Generation of correlated spike trains

Romain Brette
Odyssee Lab (ENPC Certis/ENS Paris/INRIA Sophia)
Département d'Informatique
Ecole Normale Supérieure
45, rue d'Ulm
75230 Paris Cedex 05
France

email: brette@di.ens.fr
tel: +33 1 44 32 21 76
fax: +33 1 44 32 21 56

February 11, 2008

Abstract

Neuronal spike trains display correlations at diverse time scales throughout the nervous system. The functional significance of these correlations is largely unknown, and computational investigations can help us understand their role. In order to generate correlated spike trains with given statistics, several case-specific methods have been described in the litterature. In this paper I present two general methods to generate sets of spike trains with given firing rates and pairwise correlation functions, along with efficient simulation algorithms.

Keywords: spike trains, correlations, numerical methods, algorithms, Cox processes

1 Introduction

Neuronal synchronization is ubiquitous in the central nervous system (Salinas & Sejnowski, 2001), which raises numerous questions on its signifance, e.g. what is the effect of correlations in the thalamus (Usrey *et al.*, 2000) on cortical function? or, at the cellular level, how sensitive are cortical cells to correlations in their inputs? (Salinas & Sejnowski, 2000; Konig *et al.*, 1996; Rudolph & Destexhe, 2003; Roy & Alloway, 2001). To address these questions in computational or theoretical studies, several authors have devised case-specific methods of constructing correlated spike trains. They can be classified in two categories.

The first approach consists in generating spike trains as inhomogenous Poisson processes with a common time-varying rate. Then correlations between spike trains arise from the autocorrelation of the rate. Examples in the litterature include sinusoidal rates (Salinas & Sejnowski, 2001) and jump processes (Song *et al.*, 2000; Song & Abbot, 2001). In all cases, the generated spike trains were limited to sets of spike trains with identical rates and identical pair-wise correlation functions.

The second approach consists in generating correlated spike trains by picking spikes randomly from a set of common spike trains. Essentially two cases have been described in the litterature: each spike train is a random subset of a common spike train (Feng & Brown, 2000) (Destexhe and Pare (1999) use a pool of spike trains but the algorithm is equivalent), or each spike train is the union of a common spike train and of an independent spike train (Gutig *et al.*, 2003; Kuhn *et al.*, 2003). In both cases, the generated spike trains have the same rates and the same pair-wise correlations, and the cross-correlation functions are most often delta functions.

In this paper I generalize both approaches to generate sets of correlated spike trains with arbitrary rates and pair-wise cross-correlation functions. The first method (section 3) defines spike trains as doubly stochastic processes, also known as Cox

processes (Daley & Vere-Jones, 2005). The second method (section 4) generates spike trains from random mixtures of source spike trains — I shall call it the *mixture method*. Before I present the methods in details, I will start with a few definitions and general remarks (section 2).

The code for the simulations shown in this paper is available online at `http://www.di.ens.fr/~brette/papers/Brette2008NC.html`.

2 General considerations

2.1 Definitions

A *spike train* is defined as a sum of Delta functions:

$$S(t) = \sum_k \delta(t - t^k)$$

where t^k is the time of the k^{th} spike. The *firing rate* is the time average of $S(t)$, i.e., the average number of spikes per time unit:

$$r = \langle S(t) \rangle = \lim_{T \to +\infty} \frac{1}{T} \int_0^T S(t)dt$$

Experimental cross-correlograms quantify the temporal relationship between spikes of two different trains. A cross-correlogram is an empirical estimate (histogram) of the *cross-correlation function* (CCF) which is defined for two spike trains i and j as

$$\mathrm{CCF}_{i,j}(s) = \langle S_i(t) S_j(t+s) \rangle$$

This quantity can be interpreted by observing that the probability density that neuron j fires at time $t + s$ conditioned to the fact that neuron i fired at time t is $\langle S_i(t) S_j(t+s) \rangle \,/\, \langle S_i(t) \rangle$. It is often more useful to consider the the *cross-covariance function* (CCVF), which is

$$\mathrm{CCVF}_{i,j}(s) = \langle S_i(t) S_j(t+s) \rangle - \langle S_i(t) \rangle \langle S_j(t) \rangle$$

The subtracted term corresponds to the "baseline" in cross-correlograms. For example, the CCVF of two independent realizations of homogeneous Poisson processes is null, whereas the CCVF of two identical realizations (i.e., the autocovariance function) is $r\delta(s)$, where r is the rate (the result follows easily from the probability interpretation). Thus, it makes sense to define the *total correlation* between spike trains i and j as

$$\lambda_{i,j} = \frac{1}{\langle S_i(t) \rangle} \int \mathrm{CCVF}_{i,j}(s)ds$$

where the value 1 signals perfect synchronization and 0 means no correlation.

In the following, I will consider that we want to generate N spike trains with rates r_1, r_2, ..., r_N, and cross-covariance functions $\text{CCVF}_{i,j}(\cdot)$ ($i \neq j$). For simplicity, I will assume that all CCVFs have the same functional form: $\text{CCVF}_{i,j}(s) = c_{i,j} f(s)$, and I shall call $c_{i,j}$ the *correlation coefficient* (the total correlation is then $\lambda_{i,j} = (c_{i,j}/r_i) \int f$). This hypothesis also implies that f is an even function. Note that there is no unique solution to this problem, i.e., we may find two solutions with the same second-order statistics but different higher-order statistics, and it can have important consequences (see e.g. Kuhn *et al.*, 2003).

The simplest example one can think of is a set of N spike trains with permutation-invariant second-order statistics, i.e., $r_i = r$ for all $i \in \{1, \ldots, n\}$, and $c_{i,j} = c$ for all $i \neq j$. I shall call this example the *homogeneous pool*.

2.2 Population statistics

It is useful to understand how the pair-wise correlations relate to the average activity of the pool of spike trains, i.e.,

$$S(t) = \frac{1}{N} \sum_i S_i(t)$$

More generally, we can examine the properties of a weighted average of the spike trains:

$$S(t) = \sum_i w_i S_i(t)$$

which may represent the total input to a neuron.

The temporal average of $S(t)$ is simply

$$\langle S(t) \rangle = \sum_i w_i r_i.$$

The autocovariance function (ACVF) of $S(t)$ is

$$\begin{aligned} C(s) &= \langle S(t+s)S(t) \rangle - \langle S(t) \rangle^2 \\ &= \sum_{i,j} w_i w_j \langle S_i(t+s) S_j(t) \rangle - \sum_{i,j} w_i w_j \langle S_i(t) \rangle \langle S_j(t) \rangle \\ &= \sum_{i \neq j} w_i w_j \text{CCVF}_{i,j}(s) + \sum_i w_i^2 \text{ACVF}_i(s) \end{aligned}$$

For the mixture method (section 4), we will see that individual spike trains have Poisson statistics, so that $\text{ACVF}_i(s) = r_i \delta(s)$. For the method based on doubly

stochastic processes (section 3), the ACVF is the ACVF of the underlying time-varying rate. For large N, one can see that the first term dominates unless CCVFs decrease to 0, so that the fluctuations of $S(t)$ reflect essentially the correlations of spike trains (more precisely, the first term dominates if $N \times c_{i,j} \to +\infty$, e.g. it still dominates if the CCVFs decrease to 0 as $1/\sqrt{N}$). For example, in the case of the homogeneous pool, with $w_i = 1/N$ (pool average), we obtain (for large N) $C(s) \approx c \times f(s)$. When spike trains are not correlated, the ACVF of the average activity reflects size effects (the second term in the formula).

3 Method I: doubly stochastic processes

Let us consider N independent inhomogeneous Poisson processes with instantaneous rates $r_i(t)$ (note that the Poisson processes are independent but may have correlated rates). The mean rates of the generated spike trains are simply $r_i = \langle r_i(t) \rangle$, and the CCF of spike trains i and j is, for $i \neq j$:

$$\text{CCF}_{i,j}(s) = \langle r_i(t) r_j(t+s) \rangle$$

In other words, the cross-correlation function of the spike trains is the cross-correlation function of the underlying rates. It follows that correlated spike trains can be generated as Poisson processes with correlated rates. When the rates $r_i(t)$ are themselves stochatic processes, the inhomogeneous Poisson processes are called *doubly stochastic processes* or *Cox processes* (Daley & Vere-Jones, 2005). I will start with the simple case of the homogeneous pool before I describe the more general case.

3.1 The homogeneous pool

Suppose we want to generate N spike trains with rates r and CCVFs $c(s) = a \exp(-|s|/\tau_c)$, where a is the strength of the correlations and τ_c is the time constant (the total correlation is $\lambda = 2a\tau_c/r$). How to choose a doubly stochastic process that can generate spike trains with these second-order statistics? Looking at the average population activity, we observe that it reflects the rate $x(t)$ of that process, and we expect to see some noisy function with exponential autocorrelation. Thus, it is natural to define $x(t)$ as a realization of an Ornstein-Ulhenbeck process:

$$\tau_c dx = (r - x)dt + \sigma dW$$

The average of $x(t)$ is r and its autocovariance function is

$$\text{ACVF}(s) = \text{Var}(x) \exp(-\frac{|s|}{\tau_c}) = \frac{\sigma^2}{2\tau_c} \exp(-\frac{|s|}{\tau_c})$$

Therefore, if we generate $x(t)$ as a realization of an Ornstein-Ulhenbeck process with Var$(x) = a$ (i.e., $\sigma = \sqrt{2\tau_c a}$), then the spike trains generated by inhomogeneous Poisson processes with rate $x(t)$ will have the desired mean rates and CCVFs. Note that all spike trains must share the same realization $x(t)$ as instantaneous rate. This construction is illustrated in Fig. 1.

