Details on the code

We present the code on simulated data. The code is divided in 3 main parts.

The first part is the script named `script-spiketrains.py` for Python. In this script the data are simulated, the graph of connectivity is estimated with `ADM4` and the goodness-of-fit test is performed.

The second part is the script named `script-NPL` for `R`. In this script the data are estimated with `NPL` are the goodness-of-fit test is performed. It is based on the repository `hawkes` of François Gindraud from `https://github.com/lereldarion/hawkes`. The detailed are given directly in the script.

The third part is the script named `script-hawkesdiffusion.R` for `R`. This script needs the functions relegated in the script `Functions.R`. The jump-diffusion process with jumps driven by the Hawkes process simulated in the first script in Python is simulated. Then, the coefficients b, σ^2, a are estimated nonparametrically through adaptive penalized least squares estimators. Finally, a validation is proposed with a comparison with simple diffusion model (through depth notion).

The first and third parts presented in this document have been implemented by Charlotte Dion-Blanc and Sarah Lemler. The second part is implemented by Anna Bonnet and François Gindraud.

1 Python script for spike trains

The chosen parameters are: $M = 8$ neurons (or subjects), $n = 50$ trials and the observation time horizon has been fixed at $T = 100$. The adjacency matrix A which characterize the linear exponential Hawkes process is:

$$A = \begin{pmatrix} 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 \\ 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0.7 \end{pmatrix}.$$

The baseline parameter μ is fixed $\mu = 0.5$ for all neurons. The decay parameter β is also fixed equal for all neurons to $\beta = 5$. On Figure 1 we represent the histogram of all the spike trains for all the trials and on the right the spike trains of the first trial.

Then, the matrix A is estimated using `ADM4` algorithm. A grid of decays is proposed and the best one is chosen with the least squares lost. The result is given in Figure 2. The estimation is very accurate.

Then, we compute the goodness of fit test detailed in the article. To do so, on each trial an estimation of A is computed with `ADM4`. Then, the statistic for the test is given for each neuron and compared to

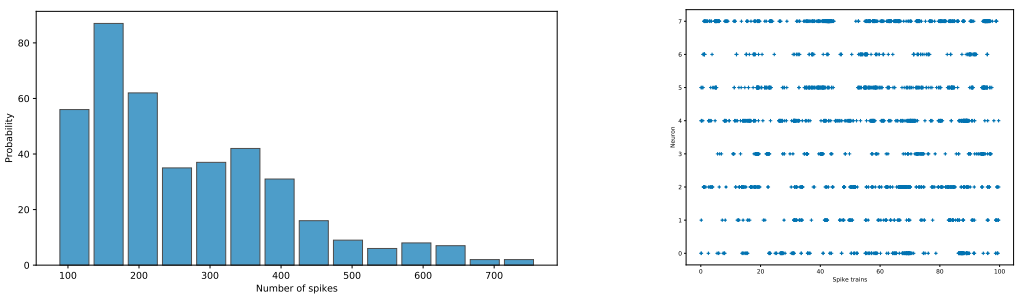

Figure 1: Simulated data. Left: histogram of all spikes, right: spike trains of the first trial.

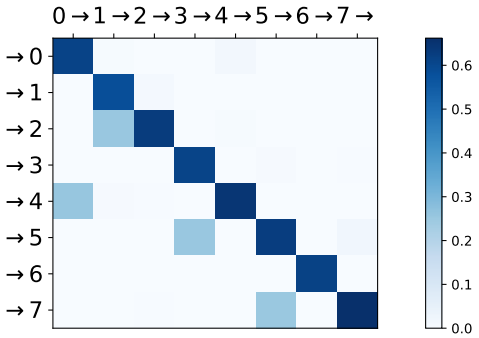

Figure 2: Estimated matrix A obtained with `ADM`, plotted using `tick` library.

the theoretical (and empirical quantile). We choose a subsample on size $\lfloor n^{2/3} \rfloor$ which is 13 in the given example.

The acceptation rates are given in Table 1 after 100 repetitions of the sampling of the subsample.

Neuron	Acceptation rate
1	0.75
2	0.97
3	0.48
4	1
5	0.86
6	0.68
7	0.7
8	0.42

Table 1: Acceptation rates of the Hawkes assumption (with 100 repetitions)

We have notice that the test performs well when the number of spikes is large enough (or often equivalently when the observation time is large enough).

2 R script for Hawkes-jump-diffusion model

To model the trajectory of the membrane potential of a fixed neuron taking into account the spikes from several neurons around, we have proposed the following model:

$$dX_t = b(X_t)dt + \sigma(X_t)dW_t + a(X_{t^-}) \sum_{j=1}^{M} dN_j(t), \tag{1}$$

where b is the drift function, σ^2 is the diffusion coefficient, a is the jump function and W is a standard Brownian motion independent of N which is a $M-$dimensional linear exponential Hawkes process. The conditional intensity of N is denoted by λ and do not depend on X. For each neuron j, it is defined by

$$\lambda_j(t) = \mu_j + \sum_{j'=1}^{M} \sum_{k:T_k^{(j')}<t} a_{j,j'} \beta e^{-\beta(t-T_k^{(j')})} \tag{2}$$

where $\mu_j > 0$ is the baseline intensity (or exogeneous intensity) and $a_{j,j'}$ is the interaction coefficient of j' on j, finally $(T_k^{(j')})_k$ is the sequence of jumps of neuron number j' observed until time t.

The aim of the script is first to estimate the unknown functions λ_j for $j = 1, \dots, M$, σ^2, a and b in Model (1) from the data and then to simulate the trajectory of a membrane potential driven by the jumps of the neurons around it from Model (1):

The jumps of the M neurons have been simulated in Python before. Then, we compute the estimators for b, σ^2 and a from procedures detailed in [2] and [1]. The estimations are based on the increments of

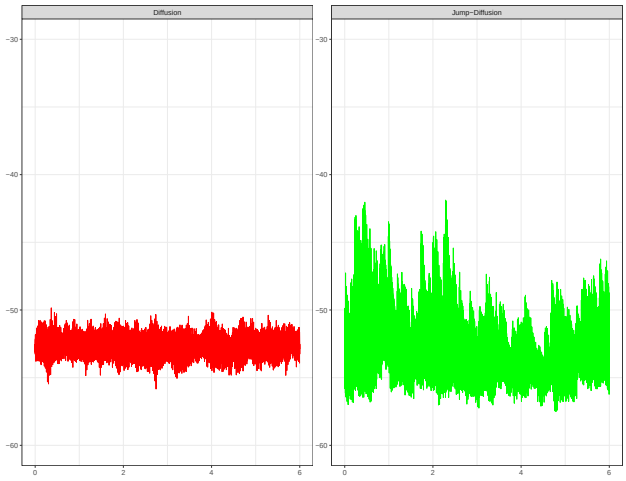

Figure 3: Comparison between diffusion paths and Hawkes-jump-diffusion paths simulated from the estimation of the coefficients.

$X_{k\Delta}$ and on the spike trains (for b and a). To compute the estimator of a we are using the knowledge of the kernel form of the Hawkes process (exponential) and the estimation of the parameters obtained by `ADM4`.

Once these estimators are obtained we are able to simulate new data from the equation:

$$dX_t = \widehat{b}(X_t)dt + \widehat{\sigma}(X_t)dWt + \widehat{a}(X_{t^-})\sum_{j=1}^{M} d\widehat{N}_t^{(j)}. \tag{3}$$

We do the same for a pure diffusion process. We are able to simulate trajectories from

$$dX_t = \tilde{b}(X_t)dt + \tilde{\sigma}(X_t)dWt.$$

Figure 3 represent 5 of each trajectories.

To end the study, we compute the depth of the first trajectory of the "real" one and the one with the estimated coefficients, in a bundle of size 100 of path coming from each one of the two processes (the one without jumps and the other with jumps). The result is:

`"Depth of the real trajectory in the bundle of diffusion paths: 0.18 Depth of the real trajectory in the bundle of jump-diffusion paths: 0.62"`, which is the attended result, because the closer the depth is to 1, the closer the trajectory is to the true trajectory. So the trajectory with jumps is closer from the "real" one than the one without jump.

More precisely, in the R script `function.R`, we have the following functions that are implemented:

- `intensM(t, param, times)` is a function that allows to implement M intensity functions of M exponential Hawkes processes, the arguments are
  - t, a real non-negative number
  - param, a list of size 3 with ξ a vector of size M
  - alpha, a matrix of size $M \times M$; beta a vector of size M
- `intens(k, M, s,times, param,xi)` is a function that sums the intensities up to an integer $k < M$, the arguments are
  - s, the time at which the intensity is calculated
  - times, the jump times defined as a list of size M
  - param, a list of size 3 with the parameters as above
  - k, the number of neuron up to which we sum up
  - M, total number of neurons

  - xi is the spontaneous firing rate

- `simuHawkesExpoM(param, M)` is a function that allows to simulate jump times from an exponential Hawkes process, the arguments are
  - param, a list of size 3 with the first list for ξ, the second list for the adjacency matrix and the third list for the decays
  - M, total number of neurons

- `a(x,namea)` is a function that allows to choose the form of the function a, the arguments are
  - x, a sequence of real number at which the function is calculated
  - namea, the form of a (none for $a(x) = 0$, constant for $a(x) = 1$, lin for $a(x) = x\mathbb{1}_{[-5,5]}(x) - 5\mathbb{1}_{(-\infty,-5)} + 5\mathbb{1}_{(5,+\infty)}$, lin2 for $a(x) = -0.1x$ and lin3 for $a(x) = \text{mean}(x) - x$)

- `projectionSm(x,q1,q2,m)` is the function that implements the trigonometric basis, the arguments are
  - x, a sequence of real number at which the trigonometric basis is calculated
  - q_1 and q_2, the bounds of the estimation interval
  - m is such that $D_m = 2m + 1$ is the dimension of the subset S_m of $L^2([q_1, q_2])$ on which we project the function to be estimated

- `alphachapeau(P,U)` returns the least squares estimator, the arguments are
  - P a matrix of size $D_m \times N$, where $N = \text{length}(U)$ and D_m is the
  - U a vector of length N

- `collecestimcoeff(X, U, q1, q2, Nn)` returns the collection of estimators of the decomposition of the estimator in the trigonometric basis for each m and the trigonometric basis evaluated at X, the arguments are
  - X a vector of observations
  - U a vector of size N
  - q_1 and q_2 are the bounds of the estimation interval
  - N_n is an integer such that the dimension of the largest model $\mathcal{S}$ is of dimension $2N_n + 1$

- `penaltyb(m, n, Delta, rho, sigma02)` is the penalty term in the adaptation procedure for b, the arguments are
  - m is such that $D_m = 2m + 1$ is the dimension of the subset S_m of $L^2([q_1, q_2])$ on which we project the function to be estimated
  - n the size of the panel
  - Delta the time step
  - rho a universal constant
  - sigma02 a bound of the volatility σ^2

- `penaltyg(Nn,n,Delta,kap)` is the penalty term in the adaptation procedure for g, the arguments are
  - N_n is an integer such that the dimension of the largest model $\mathcal{S}$ is of dimension $2N_n + 1$
  - n the size of the panel
  - Delta the time step
  - kap a universal constant

- `penaltysig(Nn,n,kap)` is the penalty term in the adaptation procedure for σ^2, the arguments are
  - N_n is an integer such that the dimension of the largest model $\mathcal{S}$ is of dimension $2N_n + 1$
  - n the size of the panel

- kap a universal constant

- `adaptiveestim(colleccoeffalpha,collecmatP, U,penalty)` returns $\hat{f}_m$, $\gamma_n(\hat{f}_m)$, $\gamma_n(\hat{f}_m)+pen(m)$ for a function f to be estimated and for different m and $\hat{m}$, the arguments are
  - colleccoeffalpha least squares estimators of the coefficients in the decomposition of the estimator of f in the trigonometric basis
  - collecmatP is the trigonometric basis evaluated at the observations
  - U, a vector of size N
  - penalty is one of the penalty function described above

  We applied this function with $f = \sigma^2$, $f = \sigma^2 + a^2$ and $f = b$ to obtain $\hat{\sigma}^2_{\hat{m}}$, $\hat{\sigma}^2_{\hat{m}} + \hat{a}^2_{\hat{m}}$ and $\hat{b}_{\hat{m}}$ respectively.

- `phifunc(x)` is a smooth version of the truncation function $\varphi(x) = \mathbb{1}_{\{|x|<1\}} + \exp\left(\frac{1}{3} + \frac{1}{x^2-4}\right)\mathbb{1}_{\{1\leq x<2\}}$

- `mNW(x, X, Y, h, K = dnorm)` returns the Nadaraya-Watson estimator, the arguments are
  - x a sequence of real values in which the function is calculated
  - X vector of size n with the predictors
  - Y vector of size n with the response variable
  - h the bandwidth
  - K the kernel (here we choose the gaussian kernel)

- `simu_jumpdiff(X0, grid, bfunc, sigfunc, afunc, Jumps, isjumpN)`
  returns the diffusion with jumps driven by Hawkes process, the arguments are
  - X_0 initialization point
  - grid, a grid of points
  - bfunc, the drift b
  - sigfunc, the volatility σ^2
  - afunc, the jump function a
  - Jumps, the jump times of the Hawkes process
  - isjumpN, a Boolean to determine if there is a jump in the considered time interval or not

- `simu_diff(X0, grid, bfunc, sigfunc)`
  returns a diffusion process, the arguments are
  - X_0 initialization point
  - grid, a grid of points
  - bfunc, the drift b
  - sigfunc, the volatility σ^2

Now we describe the methodology of the R script `script−hawkesdiffusion.R` on simulated data.

1. Define the time step `Delta`, the grid of points `grid` and the number of neurons `M`
2. Download the spike trains of the M neurons in `trial1`
3. In `jumptimes`, we put the jump times of the M neurons that are in the time interval that we consider
4. We construct a Boolean matrix `isjump1` in which we put 1 in the coordinate (i, j) if the jump time j is in the time interval (grid[i],grid[i+1]] and 0 otherwise, and then we compute the jump process by summing the rows of `isjump1`
5. We propose three functions for b, σ^2 and a

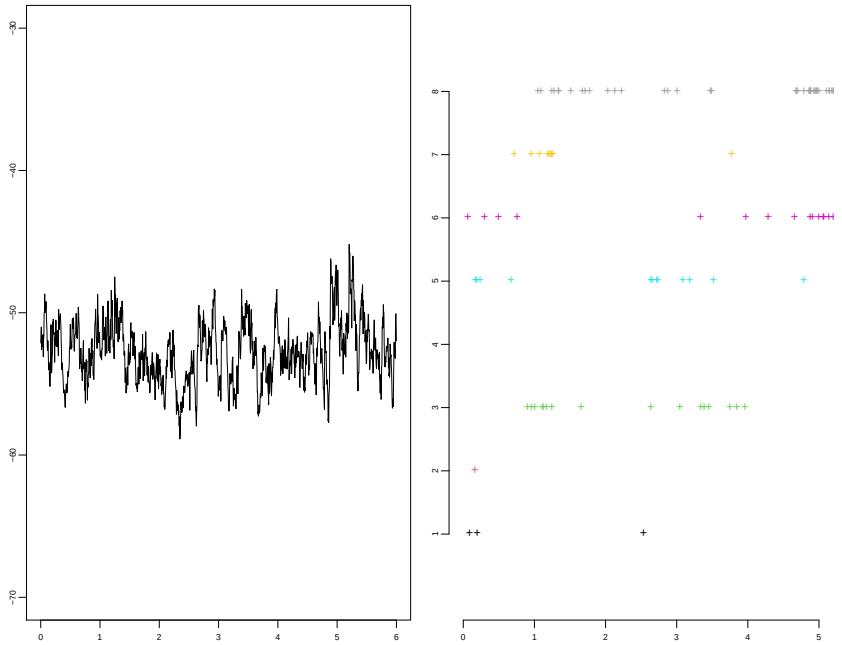

Figure 4: Diffusion process with jumps from Model (1) and jumps of the M neurons (each line correspond to one neuron)

6. From these functions and the jump process, we simulate the membrane potential from model (1), using the function `simu jumpdiff` defined above

7. We represent the trajectory of the diffusion process and of the jumps (see Figure 4)

8. We estimate the conditional intensity of the Hawkes process from the adjacency matrix and the baseline function obtained with `ADM4` in Python, `paramhawkes` is a list of size 3 with the adjacency matrix, the baseline intensity and the vector of decays respectively. We deduce from that an estimation of the conditional expectation $\mathbb{E}[\lambda_j(t)|X_t]$ using a Nadaraya-Watson estimator with the function `mNW`.

9. We estimate $g = \sigma^2 + a^2 f$ (where $f(X_t) = \sum_{j=1}^{M} \mathbb{E}[\lambda_j(t)|X_t]$) from the functions `collecestimcoeff`, `projectionSm` and `adaptiveestim` detailed above. We refer the reader to [1] for more details on the estimation procedure and the definition of `Tquad`.

10. We estimate σ^2 from the functions `collecestimcoeff`, `projectionSm` and `adaptivesig` detailed above. We refer the reader to [1] for more details on the estimation procedure and the definition of `Tquadphi`.

11. We deduce an estimator of a using the estimators of g, f and σ^2: $\hat{a}^2_{\hat{m}_1,\hat{m}_2} = \dfrac{\hat{g}_{\hat{m}_1} - \hat{\sigma}^2_{\hat{m}_2}}{\hat{f}}$, where $\hat{f}$ is the Nadaraya-Watson estimator.

12. We estimate the drift b. For that we approximate $\hat{\sigma}^2_{\hat{m}_1}$ by a constant taking the mean of the values of $\hat{\sigma}^2_{\hat{m}_1}$ and a linear approximation of $\hat{a}_{\hat{m}_1,\hat{m}_2}$ is made. We use the functions `collecestimcoeff`, `projectionSm` and `adaptiveb` described above to obtain an estimator of b and we refer the reader to [2] for more details on the estimation procedure and the definition of Y and U. We also make a linear approximation of the resulting estimator $\hat{b}_{\hat{m}}$.

13. We plot all the estimators and their approximations.

14. We plot the trajectory of the membrane potential from Model (3). For that, we have to simulate the jumps from a Hawkes process using the function `simulateHawkes`.

15. We also provide the code to estimate the unknown functions b and σ^2 in a pure diffusion model without jumps.

16. The trajectory obtained from the jump diffusion process is compared to that obtained from a diffusion process without jump using the notion of depth of each curves.

References

[1] C. Amorino, C. Dion, A. Gloter, and S. Lemler. On the nonparametric inference of coefficients of self-exciting jump-diffusion. arXiv:2011.12387, 2020.

[2] C. Dion and S. Lemler. Nonparametric drift estimation for diffusions with jumps driven by a hawkes process. To appear Stochastic Inference for Stochastic Processes, 2020.