MAPLE for Stochastic Differential Equations*†

Sasha Cyganowski

School of Computing and Mathematics
Deakin University, Geelong 3217, Australia
e-mail: sash@deakin.edu.au

Peter E. Kloeden and Thomas Pohl

Fachbereich Mathematik, Johann Wolfgang Goethe Universität
D–60054 Frankfurt am Main, Germany
e-mail: kloeden@math.uni-frankfurt.de

*The main part of this report was written while P.E. Kloeden was the Deputy Leader of the Stochastic Algorithms and Nonparametric Statistics Research Group at the Weierstrass Institute.

†1991 Mathematics Subject Classification Primary: 60H10, 60H30, 68Q40 Secondary: 65H05, 93E15, 93E30

Contents

1 Introduction

1.1 MAPLE

This report provides an introduction and description of the MAPLE software package **with(stochastic)** which consists of MAPLE routines for stochastic calculus and stochastic differential equations. These are known to work for MapleV version 5 (for Windows) and MapleV version 5 (for Unix). Its CALLING SEQUENCE is

<**function**>**(args) stochastic**[<**function**>]**(args)**

It was originally developed by Sasha Cyganowski of Deakin University, Australia, initially as a Mathematics Honours project under the supervision of Professor P.E. Kloeden and then further extended and developed, partly with the assistance of Dr. Thomas Pohl. Comments via e-mail are welcome and can be sent to Sasha Cyganowski at **sash@deakin.edu.au** or to Peter Kloeden at **kloeden@math.uni-frankfurt.de**. A copy of the software can be obtained by email from either of these addresses.

The stochastic package contains routines useful for finding explicit solutions of Stochastic Differential Equations(SDEs), and routines useful for constructing numerical schemes up to strong order 2.0 and weak order 3.0. Other features include a routine which converts SDEs with white noise into coloured noise form, and routines which check whether an SDE has commutative noise of the first and/or second kind. Some other useful routines are also available, in particular for the operators procedures $L0$ and LJ are included so that users can easily construct numerical schemes other than those already available. For more information on a particular function type the command <**function**>.

The general schemes, solutions and conditions used in the coding of the stochastic package can be found in

Kloeden, P.E, Platen, E.: *Numerical Solution of Stochastic Differential Equations*, (Springer-Verlag, 1992, Second Edition, 1994).

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.

The functions available are:

`linearsde`	`MLJ`	`explicit`	`L0`	`Euler`
`Milstein`	`Taylor1hlf`	`SL0`	`correct`	`Taylor2`

```
wkeuler          wktay2           reducible     wktay3          colour
comm1            comm2            LJ            momenteqn       sphere
pmatrix2pvector  pvector2pmatrix  ap            pa              bpb
```

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

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

Note the stochastic numerical routines require the drift and diffusion coefficients of an SDE to be entered in the variables $x[N]$ and t, where $x[N]$ are the state variables of the N-dimensional SDE and t denotes time. The routines **linearsde**, **reducible**, and **explicit** require the drift and diffusion coefficients to be entered in the variables x and t.

1.2 Stochastic Differential Equations

We consider a the N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t = a(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t)\, dW_t^j \tag{1.1}$$

and its Stratonovich counterpart

$$dX_t = \underline{a}(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t) \circ dW_t^j \tag{1.2}$$

where $\underline{a}$ is defined componentwise by

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

for $i = 1, \ldots, N$.

In many of the MAPLE procedures that follow we will use the parameters $a1$,..., aN to denote the drift coefficients and $[b11, \ldots, b1M]$,..., $[bN1, \ldots, bNM]$ to denote the diffusion coefficients.

NOTE: The indexing on the the bIJ here is the transpose of the usual matrix indexing; here the first index I corresponds to the column and the second J to the row due to the fact that $b^{i,j}$ is the jth component of the diffusion coefficient vector b^j associated with the jth component W_t^j of the Wiener process.

2 Explicitly Solvable Scalar SDE

In this Section we consider general scalar SDE

$$dX_t = a(t, X_t)\, dt + b(t, X_t)\, dW_t \tag{2.1}$$

that can be solved explicitly. First we look at the general linear scalar SDE

$$dX_t = (a_1(t)X_t + a_2(t))\ dt + (b_1(t)X_t + b_2(t))\ dW_t \tag{2.2}$$

for which an explicit solution is always available. Then we consider nonlinear scalar SDE that can be reduced to linear scalar SDE and hence solved explicitly.

2.1 Linearsde routine

The general form of a scalar *linear stochastic differential equation* is

$$dX_t = (a_1(t)X_t + a_2(t))\ dt + (b_1(t)X_t + b_2(t))\ dW_t \tag{2.3}$$

where the coefficients a_1, a_2, b_1, b_2 are specified functions of time t or constants. When $b_1(t) \equiv 0$ in (2.1) the SDE is the additive noise and when $b_1(t) \neq 0$ it has multiplicative noise.

In the general case the SDE can be solved with the integrating factor

$$\Phi_{t,t_0} = \exp\left(\int_{t_0}^{t} \left(a_1(s) - \frac{1}{2}\, b_1^2(s)\right)\ ds + \int_{t_0}^{t} b_1(s)\, dW_s\right), \tag{2.4}$$

and has the eplicit solution

$$X_t = \Phi_{t,t_0}\left(X_{t_0} + \int_{t_0}^{t} (a_2(s) - b_1(s)b_2(s))\, \Phi_{s,t_0}^{-1}\, ds + \int_{t_0}^{t} b_2(s)\Phi_{s,t_0}^{-1}\, dW_s\right). \tag{2.5}$$

In the additive noise case, the SDE reduces to

$$dX_t = (a_1(t)X_t + a_2(t))\ dt + b_2(t)\, dW_t, \tag{2.6}$$

and the integrating factor to

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

in which case the explicit solution is

$$X_t = \Phi_{t,t_0}\left(X_{t_0} + \int_{t_0}^{t} a_2(s)\Phi_{s,t_0}^{-1}\, ds + \int_{t_0}^{t} b_2(s)\Phi_{s,t_0}^{-1}\, dW_s\right). \tag{2.7}$$

The routine **stochastic[linearsde]** returns the explicit solution of an SDE (2.1) with linear drift coefficient $a(t,x) = \alpha(t)x$ or $\alpha(t)$ and linear diffusion coefficient $b(t,x)$ $= \beta(t)x$ or $\beta(t)$. Its CALLING SEQUENCE is

linearsde(a,b):

with PARAMETERS

```
        a -- algebraic, given in the variables x and t
        b -- algebraic, given in the variables x and t
```

representing the drift and diffusion coefficients of the SDE, the particular cases of which are identified by the routine. A suitable error message is returned if the coefficients of a nonlinear SDE are used.

```
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 (2.3) with drift coefficient a and 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.

The routine is used by the routine **stochastic[explicit]** and, in general, is not intended to be used on its own. It is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**, but can also be invoked via the call **stochas-**

tic[linearsde].

EXAMPLE:Consider the scalar linear SDE with aditive noise

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

with drift $a(t,x) = -x$ and diffusion coefficient $b(t,x) = 2$

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

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

SEE ALSO: **stochastic**, **stochastic[reducible]**, **stochastic[explicit].**

2.2 Reducible routine

A nonlinear scalar SDE

$$dX_t = a(X_t)\,dt + b(X_t)\,dW_t \tag{2.8}$$

can be reduced to a linear scalar stochastic differential

$$dY_t = dW_t \tag{2.9}$$

by a substitution

$$y = h(x) = \int^x \frac{ds}{b(s)}, \tag{2.10}$$

giving the solution

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

provided the drift has the form

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

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

More generally, if the drift has the form

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

then the SDE can be reduced to the Langevin equation

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

giving the 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 routine **stochastic[reducible]** returns the explicit solution of a reducible SDE 2.8. Its CALLING SEQUENCE is

reducible(a,b);

with PARAMETERS

```
        a - algebraic, given in the variables x and t.
        b - algebraic, given in the variables x and t.
```

representing the drift and diffusion coefficient of the SDE (which actually should not depend explicitly on the t variable here). If the SDE is not of the above reducible form, then a suitable error message is returned.

```
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. 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.

This routine is used by the routine **stochastic[explicit]** and, in general, is not intended to be used on its own. It is part of the stochastic package and is usually loaded via the call **with(stochastic)**, but can also be called via the call **stochastic[reducible]**.

EXAMPLE: Consider the scalar nonlinear SDE

$$dX_t = 1\,dt + 2\sqrt{X_t}\,dW_t.$$

```
>reducible(1,2*sqrt(x));
```

$$X[t] = 2\,X[0]^{1/2}\,W + W^2 + X[0]$$

This SDE is reducible and the required solution is

$$X_t = X_0 + 2\sqrt{X_0}\, W_t + (W_t)^2 .$$

SEE ALSO: **stochastic**, **stochastic[linearsde]**, **stochastic[explicit]**

2.3 Explicit routine

The routine **stochastic[explicit]** applies the routines **stochastic[linearsde]** and **stochastic[reducible]** to a general scalar SDE and returns the explicit solution if the SDE is either linear or reducible as in the preceeding subsections. Its CALLING SEQUENCE is

explicit(a,b);

with PARAMETERS

```
a - algebraic, given in the variables x and t
b - algebraic, given in the variables x and t
```

representing the drift and the diffusion coefficient of the SDE. A suitable error message is returned if the SDE is not linear or reducibale to a linear SDE.

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

The call **explicit(a,b);** returns the explicit solution for a scalar SDE with drift coefficient a and diffusion coefficient b of the appropriate 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, W denotes a standard Wiener process, and t time. The user is returned with a suitable error message if no known explicit solution exists.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked via the call **stochastic[explicit]**.

EXAMPLE: Consider the scalar SDE

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

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

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

                X[t]=exp(aW)X[0]
```

This SDE is linear and thus explicitly solvable with the solution

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

SEE ALSO: **stochastic**, **stochastic[linearsde]**, **stochastic[reducible]**

3 Ito Stochastic Calculus

We now consider a general N-dimensional Ito SDE

$$dX_t = a(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t)\, dW_t^j \tag{3.1}$$

with an M-dimensional Wiener process $W_t = (W_t^1, \cdots, W_t^M)$. The operators L^0 and $L6j$ with respect to this SDE which are defined by

$$L^0 = \frac{\partial}{\partial t} + \sum_{k=1}^{N} a^k \frac{\partial}{\partial x^k} + \frac{1}{2} \sum_{k,l=1}^{N} \sum_{j=1}^{M} b^{k,j} b^{l,j} \frac{\partial^2}{\partial x^k \partial x^l} \tag{3.2}$$

and

$$L^j = \sum_{k=1}^{N} b^{k,j} \frac{\partial}{\partial x^k}, \qquad j = 1, \ldots, M \tag{3.3}$$

play a fundamental role in stochastic calculus through the Ito formula, stochastic Taylor expansions and numerical schemes for the SDE that are based on stochastic Taylor expansions.

3.1 LJ Operator routine

The routine **stochastic[LJ]** applies the partial differential operator L^j defined by 3.3. Its CALLING SEQUENCE

LJ(X,[[b11,..,b1M], ..., **[bN1,..,bNM]],j);**

has PARAMETERS

```
                          X - algebraic, given in the variables x[N] and t,
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in the variables
                              x[N] and t,
                          j - integer,
```

where the bIJ are the components of the diffusion coefficient vectors of the SDE (3.1).

```
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 the function X, where $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$ denotes the components of the M N–dimensional diffusion coefficient vectors, where M is the dimension of the Wiener process and N is the dimension of the SDE (3.1), and $j = 1$, ..., M denotes the "current" component of the Wiener process. The output variables are consistent with the variables used as input.

This routine is used by the routines

stochastic[Euler], **stochastic[Milstein]**, **stochastic[Taylor1.5]**, **stochastic[Taylor2]**, **stochastic[wkeuler]**, **stochastic[wktay2]**, **stochastic[itoformula]**, **stochastic[MLJ]**.
In general, it is not intended for use on its own, but is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked via the call **stochastic[LJ]**

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

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

where r is a constant.

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

                r

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

                0
```

This has evaluated $L^2X(t, x_1, x_2) = r$ and $L^1X(t, x_1, x_2) = 0$

SEE ALSO:
stochastic[L0], **stochastic[MLJ]**, **stochastic[SL0]**, **stochastic[Euler]**, **stochastic[Milstein]**, **stochastic[Taylor1hlf]**, **stochastic[Taylor2]**, **stochastic[wkeuler]**, **stochastic[wktay2]**, **stochastic**.

3.2 L0 Operator routine

The routine **stochastic[L0]** applies the partial differential operator L^0 to the function X. Its CALLING SEQUENCE

L0(X,[a1,..,aN], [[b11,..,b1M],.., [bN1,..,bNM]]);

has PARAMETERS

```
                          X - algebraic, given in the variables x[N] and t,
                    a1,..,aN - algebraics, given in the variables x[N] and t,
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in the variables
                                 x[N] and t,
```

where $a1, \ldots, aN$ denotes the drift coefficients of the N–dimensional SDE (3.1), while $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$ denote the components of the corresponding M N–dimensional diffusion coefficient vectors.

```
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 the function X. It is used by the routines

stochastic[Euler],**stochastic[Milstein]**, **stochastic[Taylor1.5]**, **stochastic[wkeuler]**, **stochastic[wktay2]**,**stochastic[itoformula]**, **stochastic[LFP]**, **stochastic[MLJ]**.

In general, it is not intended for use on its own. The routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)** and can also be invoked via the call **stochastic[L0]** .

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

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} dt + \begin{pmatrix} r \\ 0 \end{pmatrix} dW_t^1 + \begin{pmatrix} 0 \\ r \end{pmatrix} dW_t^2,$$

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

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

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,0],[0,r]]);

                              x[2]
```

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

3.3 LFP Operator: Fokker–Planck Equation

The transition probabilities of the Ito stochastic differential equation (3.1) have densities $p(s,x;t,y)$ which satisfy the *Fokker-Planck equation,*

$$\frac{\partial p}{\partial t} + \sum_{i=1}^{N} \frac{\partial}{\partial y_i}\left\{a^i(t,y)p\right\} - \frac{1}{2}\sum_{i,j=1}^{N}\sum_{k=1}^{M}\frac{\partial^2}{\partial y_i \partial y_j}\left\{b^{i,k}(t,y)b^{j,k}(t,y)p\right\} = 0 \qquad (3.4)$$

(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 $\Re^N$

We define the corresponding differential operator $\mathcal{L}^*$ as , which is in fact the adjoint of the $\mathcal{L}$ or L^0 operator above as

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

In MAPLE it will be called the LFP operator.

The CALLING SEQUENCE

LFP(P,[a1,..,aN], [[b11,..,b1M],.., [bN1,..,bNM]]);

has PARAMETERS

```
                          P - algebraic, given in the variables y[N] and t,
                    a1,..,aN - algebraics, given in the variables y[N] and t,
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in the variables
                                y[N] and t,
```

where $a1,\ldots,aN$ denotes the drift coefficients of the N–dimensional SDE (3.1), while $[b11,\ldots,b1M]$, ..., $[bN1,\ldots,bNM]$ denote the components of the corresponding M N–dimensional diffusion coefficient vectors. The procedure has following code:

```
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:
```

EXAMPLE: Consider the Ito equation

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

where a and b are constant. If we want obtain the equation (3.4) for this Ito-equation we call

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

and obtain

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

If we have a density function

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

and want know, if $p(s,x,t,y)$ satisfy the Fokker–Planck equation, we call:

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

The output of **LFP** is:

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

After the MAPLE command

```
simplify(%);
```

we obtain 0. Hence $p(s,x,t,y)$ satisfies the Fokker–Planck equation.

3.4 Application of the partial differential operators

The routine **stochastic[MLJ]** applies one of the partial differential operators, $L0$ or LJ to the mapping X. Its CALLING SEQUENCE is

MLJ(X,[a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]],j);

with PARAMETERS

```
                          X - algebraic, given in the variables x[N] and t.
                  a1,..,aN - algebraics, given in the variables x[N] and t.
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in
                                variables x[N] and t.
                          j - integer.
```

where $a1, \ldots, aN$ denotes the drift coefficients of the N–dimensional SDE (3.1), while the $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$ denote the corresponding M N–dimensional diffusion coefficient vectors of the SDE (3.1).