We note already that a problem arises from the fact that the Ornstein-Uhlenbeck process is not always positive, and I will come back to this issue later (paragraph 3.3). Another observation we make here is that individually, spike trains do not have Poisson statistics (contrary to the spike trains generated with the mixture method, section 4); in particular, their autocovariance function is $c(s) + r\delta(s)$ (instead of $r\delta(s)$).

This method corresponds to the diffusion approximation for correlated inputs used in Moreno *et al.* (2002). A variant with exponentially distributed jump processes was used by Song *et al.* (2000). In the following, I generalize this approach to heterogeneous correlation structures and rates.

3.2 The general case

We want to extend the previous method and generate N spike trains with rates r_i and pair-wise CCVFs $c_{i,j}f(s)$ ($i \neq j$). For the sake of simplicity, I will restrict to exponential correlations: $f(s) = \exp(-|s|/\tau_c)$, but one can apply the method to other forms of correlations by replacing Ornstein-Uhlenbeck processes with Gaussian processes with the required autocorrelation function.

In the framework of doubly stochastic processes, our problem reduces to generating N Ornstein-Uhlenbeck processes with rates r_i and covariances $c_{i,j}$ ($i \neq j$). Because these processes are Gaussian and form a linear space, we can solve this problem with the Cholesky decomposition (Press *et al.*, 1993). Let $\mathbf{R} = (r_1, \ldots, r_N)$ the vector of mean rates and $\mathbf{C} = (c_{i,j})$ the matrix of correlations. The algorithm is as follows:

1. Generate N independent Ornstein-Uhlenbeck processes y_i with zero mean and unit variance. Note $\mathbf{Y} = (y_1, \ldots, y_N)$.
2. Compute $\mathbf{L}$ from the Cholesky decomposition of $\mathbf{C} = \mathbf{L}\mathbf{L}^T$.
3. Let $\mathbf{X} = \mathbf{R} + \mathbf{L}\mathbf{Y}$.

Then the $x_i(t)$ are Ornstein-Uhlenbeck processes with mean r_i and covariances $c_{i,j}$ ($i \neq j$). It follows that the N spike trains generated from the rates $x_i(t)$ will have the desired rates and CCVFs. This construction is illustrated in Fig. 2.

A similar method was used recently by Galan *et al.* (2006) to generate correlated input noise (not spikes).

Completion of the diagonal

One important point is missing in this algorithm: the diagonal coefficients $c_{i,i}$ (which are the variances of the processes $x_i(t)$) are unspecified. In fact any diagonal coefficients will be acceptable, provided that the completed matrix $\mathbf{C}$ is positive semi-definite (otherwise the Cholesky decomposition does not exist). A good criterion for the completion is to choose the smallest coefficients possible, because it reduces the problem of positivity (see paragraph 3.3). I propose the following procedure.

Define $\mathbf{C}^*$ the matrix with coefficients $c^*_{i,j} = c_{i,j}$ and $c^*_{i,i} = 0$. Let $\mathbf{D}$ be the diagonal matrix with entries $d_{i,i} = r_i^2$, and $\mathbf{C}_\alpha = \mathbf{C}^* + \alpha\mathbf{D}$, for $\alpha \in \mathbb{R}^+$. There is a smallest value α^* such that $\mathbf{C}_{\alpha^*}$ is positive semi-definite. For that minimum value, all eigenvalues are non negative, but there is at least one null value (otherwise we could choose a smaller α). Therefore $\mathbf{C}_{\alpha^*}$ is non-invertible, i.e., $det(\mathbf{C}_{\alpha^*}) = 0$, which is equivalent to $det(\mathbf{D^{-1}C}^* + \alpha^* I_N) = 0$. It follows that $-\alpha^*$ is the smallest real eigenvalue of $\mathbf{D^{-1}C}^*$ (matrix with coefficients $c_{i,j}\delta(i-j)/r_i^2$), and we set $c_{i,i} = \alpha^* r_i^2$.

3.3 The positivity problem

As mentioned earlier, the main problem with using an Ornstein-Uhlenbeck process for the stochastic rates is that it is not a positive process, whereas rates are, obviously, always positive. If the variance is not too large (compared to the mean), the easiest way is to rectify the process, i.e., use $x^+(t)$ instead of $x(t)$ ($x^+ = 0$ when $x < 0$). In doing so however, we modify the mean and variance of the process. If we want to be precise, it is possible to adjust the mean and variance of x so that x^+ has the desired statistics, using a nonlinear optimization procedure, as I show in appendix A. However, generating strong correlations with a rectified Gaussian process implies that the rates are zero most of the time, which does not seem very desirable in most applications. Precisely, the rectified variable is null at least half of the time when $\text{Var}[x^+] \leq (\pi - 1)E[x^+]^2$. In particular, for the homogeneous pool example, the rates are positive at least half of the time if $a \leq r^2$, i.e., if the total correlation is smaller than $2r\tau_c$. In other words, it is not reasonable to generate strongly correlated spike trains at fine time scales with Gaussian time-varying rates.

Another option is to use a non-Gaussian process instead, but then we would lose the linearity property that makes the Cholesky decomposition possible, and the generalization to complex correlation structures would be much harder. I will not consider this option here and instead I will present a very different method based on random mixtures of spike trains in section 4.

3.4 Simulation of doubly stochastic processes

There are essentially two families of algorithms for the simulation of neural networks: clock-driven (or synchronous) algorithms, in which all state variables are updated simultaneously at every tick of a clock, and event-driven (or asynchronous) algorithms, in which state variables are updated only upon reception or emission of a spike (Brette et al, 2007). I will describe an algorithm of each kind to generate correlated spike trains with doubly stochastic processes. Both algorithms share the following initial phase:

1. Construct the matrix of correlations $\mathbf{C}$ and complete the diagonal (see 3.2).
2. Find the Cholesky factor $\mathbf{L}$ ($\mathbf{C} = \mathbf{L}\mathbf{L}^T$).

Let us define $\mathbf{Y}(t) = (y_i(t))$ a vector of N independent Ornstein-Uhlenbeck processes with unit variance and zero mean (to be constructed by the algorithm) and $\mathbf{X}(t)$ the vector of instantaneous rates. We set $\mathbf{Y}(0) = 0$.

3.4.1 Clock-driven simulation

In the clock-driven simulation, time is advanced by discrete time steps $t \to t + dt$ (dt is small) and each time step involves the following operations:

1. Update the vector $\mathbf{Y}$: $\mathbf{Y}(t) \to \mathbf{Y}(t + dt)$. This operation is done simply by noting that $y_i(t+dt)$, conditionally to $y_i(t)$, is a normally distributed random variable with expectation $y_i(t)\exp(-dt/\tau_c)$ and variance $1-\exp(-2dt/\tau_c)$.
2. Calculate $\mathbf{X}(t + dt) = (\mathbf{R} + \mathbf{L}\mathbf{Y}(t + dt))^+$ (rectified).
3. For every $i \in \{1, \dots, n\}$, generate a spike with probability $x_i(t + dt)dt$.

The algorithmic complexity is dominated by step 2, which takes $O(N^2)$ operations. Therefore the computational cost per unit time is $O(N^2/dt)$, which is high compared to the simulation of independent spike trains ($O(N/dt)$).

3.4.2 Event-driven simulation

The event-driven approach is conceptually very different and much more complex. We want to generate inhomogeneous Poisson processes with instantaneous rates $\mathbf{X} \in \mathbb{R}^N$, but I will start with a single process.

A single spike train:

For a start, suppose we want to generate a single inhomogeneous Poisson process with rate $x(t)$. Assume that $x(t)$ is bounded: $x(t) \leq M$. Then consider the following algorithm (see Fig. 3):

1. Generate a realization of a homogeneous Poisson process with rate M.

2. Keep every point t with probability $x(t)/M$.

This algorithm generates realizations of a Poisson process with rate $x(t)$. Indeed, one can see that the number of spikes in two distinct intervals is independent and the intensity of the process at time t is $M * x(t)/M = x(t)$. Note that in this algorithm, $x(t)$ needs only be calculated at times of the homogeneous Poisson process (of step 1).

If $x(t)$ is not bounded the algorithm can be amended as follows, for a simulation over some finite interval $[0, T]$:

1. Generate a realization of a homogeneous Poisson process with some rate M.

2. If $x(t) > M$ for at least one point t in that realization, then increase M (e.g. $M \to 2M$) and generate a new realization (back to 1).

3. Keep every point t with probability $x(t)/M$.

This algorithm works fine if $x(t)$ is fixed and known in advance, but unfortunately there is a problem in our case where $x(t)$ is a Markov process. Indeed, if the condition of step 2 is satisfied, then we must generate a new Poisson process (step 1) and the values $x(t)$ at the corresponding times, but these values must be generated conditionally to the values of $x(\cdot)$ at the times of the previous Poisson process (or to the fact that the condition of step 2 was satisfied). This would make the algorithm much more complicated. In our case, the best way is to choose a large value of M and simulate the doubly stochastic process with upper-rectified rates $\max(x(t), M)$, which is an excellent approximation if M is large. Indeed, one can calculate that the probability that the condition of step 2 is satisfied is of order

$$MTe^{-\frac{4M^2}{\pi\sigma^2}}$$

for large M, if $x(t)$ is a Gaussian process with variance σ^2. Taking $M = 10\sigma$, we can see that the probability is extremely small.

To summarize, to simulate a doubly stochastic process for which the underlying rate is an Ornstein-Uhlenbeck process $x(t)$, mean μ, standard deviation σ and time constant τ_c, I propose the following algorithm:

1. Choose $M = \mu + 10\sigma$. Let $t = 0$, $x = \mu$ and $i = 0$.

2. Update t: $t \leftarrow t + \Delta$, where Δ is exponentially distributed with mean $1/M$.

3. Update x:

$$x \leftarrow \mu + (x - \mu)\exp(-\Delta/\tau_c) + \sigma\sqrt{1 - e^{-2dt/\tau_c}}N$$

where N is a normally distributed variable with variance 1 and mean 0.

