1 MAPLE for Stochastic Differential Equations

S. Cyganowski[1], L. Grüne[2], and P.E. Kloeden[2]

[1] Trinity College, Royal Parade, Parkville 3052, Australia,
e-mail: scyganowski@trinity.unimelb.edu.au
[2] Fachbereich Mathematik, Johann Wolfgang Goethe-Universität,
D–60054 Frankfurt am Main, Germany,
e-mail: gruene, kloeden@math.uni-frankfurt.de

Abstract. This chapter introduces the MAPLE software package **stochastic** consisting of MAPLE routines for stochastic calculus and stochastic differential equations and for constructing basic numerical methods for specific stochastic differential equations, with simple examples illustrating the use of the routines. A website address is given from which the software can be downloaded and where up to date information about installment, new developments and literature can be found.

1.1 Introduction

Stochastic calculus and stochastic differential equations (SDEs), and in particular numerical schemes for SDEs, provide an ideal context for the use of symbolic manipulator software [4,5,8,11,18,22,23]. These issues were mentioned briefly by the third co-author in his lectures at the summer school in Durham. A copy of the overhead projector transparencies of these lectures can be downloaded from

`http://www.math.uni-frankfurt.de/~numerik/kloeden`

Since extensive expositions of the contents of these lectures are readily available in the text books [13,16] as well as in review articles [14,15,17,21] and other books [1,2,9,19], the details will not be repeated here. Instead this chapter will focus on the MAPLE software package **stochastic** that consists of MAPLE routines for stochastic calculus, stochastic differential equations (SDEs) and numerical methods for SDEs.

The presentation is not self contained. It is assumed that the reader will consult the above references for the necessary background on stochastic calculus as well as suitable MAPLE textbooks and handbooks. Some historical remarks on SDEs will given in the next section, followed by a brief introduction to the **stochastic** package in Section 3 and then, in the subsequent sections, the procedures of this package will be presented and illustrated through simple examples.

1.2 Stochastic Differential Equations

Following Einstein's explanation of physically observed Brownian motion during the first decade of the 1900s, the physicists Langevin and Smoluchowski, and others, attempted to model the dynamics of such motion in terms of differential equations. Instead of a deterministic ordinary differential equation

$$\frac{dx}{dt} = a(t, x)$$

they obtained a noisy differential equation of the form

$$\frac{d}{dt}X_t = a(t, X_t) + b(t, X_t)\,\xi_t \tag{1.1}$$

with a deterministic or averaged drift term $a(t, X_t)$ perturbed by a noise intensity term $b(t, X_t)\,\xi_t$, where the ξ_t are independent $N(0, 1)$–distributed Gaussian random variables for each t and $b(t, x)$ is an intensity factor that may, in general, depend on both the t and x variables. (Note a subscript will be used to denote the time dependence of a stochastic process, i.e. X_t will be written instead of $X(t)$).

The driving process ξ_t, which is called *Gaussian white noise,* appears formally to be the pathwise derivative of a mathematical Brownian motion or Wiener process W_t, a Gaussian process with $W_0 = 0$ and $N(0, t)$–distributed W_t for each $t \geq 0$, i.e. with

$$E\,(W_t) = 0, \quad E\left((W_t)^2\right) = t,$$

which has independent increments

$$E\,((W_{t_4} - W_{t_3})\,(W_{t_2} - W_{t_1})) = 0$$

for all $0 \leq t_1 < t_2 \leq t_3 < t_4$.

However, the Gaussian white noise process is not a conventional process, having, for example, covariance equal to a constant multiple of the Dirac delta function. Moreover, it is now known that the sample paths of a Wiener process W_t are nowhere differentiable. This suggests that equation (1.1), which might be written symbolically in terms of differentials as

$$dX_t = a(t, X_t)\,dt + b(t, X_t)\,dW_t,$$

should be interpreted in some sense as an integral equation

$$X_t = X_{t_0} + \int_{t_0}^{t} a(s, X_s)\,ds + \int_{t_0}^{t} b(s, X_s)\,dW_s. \tag{1.2}$$

The first integral here is just pathwise an ordinary Riemann integral. While it might seem that the second integral could be a Riemann–Stieltjes integral for

each sample path, this is not possible because the sample paths of a Wiener process are not just not differentiable, but not even of bounded variation on any bounded time interval. In the 1940s the Japanese mathematician K. Ito proposed a means to overcome this difficulty with the definition of new type of integral, a stochastic integral which is now called an *Ito stochastic integral.* Later, in the 1960s, the Russian physicist R.L. Stratonovich proposed another kind of stochastic integral, now called the *Stratonovich stochastic integral,* which is distinguished from the Ito integral by a "$\circ$" before the differential dW_t, i.e. written symbolically in the differential form

$$dX_t = a(t, X_t)\, dt + b(t, X_t) \circ dW_t,$$

but really an integral equation,

$$X_t = X_{t_0} + \int_{t_0}^{t} a(s, X_s)\, ds + \int_{t_0}^{t} b(s, X_s) \circ dW_s. \tag{1.3}$$

There are thus two types of stochastic calculus, the Ito stochastic calculus and the Stratonovich stochastic calculus depending on the type of stochastic integral used. Both have their advantages as well as their disadvantages. Which one should be used is more a modelling than mathematical issue, but once one has been chosen a corresponding equation of the other type with the same solutions can be determined, so it is possible to switch between the two stochastic calculi. The reader is referred to text books on stochastic calculus and stochastic differential equations for the appropriate definitions and mathematical development, e.g. [1,9,13].

1.2.1 Terminology

An N-dimensional Ito stochastic differential equation with an M-dimensional Wiener process $W_t = (W_t^1, \ldots, W_t^M)$, i.e. with M pairwise independent scalar Wiener processes W_t^1, ..., W_t^M as its components, will be written in vector form as

$$dX_t = a(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t)\, dW_t^j, \tag{1.4}$$

where $X_t = (X_t^1, \ldots, X_t^N)$, and componentwise as

$$dX_t^i = a^i(t, X_t)\, dt + \sum_{j=1}^{M} b^{i,j}(t, X_t)\, dW_t^j, \quad i = 1, \ldots, N. \tag{1.5}$$

Note that superscripts are being used for for the indices of vectors and matrices here. Also, the indexing of the $b^{i,j}$ appears to be the reverse of what may have been expected, with this term representing the ith component of the column vector b^j. In fact, $b^{i,j}$ written in this way is the (i, j)th component

of the $N \times M$-matrix $B = [b^1|\cdots|b^M]$ with b^j as its jth column vector. The vector SDE (1.4) could thus have written as

$$dX_t = a(t, X_t)\, dt + B(t, X_t)\, dW_t.$$

Similar notation will be used for Stratonovich SDE.

1.3 The MAPLE software package "stochastic"

The MAPLE software package **stochastic** consisting of MAPLE routines for stochastic calculus and SDEs was developed by the first co-author, initially as an honours degree thesis supervised by the third co-author, with additional routines being added later by the three co-authors and Thomas Pohl from time to time as they were required. The basic routines will be described here. (The technical report [8] is a preliminary version of this article). All of the software and copies of the cited papers of the co-authors can be downloaded from

`http://www.math.uni-frankfurt.de/~numerik/maplestoch`

The MAPLE software package **stochastic** is known to work for MAPLE V, Release 5.1 for Windows, Unix and Macintosh and MAPLE 6 for Unix. An older version with a reduced number of routines is available for MAPLE V, Release 3 for Windows, Unix and Macintosh. Information on downloading and installing the package can be found under the above link, as well as a MAPLE-worksheet containing all the examples in this paper.
The stochastic package contains routines useful for finding explicit solutions of SDEs and routines for constructing numerical schemes up to strong order 2.0 and weak order 3.0. Additional features include a routine that converts an SDE from Ito to Stratonovich form and vice versa, a routine that converts an SDE with white noise into the corresponding SDE with coloured noise, and routines that check whether an SDE has commutative noise of the first or second kinds. Other useful routines are also available, in particular procedures for the partial differential operators $L0$ and LJ that arise in the Ito (stochastic chain rule) formula with which users can easily construct numerical schemes other than those available here.

The functions available are

```
L0              SL0             LJ              MLJ             LFP
itoformula      chainrule       correct         conv            explicit
linearsde       reducible       colour          comm1           comm2
linearize       momenteqn       position        sphere

Euler           Milstein        milcomm         Taylor1hlf      Taylor2
wkeuler         wktay2          wktay3

pa      ap      bpb             pmatrix2pvector     pvector2pmatrix
```

For most of the functions you can type the command `?<function>` for more information.

To use a stochastic function, either define that function alone using the command `with(stochastic, <function>)`, or define all stochastic functions using the command `with(stochastic)`. Alternatively, invoke the function using the long form `stochastic[<function>]`. This long form notation is necessary whenever there is a conflict between a package function name and another function used in the same session.

As an example, to find the explicit solution of a scalar SDE with drift coefficient $a(t,x) = 1/2a^2x$ and diffusion coefficient $b(t,x) = ax$ use

```
> with(stochastic,explicit);  explicit(1/2*a^2*x,a*x);
```

1.3.1 MAPLE-terminology

Recall the componentwise notation (1.5) of the stochastic differential equation

$$dX_t^i = a^i(t, X_t)\,dt + \sum_{j=1}^{M} b^{i,j}(t, X_t)\,dW_t^j, \quad i = 1, \ldots, N,$$

which in matrix notation reads

$$dX_t = a(t, X_t)\,dt + B(t, X_t)\,dW_t,$$

where

$$a = \begin{pmatrix} a^1 \\ \vdots \\ a^N \end{pmatrix} \quad \text{and} \quad B = [b^1|\cdots|b^M] = \begin{pmatrix} b^{1,1} & \cdots & b^{1,M} \\ \vdots & \ddots & \vdots \\ b^{N,1} & \cdots & b^{N,M} \end{pmatrix}.$$

In the MAPLE stochastic package such an equation is represented by the drift coefficient vector a and the diffusion coefficient matrix B written, e.g. for $N = 2$ and $M = 3$ as `[a1,a2]` and `[[b11, b12, b13],[b21, b22, b23]]`. Deviations from this rule are explicitly stated in the description of the respective routine.

1.4 Ito Stochastic Calculus

Consider the N-dimensional Ito SDE (1.4) with an M-dimensional Wiener process $W_t = (W_t^1, \cdots, W_t^M)$ written componentwise as

$$dX_t^i = a^i(t, X_t)\,dt + \sum_{j=1}^{M} b^{i,j}(t, X_t)\,dW_t^j, \qquad i = 1, \ldots, N, \tag{1.6}$$

and define differential operators L^0, L^1, ..., L^M with respect to this SDE by

$$L^0 = \frac{\partial}{\partial t} + \sum_{k=1}^{N} a^k \frac{\partial}{\partial x^k} + \frac{1}{2} \sum_{k,l=1}^{N} \sum_{j=1}^{M} b^{k,j} b^{l,j} \frac{\partial^2}{\partial x^k \partial x^l} \tag{1.7}$$

and

$$L^j = \sum_{k=1}^{N} b^{k,j} \frac{\partial}{\partial x^k}, \qquad j = 1, \ldots, M. \tag{1.8}$$

These operators play a fundamental role in Ito stochastic calculus through the Ito formula for the stochastic chain rule and subsequently for stochastic Taylor expansions and numerical schemes for the SDE that are based on stochastic Taylor expansions. They are ideally suited to implementation with MAPLE.

1.4.1 Partial differential operators

Separate routines will be given for the operator L^0 and for the operators L^j with $j = 1, \ldots, M$, and then a combined routine with $j = 0$ as well as $j = 1, \ldots, M$.

L0 Operator routine The routine `stochastic[L0]`, which applies the partial differential operator L^0 defined by (1.7), is given by

```
stochastic[L0]:=proc(X::algebraic,a::list(algebraic),
                                  b::list(list(algebraic)))
local part1,part2,part3;
    part1 := diff(X,t);
    part2 := sum('a[k]*diff(X,x[k])','k' = 1 .. nops(a));
    part3 := 1/2*sum(
      'sum('sum('op(j,op(k,b))*op(j,op(l,b))*diff(X,x[k],x[l])',
        'j' = 1 .. nops(op(1,b)))','k' = 1 .. nops(a))',
                                   'l' = 1 .. nops(a));
    part1+part2+part3;
end:
```

The call `L0(X, [a1,..,aN], [[b11,..,b1M],..,[bN1,..,bNM]]);` applies the partial differential operator L^0 to a scalar valued function X (which is arbitrary, not necessarily the solution of the SDE (1.6)). The output variables are consistent with the variables used as input following the terminology from Section 1.3.1.

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process

$$d \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ v \end{pmatrix} dW_t^2,$$

where r, s, u and v are constants, that is with drift with components $a^1 = x_1$, $a^2 = x_2$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix}.$$

Apply the corresponding operator L^0 to the function $X(t, x_1, x_2) = x_2$.

```
> L0(x[2],[x[1],x[2]],[[r,u],[s,v]]);

                    x[2]
```

The result is $L^0 X(t, x_1, x_2) = x_2$.

LJ Operator routine The routine `stochastic[LJ]` for the partial differential operator L^j defined by (1.8), is given by

```
stochastic[LJ]:=proc(X::algebraic,b::list(list(algebraic)),j::integer)
      sum('op(j,op(k,b))*diff(X,x[k])','k' = 1 .. nops(b))
end:
```

The call `LJ(X,[[b11,..,b1M],..,[bN1,..,bNM]],j);` applies the partial differential operator L^j to a function X. Here $j = 1, \ldots, M$ denotes the "current" component of the Wiener process. The output variables are consistent with the variables used as input.