```
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:
```

The call **MLJ(X,[a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]],j);** computes the application of either the operator LJ or the operator $L0$ to the function X. Here $a1$, ..., aN are the drift coefficients of the N–dimensional SDE and $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$ are the M N–dimensional diffusion coefficient vectors, where M is the dimension of the Wiener process, while $j = 1,..,M$ refers to the 'current' component of the Wiener process. The output variables are consistent with the variables used as input.

This routine is used by the routine **stochastic[wktay3]** and in general, is not intended for use on its own. It is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**, but can also be invoked directly via the call **stochastic[MLJ]**.

EXAMPLES: Consider the 2–dimensional SDE driven by a 2–dimensional Wiener process given by

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^2 \\ X_t^1 \end{pmatrix} dt + \begin{pmatrix} r \\ 0 \end{pmatrix} dW_t^1 + \begin{pmatrix} 0 \\ r \end{pmatrix} dW_t^2,$$

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

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

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

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

                    r

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

                    x[1]
```

The result is $L^2X(t, x_1, x_2) = r$ and $L^0X(t, x_1, x_2) = x_1$.

SEE ALSO: **stochastic**, **stochastic[L0]**, **stochastic[LJ]**, **stochastic[SL0]**, **stochastic[wktay3].**

3.5 The Ito Formula

For a sufficiently smooth transformation $U : [0, T] \times \Re^N \to \Re$ 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 $Y_t = U(t, X_t)$ satissfies the a vector stochastic differential

$$\begin{aligned} dY_t &= \left(\frac{\partial U}{\partial t} + \sum_{i=1}^{N} a^i \frac{\partial U}{\partial x_i} + \frac{1}{2} \sum_{i,j=1}^{N} \sum_{l=1}^{M} b^{i,l} b^{j,l} \frac{\partial^2 U}{\partial x_i \partial x_j} \right) dt \\ &\quad + \sum_{l=1}^{M} \sum_{i=1}^{N} b^{i,l} \frac{\partial U^p}{\partial x_i}\, dW_t^l \end{aligned} \tag{3.5}$$

where the terms are all evaluated at (t, X_t). This is called the *Ito Formula.*

In operator form it is

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

```
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 we have:

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

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

        ]),
```

that is

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

3.6 Coloured Noise

We convert the N-dimensional Ito SDE (3.1) with a single white noise (i.e. $m = 1$) into its counterpart with coloured noise, that is driven by an Ornstein–Uhlenbeck or exponentially correlated coloured noise process. The resulting coloured noise equation is the $(N+1)$-dimensional Ito SDE with scalar additive noise

$$dX_t = (a(t, X_t) + b(t, X_t)\,Z_t)\;dt \tag{3.6}$$

$$dZ_t = -\gamma Z_t\,dt + \beta\,dW_t \tag{3.7}$$

The routine **stochastic[colour]** converts the SDE 3.1 with scalar white noise into its coloured noise counterpart (3.6)-(3.7). Its CALLING SEQUENCE is

colour([a1,..,aN],[b1,..,bN]);

with PARAMETERS

```
a1,..,aN - algebraics, given in the variables x[N] and t.
b1,..,bN - algebraics, given in the variables x[N] and t.
```

where $a1, \ldots, aN$ denotes the drift coefficients and $b1, \ldots, bN$ the diffusion coefficients of N–dimensional SDE with scalar noise.

```
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 N–dimensional SDE (3.1) with scalar white noise into its coloured noise form (3.6)-(3.7). 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 (3.6)-(3.7) and W a standard Wiener process, while gamma and beta denote parameters, usually provided from experimental data, and t denotes time.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[colour]**.

EXAMPLE: Find the coloured noise counterpart of the 2–dimensional SDE with scalar noise

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

```
>colour([a(x[1],t)],[b(x[1],t)]);

table([
               1 = (dx[1][t] = a(x[1], t) dt + b(x[1], t) z[t] dt)

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

>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

$$\begin{aligned} dX_t^1 &= X_t^2\,dt \\ dX_t^1 &= \left(X_t^1\left(\alpha - \left(X_t^1\right)^2\right) - X_t^2 + \sigma\,Z_t\right)\,dt \\ dZ_t &= -\gamma\,Z_t\,dt + \beta\,dW_t \end{aligned}$$

SEE ALSO: **stochastic**

4 Stratonovich Stochastic Caculus

We consider a the N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t = a(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t)\, dW_t^j \tag{4.1}$$

and its Stratonovich counterpart

$$dX_t = \underline{a}(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t) \circ dW_t^j. \tag{4.2}$$

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

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

for $i = 1, \ldots, N$, while a given $\underline{a}$ is defined componentwise by

$$a^i(t, X) = \underline{a}^i(t, X) + \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{M} b^{j,k}(t, X)\, \frac{\partial b^{i,k}}{\partial x_j}(t, X)$$

for $i = 1, \ldots, N$.

4.1 Ito–Stratonovich correction

Here we introduce a procedure which applies the drift-correction formula to convert the Ito drift coefficient a into the corresponding Stratonovich drift coefficient $\underline{a}$. This is the routine **stochastic[correct]** which has CALLING SEQUENCE:

correct([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]],i);

with PARAMETERS

```
                  a1,..,aN - algebraics, given in the variables x[N] and t.
b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in the  variables
                              x[N] and t.
                        i - integer.
```

where $a1, \ldots, aN$ denote the drift coefficients of the N–dimensional SDE (4.2) and $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$ denote the components of the corresponding M N–dimensional diffusion coefficient vectors, M being the dimension of the Wiener process W. The index $i = 1, \ldots, N$ denotes the 'current' component of the SDE. The output variables are consistent with the variables used as input.

```
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:
```

The call **correct([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]],i);** converts the drift coefficient of the Ito SDE (4.1) into that of its Stratonovich form (4.2).

This routine is used by the routine **stochastic[Taylor2]**. It is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**, but can also be invoked via the call **stochastic[correct]**.

EXAMPLES: Both examples have $N = M = 2$ with 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

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

where r is a constant, in the first case and the variable diffusion coefficient vectors

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

in the second case.

```
>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 application 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.

SEE ALSO: **stochastic**, **stochastic[Taylor2]**

4.2 Ito–Stratonovich correction: both directions

The next procedure combines the Ito to Stratonovich conversion with the Stratonovich to Ito conversion procedure of the last subsection.

```
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 obtain the Stratonovich SDE use

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

and obtain

$$\begin{array}{l}\text{table}([\\ \quad 1 = 0\\ \quad]),\end{array}$$

which means that the desired Stratonovich SDE is

$$\begin{aligned} dX_t &= 0\,dt - a(1 - X_t^2) \circ dW_t \\ &= -a(1 - X_t^2) \circ dW_t. \end{aligned}$$

The other direction gives the original Ito SDE back.

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

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

4.3 Stratonovich L0 operator

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

$$\underline{L}^0 = \frac{\partial}{\partial t} + \sum_{k=1}^{N} \underline{a}^k \frac{\partial}{\partial x^k}, \tag{4.3}$$

while the L^j operator of of Ito calculus remains unchanged in Stratonovich calculus. The Stratonovich operator $\underline{L}^0$ applied to a function X is produced by the routine **stochastic[SL0]** with CALLING SEQUENCE:

SL0(X,[a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

which has PARAMETERS:

```
                          X - algebraic, given in the variables x[N] and t.
                  a1,..,aN - algebraics, given in the variables x[N] and t.
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in the variables
                               x[N] and t.
```

where $a1, \ldots, aN$ denote the drift coefficients of the N–dimensional SDE (4.2) and $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$ denote the components of the corresponding M N–dimensional diffusion coefficient vectors, M being the dimension of the Wiener process W. The output variables are consistent with the variables used as input.

```
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 X.

This routine is used by the routine **stochastic[Taylor2]**. In general, it is not intended for use on its own. It is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**, but can also be invoked directly via the call **stochastic[SL0]**.

EXAMPLE: $\underline{L}^0 X$ is computed 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

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

where r is a constant.

```
>SL0(x[2],[x[1],x[2]],[[r,0],[0,r]]);

                              x[2]
```

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

SEE ALSO: **stochastic**, **stochastic[L0]**, **stochastic[LJ]**, **stochastic[MLJ]**, **stochastic[Taylor2]**

4.4 Stratonovich transformation formula

For a sufficiently smooth transformation $U : [0,T]\times\Re^N \to \Re$ 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 a vector Stratonovich stochastic differential

$$dY_t = \left(\frac{\partial U}{\partial t} + \sum_{i=1}^{N} \underline{a}^i \frac{\partial U}{\partial x_i}\right) dt + \sum_{l=1}^{M}\sum_{i=1}^{N} b^{i,l}\frac{\partial U}{\partial x_i} \circ dW_t^l \tag{4.4}$$

where the terms are all evaluated at (t,X_t). In operator form this is

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

```
chainrule:=proc(U::list(algebraic),a::list(algebraic),
                b::list(list(alge braic)))
  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 we have:

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

  table([

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

        ]).
```

That means:

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

5 Linear Vector SDEs

5.1 Linearization

We consider the linearization of an N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t = a(t, X_t)\,dt + \sum_{j=1}^{M} b^j(t, X_t)\,dW_t^j \tag{5.1}$$

about fixed solution $\bar{X}_t$, resulting in the linear vector SDE

$$dZ_t = A(t)Z_t\,dt + \sum_{j=1}^{M} B^j(t)Z_t\,dW_t^j \tag{5.2}$$

where

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

for i, $j = 1, \ldots, N$ and $k = 1, \ldots, M$.

The routine **stochastic**[**linearize**] has the CALLING SEQUENCE:

linearize(A,B,C);

with PARAMETERS

```
A - the list of $[a[1],a[2],...,a[N]]$,
B - listlist of $[[b[11],...,b[1M]],...,[b[N1],...,b[NM] ]],
C - list of [Xbar[1],...,Xbar[N]], where Xbar is the stationary solution
    of the SDE.
```

The procedure **linearize** has the following code:

```
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:
```

EXAMPLE: Consider 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.

$$\begin{aligned} \underline{a}^2 &= X^2(t) \\ \underline{a}^2 &= -bX^2 - \sin(X^1) - c\sin(2X^1) - a^2(X^2)^3 + aX^2\sin(X^1). \end{aligned}$$

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

Then

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

so the linearized SDE (5.2) is

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

5.2 Second moment equation

We consider the N–dimensional linear Ito SDE

$$dZ_t = A(t)Z_t\, dt + \sum_{k=1}^{M} B^k(t)Z_t\, dW_t^k, \tag{5.3}$$

where A, B^1, B^2, ..., B^M are $N \times N$ matrices.

The $N\times N$ matrix valued second moment $P(t) = E(Z_t Z_t^\top)$ 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. On account of the symmetry of the matrix P, we can write this equation as a linear system of the from

$$\frac{d\tilde{p}}{dt} = \mathcal{A}(t)\tilde{p} \tag{5.4}$$

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 square matrix.

The routine **stochastic[momenteqn]** has the CALLING SEQUENCE:

momenteqn(A,B);

with PARAMETERS

```
 A - the matrix A(t),
 B - lists of matrixes B1(t),...,BM(t).
```

The procedure **momenteqn** calculates the new matrix $\mathcal{A}(t)$ and has the following code:

```
momenteqn:=proc(A,B)
  local i,j,k,N,Btmp,Ctmp;
  global Neues_A;
  if type(A,array) then Btmp:=convert(A,listlist);else Btmp:=A; fi;
  N:=nops(Btmp);

Neues_A:=array(1..N*(N+1)/2,1..N*(N+1)/2);Ctmp:=array(1..N*(N+1)/2,1..N*(N+1)/2)
;
  ap(A);
  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
    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
      Neues_A[i,j]:=B1[i,j]+B2[i,j]+Ctmp[i,j];
    od;
  od;
  RETURN(evalm(Neues_A));
end:
```

This procedure requires other procedures. It is not necesary to call these procedures, but they are must be declared (in the package) before calling the procedure **momenteqn** (see the APPENDIX).

EXAMPLE: Let

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}.$$

We set $E = [B]$ (list of arrays) and call **momenteqn(A,E)**, obtaining

$$\begin{bmatrix} 2\,A_{1,1} + {B_{1,1}}^2 & 2\,A_{1,2} + 2\,B_{1,2}\,B_{1,1} & {B_{1,2}}^2 \\ A_{2,1} + B_{1,1}\,B_{2,1} & A_{1,1} + A_{2,2} + B_{1,2}\,B_{2,1} + B_{1,1}\,B_{2,2} & A_{1,2} + B_{1,2}\,B_{2,2} \\ {B_{2,1}}^2 & 2\,A_{2,1} + 2\,B_{2,2}\,B_{2,1} & 2\,A_{2,2} + {B_{2,2}}^2 \end{bmatrix}.$$

The equation (5.4) is thus

$$\begin{aligned} d\tilde{p}_1 &= \left(2\,A_{1,1} + {B_{1,1}}^2\right)\tilde{p}_1 + (2\,A_{1,2} + 2\,B_{1,2}\,B_{1,1})\,\tilde{p}_2 + \left({B_{1,2}}^2\right)\tilde{p}_3 \\ d\tilde{p}_2 &= (A_{2,1} + B_{1,1}\,B_{2,1})\,\tilde{p}_1 + (A_{1,1} + A_{2,2} + B_{1,2}\,B_{2,1} + B_{1,1}\,B_{2,2})\,\tilde{p}_2 \\ &\quad + (A_{1,2} + B_{1,2}\,B_{2,2})\,\tilde{p}_3 \\ d\tilde{p}_3 &= \left({B_{2,1}}^2\right)\tilde{p}_1 + (2\,A_{2,1} + 2\,B_{2,2}\,B_{2,1})\,\tilde{p}_2 + \left(2\,A_{2,2} + {B_{2,2}}^2\right)\tilde{p}_3 \end{aligned}$$

The procedures **matrix2pvector** and **pvector2pmatrix** transform a symmetry matrix to a vector and a vector to an symmetry matrix, respectively. Here they are used to change the matrix P to the vector $\tilde{p}$ and the vector $\tilde{p}$ to the matrix P, respectively. The procedure **matrix2pvector**has the following code (see the APPENDIX for the procedure **pvector2pmatrix**):

```
matrix2pvector:=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:
```

EXAMPLE: Let

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

After calling **pmatrix2pvector(P);** we obtain

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

EXAMPLE: Consider the 2×2 symmetric matrix written in the listlist form

$$P := [[1,\ 2],\ [2,\ 4]].$$

For $\tilde{p}$ the procedure **pmatrix2pvector(P);** yields

[1, 2, 4].

These examples for the first procedure show that it is possible to have a matrix or a listlist as an input parameter.

5.3 Spherical coordinates

We consider the N–dimensional linear Stratonovich SDE

$$dZ_t = A(t)Z_t\,dt + \sum_{k=1}^{M} B^k(t)Z_t \circ dW_t^k, \tag{5.5}$$

where A, B^1, B^2, ..., B^M are $N\times N$ matrices, and convert it to spherical coordinates $r = |z|$ and $s = z/|z| \in \mathcal{S}^{N-1}$ (assuming $z \neq 0$). The resulting system of equations is

$$dR_t = R_t\,q^0(S_t)\,dt + \sum_{k=1}^{M} R_t\,q^k(S_t) \circ dW_t^k \tag{5.6}$$

$$dS_t = h(S_t, A)\,dt + \sum_{k=1}^{M} h(S_t, B^k) \circ dW_t^k \tag{5.7}$$

where

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

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

The routine **stochastic[sphere]** has the CALLING SEQUENCE:

sphere(A,B);

with PARAMETERS

```
A - the matrix A(t),
B - lists of matrixes $B1(t),...,BM(t).
```

It calculates the values for

$$q(s), \quad q^0(s), \quad q^k(s), \quad h(s,A) \text{ and } \quad h(s,B^k) \quad \text{ for } k = 1,\ldots,M$$

and has the following code:

```
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, we call **sphere(A,B): q;**.

EXAMPLE: Consider the 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

```
> 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([
$1=4-(3\,s_1+4\,s_2)\,s_1-(s_1+2\,s_2)\,s_2$
$2=6-(3\,s_1+4\,s_2)\,s_1-(s_1+2\,s_2)\,s_2$
])
])

6 Strong Numerical Schemes

We consider a the N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t = a(t,X_t)\,dt+\sum_{j=1}^{M} b^j(t,X_t)\,dW_t^j \tag{6.1}$$

and some strong stochastic Taylor schemes for this equation. The final order 2.0 scheme will be the corresponding Stratonovich SDE.

In all of the schemes that follow, the coefficients are all evaluated at the point (t_n,Y_n).

6.1 Euler scheme

The *Euler scheme* has the componentwise form

$$Y_{n+1}^k = Y_n^k + a^k\,\Delta_n + \sum_{j=1}^{M} b^{k,j}\,\Delta W_n^j \tag{6.2}$$

for $k = 1, \ldots, N$, where

$$\Delta_n = t_{n+1} - t_n$$

is the length of the nth time step and

$$\Delta W_n^j = W_{\tau_{n+1}}^j - W_{\tau_n}^j$$

is the $N(0;\Delta_n)$ distributed increment of the jth component of the M-dimensional standard Wiener process W on the discretization subinterval $[\tau_n, \tau_{n+1}]$. 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 stochastic Taylor scheme of strong order 0.5 known as the *Euler scheme* for an Ito SDE 6.1. Its CALLING SEQUENCE is

Euler([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

with PARAMETERS

```
                    a1,..,aN - algebraics, given in the variables x[N] and t.
 [b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given  in variables x[N]
                                and t.
```

representing the drift and the diffusion coefficient vectors of the SDE.

```
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 coefficients $a1,\ldots, aN$ and diffusion coefficients $[b11,\ldots,b1M],\ldots, [bN1,\ldots,bNM]$.

The output consists of the variables $YN[n]$, Delta$WM[n]$, and Delta$[n]$. $YN[n]$ denotes the strong order 0.5 stochastic Taylor approximation to $x[N]$ at the n-th step. Delta$WM[n]$ denotes the change in the M–dimensional Wiener process at the n-th step. Delta$[n]$ denotes the step size at the n-th step.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked driectly via the call **stochastic[Euler]**.

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

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

that is 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

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

where r and s are constants.

```
>Euler([x[2],x[1]],[[r,s],[s,r]]);

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

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

The resulting Euler scheme is

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

SEE ALSO: **stochastic**, **stochastic[L0]**, **stochastic[LJ]**, **stochastic[Milstein]**, **stochastic[Taylor1hlf]**, **stochastic[Taylor2]** .

6.2 Milstein scheme

The *Milstein scheme* has the componentwise form

$$Y_{n+1}^k = Y_n^k + a^k \Delta_n + \sum_{j=1}^{M} b^{k,j} \Delta W_n^j + \sum_{j_1,j_2=1}^{M} L^{j_1} b^{k,j_2} I_{(j_1,j_2);n}, \tag{6.3}$$

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

$$I_{(j_1,j_2);n} = \int_{\tau_n}^{\tau_{n+1}} \int_{\tau_n}^{s_1} dW_{s_2}^{j_1} \, dW_{s_1}^{j_2} \tag{6.4}$$

in general, with

$$I_{(j_1,j_1);n} = \frac{1}{2} \left\{ \left(\Delta W_n^{j_1} \right)^2 - \Delta_n \right\}$$

and similarly for $I_{(j_2,j_2);n}$.

The routine **stochastic[Milstein]** constructs stochastic Taylor scheme of strong order 1.0 known as the *Milstein scheme* for an Ito SDE (6.1). Its CALLING SEQUENCE is

Milstein([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

with PARAMETERS

```
                    a1,..,aN - algebraics, given in the variables x[N] and t.
 [b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given  in variables x[N]
                                and t.
```

representing the driftand the diffusion coefficient vectors of the SDE.

```
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 (6.1) with M–dimensional noise which has drift coefficients $a1$, ..., aN and diffusion coefficient vectors $[b11,\ldots,b1M]$, ..., $[bN1,\ldots,bNM]$. The output consists of the variables $YN[n]$, Delta$WM[n]$, Delta$[n]$ and $I[(j1,j2)]$. Here $YN[n]$ denotes the strong order 1.0 stochastic Taylor approximation to $x[N]$ at the n-th step, Delta$WM[n]$ denotes the increment in the M–dimensional Wiener process at the n-th step, Delta$[n]$ denotes the step size at the n-th step, and $I[(j1,j2)]$ denotes the double Ito integral (6.4)

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[Milstein]**.

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

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

that is 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

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

where r and s are constants.

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

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

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

The resulting Milstein scheme is

$$\begin{pmatrix} Y^1_{n+1} \\ Y^2_{n+1} \end{pmatrix} = \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} + \begin{pmatrix} Y^2_n \\ Y^1_n \end{pmatrix} \Delta_n + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W^1_n + \begin{pmatrix} s \\ r \end{pmatrix} \Delta W^2_n,$$

which is actually the same as the Euler scheme in this case because the SDE here has additive noise.

SEE ALSO: **stochastic**, **stochastic[L0]**, **stochastic[LJ]**, **stochastic[Euler]**, **stochastic[Taylor1hlf]**, **stochastic[Taylor2]** .

6.3 Order 1.5 strong stochastic Taylor scheme

The kth component of the *order* 1.5 *strong Taylor scheme* is given by

$$\begin{aligned} Y^k_{n+1} &= Y^k_n + a^k\,\Delta_n + \frac{1}{2}\,L^0 a^k\,\Delta_n^2 \\ &\quad + \sum_{j=1}^{M}\left(b^{k,j}\,\Delta W^j_n + L^0 b^{k,j} I_{(0,j);n} + L^j a^k\, I_{(j,0);n}\right) \\ &\quad + \sum_{j_1,j_2=1}^{M} L^{j_1} b^{k,j_2} I_{(j_1,j_2);n} + \sum_{j_1,j_2,j_3=1}^{M} L^{j_1} L^{j_2} b^{k,j_3} I_{(j_1,j_2,j_3);n} \end{aligned} \tag{6.5}$$

where $I_{(j_1,j_2,j_3);n}$ is the multiple Ito integral

$$I_{(j_1,j_2,j_3);n} = \int_{t_n}^{t_{n+1}} \int_{\tau_n}^{s_1} \int_{\tau_n}^{s_2} dW^{j_1}_{s_3}\, dW^{j_1}_{s_2}\, dW^{j_2}_{s_1} \tag{6.6}$$

in general, with

$$I_{(j_1,j_1,j_1);n} = \frac{1}{2}\left\{\frac{1}{3}\left(\Delta W^{j_1}_n\right)^2 - \Delta_n\right\}\Delta W^{j_1}_n$$

and

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

in specific cases. Here the random variable $\Delta Z^j_n := I_{(j,0);n}$ is $N(0;\frac{1}{3}\Delta_n)$ normally distributed and has covariance $E(\Delta Z^j_n \Delta W^j_n = \frac{1}{2}\Delta^2_n$.

The routine **stochastic[Taylor1hlf]** constructs stochastic Taylor scheme of strong order 1.0 Taylor scheme for an Ito SDE (6.1). Its CALLING SEQUENCE is

Taylor1hlf([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

with PARAMETERS

```
                    a1,..,aN - algebraics, given in the variables x[N] and t.
 [b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given  in variables x[N]
                                and t.
```

representing the drift and the diffusion coefficient vectors of the SDE.

```
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 (6.1) with M–dimensional noise which has drift coefficients $a1$, ..., aN and diffusion coefficient vectors $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$.

The output consists of the variables $YN[n]$, Delta$WM[n]$, Delta$[n]$, $I[(j1, j2)]$, and $I[(j1, j2, j3)]$. Here $YN[n]$ denotes the strong order 1.5 stochastic Taylor approximation to $x[N]$ at the n-th step, Delta$WM[n]$ denotes the change in the M–dimensional Wiener process at the n-th step, Delta$[n]$ denotes thestep size at the n-th step. while$I[(j1, j2)]$ and $I[(j1, j2, j3)]$ denote the multiple Ito integrals (6.4) and (6.6).

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[Taylor1hlf]**.

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

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

that is 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

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

where r and s are constants.

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

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] + s Delta W2[n] + r 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] + r Delta W2[n] + s I[2, 0])

     ])
```

The resulting order 1.5 strong Taylor scheme scheme is

$$\begin{pmatrix} Y^1_{n+1} \\ Y^2_{n+1} \end{pmatrix} = \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} + \begin{pmatrix} Y^2_n \\ Y^1_n \end{pmatrix} \Delta_n + \frac{1}{2} \begin{pmatrix} Y^2_n \\ Y^1_n \end{pmatrix} (\Delta_n)^2 + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W^1_n$$
$$+ \begin{pmatrix} s \\ r \end{pmatrix} \Delta W^2_n + \begin{pmatrix} s \\ r \end{pmatrix} I_{(1,0);n} + \begin{pmatrix} r \\ s \end{pmatrix} I_{(2,0);n},$$

SEE ALSO: **stochastic**, **stochastic[L0]**, **stochastic[LJ]**, **stochastic[Euler]**, **stochastic[Milstein]**, **stochastic[Taylor2]** .

6.4 2nd order stochastic Taylor scheme

We now consider the N–dimensional Stratonovich SDE with an M–dimensional Wiener process

$$dX_t = \underline{a}(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t) \circ dW^j_t \tag{6.7}$$

for which the kth component of the *order* 2.0 *strong Taylor scheme* is given by

$$\begin{aligned} Y^k_{n+1} &= Y^k_n + \underline{a}^k \Delta + \frac{1}{2} \underline{L}^0 \underline{a}^k \Delta^2 \\ &+ \sum_{j=1}^{m} \left(b^{k,j} \Delta W^j + \underline{L}^0 b^{k,j} J_{(0,j)} + \underline{L}^j \underline{a}^k J_{(j,0)} \right) \\ &+ \sum_{j_1,j_2=1}^{m} \Big(\underline{L}^{j_1} b^{k,j_2} J_{(j_1,j_2)} + \underline{L}^0 \underline{L}^{j_1} b^{k,j_2} J_{(0,j_1,j_2)} \\ &\qquad + \underline{L}^{j_1} \underline{L}^0 b^{k,j_2} J_{(j_1,0,j_2)} + \underline{L}^{j_1} \underline{L}^{j_2} \underline{a}^k J_{(j_1,j_2,0)} \Big) \\ &+ \sum_{j_1,j_2,j_3=1}^{m} \underline{L}^{j_1} \underline{L}^{j_2} b^{k,j_3} J_{(j_1,j_2,j_3)} \\ &+ \sum_{j_1,j_2,j_3,j_4=1}^{m} \underline{L}^{j_1} \underline{L}^{j_2} \underline{L}^{j_3} b^{k,j_4} J_{(j_1,j_2,j_3,j_4)}. \end{aligned} \tag{6.8}$$

The $J_{(j_1,j_2)}$ and $J_{(j_1,j_2,j_3)}$ expressions here denote the corresponding double and triple Stratonovich integrals with respect to the given Wiener process.

The routine **stochastic[Taylor2]** constructs the stochastic Taylor scheme of strong order 2.0 Taylor scheme for the Stratonovich SDE (6.7). Its CALLING SEQUENCE is

Taylor2([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

with PARAMETERS

```
                    a1,..,aN - algebraics, given in the variables x[N] and t.
 [b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given  in variables x[N]
                                and t.
```

represent the drift and the diffusion coefficient vectors of the Stratonovich SDE (6.7).

```
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]]);** returns the strong order 2.0 stochastic Taylor approximation for the N–dimensional Stratonovich SDE (6.7) with M–dimensional noise which has drift coefficients $a1,\ldots,aN$ and diffusion coefficient vectors $[b11,\ldots,b1M]$, ..., $[bN1,\ldots,bNM]$.

The output consists of 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 n-th step, Delta$WM[n]$ denotes the increment in the M–dimensional Wiener process at the n-th step, Delta$[n]$ denotes the step size

at the n-th step, while $J[(j1,j2)]$, $J[(j1,j2,j3)]$, and $J[(j1,j2,j3,j4)]$ denote multiple Stratonovich integrals.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[Taylor2]**.

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

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

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

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

where r and s are constants.

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

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] + s Delta W2[n] + r 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] + r Delta W2[n] + s J[2, 0])

    ])
```

The resulting order 2.0 strong Stratonovich Taylor scheme scheme is

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

SEE ALSO: **stochastic**, **stochastic[LJ]**, **stochastic[SL0]**, **stochastic[Euler]**, **stochastic[Milstein]**, **stochastic[Taylor1hlf]** , **stochastic[correct]** .

7 Commutative Noise

We consider a the N–dimensional Ito SDE with an M–dimensional Wiener process

$$dX_t = a(t,X_t)\,dt + \sum_{j=1}^{M} b^j(t,X_t)\,dW_t^j. \tag{7.1}$$

Strongly convergent numerical schemes can be simplified to avoid the need to simulate multiple stochastic integrals when the noise coefficients b^j, $j = 1, \ldots, M$ satisfy certain relationships known as commutative noise.

7.1 Commutative noise of 1st kind

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

$$L^{j_1} b^{k,j_2} = L^{j_2} b^{k,j_1} \tag{7.2}$$

for all j_1, $j_2 = 1, \ldots, M$, $k = 1, \ldots, N$ and $(t, x) \in \Re^+ \times \Re^N$.

For instance, additive noise, diagonal noise and linear noise all satisfy this commutativity condition, where 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

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

for all $j = 1, \ldots, M$, $k = 1, \ldots, N$ and $(t, x) \in \Re^+ \times \Re^N$.

The routine **stochastic[comm1]** informs the user if the diffusion matrix of an Ito SDE has commutative noise of the first kind. Its CALLING SEQUENCE

comm1([b11,..,b1M],..,[bN1,..,bNM]);

has PARAMETERS

```
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics,  given in the variables
                               x[N] and t.
```

which are the diffusion coefficient vectors of the SDE.

```
 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 (7.2). Here $[b11, \ldots, b1M]$,..., $[bN1, \ldots, bNM]$ denotes the M N–dimensional diffusion coefficient vectors, where N is the dimension of the SDE and M is the dimension of the Wiener process.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[comm1]**

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

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

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

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

SEE ALSO: **stochastic**, **stochastic[comm2]**

7.2 Commutative noise of 2nd kind

Commutative noise of the second kind arises when the noise coefficients satisfy the condition

$$L^{j_1} J^{j_2} b^{k,j_3} = L^{j_2} L^{j_1} b^{k,j_3} \tag{7.3}$$

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

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

comm2([b11,..,b1M],..,[bN1,..,bNM]);

has PARAMETERS

```
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in  the variables
                               x[N] and t
```

which are the difffusion coefficient vectors of the SDE.

```
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 (7.3). Here $[b11,\ldots,b1M]$,..., $[bN1,\ldots,bNM]$ denote the M N–dimensional diffusion coefficient vectors, where N is the dimension of the SDE and M is the dimension of the Wiener process.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[comm2]**.

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

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

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

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

SEE ALSO: **stochastic**, **stochastic[comm1]**

8 Weak Numerical Schemes

We consider a the N-dimensional Ito SDE with an M-dimensional Wiener process

$$dX_t = a(t, X_t)\, dt + \sum_{j=1}^{M} b^j(t, X_t)\, dW_t^j. \tag{8.1}$$

8.1 Weak Euler scheme

The general multi-dimensional case d, $m = 1, 2, \ldots$ the kth component of the *Euler scheme* has the form

$$Y_{n+1}^k = Y_n^k + a^k\, \Delta + \sum_{j=1}^{m} b^{k,j}\, \Delta W^j, \tag{8.2}$$

with initial value $Y_0 = X_0$, where

$$\Delta = \tau_{n+1} - \tau_n \quad \text{and} \quad \Delta W^j = W_{\tau_{n+1}}^j - W_{\tau_n}^j.$$

For weak convergence only the measure induced by the Ito process X needs to be approximated, so the Gaussian increments ΔW^j in (1.1) can be replaced by simpler random variables $\Delta \hat{W}^j$ with similar moment properties giving a simpler scheme by choosing more easily generated noise increments. This leads to the *simplified weak Euler scheme*

$$Y_{n+1}^k = Y_n^k + a^k\, \Delta + \sum_{j=1}^{M} b^{k,j}\, \Delta \hat{W}^j, \tag{8.3}$$

where the $\Delta \hat{W}^j$ for $j = 1, 2, \ldots, m$ must be independent $\mathcal{A}_{\tau_{n+1}}$-measurable random variables with moments satisfying the conditions

$$\left|E\left(\Delta \hat{W}^j\right)\right| + \left|E\left(\left(\Delta \hat{W}^j\right)^3\right)\right| + \left|E\left(\left(\Delta \hat{W}^j\right)^2\right) - \Delta\right| \le K\, \Delta^2$$

for some constant K; see also (5.12.7). A very simple example of such a $\Delta \hat{W}^j$ in (1.2) is a two-point distributed random variable with

$$P\left(\Delta \hat{W}^j = \pm\sqrt{\Delta}\right) = \frac{1}{2}.$$

The routine **stochastic[wkeuler]** constructs the stochastic Taylor scheme of weak order 1.0, known as weak Euler schemes. Its CALLING SEQUENCE is

wkeuler([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

with PARAMETERS:

```
                    a1,..,aN - algebraics, given in the variables x[N] and t.
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in  variables
                                 x[N] and t.
```

representing the drift and diffusion coefficient vectors of the SDE.

```
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]]);** returns the simplified weak Euler scheme for an N–dimensional SDE with M–dimensional noise which has drift coefficients $a1$, ..., aN and diffusion coefficient vecotrs $[b11, \ldots, b1M]$, ..., $[bN1, \ldots, bNM]$. The output consists of the variables $YN[n]$, Delta$WsM[n]$ and Delta$[n]$. Here $YN[n]$ denotes the 1st order simplified weak approximation to $x[N]$ at the n-th step, Delta$WsM[n]$ denotes the increment in the M–dimensional noise process at the n-th 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 n-th step.

This routine is part of the stochastic package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[wkeuler]**.

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

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} dt + \begin{pmatrix} r \\ 0 \end{pmatrix} dW_t^1 + \begin{pmatrix} 0 \\ r \end{pmatrix} dW_t^2,$$

that is with drift components $a^1(t, x_1, x_2) = x1$, $a^2(t, x_1, x_2) = x_2$ and the constant diffusion coefficient vectors

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

where r is a constant.

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

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

            2 = (Y2[n + 1] = Y2[n] + Y2[n] Delta[n] + r Delta Ws2[n])
     ])
```

The resulting simplified weak Euler scheme is

$$\begin{pmatrix} Y_{n+1}^1 \\ Y_{n+1}^2 \end{pmatrix} = \begin{pmatrix} Y_n^1 \\ Y_n^2 \end{pmatrix} + \begin{pmatrix} Y_n^2 \\ Y_n^1 \end{pmatrix} \Delta_n + \begin{pmatrix} r \\ 0 \end{pmatrix} \Delta\hat{W}_n^1 + \begin{pmatrix} 0 \\ r \end{pmatrix} \Delta\hat{W}_n^2.$$

SEE ALSO: **stochastic**, **stochastic[LO]**, **stochastic[LJ]**,**stochastic[wktay2]** .

8.2 Second order weak Taylor scheme

The kth component of the *order* 2.0 *weak Taylor scheme* takes the form

$$\begin{aligned} Y_{n+1}^k &= Y_n^k + a^k\,\Delta + \frac{1}{2}\,L^0 a^k\,\Delta^2 \\ &\quad + \sum_{j=1}^{m}\left\{b^{k,j}\Delta W^j + L^0 b^{k,j}\,I_{(0,j)} + L^j a^k\,I_{(j,0)}\right\} \\ &\quad + \sum_{j_1,j_2=1}^{m} L^{j_1} b^{k,j_2}\,I_{(j_1,j_2)}. \end{aligned} \tag{8.4}$$

Here multiple Ito integrals involving different components of the Wiener process are used. These are generally not easy to generate, so the above scheme is more of theoretical interest than of practical use. However, for weak convergence we can substitute simpler random variables for the multiple Ito integrals to obtain the following *simplified order* 2.0 *weak Taylor scheme* with kth component

$$\begin{aligned} Y_{n+1}^k &= Y_n^k + a^k\,\Delta + \frac{1}{2}\,L^0 a^k\,\Delta^2 \\ &\quad + \sum_{j=1}^{m}\left\{b^{k,j} + \frac{1}{2}\,\Delta\,\left(L^0 b^{k,j} + L^j a^k\right)\right\}\,\Delta\hat{W}^j \\ &\quad + \frac{1}{2}\sum_{j_1,j_2=1}^{m} L^{j_1} b^{k,j_2}\left(\Delta\hat{W}^{j_1}\Delta\hat{W}^{j_2} + V_{j_1,j_2}\right). \end{aligned} \tag{8.5}$$

Here the $\Delta\hat{W}^j$ for $j = 1, 2, \ldots, m$ are independent random variables satisfying the moment conditions

$$\begin{aligned} &\left|E\left(\Delta\hat{W}\right)\right| + \left|E\left(\left(\Delta\hat{W}\right)^3\right)\right| + \left|E\left(\left(\Delta\hat{W}\right)^5\right)\right| \\ &+ \left|E\left(\left(\Delta\hat{W}\right)^2\right) - \Delta\right| + \left|E\left(\left(\Delta\hat{W}\right)^4\right) - 3\Delta^2\right| \le K\,\Delta^3 \end{aligned} \tag{8.6}$$

for some constant K and the V_{j_1,j_2} are independent two-point distributed random variables with

$$P\left(V_{j_1,j_2} = \pm\Delta\right) = \frac{1}{2} \tag{8.7}$$

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

$$V_{j_1,j_1} = -\Delta \tag{8.8}$$

and

$$V_{j_1,j_2} = -V_{j_2,j_1} \tag{8.9}$$

for $j_2 = j_1 + 1, \ldots, m$ and $j_1 = 1, \ldots, m$. Here we can use a three-point distributed random variable $\Delta\hat{W}$ with

$$P\left(\Delta\hat{W} = \pm\sqrt{3\Delta}\right) = \frac{1}{6}, \qquad P\left(\Delta\hat{W} = 0\right) = \frac{2}{3}. \tag{8.10}$$

The routine **stochastic[wktay2]** constructs simplified stochastic Taylor schemes of weak order 2.0. Its CALLING SEQUENCE is

wktay2([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

with PARAMETERS:

```
                    a1,..,aN - algebraics, given in the variables x[N] and t.
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in  variables
                                 x[N] and t.
```

representing the drift and diffusion coefficient vectors of the SDE.

```
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 weak order 2.0 stochastic Taylor scheme for an N–dimensional SDE with M–dimensional noise which has drift coefficients $a1$, ...,aN and diffusion coefficient vectors $[b11,\ldots,b1M]$,..., $[bN1,\ldots,bNM]$.

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 n-th step, Delta$WsM[n]$ denotes the change in the M–dimensional noise process at the n-th step (note here that $WsM[n]$ does not denote standard Wiener processes, but instead are independent 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 n-th step.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked via the call **stochastic[wktay2]**.

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

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} dt + \begin{pmatrix} r \\ 0 \end{pmatrix} dW_t^1 + \begin{pmatrix} 0 \\ r \end{pmatrix} dW_t^2,$$

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

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

where r is a constant.