4. Pick a number u in $[0, M]$ uniformly at random. If $u \leq x$, then set time $t_i \leftarrow t$ and $i \leftarrow i + 1$.

5. Repeat from step 2 until $t > T$.

When the algorithm ends, the variables t_i contain the timings of the spikes. The number of operations is proportional to $M \times T$.

Multiple spike trains:

We turn to the more general problem, i.e., generating N spike trains with instantaneous rates $x_1(t), \ldots, x_N(t)$. First, we note that the union of all spike trains is an inhomogeneous Poisson process with instantaneous rate $s(t) = \sum_i x_i(t)$. Then the idea is as follows: generate the union of all spike trains according to the previous algorithm, then allocate each spike at time t to one of the N spike trains according to the weights $x_i(t)/s(t)$. In our problem, the sum $s(t) = \sum_i x_i(t)$ is a Gaussian process with variance

$$\begin{aligned} \sigma_s^2 &= \sum_i \text{Var}(x_i) + \sum_{i \neq j} \text{Cov}(x_i, x_j) \\ &= \sum_{i,j} c_{i,j} \end{aligned}$$

($\mathbf{C}$ is the completed correlation matrix). Let $\mathbf{V}$ be the (column-)vector of ones ($v_i = 1$ for all i). Then

$$\begin{aligned} s &= \mathbf{V}^T\mathbf{R} + \mathbf{V}^T\mathbf{L}\mathbf{Y} \\ &= s^* + \mathbf{A}\mathbf{Y} \end{aligned}$$

where $\mathbf{A} = \mathbf{V}^T\mathbf{L}$ (a row vector) and $s^* = \sum_i r_i$. The cost of this operation is $O(N)$. To allocate a spike to one of the N spike trains, we would calculate $\mathbf{X} = \mathbf{R} + \mathbf{L}\mathbf{Y}$, pick a number at random $v \in [0, s]$, and find i such that

$$\sum_{k=1}^{i-1} x_k < v \leq \sum_{k=1}^{i} x_k.$$

With this method, the cost of allocation is $O(N^2)$ because of the calculation of $\mathbf{X}$. This calculation can be avoided as follows. Define $\mathbf{A}_i = \sum_{k=1}^{i} \mathbf{L}_k$, where

$\mathbf{L}_k$ is the k^{th} row of $\mathbf{L}$ (and $\mathbf{A}_0 = 0$), and $b_i = \sum_{k=1}^{i} r_k$. Then pick a number at random $v \in [0, s]$ and find i such that $b_{i-1} + \mathbf{A}_{i-1}\mathbf{Y} < v \leq b_i + \mathbf{A}_i\mathbf{Y}$ by binary search.The search involves $O(\log N)$ dot product operations, so that the total cost is $O(N \log N)$.

Thus, the algorithm is the following:

1. Set $M = \sum_i r_i + 10(\sum_{i,j} c_{i,j})^{1/2}$. Let $t = 0$, $y_k = 0$ and $i_k = 0$ for all $k \in \{1, \ldots, n\}$. Define $\mathbf{A}_i = \sum_{k=1}^{i} \mathbf{L}_k$ and $\mathbf{A} = \mathbf{A}_N$.
2. Update t: $t \leftarrow t + \Delta$, where Δ is exponentially distributed with mean $1/M$.
3. Update y_i for each $i \in \{1, \ldots, n\}$:

$$y_i \leftarrow y_i \exp(-\Delta/\tau_c) + \sqrt{1 - e^{-2dt/\tau_c}} N$$

   where N is a normally distributed variable with variance 1 and mean 0.
4. Calculate $s = s^* + \mathbf{A}\mathbf{Y}$.
5. Pick a number $u \in [0, M]$ uniformly at random.
6. If $u \leq s$, then pick a number $v \in [0, s]$ at random (uniformly) and find k such that $b_{k-1} + \mathbf{A}_{k-1}\mathbf{Y} < v \leq b_k + \mathbf{A}_k\mathbf{Y}$ by binary search. Then set time $t_{i_k}^k \leftarrow t$ and $i_k \leftarrow i_k + 1$.
7. Repeat from step 2 until $t > T$.

The costly part in this algorithm is the binary search, which requires $O(N \log N)$ operations. A binary search is executed for every spike in the union train, i.e., on average $T \sum_i r_i$ times. Therefore the algorithmic complexity is $O((\sum_i r_i)T \times N \log N)$. For the clock-driven algorithm, we obtained $O(TN^2/dt)$. If we note $r = \langle r_i \rangle$ (average rate), then for large N the condition for the event-driven method to be more efficient than the clock-driven method is

$$r \log_2 N < \frac{1}{dt}.$$

Thus in practice, we can expect the event-driven method to be more efficient. Indeed, if we take for example $dt = 1$ ms and $r = 10$ Hz, then the condition above is satisfied up to $N = 2^{100}$.

4 Method II: the mixture method

In this section I describe an alternative method to generate correlated spike trains, which consists in selecting spikes from common spikes trains. I start with the case of a homogeneous pool with instantaneous correlations, for which several authors have described such algorithms. Then I generalize these algorithms to the case of heterogeneous correlation structures, first with instantaneous correlations, then with predefined correlation functions (e.g. exponential or Gaussian).

4.1 The homogeneous pool

We want to generate a homogenous pool of N spike trains with rate r and CCVFs $c_{i,j}(s) = cr\delta(s)$ for $i \neq j$ (the autocorrelation function is $c_{i,i}(s) = r\delta(s)$; note that $c \leq 1$). Several authors have proposed the following procedure, sketched in Fig. 4A (Kuhn *et al.*, 2003; Feng & Brown, 2000). Consider a source spike train defined as a realization of a Poisson process with rate r/c. Then generate N spike trains as follows: for each target spike train $i \in \{1, \ldots, N\}$, insert each spike from the source spike train with probability c. This procedure generates N Poisson spike trains with rates r and CCVFs $c_{i,j}(s) = cr\delta(s)$.

Other authors have proposed a similar method, but with $N + 1$ source spike trains (Gutig *et al.*, 2003; Kuhn *et al.*, 2003; Galan *et al.*, 2006), as shown in Fig. 4B. Define $N + 1$ source spike trains as independent Poisson processes, the first N with rate $(1 - c)r$, the last one with rate cr. Then define each target spike train i as the union of the source spike trains i and $N + 1$. Again, this procedure generates N Poisson spike trains with rates r and CCVFs $c_{i,j}(s) = cr\delta(s)$. Note however that the higher-order statistics are not the same for the two methods.

Other variants have been proposed (Destexhe & Pare, 1999; Rudolph & Destexhe, 2003; Niebur, 2007), which are also particular cases of the general method I present in the next section.

4.2 The general case

The examples for the homogeneous pool generalize to the following method, sketched in Fig. 5, which I call the *mixture method*. Consider M source spike trains, defined as independent Poisson processes with rates $\nu_1, \ldots, \nu_M$. Then generate N target spike trains by copying every spike from source k to target i with probability $p_{i,k}$.

The firing rate of target train i is

$$r_i = \sum_{k=1}^{M} p_{i,k}\nu_k$$

or in vector form: $\mathbf{R} = \mathbf{P}\nu$. The CCVFs are

$$c_{i,j}(s) = \sum_{k=1}^{M} p_{i,k} p_{j,k} \nu_k \delta(s)$$

for $i \neq j$, or in vector form: $\mathbf{C}(s) = \mathbf{P}\mathrm{Diag}(\nu)\mathbf{P}^T\delta(s)$, except for the diagonal (the ACVFs are $r_i\delta(s)$).

Thus, if $\mathbf{P}$ and ν are cleverly chosen (see sections 4.4 and 4.5), then one can generate spike trains with various firing rates and correlation structures. I describe specific examples in section 4.4 and a general method to determine $\mathbf{P}$ and ν in section 4.5. We note however that only positive correlations are possible.

The spike trains produced by the mixture method are individually Poisson processes. In particular, contrary to Cox processes, their ACVFs are simply $r_i\delta(s)$ and the interspike intervals are exponentially distributed.

A similar attempt to generalize the homogeneous mixture method (the second method, with N independent spike trains and one common train) to general correlation structures was recently made in (Niebur, 2007) (although in a discrete time framework). It corresponds to a specific case of the mixture method where $M = N + 1$, $p_{i,j} = \delta(i - j)(1 - \sqrt{q_i})$ and $p_{i,N+1} = \sqrt{q_i}$ ($i \in \{1, \ldots, N\}$, $j \in \{1, \ldots, N+1\}$), and thus is restricted to specific forms of correlation matrices (specifically, of the form $\mathbf{C} = \mathbf{X^T X}$ where $\mathbf{X}$ is a column vector; the method was also restricted to instantaneous correlations).

4.3 Non-instantaneous correlations

The mixture method described above produces only delta-shaped correlations. A simple way to obtain non-instantaneous correlations is to add independent random values to each spike time in the target spike trains. The rates and the Poisson statistics are not modified by this procedure (drawing a Poisson process on the line and moving the points independently still produces a Poisson process with the same intensity). The CCVFs, however, are modified. For example, if normally distribued random values with standard deviation σ are added to the spike times, then the delta function $\delta(s)$ in the CCVFs is changed to a normal distribution with standard deviation $\sqrt{2}\sigma$.

More generally, if i.i.d numbers are added to all spike timings, then the statistics of individual spike trains are unchanged (all-order statistics), while the delta function in the cross-correlation functions is changed to the convolution

$$g(s) = \int_{-\infty}^{+\infty} f(x) f(x + s) dx$$

where f is the probability density of the random shifts. Thus, one can obtain the desired correlation function by choosing the appropriate distribution function f for the random shifts. In the next paragraph I give the important example of exponential correlations.