EXAMPLE: Consider the function $X(t, x_1, x_2) = x_2$ and a 2-dimensional SDE driven by a 2-dimensional Wiener process with diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where r, s, u and v are constants.

```
> LJ(x[2],[[r,u],[s,v]],2);

                    v

> LJ(x[2],[[r,u],[s,v]],1);

                    s
```

The results are thus $L^2 X(t, x_1, x_2) = v$ and $L^1 X(t, x_1, x_2) = s$

Combined MLJ Operator routine The routine `stochastic[MLJ]` applies one of the partial differential operators, $L0$ or LJ to a function X, thus combining the routines `stochastic[L0]` and `stochastic[LJ]` in a single routine through an appropriate choice of index j.

```
stochastic[MLJ]:=proc(X::algebraic, a::list(algebraic),
                      b::list(list(algebraic)), j::integer)
local flag;
    flag := 0;
    if j = 0 then flag := L0(X,a,b) fi;
    if flag = 0 then flag := sum('op(j,op(k,b))*diff(X,x[k])',
                                 'k' = 1 .. nops(b)) fi;
    RETURN(flag)
end:
```

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^2 \\ X_t^1 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ v \end{pmatrix} dW_t^2,$$

where r, s, u and v are constants, that is with drift with components $a^1 = x_2$, $a^2 = x_1$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix}.$$

Applying the corresponding operators L^2 and L^0 to the function $X(t, x_1, x_2) = x_2$ produces

```
> MLJ(x[2],[x[2],x[1]],[[r,u],[s,v]],2);

                v

> MLJ(x[2],[x[2],x[1]],[[r,u],[s,v]],0);

                x[1]
```

which means $L^2X(t, x_1, x_2) = v$ and $L^0X(t, x_1, x_2) = x_1$.

1.4.2 Ito Formula

For a sufficiently smooth transformation $U : [0, T] \times \mathbb{R}^N \to \mathbb{R}$ of the solution X_t of the Ito SDE

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

the scalar process $Y_t = U(t, X_t)$ satisfies the a vector stochastic differential

$$dY_t = L^0 U(t, X)t)\, dt + \sum_{j=1}^{M} L^j U(t, X_t)\, dW_t^j.$$

where the operators L^0 and L^j were defined by (1.7) and (1.8), respectively. This is called the *Ito Formula* and is the chain rule for the Ito stochastic calculus. It differs from what might be expected from deterministic calculus by the presence of the second order term in the L^0 operator, which is essentially due to the fact that $E((\Delta W)^2) = \Delta t$ for the increment of a Wiener process over an interval of length Δt.

```
stochastic[itoformula]:=proc(U::list(algebraic),a::list(algebraic),
                             b::list(list(algebraic)))
local i,k,l0,lj,soln;
  for i from 1 to nops(U) do
    l0:=L0(U[i],a,b)*dt;
    lj:=0;
    for k from 1 to nops(b) do
      lj:=lj+LJ(U[i],b,k)*dW.i;
    od;
    soln[i]:=dX.i=l0 +lj;
  od;
  RETURN(eval(soln));
end:
```

EXAMPLE: Consider the function $Y_t = U(t, X_t) = X_t^2$ where X_t is a solution of the Ito SDE

$$dX_t = aX_t\, dt + bX_t\, dW_t$$

Then

```
> itoformula([(x[1])^2],[a*x[1]],[[b*x[1]]]);

  table([

                              2    2     2                2
        1 = (dX1 = (2 a x[1]  + b  x[1] ) dt + 2 b x[1]  dW1)

        ])
```

that is

$$dY_t = (2aY_t^2 + b^2 Y_t^2)\, dt + 2bY_t^2\, dW_t.$$

1.4.3 LFP Operator: Fokker–Planck Equation

The transition probabilities of the diffusion process solution of Ito SDE (1.6) have densities $p(s, x; t, y)$ which satisfy the *Fokker–Planck equation,*

$$\frac{\partial p}{\partial t} + \sum_{i=1}^{N} \frac{\partial}{\partial y_i} \left\{ a^i(t, y) p \right\} - \frac{1}{2} \sum_{k,l=1}^{N} \sum_{j=1}^{M} \frac{\partial^2}{\partial y_k \partial y_l} \left\{ b^{k,j}(t, y) b^{l,j}(t, y) p \right\} = 0 \quad (1.9)$$

(s, x in p fixed) with the initial condition

$$\lim_{t \downarrow s} p(s, x; t, y) = \delta(x - y),$$

where δ is the Dirac delta function on $\mathbb{R}^N$. The corresponding differential operator $\mathcal{L}^*$, which is in fact the adjoint of the $\mathcal{L} = L^0$ operator (1.7) is defined as

$$\mathcal{L}^* p \frac{\partial p}{\partial t} + \sum_{i=1}^{N} \frac{\partial}{\partial y_i} \left\{ a^i(t, y) p \right\} - \frac{1}{2} \sum_{k,l=1}^{N} \sum_{j=1}^{M} \frac{\partial^2}{\partial y_k \partial y_l} \left\{ b^{k,j}(t, y) b^{l,j}(t, y) p \right\}.$$

The MAPLE routine `LFP` for the operator $\mathcal{L}^*$ is given by

```
stochastic[LFP]:=proc(P::algebraic,a::list(algebraic),
                      b::list(list(algebraic)))
local part1,part2,part3;
  part1 := diff(P,t);
  part2 := sum('diff(a[k]*P,y[k])','k' = 1 .. nops(a));
  part3 := 1/2*sum(
    'sum('sum('diff(op(j,op(k,b))*op(j,op(l,b))*P,y[k],y[l])',
      'j' = 1 .. nops(op(1,b)))','k' = 1 .. nops(a))',
      'l' = 1 .. nops(a));
  part1+part2-part3;
end:
```

The call is `LFP(P,[a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);` where P is the transition density, and the ai and bij as usual denote the coefficients of the Ito SDE (1.6).

EXAMPLE: Consider the Ito SDE

$$dX_t = a\, dt + b\, dW_t,$$

where a and b are constant. To obtain the Fokker–Planck equation (1.9) for this Ito SDE call

```
> LFP(p(s,x,t,y[1],[a],[[b]]);
```

which gives

$$(\frac{\partial}{\partial t}\, \mathrm{p}(s,\, x,\, t,\, y_1)) + a\, (\frac{\partial}{\partial y_1}\, \mathrm{p}(s,\, x,\, t,\, y_1)) - \frac{1}{2}\, b^2\, (\frac{\partial^2}{\partial {y_1}^2}\, \mathrm{p}(s,\, x,\, t,\, y_1)) = 0.$$

On the other hand, to check that a density function

$$p(s, x, t, y) = \frac{1}{\sqrt{2\pi b^2 (t - s)}} e^{\frac{-(y - x - a*(t-s))^2}{2b^2(t-s)}}).$$

satisfies a particular Fokker–Planck equation call

```
> p:=(s,x,t,y)->1/(sqrt(2*pi*b^2*(t-s)))
                  *exp(-(y-x-a*(t-s))^2/(2*b^2*(t-s))):
> LFP(p(s,x,t,y[1]),[a],[[b]]);
```

The output of `LFP` is:

$$\frac{1}{4}\frac{\sqrt{2}\,\%2\,\pi\,b^2}{(\pi\,b^2\,(t-s))^{3/2}}+\frac{1}{2}\,\frac{\sqrt{2}\,(\dfrac{\%1\,a}{b^2\,(t-s)}+\dfrac{1}{2}\,\dfrac{\%1^2}{b^2\,(t-s)^2})\,\%2}{\sqrt{\pi\,b^2\,(t-s)}}$$
$$-\frac{1}{2}\,\frac{a\,\sqrt{2}\,\%1\,\%2}{\sqrt{\pi\,b^2\,(t-s)}\,b^2\,(t-s)}$$
$$+\frac{1}{4}\,\frac{\sqrt{2}\,\%2}{\sqrt{\pi\,b^2\,(t-s)}\,(t-s)}-\frac{1}{4}\,\frac{\sqrt{2}\,\%1^2\,\%2}{\sqrt{\pi\,b^2\,(t-s)}\,(t-s)^2\,b^2}$$
$$\%1:=y_1-x-a\,(t-s)$$
$$\%2:=e^{(-1/2\,\frac{\%1^2}{b^2\,(t-s)})}.$$

The MAPLE command

```
> simplify(%);
```

reduces this to 0, so this $p(s,x,t,y)$ satisfies the Fokker–Planck equation with the given coefficients.

1.5 Stratonovich Stochastic Calculus

The Stratonovich SDE

$$dX_t=\underline{a}(t,X_t)\,dt+\sum_{j=1}^{M}b^j(t,X_t)\circ dW_t^j \tag{1.10}$$

with the same solutions as the N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t=a(t,X_t)\,dt+\sum_{j=1}^{M}b^j(t,X_t)\,dW_t^j \tag{1.11}$$

has drift coefficient $\underline{a}$ that is defined given a componentwise by

$$\underline{a}^i(t,X)=a^i(t,X)-\frac{1}{2}\sum_{k=1}^{N}\sum_{j=1}^{M}b^{k,j}(t,X)\,\frac{\partial b^{i,j}}{\partial x_k}(t,X),\qquad i=1,\ldots,N, \tag{1.12}$$

whereas a given $\underline{a}$ is defined componentwise by

$$a^i(t,X) = \underline{a}^i(t,X) + \frac{1}{2}\sum_{k=1}^{N}\sum_{j=1}^{M} b^{k,j}(t,X)\,\frac{\partial b^{i,j}}{\partial x_k}(t,X), \qquad i = 1,\ldots,N. \tag{1.13}$$

These are called the *drift-correction* formulas. Note that the diffusion coefficients are the same in both the Ito and Stratonovich SDEs.

1.5.1 Ito-Stratonovich drift correction procedures

A procedure for the Ito to Stratonovich drift conversion (1.12) will be presented and then a combined procedure for conversion of the drift in either direction.

Ito to Stratonovich drift correction procedure

The routine `stochastic[correct]`, which converts the Ito drift coefficient a into the corresponding Stratonovich drift coefficient $\underline{a}$, is given by

```
stochastic[correct]:=proc(a::list(algebraic),
                          b::list(list(algebraic)),i)
    a[i]-1/2*sum('LJ(op(j,op(i,b)),b,j)','j' = 1 .. nops(op(1,b)));
end:
```

Stratonovich to Ito drift correction procedure

The call `correct([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]],i);` is used to convert the drift coefficient of the Ito SDE (1.11) into that of its Stratonovich form (1.10). The index $i = 1, \ldots, N$ denotes the "current" component of the SDE. The output variables are consistent with the variables used as input which follow the terminology of Section 1.3.1.
EXAMPLES: Consider two Ito SDEs with $N = M = 2$ having the same Ito drift vector

$$a^1(t,x_1,x_2) = x_1, \qquad a^2(t,x_1,x_2) = x_2,$$

and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$$

in the first case and the variable diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} x_1 \\ 0 \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$$

in the second case, where r is a constant in both cases.

```
> correct([x[1],x[2]],[[r,0],[0,r]],2);

                x[2]

> correct([x[1],x[2]],[[x[1],0],[0,r]],1);

                1/2 x[1]
```

The applications of the routine here produces the result $\underline{a}^1(t, x_1, x_2) = x_2$ in the first case and $\underline{a}^2(t, x_1, x_2) = \frac{1}{2}x_1$ in the second case.

Ito–Stratonovich drift correction: both directions

The next procedure combines the Stratonovich to Ito conversion formula (1.13) with the procedure for the Ito to Stratonovich conversion formula (1.12) of the last subsection.

```
stochastic[conv]:=proc(a::list(algebraic),b::list(list(algebraic)),
                                          c::algebraic)
local temp,i;
   if c=ito then
      for i from 1 to nops(a) do
          temp[i]:=op(i,a)-1/2*sum('sum('op(k,op(j,b))
             *diff(op(k,op(i,b)),x[j])',
               'k'=1..nops(op(1,b)))','j'=1..nops(a));
      od;
   elif c=strat then
       for i from 1 to nops(a) do
          temp[i]:=op(i,a)+1/2*sum('sum('op(k,op(j,b))
             *diff(op(k,op(i,b)),x[j])',
               'k'=1..nops(op(1,b)))','j'=1..nops(a));
       od;
   else
      ERROR(`Must enter either ito or strat for the 3rd argument`)
   fi;
   RETURN(map(simplify,eval(temp)))
end:
```

EXAMPLE: Consider the Ito SDE

$$dX_t = -a^2 X_t(1 - X_t^2)\,dt + a(1 - X_t^2)\,dW_t$$

To derive the Stratonovich SDE apply

```
> conv([-a^2*x[1](1-x[1]^2)],[[a*(1-x[1]^2)]],ito);
```

and obtain

```
table([
 1 = 0
 ]),
```

which means that the desired Stratonovich SDE is

$$dX_t = 0\,dt + a(1 - X_t^2) \circ dW_t = a(1 - X_t^2) \circ dW_t.$$

The other direction gives the original Ito SDE back.

```
> conv([0],[[a*(1-x[1]^2)]],strat);
```

$$\begin{array}{l} \text{table([} \\ \quad 1 = -a^2\, x_1\, (-1 + {x_1}^2) \\ \text{])}. \end{array}$$

1.5.2 Stratonovich L0 operator

The L^0 operator of Ito calculus needs to be changed in Stratonovich calculus to

$$\underline{L}^0 = \frac{\partial}{\partial t} + \sum_{i=1}^{N} \underline{a}^i \frac{\partial}{\partial x^i}, \tag{1.14}$$

while the L^j operators of Ito calculus remain unchanged in Stratonovich calculus. The Stratonovich operator $\underline{L}^0$ applied to a scalar valued function X is produced by the routine `stochastic[SL0]` given by

```
stochastic[SL0]:=proc(X:algebraic,a:list(algebraic),
                      b:list(list(algebraic)))
local part1,part2;
    part1 := diff(X,t);
    part2 := sum('a[k]*diff(X,x[k])','k' = 1 .. nops(a));
    part1 + part2;
end:
```

The call `SL0(X, [a1,..,aN], [[b11,..,b1M],..,[bN1,..,bNM]]);` computes the application of the Stratonovich version of the operator $L0$ to a scalar valued function X.

EXAMPLE: Compute $\underline{L}^0 X$ for the function $X(t, x_1, x_2) = x_2$ and the 2-dimensional Stratonovich SDE with drift components

$$\underline{a}^1(t, x_1, x_2) = x_1, \qquad \underline{a}^2(t, x_1, x_2) = x_2,$$

and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where r is a constant, i.e. apply

```
> SL0(x[2],[x[1],x[2]],[[r,u],[s,w]]);

                        x[2]
```

giving the result $\underline{L}^0 X(t, x_1, x_2) = x_2$.

