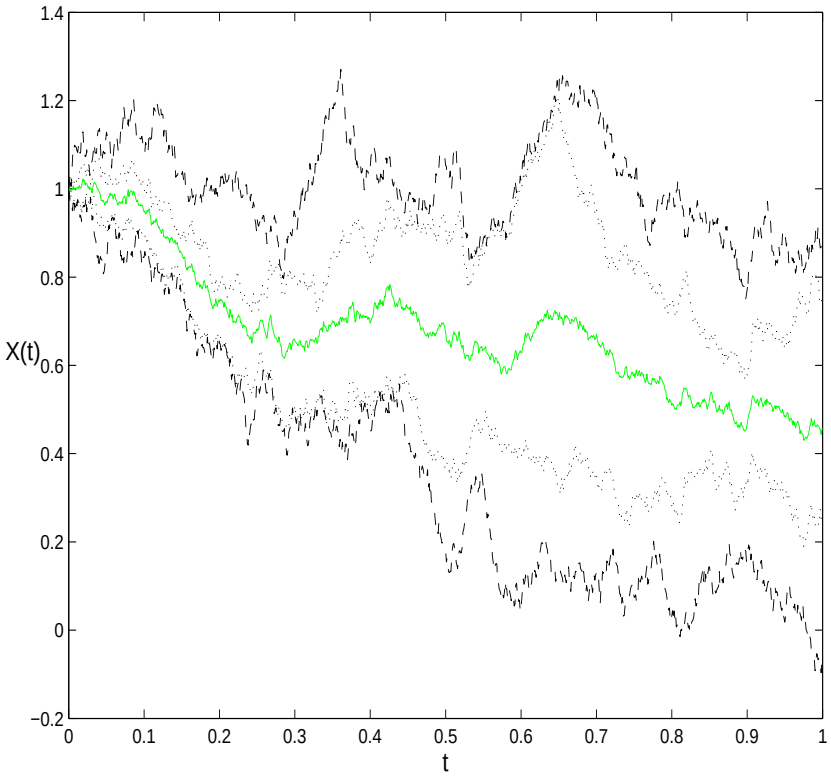
SDE Toolbox

Simulation and Estimation of Stochastic Differential Equations with MATLAB

`http://sdetoolbox.sourceforge.net`

by Umberto Picchini

`umberto.picchini@biomatematica.it`
`http://www.biomatematica.it/pages/picchini.html`



User's Guide for version 1.4.1

Contents

Preface

The area of deterministic differential equations (ordinary (ODE), partial (PDE), or delay (DDE)) is a rich one, well–researched with plenty of software packages and tools available for the numerical solution of such systems. On the other side, a Mathworks[1] supported MATLAB toolbox for the numerical treatment of *stochastic differential equations* (SDE) is lacking.

The present SDE TOOLBOX is *not* intended to provide a complete package filling the gap above: this is a toolbox for simulating sample paths of an SDE solution, computing statistics and estimating the parameters from data. Other important issues (e.g. stability of the solutions) are not treated. This has to be intended as a customizable piece of code which, in the author's intentions, should furnish ideas to stimulate the users in developing their own SDE package, and give some programming hints to SDE newbies. In particular, users may receive suggestions from this package to develop a more efficient code in their favorite programming language (e.g. in C/C++, Fortran, etc.).

SDE's newbies are highly encouraged in taking a look into the excellent monographies [1, 2] and into the article [3], the latter giving a MATLAB-based introduction to SDE simulation. Other useful references for numerical methods are [4, 5, 6, 7, 8]. Highly specialistic references for SDE theory and stochastic calculus are [1, 9, 10, 11, 12, 13].

This toolbox has been created with the support of the Biomathematics Laboratory (`http://www.biomatematica.it`) at the Institute for Systems Analysis and Informatics "A. Ruberti" (IASI, `http://www.iasi.cnr.it`), organ of the Italian National Research Council; though, possible bugs, errors and misprints are on my own responsibility.

Umberto Picchini, November 2007
`umberto.picchini@biomatematica.it`

[1] http://www.mathworks.com

How To Use This Guide

SDE Toolbox can be used in two ways: the first one is considered in Chapter 1, where it is described how to use the package by means of an interactive, user friendly implementation. The SDE's newbies as well as the occasional users may appreciate this feature which, together with the present guide, provides an easy way to simulate and estimate SDE models following step-by-step instructions. The second, more advanced, way to use the Toolbox is considered in Chapter 2, where some practical, "non-interactive" examples are presented. It is suggested to not skip Chapter 1, since many features, hints and limitations of the package are described there. Finally, an appendix with some methodological background closes the guide.

Chapter 1

Using SDE Toolbox Interactively

1.1 Functionalities

SDE Toolbox is a Matlab package for simulating sample paths of the solution of a user defined Itô or Stratonovich SDE, estimating the parameters from data and visualizing statistics; users can also simulate and estimate an SDE model chosen from a models library. Notice that, in this version of the toolbox, **multidimensional SDEs need to have *diagonal noise***, see appendix A.3.

1.2 Requirements

Only Matlab[1] base is required to run the toolbox. SDE Toolbox has been tested on Matlab 6.5 (R13) and 7.3.0 (R2006b) for Windows; however it may work under different platforms as well.

1.3 Download and Installation

Download SDE Toolbox (`.zip` or `.tgz`) from `http://sdetoolbox.sourceforge.net`. Unpack the archive in any place recognizable by Matlab, then add it to the Matlab search path (do not forgot to include the `models_library` subfolder): e.g. from the Matlab File Menu, select *set path*, then choose *Add with Subfolders* and finally select the `SDE_Toolbox` folder.

1.4 Quick Start

Run `SDE_demo.m` to get an idea of some of the toolbox functionalities. Further tools (parameter estimation capabilities) are considered in section 1.5.

1.4.1 SDE_demo.m

This demo considers the following Itô SDE with given initial condition X_0

$$dX_t = \frac{1}{2}a^2 X_t dt + aX_t dW_t, \qquad t \in [t_0, T] \tag{1.1}$$

or equivalently the following Stratonovich SDE

$$dX_t = aX_t \circ dW_t, \qquad t \in [t_0, T]. \tag{1.2}$$

[1] http://www.mathworks.com

Models (1.1) and (1.2) have solution given by

$$X_t = X_0 \exp(aW_t), \quad t \in [t_0, T]. \tag{1.3}$$

where $X_0 = X_{t_0}$. This demo asks the user to provide:

1. the number of trajectories to be simulated for the numerical solution of model (1.1) and (1.2);
2. the values of t_0 and T;
3. the value of the integration stepsize $0 < h \ll T - t_0$ to be used for the numerical approximation of the SDE solution, see section A.3;
4. the value of a;
5. the value of X_0 ($\neq 0$ to avoid a straightforward solution).

E.g. by specifying only 3 trajectories[2] in $[t_0, T] = [0, 1]$, with integration stepsize $h = 0.001$, $a = 0.5$ and $X_0 = 1$ we get the plots in Figure 1.1. Figure 1.1(a) compare the Euler-Maruyama solutions (black solid lines) of the Itô SDE (1.1) with the corresponding Milstein solutions (dotted lines) and the true solutions (1.3) (magenta solid lines): the Milstein and the analytic solutions are so close that they result practically undistinguishable. Figure 1.1(b) reports the point-by-point sample mean (green solid lines) of the Euler-Maruyama solutions of the Itô SDE (1.1), their empirical 95% confidence bands (from the 2.5th to the 97.5th percentile; dashed lines) and their first and third quartile (dotted lines). Figure 1.1(c) is the same as Figure 1.1(b) but refers to the Milstein solution of the Itô SDE (1.1). Figure 1.1(d) compare the Milstein solutions (black solid lines) of the Stratonovich SDE (1.2) with the corresponding true solutions (1.3) (magenta solid lines): again, the Milstein and the analytic solutions are so close that they result practically undistinguishable. Figure 1.1(e) is the same as Figure 1.1(c) but refers to the Stratonovich SDE (1.2). Since the considered SDE has known analytic solution, this demo provides some statistics regarding the error induced by the approximation scheme, as given in equation (A.6) (appendix A.5): using the settings above we notice that at time $T = 1$ the Euler-Maruyama method for the Itô SDE (1.1) implies an average error equals to 1.048×10^{-2}, while the Milstein scheme for the Itô SDE implies an average error of 5.962×10^{-5} (the same value is obtained for the Milstein approximation of the Stratonovich SDE (1.2)). This demo also produces further statistics, but we are going to comment them under more interesting settings, as given below.

If we simulate 1000 trajectories and let the settings above unchanged (except for the number of simulations, of course), we have that at time $T = 1$ the Euler-Maruyama method for the Itô SDE (1.1) implies an average error equal to 5.268×10^{-3}, while the Milstein scheme implies an average error of 7.372×10^{-5} (the same value is obtained for the Milstein approximation of the Stratonovich SDE (1.2)). From these results we conclude that the Milstein method is more accurate than the Euler-Maruyama one: this is in general true when the diffusion term is non-constant, see appendix A.3. Descriptive statistics are reported with respect to the simulated values at the endpoint T (see section A.5): e.g. for the Euler-Maruyama approximation of the Itô SDE we have that at $T = 1$ $\mathbb{E}(X_T) \simeq 1.161$ where $\mathbb{E}(\cdot)$ denotes expectation, $Var(X_T) \simeq 0.367$, $Median(X_T) = 1.029$, etc. For each SDE type and for each numerical scheme the histograms of X_t at $t = T$ are reported, see e.g. Figure 1.2: the empirical distribution of X_t is clearly asymmetric, and in fact the conditional distribution of $X_t|X_0$ is log-normal.

1.5 Using the SDE Toolbox Models Library

The SDE Toolbox comes with a library of SDE models which can be easily simulated and estimated. The main function of the library is `SDE_library_run.m`. By running `SDE_library_run.m`

[2]Here we choose a small number of trajectories for ease of illustration of Figure 1.1: users may choose a larger number to get more interesting results.

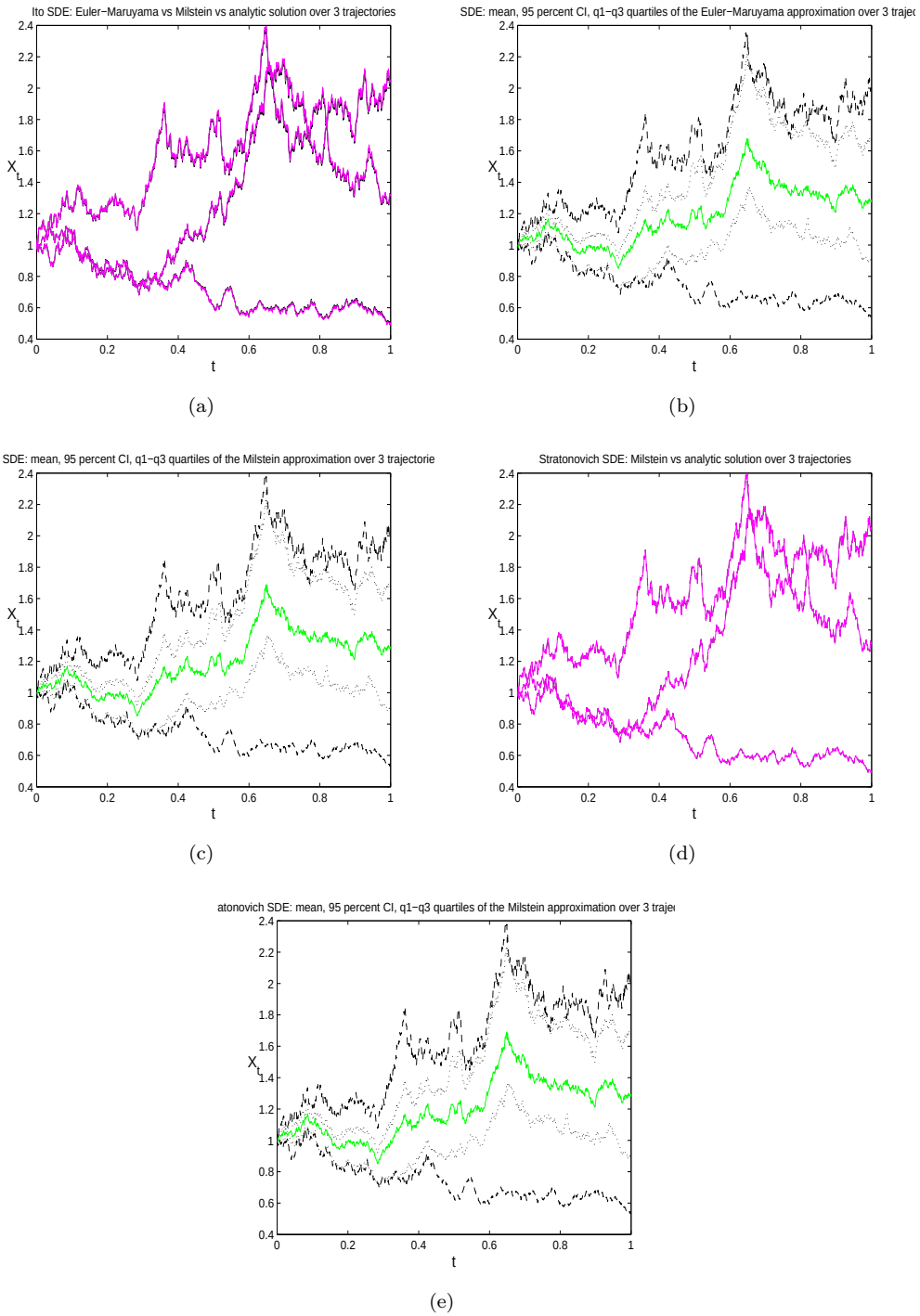


Figure 1.1: SDE_demo.m output. See the main text for details.

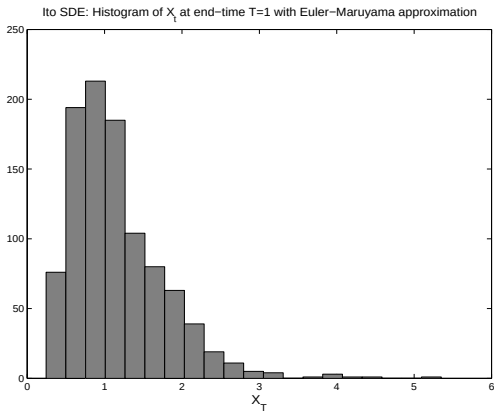


Figure 1.2: Itô SDE with Euler-Maruyama approximation: histogram of X_t at $t = T = 1$ over 1000 trajectories.

the user can choose among several Itô and Stratonovich SDE to be simulated in the time-interval $[t_0, T]$ and (this is optional) estimate the parameters from data. The library includes the models in Table 1.1 (see section 1.6 to define your own model): `Mxa` is an Itô SDE and `Mxb` is the corresponding Stratonovich SDE (`x`=1,2,...). `SDE_library_run.m` asks the user to interactively specify:

1. "do you want to estimate parameters from data [Y/N]?" If *yes* then it is asked to specify whether the data should be loaded from an ASCII tab-delimited `.dat` file (and in this case it is asked to write the data filename *without* the `.dat` extension, e.g. `sampledata1`) or not; in the latter case the data are simulated.

2. the model name, e.g. `M1a`, `M1b` etc.;

3. the parameter(s) value(s), e.g. the value of `a` (a) and `sigma` (σ) for model `M1a`. *No assumption on the parameters is checked: the user is responsible for giving appropriate values to the parameters such that conditions for the SDE solution existence are satisfied*;

4. the initial condition(s) `X0` (X_0); however, if the data are loaded from a file, the initial condition will be overwritten by the first X value stored in the datafile;

5. the number of trajectories to be simulated for the numerical solution of the chosen model;

6. the values of `T0` (t_0) and `T` (T) (only for simulated data);

7. the value of the integration stepsize $0 < h \ll T - t_0$ to be used for the numerical approximation of the SDE solution;

8. the integration method, only for Itô SDEs: `EM` (Euler-Maruyama) or `Mil`(Milstein). For Stratonovich SDEs the Milstein method is automatically employed;

9. if the parameters need to be estimated then, in the case of simulated data, it is asked to supply the number $n + 1$ of the equally spaced observation-times $t_0, t_1, ..., t_n = T$ at which data have been created and the desired parameter estimation method (see section A.6);

10. if some parameter need to be estimated then an "initial value" should be provided, and will be used as a starting point for the estimation algorithm (but see the recommendations in section A.6).
    Q: *Ok, but how can we specify the parameters to be held fixed (constant) and those to be estimated?*
    A: this is controlled by the `PARMASK` array in `SDE_library_setup.m`, see section 1.5.2.

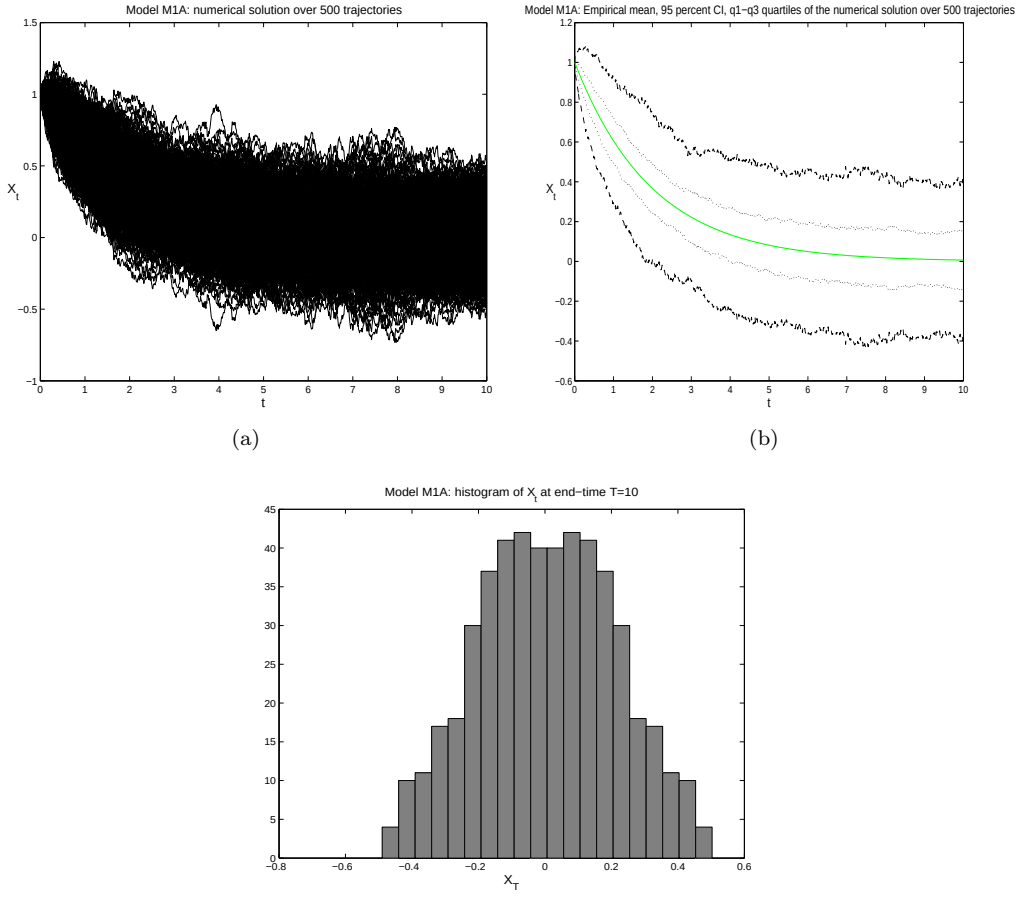
Model name	Definition
`M1a`	$dX_t = -aX_t dt + \sigma dW_t$
`M1b`	$dX_t = -aX_t dt + \sigma \circ dW_t$
`M2a`	$dX_t = (aX_t + b)dt + \sigma dW_t$
`M2b`	$dX_t = (aX_t + b)dt + \sigma \circ dW_t$
`M3a`	$dX_t = (a - \sigma^2/2)dt + \sigma dW_t$
`M3b`	$dX_t = (a - \sigma^2/2)dt + \sigma \circ dW_t$
`M4a`	$dX_t = aX_t dt + bX_t dW_t$
`M4b`	$dX_t = (aX_t - 1/2b^2 X_t)dt + bX_t \circ dW_t$
`M5a`	$dX_t = (aX_t + c)dt + (bX_t + d)dW_t$
`M5b`	$dX_t = [(a - 1/2b)X_t + c - 1/2bd]dt + (bX_t + d) \circ dW_t$
`M6a`	$dX_t = [1/2a(a-1)X_t^{1-2/a}]dt + aX_t^{1-1/a} dW_t$
`M6b`	$dX_t = [aX_t^{1-1/a}] \circ dW_t$
`M7a`	$dX_t = [-1/2a^2 X_t]dt + a\sqrt{1 - X_t^2} dW_t$
`M7b`	$dX_t = a\sqrt{1 - X_t^2} \circ dW_t$
`M8a`	$dX_t = [a^2 X_t(1 + X_t^2)]dt + a(1 + X_t^2)dW_t$
`M8b`	$dX_t = a(1 + X_t^2) \circ dW_t$
`M9a`	$dX_t^1 = [\beta_{11}\alpha_1 + \beta_{12}\alpha_2 - \beta_{11}X_t^1 - \beta_{12}X_t^2]dt + \sigma_1 dW_t^1$ $dX_t^2 = [\beta_{21}\alpha_1 + \beta_{22}\alpha_2 - \beta_{21}X_t^1 - \beta_{22}X_t^2]dt + \sigma_2 dW_t^2$
`M9b`	$dX_t^1 = [\beta_{11}\alpha_1 + \beta_{12}\alpha_2 - \beta_{11}X_t^1 - \beta_{12}X_t^2]dt + \sigma_1 \circ dW_t^1$ $dX_t^2 = [\beta_{21}\alpha_1 + \beta_{22}\alpha_2 - \beta_{21}X_t^1 - \beta_{22}X_t^2]dt + \sigma_2 \circ dW_t^2$
`M10a`	$dX_t = a(b - X_t)dt + \sigma\sqrt{X_t} dW_t$
`M10b`	$dX_t = [a(b - X_t) - \sigma^2/4]dt + \sigma\sqrt{X_t} \circ dW_t$

Table 1.1: SDE models library. *Mxa* is an Itô SDE and *Mxb* is the corresponding Stratonovich SDE ($x = 1, 2, ...$).

E.g. suppose we do not want to estimate parameters nor load data from a file, but we want to simulate a model, then by choosing `500` trajectories to be simulated from model `M1a`, with `a=0.5`, `sigma=0.2`, `X0=1`, in the time-frame `[T0,T]=[0,10]` using the Euler-Maruyama method with stepsize `0.01`, we get the plots in Figure 1.3. Figure 1.3(a) reports the trajectories obtained using the specified settings. Figure 1.3(b) reports the point-by-point sample mean (green solid line) of the trajectories, their empirical 95% confidence bands (from the 2.5th to the 97.5th percentile; dashed lines) and their first and third quartile (dotted lines). Figure 1.3(c) reports the empirical distribution of X_t at time $T = 10$. Finally, descriptive statistics of X_t at $t = T$ are returned (see also section 1.4.1 and appendix A.5).

If some parameters need to be estimated, but data are not loaded from a `.dat` file, then an $(n+1) \times d$ array of simulated data is created from the chosen model at equally spaced times $t_0, t_1, ..., t_n$, using the specified parameters; then, the parameters can be estimated from the simulated data using one of the methods described in section A.6.1. E.g. using the settings above but specifying $n = 150$ observational time-points and `2000` simulated trajectories, after few iterations of the `NPAR` procedure (Non-PARametric, see A.6.1) using 1 and 0.4 as starting values for a and σ respectively, we get the following parameter estimates and asymptotic 95% confidence intervals: $\hat{a} = 0.653$ $[0.223, 1.083]$ and $\hat{\sigma} = 0.187$ $[0.159, 0.215]$. Using the `PAR` (PARametric, see A.6.2) procedure with the same settings the estimates $\hat{a} = 0.714$ $[0.311, 1.118]$ and $\hat{\sigma} = 0.195$ $[0.173, 0.218]$ are returned.

Q: *Why should be useful to estimate the parameters if we already know their true values?*
A: Before estimating the parameters from *raw* (not simulated) data, we may desire to check for the appropriate number of simulations, the appropriate stepsize, the number of observations etc. necessary to obtain reasonable estimates using our favorite estimation method. We can do that by *simulating* data from an appropriate SDE using some settings and some parameters values: then we forget about the specified parameters values and we try to estimate them from the simulated data. The estimation procedure is repeated again and again by changing the parameters starting values, the number of trajectories, the stepsize etc. until an estimate close to the true parameters values is returned. At this point we have



(a) (b) (c)

Figure 1.3: Plots from the model *M1a* output.

(hopefully!) obtained a reasonable setup to estimate the parameters efficiently from the raw data with the chosen SDE model.

The datafile `sampledata1.dat` contains 167 observations (read section 1.5.1 for further details) generated from model `M1a` in the time-interval $[t_0, T] = [0, 5]$ with $a = 2$, $\sigma = 0.5$ and $X_0 = 5$ using the Euler-Maruyama scheme[3] with a stepsize $h = 0.03$ (and random seed `0`, this is the default value, see the examples in Chapter 2). If we estimate a and σ from these data by the `PAR` procedure with 1000 Euler-Maruyama trajectories from model `M1a`, using a smaller stepsize $h = 0.003$ and 4 and 0.8 as starting values for a and σ respectively, after few iterations of the estimation algorithm we get $\hat{a} = 1.986$ $[1.666, 2.306]$ and $\hat{\sigma} = 0.453$ $[0.407, 0.500]$; the fit of the simulated model at the estimated parameters vs the observations is given in Figure 1.4. Using the `NPAR` procedure with the same settings, the estimates $\hat{a} = 1.961$ $[1.529, 2.394]$ and $\hat{\sigma} = 0.430$ $[0.384, 0.476]$ are returned. Notice that in both the cases a is well identified whereas the true value of σ is not included in the 95% confidence intervals (in the first case it is only marginally included).

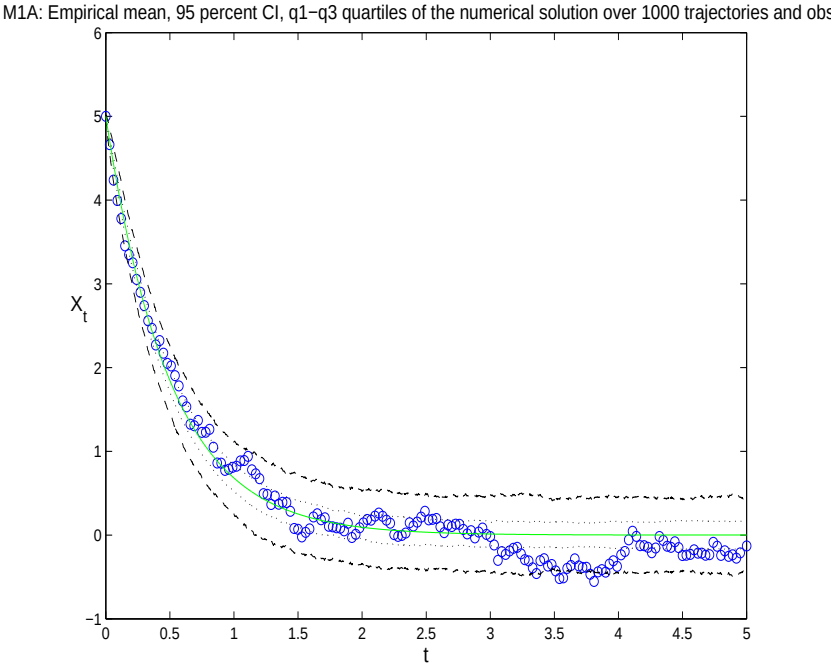


Figure 1.4: Data (∘) vs the empirical mean (green line), the 95% confidence bands (dashed lines) and the first-third quartile (dotted lines) of 1000 simulated trajectories of model *M1a* with estimated parameters $(\hat{a}, \hat{\sigma}) = (1.986, 0.453)$; see main text for details.

Warning: the `NPAR` procedure can be used either with Itô and Stratonovich SDEs and with the Euler-Maruyama or the Milstein integration scheme; on the other side the `PAR` procedure can be used only with Itô SDEs and Euler-Maruyama integration scheme; see section A.6.

1.5.1 Working with datafiles

Using SDE Toolbox it is possible to work with your own data, stored in an ASCII tab-delimited file with `.dat` extension[4]. The datafile should always contain three columns: the first one should contain the times sorted in non-decreasing order, the second one the corresponding measured values from the dynamical process being observed and the third one the numerical labels identifying the

[3]Notice that, in this case, the Euler-Maruyama and the Milstein schemes coincide since the diffusion part of the SDE is constant (*additive noise*), see also section A.3

[4]if you prefer to work with a different extension, modify *SDE_ getdata.m* accordingly.

state variables. E.g. `sampledata1.dat` contains observations generated from model `M1a` in the time-interval $[t_0, T] = [0, 5]$ using the Euler-Maruyama scheme with a stepsize $h = 0.03$ (see the previous section for details). Thus the first column contains the times $\{t_0 = 0, 0.03, 0.06, ..., 5 = T\}$ whereas column 2 contains the corresponding values from the X process, and since X is one-dimensional the third column contains a series of 1's, that is each value in column 2 correspond to a measurement from the first (and only) variable of model `M1a`.

The datafile `sampledata2.dat` contains observations generated from the two-dimensional model `M9a` in the time-interval $[t_0, T] = [2, 10]$ with $X_0^1 = 1$, $X_0^2 = 2$, $\alpha_1 = 0.3$, $\alpha_2 = 0.2$, $\beta_{11} = 0.1$, $\beta_{12} = 0.2$, $\beta_{21} = 0.2$, $\beta_{22} = 0.3$, $\sigma_1 = 0.1$, $\sigma_2 = 0.2$ using the Euler-Maruyama scheme (see footnote 3) with a stepsize $h = 0.03$ (and random seed `0`, this is the default value). Here we have two variables, X^1 and X^2, thus the first row in `sampledata2.dat` contains the observation time $t_0 = 2$ with respect to the corresponding variable considered in column 3: since the label-value in {row 1, column 3} equals 1, then the measured value in column 2 correspond to $X_t^1|_{t=2}$; the second row in `sampledata2.dat` contains the observation time $t_0 = 2$ with respect to the variable considered in column 3: this time the label-value in row 2 column 3 equals 2, thus the measured value in column 2 correspond to $X_t^2|_{t=2}$, etc.

See Example 5 in Chapter 2 for the description of a further datafile (`sampledata3.dat`).

Notice that the estimation methods considered in section A.6.1 and A.6.2 only allow for *fully observed* SDEs and thus, if X is multivariate, then for each observational time t_i all the coordinates of the X process must have been measured, otherwise an error is returned. E.g. a datafile like the following is *not allowed*

```
2.00    0.970    1
2.00    0.180    2
2.04    0.974    1
2.08    0.946    1
2.08    0.184    2
...
```

because at time $t = 2.04$ the value for the second variable is not given.

1.5.2 Optimization Settings

Skip this section if you are not interested in parameter estimation issues.

In `SDE_library_setup.m` it is possible to specify, among other things, the set of parameters to be estimated, i.e. here the users can identify the parameters to be estimated and those to be held constant via the `PARMASK` array. `PARMASK` and `bigtheta` (the complete set of parameters) have the same length and if `PARMASK(p)` equals `0` then the pth parameter in `bigtheta` is held constant (i.e. it will not be estimated) whereas if `PARMASK(p)` equals `1` then it is *free to vary* (i.e. it will be estimated from data). Thus if we set, e.g. in model `M1a`, `PARMASK = [0,1,1]` then both the second and the third parameters will be estimated; if `PARMASK = [0,0,1]` only the third parameter will be estimated etc. *Notice that the parameters corresponding to the SDE initial conditions must always be held constant*, e.g. in model `M1a` `PARMASK(1)` (corresponding to X_0) should always be set to zero.

The parameters to be estimated are free to vary into an hypercube whose limits are defined in `SDE_library_optsetup.m`: default limits for a given parameter θ are $[-20 \cdot \theta, 20 \cdot \theta]$. However the user can modify the region boundaries: in fact `PARMIN(p)` (`PARMAX(p)`) represents the lower (upper) limit for the pth parameter in `bigtheta` [5], thus in `SDE_library_run.m` the user can specify e.g. `PARMIN=[0,1,-5]` and `PARMAX=[1,5,20]` *after* the invocation to `SDE_library_optsetup(bigtheta)` but *before*

```
[theta,thetandx, ACTMIN, ACTMAX]  = SDE_param_mask(bigtheta);
```

[5] Thus constant parameters have an upper and lower limit also: this may result useful for future versions of the toolbox.

In fact the previous statement creates automatically the arrays `ACTMIN` and `ACTMAX` which contain the limits only for the free to vary parameters, and which are passed as argument to the (Nelder-Mead simplex) minimization algorithm `fminsearchbnd.m`[6]. Default values for the minimization algorithm are provided via the structure `MYOPT`, created in `SDE_library_optsetup.m`: type

```
help optimset
```

at the MATLAB prompt for further details. See also Example 5 in Chapter 2.

Hint: *it is strongly suggested the modification of the default parameters space boundaries (by using the procedure above) every time the user "knows" the real limits of the parameters. This is useful to localize the search of the minimization algorithm within trusted regions of the parameters space. Otherwise the algorithm may return unrealistic parameter estimates. For example, if the user thinks that e.g. the third parameter value should be non-negative and should not exceed 10* then he/she may specify `PARMIN(3)=0` and `PARMAX(3)=10`.

1.6 Running Your Own SDE Model

To implement your own SDE model it is necessary to update `SDE_library_setup.m` (this step is necessary if you use the Toolbox interactively; otherwise see Chapter 2) and create `mySDE_sdefile.m`, where `mySDE` can be substituted with any other string. Implementing a new SDE model is easy, just follow the instructions below. You can also modify `SDE_model_description.m`, though this is unnecessary.

1.6.1 One-Dimensional SDEs

Cut one of the m-files stored into the `models_library` folder, e.g. `M4_sdefile.m`, then rename the file into e.g. `mySDE_sdefile.m` and paste it into the `SDE_Toolbox\models_library` directory. Then modify the code according to the following instructions (you are free to change the names of the variables and the parameters):

1. change the first line of `mySDE_sdefile.m` into

```
function [out1,out2,out3] = mySDE_sdefile(t,x,flag,bigtheta,...
                            SDETYPE,NUMDEPVARS,NUMSIM)
```

2. store all the parameters and the initial condition of the SDE into the `bigtheta` array, e.g.
```
X0     = bigtheta(1); % this is the mySDE initial condition X_0
p1     = bigtheta(2); % this is the first parameter of mySDE
p2     = bigtheta(3); % this is the second parameter of mySDE
```
etc.