Exponential correlations

We want to generate spike trains with exponential correlations with time constant τ_c, i.e.:

$$g(s) = \frac{1}{2\tau_c} e^{-\frac{|s|}{\tau_c}}$$

One easily checks that this can be achieved by choosing

$$f(x) = \frac{1}{\tau_c} e^{-\frac{x}{\tau_c}}$$

for $x \geq 0$ (otherwise $f(x) = 0$). Thus, to generate exponentially correlated spike trains, it is enough to shift all spike times randomly with an exponential distribution.

4.4 A few examples of mixture processes

In the following, I describe a few non trivial examples of mixture processes.

4.4.1 Global synchronization

Here I generalize the mixture for the homogeneous pool to the case where the rates r_i are different (see Fig. 6A). Let us define the correlation coefficients as follows:

$$c_{i,j} = c\frac{r_i r_j}{\langle r_k \rangle}.$$

which is a generalization of the correlations in the homogeneous pool ($c_{i,j} = cr$).

Let us choose $M = N$ and the constant source rates

$$\nu = \nu_k = \frac{1}{c}\frac{\langle r_i^2 \rangle}{\sum_i r_i}$$

for all k, and

$$p_{i,k} = c\frac{r_i r_k}{\left\langle r_j^2 \right\rangle}.$$

One can check that this choice indeed produces target spike trains with the required rates and correlation coefficients. To ensure that $p_{i,k} \in [0,1]$ for all i,k, the global correlation coefficient c must meet the following criterion:

$$c \leq \frac{\langle r_i^2 \rangle}{\max r_i^2}$$

4.4.2 Topographic synchronization

We consider neurons arranged topographically in a line, and we want to generate spike trains with the same rates and with pairwise correlations that decrease with the distance between the two neurons.

Let us define a line of source spike trains with constant rate ν and consider the mixture matrix defined by $p_{i,j} = \lambda a^{|i-j|}$, where λ is a synchronization strength parameter and $\log a$ is the spatial constant of correlations ($0 < a < 1$,$0 < \lambda < 1$).

The target rates are given by

$$r_i = \sum_k p_{i,k} \nu$$

Neglecting the boundary effects, i.e., assuming the sum runs from $-\infty$ to $+\infty$, we obtain

$$\begin{aligned} r_i &= \nu\lambda \sum_{k=-\infty}^{+\infty} a^{|i-k|} \\ &= 2\nu\lambda \sum_{k=0}^{+\infty} a^k \\ &= \frac{2\nu\lambda}{1-a} \end{aligned}$$

for all i. If r the desired target rate, then ν can be chosen according to the following formula:

$$\nu = \frac{r(1-a)}{2\lambda}$$

For $j > i$, the correlation coefficient $c_{i,j}$ is (again, far from the boundaries):

$$\begin{aligned} c_{i,j} &= \sum_k p_{i,k} p_{j,k} \nu \\ &= \nu\lambda^2 (\sum_{k \le i} a^{i+j-2k} + \sum_{i<k\le j} a^{j-i} + \sum_{k>j} a^{2k-i-j}) \\ &= \nu\lambda^2 (a^{j-i} \sum_{k=0}^{+\infty} a^{2k} + (j-i)a^{j-i} + a^{j-i} \sum_{k=1}^{+\infty} a^{2k}) \\ &= \nu\lambda^2 a^{j-i} (\frac{1+a^2}{1-a^2} + j - i) \\ &= \frac{\lambda(1-a)}{2} (\frac{1+a^2}{1-a^2} + j - i) a^{j-i} r \end{aligned}$$

and we obtain indeed a correlation coefficient that decreases with the interneuronal distance. If $c_{\max} r$ is the maximum correlation, then the parameter can be chosen according to the following formula:

$$\lambda = 2 c_{\max} \frac{1+a^2}{1+a}$$

Fig. 6B shows sample spike trains generated with this topographic mixture.

4.4.3 Weak synchronization

For more general correlation structures, it is more difficult to define an appropriate mixture (see section 4.5). Here I describe a simple generic construction when correlations are small (i.e., of order $1/N$). The desired rates are r_i and the desired correlation coefficients are $c_{i,j}$.

Index k on $A \subset \{1, \ldots, N\}^2$, where $(i,j) \in A$ if $i \le j$. Set $p_{i,(i,m)} = 1$ and $p_{i,(l,m)} = 0$ if $l \neq i$ and $m \neq i$. Set also $p_{i,(i,i)} = 1$. Then define

$$\begin{aligned} \nu_{(i,j)} &= c_{i,j} \\ \nu_{(i,i)} &= r_i - \sum_{j>i} c_{i,j} \end{aligned}$$

provided these are positive numbers. Then we obtain the desired rates r_i and the correlation coefficients $c_{i,j}$. But one can see that the positiveness requirement means $\sum_{j>i} b_{i,j} < r_i$, thus the correlations must be of order $1/N$ for this scheme to be applicable.

4.5 Finding the mixture in the general case

In general, we are given the rates r_i and the correlations $c_{i,j}$, and we want to construct a mixture defined by the matrix $\mathbf{P}$ and the rates ν_k, with the constraints $p_{i,k} \in [0,1]$ and $\nu_k \geq 0$. This problem is far from trivial in general. First we note that for all i, $c_{i,j} \leq r_i$, and equality is satisfied for $\{0,1\}$ matrices $\mathbf{P}$. Formally, the inverse problem is to find $\mathbf{P}$ and ν (vector) such that

1. $\mathbf{R} = \mathbf{P}\nu$ ($\mathbf{R}$ is the vector of rates),
2. $\mathbf{C} = \mathbf{Q}\mathbf{Q}^\mathbf{T}$, where $\mathbf{Q} = \mathbf{P}\sqrt{Diag(\nu)}$, except for the diagonal entries ($Diag(\nu)$ is the diagonal matrix with diagonal entries ν_k), which are unspecified,
3. ν is a positive vector,
4. $\mathbf{P}$ has entries in $[0,1]$.

In step 2, writing $\mathbf{C} = \mathbf{Q}\mathbf{Q}^\mathbf{T}$ with a positive matrix $\mathbf{Q}$ means that $\mathbf{C}$ is a *completely positive matrix* (which is not the same as a positive definite matrix). Unfortunately, there is no practical algorithm for decomposing a completely positive matrix (in the general case) or even deciding whether a matrix is completely positive (Berman & Shaked-Monderer, 2003), and the fact that the diagonal entries are unspecified makes the problem even harder.

The equality in step 1 can be relaxed to the inequality $\mathbf{R} \geq \mathbf{P}\nu$. Indeed, if this inequality holds, then we can obtain the equality by completing the matrix $\mathbf{P}$ to $(\mathbf{P}|\mathbf{I_N})$ (block form) and by completing the vector ν with the N elements $\nu_{N+k-1} = r_i - \sum_k p_{i,k}\nu_k$ (k from 1 to N).

In appendix B, I describe two ways of solving this problem. The first one consists in looking for a particular solution with fixed column sums for $\mathbf{P}$. The algorithm is fast but not general because such a solution does not always exist. The second way consists in expressing the constraints as a nonlinear optimization problem and finding the solution with an optimization algorithm (e.g. gradient descent). The output of these algorithms are shown in Fig. 7.

4.6 Simulation of mixture processes

4.6.1 Offline simulation

The simple (naive) algorithm is as follows: first generate the M independent Poisson spike trains (using the fact that ISIs are exponentially distributed), then for each spike, copy it to target spike train i with probability $p_{i,k}$, then shift it according to

the delay distribution $f(x)$. The number of operations is $MN\nu t$, where t is the duration of the simulation and ν is the average rate of the source spike trains.

This method can be efficient if $\mathbf{P}$ is a low rank matrix, e.g. in the case of the homogeneous pool (then the simulation cost is $O(N\nu t)$). However, in general, M has the same magnitude as N, so that the simple algorithm requires $O(N^2\nu t)$ operations. I propose an algorithm which requires $O(N \log Nrt)$ operations, where r is the average rate of the target spike trains.

We want to generate the spike trains in a temporal window $[0, t]$. Let $n_{i,k}$ be the number of spikes in target train i that come from the source train k. This number follows a binomial distribution with probability $p_{i,k}$ and number of trials $n =$ number of spikes in spike train k, which suggests the following algorithm:

1. Generate M independent Poisson spike trains over $[0, t]$.
2. For each target train i and each source train k, draw the number of spikes coming from k ($n_{i,k}$) from a binomial distribution with probability $p_{i,k}$ and number of trials $n =$ number of spikes in spike train k, then pick $n_{i,k}$ spikes from source train k.
3. Shifts all spikes according to the delay distribution $f(x)$.
4. Sort the spikes.

Generating the source spike trains (1) takes $O(M\nu t)$ operations (ν is the average source rate). Calculating the numbers $n_{i,k}$ (2) takes $O(MN)$ operations, and picking the spikes takes $O(Nrt)$ operations (i.e., the total number of spikes in the target trains; r is the average target rate). Shifting the spikes (3) also requires $O(Nrt)$ operations. Finally, sorting the spikes (4) requires $O(Nrt \log(Nrt))$ operations, which dominates the total cost. This cost can be reduced by splitting the full interval $[0, t]$ into successive windows of size T. If T is small, then sorting will be faster, but there will be more calculations of binomially distributed numbers (MNt/T). Thus we can see that the T-dependent part is of order $Nrt \log(NrT) + MNt/T$. Writing $f(x) = -r \log(x) + aMx$ (a is an implementation-dependent constant), we can see that we are looking for T such that $f'(1/T) = 0$, and we find $x^* = r/(aM)$, thus $T \propto M/r$. Inserting back this value, we find that the total cost for this optimal value is of order $Nrt \log(MN) + M\nu t$, and if M is of order N, then we get $O(Nrt \log(N))$.

4.6.2 Online simulation

It is possible, although more complicated, to simulate this procedure in an event-driven way, as described below.

