Computer algebra derives normal forms of stochastic differential equations

A. J. Roberts[*]

January 23, 2007

Abstract

Modelling stochastic systems has many important applications. Normal form coordinate transforms are a powerful way to untangle interesting long term dynamics from undesirably detailed microscale dynamics. I aim to explore normal forms of stochastic differential equations when the dynamics has both slow modes and quickly decaying modes. The thrust is to derive normal forms useful for macroscopic modelling of detailed microscopic systems. Thus we not only must reduce the dimensionality of the dynamics, but also endeavour to remove all fast time processes. Sri Namachchivaya, Leng and Lin (1990–1) emphasise the importance of quadratic stochastic effects "in order to capture the stochastic contributions of the stable modes to the drift terms of the critical modes." I derive such important quadratic effects using the normal form coordinate transform to separate slow and fast modes. The results will help us accurately model multiscale stochastic systems.

Contents

[*]Computational Engineering and Sciences Research Centre, Department of Mathematics & Computing, University of Southern Queensland, Toowoomba, Queensland 4352, Australia. http://www.sci.usq.edu.au/staff/robertsa

[0]This technical report published via Roberts (2007).

1 Introduction

Stochastic ODEs and PDEs have many important applications. Normal form coordinate transforms are a powerful way to untangle interesting long time dynamics from uninteresting but necessary short time dynamics (Murdock 2003, Arnold 2003, e.g.). Here we further explore normal forms of SDEs when the dynamics of the SDE has both slow modes and quickly decaying modes (Arnold & Imkeller 1998, e.g.). The thrust is to derive normal forms useful for macroscopic modelling of detailed microscopic stochastic systems. Thus we endeavour to remove all fast time processes from the slow modes (Chao & Roberts 1996, Roberts 2006*b*, e.g.). In contrast, almost all previous approaches have been content to derive normal forms that support reducing the dimensionality of the dynamics. Here we go further than other researchers and *additionally remove fast time integrals in the slow modes.*

To derive a normal form from which to extract a stochastic slow/centre manifold, we construct coordinate transformations that "simplify" a SDE system. I invoke the following principles:

1. Avoid unbounded (secular) terms in the transformation and the evolution;

2. Decouple the slow dynamics from the fast;

3. Insist that the stochastic slow manifold is precisely the transformed fast modes being zero;

4. Ruthlessly eliminate as many as possible of the terms in the evolution;

5. Avoid as far as possible fast time memory integrals in the evolution.

Sri Namachchivaya & Leng (1990) and Sri Namachchivaya & Lin (1991) emphasise the importance of quadratic stochastic effects "in order to capture the stochastic contributions of the stable modes to the drift terms of the critical modes." Here we capture such important quadratic effects in the quadratic noise terms.

This report documents the iterative computer algebra used to construct stochastic normal forms of a simple example system of SDEs, the Duffing–van der Pol oscillator:

$$\ddot{x}_1 = (\alpha + \sigma\phi(t))x_1 + \beta\dot{x}_1 - x_1^3 - x_1^2\dot{x}_1\,, \tag{1}$$

where ϕ is some white noise. Arnold, Sri Namachchivaya & Schenk-Hoppé (1996) describe the importance of the stochastic system (1) in applications. Section 2 analyses a stochastic pitchfork bifurcation, whereas Section 3 analyses a stochastic Hopf bifurcation.

In the constrcution, convolutions of the noise arise, both backwards over history and forwards to anticipate the noise. For any non-zero parameter μ, define the convolution

$$e^{\mu t}\star\phi = \begin{cases} \int_{-\infty}^{t} \exp[\mu(t-\tau)]\phi(\tau)\,d\tau\,, & \mu<0\,,\\ \int_{t}^{+\infty} \exp[\mu(t-\tau)]\phi(\tau)\,d\tau\,, & \mu>0\,, \end{cases} \tag{2}$$

so that the convolution is always with a bounded exponential (Principle 1). Five useful properties of this convolution are

$$e^{\mu t}\star 1 = \frac{1}{|\mu|}\,, \tag{3}$$

$$\frac{d}{dt}e^{\mu t}\star\phi = -\operatorname{sgn}\mu\,\phi + \mu e^{\mu t}\star\phi\,, \tag{4}$$

$$E[e^{\mu t}\star\phi] = e^{\mu t}\star E[\phi]\,, \tag{5}$$

$$E[(e^{\mu t}\star\phi)^2] = \frac{1}{2|\mu|}\,, \tag{6}$$

$$e^{\mu t}\star e^{\nu t}\star = \begin{cases} \frac{1}{|\mu-\nu|}\left[e^{\mu t}\star + e^{\nu t}\star\right], & \mu\nu < 0\,,\\ \frac{-\operatorname{sgn}\mu}{\mu-\nu}\left[e^{\mu t}\star - e^{\nu t}\star\right], & \mu\nu > 0 \;\&\; \mu\neq\nu\,. \end{cases} \tag{7}$$

Also remember that although with $\mu < 0$ the convolution $e^{\mu t}\star$ integrates over the past, with $\mu > 0$, as we will soon need, the convolution $e^{\mu t}\star$ integrates into the future over a time scale of order $1/\mu$.