1.5.3 Stratonovich chain rule transformation

For a sufficiently smooth transformation $U : [0, T] \times \mathbb{R}^N \to \mathbb{R}$ of the solution X_t of the Stratonovich SDE

$$dX_t = \underline{a}(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t) \circ dW_t^j$$

the scalar process $Y_t = U(t, X_t)$ satisfies the vector Stratonovich stochastic differential

$$dY_t = \left(\frac{\partial U}{\partial t} + \sum_{i=1}^{N} \underline{a}^i \frac{\partial U}{\partial x_i} \right) dt + \sum_{i=1}^{N} \sum_{j=1}^{M} b^{i,j} \frac{\partial U}{\partial x_i} \circ dW_t^j \tag{1.15}$$

where the terms are all evaluated at (t, X_t). (This expression corresponds to what would be obtained if the rules of deterministic calculus were valid here). In operator form this is

$$dY_t = \underline{L}^0 U(t, X)t)\, dt + \sum_{j=1}^{M} L^j U(t, X_t) \circ dW_t^j.$$

It is computed by the procedure

```
stochastic[chainrule]:=proc(U::list(algebraic),a::list(algebraic),
                            b::list(list(algebraic)))
  local i,k,l0,lj,soln;
  for i from 1 to nops(U) do
    l0:=SL0(U[i],a,b)*dt;
    lj:=0;
    for k from 1 to nops(b) do
      lj:=lj+LJ(U[i],b,k)*odW.i;
    od;
    soln[i]:=dX.i=l0 +lj;
  od;
  RETURN(eval(soln));
end:
```

EXAMPLE: Consider the function $U(t, X_t) = X_t^2$ and the Stratonovich SDE

$$dX_t = aX_t\, dt + bX_t \circ dW_t.$$

Then

```
> chainrule([(x[1])^2],[a*x[1]],[[b*x[1]]]);

  table([

                              2                  2
       1 = (dX1 = 2 a x[1]  dt + 2 b x[1]  odW1)

       ])
```

which corresponds to the Stratonovich SDE

$$dY_t = 2aY_t^2\,dt + 2bY_t^2 \circ dW_t.$$

1.6 Explicitly Solvable Scalar SDEs

Several classes of scalar Ito SDEs

$$dX_t = a(t, X_t)\,dt + b(t, X_t)\,dW_t \tag{1.16}$$

that can be solved explicitly will be considered in this Section. It begins with a general linear scalar Ito SDE for which an explicit solution is always available and then turns to certain types of nonlinear scalar Ito SDEs that can be reduced to linear scalar Ito SDEs and hence solved explicitly. These methods can also be applied to a scalar Stratonovich SDE provided it is first converted to the corresponding Ito SDE.

1.6.1 Linearsde routine

The general form of a scalar *linear* SDE is

$$dX_t = (a_1(t)X_t + a_2(t))\,dt + (b_1(t)X_t + b_2(t))\,dW_t \tag{1.17}$$

where the coefficients a_1, a_2, b_1, b_2 are specified functions of time t or constants. The SDE is said to have *additive noise* when $b_1(t) \equiv 0$ in (2.1) or *multiplicative noise* when $b_1(t) \neq 0$.

In the general case the SDE (1.17) has the explicit solution

$$X_t = \Phi_{t,t_0}\left(X_{t_0} + \int_{t_0}^{t} (a_2(s) - b_1(s)b_2(s))\,\Phi_{s,t_0}^{-1}\,ds + \int_{t_0}^{t} b_2(s)\Phi_{s,t_0}^{-1}\,dW_s\right) \tag{1.18}$$

where

$$\Phi_{t,t_0} = \exp\left(\int_{t_0}^{t}\left(a_1(s) - \frac{1}{2}\,b_1^2(s)\right)\,ds + \int_{t_0}^{t} b_1(s)\,dW_s\right). \tag{1.19}$$

In the additive noise case, the SDE (1.17) reduces to

$$dX_t = (a_1(t)X_t + a_2(t))\ dt + b_2(t)\, dW_t, \tag{1.20}$$

and the explicit solution to

$$X_t = \Phi_{t,t_0}\left(X_{t_0} + \int_{t_0}^{t} a_2(s)\Phi_{s,t_0}^{-1}\, ds + \int_{t_0}^{t} b_2(s)\Phi_{s,t_0}^{-1}\, dW_s\right), \tag{1.21}$$

where

$$\Phi_{t,t_0} = \exp\left(\int_{t_0}^{t} a_1(s)\, ds\right).$$

These solutions can be confirmed with the Ito formula.

The routine `stochastic[linearsde]` determines the explicit solution of an Ito SDE (1.16) with the linear drift coefficient $a(t,x) = \alpha(t)x + \gamma(t)$ and linear diffusion coefficient $b(t,x) = \beta(t)x + \delta(t)$.

```
stochastic[linearsde]:=proc(a::algebraic,b::algebraic)
local temp1,alpha,beta,gamma,delta,fundsoln,fundsoln2,soln,
      default1,default2,default3;
   if diff(a,x,x) <> 0 or diff(b,x,x) <> 0 then
   ERROR(‘SDE not linear, try a reducible procedure‘)
   else
       alpha := diff(a,x);
       alpha := subs(t = s,alpha);
       beta := diff(b,x);
       beta := subs(t = s,beta);
       if diff(beta,s) = 0 then temp1 := beta*W;
       else temp1:=Int(beta,W = 0 .. t);
       fi;
       gamma := coeff(a,x,0);
       gamma := subs(t = s,gamma);
       delta := coeff(b,x,0);
       delta := subs(t = s,delta);
       fundsoln := exp(int(alpha-1/2*beta^2,s = 0 .. t)+temp1);
       fundsoln2 := subs(t = s,fundsoln);
       if beta = 0 then
           soln := fundsoln*(X[0]+int(1/fundsoln2*(gamma-beta*delta),
           s = 0 .. t)+Int(1/fundsoln2*delta,W = 0 .. t))
       else soln := fundsoln*(X[0]+Int(1/fundsoln2*(gamma-beta*delta),
       s = 0 .. t)+Int(1/fundsoln2*delta,W = 0 .. t))
       fi;
       default1 := Int(0,W = 0 .. t) = 0;
       default2 := Int(0,W = 0 .. s) = 0;
       default3 := Int(0,s = 0 .. t) = 0;
       soln := X[t] = subs(default1,default2,default3,soln);
   fi;
end:
```

The call `linearsde(a,b);` returns the explicit solution for a linear SDE (1.17) with linear drift coefficient a and linear diffusion coefficient b. The output consists of the variables $X[t]$, $X[0]$ and W, where $X[t]$ denotes the explicit solution, $X[0]$ the initial value of the solution, W a standard Wiener process, and t time. An error message is returned if the coefficients of a nonlinear SDE are used.

EXAMPLE: Consider the scalar linear Ito SDE with additive noise

$$dX_t = -X_t\,dt + 2\,dW_t,$$

i.e. with drift coefficient $a(t,x) = -x$ and diffusion coefficient $b(t,x) = 2$. The call `linear(-x,2);` returns the explicit solution in the following form.

```
> linearsde(-x,2);
```

$$X_t = e^{(-t)}\left(X_0 + \int_0^t 2\,\frac{1}{e^{(-s)}}\,d\mathrm{W}\right)$$

The required solution is thus

$$X_t = e^{-t}\left\{X_0 + \int_0^t 2e^s\,dW_s\right\}.$$

1.6.2 Reducible routine

An autonomous nonlinear scalar Ito SDE

$$dX_t = a(X_t)\,dt + b(X_t)\,dW_t \tag{1.22}$$

with drift coefficient

$$a(x) = \frac{1}{2}\,b(x)b'(x).$$

can be reduced to the linear scalar SDE

$$dY_t = dW_t$$

by the substitution

$$y = h(x) = \int^x \frac{ds}{b(s)},$$

giving the solution

$$X_t = h^{-1}(W_t + h(X_0)),$$

where $x = h^{-1}(y)$ is the inverse function of the function $y = h(x)$.

More generally, if the drift has the form

$$a(x) = \alpha\, b(x)h(x) + \frac{1}{2}\, b(x)b'(x) \tag{1.23}$$

then the SDE (1.22) can be reduced to the Langevin equation

$$dY_t = \alpha Y_t\, dt + \beta\, dW_t$$

with the explicit solution

$$X_t = h^{-1}\left(e^{\alpha t}h(X_0) + e^{\alpha t}\int_0^t e^{-\alpha s}\, dW_s\right).$$

The following routine `stochastic[reducible]` returns the explicit solution of a reducible SDE (1.22) with drift of the form (1.23). Its calling sequence is `reducible(a,b);` with parameters a and b representing the drift and diffusion coefficient of the SDE, which should not depend explicitly on the t variable.

```
stochastic[reducible]:=proc(a::algebraic,b::algebraic)
local beta,temp1,h,temp3,alpha,soln,soln1;
    h := int(1/b,x);
    temp1 := alpha*b*h+1/2*b*simplify(diff(b,x));
    temp1 = a;
    alpha := simplify(solve(",alpha));
    beta := alpha*h;
    if diff(alpha,x) = 0 then
        if alpha=0 then
            soln:=h=subs(x=X[0],h)+W;
            X[t]=simplify(solve(soln,x));
        else
        soln1 := h = exp(alpha*t)*subs(
                     x = X[0],h)+exp(alpha*t)*Int(exp(-alpha
                                    *s),W = 0 .. t);
        X[t] = solve(soln1,x);
        fi
    elif diff(beta,x) = 0 then
    X[t]=simplify(solve(h = beta*t+W+subs(x = X[0],h),x));
    else ERROR(‘non-linear SDE not reducible‘)
    fi
end:
```

The call `reducible(a,b);` returns the explicit solution for a reducible SDE with drift a and diffusion coefficient b if the SDE is of the appropriate reducible form. The output consists of the variables X[t], X[0] and W, where X[t] denotes the explicit solution, X[0] the initial value of the solution, W a standard Wiener process and t time. If the SDE is not of the above reducible

form, then a suitable error message is returned.

EXAMPLE: Consider the scalar nonlinear Ito SDE

$$dX_t = X_t^3\,dt + X_t^2\,dW_t.$$

```
> reducible(x^3,x^2);
```

$$\mathtt{X[t]} = -\frac{\mathtt{X[0]}}{-1 + \mathtt{W}\ \mathtt{X[0]}}$$

This SDE is reducible and the required solution is

$$X_t = \frac{X_0}{1 - W_t X_0}.$$

1.6.3 Explicit routine

The routine `stochastic[explicit]` combines the two previously described routines `stochastic[linear]` and `stochastic[reducible]`, which are applied to a general scalar Ito SDE: It returns the explicit solution if the SDE is either linear or reducible as in the preceding subsections, otherwise a suitable error message is returned.

```
stochastic[explicit]:= proc(a::algebraic,b::algebraic)
   if diff(a,x,x) = 0 and diff(b,x,x) = 0
     then linear (a,b)
     else reducible(a,b)
   fi
end:
```

EXAMPLE: Consider the scalar Ito SDE

$$dX_t = \frac{1}{2}a^2 X_t\,dt + aX_t\,dW_t,$$

with drift $a(x) = \frac{1}{2}a^2 x$ and diffusion coefficient $b(x) = ax$, where a is a constant.

```
> explicit(1/2*a^2*x,a*x);
```

$$\mathtt{X[t]=exp(aW)X[0]}$$

This SDE is linear and thus explicitly solvable with the solution

$$X_t = X_0 e^{aW_t}.$$

1.7 Linear Vector SDEs

The general form of a N-dimensional *linear* vector Ito SDE is

$$dX_t = \{A(t)X_t + a(t)\}\, dt + \sum_{l=1}^{M} \left\{B^l(t)X_t + b^l(t)\right\}\, dW_t^l \tag{1.24}$$

where $A(t)$, $B^1(t)$, $B^2(t)$, ..., $B^M(t)$ are $N \times N$-matrix functions and $a(t)$, $b^1(t)$, $b^2(t)$, ..., $b^M(t)$ are N-dimensional vector functions. The SDE (1.24) is said to be *linear in the narrow-sense* or to have *additive noise* when the B^l are all identically zero and to be *homogeneous* when a and the b^l are all zero.

The solution of the linear SDE (1.24) is

$$X_t = \Phi_{t,t_0}\left(X_{t_0} + \int_{t_0}^{t} \Phi_{s,t_0}^{-1}\left(a(s) - \sum_{l=1}^{M} B^l(s)b^l(s)\right) ds + \sum_{l=1}^{M}\int_{t_0}^{t} \Phi_{s,t_0}^{-1} b^l(s)\, dW_s^l\right), \tag{1.25}$$

where Φ_{t,t_0} is the $N \times N$-matrix satisfying $\Phi_{t_0,t_0} = I$ and the homogeneous *matrix* SDE

$$d\Phi_{t,t_0} = A(t)\Phi_{t,t_0}\, dt + \sum_{l=1}^{M} B^l(t)\Phi_{t,t_0}\, dW_t^l, \tag{1.26}$$

which is interpreted columnwise as N vector SDEs. Unlike the scalar homogeneous linear equations, it is generally not possible to solve (1.26) explicitly for Φ_{t,t_0}, even when all of the matrices are constant matrices. If, however, the matrices A, B^1,B^2, ..., B^M are constants and commute, i.e. if

$$AB^l = B^l A \quad \text{and} \quad B^l B^k = B^k B^l$$

for all k, $l = 1, 2, \ldots, M$, then Φ_{t,t_0} is given explicitly by

$$\Phi_{t,t_0} = \exp\left(\left(A - \frac{1}{2}\sum_{l=1}^{M}\left(B^l\right)^2\right)(t-t_0) + \sum_{l=1}^{M} B^l\left(W_t^l - W_{t_0}^l\right)\right), \tag{1.27}$$

which can be evaluated with the help of MAPLE for specific values of $t \geq t_0$ and for a specific sample path of the Wiener process. In the special case that the linear SDE (1.24) has additive noise it reduces to

$$\Phi_{t,t_0} = \exp\left(A(t-t_0)\right), \tag{1.28}$$

the fundamental matrix of the deterministic linear system $\dot{x} = Ax$, which can be evaluated for all $t - t_0 \geq 0$ using the procedure `exponential(A,t-t_0)` from the MAPLE **linalg** package.

1.7.1 Linearization

The linearization of an N-dimensional vector Ito SDE with an M-dimensional Wiener process

$$dX_t = a(t, X_t)\,dt + \sum_{j=1}^{M} b^j(t, X_t)\,dW_t^j \tag{1.29}$$

about a given point or solution $\bar{X}_t$, results in the linear vector SDE

$$dZ_t = A(t)Z_t\,dt + \sum_{j=1}^{M} B^j(t)Z_t\,dW_t^j \tag{1.30}$$

where the $N \times N$-matrices A, B^1, ..., B^M are defined componentwise by

$$A(t)^{i,j} = \frac{\partial a^i}{\partial x^j}(t, \bar{X}_t), \qquad B^k(t)^{i,j} = \frac{\partial b^{i,k}}{\partial x^j}(t, \bar{X}_t)$$

for $i, j = 1, \ldots, N$ and $k = 1, \ldots, M$. These matrices are determined by the following MAPLE procedure.

```
stochastic[linearize]:=proc(a::list(algebraic),
                            b::list(list(algebraic)),
                            c::list(algebraic))
local i,tempA,tempB,j,k,l;
   tempA:=array(1..nops(a),1..nops(a));
   for i from 1 to nops(a) do
     for j from 1 to nops(a) do
       tempA[i,j]:=diff(op(i,a),x[j]);
   od; od;
   for i from 1 to nops(a) do
     for j from 1 to nops(a) do
       for l from 1 to nops(c) do
         tempA[i,j]:=subs(x[l]=op(l,c),tempA[i,j]);
   od; od; od;
   for k from 1 to nops(op(1,b)) do
     tempB[k]:=array(1..nops(a),1..nops(a));
     for i from 1 to nops(a) do
       for j from 1 to nops(a) do
         tempB[k][i,j]:=diff(op(k,op(i,b)),x[j]);
     od; od;
     for i from 1 to nops(a) do
       for j from 1 to nops(a) do
         for l from 1 to nops(c) do
           tempB[k][i,j]:=subs(x[l]=op(l,c),tempB[k][i,j]);
     od; od; od;
   od;
   RETURN(A=map(simplify,convert(eval(tempA),matrix)),B=eval(tempB))
end:
```

The call

```
linearize([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]],[c1,..,cN]);
```

determines the coefficient matrices of the linearized SDE (1.30) for the SDE (1.29) about a point or solution $\bar{X}$. Here the coefficients `ai` and `bij` again follow the convention from Section 1.3.1 and the `c1,..,cN` denote the components of the point or solution $\bar{X}$ about which the linearization is taken.

EXAMPLE: Linearize the 2-dimensional Ito SDE

$$\begin{aligned} dX_t^1 &= X_t^2 dt, \\ dX_t^2 &= (-bX_t^2 - \sin X_t^1 - c \sin 2X_t^1)dt + (-a(X_t^2)^2 + \sin X_t^1)dW_t, \end{aligned}$$

where a, b and c are constants and W_t is a scalar Wiener process, about the zero solution $\bar{X}_t^1 = \bar{X}_t^2 = 0$, i.e. apply

```
> linearize([x[2],-b*x[2]-sin(x[1])-c*sin(2*x[1])],[[0],
            -a*(x[2])^2+sin(x[1])]],[0,0]);
```

to obtain

$$A = \begin{bmatrix} 0 & 1 \\ -1-2c & -b \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0\,0 \\ 1\,0 \end{bmatrix}.$$

The linearized SDE (1.30) is thus

$$\begin{aligned} dZ_t^1 &= Z_t^2\, dt \\ dZ_t^2 &= \left((-1-2c)Z_t^1 - bZ_t^2\right)\, dt + Z_t^1\, dW_t. \end{aligned}$$

1.7.2 Spherical coordinates

The N-dimensional linear Stratonovich SDE with an M-dimensional Wiener process

$$dZ_t = A(t)Z_t\, dt + \sum_{k=1}^{M} B^k(t)Z_t \circ dW_t^k, \tag{1.31}$$

where A, B^1, B^2, ..., B^M are $N \times N$ matrices, converts in spherical coordinates $r = |z|$ and $s = z/|z| \in \mathcal{S}^{N-1}$ (assuming $z \neq 0$) to the system of Stratonovich SDEs

$$dR_t = R_t\, q^0(S_t)\, dt + \sum_{k=1}^{M} R_t\, q^k(S_t) \circ dW_t^k \tag{1.32}$$

$$dS_t = h(S_t, A)\, dt + \sum_{k=1}^{M} h(S_t, B^k) \circ dW_t^k \tag{1.33}$$

where

$$q(s) = s^\top As + \sum_{k=1}^{M} \left(\frac{1}{2} s^\top \left(B^k + \left(B^k\right)^\top \right) s - \left(s^\top B^k s \right)^2 \right),$$

$$q^0(s) = s^\top As, \quad q^k(s) = s^\top B^k s, \quad h(s,A) = \left(A - (s^\top As) I \right) s.$$

This system is used to investigate the stability of the zero solution of the original linear SDE (1.31). The above coefficients are determined by the routine `stochastic[sphere]`.

```
stochastic[sphere]:=proc(a,b)
global q,q0,qk,h,hk;
local i,j,k,tempa,tempb,stempbs,N,tmp;

  if type(a,array) then tmp:=convert(a,listlist);else tmp:=a; fi;
  N:=nops(tmp);
  hk:=evaln(hk); h:=evaln(h); qk:=evaln(qk);
  q:=evaln(q); q0:=evaln(q0);
  tempa:=evaln(tempa);tempb:=evaln(tempb);
  q0:=sum('sum('s[i]*a[i,j]','i'=1..N)*s[j]','j'=1..N);
  for k from 1 to nops(b) do
    qk[k]:=sum('sum('s[i]*b[k][i,j]','i'=1..N)*s[j]','j'=1..N);
  od;
  for k from 1 to nops(b) do
    stempbs[k]:=sum('sum('s[i]*(b[k][i,j]+b[k][j,i])','i'=1..N)
                  *s[j]','j'=1..N);
  od;
  q:=q0+ sum('0.5*stempbs[k]-qk[k]^2','k'=1..nops(b));
  for i from 1 to N do
    for j from 1 to N do
      if (i=j) then tempa[i,i]:=a[i,i]-q0;
               else tempa[i,j]:=a[i,j];
      fi;
  od; od;
  for i from 1 to N do
    h[i]:=sum('tempa[i,j]','j'=1..N);
  od;
  for k from 1 to nops(b) do
    for i from 1 to N do
      for j from 1 to N do
        if (i=j) then tempb[k][i,i]:=b[k][i,i]-qk[k];
                 else tempb[k][i,j]:=b[k][i,j];
        fi;
  od; od; od;
  for k from 1 to nops(b) do
    for i from 1 to N do
      hk[k][i]:=sum('tempb[k][i,j]','j'=1..N);
  od; od;
```

```
end:
```

The variables here are defined as **global**, which means that to determine the value of $q(s)$, for example, one uses the call `sphere(A,B): q;`. Note that here a list of matrices $B^1, \ldots, B^M$ is needed as input. Each of the input matrices can be entered either following the terminology from Section 1.3.1 or as a MAPLE array. This second method is used in the following example.

EXAMPLE: Consider the linear Stratonovich SDE (1.31) with coefficient matrices

$$A := \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \quad \text{and } B := \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$$

which are input into Maple as

```
>A:=array(1..2,1..2,[[2,2],[1,1]]):
>B1:=array(1..2,1..2,[[3,1],[4,2]]):
>B:=[B1]:
```

Then the coefficients of the system (1.32)–(1.33) are determined then as

```
> sphere(A,B):
> q;
```

$$(2\,s_1 + s_2)\,s_1 + (2\,s_1 + s_2)\,s_2 + .5000000000\,(6.\,s_1 + 5.\,s_2)\,s_1 + .5000000000\,(5.\,s_1 + 4.\,s_2)\,s_2 - 1.\,((3.\,s_1 + 4.\,s_2)\,s_1 + (s_1 + 2.\,s_2)\,s_2)^2$$

```
> q0;
```

$$(2\,s_1 + s_2)\,s_1 + (2\,s_1 + s_2)\,s_2$$

```
> print(qk);
```

table([
$1 = (3\,s_1 + 4\,s_2)\,s_1 + (s_1 + 2\,s_2)\,s_2$
])

```
> print(h);
```

table([
$1 = 4 - (2\,s_1 + s_2)\,s_1 - (2\,s_1 + s_2)\,s_2$
$2 = 2 - (2\,s_1 + s_2)\,s_1 - (2\,s_1 + s_2)\,s_2$
])

```
> print(hk);
```

```
table([
  1 = table([
  1 = 4 − (3 s1 + 4 s2) s1 − (s1 + 2 s2) s2
  2 = 6 − (3 s1 + 4 s2) s1 − (s1 + 2 s2) s2
  ])
  ])
```

1.7.3 Second moment equation

The N-dimensional linear Ito SDE

$$dZ_t = A(t)Z_t\,dt + \sum_{j=1}^{M} B^j(t)Z_t\,dW^j \tag{1.34}$$

with $N\times N$ matrices A, B^1, B^2, …, B^M has the $N\times N$ matrix valued second moment $P(t) = E(Z_t Z_t^\top)$ which satisfies the deterministic matrix differential equation

$$\frac{dP}{dt} = A(t)P + PA(t)^\top + \sum_{k=1}^{M} B^k(t)PB^k(t)^\top,$$

which is linear in P. Due to the symmetry of the matrix P this equation can be rewritten as a linear system of the from

$$\frac{d\tilde{p}}{dt} = \mathcal{A}(t)\tilde{p} \tag{1.35}$$

where $\tilde{p}$ is an $\frac{1}{2}N(N+1)$-dimensional vector consisting of the free components of P and $\mathcal{A}(t)$ is a $\frac{1}{2}N(N+1)! \times \frac{1}{2}N(N+1)!$ matrix.

The following procedure `stochastic[momenteqn]` calculates this matrix $\mathcal{A}(t)$.

```
stochastic[momenteqn]:=proc(A,B)
local i,j,k,N,Btmp,Ctmp;
global New_A;
  if type(A,array) then Btmp:=convert(A,listlist);
                   else Btmp:=A;
  fi;
  N:=nops(Btmp);
  New_A:=array(1..N*(N+1)/2,1..N*(N+1)/2);
  Ctmp:=array(1..N*(N+1)/2,1..N*(N+1)/2);
  stochastic[ap](A);
  stochastic[pa](A);
  for i from 1 to N*(N+1)/2 do
    for j from 1 to N*(N+1)/2 do
```

```
      Ctmp[i,j]:=0;
  od; od;
  for k from 1 to nops(B) do
    stochastic[bpb](B[k]);
    for i from 1 to N*(N+1)/2 do
      for j from 1 to N*(N+1)/2 do
        Ctmp[i,j]:=Ctmp[i,j]+B3[i,j];
  od; od; od;
  for i from 1 to N*(N+1)/2 do
    for j from 1 to N*(N+1)/2 do
      New_A[i,j]:=B1[i,j]+B2[i,j]+Ctmp[i,j];
  od; od;
  RETURN(evalm(New_A));
end:
```

The input format is similar to the `sphere` command. Note that this procedure requires other procedures that are described in the Section 1.7.5.

EXAMPLE: We apply the routine to the matrices

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad B^1 = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

```
> A:=array(1..2,1..2,[[a11,a12],[a21,a22]]):
> B1:=array(1..2,1..2,[[b11,b12],[b21,b22]]):
> B:=[B1];
> momenteqn(A,B);
```

$$\begin{bmatrix} 2\,a11 + b11^2 & 2\,a12 + 2\,b12\,b11 & b12^2 \\ a21 + b11\,b21 & a11 + a22 + b12\,b21 + b11\,b22 & a12 + b12\,b22 \\ b21^2 & 2\,a21 + 2\,b22\,b21 & 2\,a22 + b22^2 \end{bmatrix}$$

The equation (1.35) is thus

$$\begin{aligned} d\tilde{p}_1 &= \left(2\,a_{11} + {b_{11}}^2\right)\tilde{p}_1 + \left(2\,a_{12} + 2\,b_{12}\,b_{11}\right)\tilde{p}_2 + \left({b_{12}}^2\right)\tilde{p}_3 \\ d\tilde{p}_2 &= \left(a_{21} + b_{11}\,b_{21}\right)\tilde{p}_1 + \left(a_{11} + a_{22} + b_{12}\,b_{21} + b_{11}\,b_{22}\right)\tilde{p}_2 \\ &\quad + \left(a_{12} + b_{12}\,b_{22}\right)\tilde{p}_3 \\ d\tilde{p}_3 &= \left({b_{21}}^2\right)\tilde{p}_1 + \left(2\,a_{21} + 2\,b_{22}\,b_{21}\right)\tilde{p}_2 + \left(2\,a_{22} + {b_{22}}^2\right)\tilde{p}_3 \end{aligned}$$

1.7.4 The procedures pmatrix2pvector and pvector2pmatrix

The procedures `pmatrix2pvector` and `pvector2pmatrix` transform a symmetric matrix to a vector and a vector to a symmetric matrix, respectively. They are useful above to change the matrix P to the vector $\tilde{p}$ and the vector $\tilde{p}$ to the matrix P, respectively.

The procedure `stochastic[pmatrix2pvector]` has the following code.

```
stochastic[pmatrix2pvector]:=proc(p)
local i,j,k,ptmp;
global pvector;
  if type(p,array) then ptmp:=convert(p,listlist);
                   else ptmp:=p;
  fi;
  pvector:=array(1..nops(ptmp)*(nops(ptmp)+1)/2);
  k:=0;
  for i from 1 to nops(ptmp) do
    if (i>1) then k:=k+(nops(ptmp)-i+2); fi;
    for j from i to nops(ptmp) do
       pvector[k+j-i+1]:=ptmp[i,j];
  od; od;
  RETURN(eval(pvector));
end:
```

The matrix P can be entered following Section 1.3.1 or as a MAPLE array.

EXAMPLE: We apply the procedure to the matrix

$$P = \begin{bmatrix} 2 & 5 & 10 & 17 & 26 \\ 5 & 6 & 11 & 18 & 27 \\ 10 & 11 & 12 & 19 & 28 \\ 17 & 18 & 19 & 20 & 29 \\ 26 & 27 & 28 & 29 & 30 \end{bmatrix}$$

in array notation.

```
> P:=array(1..5,1..5,[[2,5,10,17,26],[5,6,11,18,27],
           [10,11,12,19,28],[17,18,19,20,29],[26,27,28,29,30]]):
> pmatrix2pvector(P);

     [2, 5, 10, 17, 26, 6, 11, 18, 27, 12, 19, 28, 20, 29, 30]
```

EXAMPLE: Now we use the notation of Section 1.3.1.

```
> P:=[[1,2],[2,4]]:
> pmatrix2pvector(P);

                              [1, 2, 4]
```

The procedure `pvector2pmatrix` is the inverse of `pmatrix2pvector`. It transforms a vector to a symmetric matrix (in MAPLE array notation) and has following code:

```
stochastic[pvector2pmatrix]:=proc(pvector)
local i,j,k,ptmp,N;
global p;
  if type(pvector,array) then ptmp:=convert(pvector,list);
                         else ptmp:=pvector;
```

```
  fi;
  N:=-1/2+sqrt(1/4+2*nops(ptmp));
  p:=array(1..N,1..N);
  k:=0;
  for i from 1 to N do
    if (i>1) then k:=k+(N-i+2); fi;
    for j from i to N do
      p[i,j]:=ptmp[k+j-i+1];
      if (i<>j) then p[j,i]:=p[i,j]; fi;
  od; od;
  RETURN(eval(p));
end:
```

EXAMPLE :

```
> p:=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]:
> pvector2pmatrix(p);
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 6 & 7 & 8 & 9 \\ 3 & 7 & 10 & 11 & 12 \\ 4 & 8 & 11 & 13 & 14 \\ 5 & 9 & 12 & 14 & 15 \end{bmatrix}$$

1.7.5 Subprocedures for momenteqn

The following procedures are subprocedures for the calculation or transformation parts of the procedure `momenteqn`. For example, the procedure `position` determines the position in the new vector and the procedure `ap` transforms the product AP in a vector equation.

```
stochastic[position]:=proc(N,i,j)
global stelle;
  stelle:=sum('N-k+1','k'=1..i-1)+j-i+1;
end:

stochastic[ap]:=proc(A)
local i,j,k,Atmp,N,counter;
global B1;
  if type(A,array) then Atmp:=convert(A,listlist);
                   else Atmp:=A;
  fi;
  N:=nops(Atmp);
  B1:=array(1..N*(N+1)/2,1..N*(N+1)/2);
  for i from 1 to N*(N+1)/2 do
    for j from 1 to N*(N+1)/2 do
      B1[i,j]:=0;
```

```
  od; od;
  counter:=0:
  for i from 1 to N do
    for j from i to N do
      counter:=counter+1;
      for k from 1 to N do
        if (j<=k)  then B1[counter,position(N,j,k)]:=A[i,k];
                   else B1[counter,position(N,k,j)]:=A[i,k];
        fi;
  od; od; od;
  RETURN(evalm(B1));
end:

stochastic[pa]:=proc(A)
local i,j,k,Atmp,N,counter;
global B2;
  if type(A,array) then Atmp:=convert(A,listlist);
                   else Atmp:=A;
  fi;
  N:=nops(Atmp);
  B2:=array(1..N*(N+1)/2,1..N*(N+1)/2);
  for i from 1 to N*(N+1)/2 do
    for j from 1 to N*(N+1)/2 do
      B2[i,j]:=0;
  od; od;
  counter:=0:
  for i from 1 to N do
    for j from i to N do
      counter:=counter+1;
      for k from 1 to N do
        if (i<=k)  then B2[counter,position(N,i,k)]:=A[j,k];
                   else B2[counter,position(N,k,i)]:=A[j,k];
        fi;
  od; od; od;
  RETURN(evalm(B2));
end:

stochastic[bpb]:=proc(B)
local i,j,k,l,Btmp,N,counter;
global B3;
  if type(B,array) then Btmp:=convert(B,listlist);
                   else Btmp:=B;
  fi;
  N:=nops(Btmp);
  B3:=array(1..N*(N+1)/2,1..N*(N+1)/2);
  for i from 1 to N*(N+1)/2 do
    for j from 1 to N*(N+1)/2 do
      B3[i,j]:=0;
```

```
  od; od;
  counter:=0:
  for i from 1 to N do
    for j from i to N do
      counter:=counter+1;
      for l from 1 to N do
        for k from 1 to N do
          if (k<=l) then
            B3[counter,position(N,k,l)]:=B3[counter,position(N,k,l)]
                                          +B[i,k]*B[j,l];
          else
            B3[counter,position(N,l,k)]:=B3[counter,position(N,l,k)]
                                          +B[i,k]*B[j,l];
          fi;
  od; od; od; od;
  RETURN(evalm(B3));
end:
```

1.8 Commutative and Coloured Noise

Certain structural relationships between the noise coefficient vectors b^j, $j = 1, \ldots, M$, of an SDE that are known as commutative noise allow considerable simplifications to numerical schemes, in particular the avoidance of the need to simulate multiple stochastic integrals. MAPLE routines that test for these conditions are presented here along with a routine that converts the SDE driven by Wiener processes to the corresponding SDE driven by a coloured noise Ornstein–Uhlenbeck process.

1.8.1 Commutative noise of 1st kind

An N-dimensional Ito SDE with an M-dimensional Wiener process in component form

$$dX_t^i = a^i(t, X_t)\, dt + \sum_{j=1}^{M} b^{i,j}(t, X_t)\, dW_t^j, \qquad i = 1, \ldots, N, \tag{1.36}$$

is said to have *commutative noise of the first kind* when the diffusion coefficients satisfy the condition

$$L^{j_1} b^{i,j_2}(t, x) = L^{j_2} b^{i,j_1}(t, x) \tag{1.37}$$

for all $i = 1, \ldots, N$, $j_1, j_2 = 1, \ldots, M$, and $(t, x) \in \mathbb{R}^+ \times \mathbb{R}^N$.

For instance, additive noise, diagonal noise and linear noise all satisfy this commutativity condition. Here *diagonal noise* means that

$$b^{k,j}(t, x) \equiv 0 \quad \text{and} \quad \frac{\partial b^{j,j}}{\partial x^k}(t, x) \equiv 0$$

and *linear noise* means that

$$b^{i,j}(t,x) = b^{i,j}(t)\, x^i$$

for all $i = 1, \ldots, N$, $j = 1, \ldots, M$, and $(t,x) \in \mathbb{R}^+ \times \mathbb{R}^N$.

The routine `stochastic[comm1]` informs the user whether or not an Ito SDE (1.36) has commutative noise of the first kind.

```
stochastic[comm1]:=proc()
local LJ1,LJ2,k,j1,j2,flag,p;
   for p to nargs do
     if type(args[p],list) <> true then
       ERROR(‘Expecting input to be an expression sequence of lists‘)
    fi;
    od;
    for k to nargs do
      for j1 to nops(args[1]) do
        for j2 to nops(args[1]) do
           LJ1 := sum(’op(j1,args[l])*diff(op(j2,args[k]),x[l])’,
                      ’l’ = 1 .. nargs);
           LJ2 := sum(’op(j2,args[l])*diff(op(j1,args[k]),x[l])’,
                      ’l’ = 1 .. nargs);
           if LJ1 <> LJ2 then flag := 1 fi;
    od; od; od;
    if flag = 1 then
      RETURN(‘Commutative noise of the first kind
              doesn’t exist for this system‘)
    else
      RETURN(‘This system exhibits commutative noise
              of the first kind‘)
    fi;
end:
```

The call `comm1([b11,..,b1M],..,[bN1,..,bNM]);` returns a statement indicating whether or not the SDE with this diffusion coefficient matrix has commutative noise of the first kind (1.37).

EXAMPLE: Consider a 2-dimensional Ito SDE with the variable diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} 0 \\ x_1 \end{pmatrix}.$$

```
> comm1([1,0],[0,x[1]]);

Commutative noise of the first kind doesn’t exist for this system
```

1.8.2 Commutative noise of 2nd kind

The SDE (1.36) has *commutative noise of the second kind* when the noise coefficients satisfy the condition

$$L^{j_1} J^{j_2} b^{k,j_3}(t,x) = L^{j_2} L^{j_1} b^{k,j_3}(t,x) \tag{1.38}$$

for all j_1, j_2, $j_3 = 1, \ldots, M$, $k = 1, \ldots, N$, and $(t,x) \in \mathbb{R}^+ \times \mathbb{R}^N$.

The routine `stochastic[comm2]` informs the user if the diffusion matrix of an Ito SDE has commutative noise of the second kind.

```
stochastic[comm2]:= proc()
local LJ1LJ2,LJ2LJ1,k,p,j1,j2,j3,flag;
  for p to nargs do
    if type(args[p],list) <> true then
      ERROR(‘Expecting input to be an expression sequence of lists‘)
    fi;
    od;
    for k to nargs do
      for j1 to nops(args[1]) do
        for j2 to nops(args[1]) do
          for j3 to nops(args[1]) do
             LJ1LJ2 := sum(’op(j1,args[m])*diff(sum(’op(j2,args[l])*
                       diff(op(j3,args[k]),x[l])’,
                       ’l’ = 1 .. nargs),x[m])’, ’m’ = 1 .. nargs);
             LJ2LJ1 := sum(’op(j2,args[m])*diff(sum(’op(j1,args[l])*
                       diff(op(j3,args[k]),x[l])’,
                       ’l’ = 1 .. nargs),x[m])’, ’m’ = 1 .. nargs);
             if LJ1LJ2 <> LJ2LJ1 then flag := 1 fi;
    od; od; od; od;
    if flag = 1 then
      RETURN(‘Commutative noise of the second kind
              doesn’t exist for this system‘)
    else
      RETURN(‘This system exhibits commutative noise
              of the second kind‘)
    fi;
end:
```

The call `comm2([b11,..,b1M],..,[bN1,..,bNM]);` returns a statement indicating whether or not the diffusion matrix of the SDE has commutative noise of the second kind (1.38).

EXAMPLE: Consider an 2-dimensional Ito SDE with the variable diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} 1 \\ (x_2)^2 (x_1)^4 \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} 0 \\ (x_1)^2 \end{pmatrix}.$$

```
> comm2([1,0],[(x[2])^2*(x[1])^4,(x[1])^2]);

Commutative noise of the second kind doesn't exist for this system
```

1.8.3 Coloured Noise

An N-dimensional Ito SDE with a scalar Wiener process W_t (i.e., with $M = 1$)

$$dX_t = a(t, X_t)\, dt + b(t, X_t)\, dW_t \tag{1.39}$$

can be converted into into an associated SDE with coloured noise, i.e., driven by an Ornstein–Uhlenbeck or exponentially correlated coloured noise process Z_t [20]. The resulting coloured noise equation is the $(N+1)$-dimensional Ito SDE with scalar additive noise

$$dX_t = (a(t, X_t) + b(t, X_t)\, Z_t)\ dt \tag{1.40}$$

$$dZ_t = -\gamma Z_t\, dt + \beta\, dW_t \tag{1.41}$$

The routine `stochastic[colour]` converts the SDE (1.39) with scalar white noise into its coloured noise counterpart (1.40)–(1.41).

```
stochastic[colour]:=proc(a:list(algebraic),b:list(algebraic))
local temp1,i;
  for i to nops(a) do
    temp1[i] := dx[i][t] = a[i]*dt+b[i]*z[t]*dt
  od;
  temp1[i] := dz[t] = -gamma*z[t]*dt+beta*dW[t];
  RETURN(eval(temp1))
end:
```

The call `colour([a1,..,aN], [b1,..,bN]);` converts an SDE (1.39) in dimension N with scalar white noise into its coloured noise form (1.40)–(1.41). The output consists of the variables z, $x[N]$, W, gamma, beta and t. Here z denotes the Ornstein–Uhlenbeck process, $(x[N], z)$ the state variable of the $(N+1)$-dimensional SDE (1.40)–(1.41) and W a standard Wiener process, while gamma and beta denote parameters and t denotes time.

EXAMPLE: Convert the 2-dimensional SDE with scalar white noise

$$dX_t^1 = X_t^2\, dt, \qquad dX_t^2 = \left(X_t^1 \left(\alpha - \left(X_t^1\right)^2\right) - X_t^2\right)\, dt + \sigma\, dW_t$$

into its coloured noise counterpart.

```
> colour([x[2],x[1]*(alpha-x[1]^2)-x[2]],[0,sigma*x[1]]);

table([
```

```
    1 = (dx[1][t] = x[2] dt)

                                          2
    2 = (dx[2][t] = (x[1] (alpha - x[1] ) - x[2]) dt
         + sigma x[1] z[t] dt)

    3 = (dz[t] = - gamma z[t] dt + beta dW[t])
])
```

The resulting coloured noise system is

$$dX_t^1 = X_t^2\, dt$$

$$dX_t^2 = \left(X_t^1 \left(\alpha - \left(X_t^1\right)^2\right) - X_t^2 + \sigma\, Z_t\right) dt$$

$$dZ_t = -\gamma\, Z_t\, dt + \beta\, dW_t$$

1.9 Strong Numerical Schemes

Strong stochastic Taylor schemes of orders 0.5, 1.0 and 1.5 are considered for the N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t^i = a^i(t, X_t)\, dt + \sum_{j=1}^{M} b^{i,j}(t, X_t)\, dW_t^j, \qquad i = 1, \ldots, N, \tag{1.42}$$

as well as the strong order 2.0 stochastic Taylor scheme for the corresponding Stratonovich SDE.

The coefficients are all evaluated at the point (t_n, Y_n) in all of the schemes that follow, although for conciseness (t_n, Y_n) will not be explicitly written.

1.9.1 Euler scheme

The strong stochastic Taylor scheme of order 0.5 for the SDE (1.42), usually called the stochastic *Euler scheme,* has the componentwise form

$$Y_{n+1}^i = Y_n^i + a^i\, \Delta_n + \sum_{j=1}^{M} b^{i,j}\, \Delta W_n^j, \qquad i = 1, \ldots, N, \tag{1.43}$$

where $\Delta_n = t_{n+1} - t_n$ is the length of the nth time step and $\Delta W_n^j = W_{t_{n+1}}^j - W_{t_n}^j$ is the $N(0; \Delta_n)$–distributed increment of the jth component of the M-dimensional standard Wiener process W_t on the discretization subinterval $[t_n, t_{n+1}]$. Here $\Delta W_n^{j_1}$ and $\Delta W_n^{j_2}$ are independent for $j_1 \neq j_2$.

The routine `stochastic[Euler]` constructs the stochastic Euler scheme for the Ito SDE (1.42).

