
\pi_{\text{post}}(\theta, C) = \frac{\exp[\beta E(\theta, C)]}{Z(\beta)}
\]

That is, compared to Eq. 17, we now explicitly allow \( \beta \) to be different from 1. To construct our “complete” transition kernel we apply our previous procedure; that is, we construct it as a sequence of parameter, label, and interreplica-specific MH transitions. We already know how to obtain the parameter and label specific transitions for each replica. What we really need is a transition to exchange replicas, say the replicas at inverse temperature \( \beta \) and \( \beta_{i+1} \), such that the detailed balance is preserved (Eq. A13)

\[
\pi_r(\theta, C_1, \ldots, \theta, C_R)T_{i+1}\left[\theta, C_i, \theta, C_{i+1}\right] = \pi_r(\theta, C_1, \ldots, \theta, C_{i+1}, \theta, C_i)T_i\left[\theta, C_i, \theta, C_{i+1}\right]
\]

which leads to

\[
T_{i+1}\left[\theta, C_i, \theta, C_{i+1}\right] = \pi_i(\theta, C_1, \ldots, \theta, C_i, \theta, C_{i+1}, \theta, C_R) T_i\left[\theta, C_i, \theta, C_{i+1}\right]
\]

Again we write \( T_{i+1} \) as a product of a proposal transition kernel and an acceptance probability. Here we have already explicitly chosen a deterministic proposal (we only propose transitions between replicas at neighboring inverse temperatures), which gives us

\[
A_{i+1}\left[\theta, C_i, \theta, C_{i+1}\right] = \exp[-(\beta - \beta_{i+1})E(\theta, C_i) - E(\theta, C_{i+1})]
\]

It is therefore enough to take

\[
A_{i+1}\left[\theta, C_i, \theta, C_{i+1}\right] = \min(1, \exp[-(\beta - \beta_{i+1})E(\theta, C_i) - E(\theta, C_{i+1})])
\]

The reader sees that if the state of the “hot” replica (\( \beta_{i+1} \)) has a lower energy \( E(\theta, C_i) \) than the “cold” one, the proposed exchange is always accepted. The exchange can pictorially be seen as cooling down the hot replica and warming up the cold one. Fundamentally this process amounts to making the replica that is at the beginning and at the end of the replica exchange transition at the cold temperature to jump from one local minimum (\( \theta, C_{i+1} \)) to another one (\( \theta, C_i \)). That is precisely what we were looking for. The fact that we can as well accept unfavorable exchanges (i.e., raising the energy of the
“cold” replica and decreasing the one of the “hot” replica) is the price we have to pay for our algorithm to generate samples from the proper posterior (we are not doing optimization here).

For the replica exchange transition to work well we need to be careful with our choice of inverse temperatures. The typical energy of a replica (i.e., its expected energy) increases when careful with our choice of inverse temperatures. The typical energy of exchanges will practically never be accepted. To the exchange. That implies that the acceptance ratio (Eq. A28) for the replica exchange will be typically smaller than 1. Obviously, if it becomes too small, exchanges will practically never be accepted. To avoid this situation we need to choose our inverse temperatures such that the typical product: $\Delta E \Delta \beta$, where $\Delta E = \beta_i \cdot \beta_{i+1}$ is close enough to zero (Hukushima and Nemoto 1996; Iba 2001; Neal 1994). In practice we used preruns with an a priori too large number of $\beta$s, checked the resulting energy histograms and kept enough inverse temperatures to have some overlap between successive histograms (Fig. 6B).

The replica exchange transitions were performed between each pair $\beta_i$, $\beta_{i+1}$ with an even, respectively odd, $i$ at the end of each even, respectively odd, MC time step. With the replica exchange scheme, each MC time step was therefore composed of a complete parameter and label transition for each replica, followed by a replica exchange transition. This scheme corresponds to the one described by Hukushima and Nemoto (1996). A rather detailed account of the REM can be found in Mitsutake et al. (2001). Variations on this scheme do exist (Celleux et al. 2000; Neal 1994).

Random number generators for the simulated data

A vector $n$ from an isotropic $t$ density with $n > 2$ degrees of freedom and a $\delta$ uniform to 1 (for each component) was generated with the following algorithm (Fishman 1996; Gelman et al. 1995):

1) Generate: $z \sim \text{Norm}(0, 1)$
2) Generate: $x = z^2 + 2z$ $z$
3) Take: $n = \sqrt{z(1-z^2)}$

The Normal and $\chi^2$ (pseudo-)random number generators are built-in features of Scilab (and MATLAB).

An $isi$ from a log-Normal density with scale parameter $s$ and shape parameter $\sigma$ was generated with the following algorithm:

1) Generate: $z \sim \text{Norm}(0, \sigma^2)$
2) Take: $isi = s \exp(z)$

For the neuron with a Gamma $ISI$ density, the Gamma random number generator of Scilab was used.

Acknowledgments

We thank C. van Vreeswijk, N. Brunel, O. Mazor, L. Forti, A. Marty, J. Chavas, and M.-G. Lucas for comments and suggestions on the manuscript.

Grants

This work was supported in part by a grant from the Ministère de la Recherche (ACI Neurosciences Intégratives et Computationnelles, projet-protocole, 2001–2003), by a grant from inter-Établissements Publics Scientifiques et Techniques (Bioinformatique), and by the région Île-de-France (programme SéSAME). M. Delescluse was supported by a fellowship from the Ministère de l’Éducation National et de la Recherche.

References


