# Computer algebra resolves a multitude of microscale interactions to model stochastic partial differential equations

A. J. Roberts[*]

December 17, 2005

**Abstract**

The computer algebra routines[1] documented here empower you to reproduce and check many of the details described by Roberts [2]. We consider a small spatial domain, representing a finite element, and apply stochastic centre manifold techniques to derive a one degree of freedom model for the dynamics in the element. The approach automatically parametrises the microscale structures induced by spatially varying stochastic noise within the element. The crucial aspect of this work is that we explore how many noise processes may interact in nonlinear dynamics.

## Contents

**1 Iterative computer algebra derives the model** **2**

1.1 Initialisation . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.2 The deterministic model . . . . . . . . . . . . . . . . . . . . 4

[*]Computational Engineering and Science Research Centre, Department of Mathematics & Computing, University of Southern Queensland, Toowoomba, Queensland 4352, Australia. mailto:aroberts@usq.edu.au

[1] The computer algebra routines are written in the package REDUCE. At the time of writing, information about REDUCE was available via http://www.reduce-algebra.com/. Free demonstration versions of REDUCE were available and will execute these routines.

1.3 Linear noise effects . . . . . . . . . . . . . . . . . . . . . . . . . . 5
1.4 Quadratic noise effects . . . . . . . . . . . . . . . . . . . . . . . . 7
1.5 Transform quadratic noise . . . . . . . . . . . . . . . . . . . . . . 9

**2 Computer algebra analyses the Fokker–Planck dynamics 11**
2.1 Preamble . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.2 Gaussian neutral mode . . . . . . . . . . . . . . . . . . . . . . . . 14
2.3 Quadratic corrections give drift . . . . . . . . . . . . . . . . . . . 15
2.4 Quartic corrections give volatility . . . . . . . . . . . . . . . . . . 17

**3 The Cholesky factorisation of convolution covariances 19**
3.1 Compute the convolutions $h_m(t)$ . . . . . . . . . . . . . . . . . 19
3.2 Integrate to find covariances . . . . . . . . . . . . . . . . . . . . . 20
3.3 The Cholesky factorisation is simpler . . . . . . . . . . . . . . . . 20

**References 21**

## 1 Iterative computer algebra derives the model

Construct a one element model of stochastic modified Burgers' equation

$$\begin{gathered}\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + u + \sigma\phi(x,t) \\ \text{such that}\quad u = 0 \text{ at } x = 0, \pi\,,\end{gathered} \tag{1}$$

to effects quadratic in the noise amplitude $\sigma$, and seeks the normal form where the evolution involves no convolutions. Also, transform the quadratic noise in the evolution. Throughout we adopt the Stratonovich interpretation of stochastic differential equations so that the ordinary rules of calculus apply.

In outline, the algorithm determines the model for each power of the noise in turn, then finally transforms to a weak model by replacing quadratic noises by their long time equivalents (as determined in Section 2).

▷▷ `mnone` ◁◁

```
% see multinoise.pdf for documentation
```

```
◁◁ initialisation ▷▷
◁◁ deterministic model ▷▷
◁◁ linear noise effects ▷▷
◁◁ quadratic noise effects ▷▷
◁◁ transform quadratic noise ▷▷
end;
```

## 1.1 Initialisation

Trivially improve printing.

```
                    ▷▷ initialisation ◁◁
on div; off allfac; on revpri;
factor a,eps,sig;
```

Linearise products of trigonometric functions.

```
                    ▷▷ initialisation ◁◁+
let sin(~a)*cos(~b)=>(sin(a+b)+sin(a-b))/2;
```

Define $\beta_n$ to be the decay rate of linear modes on the element, here $\beta_n = n^2{-}1$, so that the spatial modes decay linearly like $\sin(nx)\exp(-\beta_n t)$.

```
                    ▷▷ initialisation ◁◁+
procedure beta(n); (n^2-1);
```

Define the inverse of the linear operator, $\mathcal{L}^{-1}\sin(nx) = \sin(nx)/\beta_n$, as the linear operator is $\mathcal{L} = -1 - \partial_{xx}$. Note: we only define and use this for $n \geq 2$.

```
                    ▷▷ initialisation ◁◁+
operator linv; linear linv;
let linv(sin(~n*x),x)=>sin(n*x)/beta(n);
```