```
>wktay2([x[1],x[2]],[[r,0],[0,r]]);

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])

                                                                    2
        2 = (Y2[n + 1] = Y2[n] + Y2[n] Delta[n] + 1/2 Y2[n] Delta[n]
                               + (r + 1/2 Delta[n] r) Delta Ws2[n])
    ])
```

The resulting weak order 2.0 Taylor scheme is

$$\begin{pmatrix} Y^1_{n+1} \\ Y^2_{n+1} \end{pmatrix} = \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} + \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} \Delta_n + \frac{1}{2} \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} (\Delta_n)^2 + \begin{pmatrix} r + \frac{1}{2} r \Delta_n \\ 0 \end{pmatrix} \Delta \hat{W}^1_n + \begin{pmatrix} 0 \\ r + \frac{1}{2} r \Delta_n \end{pmatrix} \Delta \hat{W}^2_n.$$

SEE ALSO: **stochastic**, **stochastic[L0]**, **stochastic[LJ]**, **stochastic[wkeuler]** .

8.3 Order 3 weak Taylor scheme

The kth component of the *order* 3.0 *weak Taylor scheme* takes the form

$$\begin{aligned} Y^k_{n+1} &= Y^k_n + a^k \Delta + \sum_{j=1}^{m} b^{k,j} \Delta W^j + \sum_{j=0}^{m} L^j a^k I_{(j,0)} \\ &\quad + \sum_{j_1=0}^{m} \sum_{j_2=1}^{m} L^{j_1} b^{k,j_2} I_{(j_1,j_2)} + \sum_{j_1,j_2=0}^{m} L^{j_1} L^{j_2} a^k I_{(j_1,j_2,0)} \\ &\quad + \sum_{j_1,j_2=0}^{m} \sum_{j_3=1}^{m} L^{j_1} L^{j_2} b^{k,j_3} I_{(j_1,j_2,j_3)}. \end{aligned} \tag{8.11}$$

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 [7]

The routine **stochastic[wktay3]** constructs stochastic Taylor schemes of weak order 3.0. Its CALLING SEQUENCE is

wktay3([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);

with PARAMETERS:

```
                    a1,..,aN - algebraics, given in the variables x[N] and t.
[b11,..,b1M],..,[bN1,..,bNM] - lists of algebraics, given in  variables
                               x[N] and t.
```

representing the drift and diffusion coefficient vectors of the SDE.

```
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:
```

SYNOPSIS: The call **wktay3([a1,..,aN],[[b11,..,b1M],..,[bN1,..,bNM]]);** returns the weak order 3.0 stochastic Taylor scheme for an N–dimensional SDE with M–dimensional noise which has drift coefficients $a1$, ..., aN and diffusion coefficient vectors $[b11,\ldots,b1M]$, ..., $[bN1,\ldots,bNM]$. 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 3rd order weak approximation to $x[N]$ at the n-th step, Delta$WM[n]$ denotes the increment in the M–dimensional Wiener process at the n-th step, $I[(j1,j2)]$ and $I[(j1,j2,j3)]$ denote multiple Ito integrals described above, and Delta$[n]$ denotes the step size at the n-th step.

This routine is part of the **stochastic** package and is usually loaded via the call **with(stochastic)**. It can also be invoked directly via the call **stochastic[wktay3]**.

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

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

that is 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

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

where r and s are constants.

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

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

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

The resulting order 3.0 weak Taylor scheme scheme is

$$\begin{aligned} \begin{pmatrix} Y^1_{n+1} \\ Y^2_{n+1} \end{pmatrix} &= \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} + \begin{pmatrix} Y^2_n \\ Y^1_n \end{pmatrix} \Delta_n + \begin{pmatrix} r \\ s \end{pmatrix} \Delta W^1_n + \begin{pmatrix} s \\ r \end{pmatrix} \Delta W^2_n \\ &\quad + \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} I_{(0,0);n} + \begin{pmatrix} s \\ r \end{pmatrix} I_{(1,0);n} + \begin{pmatrix} r \\ s \end{pmatrix} I_{(2,0);n} \\ &\quad + \begin{pmatrix} Y^1_n \\ Y^2_n \end{pmatrix} I_{(0,0,0);n}, + \begin{pmatrix} s \\ r \end{pmatrix} I_{(1,0,0);n} + \begin{pmatrix} r \\ s \end{pmatrix} I_{(2,0,0);n} \end{aligned}$$

SEE ALSO: **stochastic**, **stochastic[L0]**, **stochastic[MLJ]**,**stochastic[wkeuler]**, **stochastic[wktay2].**

9 APPENDIX

9.1 Subprocedures for momenteqn

The following procedures are subprocedures for the calculation or transformtion parts of (5.3). For example, the procedure **position** determines the position in the new vector and the procedure **ap** transforms the product AP in a vector equation.

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

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:

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:

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:
```

9.2 The inverse procedure pvector2pmatrix

The procedure **pvector2pmatrix** is the inverse of the procedure **matrix2pvector**. It transforms a vector to an symmetry matrix and has following code:

```
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 : Consider the 15-dimensional vector

$$\tilde{p} = [1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13,\ 14,\ 15].$$

The result of **pvector2pmatrix($\tilde{p}$);** gives the 5×5 symmetric matrix P.

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

References

[1] R.E. Crandall, *Topics in Advanced Scientific Computation,* Springer–Verlag, Heidelberg (1996)

[2] S.O. Cyganowski, Solving Stochastic Differential Equations with Maple, *MapleTech Newsletter* **3**(2) (1996), 38–.

[3] 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)

[4] S. Cyganowski and P.E. Kloeden, Stochastic stability examined through MAPLE, in *Proc.* 15*th IMACS World Congress,* Volume 1: Computational Mathematics (Editor: A. Sydow), Wissenschaft & Technik Verlag, Berlin, 1997, 4372–4377.

[5] W. Gander and J. Hrebicek, *Solving Problems in Scientific Computing using Maple and Matlab,* Second Eddition, Springer–Verlag, Heidelberg (1995)

[6] W.S. Kendall, Computer algebra and stochastic calculus, *Notices Amer. Math. Soc.* **37** (1990), 1254– .

[7] P.E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* Springer-Verlag, Heidelberg (1992)

[8] P.E. Kloeden, E. Platen and H. Schurz, *Numerical Solution of Stochastic Differential Equations through Computer Experiments*, Springer-Verlag, Heidelberg (1993).

[9] P.E. Kloeden and W.D. Scott, Construction of Stochastic Numerical Schemes through Maple, *MapleTech Newsletter* **10** (1993), 60–65.

[10] G.G. Milshtein and M.V. Tret'yakov, Numerical Solution of Differential Equations with Coloured Noise, *J. Stat. Physics,* **77** (1994) 691–715.

[11] E. Valkeila, Computer algebra and stochastic analysis, some possibilities, *CWI Quarterly* **4** (1991), 229–