1. Find the next spike in the pool of M target spike trains. There are two ways to do it: either simulate M independent Poisson queues with rate ν_k and extract the next event by using a global priority queue (the cost of insertion/extraction is $O(\log M)$)), or simulate a Poisson process with rate $\sum_k \nu_k$ (the union of all spike trains), and pick the origin of the next spike according to probabilities $\nu_i / \sum \nu_k$ (the cost is $O(1)$ with a precalculated table, $O(\log M)$ otherwise).

2. Calculate the number of copies of this spike in the target spike trains. This number is a random Poisson variable with mean $\sum_i p_{i,k}$. Since this sum can be calculated only once at the beginning, this operation requires $O(1)$ operations.

3. Distribute each copy to the target spike trains with probabilities proportional to $p_{i,k}$ (with no duplication). This operation can be done with precalculated tables.

If precalculated tables are used (to generate the random numbers), then the cost of this algorithm is no more than the total number of spikes in the original and target pools, i.e., $O(\sum \nu_k t + \sum r_i t)$. With random shifts (to generate non-instantaneous correlations), events need to be inserted in a priority queue. The size of the queue is proportional to N, so if M is of order N the simulation cost is also $O(Nrt \log(N))$ with a standard priority queue.

5 Discussion

Summary

I presented two methods to generate correlated spike trains with given rates and pairwise correlation functions. Compared to previous propositions, both methods are general and not restricted to instantaneous correlations, and I also provided efficient simulation algorithms. Except in specific cases (e.g., permutation-invariant statistics or instantaneous correlations), the most efficient algorithm I proposed is slower than simulating uncorrelated spike trains ($Nr \log(N)$ vs. Nr per second, where r is the average firing rate in the population), but only by a factor $\log(N)$.

The two methods are conceptually quite different, and it is important to observe that the second-order constraints do not fully constrain the statistics of the spike trains, so that different solutions to the problem of generating correlating spike trains with given rates and pairwise correlations are not equivalent. The first method consists in generating spike trains with underlying time-varying rates which are correlated. This is reflected in the fact the autocorrelation of a single

spike train is the autocorrelation of the underlying rate, in particular it has the same time constant. Therefore this method would not seem very appropriate to model tight synchronization arising from shared presynaptic neurons; besides, the positivity of rates implies that one cannot generate strongly correlated spike trains with short correlation time constants. On the other hand, the mixture method can be seen as an abstract representation of the generation of synchronized spike trains arising from shared presynaptic neurons (see Fig. 5), and in this way, it seems more appropriate for short correlation time constants. The mixture method generates spike trains with Poisson statistics (in particular, flat autocorrelation) and its simulation is relatively efficient. However, finding the correct mixture in the general case requires a nonlinear optimization procedure which may not converge to a unique solution, and it can be a problem given that different solutions can have different higher-order statistics. Table 1 summarizes the differences between the two methods.

	Method I (Cox processes)	Method II (mixture method)
Statistics of spike trains	Not Poisson	Poisson
Negative correlations	Yes	No
Fast and strong correlations	Not appropriate	Appropriate
Programming	Simple	Complicated
Simulation cost (/s)	$O(rN^2 \log N)$	$O(rN \log N)$
Population rate	Underlying rate	Solution-dependent
Specific issues	Positivity of rates	Non-unique solutions

Table 1: Comparison of methods I and II.

Example: response of an integrate-and-fire model to correlated inputs

In order to illustrate the possible applications of the algorithms, I describe a brief example about the sensitivity of neurons to correlations. Moreno et al (2002) have calculated the output rate of an integrate-and-fire model driven by correlated inputs, using a diffusion approximation of the input current. The methods I presented here provide a numerical way of assessing the quality of these expressions. In Fig. 8, I simulated correlated inputs with both methods I (Cox processes) and II (mixture method) and observed the firing rate of an integrate-and-fire model with these inputs. Although the simulation results match the analytical prediction qualitatively, there are some significant quantitative differences due to size effects, particularly for short correlation time constants. The figure also illustrates the fact that two different mixture processes with identical second order properties are not

equivalent. Mixture process A uses a single source spike train with rate r/c while mixture process B uses N independent source trains and one common train with rate cr. In the latter case, correlations have a minor effect on the postsynaptic rate (not significantly different from the uncorrelated case); this is not surprising since the increase in firing rate cannot be higher than the firing rate of the common spike train, i.e., cr (as noted also in Kuhn *et al.* (2003)).

Perspectives

Both methods can be extended in several ways. First, the problem of generating strong correlations on short time scales with doubly stochastic processes may be addressed by using non-Gaussian processes for the underlying rates. This is possible in principle, but it would probably make the algorithms more complicated, because the use of the Cholesky decomposition relies on the fact that the addition of two independent Gaussian variables is also Gaussian. Oscillatory correlations can be included with minor modifications by introducing oscillatory rates in the doubly stochastic method, and by changing the delay function $f(s)$ in the mixture method (or alternatively, by using inhomogeneous Poisson processes with oscillatory rates instead of homogeneous Poisson processes as source spike trains).

Beyond the simulation of correlated spike trains, the methods I described also provide a theoretical model of correlations that could be used to investigate their role by analytical means. In particular, the mixture method provides an idealized model of the correlations induced by shared presynaptic neurons, which could be useful in studying neural computation in early sensory pathways (Bair, 1999).

Acknowledgments

I thank Alain Destexhe and Rubén Moreno-Bote for fruitful discussions. This work was partially supported by the EC IP project FP6-015879, FACETS, and the EADS Corporate Research Foundation.

A Rectification of Gaussian processes

Using Gaussian processes (e.g., Ornstein-Uhlenbeck processes) as rates for the spike trains (Method I, section 3) poses a problem of positivity, since rates must be positive. If the variance is not too large (compared to the mean), the easiest method is to rectify the rates, i.e., use $x^+(t)$ instead of $x(t)$ ($x^+ = 0$ when $x < 0$), but in doing so, we modify the mean and variance. If X is a Gaussian random variable

with mean $E[X] = \mu$ and variance $\text{Var}[X] = \sigma^2$, then the mean and variance of the rectified variable X^+ are as follows:

$$\begin{aligned} E[X^+] &= \frac{\sigma}{\sqrt{2\pi}} \exp(-\frac{\mu^2}{2\sigma^2}) + \frac{\mu}{2} + \frac{\mu}{2}\text{Erf}(\frac{\mu}{\sqrt{2}\sigma}) \\ \text{Var}[X^+] &= (\mu - E[X^+])E[X^+] + \frac{\sigma^2}{2}(1 + \text{Erf}(\frac{\mu}{\sqrt{2}\sigma})) \end{aligned}$$

where

$$\text{Erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{t^2} dt.$$

Thus, by inverting these relationships, we can obtain a rectified Gaussian variable with the desired variance and mean. First observe that

$$\frac{E[X^+]}{\sqrt{\text{Var}[X^+]}} = f(\frac{\mu}{\sigma})$$

where f is a positive function. Then, given $\text{Var}[X^+]$ and $E[X^+]$, find μ/σ using a root finding algorithm. Finally, find σ using the formula for $E[X^+]$ above, and deduce μ.

One can see that the rectified variable is null at least half of the time when $\text{Var}[X^+] \leq (\pi - 1)E[X^+]^2$ (let $\mu = 0$ in the equations above). Thus, the method can generate strong correlations only with rates that are zero most of time.

The correction procedure described above addresses the rectification of a single Gaussian variable, it can be used for example to generate a homogeneous pool of correlated spike trains (identical rates and pairwise correlations). In principle, the same kind of correction can be applied in the multidimensional case by calculating $E[X^+Y^+]$ for a bidimensional Gaussian variable (X, Y) and using a multidimensional optimization procedure. However the expression of $E[X^+Y^+]$ involves integrals and the optimization procedure can be compution nally heavy. In this case, if the distorsions due to the rectification are too important, it might be better to use a different method (e.g. the mixture method).

B Solving the inverse problem for the mixture method

As shown in section 4.5, to generate spike trains with the desired rates r_i and correlation coefficients $c_{i,j}$, we must find a matrix $\mathbf{P}$ and a vector ν such that

1. $\mathbf{R} \geq \mathbf{P}\nu$ ($\mathbf{R}$ is the vector of rates),
2. $\mathbf{C} = \mathbf{Q}\mathbf{Q}^\mathbf{T}$, where $\mathbf{Q} = \mathbf{P}\sqrt{Diag(\nu)}$, except for the diagonal entries ($Diag(\nu)$ is the diagonal matrix with diagonal entries ν_k), which are unspecified,

3. ν is a positive vector,

4. $\mathbf{P}$ has entries in $[0, 1]$.

I describe two methods to solve this problem. The first one is fast but not general, and consists in finding a particular solution; the second one is a general optimization method.

Matrices with fixed column sums

We look for a particular solution of the problem such that the matrix $\mathbf{P}$ has fixed column sums, i.e., $\sum_i p_{i,k} = c$ for all k, which means that each spike from spike train k is duplicated the same average number of times. This assumption will help us specify the diagonal entries of $\mathbf{Q}\mathbf{Q}^T$, since only the coefficients $c_{i,j}, i \neq j$ are specified. We shall call a matrix obtained by specifying the diagonal entries of $\mathbf{C}$ a completion of the matrix $\mathbf{C}$.

Suppose that $\mathbf{P}$ defines a solution with fixed column sums. Then the completion $\mathbf{D} = \mathbf{Q}\mathbf{Q}^T$ is such that $\sum_j d_{i,j} = cr_i$, so that $d_{i,i} = cr_i - \sum_{j \neq i} c_{i,j}$.