```
stochastic[Euler]:=proc(a::list(algebraic),b::list(list(algebraic)))
local i,u,soln;
  for i to nops(a) do
    soln[i] := Y.i[n+1] = Y.i[n]+L0(x[i],a,b)*Delta[n]+
          sum('LJ(x[i],b,j)*Delta*W.j[n]','j' = 1 .. nops(op(1,b)));
    for u to nops(a) do soln[i] := subs(x[u] = Y.u[n],soln[i]) od;
  od;
  RETURN(eval(soln))
end:
```

The call `Euler([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);` returns the Euler scheme for an N-dimensional Ito SDE with M-dimensional noise which has drift coefficient components $a1$,..., aN and diffusion coefficient matrix $[b^{i,j}]$ with rows $[b11,\ldots,b1M]$,..., $[bN1,\ldots,bNM]$.

The output variables are consistent with the variables used as input. The output consists of the variables $YN[n]$, Delta$WM[n]$, and Delta$[n]$. $YN[n]$ denotes the Euler approximation to $x[N]$ at the nth step. Delta$WM[n]$ denotes the change in the M-dimensional Wiener process at the nth step. Delta$[n]$ denotes the step size at the nth step.

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process $W_t = (W_t^1, W_t^2)$, given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^2 \\ X_t^1 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ vX_t^1 \end{pmatrix} dW_t^2,$$

i.e. with drift components $a^1(t,x_1,x_2) = x_2$, $a^2(t,x_1,x_2) = x_1$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ vX_t^1 \end{pmatrix},$$

where r, s, u and v are constants.

```
> Euler([x[2],x[1]],[[r,u],[s,v*x[1]]]);
table([
    1 = (Y1[n + 1] = Y1[n] + Y2[n] Delta[n]
                    + r Delta W1[n] + u Delta W2[n])

    2 = (Y2[n + 1] = Y2[n] + Y1[n] Delta[n]
                    + s Delta W1[n] + v Y1[n] Delta W2[n])
     ])
```

The resulting Euler scheme is

$$\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W_n^1 + \begin{pmatrix} u \\ vY_n^1 \end{pmatrix} \Delta W_n^2.$$

1.9.2 Milstein scheme

The strong stochastic Taylor scheme of order 1.0 for the SDE (1.42), usually called the *Milstein scheme,* has the componentwise form

$$Y_{n+1}^i = Y_n^i + a^i\,\Delta_n + \sum_{j=1}^{M} b^{i,j}\Delta W_n^j + \sum_{j_1,j_2=1}^{M} L^{j_1} b^{k,j_2} I_{(j_1,j_2);n}, \qquad i = 1,\ldots,N, \tag{1.44}$$

where $I_{(j_1,j_2);n}$ is the multiple Ito integral

$$I_{(j_1,j_2);n} = \int_{t_n}^{t_{n+1}} \int_{t_n}^{s_1} dW_{s_2}^{j_1}\, dW_{s_1}^{j_2}, \tag{1.45}$$

which simplifies to

$$I_{(j,j);n} = \frac{1}{2}\left\{\left(\Delta W_n^j\right)^2 - \Delta_n\right\}$$

for $j_1 = j_2 = j$.
The routine `stochastic[Milstein]` constructs the Milstein scheme for the Ito SDE (1.42).

```
stochastic[Milstein]:=proc(a::list(algebraic),
                           b::list(list(algebraic)))
local u,i,soln;
  for i to nops(a) do
    soln[i] := Y.i[n+1] = Y.i[n]+L0(x[i],a,b)*Delta[n]
        +sum('LJ(x[i],b,j)*Delta*W.j[n]','j' = 1 .. nops(op(1,b)))
        +sum('sum('LJ(op(j2,op(i,b)),b,j1)*I[j1,j2]',
          'j1' = 1 .. nops(op(1,b)))','j2' = 1 .. nops(op(1,b)));
    for u to nops(a) do  soln[i] := subs(x[u] = Y.u[n],soln[i]) od;
  od;
  RETURN(eval(soln))
end:
```

The call `Milstein([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);` returns the Milstein scheme for an N-dimensional SDE (1.42) with M-dimensional white noise which has drift coefficient components $a1,\ldots, aN$ and diffusion coefficient matrix $[b^{i,j}]$ with rows $[b11,\ldots,b1M],\ldots, [bN1,\ldots,bNM]$.

The output consists of the variables $YN[n]$, Delta$WM[n]$, Delta$[n]$ and $I[(j1, j2)]$. Here $YN[n]$ denotes the Milstein approximation to $x[N]$ at the nth step, Delta$WM[n]$ denotes the increment in the M-dimensional Wiener process at the nth step, Delta$[n]$ denotes the step size at the nth step, and $I[(j1, j2)]$ denotes the double Ito integral (1.45).

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process $W_t = (W_t^1, W_t^2)$, given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^2 \\ X_t^1 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ vX_t^1 \end{pmatrix} dW_t^2,$$

i.e. with drift components $a^1(t, x_1, x_2) = x_2$, $a^2(t, x_1, x_2) = x_1$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ vX_t^1 \end{pmatrix},$$

where r and s are constants.

```
> Milstein([x[2],x[1]],[[r,u],[s,v*x[1]]]);

table([
    1 = (Y1[n + 1] =  Y1[n] + Y2[n] Delta[n]
                      + r Delta W1[n] + u Delta W2[n])

    2 = (Y2[n + 1] = Y2[n] + Y1[n] Delta[n]
                     + s Delta W1[n] + v Y1[n] Delta W2[n]
                     + r v I[1, 2] + u v I[2, 2])
      ])
```

The resulting Milstein scheme is

$$\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W_n^1 + \begin{pmatrix} u \\ vY_n^1 \end{pmatrix} \Delta W_n^2 + \begin{pmatrix} 0 \\ r\,v \end{pmatrix} I_{(1,2)};n + \begin{pmatrix} 0 \\ u\,v \end{pmatrix} I_{(2,2)};n.$$

1.9.3 Milstein scheme for commutative noise

Recall from (1.37) that the SDE is said to have commutative noise (of the first kind) when

$$L^{j_1} b^{k,j_2}(t,x) \equiv L^{j_2} b^{k,j_1}(t,x)$$

for $k = 1, \ldots, n$ and $j_1, j_2 = 1, \ldots, m$. Then the identities

$$\int_{t_n}^{t_{n+1}} \int_{t_n}^{t} dW_s^{j_1}\, dW_t^{j_2} + \int_{t_n}^{t_{n+1}} \int_{t_n}^{t} dW_s^{j_2}\, dW_t^{j_1} = \Delta W_n^{j_1}\, \Delta W_n^{j_2} \tag{1.46}$$

for $j_1, j_2 = 1, \ldots, m$ with $j_1 \neq j_2$ can be used to simplify the Milstein scheme (1.44) to give

$$X_{n+1}^i = X_n^i + a^i(t_n, X_n)\, \Delta_n + \sum_{j=1}^{m} b^{i,j}(t_n, X_n) \Delta W_n^j \tag{1.47}$$

$$+\frac{1}{2}\sum_{j_1=1}^{m} L^{j_1} b^{i,j_1}(t_n, X_n)\left\{(\Delta W_n^{j_1})^2 - \Delta_n\right\}$$
$$+\frac{1}{2}\sum_{\substack{j_1,j_2=1 \\ j_1\neq j_2}}^{m} L^{j_1} b^{i,j_2}(t_n, X_n)\Delta W_n^{j_1}\,\Delta W_n^{j_2}$$

which is called the *Milstein scheme for commutative noise.*

The routine `stochastic[milcomm]` constructs the Milstein scheme for SDEs with commutative noise. Input and output format are the same as for the `stochastic[Milstein]` routine.

```
stochastic[milcomm]:=proc(a::list(algebraic),b::list(list(algebraic)))
  local u,i,l,soln;
  for i to nops(a) do
    soln[i]:=Y.i[n+1]=Y.i[n]+L0(x[i],a,b)*Delta[n]
           +sum('LJ(x[i],b,j)*Delta.W.j[n]','j'=1..nops(op(1,b)))
           +1/2*sum('sum('LJ(op(j2,op(i,b)),b,j1)*
                 (Delta.W.j1[n])*(Delta.W.j2[n])',
                 'j1'=1..nops(op(1,b)))','j2'=1..nops(op(1,b)));
    for l to nops(op(1,b)) do
      soln[i]:=subs((Delta.W.l[n])^2=((Delta.W.l[n])^2-Delta[n]),
                                      soln[i]) od;
    for u to nops(a) do
      soln[i]:=subs(x[u]=Y.u[n],soln[i]);
  od; od;
  RETURN(eval(soln));
end:
```

EXAMPLE: The scalar bilinear Ito SDE with two independent Wiener processes,

$$dX_t = aX_t\,dt + bX_t\,dW_t^1 + cX_t\,dW_t^2$$

has commutative noise.

```
> comm1([b*x[1],c*x[1]]);

      "This system exhibits commutative noise of the first kind"
```

Thus we can apply the `stochastic[milcomm]` routine.

```
> milcomm([a*x[1]], [[b*x[1],c*x[1]]]);

  table([
        1 = (Y1[n + 1] = Y1[n] + a Y1[n] Delta[n]
         + b Y1[n] DeltaW1[n] + c Y1[n] DeltaW2[n]
                2                   2
         + 1/2 b  Y1[n] (DeltaW1[n]  - Delta[n])
```

```
  + c Y1[n] b DeltaW2[n] DeltaW1[n]
        2                  2
  + 1/2 c  Y1[n] (DeltaW2[n]  - Delta[n]))
 ])
```

i.e., the Milstein scheme for commutative noise here is

$$\begin{aligned} X_{n+1} = X_n + aX_n\,\Delta_n + bX_n\,\Delta W_n^1 + cX_n\,\Delta W_n^1 \\ +\frac{1}{2}b^2X_n\,\left\{(\Delta W_n^1)^2 - \Delta_n\right\} + \frac{1}{2}c^2X_n\,\left\{(\Delta W_n^2)^2 - \Delta_n\right\} \\ +bcX_n\,\Delta W_n^1\,\Delta W_n^2 \end{aligned}$$

1.9.4 Order 1.5 strong stochastic Taylor scheme

The ith component of the *order* 1.5 *strong Taylor scheme* for the Ito SDE (1.42) is given by

$$\begin{aligned} Y_{n+1}^i = Y_n^i + a^i\,\Delta_n + \frac{1}{2}\,L^0a^i\,\Delta_n^2 \\ +\sum_{j=1}^{M}\left(b^{i,j}\,\Delta W_n^j + L^0b^{i,j}I_{(0,j);n} + L^ja^i\,I_{(j,0);n}\right) \\ +\sum_{j_1,j_2=1}^{M} L^{j_1}b^{i,j_2}I_{(j_1,j_2);n} + \sum_{j_1,j_2,j_3=1}^{M} L^{j_1}L^{j_2}b^{i,j_3}I_{(j_1,j_2,j_3);n}, \end{aligned} \tag{1.48}$$

for $i = 1, \ldots, N$, where $I_{(j_1,j_2,j_3);n}$ is the multiple Ito integral

$$I_{(j_1,j_2,j_3);n} = \int_{t_n}^{t_{n+1}}\int_{t_n}^{s_1}\int_{t_n}^{s_2} dW_{s_3}^{j_1}\,dW_{s_2}^{j_1}\,dW_{s_1}^{j_2}, \tag{1.49}$$

with the special case

$$I_{(j,j,j);n} = \frac{1}{2}\left\{\frac{1}{3}\left(\Delta W_n^j\right)^2 - \Delta_n\right\}\Delta W_n^j$$

for $j_1 = j_2 = j_3 = j$. Also

$$I_{(0,j);n} = \Delta W_n^{j_1}\Delta_n - I_{(j,0);n},$$

where the random variable $\Delta Z_n^j := I_{(j,0);n}$ is $N(0;\frac{1}{3}\Delta_n^3)$–distributed and has covariance $E(\Delta Z_n^j\Delta W_n^j) = \frac{1}{2}\Delta_n^2$.

The routine `stochastic[Taylor1hlf]` constructs the strong order 1.5 Taylor scheme for an Ito SDE (1.42).