3. for both the Itô and the Stratonovich definition define the `driftX` and the `diffusionX` part of the SDE into the appropriate slots (see appendix A.2 for the definitions of *drift* and *diffusion*); define `derivativeX`, which is given by $dg(X)/dX$, where $g(\cdot)$ is the diffusion term of the SDE. Notice that:

   - the variable `x` is a row array (the code is vectorized over the simulations, i.e. `x` contains the values of the process X at time `t` on every simulated trajectory), thus operate on `x` using elementwise operations, i.e. use `.*`, `.^`, `./` in place of `*`, `^`, `/`;
   - Itô SDEs can be solved using either the Euler-Maruyama and the Milstein methods whereas Stratonovich SDE can only be solved via Milstein integration;

[6] This function by John D'Errico (woodchips@rochester.rr.com) is a modification of the official MATLAB *fminsearch.m* function enabling bound constrained optimization.

- it is useless to specify `derivativeX` if you plan to solve the SDE using *only* the Euler-Maruyama method: e.g. in this case you can put `derivativeX = []` both in the Itô and in the Stratonovich definition.

4. insert the SDE initial condition: go to the bottom of the file and change the line
   `out2 = Xzero;  % write here the SDE initial value(s)`
   into
   `out2 = X0;  % write here the mySDE initial value`

5. modify `SDE_library_setup.m` (this is necessary only when using the Toolbox interactively, otherwise see Chapter 2) by adding the new SDE model features to the list of the available models, e.g. by mimicking the other models and assuming `mySDE` to be defined via three parameters (X_0, p_1, p_2) just add something like the following[7]:

```
case 'mySDE_Ito'  % for example
     PROBLEM = 'mySDE';  % for example
     SDETYPE = 'Ito';
     NUMDEPVARS = 1;
     fprintf('\n\nYou choose     dXt = write here the Ito SDE definition');
     if((strcmp(LOADDATA,'N')&&strcmp(PARESTIMATE,'N'))||(strcmp(LOADDATA,'N')&&...
         strcmp(PARESTIMATE,'Y'))||(strcmp(LOADDATA,'Y')&&strcmp(PARESTIMATE,'N')))
        p1  = input('\n\nWrite the value of the ''p1'' parameter: ');
        if(isempty(p1))
           error('''p1'' must be specified');
        end
        p2  = input('\nWrite the value of the ''p2'' parameter: ');
        if(isempty(p2))
           error('''p2'' must be specified');
        end
        X0 = input('\nWrite the value of the initial condition X0: ');
        if(isempty(X0))
           error('X0 must be specified');
        end
        bigtheta(1) = X0;
        bigtheta(2) = p1;
        bigtheta(3) = p2;
        PARBASE = bigtheta;
     elseif( strcmp(LOADDATA,'Y') && strcmp(PARESTIMATE,'Y') )
        bigtheta(1:NUMDEPVARS) = XOBS(1,:);
     end
     PARMASK = [0,1,1];  % for example
case 'mySDE_Strat'
     PROBLEM = 'mySDE';   % for example
     SDETYPE = 'Strat';
     NUMDEPVARS = 1;
     fprintf('\n\nYou choose     dXt = write here the Stratonovich SDE definition');
     if((strcmp(LOADDATA,'N')&&strcmp(PARESTIMATE,'N'))||(strcmp(LOADDATA,'N')&&...
         strcmp(PARESTIMATE,'Y'))||(strcmp(LOADDATA,'Y')&&strcmp(PARESTIMATE,'N')))
        p1  = input('\n\nWrite the value of the ''p1'' parameter: ');
        if(isempty(p1))
           error('''p1'' must be specified');
        end
        p2  = input('\nWrite the value of the ''p2'' parameter: ');
        if(isempty(p2))
           error('''p2'' must be specified');
        end
        X0 = input('\nWrite the value of the initial condition X0: ');
        if(isempty(X0))
           error('X0 must be specified');
        end
        bigtheta(1) = X0;
        bigtheta(2) = p1;
        bigtheta(3) = p2;
```

[7] I know...that's definitely boring and not elegant. I will try to improve the style in future versions.

```
   PARBASE = bigtheta;
elseif( strcmp(LOADDATA,'Y') && strcmp(PARESTIMATE,'Y') )
   bigtheta(1:NUMDEPVARS) = XOBS(1,:);
end
PARMASK = [0,1,1];  % for example
```

6. Optional: for you ease modify `SDE_model_description.m` by adding the `mySDE` model definition to the list of the available models.

1.6.2 Multi-Dimensional SDEs

This section assumes familiarity with section 1.6.1. Remember that this version of the toolbox handles SDE with *diagonal noise* only (see appendix A.3), e.g. model `M9a` can be written as

$$dX_t = \beta(\alpha - X_t)dt + gdW_t, \qquad X_0 = x_0$$

where $X_t = (X_t^1, X_t^2)^T$, $\alpha = (\alpha_1, \alpha_2)^T$, $W_t = (W_t^1, W_t^2)^T$, $x_0 = (x_0^1, x_0^2)^T$

$$\beta = \begin{pmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{pmatrix} \qquad g = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$$

and T denotes transposition. Thus g is a diagonal matrix.

1. Modify `SDE_library_setup.m` and `SDE_model_description.m` (this is optional) according to section 1.6.1, the only relevant difference being that in `SDE_library_setup.m` `NUMDEPVARS` should be set to the dimension of the SDE, e.g. for the two-dimensional SDE `M9a-M9b` we have `NUMDEPVARS = 2`, for a three-dimensional SDE we have `NUMDEPVARS = 3` and so on;

2. create the file `mySDE_sdefile.m` as described in section 1.6.1, then use `M9_sdefile.m` as a guideline to the next steps (the actions similar to the one-dimensional case are skipped).

3. in the section after `xsplitted = SDE_split_sdeinput(x,NUMDEPVARS)` split the input `x` of `mySDE_sdefile.m` according to the dimension of the SDE: e.g. if X is two-dimensional ($\Rightarrow X_t = (X_t^1, X_t^2)$) write

   ```
   X1 = xsplitted{1}; X2 = xsplitted{2};
   ```

   if X is three-dimensional ($\Rightarrow X_t = (X_t^1, X_t^2, X_t^3)$) write

   ```
   X1 = xsplitted{1}; X2 = xsplitted{2}; X3 = xsplitted{3};
   ```

   etc.;

4. for each coordinate of both the Itô and the Stratonovich definition of the SDE, define the corresponding `driftXk` and `diffusionXk`, where `k` goes from 1 to the dimension of X. Define the corresponding `derivativeXk`, which is given by $dg^{k,k}(X)/dX_k$ where $g^{k,k}$ denotes the (k,k)-th element in g. [8]

5. define `out1`, `out2` and `out3` [9]. E.g. for a three-dimensional SDE ($\Rightarrow$ `NUMDEPVARS=3`) write:

   ```
   out1 = zeros(1,NUMDEPVARS*NUMSIM);
   out1(1:NUMDEPVARS:end) = driftX1;
   out1(2:NUMDEPVARS:end) = driftX2;
   out1(3:NUMDEPVARS:end) = driftX3;
   ```

[8] Remember that, according to section 1.6.1, it is useless to specify the derivativeXk's when planning to solve the SDE using *only* the Euler-Maruyama method: e.g. in this case you can put derivativeXk = [ ] both in the Itô and in the Stratonovich definition.

[9] Notice that if the case in footnote 8 applies, it is possible to define out3=[ ].

```
out2 = zeros(1,NUMDEPVARS*NUMSIM);
out2(1:NUMDEPVARS:end) = diffusionX1;
out2(2:NUMDEPVARS:end) = diffusionX2;
out2(3:NUMDEPVARS:end) = diffusionX3;
out3 = zeros(1,NUMDEPVARS*NUMSIM);
out3(1:NUMDEPVARS:end) = derivativeX1;
out3(2:NUMDEPVARS:end) = derivativeX2;
out3(3:NUMDEPVARS:end) = derivativeX3;
```

6. at the bottom of `mySDE_sdefile.m` insert the SDE initial conditions, e.g. for a three-dimensional SDE with initial conditions `Xzero1`, `Xzero2` and `Xzero3` write:

```
case 'init'
    out1 = t;

    out2 = [Xzero1 Xzero2 Xzero3];

    out3 = [];
```

Chapter 2

Using SDE Toolbox In Your Programs

In the previous chapter the essential features of the Toolbox have been described via a user-friendly implementation (by means of `SDE_library_run.m`). Here some of the simulations considered in Chapter 1 are reproduced by suggesting some pieces of code, so that advanced users may cut-paste-modify and insert them into their own MATLAB programs. Notice that we have always used a *seed* equal to zero for the generation of pseudo-random Wiener increments.

Example 1. The following commands simulate 500 trajectories from the following Itô SDE (model `M1a` in Table 1.1)

$$dX_t = -aX_t dt + \sigma dW_t, \qquad X_0 = x_0$$

with $t \in [0, 10]$, using the Euler-Maruyama method with fixed stepsize $h = 0.01$, when $x_0 = 1$ and $(a, \sigma) = (0.5, 0.2)$, using `0` as seed for the generation of pseudo-random Wiener increments.

```
>> x0 = 1;             % the SDE initial condition
>> a = 0.5;            % the SDE structural parameter 'a'
>> sigma = 0.2;        % the SDE structural parameter 'sigma'
>> problem = 'M1';     % the name of the experiment
>> t0 = 0;             % the time-span initial value
>> T = 10;             % the time-span last value
>> h = 0.01;           % the stepsize for the numerical integration
>> numsim = 500;       % the number of trajectories
>> sdetype = 'Ito';    % must be 'Ito' when using the EM scheme
>> randseed = 0;       % this is the seed for the generation of pseudo-random Wiener increments
>> integrator = 'EM';  % must be 'EM' (Euler-Maruyama, only for Ito SDEs) or 'Mil'
>> model = 'M1a';
>> numdepvars = 1;  % the dimension of the SDE
>> yesdata = 0; % can be 0 (no raw data available) or 1 (data available)
   % store the simulated trajectories into 'xhat'
>> xhat = SDE_euler([x0,a,sigma],problem,[t0:h:T],numdepvars,numsim,sdetype,...
                    randseed);
   % plot the trajectories
>> SDE_graph([x0,a,sigma],xhat,yesdata,problem,sdetype,integrator,numdepvars,...
            [t0:h:T],model,numsim,[],[],randseed)
```

and the plots in Figure 1.3 are returned. Notice that, here and in the following examples, it is possible to set `model=[]`. Figure 1.3(a) reports the trajectories obtained using the specified settings. Figure 1.3(b) reports the point-by-point sample mean (solid line) of the simulated trajectories, their empirical 95% confidence bands (from the 2.5th to the 97.5th percentile; dashed lines) and their first and third quartile (dotted lines). Figure 1.3(c) reports the empirical distribution of

X_t at the simulation end-time $t = T = 10$. The following command returns a series of Monte Carlo statistics for the solution process X at $T = 10$ based on the simulated trajectories: e.g. the approximated process mean, variance, median, skewness, kurtosis, etc.:

```
>> SDE_stats([x0,a,sigma],xhat,problem,[t0:h:T],numdepvars,numsim,sdetype,...
   integrator,randseed)
```

For example, we get $\mathbb{E}X_T \simeq 6.654 \cdot 10^{-3}$, $Var(X_T) \simeq 0.043$, $Skewness(X_T) \simeq 0$ and $Kurtosis(X_T) \simeq 2.433$.

Example 2. Here we use the same settings considered in Example 1 to simulate trajectories with the Milstein scheme from the Itô SDE `M1a`:

```
>> integrator = 'Mil';
>> xhat = SDE_milstein([x0,a,sigma],problem,[t0:h:T],numdepvars,numsim,sdetype,...
                       randseed);
>> SDE_graph([x0,a,sigma],xhat,yesdata,problem,sdetype,integrator,numdepvars,...
             [t0:h:T],model,numsim,[],[],randseed)
```

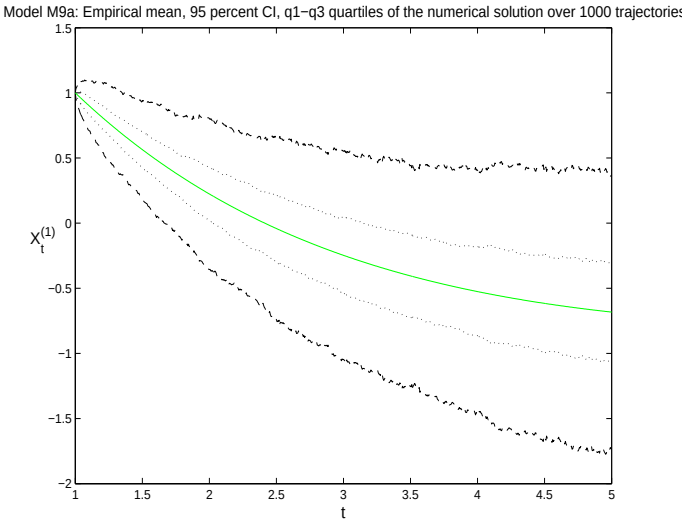
To simulate from the Stratonovich definition given in `M1b` we use (notice that for Stratonovich SDEs the Milstein scheme *must* be employed):

```
>> sdetype = 'Strat';
>> xhat = SDE_milstein([x0,a,sigma],problem,[t0:h:T],numdepvars,numsim,sdetype,...
                       randseed);
>> SDE_graph([x0,a,sigma],xhat,yesdata,problem,sdetype,integrator,numdepvars,...
             [t0:h:T],model,numsim,[],[],randseed)
```

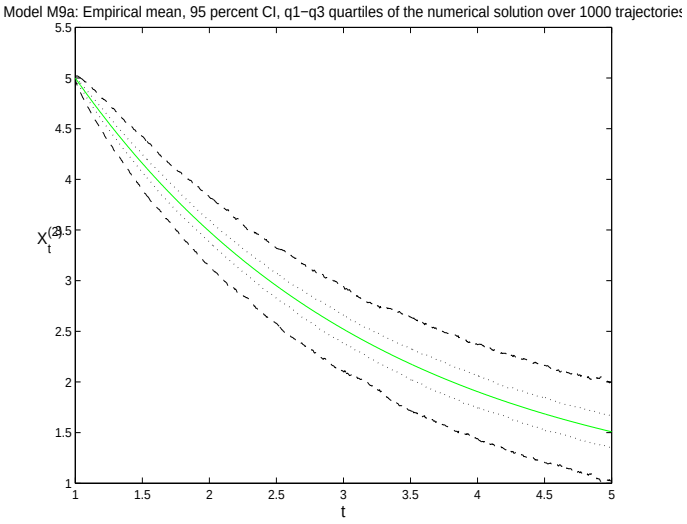
Example 3. Here we simulate 1000 trajectories from the two-dimensional Itô SDE `M9a`, using the Euler-Maruyama method and the following settings: initial conditions given by $(X_0^1, X_0^2) = (1, 5)$ and $(\alpha_1, \alpha_2, \beta_{11}, \beta_{12}, \beta_{21}, \beta_{22}) = (.2, .5, .1, .2, .1, .4, .3, .2)$, $[t_0, T] = [1, 5]$, $h = 0.005$.

```
>> parameters = [1,5,.2,.5,.1,.2,.1,.4,.3,.2];
>> problem = 'M9';
>> t0 = 1;
>> T = 5;
>> h = 0.005;
>> numsim = 1000;
>> sdetype = 'Ito';
>> randseed = 0;
>> integrator = 'EM';
>> model = 'M9a';
>> numdepvars = 2;
>> yesdata = 0;
>> xhat = SDE_euler(parameters,problem,[t0:h:T],numdepvars,numsim,sdetype,...
                    randseed);
>> SDE_graph(parameters,xhat,yesdata,problem,sdetype,integrator,numdepvars,...
             [t0:h:T],model,numsim,[],[],randseed)
```

For each coordinate of the process the corresponding plots are produced. In Figure 2.1 a subset of the output is reported: Figure 2.1(a) reports the point-by-point sample mean (solid line) of the X_t^1 trajectories, their empirical 95% confidence bands (dashed lines) and their first and third quartile (dotted lines). The results for the X_t^2 process are given in Figure 2.1(b). Monte Carlo statistics can be obtained using the `SDE_stats.m` function as previously described.



(a) X_t^1



(b) X_t^2

Figure 2.1: Example 3: a subset of the model *M9a* output; see main text for details.

Notice that, in the previous examples, we did not considered raw data but only simulated data, and thus the parameter `yesdata` was set to `0`. With "raw data" here we mean the observations belonging to a *discretely observed process*, whereas "simulated data" belongs (at least theoretically) to a *continuously observed process*. In other words the simulated data have been generated using a finer mesh than raw data.

Of course when raw data are available (`yesdata=1`) it is possible to work with them, and the observational-times `time` and the corresponding observational-values `xobs` are passed to the integrators and the graphical functions, whereas when `yesdata=0` empty matrices `[]` are passed as in examples 1-3.

Example 4. Here we plot the observations stored in `sampledata1.dat` against model `M1a`, simulated with a finer time-grid ($h = 0.01$); the observations vs the simulated model are given in Figure 2.2.

```
>> x0 = 5;
>> a = 2;
>> sigma = 0.5;
>> problem = 'M1';
>> t0 = 0;
>> T = 5;
>> h = 0.01;
>> numsim = 1000;
>> sdetype = 'Ito';
>> randseed = 0;
>> integrator = 'EM';
>> model = 'M1a';
>> numdepvars = 1;
>> yesdata = 1;
>> xhat = SDE_euler([x0,a,sigma],problem,[t0:h:T],numdepvars,numsim,sdetype,...
                    randseed);
   % get the times and the observed values from 'sampledata1.dat'
>> [xobs,time] = SDE_getdata('sampledata1');
>> SDE_graph([x0,a,sigma],xhat,yesdata,problem,sdetype,integrator,numdepvars,...
             [t0:h:T],model,numsim,time,xobs,randseed)
```

We conclude with a parameter estimation example.

Example 5. The datafile `sampledata3.dat` contains 101 equally spaced observations in $[t_0, T] = [0, 1]$, obtained from the simulated values generated by model `M2a` with the Euler-Maruyama scheme using the parameters $(a, b, \sigma) = (1, 0.1, 0.3)$, $x_0 = 1$ and stepsize $h = 0.01$ (and random seed equal to `0`). We estimate (a, b, σ) from these data by the `NPAR` procedure, using 2000 simulated trajectories following the Euler-Maruyama scheme, using $h = 0.001$ and 3, 0.5 and 0.1 as starting values for a, b and σ respectively. We get the estimates (and asymptotic 95% confidence intervals) $\hat{a} = 0.835$ $[0.060, 1.609]$, $\hat{b} = 0.500$ $[-1.065, 2.065]$ and $\hat{\sigma} = 0.260$ [0.227, 0.293]; the plot of the observations vs the simulated model at the estimated parameters is given in Figure 2.3.

```
>> data = load('sampledata3.dat');    % load the data
>> time = data(:,1);                  % the observational times
>> xobs = data(:,2);                  % the values recorded at 'time'
>> vrbl = data(:,3);                  % the label-variables
>> h = 0.001;                         % the stepsize
>> owntime = [time(1):h:time(end)];   % the simulation time-frame
>> x0 = xobs(1);                      % the initial value for the SDE simulation
>> a = 3;                             % starting value for the optimization
>> b = 0.5;                           % starting value for the optimization
>> sigma = 0.1;                       % starting value for the optimization
>> freeparstart = [a, b, sigma];      % the array of the starting values for the parameters to be estimated
```

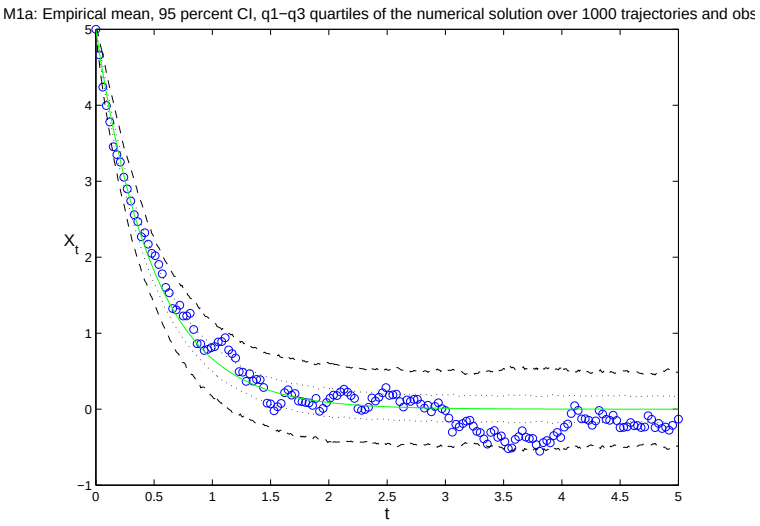


Figure 2.2: Example 4: simulated model *M1a* vs the observations stored in *sampledata1.dat*.

```
>> freeparmin = [1e-6,1e-6,1e-6];     % the lower bounds for the parameters to be estimated
>> freeparmax = [3,1,1];              % the upper bounds for the parameters to be estimated
>> totparmin = [x0,freeparmin];       % the lower bounds for the fixed (x0) and the free to vary parameters
>> totparmax = [x0,freeparmax];       % the upper bounds for the fixed (x0) and the free to vary parameters
>> parmask = [0,1,1,1];               % use 0 to denote constant parameters and 1 otherwise
>> parbase = [x0,freeparstart];       % the array of starting values for constant and free parameters
>> problem = 'M2';
>> numsim = 2000;
>> sdetype = 'Ito';
>> integrator = 'EM';
>> numdepvars = 1;
>> randseed = 0;
>> myopt = optimset('fminsearch');    % optimization settings, type 'help optimset' for details
>> myopt = optimset(myopt,'MaxFunEvals',20000,'MaxIter',5000,'TolFun',1.e-4,'TolX',1.e-4,...
          'Display','iter');
   % 'freeparest' contains the approximated parameters maximum likelihood estimates
>> freeparest = fminsearchbnd('SDE_NPSML',freeparstart,freeparmin,freeparmax,myopt,owntime,...
               time,vrbl,xobs,problem,numsim,sdetype,parbase,totparmin,totparmax,parmask,...
               integrator,numdepvars,randseed);
   % 'totparam' contains the array of fixed + estimated parameters
>> totparam = SDE_param_unmask(freeparest,parmask,parbase);
   % 95% confidence intervals calculation for the free parameters
>> SDE_ParConfInt('SDE_NPSML',freeparest,owntime,time,vrbl,xobs,problem,numsim,sdetype,...
                 parbase,totparmin,totparmax,parmask,integrator,numdepvars,randseed);
>> yesdata = 1;
>> SDE_graph(totparam,[],yesdata,problem,sdetype,integrator,numdepvars,owntime,[],numsim,...
           time,xobs,0);
>> SDE_stats(totparam,[],problem,owntime,numdepvars,numsim,sdetype,integrator,0);
```

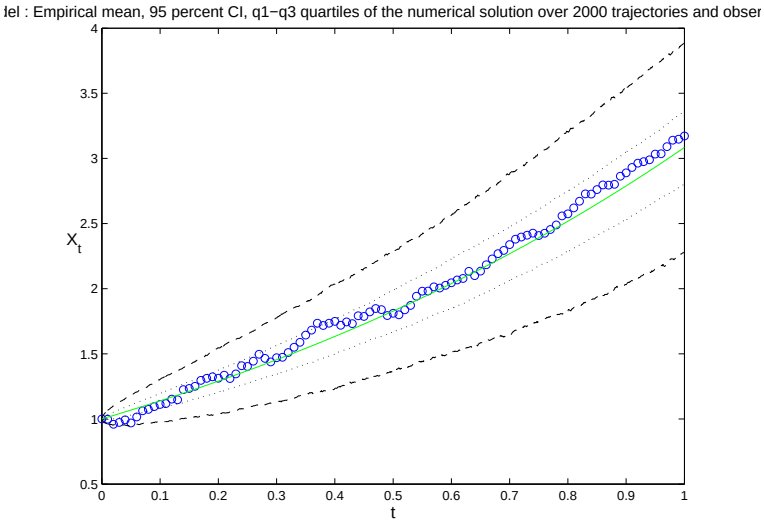


Figure 2.3: Example 5: simulated model *M2a* at $(\hat{a}, \hat{b}, \hat{\sigma}) = (0.835, 0.500, 0.260)$ vs the observations stored in *sampledata3.dat*.

Appendix A

Methodological Background

A.1 Simulating the Wiener Process

A real valued stochastic process W_t, $t \in [0, +\infty)$, is named Wiener process (or Brownian motion) if:

1. $W_0 = 0$ a.s.;
2. $W_{t+h} - W_t \sim \mathcal{N}(0, h)$ $\forall t$, $h > 0$;
3. the increments $W_{t_1} - W_{t_0}, \ldots, W_{t_n} - W_{t_{n-1}}$ are independent for $t_0 < t_1 < \cdots < t_n$.

and $\mathcal{N}(\cdot, \cdot)$ is the normal distribution. Furthermore we have

$$\mathbb{E}(W_t) = 0, \quad \mathbb{E}(W_t^2) = t, \quad \mathbb{E}(W_s W_t) = \min(s, t) \qquad 0 \le s \le t \quad \forall t$$

$$Var(W_t - W_s) = t - s \qquad 0 \le s \le t.$$

W_t is a gaussian process, i.e. for all $t_0 \le t_1 \le \cdots \le t_n$ the random variable $(W_{t_0}, \ldots, W_{t_n}) \in \mathbb{R}^{n+1}$ has a (multi)normal distribution.

When considering a numerical solution of a differential equation, we *must* restrict our attention to a finite subinterval $[t_0, T]$ of the time-interval $[t_0, +\infty)$ and, in addition, it is necessary to choose an appropriate *discretization* $t_0 < t_1 < \cdots < t_n < \cdots < t_N = T$ of $[t_0, T]$, because of computer limitations. The other crucial problem is simulating a sample path from the Wiener process over the discretization of $[t_0, T]$: so considering an equally-spaced discretization, i.e. $t_n - t_{n-1} = (T - t_0)/N = h$, $n = 1, \ldots, N$, where h is the *integration stepsize*, we have the following (independent) random increments

$$W_{t_n} - W_{t_{n-1}} \sim \mathcal{N}(0, h) \qquad n = 1, \ldots, N \tag{A.1}$$

of the Wiener process $\{W_t, t_0 \le t \le T\}$. Moreover, the sampling of normal variates to approximate the Wiener process in the SDE is achieved by computer generation of pseudo-random numbers. However, the use of a pseudo-random number generator needs to be evaluated in terms of statistical reliability. Nevertheless, most commonly used pseudo-random number generators have been found to fit their supposed distribution reasonably well, but the generated numbers often seem not to be independent as they are supposed to be: this is not surprising since, for congruential generators at least, each number is determined exactly by its predecessor [2]. Classical methods for generating independent standard Gaussian distributed pseudo-random numbers are the *Box-Muller* and the *Polar Marsaglia* methods [2], but we used the built-in MATLAB `randn` function implementing the *ziggurat* method [14].

It is straightforward to simulate Wiener trajectories (or Brownian paths), see also section A.4; e.g. the following MATLAB code simulates three trajectories of W_t with $t \in [t_0, T] = [0, 1]$ using 1000 points for the discretization of $[t_0, T]$:

```
% Simulations of Brownian paths on [T0,T]

%---  Settings  ------------------------------------------------------%
randn('state',0)   % fix the initial state to get repeatable results
T0 = 0; T  = 1; N  = 1000;
h = (T - T0) / N;  % the stepsize
NUMSIM = 3;        % the number of desired simulations
%--------------------------------------------------------------------%

NORMRAND = [zeros(1,NUMSIM);randn(N,NUMSIM)]; % an (N+1)x(NUMSIM) matrix...
                                              % of (pseudo)random entries...
                                              % from N(0;1), except for the first row

for(i=1:NUMSIM)
   dW = sqrt(h)*NORMRAND(:,i);  % the Wiener increments (ith simulation)
   W = cumsum(dW);              % the Brownian path (ith simulation)
   plot([T0:h:T],W,'k-'), hold on
   xlabel('t','Fontsize',13)
   ylabel('W(t)','Fontsize',13,'Rotation',0)
end
```

The code above will always produce the same result, since the seed of the random generator has been fixed: substitute `randn('state',0)` with `randn('state',sum(100*clock))` to get different sequences of pseudo-random numbers at each run. Notice that for ease of illustration the code above does not implement the *antithetic variates* method considered in section A.4.

Different implementations are available in [2] and [3], using Pascal and MATLAB codes respectively.

A.2 Itô and Stratonovich SDEs

We can write an d-dimensional SDE in Itô form as

$$dX_t = f(X_t)dt + g(X_t)dW_t, \tag{A.2}$$

or in Stratonovich form as

$$dX_t = f(X_t)dt + g(X_t) \circ dW_t, \tag{A.3}$$

where $f(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ is called the *drift* of the SDE, $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d\times m}$ is called the *diffusion* of the SDE, and W_t is an m-dimensional process having independent scalar Wiener process components ($t \geq t_0$). It is possible to convert from one interpretation to the other in order to take advantage of one of the approaches as appropriate: in the scalar case ($d = 1$), if the Itô SDE is as given in (A.2) then the Stratonovich SDE is given by

$$dX_t = \bar{f}(X_t)dt + g(X_t) \circ dW_t$$

where

$$\bar{f}(X_t) = f(X_t) - \frac{1}{2}\frac{\partial g}{\partial X}(X_t)g(X_t)$$

In other words (A.2) and (A.3), under different rules of calculus, have the same solution: for example, $dX_t = aX_tdt + bX_tdW_t$ has solution $X_t = \exp((a - \frac{1}{2}b^2)t + bW_t)X_0$ as does $dX_t = (a - \frac{1}{2}b^2)X_tdt + bX_t \circ dW_t$. Obviously, in the case of additive noise (g independent of $x \Rightarrow \partial g/\partial x \equiv 0$) the Itô and Stratonovich representations are equivalent. For multidimensional SDEs the relationship between the two representations is given by:

$$\bar{f}_i(X_t) = f_i(X_t) - \frac{1}{2}\sum_{j=1}^{d}\sum_{k=1}^{m} g_{jk}(X_t)\frac{\partial g_{ik}}{\partial X_j}(X_t), \qquad i = 1, \ldots, d.$$

A.3 Euler–Maruyama and Milstein Approximations

Consider the one-dimensional Itô SDE

$$dX_t = f(X_t, \theta)dt + g(X_t, \theta)dW_t, \qquad X_0 = x_0 \tag{A.4}$$

where W is an m-dimensional standard Wiener process, $f : \mathbb{R} \times \Theta \to \mathbb{R}$ and $g : \mathbb{R} \times \Theta \to \mathbb{R}^{1\times m}$ are known functions depending on an unknown finite-dimensional parameter vector $\theta \in \Theta$. For now we drop the reference to θ when not necessary, e.g. we write $f(X_t)$ instead of $f(X_t, \theta)$.

Many SDE systems do not have a (known) analytic solution, so it is necessary to solve these systems numerically: the simplest stochastic numerical approximation is the *Euler–Maruyama* method. Considering the Itô SDE (A.4) on $[t_0, T]$, for a given discretization $t_0 < t_1 < \cdots < t_n < \cdots < t_N = T$ of $[t_0, T]$, an Euler–Maruyama approximation is a continuous time stochastic process satisfying the iterative scheme

$$y_{n+1} = y_n + h_n f(y_n) + g(y_n)\Delta W_n \qquad y_0 = x_0, \quad n = 0, 1, \ldots, N-1 \tag{A.5}$$

where $y_n = y(t_n)$, $h_n = t_{n+1} - t_n$ is the *stepsize*, $\Delta W_n = W(t_{n+1}) - W(t_n) \sim \mathcal{N}(0, h_n)$ with $W(t_0) = 0$, and $\mathcal{N}$ is the normal distribution (the increments ΔW_n can be generated as suggested in section A.1.

The Euler-Maruyama method has strong order of convergence 1/2 (and weak order of convergence 1, [1]). Note that a method may have order of accuracy p in general, but this order may be increased for SDEs of a particular type: for example, the Euler–Maruyama method has strong order of accuracy 1 for systems with *additive noise* (i.e. the diffusion term g is a constant).

Other numerical methods may have a much simpler form when being used to solve additive noise SDEs, and this often leads to a cheaper implementation. As the order of the Euler–Maruyama method is low, the numerical results are inaccurate unless a small stepsize is used, and clearly more efficient methods are needed: one possible improvement is the *Milstein scheme*. For one-dimensional Itô SDE the Milstein scheme is given by

$$y_{n+1} = y_n + hf(y_n) + g(y_n)\Delta W_n + \frac{1}{2}g(y_n)g'(y_n)((\Delta W_n)^2 - h), \qquad y_0 = x_0$$

where the superscript $'$ denotes differentiation with respect to X. This scheme converges with strong order 1 if $\mathbb{E}(x_0^2) < \infty$, if f and g are twice continuously differentiable, and if f, f', g, g' and g'' satisfy a uniform Lipschitz condition. There exists also a Milstein scheme for the corresponding scalar Stratonovich SDE, which is given by

$$y_{n+1} = y_n + hf(y_n) + g(y_n)\Delta W_n + \frac{1}{2}g(y_n)g'(y_n)(\Delta W_n)^2, \qquad y_0 = x_0.$$

Notice that for SDE with additive noise the Euler-Maruyama and the Milstein scheme coincide.

Consider now the following d-dimensional system of (Itô) SDEs

$$dX_t = f(X_t; \theta)dt + g(X_t; \theta)dW_t, \qquad X_0 = x_0$$

where W is m-dimensional, $f : \mathbb{R}^d \times \Theta \to \mathbb{R}^d$ and $g : \mathbb{R}^d \times \Theta \to \mathbb{R}^{d\times m}$. Notice that, **in this version of SDE Toolbox it is assumed that $g(\cdot, \cdot)$ is a diagonal matrix**, and thus we consider the case $m = d$: in this case the SDE is said to have *diagonal noise*, and the multi-dimensional Milstein scheme for the kth component ($k = 1, \ldots, d$) of the Itô SDEs has the form (see [1] for the general case)

$$y_{n+1}^k = y_n^k + hf^k + g^{k,k}\Delta W_n^k + \frac{1}{2}g^{k,k}\frac{\partial g^{k,k}}{\partial X^k}((\Delta W_n^k)^2 - h), \qquad y_0^k = x_0^k$$

whereas for Stratonovich SDEs it has the form

$$y_{n+1}^k = y_n^k + h\bar{f}^k + g^{k,k}\Delta W_n^k + \frac{1}{2}g^{k,k}\frac{\partial g^{k,k}}{\partial X^k}(\Delta W_n^k)^2, \qquad y_0^k = x_0^k$$

where $\bar{f}$ denotes the drift of the SDE in the Stratonovich representation (see section A.2). Here f^k and $\bar{f}^k$ denote the k-th element in f and $\bar{f}$ respectively, whereas $g^{k,k}$ denotes the (k,k)-th element in g.