Choosing different values for c defines different completions of $\mathbf{C}$. Since $c_{i,i}$ must be positive, we must choose:

$$c > \frac{\sum_{j \neq i} c_{i,j}}{r_i}$$

and besides $\mathbf{C}$ must be positive definite. If we note C_0 the completion with $c_{i,i} = -\sum_{j \neq i} c_{i,j}$, then we want to find c such that $\mathbf{C}_0 + c\, Diag(\mathbf{R})$ has a purely positive spectrum. The set of positive definite matrices is convex, so that there is a minimum c_0 such that $\mathbf{C}_0 + c\, Diag(\mathbf{R})$ is positive definite for any $c > c_0$. The spectrum of $\mathbf{C}_0 + c_0\, Diag(\mathbf{R})$ must contain 0, i.e., $\det(\mathbf{C}_0 + c_0\, Diag(\mathbf{R})) = 0$ or, equivalently, $\det(Diag(\mathbf{R})^{-1}\mathbf{C}_0 + c_0 I_N) = 0$. Thus, $-c_0$ is an eigenvalue of $Diag(\mathbf{R})^{-1}\mathbf{C}_0$. By convexity of the set of positive definite matrices, it is necessarily the smallest real eigenvalue (i.e., largest negative eigenvalue). This way, we obtain a semi-definite positive matrix with positive entries (this last assertion follows from the fact that the diagonal entries of a positive definite matrix are positive, since they are a sum of squares). This is a necessary condition for the matrix to be completely positive, but unfortunately, not a sufficient one for $N > 4$; also, it does not give a construction of the decomposition.

One c_0 has been found, one can apply a decomposition method to the corresponding completion (which is real positive semidefinite matrix) in order to obtain $\mathbf{Q}$, such as the square root or the Cholesky decomposition. The Cholesky decomposition is unique if we impose that diagonal entries are non-negative. The square

root is unique if we impose it to be positive semidefinite. Once $\mathbf{Q}$ has been found, we can choose the rates ν_k so that all entries of $\mathbf{P}$ are smaller than 1. Indeed, since $q_{i,k} = p_{i,k}\sqrt{\nu_k}$, choosing $\nu_k = \max_i q_{i,k}^2$ ensures that $p_{i,k} \leq 1$ for all i, k.

Unfortunately, this method does not ensure that the matrix $\mathbf{P}$ has only positive entries, so that a more general method is required in the general case.

Nonlinear optimization

An alternative approach consists in expressing the solutions of the problem as minima of an energy. For example, finding a correct mixture given the rates r_i and the correlation coefficients $c_{i,j}$ amounts to finding the matrix $\mathbf{P}$ and the vector ν which minimize

$$E = \sum_{i \neq j} (\sum_k p_{i,k} p_{j,k} \nu_k - c_{i,j})^2$$

with the constraints $p_{i,j} \in [0, 1]$, $\nu_k \geq 0$ and $\sum_k p_{i,k}\nu_k \leq r_i$ (for all i, j). These latter constraints are unfortunately not convex, but they can be transformed into an energy with

$$F_i = (\sum_k p_{i,k}\nu_k - r_i)^+$$

(or some smooth version, e.g. replacing the $\cdot^+$ operation by the hyperbola $f(x) = (x/2) + \sqrt{1 + x^2/4}$, or multiplying by a sigmoidal function (in $[0, 1]$), or even replacing with a square). Then a solution can be found with standard nonlinear minimization procedures, e.g. gradient descent, restricted on the hypercube for $\mathbf{P}$ and the positive cone for ν (simple clipping works). Calculation of the gradients gives the following expressions:

$$\begin{aligned}
\nabla_{\mathbf{P}}\mathbf{E} &= 4\mathbf{A}\mathbf{P}\, Diag(\nu) \\
\nabla_{\nu}\mathbf{E} &= 2\, Diag(\mathbf{P}^{\mathbf{T}}\mathbf{A}\mathbf{P}) \\
\nabla_{\mathbf{P}}\mathbf{F} &= H(\mathbf{P}\nu^{\mathbf{T}} - \mathbf{R})\nu \\
\nabla_{\nu}\mathbf{F} &= H(\mathbf{P}\nu^{\mathbf{T}} - \mathbf{R})^T\mathbf{P}
\end{aligned}$$

where ν is a row vector and H is the Heavyside function (applied on all components of a vector), and $\mathbf{A}$ is a matrix with entries $A_{ij} = c_{ij} - \sum_k p_{ij}p_{jk}\nu_k$ for $i \neq j$ and $A_{ii} = 0$. Empirically, gradient descent was able to find admissible solutions on all examples (i.e., with $E \approx 0$ and $F = 0$). A simple initialization is $\mathbf{P} = \mathbf{I}_N$ and $\nu = \mathbf{R}'$. (Note that $\mathbf{P}$ and ν must be completed to obtain the equality $\mathbf{R} = \mathbf{P}\nu$, as explained in section 4.5).

References

Bair, W. 1999. Spike timing in the mammalian visual system. *Curr. Opin. Neurobiol.*, **9**(4), 447–453.

Berman, A., & Shaked-Monderer, N. 2003. *Completely Positive Matrices*. World Scientific Publishing Company.

Daley, D.J., & Vere-Jones, D. 2005. *An Introduction to the Theory of Point Processes. Volume I: Elementary Theory and Methods*. 2nd edn. Springer.

Destexhe, A., & Pare, D. 1999. Impact of network activity on the integrative properties of neocortical pyramidal neurons in vivo. *J. Neurophysiol.*, **81**(4), 1531–1547.

Feng, J., & Brown, D. 2000. Impact of correlated inputs on the output of the integrate-and-fire model. *Neural. Comput.*, **12**(3), 671–692.

Galan, R. F., Fourcaud-Trocme, N., Ermentrout, G. B., & Urban, N. N. 2006. Correlation-Induced Synchronization of Oscillations in Olfactory Bulb Neurons. *J. Neurosci.*, **26**(14), 3646–3655.

Gutig, R., Aharonov, R., Rotter, S., & Sompolinsky, H. 2003. Learning Input Correlations through Nonlinear Temporally Asymmetric Hebbian Plasticity. *J. Neurosci.*, **23**(9), 3697–3714.

Konig, P., Engel, A. K., & Singer, W. 1996. Integrator or coincidence detector? The role of the cortical neuron revisited. *Trends Neurosci.*, **19**(4), 130–137.

Kuhn, A., Aertsen, A., & Rotter, S. 2003. Higher-Order Statistics of Input Ensembles and the Response of Simple Model Neurons. *Neural Comp.*, **15**(1), 67–101.

Moreno, R., de la Rocha, J., Renart, A., & Parga, N. 2002. Response of spiking neurons to correlated inputs. *Phys. Rev. Lett.*, **89**(28 Pt 1), 288101.

Niebur, E. 2007. Generation of synthetic spike trains with defined pairwise correlations. *Neural Comput.*, **19**(7), 1720–1738.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. 1993. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

Roy, S. A., & Alloway, K. D. 2001. Coincidence detection or temporal integration? What the neurons in somatosensory cortex are doing. *J. Neurosci.*, **21**(7), 2462–2473.

Rudolph, M., & Destexhe, A. 2003. Tuning neocortical pyramidal neurons between integrators and coincidence detectors. *J. Comput. Neurosci.*, **14**(3), 239–251.

Salinas, E., & Sejnowski, T. 2000. Impact of correlated synaptic input on output firing rate and variability in simple neuronal models. *J. Neurosci.*, **20**, 6193–6209.

Salinas, E., & Sejnowski, T. J. 2001. Correlated neuronal activity and the flow of neural information. *Nat. Rev. Neurosci.*, **2**(8), 539–550.

Song, S., & Abbot, L. 2001. Cortical development and remapping through spike timing-dependent plasticity. *Neuron*, **32**, 339–350.

Song, S., Miller, K. D., & Abbott, L. F. 2000. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neurosci.*, **3**(9), 919–926.

Usrey, W., Alonso, J., & Reid, R. 2000. Synaptic interactions between thalamic inputs to simple cells in cat visual cortex. *J. Neurosci.*, **20**(14), 5461–5467.

Figure legends

Figure 1

Generation of a homogeneous pool of correlated spike trains with doubly stochastic processes. The average rate is 20 Hz, the correlation time constant is $\tau_c = 10$ ms and the total correlation strength is $\lambda = .3$. A. Underlying time-varying rate (Ornstein-Uhlenbeck process — N.B.: the value of the rate is calculated only at spike times). B. Ten sample spike trains. C. Population histogram of spike times for 200 spike trains. D. Cross-correlogram with 2 ms time bins for a pair of spike trains (calculated over 1 hour), and theoretical prediction (dashed line).

Figure 2

Generation of 5 correlated spike trains with doubly stochastic processes. Rates ranged between 20 and 25 Hz, and pair-wise correlations were chosen randomly (correlation time constant $\tau_c = 10$ ms). A. Five independent Ornstein-Uhlenbeck processes with unit variance are generated. The correlation matrix C is decomposed as $\mathbf{C} = \mathbf{L}\mathbf{L}^T$ (Cholesky decomposition). B. The rates for the 5 spike trains are calculated by mixing the independent spike trains with the matrix $\mathbf{L}$ ($\mathbf{X}(t) = \mathbf{R} + \mathbf{L}\mathbf{Y}(t)$). C. Five correlated spike trains are generated according to

these rates. D. The empirical cross-correlograms match the theoretical predictions (solid line).

Figure 3

Event-driven algorithm for generating a spike train with time-varying rate $x(t)$. Generate a realization of a Poisson spike train with rate M, and pick a number $y(t)$ at random in $[0, M]$ for each spike occurring at time t. Select only those spikes such that $y(t) \leq x(t)$. Equivalently, generate a realization of a uniform spatial Poisson process on $[0, T] \times [0, M]$ with intensity 1 (crosses) and select the points below the graph of $x(\cdot)$ (circled crosses).

Figure 4

The mixture method for homogeneous pools with correlation strength c and rate r. A. First method: spikes are randomly selected from a common source spike train. B. Second method: target spike trains consist of the union of an independent and a common spike train.

Figure 5

The general mixture method. M independent Poisson spike trains with rates ν_k are generated. Target spike trains are obtained by selecting spikes from the source spike trains at random, according to the probability matrix $\mathbf{P}$.