```
stochastic[Taylor1hlf]:=proc(a::list(algebraic),
                             b::list(list(algebraic)))
local u,i,soln;
  for i to nops(a) do
    soln[i] := Y.i[n+1] =
        Y.i[n]+a[i]*Delta[n]+1/2*L0(a[i],a,b)*Delta[n]^2
        +sum('op(j,op(i,b))*Delta*W.j[n]+L0(op(j,op(i,b)),a,b)*I[0,j]
        +LJ(a[i],b,j)*I[j,0]','j' = 1 .. nops(op(1,b)))
        +sum('sum('LJ(op(j2,op(i,b)),b,j1)*I[j1,j2]',
          'j1' = 1 .. nops(op(1,b)))','j2' = 1 .. nops(op(1,b)))+sum(
          'sum('sum('LJ(LJ(op(p3,op(i,b)),b,p2),b,p1)*I[p1,p2,p3]',
          'p1' = 1 .. nops(op(1,b)))','p2' = 1 .. nops(op(1,b)))',
          'p3' = 1 .. nops(op(1,b)));
    for u to nops(a) do  soln[i] := subs(x[u] = Y.u[n],soln[i]) od;
  od;
  RETURN(eval(soln))
end:
```

The call `Taylor1hlf([a1,..,aN], [[b11,..,b1M],.., [bN1,..,bNM]]);` returns the strong order 1.5 approximation for an N-dimensional SDE (1.42) with M-dimensional noise, which has drift coefficient components $a1, \ldots, aN$ and diffusion matrix with rows $[b11, \ldots, b1M]$, …, $[bN1, \ldots, bNM]$.

The routine returns the variables $YN[n]$, Delta$WM[n]$, Delta$[n]$, $I[(j1, j2)]$, and $I[(j1, j2, j3)]$. Here $YN[n]$ denotes the order 1.5 strong stochastic Taylor approximation to $x[N]$ at the nth step, Delta$WM[n]$ denotes the change in the M-dimensional Wiener process at the nth step, Delta$[n]$ denotes the step size at the nth step, while $I[(j1, j2)]$ and $I[(j1, j2, j3)]$ denote the multiple Ito integrals (1.45) and (1.49).

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process $W_t = (W_t^1, W_t^2)$, given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^2 \\ X_t^1 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ v \end{pmatrix} dW_t^2,$$

i.e. with drift components $a^1(t, x_1, x_2) = x_2$, $a^2(t, x_1, x_2) = x_1$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where r, s, u and v are constants.

```
> Taylor1hlf([x[2],x[1]],[[r,u],[s,v]]);

table([
```

```
                                                                    2
1 = (Y1[n + 1] = Y1[n] + Y2[n] Delta[n] + 1/2 Y1[n] Delta[n]
      + r Delta W1[n] + s I[1, 0] + u Delta W2[n] + v I[2, 0])

                                                                    2
2 = (Y2[n + 1] = Y2[n] + Y1[n] Delta[n] + 1/2 Y2[n] Delta[n]
      + s Delta W1[n] + r I[1, 0] + v Delta W2[n] + u I[2, 0])

    ])
```

The resulting order 1.5 strong Taylor scheme scheme is

$$\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n + \frac{1}{2} \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n^2 + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W_n^1 + \begin{pmatrix} s \\ r \end{pmatrix} \Delta W_n^2 + \begin{pmatrix} u \\ v \end{pmatrix} I_{(1,0);n} + \begin{pmatrix} v \\ u \end{pmatrix} I_{(2,0);n},$$

1.9.5 Order 2.0 strong stochastic Taylor scheme

The *order* 2.0 *strong Taylor scheme* for the N-dimensional Stratonovich SDE with an M-dimensional Wiener process

$$dX_t^i = \underline{a}^i(t, X_t)\, dt + \sum_{j=1}^{M} b^{i,j}(t, X_t) \circ dW_t^j, \qquad i = 1, \ldots, N, \tag{1.50}$$

is given componentwise by

$$\begin{aligned} Y_{n+1}^i = Y_n^i &+ \underline{a}^i \, \Delta_n + \frac{1}{2} \underline{L}^0 \underline{a}^i \, \Delta_n^2 \\ &+ \sum_{j=1}^{m} \left(b^{i,j} \, \Delta W_n^j + \underline{L}^0 b^{i,j} J_{(0,j);n} + \underline{L}^j \underline{a}^i \, J_{(j,0);n} \right) \\ &+ \sum_{j_1,j_2=1}^{m} \Big(\underline{L}^{j_1} b^{i,j_2} J_{(j_1,j_2);n} + \underline{L}^0 \underline{L}^{j_1} b^{i,j_2} J_{(0,j_1,j_2);n} \\ &\qquad + \underline{L}^{j_1} \underline{L}^0 b^{i,j_2} J_{(j_1,0,j_2);n} + \underline{L}^{j_1} \underline{L}^{j_2} \underline{a}^i \, J_{(j_1,j_2,0);n} \Big) \\ &+ \sum_{j_1,j_2,j_3=1}^{m} \underline{L}^{j_1} \underline{L}^{j_2} b^{i,j_3} J_{(j_1,j_2,j_3);n} \\ &+ \sum_{j_1,j_2,j_3,j_4=1}^{m} \underline{L}^{j_1} \underline{L}^{j_2} \underline{L}^{j_3} b^{i,j_4} J_{(j_1,j_2,j_3,j_4);n} \end{aligned} \tag{1.51}$$

for $i = 1, \ldots, N$. The $J_{(j_1,j_2);n}$ and $J_{(j_1,j_2,j_3);n}$ expressions here denote the corresponding double and triple Stratonovich integrals with respect to the components of the given Wiener process.

The routine `stochastic[Taylor2]` constructs the order 2.0 strong stochastic Taylor scheme for the Stratonovich SDE (1.50).

```
stochastic[Taylor2]:=proc(a::list(algebraic),b::list(list(algebraic)))
local u,i,soln;
  for i to nops(a) do
    soln[i] := Y.i[n+1] = Y.i[n]+correct(a,b,i)*Delta[n]+
      1/2*SL0(correct(a,b,i),a,b)*Delta[n]^2+
      sum('op(j,op(i,b))*Delta*W.j[n]+SL0(op(j,op(i,b)),a,b)*J[0,j]+
      LJ(correct(a,b,i),b,j)*J[j,0]','j' = 1 .. nops(op(1,b)))
      +sum('sum('LJ(op(j2,op(i,b)),b,j1)*J[j1,j2]+
      SL0(LJ(op(j2,op(i,b)),b,j1),a,b)*J[0,j1,j2]+
      LJ(SL0(op(j2,op(i,b)),a,b),b,j1)*J[j1,0,j2]+
      LJ(LJ(correct(a,b,i),b,j2),b,j1)*J[j1,j2,0]',
      'j1' = 1 .. nops(op(1,b)))',
      'j2' = 1 .. nops(op(1,b)))+sum(
      'sum('sum('LJ(LJ(op(p3,op(i,b)),b,p2),b,p1)*J[p1,p2,p3]',
      'p1' = 1 .. nops(op(1,b)))','p2' = 1 .. nops(op(1,b)))',
      'p3' = 1 .. nops(op(1,b)))+sum('sum('sum(
      'sum('LJ(LJ(LJ(op(m4,op(i,b)),b,m3),b,m2),b,m1)*J[m1,m2,m3,m4]',
      'm1' = 1 .. nops(op(1,b)))','m2' = 1 .. nops(op(1,b)))
      ','m3' = 1 .. nops(op(1,b)))','m4' = 1 .. nops(op(1,b)));
    for u to nops(a) do  soln[i] := subs(x[u] = Y.u[n],soln[i]) od
  od;
  RETURN(eval(soln))
end:
```

The call `Taylor2([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);` computes the order 2.0 strong stochastic Taylor approximation for the N-dimensional Stratonovich SDE (1.50) with M-dimensional noise which has drift coefficient components $a1,\ldots,aN$ and diffusion matrix with rows $[b11, \ldots, b1M]$, $\ldots$, $[bN1, \ldots, bNM]$.

The output gives the variables $YN[n]$, Delta$WM[n]$, Delta$[n]$, $J[(j1, j2)]$, $J[(j1, j2, j3)]$, and $J[(j1, j2, j3, j4)]$. here $YN[n]$ denotes the strong order 2.0 stochastic Taylor approximation to $x[N]$ at the nth step, Delta$WM[n]$ denotes the increment in the M-dimensional Wiener process at the nth step, Delta$[n]$ denotes the step size at the nth step, while $J[(j1, j2)]$, $J[(j1, j2, j3)]$, and $J[(j1, j2, j3, j4)]$ denote multiple Stratonovich integrals.

EXAMPLE: Consider the 2-dimensional Stratonovich SDE driven by a 2-dimensional Wiener process $W_t = (W_t^1, W_t^2)$, given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^2 \\ X_t^1 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} \circ dW_t^1 + \begin{pmatrix} u \\ v \end{pmatrix} \circ dW_t^2,$$

that is with drift components $\underline{a}^1(t, x_1, x_2) = x_2$, $\underline{a}^2(t, x_1, x_2) = x_1$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where r, s, u and v are constants.

```
> Taylor2([x[2],x[1]],[[r,u],[s,v]]);

table([
                                                                    2
  1 = (Y1[n + 1] = Y1[n] + Y2[n] Delta[n] + 1/2 Y1[n] Delta[n]
      + r Delta W1[n] + s J[1, 0] + u Delta W2[n] + v J[2, 0])
                                                                    2
  2 = (Y2[n + 1] = Y2[n] + Y1[n] Delta[n] + 1/2 Y2[n] Delta[n]
      + s Delta W1[n] + r J[1, 0] + v Delta W2[n] + u J[2, 0])

      ])
```

The resulting order 2.0 strong Stratonovich Taylor scheme scheme is

$$\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n + \frac{1}{2}\begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n^2 + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W_n^1 + \begin{pmatrix} s \\ r \end{pmatrix} J_{(1,0);n} + \begin{pmatrix} u \\ v \end{pmatrix} \Delta W_n^2 + \begin{pmatrix} v \\ u \end{pmatrix} J_{(2,0);n}.$$

1.10 Weak Numerical Schemes

Weak Taylor schemes of order 1.0, 2.0 and 3.0 along with various simplifications will be considered for an N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t^i = a^i(t, X_t)\, dt + \sum_{j=1}^{M} b^{i,j}(t, X_t)\, dW_t^j, \qquad i = 1, \ldots, N. \tag{1.52}$$

1.10.1 Weak Euler scheme

The weak stochastic Taylor scheme of order 1.0 for the SDE (1.52) is known as the *weak Euler scheme* and has the componentwise form

$$Y_{n+1}^i = Y_n^i + a^i\,\Delta_n + \sum_{j=1}^{M} b^{i,j}\,\Delta W_n^j, \qquad i = 1, \ldots, N, \tag{1.53}$$

where $\Delta_n = t_{n+1} - t_n$ and $\Delta W_n^j = W_{t_{n+1}}^j - W_{t_n}^j$.

Since only the probability measure induced by the solution process X_t needs to be approximated for weak convergence, the Gaussian increments ΔW_n^j in (1.53) can be replaced by simpler random variables $\Delta\widehat{W}^j$ with similar lower moment properties that are easier to generate. This leads to the *simplified weak Euler scheme*

$$Y_{n+1}^i = Y_n^i + a^i\,\Delta_n + \sum_{j=1}^{M} b^{i,j}\,\Delta\widehat{W}_n^j, \qquad i = 1, \ldots, N, \tag{1.54}$$

for $i = 1, \ldots, N$, where the $\Delta\widehat{W}_n^j$ are independent two-point distributed random variables with

$$P\left(\Delta\widehat{W}_n^j = \pm\sqrt{\Delta_n}\right) = \frac{1}{2}$$

for $j = 1, 2, \ldots, M$.

The routine `stochastic[wkeuler]` constructs the (simplified) weak Euler scheme for the SDE (1.52).

```
stochastic[wkeuler]:=proc(a::list(algebraic),b::list(list(algebraic)))
local u,i,soln;
  for i to nops(a) do
     soln[i] := Y.i[n+1] = Y.i[n]+L0(x[i],a,b)*Delta[n]+
       sum('LJ(x[i],b,j)*Delta*Ws.j[n]','j' = 1 .. nops(op(1,b)));
       for u to nops(a) do  soln[i] := subs(x[u] = Y.u[n],soln[i]) od
  od;
  RETURN(eval(soln))
end:
```

The call `wkeuler([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);` computes the (simplified) weak Euler scheme for an N-dimensional SDE with M-dimensional noise which has drift coefficients $a1$, ..., aN and diffusion matrix with rows $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$.

The output consists of the variables $YN[n]$, Delta$WsM[n]$ and Delta$[n]$. Here $YN[n]$ denotes the first order simplified weak approximation to $x[N]$ at the

nth step, Delta$WsM[n]$ denotes the increment in the M-dimensional noise process at the nth step (note here that $WsM[n]$ does not need to denote a standard Wiener processes, but can instead be independent random variables as described above) and Delta$[n]$ denotes the step size at the nth step.

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process $W_t = (W_t^1, W_t^2)$, given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ v \end{pmatrix} dW_t^2,$$

i.e. with drift components $a^1(t, x_1, x_2) = x_1$, $a^2(t, x_1, x_2) = x_2$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where r is a constant.

```
> wkeuler([x[1],x[2]],[[r,u],[s,v]]);

table([
        1 = (Y1[n + 1] = Y1[n] + Y1[n] Delta[n]
                         + r Delta Ws1[n] + u Delta Ws2[n])

        2 = (Y2[n + 1] = Y2[n] + Y2[n] Delta[n]
                         + s Delta Ws1[n] + v Delta Ws2[n])
      ])
```

The resulting simplified weak Euler scheme is

$$\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n + \begin{pmatrix} r \\ s \end{pmatrix} \Delta\widehat{W}_n^1 + \begin{pmatrix} u \\ v \end{pmatrix} \Delta\widehat{W}_n^2.$$

1.10.2 Order 2.0 weak stochastic Taylor scheme

The *order* 2.0 *weak stochastic Taylor scheme* for the SDE (1.52) takes the form

$$\begin{aligned} Y_{n+1}^i = Y_n^i + a^i\,\Delta_n + \frac{1}{2} L^0 a^i\,\Delta_n^2 \\ + \sum_{j=1}^{M} \left\{ b^{i,j} \Delta W_n^j + L^0 b^{k,j}\, I_{(0,j);n} + L^j a^i\, I_{(j,0);n} \right\} \\ + \sum_{j_1,j_2=1}^{m} L^{j_1} b^{k,j_2}\, I_{(j_1,j_2);n} \end{aligned} \tag{1.55}$$

for $i = 1, \ldots, N$. Here multiple Ito integrals involving different components of the Wiener process are used. Since these are generally not easy to generate, the above scheme is more of theoretical interest than of practical use. However, for weak convergence the multiple integrals can be replaced by simpler random variables, which leads to the *simplified order* 2.0 *weak Taylor scheme*

$$Y_{n+1}^i = Y_n^i + a^i\,\Delta_n + \frac{1}{2}\,L^0 a^i\,\Delta_n^2 \tag{1.56}$$
$$+\sum_{j=1}^{M}\left\{b^{i,j} + \frac{1}{2}\,\Delta_n\,\left(L^0 b^{i,j} + L^j a^i\right)\right\}\,\Delta\widehat{W}_n^j$$
$$+\frac{1}{2}\sum_{j_1,j_2=1}^{M} L^{j_1} b^{i,j_2}\left(\Delta\widehat{W}_n^{j_1}\Delta\widehat{W}_n^{j_2} + V_{(j_1,j_2);n}\right).$$

Here the $\Delta\widehat{W}_n^j$ for $j = 1, 2, \ldots, M$ are independent three-point distributed random variables with

$$P\left(\Delta\widehat{W}_m^j = \pm\sqrt{3\Delta_n}\right) = \frac{1}{6}, \qquad P\left(\Delta\widehat{W}_n^j = 0\right) = \frac{2}{3}. \tag{1.57}$$

and the $V_{(j_1,j_2);n}$ are independent two-point distributed random variables with

$$P\left(V_{(j_1,j_2);n} = \pm\Delta_n\right) = \frac{1}{2} \tag{1.58}$$

for $j_2 = 1, \ldots, j_1 - 1$, with

$$V_{(j_1,j_1);n} = -\Delta_n \tag{1.59}$$

and

$$V_{(j_1,j_2);n} = -V_{(j_2,j_1);n} \tag{1.60}$$

for $j_2 = j_1 + 1, \ldots, M$ and $j_1 = 1, \ldots, M$.

The routine `stochastic[wktay2]` constructs the simplified stochastic Taylor scheme of weak order 2.0.