2 Derive a normal form of a noisy pitchfork bifurcation

Construct a stochastic normal form for the pitchfork bifurcation of the Duffing–van der Pol oscillator (1) where for a pitchfork bifurcation set $\beta = -1$ and vary the parameter α through zero. Recall that linearly, the slow variable is $x = x_1 + \dot x_1$ and the fast variable $y = \dot x_1$. I transform these coordinates to find a normal form of the SDE.

Introduce a parameter ϵ to count the number of variables in nonlinear terms (here envisage $\epsilon = X^2 + Y^2$); thus solve

$$\ddot x_1 = (\alpha + \sigma\phi(t))x_1 - \dot x_1 - \epsilon(x_1^3 + x_1^2\dot x_1)\,. \tag{8}$$

This code will work with any two variable, one noise, coupled pair of SDEs where the linear dynamics has eigenvalues 0 and -1.

▷▷ duffpol ◁◁

```
% see cadnfsde.pdf for documentation
on div; off allfac; on revpri;
```

The variable `epsilon` represents ϵ. Scale the bifurcation parameter α and the noise magnitude parameter σ together with the parameter `del`; in essence `del` measures $\varepsilon = X^2 + Y^2 + \alpha + \sigma$. This parameter `del`, ε, conveniently controls the truncation of the multivariate asymptotic expansions.

▷▷ duffpol ◁◁+

```
factor epsilon,alpha,sigma,del;
alpha:=alf*del;
sigma:=sig*del;
epsilon:=eps*del;
```

2.1 Represent the noise

Represent the parametric noise by `phi` which depends upon time. But we find it useful to discriminate upon the notionally fast time fluctuations of the noise processes, and the notionally ordinary time variations of the dynamic variables x and y. Thus introduce a notionally fast time variable `tt`, which depends upon the ordinary time `t`. Equivalently, view `tt`, a sort of 'partial t', as representing variations in time independent of those in the variables x and y.

▷▷ duffpol ◁◁+

```
depend phi,tt;
depend tt,t;
```

The operator `z(f,tt,mu)` represents the convolution $e^{\mu t}\star f$ as defined by (2). It is linear over fast time `tt` as the convolution only arises from solving PDEs in the operator $\partial_t - \mu$. Code its derivative (4) and its action upon deterministic terms (3):

▷▷ duffpol ◁◁+

```
operator z; linear z;
let { df(z(~f,tt,~mu),t)=>-sign(mu)*f+mu*z(f,tt,mu)
    , z(1,tt,~mu)=>1/abs(mu)
```

Also code the transform (7) that successive convolutions at different rates may be transformed into several single convolutions.

▷▷ duffpol ◁◁+

```
    , z(z(~r,tt,~nu),tt,~mu) =>
```

```
      (z(r,tt,mu)+z(r,tt,nu))/abs(mu-nu) when (mu*nu<0)
    , z(z(~r,tt,~nu),tt,~mu) =>
      -sign(mu)*(z(r,tt,mu)-z(r,tt,nu))/(mu-nu)
      when (mu*nu>0)and(mu neq nu)
    };
```

The above properties are *critical*: they must be correct for the resulting transform to be correct.

2.2 Set basic approximation

Express the normal form in terms of new evolving variables X and Y, denoted by `xx` and `yy`, which are nonlinear modifications to x and y.

▷▷ duffpol ◁◁+

```
depend yy,t;
depend xx,t;
let { df(xx,t)=>ff, df(yy,t)=>gg };
```

The first linear approximation is then $x \approx X$ and $y \approx Y$ such that $\dot{X} \approx 0$ (in `ff`) and $\dot{Y} \approx -Y$ (in `gg`).

▷▷ duffpol ◁◁+

```
x:=xx;
y:=yy;
ff:=0;
gg:=-yy;
```

2.3 Solve homological equation with noise

When solving homological equations of the form $F + \xi_t = \text{Res}$ (the resonant case $\mu = 0$), we separate the terms in the right-hand side Res into those that are integrable in fast time, and hence modify the coordinate transform by changing ξ, and those that are not, and hence must remain in the evolution by changing F. the operator zint extracts those parts of a term that we know are integrable; the operator znon extracts those parts which are not. Note: with more research, more types of terms may be found to be integrable; hence what is extracted by zint and what is left by zint may change with more research. These transforms are not critical: changing the transforms may change the results, but as long as the iteration converges, the computer algebra results will be algebraically correct.

```
                         ▷▷ duffpol ◁◁+
operator zint; linear zint;
operator znon; linear znon;
```

First, avoid obvious secularity.

```
                         ▷▷ duffpol ◁◁+
let { zint(phi,tt)=>0,    znon(phi,tt)=>phi
    , zint(1,tt)=>0,      znon(1,tt)=>1
    , zint(phi*~r,tt)=>0, znon(phi*~r,tt)=>phi*r
```

Second, by (4) a convolution may be split into an integrable part, and a part in its argument which in turn may be integrable or not.

```
                         ▷▷ duffpol ◁◁+
, zint(z(~r,tt,~mu),tt)=>z(r,tt,mu)/mu+zint(r,tt)/abs(mu)
, znon(z(~r,tt,~mu),tt)=>znon(r,tt)/abs(mu)
```

Third, squares of convolutions may be integrated by parts to an integrable term and a part that may have integrable or non-integrable parts.

```
                         ▷▷ duffpol ◁◁+
, zint(z(~r,tt,~mu)^2,tt)=>z(~r,tt,~mu)^2/(2*mu)
                        +zint(r*z(r,tt,mu),tt)/abs(mu)
, znon(z(~r,tt,~mu)^2,tt)=>znon(r*z(r,tt,mu),tt)/abs(mu)
```

Fourth, different products of convolutions may be similarly separated using integration by parts.

```
                    ▷▷ duffpol ◁◁+
, zint(z(~r,tt,~mu)*z(~s,tt,~nu),tt)
  =>z(r,tt,mu)*z(s,tt,nu)/(mu+nu)
  +zint(sign(mu)*r*z(s,tt,nu)+sign(nu)*s*z(r,tt,mu),tt)
  /(mu+nu) when mu+nu neq 0
, znon(z(~r,tt,~mu)*z(~s,tt,~nu),tt)=>
  +znon(sign(mu)*r*z(s,tt,nu)+sign(nu)*s*z(r,tt,mu),tt)
  /(mu+nu) when mu+nu neq 0
```

However, a zero divisor arises when $\mu+\nu=0$ in the above. Here code rules to cater for such terms by increasing the depth of convolutions over past history.

```
                    ▷▷ duffpol ◁◁+
, zint(z(~r,tt,~mu)*z(~s,tt,~nu),tt)=>
  z(z(r,tt,-nu),tt,-nu)*z(s,tt,nu)
  +zint(z(z(r,tt,-nu),tt,-nu)*s,tt) when (mu+nu=0)and(nu>0)
, znon(z(~r,tt,~mu)*z(~s,tt,~nu),tt)=>
  znon(z(z(r,tt,-nu),tt,-nu)*s,tt) when (mu+nu=0)and(nu>0)
```

The above handles quadratic products of convolutions. Presumably, if we seek cubic noise effects then we may need cubic products of convolutions. However, I do not proceed so far and hence terminate the separation rules.

```
                    ▷▷ duffpol ◁◁+
};
```

2.4 Use iteration to solve

Now we iterate to a solution of the governing equations to residuals of some order of error. To compare with the analysis by Arnold & Imkeller (1998) ofthe Duffing–van der Pol oscillator we could truncate to the same order they did by satisfying the SDE (8) to residuals $\mathcal{O}\big(\epsilon^2, \alpha^3+\sigma^3\big)$. However, for now I choose to instead solve to residuals $\mathcal{O}\big(\epsilon^3\big)$.

```
                    ▷▷ duffpol ◁◁+
let del^3=>0;
repeat begin
  ◁◁ update fast xform ▷▷
  ◁◁ update slow xform ▷▷
  showtime;
end until {resx,resy}={0,0};
end;
```

Compute the residual of the y equation and trace print its length for information. Since $y = \dot{x}_1$, the Duffing–van der Pol equation (8) codes as follows. Simply change the computation of the residual in order to explore a different stochastic system.

```
                ▷▷ update fast xform ◁◁
x1:=x-y;
resy:=-df(y,t)+(alpha+sigma*phi)*x1-y-epsilon*(x1^3+x1^2*y);
```

Trace print the length of the residual to check how the iteration is progressing.

```
                ▷▷ update fast xform ◁◁+
write lengthresy:=length(resy);
```