Figure 6

Generation of 7 spike trains with different rates and homogeneous synchronization with mixture processes. A. Rates of the spike trains. B. Generated spike trains over 500 ms with the given rates (A), correlation strength $c = 0.2$ and correlation time constant $\tau_c = 5$ ms. C. Empirical cross-correlogram for the first two spike trains and theoretical prediction (solid line). D. Empirical cross-correlogram for the next two spike trains and theoretical prediction (solid line).

Figure 7

Generation of spike trains with arbitrary pairwise correlations and rates (mixture process). A. Rates of 5 target spike trains. B. Correlation matrix for the target spike trains. The diagonal entries are unspecified (zero on the graph). C, D. The source rates (C) and the mixture matrix (D) are computed with a nonlinear optimization

algorithm (here, gradient descent). Ten source trains were used. E. Resulting correlated spike trains over 500 ms. F. The empirical cross-correlogram matches the theoretical prediction (correlation time constant $\tau_c = 5$ ms).

Figure 8

Response of an integrate-and-fire model to correlated inputs generated by the mixture method and comparison with analytical results obtained with a diffusion approximation (Moreno *et al.*, 2002). A. A leaky integrate-and-fire neuron is driven by 1000 correlated input spike trains (80% excitatory, 20% inhibitory) at 10 Hz with instaneous synapses, the total input is balanced (zero mean). B. Output firing rate as a function of a short correlation time constant τ_c for correlation strength $c = 0.001$ and different construction methods: doubly stochastic processes (+), doubly stochastic processes with firing rates 20 times higher and scaled synaptic weights (x), mixture process A (filled circles), mixture process B (empty circles) and theoretical prediction (solid line). C. Output firing rate as a function of a long correlation time constant τ_c for correlation strength $c = 0.02$ and different construction methods (as in B; qualitatively similar effects were obtained with $c = 0.001$).

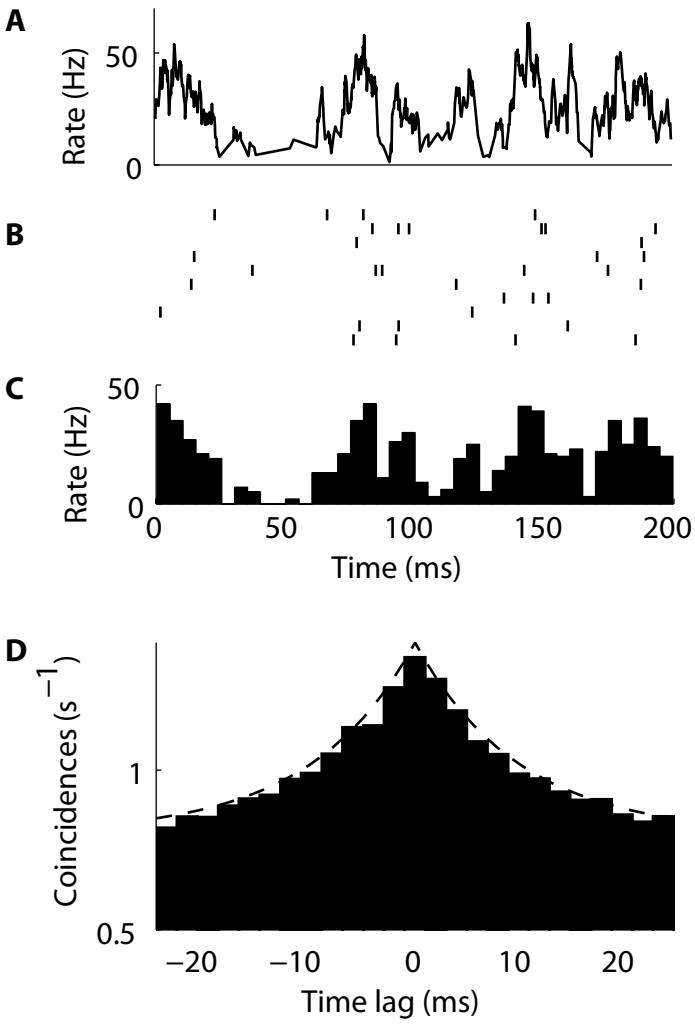
A
Rate (Hz)
50
0
B
C
Rate (Hz)
50
0
0
50
100
150
200
Time (ms)
D
Coincidences (s^{-1})
1
0.5
−20
−10
0
10
20
Time lag (ms)

Figure 1:

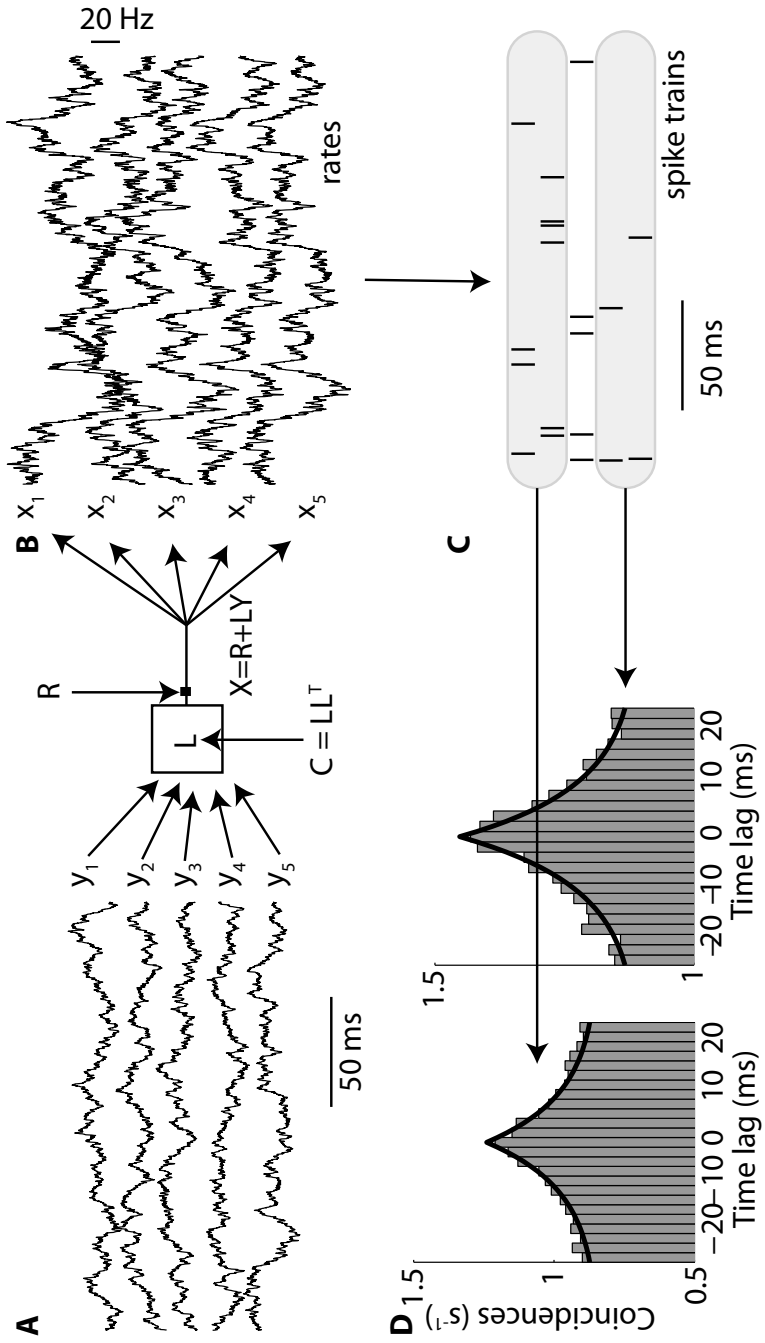
20 Hz
rates
spike trains
50 ms
50 ms
A
B
C
D
R
L
X=R+LY
C = LL^T
Coincidences (s^-1)
Time lag (ms)
Time lag (ms)

Figure 2:

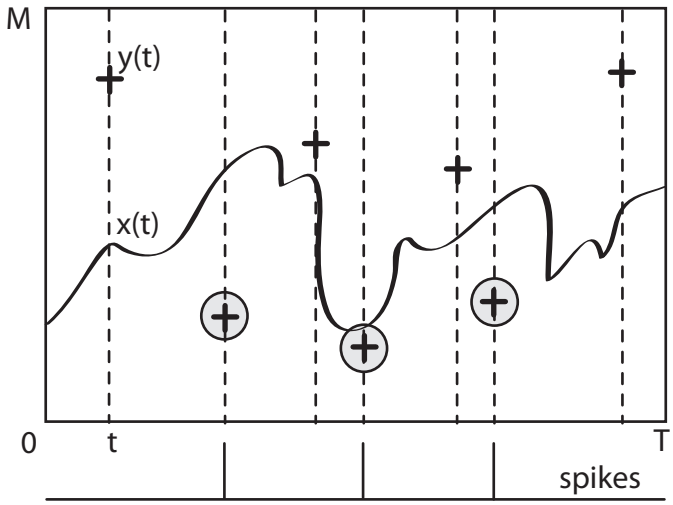

Figure 3:

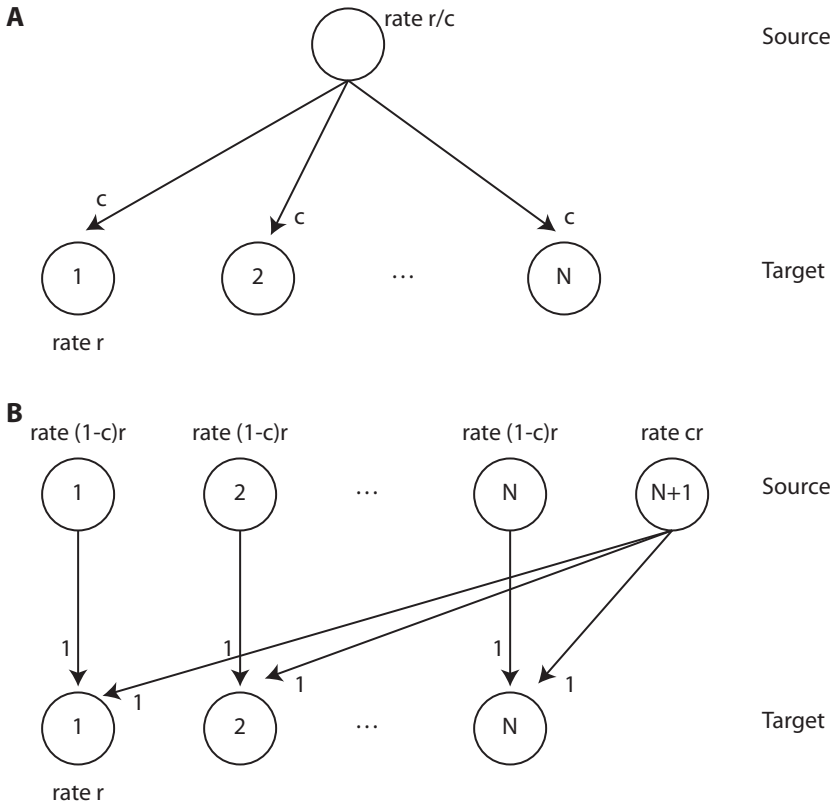
A
rate r/c
Source
c
c
c
1
2
…
N
Target
rate r
B
rate (1-c)r
rate (1-c)r
rate (1-c)r
rate cr
1
2
…
N
N+1
Source
1
1
1
1
1
1
1
2
…
N
Target
rate r

Figure 4:

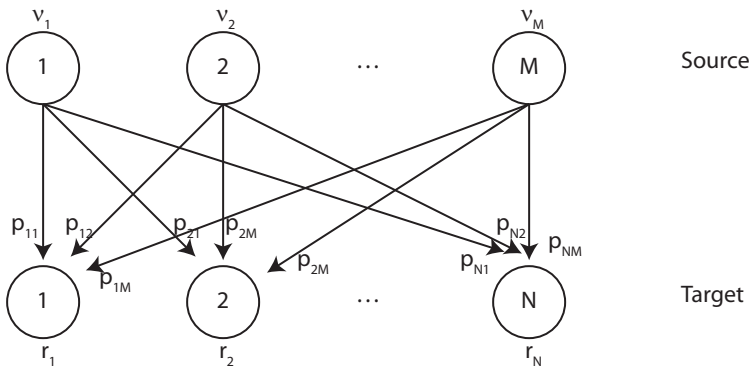

Figure 5:

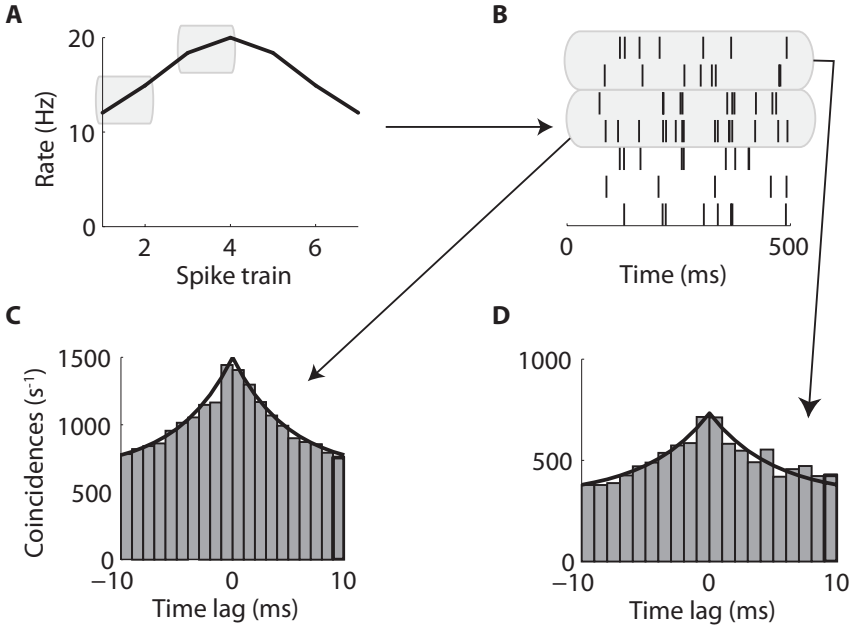

Figure 6:

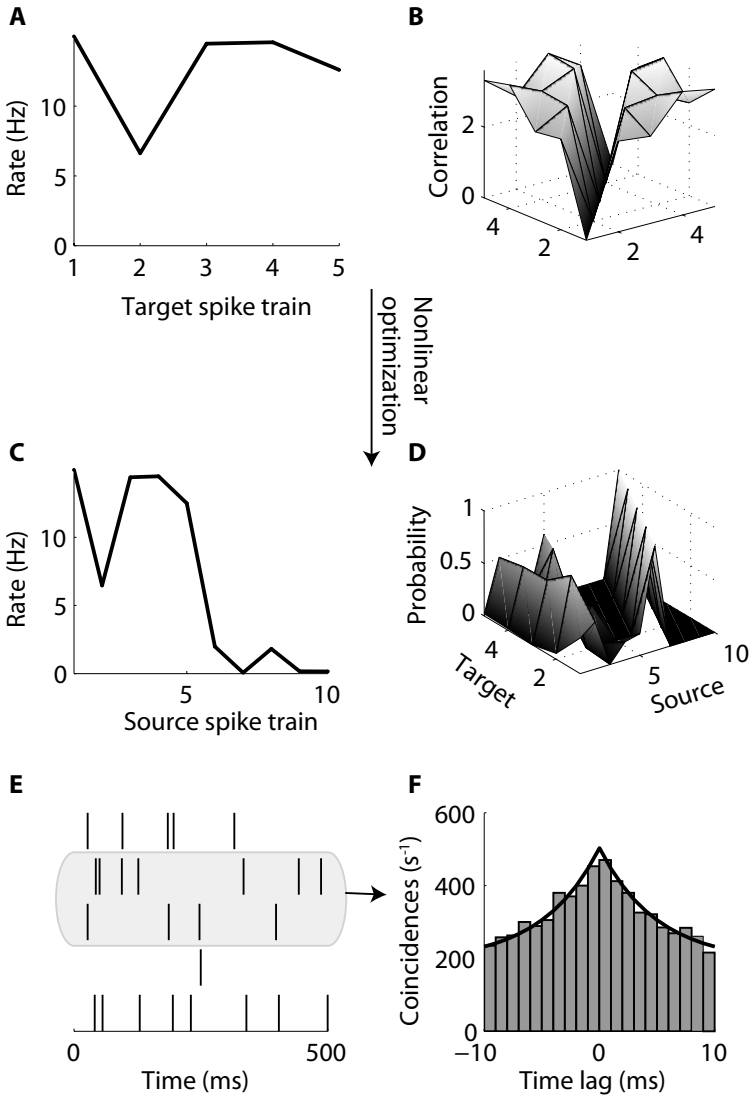
A
Rate (Hz)
10
5
0
1
2
3
4
5
Target spike train
B
Correlation
2
0
4
2
2
4
Nonlinear optimization
C
Rate (Hz)
10
5
0
5
10
Source spike train
D
Probability
1
0.5
0
4
2
Target
5
10
Source
E
0
500
Time (ms)
F
Coincidences (s-1)
600
400
200
0
−10
0
10
Time lag (ms)

Figure 7:

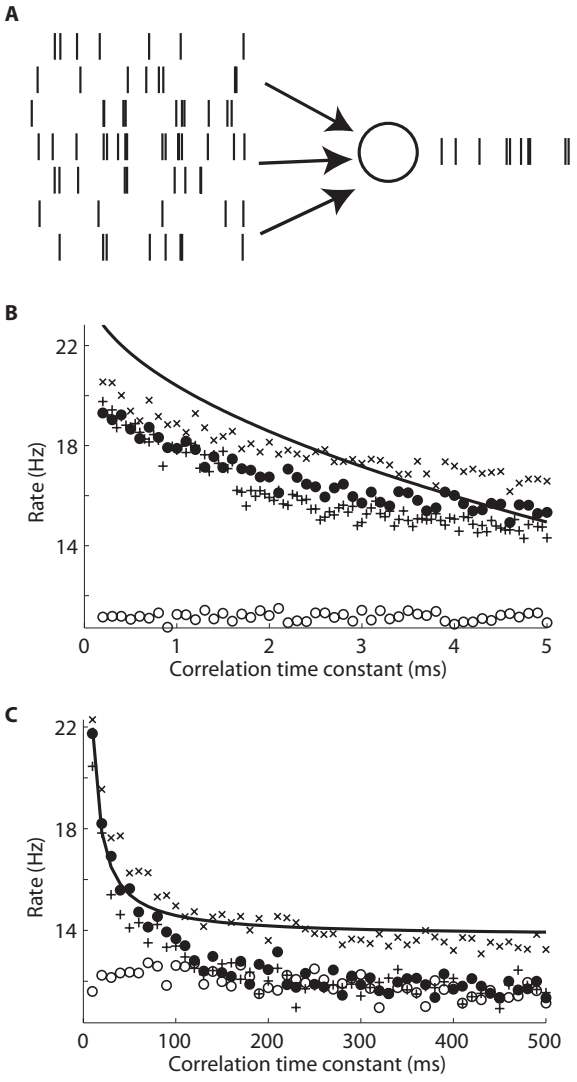
A
B
22
18
14
Rate (Hz)
0
1
2
3
4
5
Correlation time constant (ms)
C
22
18
14
Rate (Hz)
0
100
200
300
400
500
Correlation time constant (ms)

Figure 8: