SdeEstimation Toolbox: Instructions for the users

February 21, 2023

Abstract

SdeEstimation toolbox is a set of procedures designed for the state and parameters estimation of stochastic differential equation from discrete observations. The toolbox provides various implementations of Local Linearization filters for the state estimation and, consequently, of the Innovation Estimators for the parameters. The users can choose the implementation that better fits their needs in correspondence with the particular model and data under consideration, and of the required accuracy for the estimation. This includes deterministic and stochastic filters with fixed step sizes and number of samples, with adaptive time stepping algorithms, with adaptive sampling algorithms, as well as local and global optimization algorithms for computing the innovation estimators. Parallel computations are automatically set when more than one core are available, plus Krylov-subspace approximations for computing the predictions when the model has more than six equations. Confidence intervals are provided for the estimated parameters as well as the AIC and BIC values of the fitted model. For the estimated parameters, the filtering algorithms return the predictions and filters of the states, the innovation process, the values of the Kolmogorov-Smirnov, Jarque-Bera, Ljung-Box, and Engle test for the innovation process, and the log likelihood value. The provided measures of goodness of fit of the model to the data make the toolbox also useful for the model selection, for designing of models and for optimal experimental design. The toolbox is intended for the recurrent practical situation in which a diffusion process should be identified from a reduced number of possible noisy and partial observations of the state variables, distant in time and with missing data. A number of illustrative test models and demos are included.

Contents

1 Introduction

SdeEstimation toolbox is a set of Matlab functions for the estimation of unobserved states and unknown parameters of stochastic differential equations given a time series of discrete observations. The stochastic differential equations have the form

$$d\mathbf{x}(t) = \mathbf{f}(t, \mathbf{x}(t); \theta)dt + \sum_{i=1}^{m} \mathbf{g}_i(t, \mathbf{x}(t); \theta)d\mathbf{w}^i(t), \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^d$ describes the time evolution of d state variables of the modeled phenomena for all time instant $t \geq t_0 \in \mathbb{R}$, $\mathbf{f}$ and $\mathbf{g}_i$ are differentiable functions, $\mathbf{w}= (\mathbf{w}^1, \ldots, \mathbf{w}^m)$ is an m-dimensional standard Wiener process, and $\theta \in \mathbb{R}^p$ is a vector of parameters whose values determine the transition density function for the state $\mathbf{x}(t_2)$ given the state $\mathbf{x}(t_1)$, for all $t_2 > t_1$. The available measurements $\mathbf{z}_{t_0}, \ldots, \mathbf{z}_{t_{M-1}}$ of the state variable $\mathbf{x}$ are taken on a sequence of M observation times $t_0, \ldots, t_{M-1}$ through physical devices that introduce random measurement errors. These measurements of the state variables are assumed to be described by a discrete observation equation of the form

$$\mathbf{z}_{t_k} = \mathbf{h}(t_k, \mathbf{x}(t_k)) + \sum_{i=1}^{n} \mathbf{p}_i(t_k, \mathbf{x}(t_k))\xi^i_{t_k} + \mathbf{e}_{t_k}, \text{ for } k = 0, 1, .., M-1, \tag{2}$$

where $\mathbf{z}_{t_k} \in \mathbb{R}^r$ is the observation vector at the instant of time t_k, $\{\xi_{t_k} : \xi_{t_k} \sim \mathcal{N}(0, \mathbf{\Lambda}_k),\ \mathbf{\Lambda}_k = diag(\lambda^1_k, \ldots, \lambda^n_k),\ k = 0, \ldots, M-1\}$ and $\{e_{t_k} : e_{t_k} \sim \mathcal{N}(0, \mathbf{\Pi}_k), k = 0, \ldots, M-1\}$ are sequences of i.i.d. random vectors independent of $\mathbf{w}$, and $\mathbf{h}$ and $\mathbf{p}_i$ are differentiable functions. Here, the observation times $(t)_M = \{t_k : k = 0, 1, \ldots, M-1\}$ are assumed to be increasing, i.e., $t_{k-1} < t_k$ for all $k = 1, \ldots, M-1$.

In the case that $\mathbf{h}$ and $\mathbf{p}_i$ are twice continuously differentiable functions, it is recommended to transform the observation equation (2) to the simpler one

$$\mathbf{z}_{t_k} = \mathbf{C}\mathbf{x}(t_k) + \sum_{i=1}^{n} \mathbf{D}_i\mathbf{x}\xi^i_{t_k} + \mathbf{e}_{t_k}, \text{ for } k = 0, 1, .., M-1, \tag{3}$$

in the way indicated in [8], where $\mathbf{C}$ and $\mathbf{D}_i$ are constant matrices.

The matrix $\mathbf{\Pi}_k$ is assumed to be positive definite in (3) (or (2)), for all $k = 1, \ldots, M-1$. Otherwise, the variance of the diffusion process $\mathbf{x}$ must be positive definite for all $t \geq t_0$ and $\theta \in D_\theta$, where D_θ is an compact set in $\mathbb{R}^p$.

2 Mathematical and Statistical Requirements

The user should have basic notions on stochastic differential equations, time series analysis, filtering methods, optimization and numerical integration.

3 Computer and Software Requirements

SdeEstimation toolbox requires of a multi-core computer with a Matlab 2018 (or higher) license for Windows, Linux or Mac, and license for the Optimization Toolbox, the Statistics and Machine Learning Toolbox, Econometrics, and the Parallel Computing Toolbox. The global optimization algorithms of the Matlab Global Optimization Toolbox can also be used.

SdeEstimation toolbox uses the pre-defined Matlab 'local' cluster. When validating this cluster in the Matlab Cluster Profile Manager, the number of workers in SPMD job test, Pool job test and Parallel pool test must be the same.

4 Implementation Knowhow

The main algorithms and functionalities implemented in the **SdeEstimation** toolbox were taken from the papers [8, 9, 12]. This section presents a summary.

Let $\{t\}_M$ be the sequence of M observation times $t_0, \ldots, t_{M-1}$ of the states variables of (1), and $Z_\rho = \{\mathbf{z}_{t_k} : t_k \leq \rho,\ t_k \in \{t\}_M\}$ the time series of partial and noisy measurements of $\mathbf{x}$ described by the observation equation (3) (or (2)) until the time instant ρ. Further, let $\mathbf{x}_{t/\rho} = E(\mathbf{x}(t)|Z_\rho)$ and $\mathbf{U}_{t/\rho} = E(\mathbf{x}(t)\mathbf{x}^\intercal(t)|Z_\rho) - \mathbf{x}_{t/\rho}\mathbf{x}_{t/\rho}^\intercal$ be the conditional mean and variance of $\mathbf{x}$ with $\rho \leq t$, where $E(\cdot)$ denotes the expected value of random vectors.

4.1 Local Linearization (LL) Filters

Given M measurements $Z_{t_{M-1}}$ and the initial filter estimates $\mathbf{y}_{t_0/t_0} = \mathbf{x}_{t_0/t_0}$, $\mathbf{V}_{t_0/t_0} = \mathbf{U}_{t_0/t_0}$, the **approximate Linear Minimum Variance (LMV) filter** for the model (1)+(3) is iteratively defined at each observation time $t_k \in \{t\}_M$ by the prediction estimates

$$\mathbf{y}_{t_{k+1}/t_k} = E(\mathbf{y}(t_{k+1})|Z_{t_k}) \qquad \text{and} \qquad \mathbf{V}_{t_{k+1}/t_k} = E(\mathbf{y}(t_{k+1})\mathbf{y}^\intercal(t_{k+1})|Z_{t_k}) - \mathbf{y}_{t_{k+1}/t_k}\mathbf{y}_{t_{k+1}/t_k}^\intercal, \tag{4}$$

with initial conditions $\mathbf{y}_{t_k/t_k}$ and $\mathbf{V}_{t_k/t_k}$, and the filter estimates

$$\mathbf{y}_{t_{k+1}/t_{k+1}} = \mathbf{y}_{t_{k+1}/t_k} + \mathbf{K}_{t_{k+1}}(\mathbf{z}_{t_{k+1}} - \mathbf{C}\mathbf{y}_{t_{k+1}/t_k}) \quad \text{and} \quad \mathbf{V}_{t_{k+1}/t_{k+1}} = \mathbf{V}_{t_{k+1}/t_k} - \mathbf{K}_{t_{k+1}}\mathbf{C}\mathbf{V}_{t_{k+1}/t_k}, \tag{5}$$

with filter gain $\mathbf{K}_{t_{k+1}}$, for all $t_k, t_{k+1} \in \{t\}_M$, where $\mathbf{y}$ is an approximation to the solution $\mathbf{x}$ of (1) on the time discretization $(\tau)_h \supset t_M$ satisfying $\tau_{n+1} - \tau_n \leq h$, with $h > 0$.

Based on Local Linear approximations $\mathbf{y}$ to $\mathbf{x}$, the **SdeEstimation** toolbox provides four types of approximate LMV filters:

1. **Classical LL filter**: Definition 2.3 in [8]. Predictions (4) computed by the expressions (5.3) of [8] with $(\tau)_h \equiv t_M$,
2. **Deterministic LL filter with artificial points between observation times**: Predictions (4) computed by the expressions (5.3) of [8] with a priori given set $(\tau)_h$ of artificial points between the observation times $\{t_M\}$,

3. **Adaptive deterministic LL filter**: Deterministic algorithm of Section 5.5 in [8] with adaptive time stepping strategy to construct $(\tau)_h$,
4. **Stochastic LL filter with artificial points between observation times**: Stochastic algorithm of Section 5.4 in [8] with a priori given set $(\tau)_h$ of artificial points between the observation times $\{t_M\}$. The numbers of stochastic simulations of $\mathbf{y}$ for computing the predictions (4) can be a priory given or adaptively estimated as in Section 5.3 of [8],

and, optionally, when the exact predictions $\mathbf{x}_{t_k/t_{k-1}}$ and $\mathbf{U}_{t_k/t_{k-1}}$ of $\mathbf{x}$ are available, the

5. **Exact LMV filter**: Definition 2.1 in [8].

The filter gain $\mathbf{K}_{t_{k+1}}$ in (5) for the observation equation (3) is computed as indicated by Theorem 1 in [10]. This filter gain reduces to that given in [8] when there is not multiplicative noise in (3). For models with the nonlinear observation equation (2), the approximate filter estimates given in (5) are replaced by those of the Definition 3 in [11].

4.2 Innovation Estimators

Given M measurements $Z_{t_{M-1}}$ of the state space model (1)+(3) with $\theta = \theta_0$ on $\{t\}_M$, the **approximate innovation estimator** for the parameters θ_0 of (1) is defined by [11, 9]

$$\widehat{\vartheta}_M = \arg\{\min_{\theta \in \mathcal{D}_\theta} \widetilde{U}_M(\theta, Z_{t_{M-1}})\}, \tag{6}$$

where

$$\widetilde{U}_M(\theta, Z_{t_{M-1}}) = (M-1)\ln(2\pi) + \sum_{k=1}^{M-1} \ln(\det(\widetilde{\mathbf{\Sigma}}_{t_k})) + \widetilde{\nu}_{t_k}^{\intercal}(\widetilde{\mathbf{\Sigma}}_{t_k})^{-1}\widetilde{\nu}_{t_k},$$

being

$$\widetilde{\nu}_{t_k} = \mathbf{z}_{t_k} - \mathbf{C}\mathbf{y}_{t_k/t_{k-1}}$$

$$\widetilde{\mathbf{\Sigma}}_{t_k} = \mathbf{C}\mathbf{V}_{t_k/t_{k-1}}\mathbf{C}^{\intercal} + \sum_{i=1}^{n} \lambda_k^i \mathbf{D}_i(\mathbf{V}_{t_k/t_{k-1}} + \mathbf{y}_{t_k/t_{k-1}}\mathbf{y}_{t_k/t_{k-1}}^{\intercal})\mathbf{D}_i^{\intercal} + \mathbf{\Pi}_k,$$

and $\mathbf{y}_{t_k/t_{k-1}}$ and $\mathbf{V}_{t_k/t_{k-1}}$ the approximate predictions (4) resulting from the filtering algorithm (4)-(5).

