Mixedsde: a R package to fit mixed stochastic differential equations

Charlotte Dion, Simone Hermann, Adeline Samson

`Mixedsde`: an `R` package to fit mixed stochastic differential equations

Charlotte Dion[(1),(2)], Simone Hermann[(3)], Adeline Samson[(1)]

(1) *UMR CNRS 5224, Laboratoire LJK, Université Grenoble Alpes, France*
(2) *UMR CNRS 8145, Laboratoire MAP5, Université Paris Descartes, Sorbonne Paris Cité, France*
(3) *TU Dortmund University, Faculty of Statistics, Germany*

Abstract

Stochastic differential equations (SDEs) are useful to model continuous stochastic processes. When (independent) repeated temporal data are available, variability between the trajectories can be modeled by introducing random effects in the drift of the SDEs. These models are useful to analyse neuronal data, crack length data, pharmacokinetics, financial data, to cite some applications among other. The `R` package focuses on the estimation of SDEs with linear random effects in the drift. The goal is to estimate the common density of the random effects from repeated discrete observations of the SDE. The package `mixedsde` proposes three estimation methods: a Bayesian parametric, a frequentist parametric and a frequentist nonparametric method. The three procedures are described as well as the main functions of the package. Illustrations are presented on simulated and real data.

Keywords: `R`, simulation, data analysis, nonparametric estimation, parametric estimation, Bayesian method.

1 Introduction

Continuous stochastic processes are usually observed discretely in time (with equidistant time points or not) leading to times series, although their intrinsic nature is of continuous time. While discrete time stochastic models such as auto-regressive models (ARMA, GARCH, ...) have been widely developed for time series with equidistant times, more and more attention have been focused on Stochastic Differential Equations (SDEs). Examples of applications where SDEs have been used include dynamics of thermal systems (Bacher *et al.*, 2011), solar and wind power forecasting (Iversen *et al.*, 2014), neuronal dynamics (Ditlevsen and Samson, 2014), pharmacokinetic/pharmacodynamic (PK/PD) (Hansen *et al.*, 2014), crack growth (Hermann *et al.*, 2016).

Depending on the applications, independent repeated temporal measures might be available. For examples, drug concentration of several subjects is usually observed in PK; dynamics of several neurons is measured along time; time to crack lengths can be measured repeatedly in crack growth study. Each trajectory represents the behaviour of a unit/subject. The functional form is similar for all the trajectories. Fitting the overall data simultaneously obviously improves the quality of estimation, but one has to take into account these variabilities between experiments. This is the typical framework of mixed-effects models where some parameters are considered as random variables (random effects) and proper to each trajectory. Hence the random effects represent the particularity of each subject. Some parameters can also be considered as common to all the trajectories (fixed effects).

In this work the model of interest is thus a mixed-effects stochastic differential equation (MSDE), mixed-effects for both fixed and random effects. The `mixedsde` package has been developed to estimate the density of the random effects from the discrete observations of M independent trajectories of a MSDE. It is available from `https://github.com/charlottedion/mixedsde`.

More precisely, we focus on MSDE with linear drift. We consider M diffusion processes $(X_j(t), t \geq 0)$, $j = 1, \ldots, M$ with dynamics ruled by SDE, for $t \in [0, T]$

$$\begin{cases} dX_j(t) &= (\alpha_j - \beta_j X_j(t))dt + \sigma a(X_j(t))dW_j(t) \\ X_j(0) &= x_j \end{cases} \tag{1}$$

where $(W_j)_{1 \ldots j \ldots M}$ are M independent Wiener processes, (α_j, β_j) are two (random) parameters, $\sigma a(X_j(\cdot))$ is the diffusion coefficient with a a known function and σ an unknown constant. The initial condition x_j is assumed fixed (and known) in the paper with possibly different values for each trajectory.

In the package, we restrict the models to the two famous SDEs with linear drift, namely the Ornstein-Uhlenbeck model (OU) with $a(x) = 1$ and the Cox-Ingersoll-Ross model (CIR) with $a(x) = \sqrt{x}$. For the CIR model, we assume that $x_j > 0$, $\sigma > 0$, $\alpha_j > \sigma^2/2$ and $\beta_j > 0$ to ensure that the process never crosses zero.

The random parameters are denoted ϕ_j and belong to $\mathbb{R}^d$ with either $d = 1$ or $d = 2$:

- $(d = 1)$ $\phi_j = \alpha_j$ random and for all $j = 1, \ldots, M$, $\beta_j = \beta$ fixed
- $(d = 1)$ $\phi_j = \beta_j$ random and for all $j = 1, \ldots, M$, $\alpha_j = \alpha$ fixed
- $(d = 2)$ $\phi_j = (\alpha_j, \beta_j)$ random

The ϕ_j's are assumed independent and identically distributed (*i.i.d.*) and independent of the W_j's. The `mixedsde` package aims at estimating the random effects ϕ_j and their distribution whose density is denoted f, from N discrete observations of the M trajectories $(X_j(t))_j$ from equation (1) at discrete times $t_0 = 0 < t_1 < \ldots < t_N = T$ (not necessarily equidistant). To the best of our knowledge, this is the first package in `R` language dedicated to the estimation of MSDE.

Estimation procedures for MSDE have been proposed in the non-parametric and the parametric frameworks, with a frequentist and a Bayesian point of view. The parametric approaches assume Gaussian random effects ϕ_j. Among other references, for parametric maximum likelihood estimation, we can cite Ditlevsen and de Gaetano (2005); Picchini *et al.* (2010) (Hermite expansion of the likelihood); Delattre *et al.* (2013) (explicit integration of the Girsanov likelihood) or Delattre *et al.* (2016) (mixture of Gaussian distributions for the random effects); for parametric Bayesian estimation, we can cite Oravecz et al. (2009) (restricted to Ornstein-Uhlenbeck) and Hermann *et al.* (2016) (general methodology); for non-parametric estimation, we can cite Comte *et al.* (2013); Dion (2014); Dion and Genon-Catalot (2015) (kernel estimator and deconvolution estimators).

Three estimation procedures are implemented in the `mixedsde` package: a kernel nonparametric estimator (Dion and Genon-Catalot, 2015), a parametric maximum likelihood estimator (Delattre *et al.*, 2013) and a parametric Bayesian estimator (Hermann *et al.*, 2016). The parametric frequentist and Bayesian approaches assume the random effects Gaussian. The Bayesian approach seems the most appropriate method for a small time of observation T and a small number of trajectories M. The nonparametric approach can be used when no prior idea on the density is available and when T and M are both large enough. Finally, the parametric frequentist estimation can be used when data are symmetric with a large number of discrete observations.

This paper reviews in Section 2 the three estimation methods. An overview of the `mixedsde` package is given in Section 3 through a description of the main functions and of other related companion functions. The practical use of this package is illustrated in Section 4 on simulated data and in Section 5 on one real dataset in neuronal modelling.

2 Density estimation in mixed stochastic differential models

We briefly recall the methodology of the three estimators implemented in the `mixedsde` package. We start with the nonparametric approach, then the frequentist parametric Gaussian method and finally the Bayesian parametric Gaussian method.

2.1 Nonparametric estimation of the random effects density

The first step of the nonparametric approach is to estimate the random effects. The idea is to maximize the likelihood of the process X_j^φ solution of the stochastic differential equation with fixed φ. Assuming continuous observations of $(X_j(t), 0 \leq t \leq T)$, the likelihood function is obtained with the Girsanov formula:

$$\ell_T(\varphi) = \exp\left(\int_0^T \frac{\alpha - \beta X_j^\varphi(s)}{\sigma^2 a^2(X_j^\varphi(s))} dX_j(s) - \frac{1}{2}\int_0^T \frac{(\alpha - \beta X_j^\varphi(s))^2}{\sigma^2 a^2(X_j^\varphi(s))} ds \right).$$

Maximizing the likelihood yields to the following estimator of ϕ_j

$$A_j := V_j^{-1} U_j \tag{2}$$

where U_j and V_j are the two sufficient statistics of the model. They are explicit depending on the form of the random effects:

- α_j random and β known

$$U_j := \int_0^T \frac{1}{\sigma^2 a^2(X_j(s))} dX_j(s) + \beta \int_0^T \frac{X_j(s)}{\sigma^2 a^2(X_j(s))} ds,\ V_j := \int_0^T \frac{1}{\sigma^2 a^2(X_j(s))} ds$$

- β_j random and α known

$$U_j := -\int_0^T \frac{X_j(s)}{\sigma^2 a^2(X_j(s))} dX_j(s) + \alpha \int_0^T \frac{X_j(s)}{\sigma^2 a^2(X_j(s))} ds, \quad V_j := \int_0^T \frac{X_j(s)^2}{\sigma^2 a^2(X_j(s))} ds$$

- (α_j, β_j) random, denote $b(x) = (1, -x)^t$ with u^t the transposition of vector u. Here U_j is a column vector with size 2×1 and $V_j = (V_{j,k,\ell})_{k,\ell \in \{1,2\}}$ a 2×2 symmetric matrix:

$$U_j := \int_0^T \frac{b}{\sigma^2}(X_j(s)) dX_j(s),\ V_j := \int_0^T \frac{b\, b^t}{\sigma^2}(X_j(s)) ds. \tag{3}$$

Truncated versions of this estimator have been introduced for theoretical reasons. In the bidimensional case $\phi_j = (\alpha_j, \beta_j)$, Dion and Genon-Catalot (2015) propose the following estimator

$$\widehat{A_j} := A_j \mathbb{1}_{B_j}, \quad B_j := \{V_j \geq \kappa\sqrt{T} I_2\} = \{\min(\lambda_{1,j}, \lambda_{2,j}) \geq \kappa\sqrt{T}\} \tag{4}$$

with I_2 the 2×2 identity matrix and $\lambda_{i,j}, i = 1, 2$ the two eigenvalues of the symmetric non negative matrix V_j, and κ a numerical constant calibrated in practice. In the one-dimensional case $\phi_j = \beta_j$ with $\alpha = 0$, Genon-Catalot and Larédo (2016) propose

$$\widehat{A_j} := A_j \mathbb{1}_{V_j \geq \kappa\sqrt{T}} \tag{5}$$

with κ a numerical constant calibrated in practice.

Based on these estimators of the ϕ_j's, we can proceed to the second step, the estimation of their density f. Several nonparametric estimators of f have been proposed (see Comte *et al.*, 2013, for example). In the package `mixedsde`, we focus on the kernel estimator of f. Let us introduce the kernel function $K : \mathbb{R}^d \to \mathbb{R}$, with $d = 1, 2$ depending on the dimension of ϕ_j. We assume K to be a $\mathcal{C}^2$ function satisfying

$$\int K(u)du = 1, \quad \|K\|^2 = \int K^2(u)du < +\infty, \quad \int (\nabla K(u))^2 du < +\infty$$

(with ∇K the gradient of K). A bandwidth $h \in (\mathbb{R}^+)^d$, for $d = 1, 2$, is used to define the function

$$K_h(x) = \frac{1}{h} K\left(\frac{x}{h}\right), x \in \mathbb{R}^d.$$

Note that in the bidimensional case, $h = (h_1, h_2)$ and the two marginal bandwidths are different. The nonparametric estimator of the density f of ϕ_j is

$$\widehat{f_h}(x) = \frac{1}{M} \sum_{j=1}^{M} K_h(x - A_j). \tag{6}$$

and the estimator $\widehat{\widehat{f_h}}(x) = \frac{1}{M} \sum_{j=1}^{M} K_h(x - \widehat{A}_j)$ is computed when the truncated estimator $\hat{A}_j$ is different than A_j.

In the `mixedsde` package, Gaussian kernel estimators are implemented with the `R`-functions `density` (available in package `stats`) when $d = 1$ and `kde2d` (available in package `MASS`) when $d = 2$ with an automatic selection of the bandwidth h. Note that when there is only one random effect, the bandwidth is selected by cross-validation with the argument `bw="ucv"`, or as the default value given by the rule-of-thumb if the chosen bandwidth is to small. Note that the estimator is unstable for small variance of the random effects.

It is important to notice that the two random effects are not assumed independent. When there is only one random effect, the fixed parameter has to be entered by the user.

The computation of $A_j = V_j^{-1} U_j$ does not require the knowledge of σ^2 as it appears both in U_j and V_j. It requires however the evaluation of the two continuous integrals in U_j and V_j while observing the trajectories (X_j) at discrete times $(t_0, t_1, \ldots, t_N)$. For $\Delta_k = t_{k+1} - t_k$, $k = 0, \ldots, N-1$, the non-stochastic integrals $\int_0^T g(X_j(s))ds$ for any function g are approximated by

$$\int_0^T g(X_j(s))ds \approx \sum_{k=0}^{N-1} g(X_j(t_k))\Delta_k.$$

For the stochastic integrals, we use the following simple discretization

$$\int_0^T g(X_j(s))dX_j(s) \approx \sum_{k=0}^{N-1} g(X_j(t_k))(X_j(t_{k+1}) - (X_j(t_k)))\Delta_k.$$

2.2 Frequentist parametric estimation approach

In this section and the following one, we assume that the random parameters ϕ_j are Gaussian:

- when $d = 1$, $\phi_j \sim \mathcal{N}(\mu, \omega^2)$ with $\mu \in \mathbb{R}$
- when $d = 2$, $\phi_j \sim \mathcal{N}(\mu, \Omega)$ with $\mu \in \mathbb{R}^2$ and a diagonal covariance matrix $\Omega = \text{diag}(\omega_1^2, \omega_2^2)$.

For the bidimensional case $d = 2$ we estimate by maximum likelihood the parameters $\theta := (\mu, \Omega)$. We define the likelihood function assuming first that the trajectories are continuously observed, similarly to the nonparametric approach (Section 2.1). Thanks to the Girsanov formula, the likelihood function of the j^{th} trajectory X_j is

$$L(X_j, \theta) = \frac{1}{\sqrt{\det(I_2 + \Omega V_j)}} \exp\left[-\frac{1}{2}(\mu - V_j^{-1}U_j)' R_j^{-1} (\mu - V_j^{-1}U_j)\right] \exp\left(\frac{1}{2} U_j' V_j^{-1} U_j\right)$$

with $R_j^{-1} = (I_2 + V_j\Omega)^{-1} V_j$ and I_2 is the 2×2 identity matrix.

For the case $d = 1$, the parameters to estimate are $\theta := (\mu, \omega, \psi)$ where ψ denotes the fixed effect α or β. We adopt the subscript r for the value of random, equal to 1 or 2, and c for the

position of the common fixed effect (thus 2 or 1). The likelihood function of the j^{th} trajectory X_j is

$$\begin{aligned} L(X_j,\theta) &= \frac{1}{\sqrt{1+\omega^2 V_{j,r,r}}} \exp\left[-\frac{1}{2} V_{j,r,r}(1+\omega^2 V_{j,r,r})^{-1}(\mu - V_{j,r,r}^{-1}(U_{j,r}-\psi V_{j,c,r}))^2\right] \\ &\quad \times \exp\left(\psi U_{j,c} - \frac{\psi^2}{2} V_{j,c,c}\right) \exp\left(\frac{1}{2}(U_{j,r}-\psi V_{j,r,c})^2 V_{j,r,r}^{-1}\right) \end{aligned}$$

with the notations U, V from (3). Details on this formula are available in the Appendix 6.

The likelihood function is defined as $L(\theta) = \prod_{j=1}^{M} L(X_j,\theta)$. The maximum likelihood estimator $\widehat{\theta} := (\widehat{\mu}, \widehat{\Omega}, \widehat{\psi})$ when $d=1$ and $\widehat{\theta} := (\widehat{\mu}, \widehat{\Omega})$ when $d=2$ is defined by

$$\widehat{\theta} = \arg\max_{\theta} L(\theta) = \arg\max_{\theta} \prod_{j=1}^{M} L(X_j,\theta)$$

This estimator is not explicit. In the `mixedsde` package, the function `optim` is used to maximize numerically the likelihood. Initialization of `optim` is obtained by computing the mean and the variance of the estimators A_j of the random parameters (see equation (2)). Sufficient statistics U_j and V_j are discretized as explained in Section 2.1.

Note that this parametric approach requires the knowledge of σ^2 to compute the sufficient statistics U_j and V_j because V_j appears alone in R_j. We plug the following estimator of σ^2

$$\widehat{\sigma^2} = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{N}\sum_{k=0}^{N-1}\frac{((X_j(t_{k+1})-X_j(t_k))^2}{\Delta_k a^2(X_j(t_k))}\right). \tag{7}$$

Selection of (non-nested) models can be performed with the BIC criteria, defined by $-2\log L(\widehat{\theta}) + 2\log(M)$ for model with one random effect and $-2\log L(\widehat{\theta}) + 4\log(M)$ with two random effects and the AIC criteria defined by $-2\log L(\widehat{\theta}) + 2$ for one random effect and $-2\log L(\widehat{\theta}) + 4$ for two random effects. These asymptotic criteria indicate the trade-off between maximizing fit and minimizing model complexity. Note that their theoretical properties are guaranteed only when σ^2 is known.

2.3 Bayesian parametric approach

For the Bayesian approach we assume similarly to the frequentist parametric estimation method a Gaussian distribution for ϕ_j, with a diagonal covariance matrix $\Omega = \text{diag}(\omega_1^2, \omega_2^2)$. In this method, we estimate in the same time the diffusion coefficient σ. The parameters of interest are thus $\theta = (\mu, \Omega, \sigma)$ and we want to estimate their posterior distribution $p(\theta|(X_j(t_k))_{j=1,\dots,M,k=1,\dots,N})$. Let denote $\mathbf{X}_{1:M} = (X_j(t_k))_{j=1,\dots,M,k=1,\dots,N}$ in the following.

We now introduce prior distributions implemented in `mixedsde` package for the parameters θ:

$$\begin{aligned} \mu &\sim \mathcal{N}(m, V),\ V = \text{diag}(v) \\ \omega_i^2 &\sim \text{IG}(\alpha_{\omega,i}, \beta_{\omega,i}),\ i = 1,2 \\ \sigma^2 &\sim \text{IG}(\alpha_\sigma, \beta_\sigma), \end{aligned}$$

where IG is the inverse Gamma distribution which is conjugate to the normal likelihood and $m, V, \alpha_{\omega,i}, \beta_{\omega,i}, \alpha_\sigma, \beta_\sigma$ are hyperparameters fixed by the user. The case of only one random effect is nested by setting ω_1^2 or ω_2^2 equal to zero.

The aim is to calculate the posterior distribution $p(\theta|\mathbf{X}_{1:M})$ which is not explicit for the whole vector of parameters. Therefore, we simulate it through a Gibbs sampler (see e.g. Robert and Casella, 2004). Here, we have a true transition density of both processes that is used for the

likelihood, see Iacus (2008). For a general hierarchical diffusion approach based on the Euler approximation, see Hermann *et al.* (2016).

Analogically to the frequentist approach, there is a first step: sample from the full conditional posterior of the random effects $p(\phi_j|(X_j(t_k))_{k=1,\dots,N},\theta), j=1,\dots,M$. This is done by a Metropolis Hastings (MH) algorithm.

The second step is the estimation of the hierarchical parameters μ and Ω. Full conditional posteriors $p(\mu|\phi_1,\dots,\phi_M,\Omega)$ (resp. $p(\Omega|\phi_1,\dots,\phi_M,\mu)$) are Gaussian (resp. inverse Gamma) and can, for example, be found in Hermann *et al.* (2016).

The last step of the Gibbs sampler is sampling from the full conditional posterior of σ^2. For the CIR model, this is also conducted by a MH step. For the OU model, the inverse Gamma distribution is conjugate to the normal likelihood. The full conditional posterior distribution is given by

$$\sigma^2|\mathbf{X}_{1:M},\phi_1,...,\phi_M \sim \text{IG}\left(\alpha_\sigma+\frac{MN}{2},\beta_\sigma+\frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{N}\frac{\beta_j}{1-e^{-2\beta_j\Delta_k}}\left(X_j(t_k)-\frac{\alpha_j}{\beta_j}-\left(X_j(t_{k-1})-\frac{\alpha_j}{\beta_j}\right)e^{-\beta_j\Delta_k}\right)^2\right).$$

In the case of one random effect, there is one additional Gibbs sampler step for the fixed effect, that is also conducted through a MH algorithm.

In the package, the starting values for the Gibbs sampler are set equal to the mean of the prior distributions. In all the MH algorithms, one each has to choose a proposal density. In the package `mixedsde`, we use a normal density for all location parameters with mean equal to the last chain iteration and a proposal variance that has to be chosen. For the CIR model, the proposal distribution for σ^2 is chosen by $\sqrt{\sigma^2}\sim\mathcal{N}(\sqrt{\sigma^2_{\text{prev}}},\text{variance})$ where σ^2_{prev} is the previous value of σ^2. The remaining question is how to choose the suitable proposal variance. This variance controls the chain dependence and the acceptance rate. If the variance is small, the acceptance rate is large and the chains gets very dependent. If the proposal variance is large, only few candidates are accepted with the advantage of weakly dependent chains. This problem is solved in the package with an adaptive Metropolis-within Gibbs algorithm (Rosenthal , 2011) using the proposal distribution $\mathcal{N}(0,e^{2l})$ with l the logarithm of the standard deviation of the increment. This parameter is chosen so that the acceptance rate is approximately 0.44 which is proposed to be optimal in the Metropolis-within Gibbs sampler. It is proposed to add/subtract an adoption amount $\delta(n)=\min(0.1,n^{-1/2})$ to/from t after every 50th iteration and adapt the proposal variance if the acceptance rate is smaller than 0.3 or larger than 0.6.

2.4 Predictions

In many cases, one is not only interested in parameter estimation but also in the prediction for future observations. The first step is the prediction of a future random effect ϕ_{pred}. The simulation of a new random effect is direct for the frequentist parametric approach sampling from $\mathcal{N}(\widehat{\mu},\widehat{\Omega})$. For the nonparametric approach, first note that $\widehat{f}_h$ is an estimator given on a discrete grid $\{x_1,\dots,x_n\}$, i.e. a vector of corresponding $\{p_1,\dots,p_n\}$ after normalisation. Simulating from the estimator $\widehat{f}_h$ can therefore be performed simulating a discrete variable from vector $\{x_1,\dots,x_n\}$ with (normalized) probabilities $\{p_1,\dots,p_n\}$. For the Bayesian approach, a new ϕ_{pred} is sampled from the predictive distribution $p(\phi_{\text{pred}}|\mathbf{X}_{1:M})=\int p(\phi_{\text{pred}}|\mu,\Omega)p(\mu,\Omega|\mathbf{X}_{1:M})\,d(\mu,\Omega)$ where the posterior of μ and Ω is approximated by the results of the Gibbs sampler. This distribution is not explicit, we propose to sample over a grid through inversion method, equal to the nonparametric case.

Given a new random effect ϕ_{pred}, we are able to simulate predictive trajectories. This is performed using the transition density $p(X(t_k)|X(t_{k-1}),\phi_{\text{pred}},\sigma^2)$ for the frequentist approach. The starting points of the process x_j are the observed ones. For the Bayesian approach, we implement

two prediction settings. Firstly, analogously to the frequentist approach a new trajectory is simulated using the transition density $p(X(t_k)|X(t_{k-1}), \phi_{\text{pred}}, \sigma^2)$ where ϕ_{pred} is sampled from the MCMC posterior distribution $p(\phi|X_{1:M})$. Secondly, we can calculate the predictive distribution

$$p(X(t_i)|\mathbf{X}_{1:M}) = \int p(X(t_i)|\phi_{\text{pred}}, \sigma^2) p(\phi_{\text{pred}}, \sigma^2|\mathbf{X}_{1:M})\, d(\phi_{\text{pred}}, \sigma^2)$$

in each time point. We can then calculate only the quantiles for a prediction interval or to draw directly samples from the predictive distribution. For this predictive distribution, we take the starting point $x_j = x_0$ to be the same for all series. If the starting points would vary, this is an additional random effect whose density has to be estimated. This is not implemented in the estimation procedure and will, therefore, left out for the prediction.

It is then interesting to compare the new trajectories to the real ones and if the number of new trajectories is large enough, to compute an empirical confidence interval.

3 Overview of the `mixedsde` functions

This Section presents an overview of the functions implemented in the package. Illustrations of the code are given in Section 4.

3.1 Data

Data is a matrix `X` of size $M \times N$ for M trajectories with N time points, not necessarily equidistant but similar for the M trajectories. The vector of times is a vector `times` of length N. Real datasets are available on the package, and detailed on Section 5.

To lead a simulation study, the function `mixedsde.sim` allows to generate a list with a $M \times N$ matrix `X` of M trajectories on the interval $[0, T]$ with N equidistant points (default value 100) and a vector `times` with the equidistant times. This function leans on function `sde.sim` available via package `sde` from the web address `http://cran.r-project.org/package=sde`, to simulate SDE. One has to choose: `model` either OU or CIR; `random` that fixes the position and the number of random effects: `random = 1` for α_j random, `random = 2` for β_j random or `random = c(1,2)` for α_j and β_j random; σ the diffusion coefficient; `invariant`, default value 0 means that X_0 is 0 (default) or fixed by the user, value 1 means that X_0 is generated from the invariant distribution (see details in the package documentation); `density.phi` to choose the distribution of the random effect (see package documentations).

3.2 Main function

Main function is `mixedsde.fit` producing estimation of the random effects and their common density. Inputs of `mixedsde.fit` are

- `X` a $M \times N$ matrix containing the trajectories by rows
- `times` the vector of observations times
- `model` the chosen model either OU or CIR
- `random` that fixes the position and the number of random effects: `random = 1` for α_j random, `random = 2` for β_j random or `random = c(1,2)` for α_j and β_j random
- `estim.method` the estimation method: `nonparam` (see Section 2.1), `paramML` (see Section 2.2) or `paramBayes` (see Section 2.3)
- `fixed` the value of the fixed effect β (resp. α) when `random = 1` (resp. `random = 2`), default 0. (Only for the frequentist approaches).
- `estim.fix` 1 if the fixed effect is estimated, default 0. (Only for the frequentist parametric approach when `random`=1 or 2).

Table 1 – Summary of the different methods for the two S4-classes `Freq.fit` and `Bayes.fit` resulting of the function `mixedsde`

class	Freq.fit	Bayes.fit
method	out	out
method	plot	plot
method	–	plot2compare
method	print	print
method	summary	summary
method	pred	pred
method	valid	valid

- `gridf` the x-axis grid on which the random effect distribution is computed: we recommend a fine grid with at least 200 points, default value is a sequence of length 500 starting in $0.8 \times \min_j \widehat{\phi}_j$ and ending in $1.2 \times \max_j \widehat{\phi}_j$. (Only for the frequentist approaches)
- `prior` the list of prior parameters `m, v, alpha.omega, beta.omega, alpha.sigma, beta.sigma` for `paramBayes` method: default values are calculated based on the estimations $(A_j)_j$ for the first $\min(3, \lceil M \cdot 0.1 \rceil)$ series and main estimation is only made with the remaining $\lfloor M \cdot 0.9 \rfloor$. (Only for the Bayesian approach)
- `nMCMC` length of the Markov chain for `paramBayes` method. (Only for the Bayesian approach)

In the following we describe the related methods, proposed in the package, they are summarised in Table 1.

3.3 Outputs

Output of `mixedsde.fit` is a S4 class called `Freq.fit` for the frequentist approaches and `Bayes.fit` for the Bayesian approach. Results of the estimation procedure are available as a list applying function `out` to the `Freq.fit` (resp. `Bayes.fit`) object.
Elements of `Freq.fit` are:

- `sigma2` estimator $\widehat{\sigma^2}$ of the diffusion coefficient
- `estimphi` estimator $(A_j)_j$ of the random effects
- `estimphi.trunc` truncated estimation $(\widehat{A_j})_j$ of the random effects
- `estim.fixed` estimator of the fixed effect if `random = 1` or 2, `estim.method = paramML; estim.fix = 1`, default 0
- `gridf` the x-axis grid on which the random effect distribution is computed
- `estimf` estimator of the density of the random effects (for both `paramML` and `nonparam` methods)
- `cutoff` binary M-vector of binary values indicated the truncated trajectories, default `FALSE` when no truncation
- `estimf.trunc` truncated estimation of the density of the random effects
- `mu` estimation of Gaussian mean of the random effects (only for `paramML` method)
- `omega` estimation of Gaussian variance matrix of the random effects (only for `paramML` method)
- `aic` and `bic` AIC and BIC criteria (only for `paramML` method)
- `index` index of trajectories used for the estimation, excluded are trajectories with $V_j = 0$ or $V_j = +\infty$ (one random effect) or $\det V = +\infty$ (two random effects), trajectories containing negative values for `CIR` model,

Elements of `Bayes.fit` are:

- `sigma2` trace of the Markov chain simulated from the posterior of σ^2
- `mu` trace of the Markov chain simulated from the posterior of μ
- `omega` trace of the Markov chain simulated from the posterior of ω^2
- `alpha` trace of the Markov chain simulated from the posterior of α_j, `nMCMC`$\times M$ matrix if α is random effect, `nMCMC`$\times 1$ otherwise
- `beta` trace of the Markov chain simulated from the posterior of β_j, `nMCMC`$\times M$ matrix if β is random effect, `nMCMC`$\times 1$ otherwise
- `burnIn` a proposal for the burn-in phase
- `thinning` a proposal for the thin rate
- `ind.4.prior` the indices used for the prior parameter calculation, $M+1$ if prior parameters were specified.

Outputs `burnIn` and `thinning` are only proposals for a burn-in phase and a thin rate. The proposed `burnIn` is calculated by dividing the Markov chains into 10 blocks and calculate the 95% credibility intervals and the respective mean. Starting in the first one, the block is taken as burn-in as long as the mean of the current block is not in the credibility interval of the following block or vice versa. The thinning rate is proposed by the first lag which leads to a chain autocorrelation of less than 80%. It is not easy to automate these choices, so it is highly recommended by the authors to verify the chains manually.

Command `plot()` applied to a `Freq.fit` object produces a frequencies histogram of $(A_j(T))_j$ (one or two according to the number of random effects) with the estimated density (red curve) and the truncated estimator if available (dotted grey red curve) and a quantile-quantile graph with the quantiles of the A_j's versus the quantiles of a normal sample of the same length, with the same empirical mean and standard deviation. This illustrates the normality of the sample. Applying this function to the nonparametric results indicates if the Gaussian assumption of the parametric approach is appropriate. When `plot()` is applied to a `Bayes.fit` object, one can choose four different options, named `style`. The default value is `chains`, it plots the Markov chains for the different parameter values. `acf` leads to the corresponding autocorrelation functions, `density` to the approximated densities for each parameter and `cred.int` leads to the credibility intervals of the random parameters with the input parameter `level` with default 0.05. For all options, with the input parameter `reduced = TRUE`, the burn-in period is excluded and a thinning rate is taken, default is `FALSE`. There is also a possibility to include the prior means in the plots by lines with `plot.priorMean = TRUE`, default is `FALSE`.

In the Bayesian estimation the influence of prior parameters is interesting, thus for the `Bayes.fit` object, there is a second plot method, named `plot2compare` where three estimation objects can be compared. For reasons of clarity, only the densities are compared, with the default `reduced = TRUE`. Here, there is also a possibility to include `true.values`, a list of the true parameters for the comparison in a simulation example.

Command `summary()` applied to a `Freq.fit` object computes the kurtosis and the skewness of the distribution, $\widehat{\sigma^2}$, the empirical mean and standard deviation computed from the estimators $(A_j)_j$, $\widehat{\mu}$, $\widehat{\Omega}$ (and the fixed effect $\widehat{\alpha}$ or $\widehat{\beta}$), AIC, BIC criteria for the frequentist MLE method. When applied to a `Bayes.fit` object, it computes means and credibility interval (default level 95%) for each parameter $(\mu, \Omega, \sigma, \alpha, \beta)$. Here, there is also a possibility to choose the burn-in and the thinning rate manually by the input parameters `burnIn` and `thinning`.

Command `print()` applied to a `Freq.fit` object returns the use or not of the cutoff and the vector of excluded trajectories. When applied to a `Bayes.fit` object, it returns the acceptance rates of the MCMC procedure.

3.4 Validation methods

Validation of a mixed model, obtained with function `valid`, is an individual validation. Indeed, the validation of trajectory j is obtained comparing it to M new trajectories simulated with parameters (α, β) fixed to the estimator A_j (or $\widehat{A}_j$) in the frequentist approaches and to the posterior means in the Bayesian approach. Inputs of the function are

- `Freq.fit` or `Bayes.fit` object
- `plot.valid` 1 to generate a figure (default value is 1)
- `numj` a specific individual trajectory to validate (default all trajectories)
- `Mrep` number of simulated trajectories (default value 100)

Each observation $X_{\text{numj}}(t_k)$ is compared with the `Mrep` simulated values $(X^1_{\text{numj}}(t_k), \ldots, X^{M_{\text{rep}}}_{\text{numj}}(t_k))$, for $k = 1, \ldots, N$.

Outputs are the list of the $(X^1_{\text{numj}}(t_k), \ldots, X^{M_{\text{rep}}}_{\text{numj}}(t_k))$. If `plot.valid`=1, two plots are produced. Left: plot of the `Mrep` new trajectories (black) and the true trajectory number `numj` (in grey/red). Right: quantile-quantile plot of the quantiles of a uniform distribution and the N quantiles obtained comparing $X_{\text{numj}}(t_k)$ with the `Mrep` simulated values $(X^1_{\text{numj}}(t_k), \ldots, X^{M_{\text{rep}}}_{\text{numj}}(t_k))$, for $k = 1, \ldots, N$.

This is an empirical method. The recent work Kuelbs and Zinn (2015) on depth and quantile regions for stochastic processes (see for example Zuo and Serfling (2000) for depth functions definitions) should provide the theoretical context for a more extensive study. This could be done in further works.

3.5 Prediction methods

Prediction (see Section 2.4) is implemented in function `pred`. Main inputs of the function are

- `Freq.fit` or `Bayes.fit` object
- `invariant` TRUE if the new trajectories are simulated according to the invariant distribution
- `level` level of the empiric prediction intervals (default 0.05)
- `plot.pred` TRUE to generate a figure (default TRUE)

(and optional plot parameters). Function `pred` applied to a `Freq.fit` object returns a list with predicted random effects `phipred`, predicted trajectories `Xpred` and indexes of the corresponding true trajectories `indexpred` (see Section 2.4 for details of simulation). If `plot.pred = TRUE` (default) three plots are produced. Left predicted random effects versus estimated random effects. Middle: true trajectories. Right predicted trajectories and their empirical 95% prediction intervals (default value `level=0.05`). The prediction can also be done from the truncated estimator $\widehat{\widehat{f}_h}$ based on the $\widehat{A}_j$ (5), if the argument `pred.trunc = 1`.

Function `pred` applied to a `Bayes.fit` object returns a S4 class object `Bayes.pred`. The first element of this class is `Xpred`, which depends on the input parameters. Including the input `trajectories = TRUE`, matrix `Xpred` contains the M drawn trajectories by rows (see first method described for the Bayesian approach in Section 2.4). Default is `trajectories = FALSE` which leads to the calculation of the predictive distribution explained in Section 2.4. With the input `only.interval = TRUE` (default), only the quantiles for the 1- `level` prediction interval are calculated, stored in `qu.l` and `qu.u`. Input `only.interval = FALSE` provides additionally `Xpred` containing `sample.length` (default 500) samples from the predictive distribution in each time point of the observations (except the first). In both cases, with `plot.pred = TRUE`, two figures are produced. On the left side, the data trajectories are compared with the prediction intervals and on the right side, the coverage rate is depicted which is stored in entry `coverage.rate`, namely the amount of series covered by the prediction intervals for each time point. The last class entry `estim` stores the results from the `Bayes.fit` object in a list. Other input parameters are `burnIn`

and `thinning` which allow for the choice of other burn-in phase and thinning rate than proposed in the `Bayes.fit` object.

For the `Bayes.pred` class object, two plot methods are available. `plot()` repeats the figures that are created with the `plot.pred = TRUE` command in the `pred` method. `plot2compare()` compares up to three `Bayes.pred` objects, where in a first figure the prediction intervals are presented in colors black, red and green and the observed data series in grey and in a second figure the corresponding coverage rates are compared. With the input parameter `names` a vector of characters to be written in a legend can be indicated.

4 Package `mixedsde` through simulated examples

In this part two simulated examples are given to illustrate the strengths of each proposed method. Two datasets are simulated according to:

1. CIR model with one non-Gaussian random effect $\beta_j \sim \Gamma(1.8, 0.8)$, $\alpha_j = 1$, $T = 50$, $M = 200$, $N = 1000$:

```
model1 <- "CIR"; random1 <- 2; fixed1 <- 1; sigma1 <- 0.1 ; M1 <- 200; T1 <- 50;
N1 <- 1000; X01 <- 1; density.phi1 <- "gamma"; param1 <- c(1.8, 0.8);

simu1 <- mixedsde.sim(M=M1, T=T1, N=N1, model=model1, random=random1,
        fixed=fixed1, density.phi=density.phi1, param=param1, sigma=sigma1, X0=X01)
X1<- simu1$X; phi1 <- simu1$phi; times1 <-simu1$times
```

2. OU model with one Gaussian random effect $\alpha_j \sim \mathcal{N}(3, 0.5^2)$, $\beta_j = 5$, $T = 1$, $M = 50$, $N = 500$:

```
model2 <- "OU"; random2 <- 1; sigma2 <- 0.1; fixed2 <- 5; M2 <- 50; T2 <- 1;
N2 <- 500; X02 <- 0; density.phi2 <- "normal"; param2 <- c(3, 0.5);

simu2 <- mixedsde.sim(M=M2, T=T2, N=N2, model=model2, random=random2,
        fixed=fixed2, density.phi=density.phi2, param=param2, sigma=sigma2, X0=X02)
X2 <- simu2$X; phi2 <- simu2$phi; times2 <- simu2$times
```

Example 1 has non Gaussian random effect, the nonparametric method is the most appropriate approach. Example 2 has T small and Gaussian random effect, nonparametric method is therefore not the most appropriate approach. Parametric methods should performed well with a preference to the Bayesian as the number of trajectories $M2 = 50$ is not large.

4.1 Frequentist nonparametric estimation

We illustrate nonparametric estimation on Example 1. Code for the nonparametric estimation is

```
estim.method <- 'nonparam'
estim_nonparam <- mixedsde.fit(times=times1, X=X1, model=model1, random=random1,
              fixed=fixed1, estim.method=estim.method)
outputsNP <- out(estim_nonparam)                    # stores the results in a list
```

Summary function provides:

```
summary(estim_nonparam, newwindow=FALSE)
     [,1]       [,2]
[1,] "kurtosis" "11.0008360676508"
[2,] "skewness" "2.16789214842904"
     [,1]           [,2]
[1,] "sigma"        "0.100164891397244"
[2,] "empiric mean" "1.38241196209623"
[3,] "empiric sd"   "1.10890963312551"
```

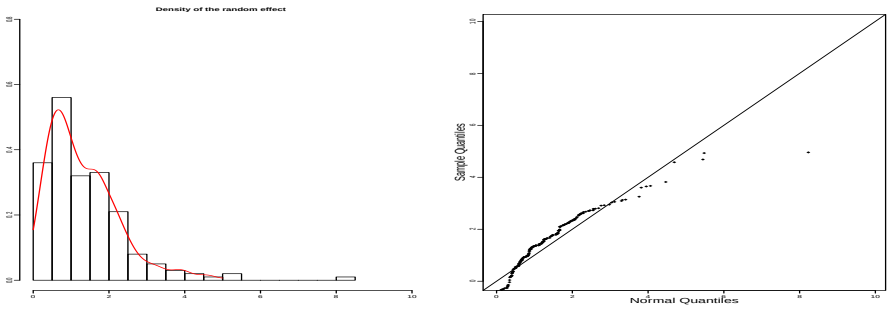

Figure 1 – Simulated example 1 (CIR with one Gamma random effect), nonparametric estimation. Left: histogram of estimated random effects (A_j) and nonparametric estimation of f. Right: qqplot of (A_j) versus a Normal sample (true distribution is Gamma)

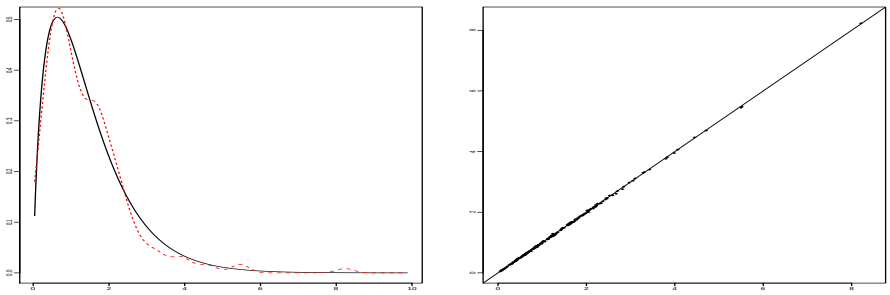

Figure 2 – Simulated example 1 (CIR with one Gamma random effect), nonparametric estimation, comparison to the truth. Left: estimation $\widehat{f}$ (dotted line) and true density f (plain line). Right: Estimated random effects A_j versus true random effects ϕ_j

As expected kurtosis is larger than 3 and skewness is positive which means that the distribution is right-tail. Figure 1 is provided by

```
plot(estim_nonparam)
```

Nonparametric estimation fits well the histogram of (A_j) (left plot) and we see that the random effects are non-Gaussian (right plot). Because we are working on simulated data, we can compare the estimations with the true random effects and the true f:

```
# comparison of the true f and its estimation
gridf1 <- outputsNP$gridf
f1 <- dgamma(gridf1, param1[1], param1[2])            # true density function
fhat <- outputsNP$estimf    # nonparametric estimated density function
plot(gridf1, f1, type='l', lwd=2, xlab='', ylab='')   # plot of the true density
lines(gridf1, fhat, col='red')   # plot of the nonparametric estimated density
# comparison of the true random effects and their estimations
phihat1 <- outputsNP$estimphi                         # estimated random effects
plot(phi1, phihat1, type = "p", pch = 18, xlab='', ylab='')  # true vs estimated
abline(0, 1)
```

It yields to Figure 2. On the left plot, the estimated density (dotted curve) is very close to the true density f (plain line). The right plot shows that A_j is a good estimation of ϕ_j. This confirms that the nonparametric approach performs well for this settings. Validation of the MSDE is produced by function `valid`. The two graphs on the right of Figure 5 are obtained by

```
choice1 <- floor(runif(1,1,M))
validationCIR <- valid(estim_nonparam, numj=choice1)
```

Prediction are obtained with `pred` and similar Figure 6 (not shown) can be obtained with

```
predNPCIR <- pred(estim_nonparam)
```

4.2 Frequentist parametric estimation

We present the parametric estimation on Example 2. The code is

```
#- parametric estimation
estim.method<-'paramML';
estim_param <- mixedsde.fit(times2, X2 = X2, model2 = model2, random2 = random2,
                    estim.fix = 1, estim.method = 'paramML' )
outputsP <- out(estim_param)      # stores the results in a list
```

Summary function provides:

```
> summary(estim_param)
     [,1]  [,2]
[1,] "BIC" "-3719.87172444398"
[2,] "AIC" "-3735.51554873924"
     [,1]       [,2]
[1,] "kurtosis" "2.64264939857654"
[2,] "skewness" "0.206900579156786"
     [,1]           [,2]
[1,] "sigma"        "0.107968951754119"
[2,] "estim.fixed"  "4.82151311689343"
[3,] "empiric mean" "2.86790431456589"
[4,] "MLE mean"     "2.86796970882836"
[5,] "empiric sd"   "0.568834742920312"
[6,] "MLE sd"       "0.552572219170386"
```

Kurtosis is, as expected, close to 3 and skewness close to 0. The diffusion parameter σ is well estimated (true value 0.1). The fixed effect is also well estimated (true value 5). Empirical mean and standard deviations are very close to MLE (estimator of the mean is the same in that case) and close to the real ones (3, 0.5). Then, Figure 3 (left and right) is provided by

```
plot(estim_param)
valid(estim_param)
```

The small number of observations makes the frequentist estimation harder, nevertheless here, the histogram seems pretty well fitted by the parametrically estimated density. Because we are working on simulated data, we can compare the estimations with the true random effects and the true f:

```
# comparison of the true f and its estimation
gridf2 <- outputsP$gridf
f2 <- dnorm(gridf, param[1], param[2])                  # true density
fhat_param <- outputsP$estimf                           # parametric estimated density
plot(gridf2, f2, type = 'l', lwd = 2, xlab = '', ylab = '') # plot of the true density
lines(gridf2, fhat_param, col='red', lty = 2, lwd = 2)  # plot of the estimated density
# comparison of the true random effects and their estimations
phihat2 <- outputsP$estimphi                            # estimated random effects
plot(phi2, phihat2, type="p", pch=18, xlab='', ylab='') # true vs estimated
abline(0, 1)
```

It yields to Figure 4. It shows that estimation of the density is satisfactory (left) and estimation of the random effects is very good (right).

Validation of the MSDE is produced by function `valid`. For example the individual validation of the first trajectory is plotted Figure 5, the first two graphs on the left, using

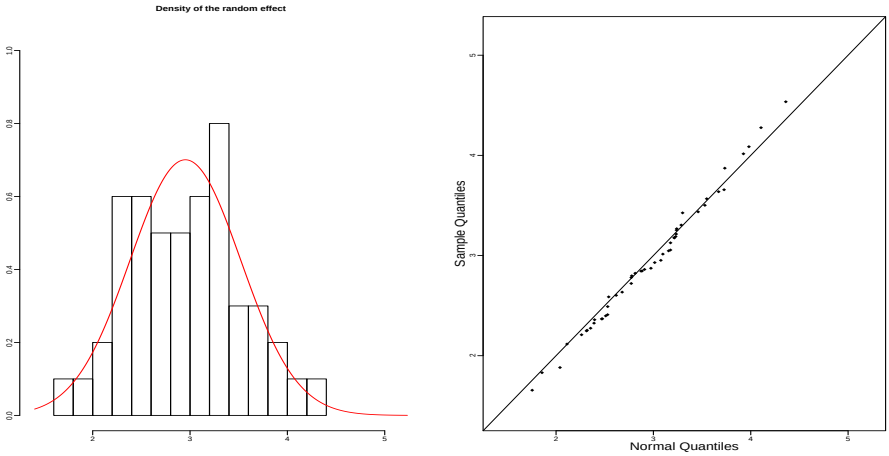

Figure 3 – Simulated example 2 (OU with one Gaussian random effect) frequentist parametric estimation. Left: histogram of the (A_j) and Gaussian parametric estimation of f. Right parametric qqplot of (A_j) versus a Normal sample

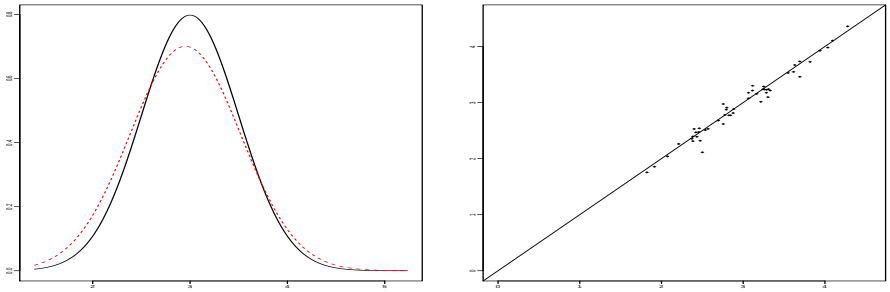

Figure 4 – Simulated example 2 (OU with one Gaussian random effect) frequentist parametric estimation, comparison to the truth. Left: parametric estimation $\mathcal{N}(\widehat{\mu}, \widehat{\omega}^2)$ (dotted line) and true f (plain line). Right: true ϕ_j versus estimated random effects A_j

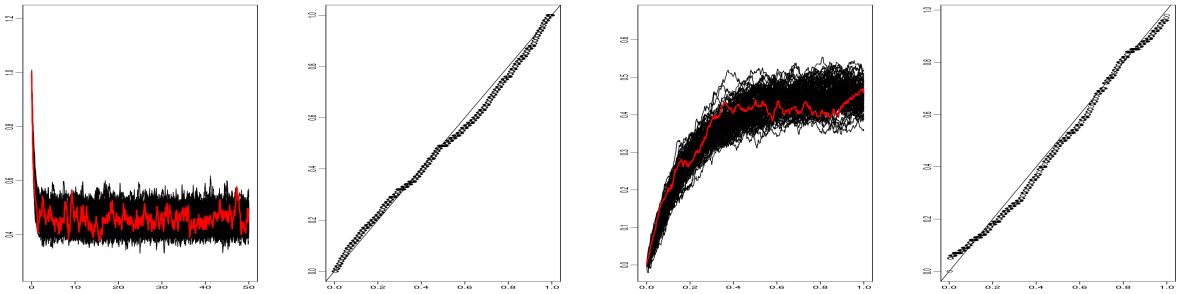

Figure 5 – Simulated examples frequentist approaches, outputs of `valid` method. Two left plots: frequentist nonparametric estimation on example 1 (CIR process). Two right plots: frequentist parametric estimation on example 2 (OU process)

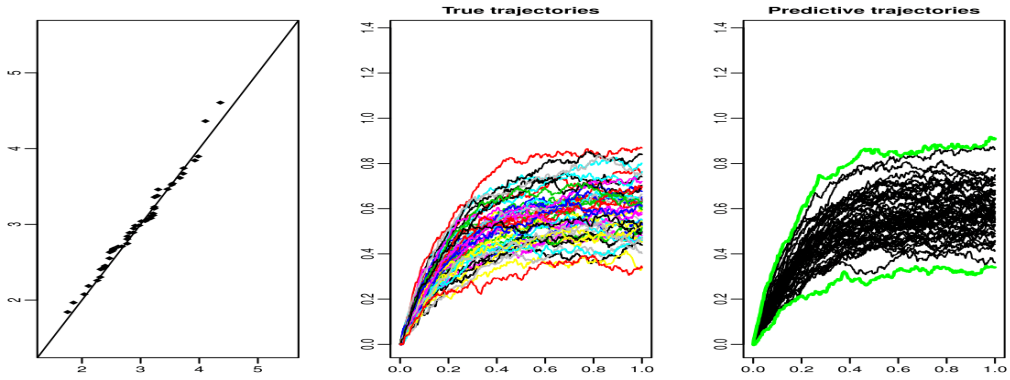

Figure 6 – Simulated example 2 (OU with one Gaussian random effect), frequentist parametric estimation. Left: predicted random effects versus estimated random effects. Middle: true trajectories. Right: predicted trajectories in black and 95% prediction interval in grey (green)

```
choice2 <- floor(runif(1,1,M))
validationOU <- valid(estim_param, numj =choice2)
```

This illustrates the good estimation of the random effects: a beam of trajectories with the true one in the middle and the lining up of the quantiles.

Finally, we can predict some trajectories using `pred`. Predictions are shown on Figure 6, as a result of

```
predPOU <- pred(estim_param)
```

Beam of predicted trajectories (right) is close to the true one (middle). The lining up of the predicted random effects versus the estimated random effects (left) shows the goodness of the prediction from the estimated density, thus of the estimation of the density.

4.3 Bayesian estimation

Bayesian method is applied to Example 2. Priors are constructed from the true values, but default values can be used.

```
prior2 <- list( m = c(param2[1],  fixed2), v = c(param2[1], fixed2), alpha.omega = 11,
              beta.omega = param2[2]^2*10, alpha.sigma=10, beta.sigma = sigma^2*9)
estim.method <- 'paramBayes'
estim_bayes <- mixedsde.fit(times2, X2, model = 'OU', random = 2, estim.method,
             prior = prior2, nMCMC = 10000)
```

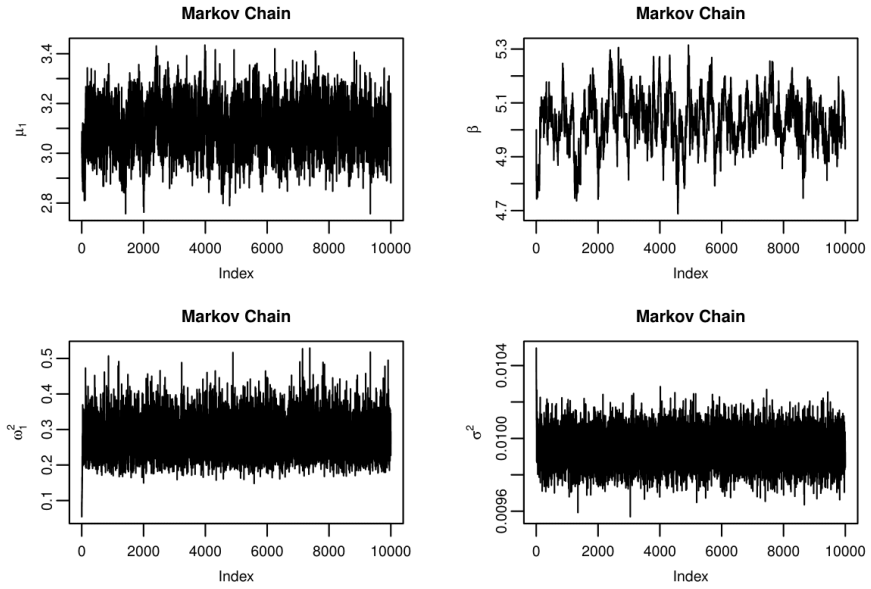

Figure 7 – Simulated example 2 (OU with one Gaussian random effect) Bayesian estimation. Markov chains of μ_1, β, ω_1^2 and σ^2

```
outputsBayes <- out(estim_bayes)
```

Figure 7 is produced by

```
plot(estim_bayes)
```

Traces of the Markov chains of μ_1, β, ω_1^2 and σ are plotted, showing that all chains converge and have the correct location. Command `print()` yields acceptance rates of the MH algorithm:

```
print(estim_bayes)
[1] "acceptance rates for random effect:"
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.5607  0.5675  0.5694  0.5705  0.5730  0.5828
[1] "acceptance rate for fixed effect:" "0.3591"
```

The fixed effect β has a small acceptance rate, explaining the dependent chain (Figure 7 top right). This is due to a very sharp likelihood because of the large amount of observations ($N \cdot M$) in comparison to the random effect (N).

Predictions in the Bayesian framework and the corresponding Figure 8 is obtained by

```
pred.result <- pred(estim_bayes)
```

Figure 8 shows the beam of simulated data trajectories together with the 95% prediction interval. Coverage rates are shown on the right plot and we see that the intervals hold the level.

5 Package `mixedsde` through a real data example

A real dataset is available (`neuronal.data.rda`) through lists of a matrix `X` and a vector `times`. We detail below the analysis of this dataset, following the next steps: run the two random effects model with both the parametric and nonparametric procedure; choose the number of random effects depending on the variability of the estimators $(A_{j,1}, A_{j,2})$, on the shape of $\widehat{f}_h$ and the variance $\widehat{\Omega}$.

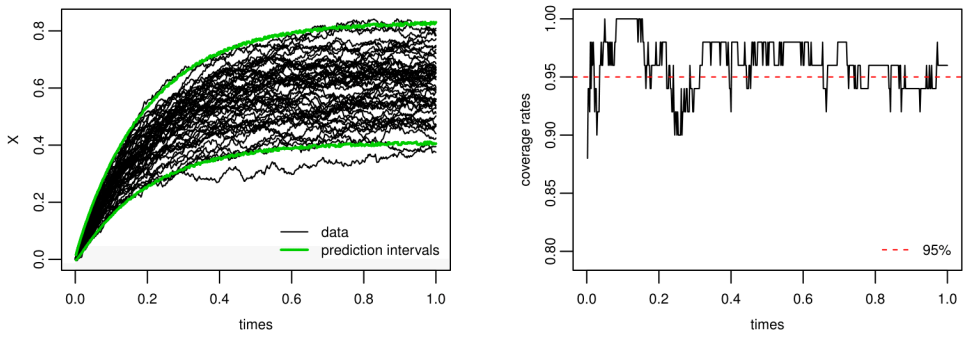

Figure 8 – Simulated example 2 (OU with one Gaussian random effect) Bayesian estimation. Left: predicted trajectories in black and 95% prediction interval in grey (green). Right: coverage rates: amount of observed values covered by the prediction intervals

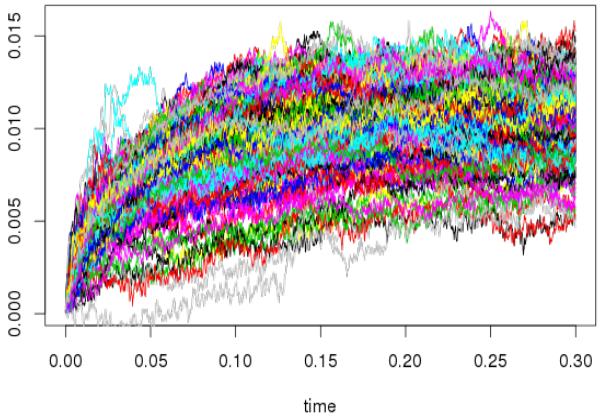

Figure 9 – Neuronal data

These data are available thanks to Rune Berg and Jufang He. Details on data acquisition can be found in Lansky *et al.* (2006).

5.1 Neuronal data (`neuronal.data`)

Neurons are the basement of nervous system and each neuron is connected with around 1000 other neurons. They are communicating through emission of electrical signal. We focus on the dynamic of the neuron membrane potential between two spikes emission measured in volts as the difference of ions concentration between the exterior and the interior of the cell. Data are obtained from one single neuron of a pig. Data are composed of $M = 240$ membrane potential trajectories with $N = 2000$ equidistant observation times. Time step is $\delta = 0.00015$ [s] and observation time is $T = 0.3$ [s]. Data are uploaded using `data(neuronal.data)`. They are presented on Figure 9.

These data have been previously analysed with a Ornstein-Uhlenbeck model with one additive random effect (α_j): Picchini *et al.* (2008) and Picchini *et al.* (2010) use parametric methods assuming the normality of the random effect, and Dion (2014) with a nonparametric method. Here α_j represents the local average input that the neuron receives after the j^{th} spike. The initial

voltage (the value following a spike) is assumed to be equal to the resting potential and set to zero: $x_j = 0$. Parameter β_j (non negative) is the time constant of the neuron. It was fixed in Picchini *et al.* (2008) and Picchini *et al.* (2010).

In this new analysis, we assume that both α_j and β_j may change from one trajectory to another because of other neurons or environment influence, for example. Indeed, the form of each trajectory lead us to think that this is the good model: each one has its mean-revert value and its own speed to reach this value. There is no reason to assume that the speed is always the same, but looking at the trajectories the stationary state seem reached nearly at the same time, thus the second random effect should have a small variance.

5.2 Fitting with MSDEs

Our goal is also to compare the two models OU and CIR, both with two random effects, and two approaches: the nonparametric density estimation and the parametric density estimation. Let us remark that for the CIR model the algorithm removes two trajectories: 168 and 224, because they contain negatives values. For two random effects the command is

```
estim <- mixedsde.fit(times, X=X, model= model, random = random,
         estim.method = estim.method)
```

and they can be found in the help data file (command `?neuronal.data`). We first apply the two frequentist approaches on models with two random effects. Kurtosis and skewness of the distribution of the estimation A_j of the random effects given in Table 2 are not closed to a symmetric distribution. The bidimensional density of (α_j, β_j) is estimated for both models with the parametric and nonparametric methods running function `mixedsde.fit`. Figure 10 gives the 4 estimated marginals. The blue (black) is for the OU model and the green (grey) for the CIR model. The dotted lines are the estimations from the parametric method, the plain lines for the nonparametric estimation. Parametric and nonparametric estimators are close, except for the second random effect with the OU model. Indeed, parametric estimation produces a very small variance for the second random effect, suggesting it could be fixed. Would this assumption be valid, it explains the difference with the nonparametric estimator which is not stable if the variance is to small. Estimation of σ is $\widehat{\sigma} = 0.0136$ for the OU model and $\widehat{\sigma} = 0.163$ for the CIR model.

To compare with previous literature results, we focus on the OU model. To select the number and the position of the random effect, we run the code with one random effect, additive or multiplicative: `random=1` or `random = 2`, for both models estimating the common fixed parameter. Estimators of the means μ_1, μ_2 and standard deviations ω_1, ω_2 are given in Table 3 and compared to values obtained in Picchini *et al.* (2008) and Picchini *et al.* (2010). Criteria AIC and BIC are also given in Table 3. The preferred model is the one minimizing both criteria. Thus, the OU model with one additive random effect $\phi_j = \alpha_j$ and $\widehat{\beta} = 37$ seems to be the best model to describe these data. The summary method gives for the kurtosis: 4.55 and for the skewness -0.95. Also $\widehat{\sigma} = 0.0136$. Estimated densities obtained for this model with $\widehat{\beta} = 37$ are given in Figure 11. The dotted line is the result of the parametric estimation and the plain line of the nonparametric estimation, plotted on the histogram of the $A_j(T)$'s. The nonparametric estimation detects a left tail that is not detected by the parametric one. Otherwise both estimators are very close.

The OU model with `random`= 1 is then validated with `valid` function. Figure 12 illustrates the result for a random trajectory (number 232): 100 simulated trajectories (black) and true trajectory (X_{232}, red) (left plot) and quantiles of the true measurement among the 100 simulated points at each time points versus uniform quantiles. The qq-plot is satisfactory (compared to the graph obtained on simulated data Figure 5). Finally some prediction plots are performed (not shown) with the `pred` method and they confirm that model OU with `random = c(1,2)` with the parameters obtain from the parametric estimation, and the OU model with `random = 1` and $\widehat{\beta} = 37$ produce very close trajectories and could be both validated.

We then apply the Bayesian procedure. As already mentioned, for the Bayesian procedure, large data sets are a problem because of the very long running time. Therefore, we thin the data

Table 2 – Neuronal data. Kurtosis and skewness estimations for samples $(A_{j,1})$'s and $(A_{j,2})$'s, for OU and CIR models

		OU	CIR
$(A_{j,1})$	Kurtosis	6.16	11.69
	Skewness	0.95	2.32
$(A_{j,2})$	Kurtosis	6.67	7.07
	Skewness	1.63	1.86

Table 3 – Neuronal data. MLE and BIC AIC criteria for OU model, depending on the number of random effects

	μ_1	ω_1	μ_2	ω_2	BIC	AIC
Picchini *et al.* (2008)						
random=1	0.28	0.04	25.64	–	-2969.70	-2978.71
Picchini *et al.* (2010)						
random=1	0.49	0.07	47.00	–	-3043.89	-3052.85
random=1	0.37	0.06	37.00	–	**-3240.45**	**-3249.41**
random=2	0.38	–	38.82	7.76	-3093.12	-3102.08
random=c(1,2)	0.38	0.06	37.70	1.10	-3229.67	-3247.5

set by 10. That means, every 10th data point of the series is used for the estimation and also for the prediction. Even with this thinning, one estimation with 20000 samples takes half an hour.

Based on the best model selected by the frequentist approach, the OU model with one random effect $\phi_j = \alpha_j$ is fitted. No prior knowledge is available, we therefore leave this information out and let the algorithm take the first 10%, i.e. 24, series for the calculation of the prior parameter, as described in Section 3.2. Figure 13 plots the Markov chains estimated from the remaining $M - 24 = 216$ trajectories and show good convergence of the chains. Bayesian point estimations, i.e. posterior means, are $\widehat{\mu}_1 = 0.34$, $\widehat{\omega}_1 = \sqrt{\widehat{\omega}_1^2} = 0.06$, $\widehat{\beta} = 33$ and $\widehat{\sigma} = \sqrt{\widehat{\sigma}^2} = 0.01$. Compared to frequentist estimation (Table 3), we notice that these results are a compromise between Picchini *et al.* (2010) and frequentist estimation.

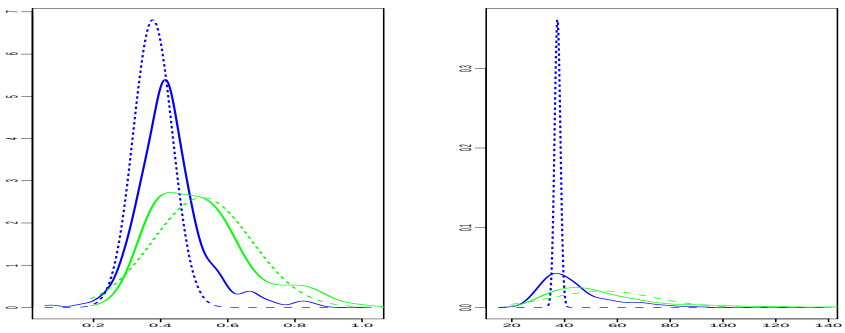

Figure 10 – Neuronal data. Frequentist estimated marginals of the bidimensionnal density of the random effects obtained from 4 estimators. Left: α_j's density, right: β_j's density. CIR model in green (grey), OU in blue (black). Nonparametric in plain line, parametric in dotted line

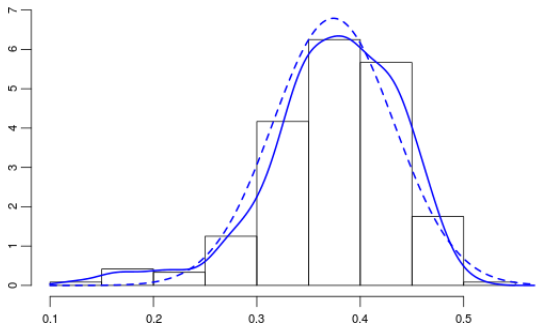

Figure 11 – Neuronal data, OU model, α random, β fixed, frequentist approaches. Estimation of the density f: parametric estimation in blue (black) plain line, non-parametric estimation blue (black) dotted line and histogram of the A_j's

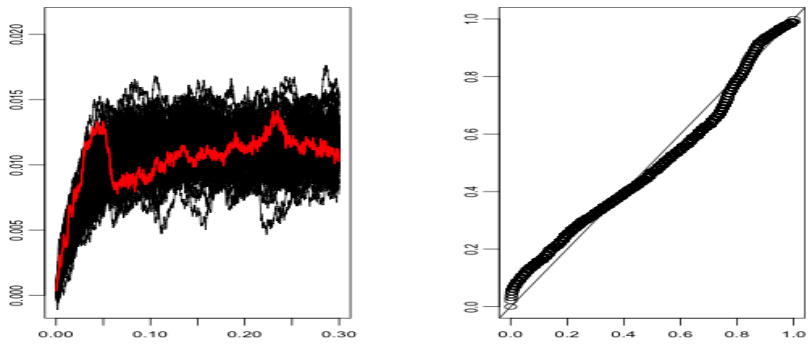

Figure 12 – Neuronal data, OU model, α random, β fixed, validation of the frequentist approaches. Individual validation of trajectory 232. Left: 100 simulated trajectories in black and true trajectory (X_j) in grey (red). Right: quantiles of the true measurement among the 100 simulated points at each time points versus uniform quantiles

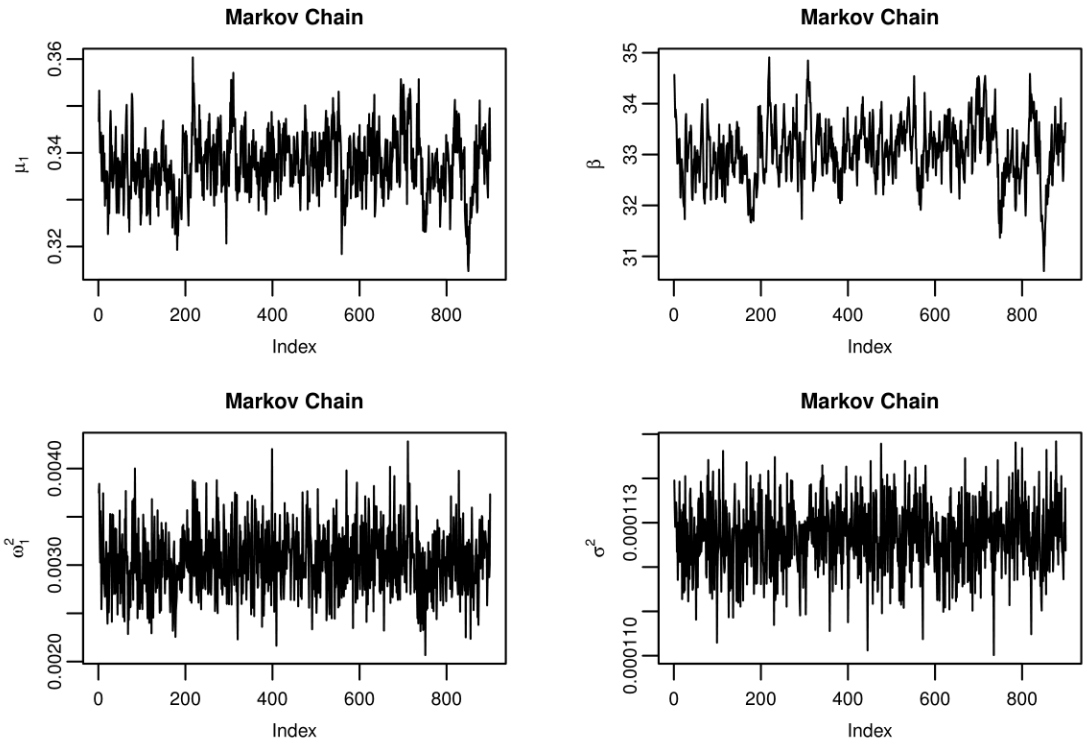

Figure 13 – Neuronal data, OU model, α random, β fixed, Bayesian estimation. Reduced Markov chains (less the burn-in phase and the thinning rate)

6 Discussion

In this paper we illustrate the functionality of the package `mixedsde` for inference of stochastic differential equations with random and/or fixed effects. This package, and mainly the function `misedsde.fit`, can be used to choose the best model to fit some data. It allows to compare two models: OU or CIR with one or two random effects. The three estimation methods can be used to help the decision maker. Nevertheless each method can be more appropriate to a specific situation, as explained before: the Bayesian method is recommended for a small number of observations, the frequentist nonparametric is a good tool with two random effects and no prior available. In particular the frequentist parametric proposes for a large sample, an estimation of the fixed effect and of the parameters of the Gaussian distribution for the fixed effect when there is only one. A neuronal dataset is studied with the three methods. To enrich this package and this study, one could implement the parameter estimation method developed in Delattre *et al.* (2016) for random effects distributed according to a Gaussian mixture distribution. Furthermore, other real data should be investigated with the present package.