Paramterise solutions by an evolving amplitude $a(t)$ (or ‘order parameter’). Its evolution is $da/dt = \dot a = g$ .

```
                    ▷▷ initialisation ◁◁+
depend a,t;
let df(a,t)=>g;
```

Then the most basic linear approximation to the dynamics on the element is $u = a \sin x$ where $\dot a = 0$ :

```
                    ▷▷ initialisation ◁◁+
u:=a*sin(x);
g:=0;
```

## 1.2 The deterministic model

Iterate towards a solution of the dterministic problem. The parameter $\epsilon$, `eps`, controls the trunction in nonlinearity. The iteration terminates when the residual of the modified Burgers’ equation is zero to the specified order of nonlinearity. Roberts [1] explained the centre manifold rationale and the computational effectiveness of this simple algorithm.

```
                 ▷▷ deterministic model ◁◁
let eps^5=>0;
repeat begin
    res:=-df(u,t)+df(u,x,2)+u-eps*u*df(u,x);
    g:=g+(gd:=(res where {sin(x)=>1,sin(~n*x)=>0}));
    u:=u+linv(res-gd*sin(x),x);
    showtime;
end until res=0;
```

## 1.3 Linear noise effects

Introduce the noise in its spatial Fourier decomposition

$$\phi(x,t) = \sum_{n=1}^{\infty} \phi_n(t) \sin nx \,.$$

Parametrise the amplitude with $\sigma$. Let `phi(n,{m1,...})` denote convolutions with `exp(-beta(m1)t)...`, that is,

$$\phi_{n,(m_1,m_2,\ldots)} = \exp(-\beta_{m_1} t) \star \exp(-\beta_{m_2} t) \star \cdots \star \phi_n(t)\,;$$

so

$$\partial_t \phi_{n,(m_1,m_2,\ldots)} = -\beta_{m_1} \phi_{n,(m_1,m_2,\ldots)} + \phi_{n,(m_2,\ldots)}\,.$$

Let $T$, `tt`, label the fast time of stochastic fluctuations so we can separate the fast stochastic fluctuations from the slow evolution of the amplitude $a$. Also introduce `xt` to label both the subgrid (fast) spatial scales and the fast time scales so we can group all factors in the fast space-time dynamics.

▷▷ linear noise effects ◁◁

```
depend tt,t;
depend x,xt;
operator phi; depend phi,tt,xt;
let { df(phi(~m,~p),t)=>df(phi(~m,~p),tt)
    , df(phi(~m,~p),tt)=>
      -beta(first(p))*phi(m,p)+phi(m,rest(p))
    };
```

Recall the equation for updates $u'$ and $g'$ is $g' + \mathcal{L}u' = \text{residual}$, where now the operator $\mathcal{L} = \partial_t - 1 - \partial_{xx}$ includes fast time variations. The operator `secular` extracts from the residual all those terms which would generate generate secular growth in the field $u$ and so instead must be placed in the model's evolution $g$. The last rule here comes from integration by parts and is essential in order to eliminate memory integrals (convolutions) in the model evolution.

```
                 ▷▷ linear noise effects ◁◁+
operator secular; linear secular;
let { secular(sin(~m*x),xt)=>0
    , secular(sin(~m*x)*~aa,xt)=>0
    , secular(sin(x),xt)=>1
    , secular(sin(x)*phi(~n,~p),xt)=>
      phi(n,{})/(for each m in p product beta(m))
    };
```

Extend the inverse operator to terms with fast time variations as well as fast (subgrid) space variations. Recursive procedure gungb extracts the non-secular parts of fluctuating $\sin x$ components.

```
                 ▷▷ linear noise effects ◁◁+
procedure gungb(n,p);
if p={} then 0 else
(gungb(n,rest(p))-phi(n,p))/beta(first(p));
let { linv(sin(~m*x),xt)=>sin(m*x)/beta(m)
    , linv(sin(~m*x)*phi(~n,~p),xt)=>phi(n,m.p)*sin(m*x)
    , linv(sin(x)*phi(~n,~p),xt)=>gungb(n,p)*sin(x)
    };
```