```
stochastic[wktay2]:=proc(a::list(algebraic),b::list(list(algebraic)))
local u,i,soln;
  for i to nops(a) do
    soln[i] := Y.i[n+1] =
      Y.i[n]+a[i]*Delta[n]+1/2*L0(a[i],a,b)*Delta[n]^2+
      sum('(op(j,op(i,b))+1/2*Delta[n]*(L0(op(j,op(i,b)),a,b)+
      LJ(a[i],b,j)))*Delta*Ws.j[n]','j' = 1 .. nops(op(1,b)))+1/2*
      sum('sum('LJ(op(j2,op(i,b)),b,j1)*(Delta^2*Ws.j1[n]*Ws.j2[n]+
      V[j1,j2])','j1' = 1 .. nops(op(1,b)))',
      'j2' = 1 .. nops(op(1,b)));
    for u to nops(a) do  soln[i] := subs(x[u] = Y.u[n],soln[i]) od;
  od;
  RETURN(eval(soln))
end:
```

The call `wktay2([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);` returns the simplified order 2.0 weak stochastic Taylor scheme for an N-dimensional SDE with M-dimensional noise which has drift coefficients $a1$, ...,aN and diffusion matrix with rows $[b11, \ldots, b1M]$,..., $[bN1, \ldots, bNM]$.

The output consists of the variables $YN[n]$, Delta$WsM[n]$, $V[(j1, j2)]$, and Delta$[n]$. Here $YN[n]$ denotes the 2nd order simplified weak approximation to $x[N]$ at the nth step, Delta$WsM[n]$ denotes the change in the M-dimensional noise process at the nth step (note here that $WsM[n]$ does not denote standard Wiener processes, but the three-point random variables described above), $V[(j1, j2)]$ denotes the independent two-point random variables described above, and Delta$[n]$ denotes the step size at the nth step.

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process $W_t = (W_t^1, W_t^2)$, given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ v \end{pmatrix} dW_t^2,$$

that is with drift components $a^1(t, x_1, x_2) = x_1$, $a^2(t, x_1, x_2) = x_2$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where r, s, u and v are constants.

```
> wktay2([x[1],x[2]],[[r,u],[s,v]]);

table([
                                                                    2
        1 = (Y1[n + 1] = Y1[n] + Y1[n] Delta[n] + 1/2 Y1[n] Delta[n]
                               + (r + 1/2 Delta[n] r) Delta Ws1[n]
                               + (u + 1/2 Delta[n] u) Delta Ws2[n])
                                                                    2
        2 = (Y2[n + 1] = Y2[n] + Y2[n] Delta[n] + 1/2 Y2[n] Delta[n]
                               + (s + 1/2 Delta[n] s) Delta Ws1[n]
                               + (v + 1/2 Delta[n] v) Delta Ws2[n])
        ])
```

The resulting order 2.0 weak stochastic Taylor scheme is

$$\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} \Delta_n + \frac{1}{2}\begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} \Delta_n^2$$

$$+ \begin{pmatrix} r + \frac{1}{2} r \Delta_n \\ s + \frac{1}{2} s \Delta_n \end{pmatrix} \Delta\widehat{W}_n^1 + \begin{pmatrix} u + \frac{1}{2} u \Delta_n \\ v + \frac{1}{2} v \Delta_n \end{pmatrix} \Delta\widehat{W}_n^2.$$

1.10.3 Order 3.0 weak stochastic Taylor scheme

The *order* 3.0 *weak stochastic Taylor scheme* for the SDE (1.52) has the form

$$Y_{n+1}^i = Y_n^i + a^i\,\Delta_n + \sum_{j=1}^{m} b^{k,j}\,\Delta W_n^j + \sum_{j=0}^{m} L^j a^i\, I_{(j,0);n} \tag{1.61}$$
$$+ \sum_{j_1=0}^{m}\sum_{j_2=1}^{m} L^{j_1} b^{k,j_2}\, I_{(j_1,j_2);n} + \sum_{j_1,j_2=0}^{m} L^{j_1} L^{j_2} a^i\, I_{(j_1,j_2,0);n}$$
$$+ \sum_{j_1,j_2=0}^{m}\sum_{j_3=1}^{m} L^{j_1} L^{j_2} b^{k,j_3}\, I_{(j_1,j_2,j_3);n}.$$

Various simplifications are possible in special cases that avoid the need to generate the multiple stochastic integrals. See Chapter 14.3 of Kloeden and Platen [13]

The routine `stochastic[wktay3]` constructs stochastic Taylor schemes of weak order 3.0.

```
stochastic[wktay3]:=proc(a::list(algebraic),b::list(list(algebraic)))
local u,i,soln;
  for i to nops(a) do
    soln[i] := Y.i[n+1] = Y.i[n]+a[i]*Delta[n]+
      sum('op(j,op(i,b))*Delta*W.j[n]','j' = 1 .. nops(op(1,b)))+
      sum('MLJ(a[i],a,b,j0)*I[j0,0]','j0' = 0 .. nops(op(1,b)))+
      sum('sum('MLJ(op(j2,op(i,b)),a,b,j1)*I[j1,j2]',
      'j2' = 1 .. nops(op(1,b)))','j1' = 0 .. nops(op(1,b)))+
      sum('sum('MLJ(MLJ(a[i],a,b,k2),a,b,k1)*I[k1,k2,0]',
      'k1' = 0 .. nops(op(1,b)))','k2' = 0 .. nops(op(1,b)))+sum(
      'sum('sum('MLJ(MLJ(op(m3,op(i,b)),a,b,m2),a,b,m1)*I[m1,m2,m3]',
      'm3' = 1 .. nops(op(1,b)))','m2' = 0 .. nops(op(1,b)))',
      'm1' = 0 .. nops(op(1,b)));
    for u to nops(a) do  soln[i] := subs(x[u] = Y.u[n],soln[i]) od;
  od;
  RETURN(eval(soln))
end:
```

The call `wktay3([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);` returns the order 3.0 weak stochastic Taylor scheme for an N-dimensional SDE with M-dimensional noise which has drift coefficients $a1$, ..., aN and diffusion matrix with rows $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$.

The output consists of the variables $YN[n]$, Delta$WM[n]$, $I[(j1, j2)]$, $I[(j1,$ $j2,$ $j3)]$ and Delta$[n]$. Here $YN[n]$ denotes the third order weak approximation to $x[N]$ at the nth step, Delta$WM[n]$ denotes the increment in the M-dimensional Wiener process at the nth step, $I[(j1, j2)]$ and $I[(j1, j2, j3)]$

denote multiple Ito integrals, and Delta[n] denotes the step size at the nth step.

EXAMPLE: Consider the 2-dimensional SDE driven by a 2-dimensional Wiener process $W_t = (W_t^1, W_t^2)$, given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^2 \\ X_t^1 \end{pmatrix} dt + \begin{pmatrix} r \\ s \end{pmatrix} dW_t^1 + \begin{pmatrix} u \\ v \end{pmatrix} dW_t^2,$$

i.e. with drift components $a^1(t, x_1, x_2) = x_2$, $a^2(t, x_1, x_2) = x_1$ and the constant diffusion coefficient vectors

$$b^1 = \begin{pmatrix} b^{1,1} \\ b^{2,1} \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}, \qquad b^2 = \begin{pmatrix} b^{1,2} \\ b^{2,2} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where r, s, u and v are constants.

```
> wktay3([x[2],x[1]],[[r,u],[s,v]]);

table([
  1 = (Y1[n + 1] = Y1[n] + Y2[n] Delta[n] + r Delta W1[n]
                 + u Delta W2[n] + Y1[n] I[0, 0] + s I[1, 0]
                 + v I[2, 0] + Y2[n] I[0, 0, 0]
                 + r I[1, 0, 0] + u I[2, 0, 0])

  2 = (Y2[n + 1] = Y2[n] + Y1[n] Delta[n] + s Delta W1[n]
                 + v Delta W2[n] + Y2[n] I[0, 0] + r I[1, 0]
                 + u I[2, 0] + Y1[n] I[0, 0, 0]
                 + s I[1, 0, 0] + v I[2, 0, 0])
     ])
```

The resulting order 3.0 weak stochastic Taylor scheme scheme is

$$\begin{aligned}\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W_n^1 + \begin{pmatrix} u \\ v \end{pmatrix} \Delta W_n^2 \\ + \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} I_{(0,0);n} + \begin{pmatrix} s \\ r \end{pmatrix} I_{(1,0);n} + \begin{pmatrix} v \\ u \end{pmatrix} I_{(2,0);n} \\ + \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} I_{(0,0,0);n} + \begin{pmatrix} r \\ s \end{pmatrix} I_{(1,0,0);n} + \begin{pmatrix} u \\ v \end{pmatrix} I_{(2,0,0);n}.\end{aligned}$$

References

1. L. Arnold, *Stochastic Differential Equations.* Wiley, New York, 1974.

2. S.S. Artemiev and T.A. Averina, *Numerical Analysis of Systems of Ordinary and of Stochastic Differential Equations.* VSP, Utrecht, 1997.
3. R.E. Crandall, *Topics in Advanced Scientific Computation,* Springer–Verlag, Heidelberg,1996.
4. S.O. Cyganowski, Solving Stochastic Differential Equations with Maple, *MapleTech Newsletter* **3**(2) (1996), 38–40.
5. S.O. Cyganowski, *A MAPLE Package for stochastic differential equations,* in "Computational Techniques and Applications: CTAC95" (Editors A. Easton, & R. May), World Scientific Publishers, Singapore, 1996, 223–230.
6. S. Cyganowski and P.E. Kloeden, Stochastic stability examined through MAPLE, in *Proc. 15th IMACS World Congress,* Volume 1: Computational Mathematics (Editor: A. Sydow), Wissenschaft & Technik Verlag, Berlin, 1997, 437–432.
7. S. Cyganowski, P.E. Kloeden and J. Ombach, *From Elementary Probability to Stochastic DEs with MAPLE,* Springer–Verlag, Heidelberg, 2001.
8. S. Cyganowski, P.E. Kloeden and T. Pohl, *MAPLE for stochastic differential equations* WIAS Berlin, Preprint Nr. 453, 1998. Availability: Postscript 467 KB, `http://www.wias-berlin.de/publications/preprints/453`
9. T. Gard, *Introduction to Stochastic Differential Equations,* Marcel–Dekker, New York, 1988.
10. W. Gander and J. Hrebicek, *Solving Problems in Scientific Computing using Maple and Matlab,* Second Edition, Springer–Verlag, Heidelberg, 1995.
11. W.S. Kendall, Computer algebra and stochastic calculus, *Notices Amer. Math. Soc.* **37** (1990), 1254–1256.
12. P.E. Kloeden, Stochastic differential equations in environmental modelling and their numerical solution, in *Stochastic and Statistical Modelling with Groundwater and Surface Water Applications,* (Editor: K. Hipel), Kluwer Academic Publ., Dordrecht, 1994, 21–32.
13. P.E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* Springer-Verlag, Heidelberg, 1992; second revised printing 1999.
14. P.E. Kloeden and E. Platen, A survey of numerical methods for stochastic differential equations, *J. Stoch. Hydrol. Hydraul.* **3** (1989), 155-178.
15. P.E. Kloeden and E. Platen, Numerical methods for stochastic differential equations, in *Stochastic Modelling and Nonlinear Dynamics: Applications to Mechanical Systems*, (Editor: W. Kliemann), CRC Press, 1994, S. 437–461.
16. P.E. Kloeden, E. Platen and H. Schurz, *Numerical Solution of Stochastic Differential Equations through Computer Experiments*, Springer-Verlag, Heidelberg, 1993.
17. P.E. Kloeden, E. Platen and H. Schurz, The numerical solution of nonlinear stochastic dynamical systems: a brief introduction, *J. Bifurcation & Chaos* **1** (1991), 277-286.
18. P.E. Kloeden and W.D. Scott, Construction of Stochastic Numerical Schemes through Maple, *MapleTech Newsletter* **10** (1993), 60–65.
19. G.N. Milstein, *Numerical Integration of Stochastic Differential Equations*, Kluwer, Dordrecht, 1995.
20. G.G. Milstein and M.V. Tret'yakov, Numerical Solution of Differential Equations with Coloured Noise, *J. Stat. Physics,* **77** (1994) 691–715.
21. E. Platen, Numerical methods for stochastic differential equations, *Acta Numerica,* (1999) 197–246.

22. E. Valkeila, Computer algebra and stochastic analysis, some possibilities, *CWI Quarterly* **4** (1991), 229–238.
23. Xu Kedai, Stochastic pitchfork bifurcation: numerical simulations and symbolic calculations using MAPLE, *Mathematics and Computers in Simulation* **38** (1995), 199–207.