For each one of the five approximate LL filters of the previous section, the **SdeEstimation** toolbox provides an approximate innovation estimator. The confidence limits for these innovation estimators are computed as indicated in [12]. Statistics for the approximate fitting-innovation process $\{\widehat{\nu}_{t_k} : \widehat{\nu}_{t_k} = \mathbf{z}_{t_k} - \mathbf{C}\mathbf{y}_{t_k/t_{k-1}}(\widehat{\vartheta}_M)\}_k$ are computed as in [12]. This includes the Kolmogorov-Smirnov test of normality N(0,1), the Jarque-Bera test of composite Gaussianity, the Ljung-Box Q-test for autocorrelations, the Eagle ARCH-test of heteroscedasticity, and the likelihood ratio test to determine the number of lags for the Ljung-Box and Eagle tests. The exit condition and the whole information about the end of the optimization process are also provided for the five innovation estimators.

The approximate innovation estimator (6) reduces to that given in [9] when there is not multiplicative noise in (3). For models with nonlinear observation equation (2), the approximate filter estimates given in (5) are replaced by those of the Definition 3 in [11] and, consequently, the approximations $\widetilde{\nu}_{t_k}$ and $\widetilde{\mathbf{\Sigma}}$ in (6) are replaced by those corresponding to the Definition 3 in [11]. In the case $\mathbf{\Pi}_k = \mathbf{\Lambda}_k = 0$, the approximate innovation estimator (6) reduces to an approximate quasi-maximum likelihood estimator [9].

For the exact LMV filter, the **SdeEstimation** toolbox provides the exact innovation estimator (Definition 2.1 in [9]), the confidence limits and the statistics for the fitting-innovation process as well.

5 Basic toolbox functions and scripts

5.1 Basic functions for model definition

SE_ModelName.m : specifies the state equation (1).
OE_ModelName.m : specifies the observation equation (2).
S_ModelName.m : specifies the exact predictions of the state $\mathbf{x}$ when are available

5.2 Basic functions for data generation

MakeFileNames.m : summarizes the file names of Section 5.1 with the model definition.
Euler.m : integrates the state equation (1) with the uniform step-size Euler-Maruyama scheme.
Observations.m : generates time series Z of noisy observations of $\mathbf{x}$ with the observation equation (2).
PlotModelRealization.m : plots the integration and the noisy observation of $\mathbf{x}$.

5.3 Basic functions for filtering

LLfilter.m : computes approximate linear filters of minimum variance for the model (1)-(2), given Z.
PlotFilterEstimates.m : plots the observations Z and their filter estimates.
PlotModelEstimates.m : plots the exact and approximate conditional moments of $\mathbf{x}$.

5.4 Basic functions for parameter estimation

EstimSet.m : sets the options for the parameter estimation.
InnovEstimator.m : computes approximate innovation estimators of the model (1)-(2), given Z.
DisplayParameterEstimates.m : displays the results of the parameter estimation.
PlotStatTestFitInn.m : plots statistics of the standarized fitting-innovation.

5.5 Basic scripts for data generation, filtering and parameter estimation

Const_ModelName.m : defines the whole model (1)-(2).
Gen_ModelName.m : generates state variables and observations of the model (1)-(2).
Test_ModelName.m : tests the approximate filter for the model (1)-(2).
Like_ModelName.m : performs the state and parameter estimation of the model (1)-(2).

6 Easy use of the SdeEstimation toolbox

6.1 Simulated or actual data

With simulated data, the identification of a particular model of the form (1)-(2), called 'ModelName', requires of two functions and four scripts. These are the functions SE_ModelName.m and OE_ModelName.m, and the scripts Const_ModelName.m, Gen_ModelName.m, Test_ModelName.m and Like_ModelName.m, all of them located in the folder 'ModelName'. This folder name is included in the script SdeEstimationPath.m that defines all the folders name of the **SdeEstimation** toolbox to be added to the Matlab path. The additional function S_ModelName.m is also included in the case that the exact predictions of the state variables $\mathbf{x}$ are available.

Roughly speaking, the functions SE_ModelName.m and OE_ModelName.m are used to define the SDE and the observation equation, respectively, whereas the script Const_ModelName.m contents the remaining information about the state-space model. The scripts Gen_ModelName.m, Test_ModelName.m and Like_ModelName.m are used, respectively, to simulate data of the model, and to compute filter estimates and innovation estimators from the data generated with the first script. See next subsections for a full explanation on the functionalities of these functions and scripts, as well as the illustrative implementations of the models in Section 7 that appear in their corresponding folder 'ModelName'.

In the case that actual data are available for the model estimation, the user must create the Matlab data file data_ModelName.mat with two variables: the matrix 'datos' (# state variable $\times$ # observation times) with the time series of observations $\mathbf{z}_{t_0}, \ldots, \mathbf{z}_{t_{M-1}}$, and the column vector 't' with the observation times $t_0, \ldots, t_{M-1}$. The model can then be estimated directly with the script Like_ModelName.m whenever the model has been defined with the functions SE_ModelName.m and OE_ModelName.m, and the script Const_ModelName.m. For this, see for instance the script of the demos available in the folder 'Demos'.

It is important to recall that the success in the parameter estimation with an M-estimator like (6) depends on several factors that must be carefully considered for each specific model and data set. These factors are: power of the model to describe the data, accuracy of the approximations to the predictions, effectiveness of the optimization algorithm, time distance between two consecutive observations and the number of them. See [12] for a discussion about how to deal with these factors.

6.2 Function SE_ModelName.m

The function SE_ModelName.m defines the state equation (1). It returns the values of the drift f, the m diffusions g_i, and their derivatives at $(t, x, theta)$. In dependence of the type of equation, the function SE_ModelName.m has the following six possible call definitions:

```
              [f,Jf] = SE_ModelName(t,x,theta), for autonomous ODEs
            [f,g,Jf] = SE_ModelName(t,x,theta), for autonomous SDEs with additive noise
         [f,g,Jf,Jg] = SE_ModelName(t,x,theta), for autonomous SDEs with multplicative noise
       [f,g,Jf,ft,gt] = SE_ModelName(t,x,theta), for non-autonomous SDEs with additive noise
    [f,g,Jf,Jg,ft,gt] = SE_ModelName(t,x,theta), for non-autonomous SDEs with multplicative noise
 [[f;1],[Jf ft; 0 0]] = SE_ModelName(t,x,theta), for non-autonomous ODEs
```

where Jf and Jg denote the Jacobian matrices, and ft and gt the derivatives with respect to t. x and f are d-dimensional column vectors, gt is a m-dimensional cell containing m column vectors of dimension d, and Jg is a m-dimensional cell containing m squared matrices of dimension d. theta is a row vector.

6.3 Function OE_ModelName.m

The function OE_ModelName.m defines the observation equation (2). It returns the values of function h, the n functions p_i, their Jacobian matrices, and the variance of the additive and multiplicative noises at $(t, x, theta)$. In dependence of the type of equation, the function OE_ModelName.m has the following two possible call definitions:

```
           [he,Jhe,See] = OE_ModelName(t,x,theta), for observations with additive noise
 [he,Jhe,See,pe,Jpe,Lee] = OE_ModelName(t,x,theta), for observations with additive and
                                                     multiplicative noise
```

where Jhe and Jpe denote the Jacobian matrices. he is a r-dimensional column vector, pe is a n-dimensional cell containing n column vectors of dimension r, and Jpe is a n-dimensional cell containing n square matrices of dimension r.

See and Lee denote, respectively the variance of the additive $\mathbf{e}_{t_k}$ and multiplicative $\xi^i_{t_k}$ noises at $(t, x, theta)$. See is a $r \times r$ symmetric matrix, and Lee a n-dimensional vector. x is a d-dimensional column vector, and theta is a row vector.

6.4 Function S_ModelName.m

The function S_ModelName.m specifies the exact predictions of the state $\mathbf{x}$ when are available. The function call definition is:

```
[m,P] = S_ModelName(t,t0,m0,P0,theta)
```

where m0 and P0 are the first two conditional moments of the state variable $\mathbf{x}$ at t0, and m and P are the exact first two conditional moments of $\mathbf{x}$ at t.

6.5 Script Const_ModelName.m

The script Const_ModelName.m defines the whole model (1)-(2) with the following Matlab constants to be set by the user:

```
  deltat_min : sampling period for integration
      deltat : sampling period for filtering
          t0 : initial time
      tTotal : sampling time
       Theta : model parameter values (row vector)
        ObsX : boolean indicating the observed state variables (column vector)
          X0 : initial state variable for integration (column vector)
         Xf0 : initial filter estimates (column vector)
         Pf0 : initial filter variance estimates (squared matrix)
     ModEq = 'SE_ModelName' : definition of the model equation
      ObsEq = 'OE_ModelName' : definition of the observation equation
   ExactSol = 'S_ModelName'    : definition of the exact predictions of the model
FileNames = MakeFileNames(ModEq,ObsEq,ExactSol);
      AbsTol : absolute tolerance for filtering (1 x 3)
      RelTol : relative tolerance for filtering (1 x 3)
```

6.6 Script Gen_ModelName.m

The purpose of the script Gen_ModelName.m is generate the time series $\{\mathbf{x}(t_0),\ldots,\mathbf{x}(t_{M-1})\}$ of the process $\mathbf{x}$ in (1) and the observations $\{\mathbf{z}_{t_0},\ldots,\mathbf{z}_{t_{M-1}}\}$ of the model (1)-(2) at the observation times $t_0,\ldots,t_{M-1}$ for a further testing of the filtering and/or estimation algorithms. These time series are saved in the Matlab data file named data_ModelName.mat with the variable name 'process' and 'data', respectively. The observation times $t_0,\ldots,t_{M-1}$ are also saved in such file with the variable name 't'.

In principle, the users are free to use their own Matlab functions to generate the state variables and observations of the model on any time discretization, and so provided their own file data_ModelName.mat.

In what follows, a basic procedure for generating the time series $\{\mathbf{x}(t_0),\ldots,\mathbf{x}(t_{M-1})\}$ and $\{\mathbf{z}_{t_0},\ldots,\mathbf{z}_{t_{M-1}}\}$ is listed, which is based on the functions Euler.m and Observations.m provided by the **SdeEstimation** toolbox. It is assumed that, in the script Const_ModelName.m, the constant deltat is multiple of deltat_min.

6.6.1 Basic procedure for generating state variables and observations

```
% Model definition
Const_ModelName;

% system noise seed
SNseed=1;
% observed noise seed
ONseed=1;

% number of points for filtering
```

```
n_m=round((tTotal - t0) ./ deltat) + 1;
% number of points for simulations
n_m_max=round((tTotal - t0) ./ deltat_min) + 1;
pp=(n_m_max-1)/(n_m-1);

% integration by Euler method
[t,proceso]=Euler(t0,X0,n_m_max,deltat_min,FileNames,Theta,SNseed);

% sampling the observations for filtering
t=t(1:pp:n_m_max);
proceso=proceso(:,1:pp:n_m_max);

% generation of noisy observations
datos=Observations(t,proceso,Theta,FileNames,ObsX,ONseed);

% plot state variables and observations
PlotModelRealization(t,datos,proceso,ObsX);

% save generated data
FilePath=fileparts(which(ModEq));
Name=[FilePath,filesep,'data' ModEq(3:end)];
save(Name,'t','datos','proceso')
```

6.7 Script Test_ModelName.m

The script Test_ModelName.m tests the approximate filter for the model (1)-(2) with the data of the file data_ModelName.mat. The function LLfilter.m provided by the **SdeEstimation** toolbox has five types of implementations:

1. classical deterministic Local Linearization filter,
2. deterministic Local Linearization filter with artificial points between observation times,
3. adaptive deterministic Local Linearization filter,
4. stochastic Local Linearization filter with artificial points between observation times,
5. exact filter of minimum variance.

The two implementations of the Stochastic Local Linearization filter uses parallel computations.

6.7.1 Basic procedure for filtering

```
% Model definition
Const_ModelName;

% Load the data and observation times
load data_ModelName
Z=datos;

% plot state variables and observations
PlotModelRealization(t,Z,proceso,ObsX);

% interval of time where the likelihood function will be computed
LKWindow=[t(1) t(end)];

% adaptive deterministic LL filter with the specified absolute and relative tolerances AbsTol,RelTol
```

```
mpts=[];                        % with no restrinction to the maximum number artificial points
  or
n_m = length(t);
mpts = 30.*ones(1,n_m);         % with a maximum of 30 artificial points between observation times
LLF = LLfilter(t,Z,Xf0,Pf0,Theta,FileNames,LKWindow,mpts,AbsTol,RelTol);

% plot observations and filter estimates
PlotFilterEstimates(t,Z,LLF,ObsX)
```

6.7.2 Filtering options

Alternatively, the command lines corresponding to the *adaptive deterministic LL filter* can be replaced by those of one of the following types of filters:

```
% Exact Linear Filter of Minimum Variance
LLF = LLfilter(t,Z,Xf0,Pf0,Theta,FileNames,LKWindow,[],0,0);

% non adaptive deterministic LL filter with number of points between observations specified in mpts
n_m = length(t);
mpts = zeros(1,n_m);          % classical LL filter
LLF = LLfilter(t,Z,Xf0,Pf0,Theta,FileNames,LKWindow,mpts);

% non adaptive deterministic LL filter with number of points between observations specified in mpts
n_m = length(t);
mpts = 3.*ones(1,n_m);        % a priori 3 artificial points between observation times
LLF = LLfilter(t,Z,Xf0,Pf0,Theta,FileNames,LKWindow,mpts);

% non-adaptive time-stepping stochastic LL filter
n_m = length(t);
mpts=32.*ones(1,n_m);           % a priori 32 artificial points between observation times
NumSim=[];                      % not restriction on the maximum number of simulations
AbsTol=[0,0,1e-6];
RelTol=[0,0,1e-4];
LLF = LLfilter(t,Z,Xf0,Pf0,Theta,FileNames,LKWindow,mpts,AbsTol,RelTol,NumSim);

% non-adaptive time-stepping stochastic LL filter with fixed number of stochastic simulations
n_m = length(t);
mpts=32.*ones(1,n_m);           % a priori 32 artificial points between observation times
NumSim=200.*ones(1,n_m);        % 200 simulations at each observation time
AbsTol=[0,0,inf];
RelTol=[0,0,inf];
LLF = LLfilter(t,Z,Xf0,Pf0,Theta,FileNames,LKWindow,mpts,AbsTol,RelTol,NumSim);
```

6.8 Script Like_ModelName.m

The script Like_ModelName.m performs the state and parameter estimation of the model (1)-(2) with the data of the file data_ModelName.mat. The function InnovEstimator.m provided by the SdeEstimation toolbox uses the five types of implementations of the function LLfilter.m listed in the previous subsection. In addition, the function InnovEstimator.m can uses the Matlab optimization functions fmincon.m, fminsearch.m and patternsearch.m, and the extra global optimization function GaussUMDA.m combined with fmincon.m.

6.8.1 Basic procedure for parameter estimation

```
% Model definition
Const_ModelName;

load data_ModelName
Z=datos;

% no initial state variable to be estimated
LB_X=[];                    % lower bound
UB_X=[];                    % upper bound
Xf0Index=[];

% parameters 2 and 3 to be estimated
LB_T=0.5*Theta;             % lower bound
UB_T=1.5*Theta;             % upper bound
ThetaIndex=[2 3];

% state initial values for local optimization
Xf0=[];

% parameter initial values for local optimization
Theta0=Theta;

% lower and upper bound of the parameters to be estimated
LB=[LB_X(Xf0Index) LB_T(ThetaIndex)];
UB=[UB_X(Xf0Index) UB_T(ThetaIndex)];

% setting the optimization method
OptimMethod='fmincon';
OptimOptions = optimset(OptimMethod);

% optimization with a priory statistics on the innovations
nu_stast=[];

% interval of time where the likelihood function will be computed
LKWindow=[t(1) t(end)];

% setting the approximate innovation estimator with adaptive deterministic LL Filter
mpts=[];                          % with no restrinction to the maximum number artificial points
  or
n_m = length(t);
mpts = 30.*ones(1,n_m);           % with a maximum of 30 artificial points between observation times
EstimOptions = EstimSet(t,Z,Xf0,Pf0,Theta0,FileNames,LKWindow,nu_stast, ...
                        LB,UB,OptimMethod,OptimOptions,mpts,AbsTol,RelTol);

% computing the innovation estimators
InnEst = InnovEstimator(Xf0Index,ThetaIndex,EstimOptions);

% Summary of the parameter estimation
DisplayParameterEstimates(InnEst,X0,Theta,Xf0Index,ThetaIndex,EstimOptions)
```

```
% plot observations and fitting-filter estimates
PlotFilterEstimates(t,Z,InnEst.LLF,ObsX)

% plot statistic of the standarized fitting-innovation
PlotStatTestFitInn(InnEst.LLF)
```

6.8.2 Parameter estimation options

Alternatively, the command lines corresponding to the *setting of the approximate innovation estimator with the adaptive deterministic LL filter* can be replaced by those of one of the following types of filters:

```
% setting the approximate innovation estimator with non adaptive Deterministic LL Filter
n_m = length(t);
mpts = zeros(1,n_m);            % classical LL filter
  or
mpts = 10.*ones(1,n_m);         % a priori 10 artificial points between observation times
EstimOptions = EstimSet(t,Z,Xf0,Pf0,Theta0,FileNames,LKWindow,nu_stast, ...
                         LB,UB,OptimMethod,OptimOptions,mpts);

% setting the exact innovation estimator
EstimOptions = EstimSet(t,Z,Xf0,Pf0,Theta0,FileNames,LKWindow,nu_stast, ...
                         LB,UB,OptimMethod,OptimOptions,[],0,0);

% setting the approximate innovation estimator with Stochastic LL Filter
n_m = length(t);
mpts=32.*ones(1,n_m);           % a priori 32 artificial points between observation times
NumSim=[];                      % not restriction on the maximum number of simulations
AbsTol=[0,0,1e-6];
RelTol=[0,0,1e-4];
  or
mpts=32.*ones(1,n_m);           % a priori 32 artificial points between observation times
NumSim=200.*ones(1,n_m);        % 200 simulations at each observation time
AbsTol=[0,0,inf];
RelTol=[0,0,inf];
EstimOptions = EstimSet(t,Z,Xf0,Pf0,Theta0,FileNames,LKWindow,nu_stast, ...
                         LB,UB,OptimMethod,OptimOptions,mpts,AbsTol,RelTol,NumSim);
```

6.8.3 Optimization options

The four optimization options in *setting the optimization method* are the following:

```
OptimMethod='fmincon';
OptimMethod='fminsearch';
OptimMethod='patternsearch';
OptimMethod='UMDA+fmincon';
```

Users are free to set convenient parameters in the four optimization methods.

Regularly, by following the practical recommendations of [3], local optimization algorithms work well for the estimation of a small set of parameters when the model is not over parameterized and when there are appropriated initial values for the parameters. Otherwise, global optimization algorithms with local refinement as the 'UMDA+fmincon' proposed in [5] are necessary. However, global optimization algorithms might fail when explore values in the space of parameters that produce divergent solutions of (1) or for which (1) has not solution. To prevent these failures, a suitable maximum number of points in between consecutive observation times and - if it is the case - a maximum number of stochastic simulations for computing the predictions must be specified in the input variables 'mpts' and 'NumSim', respectively.

There are two implementations of the global optimization algorithm 'UMDA': 1) the parallel one, that is automatically used by the function InnovEstimator.m when the innovation estimator is computed by means of a deterministic filter; and 2) the sequential one, that is used when the innovation estimator is computed by means of the stochastic filter.

6.9 Missing data

In the case of a time series of observations $\mathbf{z}_{t_0}, \ldots, \mathbf{z}_{t_{M-1}}$ with missing data, the observation equation in the file OE_ModelName.m is writing down in the usual way (i.e, like when there is not missing data). That is, just specifying the expression for the observed state variables. The time instant t_k of a missing data for an observed state variable $\mathbf{x}^i$ is only specified in the matlab variable 'Z' corresponding to the time series of observations $\mathbf{z}_{t_0}, \ldots, \mathbf{z}_{t_{M-1}}$. It is done by setting 'Z(j,k+1)=inf' when the data $\mathbf{z}^j_{t_k}$ corresponding to of the state variable $\mathbf{x}^i(t_k)$ is missing, where j denotes the row of 'Z' corresponding to the observations of the state variable $\mathbf{x}^i$. Below, the example SIR illustrates this subject.

7 Test models

The models below are included as examples in the **SdeEstimation** toolbox. Each one of them illustrates at least one functionality of the **SdeEstimation** toolbox. Each test example 'ModelName' has a file 'data_ModelName.mat' with a realization of the model in such a way that the user can call directly to the estimation scripts 'Test_ModelName.m' or 'Like_ModelName.m'. The parameter estimation obtained with different filters is summarized in file named "Summary of Results.txt" allocated in corresponding folder 'ModelName'.

7.1 Models with known exact predictions

Example 1 (named ExactMul). Non-autonomous state equation with multiplicative noise [8, 9]

$$dx = \alpha txdt + \sigma\sqrt{t}xdw \tag{7}$$

and observation equation

$$z_{t_k} = x(t_k) + e_{t_k}, \text{ for } k = 0, 1, .., M-1, \tag{8}$$

with $\alpha = -0.1$, $\sigma = 0.1$, observation noise variance $\Pi_k = 0.0001$, and initial value $x_{t_0} = 1$ at $t_0 = 0.5$. Observation times $(t)_{M=11} = \{t_k = t_0 + k : k = 0, 1, \ldots, M-1\}$. For this state equation, the exact predictions for the first two moments of x are given in [8].

This example illustrates that, with respect to the innovation estimator based on classical filters, the one based of adaptive deterministic filters significantly enhance the parameter estimation of the test equations from a reduced number of observations distant in time. See this model in [9] for a detailed simulation study concerning this result. See also [9] to observe the effect of large sampling periods over the estimated parameters with exact and approximate filters.

In this example, there is not significant difference among the estimators provided by the adaptive deterministic filter and the stochastic one, but large difference in computational cost.

Example 2 (named ExactAdd). Non-autonomous state equation with two additive noises [8]

$$dx = atxdt + \sigma_1 t^2 e^{at^2/2} dw^1 + \sigma_2\sqrt{t}dw^2 \tag{9}$$

and observation equation

$$z_{t_k} = x(t_k) + e_{t_k}, \text{ for } k = 0, 1, .., M-1 \tag{10}$$

with $a = -0.25$, $\sigma_1 = 5$, $\sigma_2 = 0.1$, observation noise variance $\Pi_k = 0.0001$, and initial value $x_{t_0} = 10$ at $t_0 = 0.01$. Observation times $(t)_{M=11} = \{t_k = t_0 + k : k = 0, 1, \ldots, M-1\}$. For this state equation, the exact predictions for the first two moments of x are given in [8].

This example also illustrates that, with respect to the innovation estimator based on classical filters, the one based of adaptive deterministic filters significantly enhance the parameter estimation of the test equations from a reduced number of observations distant in time.

In this example, there is not significant difference among the estimators provided by the adaptive deterministic filter and the stochastic one, but large difference in computational cost.

However, the confidential interval for the exact innovation estimator $\widehat{\sigma}_1$ is a 27% of its value. The Kolmogorov-Smirnov and Eagle tests for the stardarized fitting-innovation are rejected with a short margin for all the estimators including the exact one. Therefore, the fitting-innovation is not a white noise. This implies that, the sampling frequency or/and the number of data are unsuitable for a satisfactory parameter estimation.

Example 3 (named CIR). Cox-Ingersoll-Ross (CIR) model of interest rates [2]

$$dx = (a + \beta x)dt + \sigma\sqrt{x}dw \tag{11}$$

and observation equation

$$z_{t_k} = x(t_k) + e_{t_k}, \text{ for } k = 0, 1, .., M-1. \tag{12}$$

Model parameters set as those estimated in [1] from the 307 values of the annualized one-month US Treasure Bill from June, 1964 to December 1989. That is, $a = 0.0189$, $\beta = -0.2339$, and $\sigma^2 = 0.0073$. Observation noise variance $\Pi_k = 0$, and initial value $x_{t_0} = 0.1$ at $t_0 = 0$. Observation times $(t)_{M=307} = \{t_k = 0.083k : k = 0, 1, \ldots, M-1\}$. For this state equation, the exact predictions for the first two moments of x are given in [8].

With the optimization option OptimMethod='UMDA+fmincon', this example illustrates the use of the global optimization Algorithm 4 of [5] with local refinement. In this example, there is not significant difference between the parameters estimated with the exact filter and two of its deterministic approximations.

The fitting-innovation for the exact innovation estimator is not a white noise because the Ljung-Box test is rejected. The sampling frequency or/and the number of data are unsuitable for a satisfactory parameter estimation.