The multi-dimensional Euler-Maruyama scheme is only available for Itô SDE and is given by

$$y_{n+1}^k = y_n^k + hf^k + g^{k,k}\Delta W_n^k, \qquad y_0^k = x_0^k.$$

A.4 Variance Reduction

SDE Toolbox uses the method of *antithetic variates* when creating the (pseudo-) random Wiener increments (except in `SDE_demo.m` for ease of illustration). This is a commonly used variance-reduction technique in simulation-based methods. To implement antithetic variates when simulating the generic increment given in (A.1) at time t_n, one draws only $R/2$ samples (for ease of illustration here we consider a one-dimensional SDE) $\{z_{t_n}^1, ..., z_{t_n}^r, ..., z_{t_n}^{R/2}\}$ from the normal distribution with mean 0 and variance h (where R is the number of desired trajectories and h is the stepsize), and from each $z_{t_n}^r$ two increments for the Wiener process are constructed: one is built directly from $z_{t_n}^r$, and the other one from its "mirror image" $-z_{t_n}^r$, see also [15]. While we have found antithetic variates to provide only marginal benefit, the calculation cost is also negligible.

A.5 Statistics Based on Monte-Carlo Simulations

Using Monte-Carlo simulations, the following statistical measures can be approximated for the X_t process at the endpoint T (i.e. X_T, see `SDE_stats.m`):

1. the expected (mean) value $\mathbb{E}(X_T)$;
2. the variance $Var(X_T)$;
3. the median $Median(X_T)$;
4. the 95% confidence limits of X_T;
5. the first and third quartile of X_T;
6. the skewness and the kurtosis of X_T;
7. the moments of X_T up to order 7 (it is straightforward to retrieve the moments up to any desired order by modifying `SDE_stats.m`).

Finally, it is important to choose an appropriate numerical integration method: the Milstein scheme (strong order 1) is more precise than the Euler-Maruyama scheme (strong order 0.5), e.g. the *average absolute error* (A.6) for the Milstein scheme is generally lower than for the Euler-Maruyama scheme. When the analytic solution of the SDE is known, the (average absolute) error at time T, depending on the desired number of simulations R, can be computed as proposed in [2]

$$\epsilon = \frac{1}{R}\sum_{r=1}^{R} |X(T,r) - y(T,r)|, \tag{A.6}$$

were $X(T,r)$ and $y(T,r)$ denote the value of the analytic solution at time T in the rth trajectory and the value of the numerical solution for the chosen approximation scheme at time T in the rth trajectory respectively. For the calculation of the error, the analytic solution and the numerical solution *must be computed on the same Brownian path* (i.e. using the same sequence of pseudo-random numbers), see `SDE_demo.m`.

A.6 Parameter Estimation

It is often convenient to model the time evolution of dynamic phenomena by means of a *diffusion process* defined by a stochastic differential equation. Parameters in these stochastic differential equations are crucial for the characterization of dynamic phenomena being considered. It is often the case that these parameters are not known accurately, while sample data for the particular dynamic phenomena are available. Naturally, researchers are interested in obtaining better estimates of the parameters using the observation data. In practical situations the available data are discrete time series data sampled over some time interval, whereas SDEs are almost surely continuous processes; this introduces estimation problems. Thus, the parameter estimation for discretely observed diffusion processes is non-trivial and during the past decades it has generated a great deal of research effort (e.g. [16, 17, 18, 19, 20, 21, 22, 15, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]).

The vast majority of parameter estimation methods were developed for fully-observed diffusion processes (all the d coordinates of the multivariate X process are observed at each sampled time-point) driven by a Wiener process W, since in this case X is Markovian as a consequence of the fact that W is Markovian (section 4.6 in [1]), and thus the transition densities can be computed or at least numerically approximated.

The general framework is given by the following d-dimensional system of (Itô) SDEs

$$dX_t = f(t, X_t; \theta)dt + g(t, X_t; \theta)dW_t, \qquad X_0 = x_0, \quad t \geq 0 \tag{A.7}$$

where W is an m-dimensional standard Wiener process, $f : [0, +\infty) \times \mathbb{R}^d \times \Theta \to \mathbb{R}^d$ and $g : [0, +\infty) \times \mathbb{R}^d \times \Theta \to \mathbb{R}^{d \times d}$ are known functions depending on an unknown finite-dimensional parameter vector $\theta \in \Theta$. We assume that the initial value x_0 is deterministic and that $x_0, x_1, \ldots, x_n$ is a sequence of historical observations from the diffusion process X sampled at non-stochastic discrete time-points $t_0 < t_1 < \cdots < t_n$.

Since X is Markovian, the maximum likelihood estimator (MLE) of θ can be calculated if the transition densities $p(x_t; x_s, \theta)$ of X are known, $s < t$. The log-likelihood function of θ is given by (disregarding the deterministic initial value x_0)

$$l_n(\theta) = \sum_{i=1}^{n} \log p(x_i; x_{i-1}, \theta) \tag{A.8}$$

and the maximum likelihood estimator $\hat{\theta}$ can be found by maximizing (A.8) with respect to θ. Under mild regularity conditions, $\hat{\theta}$ is consistent, asymptotically normally distributed and asymptotically efficient as n tends to infinity [22].

The difficulty with the MLE is that the transition density function of the underlying diffusion process is often unknown. One response to this problem is to compute an approximation to the transition density function numerically, e.g.:

1. solving numerically the Kolmogorov partial differential equations satisfied by the transition density [34];
2. deriving a closed-form Hermite expansion to the transition density [16, 17];
3. simulating R times the process to Monte-Carlo integrate the transition density (e.g. [29, 21, 15, 26, 27, 28]): this methodology is known as *simulated maximum likelihood* (SML).

Recently a novel method using exact simulation was proposed [35]. Each of these techniques have been successfully implemented by the aforementioned authors, but each has their limitations. [36] notes that methods 1 and 3 above are computationally intense and poorly accurate. In response [15] build on their importance sampling ideas in order to improve the performance of Pedersen's (1995) (or equivalently Brandt and Santa-Clara's (2002)) method and point out that method 2 above, while accurate and fast, is only available for a small number of models.

Our opinion is that a method should be not only accurate and fast, but also "practicable", that is the simulation-based methods 3 above are highly time-consuming and have proved less accurate

than e.g. method 2 [37]. On the other hand, they are a very general tool, and have proved to be applicable over a wide range of SDE models. In general method 2 should be the tool of choice (but see [38] for some limitations), but computing the Hermite expansion of the transition density could be a very difficult task, especially if the SDE is multivariate and non-linear.

In any case a parameter estimation algorithm cannot work a miracle! The goodness of the estimation results strongly depends on factors like the number of available observations n (the larger the better), the number of simulations R (the larger the better), the stepsize h (the smaller the better), the appropriateness of the initial guess for the starting value of θ in the optimization procedure. Thus, before trying to estimate the parameters of an SDE using real data, it is strongly suggested to plot trajectories of the SDE by plugging some values for the parameters and see how the trajectories behave with respect to the data. Once an appropriate set of parameters has been found, it can be used as starting value for the estimation algorithm.

A.6.1 A Non-Parametric Method

A non-parametric simulated maximum likelihood approach is considered in [27] for the case of a one-dimensional Itô SDE, but it can be applied to Stratonovich SDEs also; here we suggest some modifications with respect to the original algorithm and extend the application to multidimensional *fully observed* SDEs. Let $p(t_i, x_i; (t_{i-1}, x_{i-1}), \theta)$ be the transition density of x_i starting from x_{i-1} and evolving to x_i. Then the maximum likelihood estimate of θ will be given by the value maximizing the function

$$L(\theta) = \prod_{i=1}^{n} p(t_i, x_i; (t_{i-1}, x_{i-1}), \theta)$$

w.r.t. θ. In practice $L(\theta)$ will be approximated through Monte Carlo simulations according to the following algorithm:

1. consider the time interval $[t_{i-1}, t_i]$ and divide it into M subintervals of length $h = (t_i - t_{i-1})/M$: then equation (A.7) is integrated on this discretization by using a standard algorithm (e.g. Euler-Maruyama, Milstein) by taking x_{i-1} at time t_{i-1} as the starting value, thus obtaining an approximation of X at t_i. This integration is repeated R times, thereby generating R approximations of the X process at time t_i starting from x_{i-1} at t_{i-1}. We denote such values with $X_{t_i}^1, ..., X_{t_i}^R$, i.e. $X_{t_i}^r$ is the integrated value of (A.7) at t_i starting from x_{i-1} at t_{i-1} in the rth simulation ($r = 1, \ldots, R$);

2. the simulated values $X_{t_i}^1, \ldots, X_{t_i}^R$ are used to construct a non-parametric kernel density estimate of the transition density $p(t_i, x_i; (t_{i-1}, x_{i-1}), \theta)$

$$p^R(t_i, x_i; t_{i-1}, x_{i-1}, \theta) = \frac{1}{Rh_i} \sum_{r=1}^{R} K\left(\frac{x_i - X_{t_i}^r}{h_i}\right)$$

   where h_i is the kernel *bandwidth* at time t_i and $K(\cdot)$ is a suitable symmetric, non-negative kernel function enclosing unit mass;

3. the previous procedure is repeated for each x_i and the $p^R(t_i, x_i; t_{i-1}, x_{i-1}, \theta)$ thus obtained used to construct $L^R(\theta) = \prod_{i=1}^{n} p^R(t_i, x_i; (t_{i-1}, x_{i-1}), \theta)$;

4. $L^R(\theta)$ is maximized w.r.t. θ to obtain the approximated MLE θ^R of θ.

Note that the correct construction of $L^R(\cdot)$ requires that the Wiener increments, once created, are kept *fixed* for a given optimization procedure. A suitable choice of $K(\cdot)$ is given by the normal kernel

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp(-u^2/2)$$

with bandwidth given by (see [39])

$$h_i = (4/3)^{1/5} s_i R^{-1/5}, \qquad i = 1, ..., n$$

where s_i is the sample standard deviation of the data presented to the kernel at time t_i, i.e. s_i is computed on $X^1_{t_i}, \ldots, X^R_{t_i}$. Notice that, for numerical reasons, it is normally more convenient to minimize the negative log-likelihood function; thus SDE Toolbox minimizes

$$-\log L^R(\theta) = -\sum_{i=1}^{n} \log p^R(t_i, x_i; (t_{i-1}, x_{i-1}), \theta)$$

and the approximated MLE is given by $\theta^R = \arg\min_\theta - \log(L^R(\theta))$.

In the case of multi-dimensional SDEs the procedure can be straightforwardly extended, e.g. by assuming the d system variables of the process to be pairwise *independent* (not just uncorrelated[1]). In this case the multidimensional kernel density estimator can be expressed as the product of the kernels of each variable and the bandwidth for the generic dimension k of the SDE is given by

$$h_{i,k} = (4/(d+2))^{1/(d+4)} s_{i,k} R^{-1/(d+4)}, \qquad i = 1, ..., n; \quad k = 1, ..., d.$$

See section 6.3.1 in [39] for further details.

A.6.2 A Parametric Method

In the previous section a non-parametric estimation technique has been considered. That approach suffers from the usual problems with non-parametric density estimation: a slow convergence rate (in the number of simulations) and the curse of dimensionality, i.e. as the number of variables increases, the convergence rate of most nonparametric estimators to their asymptotic distribution deteriorates exponentially [21]. However the non-parametric method can be applied to estimate either Itô and Stratonovich SDEs using the Euler-Maruyama and the Milstein approximation schemes; on the other side the parametric approach described here can be applied only to Itô SDEs using only the Euler-Maruyama discretization. This means that, since the Euler-Maruyama integration scheme is not defined for Stratonovich SDEs, the non-parametric strategy is the only method available in the Toolbox for the estimation of Stratonovich SDEs.