Update the Y evolution gg and the y transform: first, extract the various powers of yy to account for the different possibilities of the homological equation (appending an extra zero simply to ensure there are at least two elements in cs); second, the terms linear in Y are resonant and so the non-integrable parts must go into the evolution gg; lastly, the remaining terms get convolved at the appropriate rate to solve their respective homological equation.

```
                ▷▷ update fast xform ◁◁+
cs:=append(coeff(resy,yy),{0});
gg:=gg+znon(part(cs,2),tt)*yy;
```

```
y:=y+z(part(cs,1),tt,-1) +zint(part(cs,2),tt)*yy
  -(for k:=3:length(cs) sum z(part(cs,k),tt,k-2)*yy^(k-1));
```

Compute the residual of the x equation. Since $x = x_1 + \dot{x}_1$, then $\dot{x} = \dot{x}_1 + \ddot{x}_1$, and the Duffing–van der Pol equation (8) implies the following code for the residual. Simply change the computation of the residual in order to explore a different stochastic system.

$\rhd\rhd$ update slow xform $\lhd\lhd$

```
x1:=x-y;
resx:=-df(x,t)+(alpha+sigma*phi)*x1  -epsilon*(x1^3+x1^2*y);
```

Trace print the length of this residual.

$\rhd\rhd$ update slow xform $\lhd\lhd+$

```
write lengthresx:=length(resx);
```

Update the X evolution `ff` and the x transform: first, extract the various powers of `yy` to account for the different possibilities of the homological equation; second, the terms independent of Y are resonant and so the non-integrable parts must go into the evolution `ff`; lastly, the remaining terms get convolved at the appropriate rate to solve their respective homological equation.

$\rhd\rhd$ update slow xform $\lhd\lhd+$

```
cs:=coeff(resx,yy);
ff:=ff+znon(part(cs,1),tt);
x:=x+zint(part(cs,1),tt)
  -(for k:=2:length(cs) sum z(part(cs,k),tt,k-1)*yy^(k-1));
```

Finished.

2.5 Duffing–van der Pol pitchfork bifurcation

This section briefly reports on the normal form of the pitchfork bifurcation. In particular, this section confirms that there is no need for the anticipatory convolutions Arnold & Xu Kedai (1993) and Arnold & Imkeller (1998) record in the evolution for this particular stochastic pitchfork bifurcation.

Linearly, the slow variable is $x = x_1 + \dot x_1$ and the fast variable $y = \dot x_1$.

In gory detail, a stochastic coordinate transform to simplify the form of the evolution is

$$\begin{aligned}
x &= \big[(\alpha-2\alpha^2)Y+\tfrac13\alpha Y^3\big]+\big[1+(\tfrac12-2\alpha)Y^2+\tfrac18Y^4\big]X \\
&\quad +\big[(-2+7\alpha)Y-\tfrac23Y^3\big]X^2+\tfrac92Y^2X^3-8YX^4 \\
&\quad +\sigma\Big\{\big[(1-2\alpha-2\alpha e^{+t}\star)Y+\tfrac12Y^3\big]e^{+t}\star\phi-\tfrac12Y^3e^{3t}\star\phi \\
&\qquad +\big[\alpha e^{-t}\star\phi-4Y^2e^{+t}\star\phi+4Y^2e^{2t}\star\phi\big]X \\
&\qquad +\big[e^{-t}\star\phi+(1+5e^{+t}\star)e^{+t}\star\phi\big]YX^2-2X^3e^{-t}\star\phi\Big\} \\
&\quad -2\sigma^2Ye^{+t}\star(\phi\, e^{+t}\star\phi)+\mathcal{O}\big(\varepsilon^3\big)\,, \qquad (9) \\
y &= \big[Y+\tfrac12\alpha Y^3\big]+\big[\alpha-2\alpha^2+(1-\tfrac92\alpha)Y^2+\tfrac16Y^2\big]X \\
&\quad -Y^3X^2+\big[-1+7\alpha+\tfrac{21}{2}Y^2\big]X^3-5X^5 \\
&\quad +\sigma\Big\{\big[\alpha Y+Y^3\big]e^{+t}\star\phi-Y^3e^{2t}\star\phi \\
&\qquad +\big[(1-2\alpha-2\alpha e^{-t}\star)e^{-t}\star\phi-Y^2(\tfrac12+4e^{+t}\star\phi)e^{+t}\star\phi\big]X \\
&\qquad +\big[2e^{-t}\star\phi-3e^{+t}\star\phi\big]YX^2+\big[4+3e^{-t}\star\big]e^{-t}\star\phi\, X^3\Big\} \\
&\quad -2\sigma^2Xe^{-t}\star(\phi\, e^{-t}\star\phi)+\mathcal{O}\big(\varepsilon^3\big)\,, \qquad (10)
\end{aligned}$$

where here $\varepsilon = X^2 + Y^2 + \sigma + \alpha$ measures the order of X, Y, α and σ variables and parameters. This transformation has some differences to the that of Arnold & Xu Kedai (1993) and Arnold & Imkeller (1998) because we choose the stochastic coordinate transform to additionally remove fast time convolutions as far as possible (Principle 5). The corresponding evolution in the new stochastic coordinates is

$$\begin{aligned}
\dot X &= (\alpha-\alpha^2)X-(1-3\alpha)X^3-2X^5 \\
&\quad +\sigma\big[(1-2\alpha)X+3X^3\big]\phi-\sigma^2X\phi e^{-t}\star\phi+\mathcal{O}\big(\varepsilon^3\big)\,, \qquad (11) \\
\dot Y &= -\big[(1+\alpha-\alpha^2)-(2-7\alpha)X^2-8X^4\big]Y \\
&\quad -\sigma\big[1-2\alpha+7X^2\big]Y\phi+\sigma^2Y\phi e^{+t}\star\phi+\mathcal{O}\big(\varepsilon^3\big)\,. \qquad (12)
\end{aligned}$$

Observe in the above the following properties.

- From (12), $Y=0$ is the invariant and attractive stochastic slow manifold—at least for small enough amplitude measure ε.
- In the x and y variables this SSM is

$$\begin{aligned} x &= X+\sigma\left[\alpha X-2X^3\right]e^{-t}\star\phi+\mathcal{O}\left(\varepsilon^3\right), \qquad (13)\\ y &= (\alpha-2\alpha^2)X-(1-7\alpha)X^3-5X^5\\ &\quad+\sigma\left[X(1-2\alpha-2\alpha e^{-t}\star)e^{-t}\star\phi+X^3(4+3e^{-t}\star)e^{-t}\star\phi\right]\\ &\quad-2\sigma^2Xe^{-t}\star(\phi e^{-t}\star\phi)+\mathcal{O}\left(\varepsilon^3\right). \qquad (14)\end{aligned}$$

  The shape of the SSM is not anticipative.
- The slow X evolution has no anticipatory convolutions and thus forms a sound stochastic model.
- The X evolution is independent of Y and so we can project initial conditions and restrict rationally (Cox & Roberts 1995, 1994). However, because of anticipatory convolutions in the Y evolution and the stochastic coordinate transform, the projection of initial conditions must be stochastic as it depends upon the future, unknown values of the noise.

3 Derive a normal form for a noisy Hopf bifurcation

Construct a stochastic normal form for the noisy Hopf bifurcation of the Duffing–van der Pol oscillator (1) where for a Hopf bifurcation set $\alpha=-1$ and vary the parameter β through zero; see Figure 1. The fast mode is the phase of the oscillations, whereas the slow mode is the amplitude and frequency of the oscillations.

Use the complex exponential form of solution as I think it is more flexible and/or transparent (Roberts 2006*a*, §1.3, e.g.). So far I have only tentatively encoded effects quadratic in the noise.

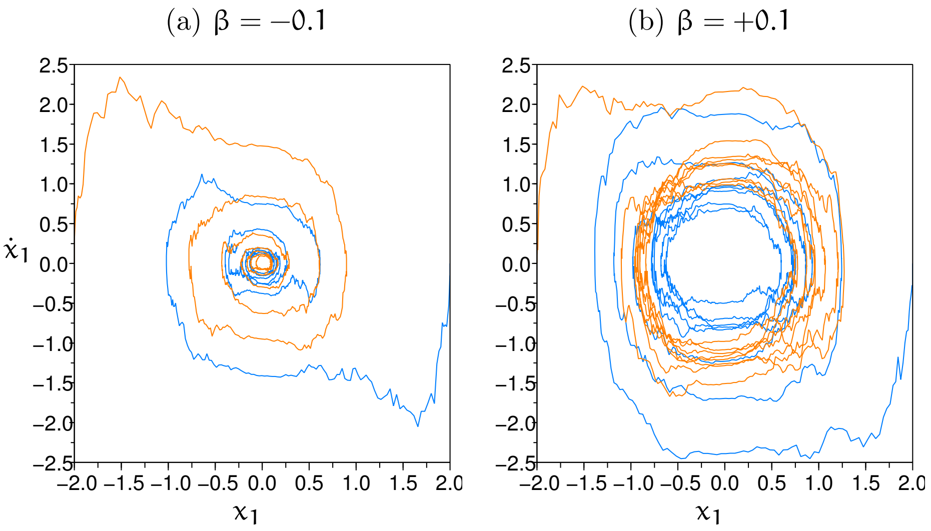

Figure 1: stochastic Hopf bifurcation in the Duffing–van der Pol oscillator (1) as parameter β crosses zero with $\alpha = -1$ and noise amplitude $\sigma = 0.5$. Two realisations are plotted in each case.

Trivial starting stuff to improve printing. The switch `gcd` helps to cancel common factors in numerator and denominator which usefully simplifies expressions.

▷▷ `hopf` ◁◁

```
% see cadnfsde.pdf for documentation
on div; off allfac; on revpri;
factor eps,del,sig;
on gcd;
```

3.1 Basic oscillation

Use a complex exponential operator `cis(q)` $= \exp(iq)$ (as REDUCE knows too much about the exponential function).

▷▷ hopf ◁◁+

```
operator cis;
let { df(cis(~v),~u)=>i*cis(v)*df(v,u)
    , cis(~u)*cis(~v)=>cis(u+v)
    , cis(~u)^2=>cis(2*u)
    , cis(0)=>1 };
```

To solve $\xi'_{tt} + \xi' + \text{Res} = 0$ define operator `linv` for any non-resonant terms in the residual. The switch `gcd` means we do not have to worry about the denominator (as yet anyway). Assumes argument of `cis` is linear in time.

▷▷ hopf ◁◁+

```
operator linv; linear linv;
let { linv(1,cis)=>-1
    , linv(cis(~a),cis)=>cis(a)/(df(a,t)^2-1)
    };
```

Let the the Duffing–van der Pol oscillator (1) evolve according to two complex amplitudes $a(t)$ and $b(t)$ where its solution $x \approx ae^{it}+be^{-it}$. These *complex amplitudes are to be slowly varying in time* to empower a time scale separation. For real solutions, b will be the complex conjugate of a.

▷▷ hopf ◁◁+

```
depend a,t;
depend b,t;
let { df(a,t)=>g, df(b,t)=>h };
```

The initial linear approximation to the oscillating dynamics is that the complex amplitudes do not evolve.

▷▷ hopf ◁◁+

```
x:=a*cis(t)+b*cis(-t);
g:=h:=0;
```

3.2 Represent the noise spectrum

Specify the forcing as the Fourier integral $\phi(t) = \int \tilde\phi(\Omega)\exp(i\Omega t)\,d\Omega$ for some Fourier transform $\tilde\phi(\Omega)$ of the noise $\phi(t)$ at some frequency Ω. Denote a general frequency Ω by `om` and the integral *implicitly* by the product `pp(om)*cis(om*t)`.

For the quadratic noise effects we need another 'integral' in another dummy variable, perhaps Ω'; denote by `ol`. In linear terms, simply ignore `pp(ol)` terms as its contributions are already catered for by `p(om)`.

However, treat the resonant frequencies separately. Note the frequencies $\Omega = \pm 4$ only enter in cubic terms at order $\epsilon^2\sigma$, and similarly for higher order resonances; I omit these for now. Consider the domain of integration to have a mesotime hole excised about each resonant frequency:

$$\int_D \cdot d\Omega \quad \text{where the domain} \quad D = \mathbb{R} \setminus \bigcup_{m=-2:2:2} (m-\delta, m+\delta)\,.$$

The operator `xint` denotes the above integral with excised holes about the resonant frequencies. The resulting normal form transform has stochastic versions of 'Cauchy' principal value integrals. Let various operators commute.

```
                    ▷▷ hopf ◁◁+
operator xint; linear xint;
let { df(xint(~a,~o),t)=>xint(df(a,t),o)
    , cis(~b)*xint(~a,~o)=>xint(a*cis(b),o)
        when (df(a,t) neq 0)
    , linv(xint(~a,~o),cis)=>xint(linv(a,cis),o)
    };
```

Then add in each resonant frequency separately into the forcing:

$$\phi(t) = \int_D \tilde\phi(\Omega)\exp(i\Omega t)\,d\Omega + \sqrt{2\delta}\sum_{m=-2:2:2} \phi_m(t)\exp(imt)\,. \qquad (15)$$