Example 4 (named LinearOscillator). Stochastic harmonic oscillator with random frequency and force [7]

$$du = vdt \tag{13}$$

$$dv = (a - \varpi u)dt - \sigma_1 u dw^1 + \sigma_2 dw^2, \tag{14}$$

where u and v denote the position and speed of the oscillatory body, respectively, and w^1 and w^2 are independent standard Wiener processes. Model parameters set as in [12], $\varpi = 30$, $a = 10$, and $\sigma_1 = \sigma_2 = 0.5$, with initial value $u(t_0) = v(t_0) = 1$ at $t_0 = 0$. Observation equation

$$z_{t_k} = v(t_k) + e_{t_k}, \text{ for } k = 0, 1, .., M-1 \tag{15}$$

on the observation times $(t)_{M=801} = \{t_k = 0.1k : k = 0, 1, \ldots, M-1\}$, with observation noise variance $\Pi_k = 0.1$. For this state equation, the exact predictions for the first two moments of x are given in [7].

This example illustrates the use of the exact innovation estimator. There is not significant difference between the parameters estimated with local and global optimization algorithms. See this model in [12] illustrating the use of the confidence interval and the statistics of the fitting-innovation for a suitable experimental design. Observe the evolution of the fitting-innovation process to a Gaussian white noise during the optimization process.

Example 5 (named DoublePotential). Stochastic dynamical system with bimodal stationary distribution [14]

$$dx = -(\alpha x + \beta x^3)dt + \sigma dw \tag{16}$$

and observation equation

$$z_{t_k} = x(t_k) + e_{t_k}, \text{ for } k = 0, 1, .., M-1 \tag{17}$$

with $\alpha = -1$, $\beta = 0.1$ and $\sigma = 2$. $\Pi_k = 0$ is the observation noise variance, and $x_{t_0} = 0$ at $t_0 = 0$. Observation times $(t)_{M=501} = \{t_k = k : k = 0, 1, \ldots, M-1\}$.

For this state equation, the conditional mean m and variance v of x satisfied the ordinary differential equations

$$dm/dt = -\alpha m - \beta(3mv + m^3)$$
$$dv/dt = -2\alpha v - 6\beta(v^2 + vm^2) + \sigma^2.$$

This example is addressed to show the convergence of the stochastic LL filter to the exact Minimum Variance filter in the case that the SDE has multimodal stationary distribution. In consequence, it also illustrates the convergence of the approximate innovation estimator with stochastic LL Filter to the exact innovation estimator. The example shows the high accuracy of the innovation estimates with the stochastic LL filter in comparison with the classical and adaptive deterministic filters.

7.2 Models with unknown exact predictions

Example 6 (named VanderPolAdd). Van der Pol oscillator with random input [4]

$$dx_1 = x_2 dt \tag{18}$$
$$dx_2 = (-(x_1^2 - 1)x_2 - x_1 + \alpha)dt + \sigma dw \tag{19}$$

and observation equation

$$z_{t_k} = x_1(t_k) + e_{t_k}, \text{ for } k = 0, 1, .., M-1. \tag{20}$$

where the intensity $\alpha = 0.5$ and the variance $\sigma^2 = (0.75)^2$ of the random input are set as in [9]. In addition, $\Pi_k = 0.001$ is the observation noise variance, and $\mathbf{x}_{t_0}^{\intercal} = [1\ 1]$ at $t_0 = 0$. Observation times $(t)_{M=31} = \{t_k = k : k = 0, 1, \ldots, M-1\}$.

This example illustrates that, with respect to the innovation estimator based on classical filters, the one based of adaptive deterministic filters significantly enhance the parameter estimation of the test equations from a reduced number of observations distant in time. See [9] for a detailed simulation study concerning this result.

In this example, there is not significant difference among the estimators provided by the adaptive deterministic filter and the stochastic one, but large difference in computational cost.

Example 7 (named VanderPolMul). Van der Pol oscillator with random frequency [4]

$$dx_1 = x_2 dt \tag{21}$$
$$dx_2 = (-(x_1^2 - 1)x_2 - \alpha x_1)dt + \sigma x_1 dw \tag{22}$$

and observation equation

$$z_{t_k} = x_1(t_k) + e_{t_k}, \text{ for } k = 0, 1, .., M-1, \tag{23}$$

where the frequency mean $\alpha = 1$ and the variance $\sigma^2 = 1$ are set as in [9]. Additionally, $\Pi_k = 0.001$ is the observation noise variance, and $\mathbf{x}_{t_0}^{\intercal} = [1\ 1]$ at $t_0 = 0$. Observation times $(t)_{M=31} = \{t_k = k : k = 0, 1, \ldots, M-1\}$.

This example illustrates that, with respect to the innovation estimator based on classical filters, the one based of adaptive deterministic filters significantly enhance the parameter estimation of the test equations from a reduced number of observations distant in time. See [9] for a detailed simulation study concerning this result.

In this example, there is not significant difference among the estimators provided by the adaptive deterministic filter and the stochastic one, but large difference in computational cost.

Example 8 (named FitzhughNagumo). Stochastic Fitzhugh-Nagumo model

$$dx_1 = \alpha_1(x_1 - \frac{x_1^3}{3} + x_2)dt \tag{24}$$

$$dx_2 = -\frac{1}{\alpha_1}(x_1 - \alpha_2)dt + \alpha_3 dw \tag{25}$$

and discrete observation equation

$$z_{t_k} = x(t_k) + e_{t_k} \ , \tag{26}$$

with model parameters as in [5]. That is, $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = 0.1$, and initial values $(x_1(t_0), x_2(t_0)) = (-0.9323, -0.6732)$ at $t_0 = 0$. In addition, $\Pi_k = 0.001\mathbf{I}$ is the observation noise variance, and $(t)_{M=101} = \{t_k = 0.5k : k = 0, 1, \ldots, M-1\}$ are observation times.

This example illustrates the effect on the innovation estimator produced by a not suitable parameter's initial value in local optimization algorithms. Observe the difference in the evolution of the fitting-innovation process for the suitable $(2, -3, 0.5)$ and not suitable $(-2, -3, 0.5)$ initial value of the parameters, and in the statistics of the final fitting-innovation process. See [5] for a detailed simulation study concerning this result.