Appendix

When there is one random effect, what is the likelihood function and the MLE of the fixed effect?

Assume that we are in the case of `random = 1`, thus the process is

$$dX_j(t) = (\alpha - \phi_j X_j(t))dt + \sigma a(X_j(t))dW_j(t)$$

Let us compute the log-likelihood function when $\phi_j = \varphi$ fixed. We omit the subscript j in the following. We use the same notation as for `random=c(1,2)`, where V is a symmetric matrix size 2×2. We have :

$$\begin{aligned}\log L(X,\alpha,\varphi) &= \int_0^T \frac{b(X(s),\varphi)}{\sigma^2(X(s))}dX(s) - \frac{1}{2}\int_0^T \frac{b^2(X(s),\varphi)}{\sigma^2(X(s))}ds \\ &= \alpha U_1 - \frac{\alpha^2}{2}V_{1,1} + \varphi[U_2 - \alpha V_{1,2}] - \frac{\varphi^2}{2}V_{2,2}.\end{aligned}$$

We assume that the random effect is Gaussian with density f_ξ, and denote $\xi = (\mu,\omega)$ and $\theta := (\mu,\omega,\alpha)$. Thus,

$$L(X,\theta) = \int \exp\left(\alpha U_1 - \frac{\alpha^2}{2}V_{1,1} + \varphi[U_2 - \alpha V_{1,2}] - \frac{\varphi^2}{2}V_{2,2}\right) f_\xi(\varphi)d\varphi = \int \exp(\mathcal{E}(\varphi))d\varphi.$$

We find:

$$\begin{aligned}\mathcal{E}(\varphi) &= \alpha U_1 - \frac{\alpha^2}{2}V_{1,1} - \frac{1}{2}\left[\varphi^2(V_{2,2}+\omega^{-2}) - 2\varphi(U_2 - \alpha V_{1,2} + \mu\omega^{-2})\right] \\ &= \alpha U_1 - \frac{\alpha^2}{2}V_{1,1} - \frac{1}{2\Sigma^2}(\varphi - m)^2 + \frac{m^2}{2\Sigma^2} - \frac{1}{2}\mu^2\omega^{-2}\end{aligned}$$

with

$$m = \frac{\mu + \omega^2 U_2 - \omega^2 V_{1,2}\alpha}{1+\omega^2 V_{2,2}}, \quad \Sigma^2 = \frac{\omega^2}{1+\omega^2 V_{2,2}}. \tag{8}$$

Finally after simplification we get:

$$\frac{m^2}{2\Sigma^2} - \frac{1}{2}\mu^2\omega^{-2} = -\frac{1}{2}(1+\omega^2 V_{2,2})^{-1}V_{2,2}[\mu - V_{2,2}^{-1}(U_2 - \alpha V_{1,2})]^2 + \frac{1}{2V_{2,2}}(U_2 - \alpha V_{1,2})^2.$$

Thus for `random=1` we get

$$L(X,\theta) = \frac{1}{\sqrt{1+\omega^2 V_{2,2}}} \exp\left[\alpha U_1 - \frac{\alpha^2}{2} V_{1,1} - \frac{V_{2,2}}{2(1+\omega^2 V_{2,2})}[\mu - V_{2,2}^{-1}(U_2 - \alpha V_{1,2})]^2 + \frac{(U_2 - \alpha V_{1,2})^2}{2V_{2,2}}\right]. \tag{9}$$

Then, when `random = 2` the roles of α and φ are exchanged. To implement a general formula, we note: r for random: 1 or 2, and c for the number of the common effect. We denote ψ the fixed effect and we the get the general formula:

$$L(X,\theta) = \frac{1}{\sqrt{1+\omega^2 V_{r,r}}} \exp\left[\psi U_c - \frac{\psi^2}{2} V_{c,c} - \frac{V_{r,r}}{2(1+\omega^2 V_{r,r})}[\mu - V_{r,r}^{-1}(U_r - \psi V_{c,r})]^2 + \frac{(U_r - \psi V_{r,c})^2}{2V_{r,r}}\right]. \tag{10}$$

Acknowledgements

The authors would like to thank Vincent Brault and Laurent Bergé for technical help on the package. This work has been partially supported by the LabExPERSYVAL-Lab(ANR-11-LABX-0025-01).

The second author, Simone Hermann, was financially supported by Project B5 “Statistical methods for damage processes under cyclic load” of the Collaborative Research Center “Statistical modeling of nonlinear dynamic processes” (SFB 823) of the German Research Foundation (DFG).

References

Bacher, P. and Madsen, H. (2011). Identifying suitable models for the heat dynamics of buildings. *Energy and Buildings* **43**, 1511–1522.

Berg, R. and Alaburda, W. and Hounsgaard, J. (2007) Balanced inhibition and excitation drive spike activity in spinal half centers. *Science* **315**, 390–393.

Comte, F., Genon-Catalot, V. and Samson, A. (2013). Nonparametric estimation for stochastic differential equation with random effects. *Stochastic Processes and their Applications* **7**, 2522–2551.

Delattre, M., Genon-Catalot, V. and Samson, A. (2013). Maximum likelihood estimation for stochastic differential equations with random effects. *Scandinavian Journal of Statistics* **40**, 322–343.

Delattre, M., Genon-Catalot, V. and Samson, A. (2016). Mixture of stochastic differential equations with random effects: application to data clustering. *to appear in Journal of Statistical Planning and Inference.*

Dion, C. and Genon-Catalot, V. (2015). Bidimensional random effect estimation in mixed stochastic differential model. *Statistical Inference for Stochastic Processes* **18**(3) 1–28

Dion, C. (2014). Nonparametric estimation in a mixed-effect Ornstein-Uhlenbeck model. *to appear in Metrika* .

Ditlevsen, S. and de Gaetano, A. (2005). Mixed Effects in Stochastic Differential Equation. *REVSTAT - Statistical Journal* **3**, 137–153.

Ditlevsen, S. and Samson, A. (2014). Estimation in partially observed stochastic Morris-Lecar neuronal model with particle filter and stochastic approximation methods. *The annals of applied statistics* **8**, 674–702.

Genon-Catalot, V. and Larédo, C. (2016). Estimation for stochastic differential equations with mixed effects. *Statistics* `http://dx.doi.org/10.1080/02331888.2016.1141910`.

Hansen, A. and Duun-Henriksen, A.K, Juhl, R., Schmidt, S., Norgaard, K., Jorgensen, J.B., and Madsen, H. (2014). Predicting plasma glucose from interstitial glucose observations using bayesian methods. *Journal of diabetes science and technology* **8**, 321–330.

Hermann, S., Ickstadt, K. and C. Müller (2016). Bayesian Prediction of Crack Growth Based on a Hierarchical Diffusion Model. *Appearing in: Applied Stochastic Models in Business and Industry.*

Iacus, S. M. (2008). *Simulation and Inference for Stochastic Differential Equations.* Springer-Verlag, New York.

Iversen, E.B., Morales, J.M., Moller, J.K., Madsen, H. (2014) Probabilistic forecasts of solar irradiance using stochastic differential equations *Environmetrics* **25**, 152–164.

Jahn, P. and Berg, R. and Hounsgaard, J. and Ditlevsen, S. (2011) Motoneuron membrane potentials follow a time inhomogeneous jump diffusion process *Journal of Computational Neuroscience* **31**, 563–579.

Kuelbs, J. and Zinn, J. (2015). Limit Theorems for Quantile and Depth Regions for Stochastic Processes *High dimensional probability VII-Progress in Probabiliy, to appear.*

Lansky, P., Sanda, P. and He, J. (2006). The parameters of the stochastic leaky integrate-and-fire neuronal model. *Journal of Computational Neuroscience* **21**, 211–223.

Oravecz, Z., Tuerlinckx, F. and Vandekerckhove, J. (2009). A Hierarchical Ornstein-Uhlenbeck Model for Continuous Repeated Measurement Data. *Psychometrica* **74**, 395–418.

Oravecz, Z. and Tuerlinckx, F. (2011). The Linear Mixed Model and the Hierarchical Ornstein-Uhlenbeck Model: Some Equivalences and Differences. *British Journal of Mathematical and Statistical Psychology* **64**, 134–160.

Picchini, U. and Ditlevsen, S. (2011). Practicle estimation of high dimensional stochastic differential mixed-effects models. *Computational Statistics & Data Analysis* **55**, 1426–1444.

Picchini, U., Ditlevsen, S., De Gaetano, A. and Lansky, P. (2008). Parameters of the diffusion leaky integrate-and-fire neuronal model for a slowly fluctuating signal. *Neural Computation* **20**, 2696–2714.

Picchini, U., De Gaetano, A. and Ditlevsen, S. (2010). Stochastic differential mixed-effects models. *Scandinavian Journal of statistics* **37**, 67–90.

Pinheiro, J. and Bates, D. (2000). *Mixed-effect models in S and Splus*. Springer-Verlag, New York.

Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer-Verlag, New York.

Rosenthal, J. S. (2011). *Optimal proposal distributions and adaptive MCMC*. Handbook of Markov Chain Monte Carlo, 93–112.

Virkler, D. A., Hillberry, B. M. and Goel, P. K. (1979). The Statistical Nature of Fatigue Crack Propagation. *Journal of Engineering Materials and Technology* **101**, 148–153.

Zuo, Y. and Serfling, R. (2000). General notions of statistical depth function *Annals of statistics* **28**, 461–482.