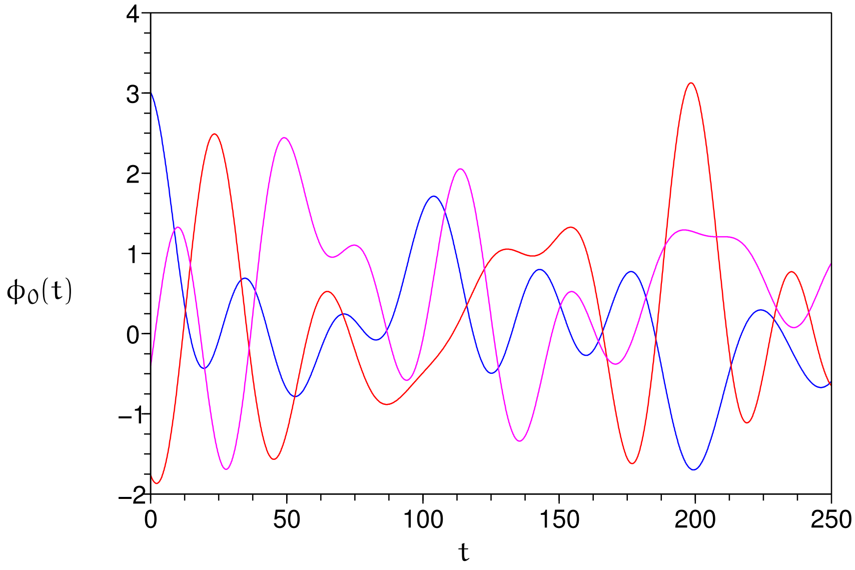

Figure 2: schematic plot of three realisations of the amplitude of resonant noise $\phi_0(t)$ for some mesoscale cutoff δ.

```
                    ▷▷ hopf ◁◁+
operator p; operator pp;
phi:=xint(pp(om)*cis(om*t),om)
    +sqrt(2)*del*(foreach m in {0,2,-2} sum p(m,0)*cis(m*t));
%   +del*(foreach m in {0,2,-2,4,-4} sum p(m,0)*cis(m*t));
```

This code also implicitly analyses the case of separately specified forcing at these resonant frequencies; in that case, one factor od `del` must be omitted.

The 'Fourier' coefficients $\phi_m(t)$ of the resonant noises are slowly varying stochastic functions: due to a notional cutoff at a mesoscopic time scale,

$$\phi_m(t) = \frac{1}{\sqrt{2\delta}} \int_{m-\delta}^{m+\delta} e^{i(\Omega-m)t} \tilde{\phi}(\Omega)\, d\Omega\,,$$

they vary slowly on the microscopic time scale; and they look like white noise on the macroscopic time scale, $\Delta t \gg 1/\delta$. Note that the windowed integrals $\sqrt{2\delta}\phi_m(t)$ have amplitudes that scale with $\sqrt{\delta} =$ del, the square-root of the window size.[1] Figure 2 plots two realisations of such $\phi_0(t)$: see smooth slow variation over fast times of $\mathcal{O}(1)$, and white noise like fluctuations over long times. Thus, p(m,0) represents *slowly varying* complex amplitudes of oscillation $\exp(imt)$, whereas p(om) represents the *time independent* Fourier transform $\tilde{\phi}(\Omega)$. To make the Fourier transform coefficients p(m,n) depend upon a slow time within the mesotime window, let the second argument n denote the number of time derivatives of the stochastic coefficient. The number of time derivatives are counted by the parameter meso $= \delta =$ del^2.

$\triangleright\triangleright$ hopf $\triangleleft\triangleleft$+

```
depend p,t;
let df(p(~m,~n),t)=>meso*p(m,n+1);
```

Quadratic noise We also generate products of integrals. Combine such products into one double integral, in say Ω and Ω' denoted by om and ol. Introduce oo to detect dependence upon either (oo may be viewed as the differential $d\Omega\, d\Omega'$ in some sense). Then combine products of integrals, and symmetrise integrands to ensure cancellation.

$\triangleright\triangleright$ hopf $\triangleleft\triangleleft$+

```
depend om,oo; depend ol,oo;
let xint(~a,om)*xint(~b,om)=>
    xint(sub(om=ol,a)*b+a*sub(om=ol,b),oo)/2;
```

Problem: the switch gcd does not seem effective for terms inside xint. So I clean residuals using procedure combin which takes expressions apart, hopefully enabling gcd to do its work on the parts, then puts the expression back together again. The code between off exp and on exp is to trace print the forcing by quadratic integrals of noise as the completion of these are handled via a fudge.