Truncate the spatial structure of noise. May include up to about 12 components which takes about 2 minutes computation time on my laptop.

```
                 ▷▷ linear noise effects ◁◁+
noise:=for n:=1:5 sum phi(n,{})*sin(n*x);
```

Now iterate to derive effects linear in the additive noise. This iteration starts from the deterministic model already deduced. Multiplying the noise by $\epsilon$ (eps) as well as its magnitude $\sigma$ has the effect of truncating the asymptotic expansion to errors $\mathcal{O}\big(\sigma^2, \epsilon^5 + \sigma^{5/2}\big)$ instead of errors $\mathcal{O}\big(\sigma^2, \epsilon^5 + \sigma^5\big)$ (depending upon the powers in various let statements).

```
              ▷▷ linear noise effects ◁◁+
let sig^2=>0;
repeat begin
    res:=-df(u,t)+df(u,x,2)+u-eps*u*df(u,x)+eps*sig*noise;
    g:=g+(gd:=secular(res,xt));
    u:=u+linv(res-gd*sin(x),xt);
    showtime;
end until res=0;
```

## 1.4 Quadratic noise effects

Now let $Z_p$ denote multiple convolutions of in time of any term, `zz(a,p)` (though I only use $Z$ for quadratic terms, it may well be able to replace the linear convolutions). That is,

$$Z_{(m_1,m_2,\ldots)} = \exp(-\beta_{m_1} t) \star Z_{(m_2,\ldots)} \quad\text{and}\quad Z_{(\,)} = 1\,.$$

```
              ▷▷ quadratic noise effects ◁◁
operator zz; depend zz,tt,xt;
let { zz(~a,{})=>a
    , df(zz(~a,~p),t)=>df(zz(a,p),tt)
    , df(zz(~a,~p),tt)
      =>-beta(first(p))*zz(a,p)+zz(a,rest(p))
    };
```

To extract quadratic corrections to the evolution, use integration by parts so all non-integrable convolutions are reduced to the cannonical form of the convolution being entirely over one noise in a quadratic term, either $\phi_n\phi_{m,(\ldots)}$ or $\phi_{n,(\ldots)}\phi_m$.

```
              ▷▷ quadratic noise effects ◁◁+
procedure gungd(n,p,m,q);
if (p={})or(q={}) then phi(n,p)*phi(m,q) else
(gungd(n,rest(p),m,q)+gungd(n,p,m,rest(q)))
```

```
/(beta(first(p))+beta(first(q)));
let { secular(sin(x)*zz(~a,~p),xt) =>secular(sin(x)*a,xt)
        /(for each m in p product beta(m))
      , secular(sin(~m*x)*zz(~a,~p),xt)=>0
      , secular(sin(x)*phi(~n,~p)*phi(~m,~q),xt)
        =>gungd(n,p,m,q)
      , secular(sin(x)*phi(~n,~p)^2,xt)=>gungd(n,p,n,p)
      };
```

Extend $\mathcal{L}^{-1}$ operator `linv` to handle quadratic terms. First, integration by parts gives all integrable contributions from direct product terms.

```
                ▷▷ quadratic noise effects ◁◁+
procedure gunge(n,p,m,q);
if (p={})or(q={}) then 0 else (-phi(n,p)*phi(m,q)
+gunge(n,rest(p),m,q)+gunge(n,p,m,rest(q)))
/(beta(first(p))+beta(first(q)));
let { linv(sin(x)*phi(~n,~p)*phi(~m,~q),xt)
        =>gunge(n,p,m,q)*sin(x)
      , linv(sin(x)*phi(~n,~p)^2,xt)=>gunge(n,p,n,p)*sin(x)
      };
```

Second, similar integration by parts gives integrable contribution from terms involving convolutions of products.

```
                ▷▷ quadratic noise effects ◁◁+
procedure gungf(a,p);
if p={} then 0 else
(gungf(a,rest(p))-zz(a,p))/beta(first(p));
let { linv(sin(~l*x)*phi(~n,~p)*phi(~m,~q),xt)
        =>sin(l*x)*zz(phi(n,p)*phi(m,q),{l})
      , linv(sin(~l*x)*phi(~n,~p)^2,xt)
        =>sin(l*x)*zz(phi(n,p)^2,{l})
      , linv(sin(~l*x)*zz(~a,~p),xt)=>sin(l*x)*zz(a,l.p)
      , linv(sin(x)*zz(~a,~p),xt)=>sin(x)*gungf(a,p)
```

```
        };
```

Now iterate to dervie effects quadratic in noise.

```
              ▷▷ quadratic noise effects ◁◁+
    let sig^3=>0;
    repeat begin
        res:=-df(u,t)+df(u,x,2)+u-eps*u*df(u,x)+eps*sig*noise;
        g:=g+(gd:=secular(res,xt));
        u:=u+linv(res-gd*sin(x),xt);
        showtime;
    end until res=0;
```

## 1.5 Transform quadratic noise

Now proceed to transform the strong model to a weak model by replacing the quadratic noises by their effective long term drift and volatility. Section 2 determines the rationale for the details of this transformation.

Set $\epsilon = 1$ as it has done its job of truncating the nonlinear terms in the asymptotic expansion.

```
              ▷▷ transform quadratic noise ◁◁
    eps:=1;
    write "Now transforming the quadratic noises";
```

Now transform the quadratic noise into new noises $\psi$ (`psi`) that are equivalent in their long time statistics: the operator `long` implements the long-time equivalent noises as determined in Section 2. For now only transform up to two convolutions. These new noises have subscripts that uniquely identify them.

```
              ▷▷ transform quadratic noise ◁◁+
    operator long; linear long;
    operator psi; depend psi,tt,xt;
```

```
let { long(1,tt)=>1
    , long(phi(~i,{}),tt)=>phi(i,{})
    , long(phi(~i,{})*phi(~j,{~k}),tt)
      => 1/2*(if i=j then 1 else 0)
      +psi(i,j,{k})/sqrt(2*beta(k))
    , long(phi(~i,{})*phi(~j,{~k2,~k1}),tt)
      => (psi(i,j,{k1})/sqrt(2*beta(k1))
      +psi(i,j,{k2,k1})/sqrt(2*beta(k2)))
      /(beta(k1)+beta(k2))
    };
gg:=long(g,tt);
```

Root sum squares of the determined noise coefficients; this procedure implicitly assumes that there is no correlation between the multitude of noises in these two terms in the amplitude equation. This assumption appears correct for these two terms in this PDE. In general we would need to do a QR factorisation of the noise terms.

```
            ▷▷ transform quadratic noise ◁◁+
operator sumsqpsi; linear sumsqpsi;
let { sumsqpsi(1,tt)=>0
    , sumsqpsi(psi(~i,~j,~p),tt)=>0
    , sumsqpsi(psi(~i,~j,~p)^2,tt)=>1
    , sumsqpsi(psi(~i,~j,~p)*psi(~ii,~jj,~pp),tt)=>0
    };
```

Have a look at the numerical coefficients.

```
            ▷▷ transform quadratic noise ◁◁+
on rounded;
gg:=gg;
```

Extract the coefficients of the terms in $\sigma^2$ and $\sigma^2 a$, both mean and fluctuating.

```
            ▷▷ transform quadratic noise ◁◁+
c20:=sqrt(sumsqpsi(coeffn(coeffn(gg,sig,2),a,0)^2,tt));
c21mean:=(coeffn(coeffn(gg,sig,2),a,1)
    where psi(~i,~j,~p)=>0);
c21:=sqrt(sumsqpsi(coeffn(coeffn(gg,sig,2),a,1)^2,tt));
```

Switch back to the rational arithmetic mode for any postprocessing.

```
            ▷▷ transform quadratic noise ◁◁+
off rounded;
showtime;
```

## 2 Computer algebra analyses the Fokker–Planck dynamics

This analyses the case of one Stratonovich noise nonlinearly compounding on itself or on another depending upon parameter same. The *canonical* dynamical system for a noise interacting with itself or another noise is

$$\begin{aligned}
dy_1 &= z_1\,dW\,, & dz_1 &= -\beta_1 z_1\,dt + d\hat W\,,\\
dy_2 &= z_2\,dW\,, & dz_2 &= (-\beta_2 z_2 + z_1)dt\,,\\
dy_3 &= z_3\,dW\,, & dz_3 &= (-\beta_3 z_3 + z_2)dt\,,\\
&\vdots & &\vdots
\end{aligned}$$

That is, in terms of noises $\phi = \frac{dW}{dt}$ and $\phi' = \frac{d\hat W}{dt}$, consider the dynamics of

$$\begin{aligned}
\frac{dy_1}{dt} &= z_1\,\phi(t)\,, & z_1 &= \exp(-\beta_1 t)\star\phi'(t)\,,\\
\frac{dy_2}{dt} &= z_2\,\phi(t)\,, & z_2 &= \exp(-\beta_2 t)\star z_1(t)\\
& & &= \exp(-\beta_2 t)\star\exp(-\beta_1 t)\star\phi'(t)\,,\\
\frac{dy_3}{dt} &= z_3\,\phi(t)\,, & z_3 &= \exp(-\beta_3 t)\star z_2(t)\,,
\end{aligned}$$

$$\vdots \qquad \vdots$$

Let `no` be the number of pairs of equations in the cannonical system. The length of the algebraic expressions in the analysis is proportional to `no`$^6$ so the compute time is large, but could be worse.

In the above cannoical system the $z$ variables, being convolutions over rapid times of the Wiener noise, will rapidly approach an equilibrium probability distribution (centred upon $0$). Then the $y$ variables will evolve relatively slowly depending upon how the fluctuations in $y$ correlate with those of the noise. This two observations imply that over long times we can factor the probability distribution in $\mathbf{y}$ and $\mathbf{z}$. Thus express the probability density function (PDF) as

$$u(\mathbf{y},\mathbf{z},t) = pG_0(\mathbf{z}) + \frac{\partial p}{\partial \mathbf{y}}G_1(\mathbf{z}) + \cdots$$

where the conditional probablity $p(y,t)$ evolves according to

$$\frac{\partial p}{\partial t} = G(p)\,.$$

**Parameters** The first of two crucial parameters is `same` which is set to $1$ when the noises are the same, that is, $W = \hat{W}$, and which is set to $0$ when the noises $W$ and $\hat{W}$ are independent Wiener processes. ▷▷ mnfp ◁◁

```
same:=1;
```

Set the number of pairs of equations in the cannonical system; that is, `no` is the maximum number of convolutions of the noises. ▷▷ mnfp ◁◁+

```
no:=2;
```

The decay rates $\beta_n$ of the modes is denoted by `beta` which we may set to be specific values, such as one or $n$, for speed at less generality. ▷▷ mnfp ◁◁+ `operator beta; let beta(~n)=>n;`

**Overview the algorithm** I use the parameter $\epsilon$ (`eps`) to count the number of $\partial_y$ derivatives. Compute the asymptotic expansion in three steps to determine terms in increasing powers of $\epsilon$.

▷▷ mnfp ◁◁+

```
% see multinoise.pdf for documentation
◁◁ preamble ▷▷
◁◁ gaussian ▷▷
◁◁ drift ▷▷
◁◁ volatility ▷▷
end;
```

## 2.1 Preamble

Improve printing.

▷▷ preamble ◁◁

```
on div; off allfac; on revpri;
factor amp,eps,z;
```

Variables and unknown coefficients in expressions involve the following subscripted quantities.

▷▷ preamble ◁◁+

```
operator a;
operator b;
operator c;
operator g;
operator y;
operator z;
```

Reduce knows too much about the exponential function, so here define the operator `ex` to denote the exponential function but with only as many properties as we need to know.

▷▷ preamble ◁◁+

```
operator ex;
let { ex(0)=>1
```

```
        , df(ex(~aa),~bb)=>ex(aa)*df(aa,bb)
        };
```

Define the conditional probability density function $p(\mathfrak{y}, t)$ which is to evolve in time according to $\frac{\partial p}{\partial t} = gg$.

▷▷ preamble ◁◁+

```
depend p,t;
for i:=1:no do depend p,y(i);
let df(p,t) => gg;
```

## 2.2 Gaussian neutral mode

First find the Gaussian $G_0(\mathfrak{z})$ such that the PDF $u = p(\mathfrak{y}, t)G_0(\mathfrak{z}) + \mathcal{O}(\epsilon)$ where you recall the parameter $\epsilon$ counts the number of $\mathfrak{y}$ derivatives. In this section discard all derivatives in $\mathfrak{y}$.

Form the generic quadratic $\sum_{i,j=1,j\leq i}^{no} a_{i,j}z_iz_j$ and the list $\{a_{i,j}\}$ of as yet unknown coefficients.

▷▷ gaussian ◁◁

```
quad:=for i:=1:no sum for j:=1:i sum a(i,j)*z(i)*z(j)$
quav:=for i:=1:no join for j:=1:i collect a(i,j)$
```

Set the evolution to zero, the Gaussian $G_0$ to be proportional to an exponential of the general quadratic, and the PDF `u` to $p$ times the as yet unknown Gaussian.

▷▷ gaussian ◁◁+

```
gg:=0;
gauss:=amp*ex(quad)$
u:=gauss*p$
```

Substitute this factored PDF into the Fokker–Planck equation for the Stratonovich system of SDEs of cannonical nonlinearly coupled processes.

Compute the residual of the Fokker–Planck equation, ignoring all $\mathfrak{y}$ derivatives for simplicity and because the errors are $\mathcal{O}(\epsilon)$.

```
                    ▷▷ gaussian ◁◁+
    res:=-df(u,t) +df(beta(1)*z(1)*u,z(1))
        +(for i:=2:no sum df((-z(i-1)+beta(i)*z(i))*u,z(i)))
        +1/2*df(df(u,z(1)),z(1))$
```

The residual is a multivariate in $\mathfrak{z}$ so find the coefficients, but recursivley finding all coefficients of each variable $z_i$. Then solve to make the residual zero. Substitute into the Gaussian $G_0$ ready for the next stages.

```
                    ▷▷ gaussian ◁◁+
    eqn:={res/gauss/p}$
    for i:=1:no do eqn:=(for each j in eqn join coeff(j,z(i)))$
    sol0:=solve(eqn,quav);
    gauss:=sub(part(sol0,1),gauss);
    showtime;
```

## 2.3 Quadratic corrections give drift

Terms of first order in $\mathfrak{y}$ derivatives may generate drift in the long term dynamics. Truncate the computations to errors $\mathcal{O}(\epsilon^2)$ to keep all terms up to linear in $\mathfrak{y}$ derivatives.

```
                    ▷▷ drift ◁◁
    let eps^2=>0;
```

Express the terms liner in $\mathfrak{y}$ derivatives in a general form for both the evolution and the modification to the Gaussian $G_0$. I find that the coefficients $b_{i,j,k}$ of the quadratic corrections are determined, but that the coefficients $b_i$ are not. This indeterminancy is because the coefficients $b_i$ multiply the Gaussian homogeneous solution; these coefficients are only determined by precisely defining the amplitude which is not necessary in this problem as the equations are *linear* in probability density functions $u$ and $p$. This non-necessity is verified by the ultimate lack of dependence of the evolution upon the undetermined coefficients $b_i$.

```
                    ▷▷ drift ◁◁+
gg:=-eps*(for i:=1:no sum g(i)*df(p,y(i)))$
u:=gauss*p +eps*gauss*(for i:=1:no sum df(p,y(i))*(b(i)+
    for j:=1:no sum for k:=1:j sum b(i,j,k)*z(j)*z(k) ) )$
```

Form a list of the coefficients to be determined.

```
                    ▷▷ drift ◁◁+
vars:=for i:=1:no join g(i)
    .(for j:=1:no join for k:=1:j collect b(i,j,k))$
```

Substitute this PDF into the Fokker–Planck equation for the Stratonovich system of SDEs of cannonical nonlinearly coupled processes. Compute the residual of the Fokker–Planck equation, ignoring all terms of second order in the $\mathfrak{y}$ derivatives for simplicity.

```
                    ▷▷ drift ◁◁+
res:=-df(u,t) +df(beta(1)*z(1)*u,z(1))
    +(for i:=2:no sum df((-z(i-1)+beta(i)*z(i))*u,z(i)))
    +1/2*df(df(u,z(1)),z(1))
    +eps*same/2*(for i:=1:no sum
      (df(z(i)*df(u,z(1)),y(i))+df(z(i)*df(u,y(i)),z(1))))$
```

The residual is a multivariate in $z$ and $\partial_{\mathfrak{y}}$ so find the coefficients and solve to make the residual zero. Then substitute into the Gaussian and the evolution ready for the next stages.

```
                    ▷▷ drift ◁◁+
res:=res/gauss/eps$
eqn:=for i:=1:no collect coeffn(res,df(p,y(i)),1)$
for i:=1:no do eqn:=for each j in eqn join coeff(j,z(i))$
```

```
sol1:=solve(eqn,vars);
u:=sub(part(sol1,1),u)$
gg:=sub(part(sol1,1),gg);
showtime;
```

## 2.4 Quartic corrections give volatility

Terms quadratic in $\mathfrak{y}$ derivatives generate the volatility terms through the diffusion matrix. Truncate the computations to errors $\mathcal{O}\big(\epsilon^3\big)$ to keep all terms up to quadratic in $\mathfrak{y}$ derivatives.

```
                    ▷▷ volatility ◁◁
let eps^3=>0;
```

Express the terms quadratic in $\mathfrak{y}$ derivatives in a general form for both the evolution and the modification to the Gaussian. We find that the coefficients $c_{i,j,k,l}$ and $c_{i,j,k,l,m,n}$ of the quadratic corrections are determined, but that the coefficients $c_{i,j}$ are not. Again, this is because the coefficients $c_{i,j}$ multiply the Gaussian homogeneous solution.

```
                    ▷▷ volatility ◁◁+
gg:=gg+eps^2*(for i:=1:no sum
              for j:=1:i sum g(i,j)*df(p,y(i),y(j)))$
u:=u +eps^2*gauss*(
    for i:=1:no sum for j:=1:i sum df(p,y(i),y(j))*(c(i,j)+
    for k:=1:no sum for l:=1:k sum (c(i,j,k,l)*z(k)*z(l)+
    for m:=1:l  sum for n:=1:m sum
        c(i,j,k,l,m,n)*z(k)*z(l)*z(m)*z(n) ) ) )$
```

Form a list of the coefficients to be determined.

```
                    ▷▷ volatility ◁◁+
vars:=for i:=1:no join for j:=1:i join g(i,j).(
    for k:=1:no join for l:=1:k join c(i,j,k,l).(
        for m:=1:l  join for n:=1:m collect c(i,j,k,l,m,n) ) )$
```

Substitute this PDF into the Fokker–Planck equation for the Stratonovich system of SDEs of cannonical nonlinearly coupled processes. Compute the residual of the full Fokker–Planck equation.

```
                     ▷▷ volatility ◁◁+
    res:=-df(u,t) +df(beta(1)*z(1)*u,z(1))
        +(for i:=2:no sum df((-z(i-1)+beta(i)*z(i))*u,z(i)))
        +1/2*df(df(u,z(1)),z(1))
        +eps*same/2*(for i:=1:no sum
          (df(z(i)*df(u,z(1)),y(i))+df(z(i)*df(u,y(i)),z(1))))
        +eps^2/2*(for i:=1:no sum df(
             (for j:=1:no sum z(i)*z(j)*df(u,y(j))),y(i)))$
```

The residual is a multivariate in $\boldsymbol{z}$ and $\partial_{\boldsymbol{y}\boldsymbol{y}}$ ($\partial_{\boldsymbol{y}}$ terms do not appear in this residual as we have already found the correct terms in $\partial_{\boldsymbol{y}}$) so find the coefficients and solve to make the residual zero. Then substitute into the Gaussian and the evolution.

```
                     ▷▷ volatility ◁◁+
    eqn:=for i:=1:no join for j:=1:i collect
            coeffn(res/gauss/eps^2,df(p,y(i),y(j)),1)$
    for i:=1:no do eqn:=for each j in eqn join coeff(j,z(i))$
    sol2:=solve(eqn,vars);
    u:=sub(part(sol2,1),u)$
    gg:=sub(part(sol2,1),gg);
    showtime;
```

Then interpret the evolution in gg as the Fokker–Planck equation of the long term model of the cannonical system. Thus Cholesky decompose the diffusion matrix to discover independent equivalent noise processes that were used in Section 1.5 to transform the products of noise processes.

## 3 The Cholesky factorisation of convolution covariances

In this section use Ito interpretation of SDEs. We find the covariance matrix $\operatorname{E}[Z_m Z_n]$ where by definition

$$Z_1 = \int_{-\infty}^{t} e^{-\beta_1 (t-s)} d\hat{W}_s \quad \text{and} \quad Z_m = \int_{-\infty}^{t} e^{-\beta_m (t-s)} Z_{m-1}(s)\, ds\,.$$

The first is an Ito integral. Turn the others into an Ito integral by defining

$$h_1(t) = e^{-\beta_1 t} \quad \text{and} \quad h_m(t) = e^{-\beta_m t} \star h_{m-1}(t) = \int_0^t e^{-\beta_m (t-s)} h_{m-1}(s)\, ds\,,$$

then

$$Z_m = \int_{-\infty}^{t} h_m(t-s)\, d\hat{W}_s\,.$$

Consequently, by an extension of the Ito isometry

$$\operatorname{E}[Z_m Z_n] = \int_0^\infty h_m(t) h_n(t)\, dt\,.$$

In the computer algebra set the maximum order of computation into the parameter `▷▷ mnchol ◁◁ o:=4;`

Then the computation follows this outline.

```
                ▷▷ mnchol ◁◁+
% see multinoise.pdf for documentation
◁◁ convolutions h ▷▷
◁◁ covariance matrix ▷▷
◁◁ Cholesky factorisation ▷▷
end;
```

### 3.1 Compute the convolutions $h_m(t)$

The switch `gcd` helps cancel out common factors between numerators and denominators. Then recursively find the convolutions.

```
                    ▷▷ convolutions h ◁◁
on gcd;
array h(o);
h(1):=exp(-b1*t);
for m:=2:o do write h(m):=int(
    exp(-mkid(b,m)*(t-s))*sub(t=s,h(m-1)) ,s,0,t);
```

## 3.2 Integrate to find covariances

Then integrate: assume each term has an exponential in it so that the evaluation of the returned integral at $t = \infty$ is zero, hence just evaluate at $t = 0$.

```
                    ▷▷ covariance matrix ◁◁
array dd(o,o),dnum(o,o),dden(o,o);
for m:=1:o do for n:=1:m do begin
    write dd(m,n):=-sub(t=0, int(h(m)*h(n),t) );
```

For later recording, factorize the numerator and denominator to try to simply the expressions.

```
                    ▷▷ covariance matrix ◁◁+
    write dnum(m,n):=factorize(num(dd(m,n)));
    write dden(m,n):=factorize(den(dd(m,n)));
end;
```

## 3.3 The Cholesky factorisation is simpler

Given the covariances in dd, compute the Cholesky factorisation.

```
                    ▷▷ Cholesky factorisation ◁◁
array ll(o,o),lden(o,o);
for m:=1:o do begin
```

```
        for n:=1:m-1 do write ll(m,n):=(dd(m,n)
            -(for k:=1:n-1 sum ll(m,k)*ll(n,k))
            )/ll(n,n);
        ll(m,m):=sqrt(dd(m,m)-(for n:=1:m-1 sum ll(m,n)^2));
```

However, REDUCE litters the resultant expressions with the `abs()` function; so explicitly get rid of them.

```
              ▷▷ Cholesky factorisation ◁◁+
        write ll(m,m):=(ll(m,m) where abs(~aa)=>aa);
```

For later recording, factorize the denominator to try to simply the expression.

```
              ▷▷ Cholesky factorisation ◁◁+
        for n:=1:m do write lden(m,n):=factorize(den(ll(m,n)));
    end;
```

Check the factorisation; these should all be zero.

```
              ▷▷ Cholesky factorisation ◁◁+
    for m:=1:o do for n:=1:m do
    write err:=-dd(m,n)+(for k:=1:n sum ll(m,k)*ll(n,k));
```

## References

[1] A. J. Roberts. Low-dimensional modelling of dynamics via computer algebra. *Computer Phys. Comm.*, 100:215–230, 1997. 4

[2] A. J. Roberts. Resolve the multitude of microscale interactions to model stochastic partial differential equations. Technical report, http://arxiv.org/abs/math.DS/0506533, June 2005. Submitted to LMS Journal of Computation and Mathematics. 1