In this example, there is not significant difference among the estimators provided by the adaptive deterministic filter and the stochastic one, but large difference in computational cost.

Example 9 (named SIR). SIR epidemic model [13]

$$dS/dt = -\beta SI \tag{27}$$

$$dI/dt = \beta SI - \gamma I \tag{28}$$

$$dR/dt = \gamma I, \tag{29}$$

where the S, I and R represent the number of individuals susceptible, infected and removed from the disease (due to immunization or death), $\beta = 0.0004$ and $\gamma = 0.04$ are the rates for infection and recovering, respectively, and $(S(t_0), I(t_0), R(t_0)) = (997, 3, 0)$ are the initial values. Discrete observation equation

$$\begin{bmatrix} z_{t_k}^1 \\ z_{t_k}^2 \end{bmatrix} = c(t_k) \begin{bmatrix} I(t_k) \\ R(t_k) \end{bmatrix} + \begin{bmatrix} e_{t_k}^1 \\ e_{t_k}^2 \end{bmatrix} \tag{30}$$

modeling the uncertainty in the data collection, where $c(t_k)$ is a random squared matrix indicating the instant of time that the variable I or R is not collected. $\Pi_k = c(t_k)\mathbf{I}$ is the observation noise variance and $(t)_{M=100} = \{t_k = k : k = 1, \ldots, M-1\}$ are the observation times.

This examples illustrates how to proceed with the observations of two state variables with missing data. For directions, see Section 6.9. See also files 'OE_SIR' and 'Gen_SIR'.

Observe the evolution of the fitting-innovation process to a Gaussian white noise during the optimization processes.

Example 10 (named Heston). Stochastic volatility model of Heston in finance [6]

$$dS = \alpha S dt + S\sqrt{\sigma} dw^1 \tag{31}$$

$$d\sigma = \lambda(\mu - \sigma)dt + \gamma\sqrt{\sigma} dw^2, \tag{32}$$

for the dynamics of a asset price S over the time t, with $cov(w^1, w^2) = \rho$, where α is the expected return, μ is the mean long-term variance, λ is the rate at which the variance reverts to its long-term mean, and

γ the volatility of the variance process. Model parameters were set as in [12], $\alpha = 0.08$, $\mu = 0.2$, $\lambda = 1.5$, $\gamma = 0.1$, and $\rho = -0.5$, which mimics the dynamics of actual financial data. For estimating purpose, with the change of variable $s = \ln(S)$, the equations (31)-(32) is rewritten as

$$\begin{aligned} ds &= (\alpha - \frac{1}{2}\sigma)dt + \sqrt{\sigma}dw^1 \qquad (33) \\ d\sigma &= (\beta_1 - \lambda\sigma)dt + \beta_2\sqrt{\sigma}dw^1 + \beta_3\sqrt{\sigma}dw^2, \qquad (34) \end{aligned}$$

with $\beta_1 = \lambda\mu = 0.3$, $\beta_2 = \gamma\rho = -0.05$ and $\beta_3 = \gamma\sqrt{1-\rho^2} = 0.0866$. The initial value for the log price and volatility are $(s(t_0), \sigma(t_0)) = (ln(353), 0.01)$. Observation times $\{t_k\}_{M=4251} = \{t_k = 0.004k : k = 0, 1, \ldots, M-1\}$ representing the 4251 daily prices in 17 years.

The example illustrates how to deal with two correlated Winner process and with the estimation of the physical parameters from artificial ones.

The fitting-innovation is not a white noise. The number of data is insufficient for a suitable estimation of the parameters. See this model in [12] illustrating the use of the confidence interval and the statistics of the fitting-innovation for a suitable experimental design.

8 Demos

The following demos are available:

1. Demo_ExactMul
2. Demo_ExactAdd
3. Demo_CIR
4. Demo_LinearOscillator
5. Demo_VanderPolAdd
6. Demo_VanderPolMul
7. Demo_FitzhughNagumo
8. Demo_SIR

References

[1] Chan K.C., Karolyi G.A., Longstaff F.A.,Sanders A.B., An empirical comparison of alternative models of the short-term interest rate, J. Finance 47 (1992) 12091227.

[2] Cox J.C., Ingersoll J.E., Ross S.A., A theory of the Term structure of interest rates. Econometrica, 53 (1985) 285-408.

[3] Gill P.E., Murray W., Wright M.H., Practical Optimization. Academic Press, 1981.

[4] Gitterman M., The noisy oscillator, World Scientific, 2005.

[5] Gonzalez-Arenas Z., Jimenez J.C., Lozada-Chang L., Santana R., Estimation of distribution algorithms for the computation of innovation estimators of diffusion processes. Math. Comput. Simul., 187 (2021) 449-467.

[6] Heston S.L., A closed-form solution for options with stochastic volatility with applications to bond and currency options, Rev. Fin. Stud. 6 (1993) 327-343.

[7] Jimenez J.C., Simplified formulas for the mean and variance of linear stochastic differential equations, Appl. Math. Letters, 49 (2015) 12-19.

[8] Jimenez J.C., Approximate linear minimum variance filters for continuous-discrete state space models: convergence and practical adaptive algorithms. IMA J. Math. Control Inform., 36 (2019) 341-378.

[9] Jimenez J.C., Bias reduction in the estimation of diffusion processes from discrete observations. IMA J. Math. Control. Inform., 37 (2020) 1468-1505.

[10] Jimenez J.C., Ozaki T., Linear estimation of continuous-discrete linear state space models with multiplicative noise. Syst. Control Lett., 47 (2002) 91101.

[11] Jimenez J.C., Ozaki T., An approximate innovation method for the estimation of diffusion processes from discrete data. J. Time Ser. Anal., 27 (2006) 7797.

[12] Jimenez J.C., Yoshimoto A., Miwakeichi F., State and parameter estimation of stochastic physical systems from uncertain and indirect measurements, Eur. Phys. J. Plus, 136 (2021) 869.

[13] Kermack W.O., McKendrick A.G., A Contribution to the Mathematical Theory of Epidemics. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. 115 (1927) 700-721.

[14] H. Singer, Parameter estimation of nonlinear stochastic differential equations: Simulated maximum likelihood versus extended Kalman filter and Ito-Taylor expansion, J. Comput. Graph. Stats., 11 (2002) 972-995.