The simulated maximum likelihood method considered here was proposed in [40] and independently in [29] (see also [21]) and was refined in [15] using *importance sampling* techniques. For ease of implementation here we consider the original version. Consider a d-dimensional fully observed SDE, then $L(\theta)$ is approximated through Monte Carlo simulations according to the following algorithm:

1. consider the time interval $[t_{i-1}, t_i]$ and divide it into M subintervals of length $h = (t_i - t_{i-1})/M$: then equation (A.7) is integrated on the discretization $\{t_{i-1}, t_{i-1} + h, t_{i-1} + 2h, ..., t_{i-1} + (M-1)h\}$ by using a standard algorithm (e.g. Euler-Maruyama, Milstein) and taking x_{i-1} at time t_{i-1} as the starting value, thus obtaining an approximation of X at $t_{i-1} + (M-1)h$. This integration is repeated R times, thereby generating R approximations of the X process at time $t_{i-1} + (M-1)h$ starting from x_{i-1} at t_{i-1}. For ease of notation we denote such values with $X^1_{t_{i-1}}, ..., X^R_{t_{i-1}}$, i.e. $X^r_{t_{i-1}}$ is the integrated value of (A.7) at time $t_{i-1} + (M-1)h$ starting from x_{i-1} at t_{i-1} in the rth simulation ($r = 1, \ldots, R$);

2. the simulated values $X^1_{t_{i-1}}, \ldots, X^R_{t_{i-1}}$ are used to construct an estimate of the transition density $p(t_i, x_i; (t_{i-1}, x_{i-1}), \theta)$ given by

$$p^R\big(t_i, x_i; t_{i-1}, x_{i-1}, \theta\big) = \frac{1}{R} \sum_{r=1}^{R} \phi(x_i; mean_i^R, variance_i^R)$$

where

$$\begin{aligned} mean_i^R &= X^r_{t_{i-1}} + h \cdot f(t_{i-1} + (M-1)h, X^r_{t_{i-1}}; \theta), \\ variance_i^R &= h \cdot \Sigma(t_{i-1} + (M-1)h, X^r_{t_{i-1}}; \theta), \end{aligned}$$

[1] Future versions of the SDE Toolbox may consider the general case.

$\phi(x;\cdot,\cdot)$ denoting the multivariate normal density at x and $\Sigma(t,x;\theta) = g(t,x;\theta)g(t,x;\theta)^T$, where ^{T} denotes transposition;

3. the previous procedure is repeated for each x_i and the $p^R(t_i,x_i;t_{i-1},x_{i-1},\theta)$ thus obtained used to construct $L^R(\theta) = \prod_{i=1}^n p^R(t_i,x_i;(t_{i-1},x_{i-1}),\theta)$;

4. $-\log L^R(\theta)$ is minimized w.r.t. θ to obtain the approximated MLE θ^R of θ.

Under mild regularity conditions [21] θ^R converge to the MLE of θ as $M \to \infty$ and $R \to \infty$, with $R^{1/2}/M \to 0$.

In [21] some practical considerations for the choice of M and R in the financial framework are suggested: e.g. "[...] for fairly persistent daily or weekly data, an M of five to ten is sufficient to capture the shape of the transition densities for reasonably well-behaved univariate and multivariate diffusions. Regarding the choice of R, our experience with the estimator suggests that even for a four-dimensional diffusion, 2000-5000 simulations are sufficient".

A.6.3 Asymptotic 95% Confidence Intervals

The parameter estimation methods considered in the Toolbox are based on the maximization of an approximation of the likelihood function. Thus, the obtained (approximated) maximum likelihood estimates θ^R of the free to vary parameters $\tilde{\theta} \subseteq \theta$ (here θ denotes the vector containing either the free and the constant parameters), for R and M "large enough", should be asymptotically normally distributed as $n \to \infty$ with mean $\tilde{\theta}$ and variance given by the inverse of the *expected* Fisher information matrix [22]. The latter is often unknown, thus we considered the *observed* Fisher information in place of the expected Fisher information, since it often makes little difference numerically (e.g. [41], p. 133). The observed Fisher information at θ^R is given by $-H(\theta^R)$, where $H(\theta^R)$ is the Hessian matrix of the log-likelihood function $l(\theta^R)$, that is the matrix of the second derivatives of $l(\tilde{\theta})|_{\theta^R}$ with respect to $\tilde{\theta}$. The Hessian is computed using the central approximation. By denoting with $\tilde{H}(\theta^R)$ such approximation and with $\tilde{I}(\theta^R) = -\tilde{H}(\theta^R)$ the corresponding approximation of the observed Fisher information, the asymptotic 95% confidence interval for $\tilde{\theta}_p$ is approximated by $[\theta_p^R - 1.96 \cdot \tilde{I}_p^{-1/2}, \theta_p^R + 1.96 \cdot \tilde{I}_p^{-1/2}]$ where θ_p^R is the p-th element of θ^R and $\tilde{I}_p$ is the p-th element on the diagonal of $\tilde{I}(\theta^R)$.

Bibliography

[1] P.E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*. Springer, 1992.

[2] P.E. Kloeden, E. Platen, and H. Schurz. *Numerical solution of SDE through computer experiments*. Springer, 1994.

[3] D.J. Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM J. Numer. Anal.*, 43(3):525–546, 2001. `http://www.caam.rice.edu/~cox/stoch/dhigham.pdf`.

[4] K. Burrage and P.M. Burrage. High strong order explicit Runge–Kutta methods for stochastic ordinary differential equations. *Applied Numer. Mathematics*, 22:81–101, 1996.

[5] P.M. Burrage and K. Burrage. A variable stepsize implementation for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(3):848–864, 2002.

[6] P. M. Burrage. *Runge–Kutta methods for stochastic differential equations*. PhD thesis, Department of Mathematics, University of Queensland (Australia), 1999.

[7] A. Rößler. *Runge–Kutta methods for the numerical solution of stochastic differential equations*. PhD thesis, Department of Mathematics, University of Darmstadt, 2003.

[8] W. Rümelin. Numerical treatment of stochastic differential equations. *SIAM J. Numer. Anal.*, 19:604–613, 1982.

[9] I. Karatzas and S.E. Shreve. *Brownian motion and stochastic calculus*. Springer-Verlag, 1991.

[10] B. Øksendal. *Stochastic differential equations: an introduction with applications*. Springer, second edition, 2000.

[11] L.C.G. Rogers and D. Williams. *Diffusions, Markov processes and martingales. Volume 2: Itô calculus*. John Wiley & Sons, 1987.

[12] L.C.G. Rogers and D. Williams. *Diffusions, Markov processes and martingales. Volume 1: Foundations*. John Wiley & Sons, second edition, 1994.

[13] D.W. Stroock and S.R.S. Varadhan. *Multidimensional Diffusion Processes*. Springer-Verlag, 1979.

[14] G. Marsaglia and W.W. Tsang. The ziggurat method for generating random variables. *Journal of Statistical Software*, 5(8):1–7, October 2000.

[15] G.B. Durham and A.R. Gallant. Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business and Economic Statistics*, 20(3):297–316, July 2002.

[16] Y. Aït-Sahalia. Closed-form likelihood expansion for multivariate diffusions. Technical report, Princeton University, 2001. Forthcoming on Annals of Statistics.

[17] Y. Aït-Sahalia. Maximum likelihood estimation of discretely sampled diffusions: a closed-form approximation approach. *Econometrica*, 70(1):223–262, January 2002.

[18] J. Alcock and K. Burrage. A genetic estimation algorithm for parameters of stochastic ordinary differential equations. *Computational Statistics & Data Analysis*, 47:255–275, 2004.

[19] B.M. Bibby, M. Jacobsen, and M. Sørensen. Estimating functions for discretely sampled diffusion-type models. In Y. Aït-Sahalia and L.P. Hansen, editors, *Handbook of Financial Econometrics*. Amsterdam: North-Holland, 2005.

[20] B.M. Bibby and M. Sørensen. Martingale estimation functions for discretely observed diffusion processes. *Bernoulli*, 1(1/2):17–39, 1995.

[21] M.W. Brandt and P. Santa-Clara. Simulated likelihood estimation of diffusions with an application to exchange rate dynamics in incomplete markets. *Journal of Financial Economics*, 63:161–210, 2002.

[22] D. Dacunha-Castelle and D. Florens-Zmirnou. Estimation of the coefficients of a diffusion from discrete observations. *Stochastics*, 19:263–284, 1986.

[23] O. Elerian, N. Shephard, and S. Chib. Likelihood inference for discretely observed nonlinear diffusions. *Econometrica*, 69:959–993, 2001.

[24] A. Gallant and J. Long. Estimating stochastic differential equations efficiently by minimum chi-square. *Biometrika*, 84:124–141, 1997.

[25] C. Gouriéroux, A. Monfort, and E. Renault. Indirect inference. *Journal of Applied Econometrics*, 8:85–118, 1993.

[26] A.S. Hurn and K.A. Lindsay. Estimating the parameters of stochastic differential equations. *Mathematics And Computers In Simulation*, 48:373–384, 1999.

[27] A.S. Hurn, K.A. Lindsay, and V.L. Martin. On the efficacy of simulated maximum likelihood for estimating the parameters of stochastic differential equations. *Journal of Time Series Analysis*, 24(1):45–63, 2003.

[28] J. Nicolau. A new technique for simulating the likelihood of stochastic differential equations. *Econometrics Journal*, 5:91–103, 2002.

[29] A.R. Pedersen. A new approach to maximum likelihood estimation for stochastic differential equations based on discrete observations. *Scand. J. Statist.*, 22(1):55–71, 1995.

[30] I. Shoji and T. Ozaki. Estimation for nonlinear stochastic differential equations by a local linearization method. *Stochastic Analysis and Applications*, 16:733–752, 1998.

[31] H. Sørensen. *Inference for diffusion processes and stochastic volatility models*. PhD thesis, University of Copenaghen, September 2000.

[32] M. Sørensen. Prediction-based estimating functions. *Econometrics Journal*, 3:123–147, 2000.

[33] N. Yoshida. Estimation of diffusion processes from discrete observations. *Journal of Multivariate Analysis*, 41:220–242, 1992.

[34] A. Lo. Maximum likelihood estimation of generalized Ito processes with discretely-sample data. *Econometric Theory*, 4:231–247, 1988.

[35] A. Beskos, O. Papaspiliopoulos, G.O. Roberts, and P. Fearnhead. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes. *J. R. Statist. Soc. B*, 68:333–382, 2006.

[36] Y. Aït-Sahalia. Comment on G. Durham, A. Gallant, Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *J. Bus. Econom. Statist.*, 20:317–321, 2002.

[37] B. Jensen and R. Poulsen. Transition densities of diffusion processes: numerical comparison of approximation techniques. *Journal of Derivatives*, 9:1–15, 2002.

[38] O. Stramer and J. Yan. On simulated likelihood of discretely observed diffusion processes and comparison to closed-form approximation. *Journal of Computational & Graphical Statistics*, 16(3):672–691, September 2007.

[39] D.W. Scott. *Multivariate density estimation*. Wiley & Sons, 1992.

[40] P. Santa-Clara. *Simulated likelihood estimation of diffusions with an application to the short term interest rate*. PhD thesis, INSEAD, 1995.

[41] O.E. Barndorff-Nielsen and M. Sørensen. A review of some aspects of asymptotic likelihood theory for stochastic processes. *Int. Stat. Rev.*, 62(1):133–165, 1994.