[1]See confirmation in mesoscale.sce

```
                              ▷▷ hopf ◁◁+
operator comb; linear comb;
procedure combin(res);
begin scalar a0,a1,a2;
    a0:=(comb(res,om) where {comb(1,om)=>1
        ,comb(xint(~a,om),om)=>0
        ,comb(xint(~a,oo),om)=>0});
    a1:=(comb(res,om) where {comb(1,om)=>0
        ,comb(xint(~a,om),om)=>a
        ,comb(xint(~a,oo),om)=>0});
    a2:=(comb(res,oo) where {comb(1,oo)=>0
        ,comb(xint(~a,om),oo)=>0
        ,comb(xint(~a,oo),oo)=>a});
    off exp;
    write kk:=sub({om=w+wd/2,ol=-w+wd/2},a2);
    write kk:=a2;
    write lengthkk:=length(kk);
    write kka:=df(kk,a);
    write kka0:=sub(wd=0,kka);
    write kkb:=df(kk,b);
    write kkb0:=sub(wd=0,kkb);
    on exp;
    return a0+xint(a1,om)+xint(a2,oo);
end;
```

For repeated integrals $\int_D\int_D \cdot\, d\Omega\, d\Omega'$ excise a resonant strip of width say $\sqrt{2}\delta$ about $\Omega+\Omega'=0$ as shown in Figure 3: `xint(,oo)` denotes integration over this excised domain.

The integral over the excised strip contributes to the evolution of the complex amplitudes a and b. Change variables to $\omega=\frac{1}{2}(\Omega-\Omega')$ and $\omega'=\Omega+\Omega'$ so that $\Omega=\omega+\frac{1}{2}\omega'$ and $\Omega'=-\omega+\frac{1}{2}\omega'$ and the Jacobian of the transform is one. Then components

$$\int_D\int_D e^{i(\Omega+\Omega'\pm1)t}K_\pm(\Omega,\Omega')\tilde\phi(\Omega)\tilde\phi(\Omega')\,d\Omega\,d\Omega'$$

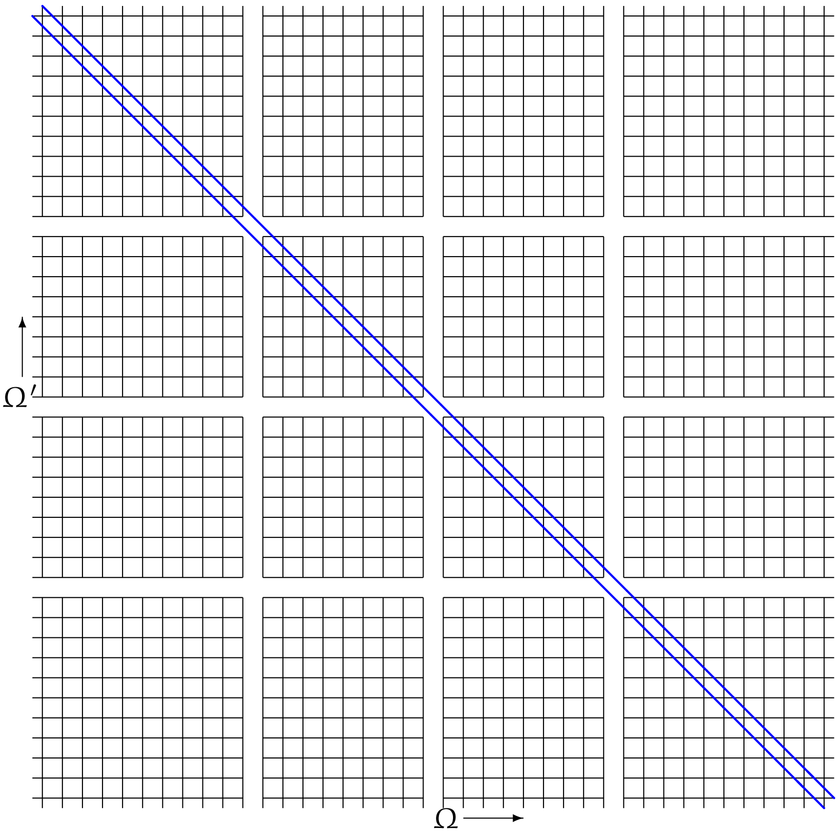

Figure 3: the integration domain $\mathrm{D}\times\mathrm{D}$, hatched, also has a further resonant region, the diagonal blue strip, excised to give the integration domain D'.

$$
\begin{aligned}
&= \iint_{D'} e^{i(\Omega+\Omega'\pm 1)t} K_\pm(\Omega,\Omega')\tilde\phi(\Omega)\tilde\phi(\Omega')\,d\Omega\,d\Omega' \\
&\quad + e^{\pm it}\int_{-\delta}^{\delta} e^{i\omega' t}\tilde\psi_\pm(\omega')\,d\omega'
\end{aligned}
$$

where $$\tilde\psi_\pm(\omega') = \int_D K_\pm(\omega+\tfrac{\omega'}{2}, -\omega+\tfrac{\omega'}{2})\tilde\phi(\omega+\tfrac{\omega'}{2})\tilde\phi(-\omega+\tfrac{\omega'}{2})\,d\omega\,,$$

and where domain $D' = D\times D$ but with the resonant strip excised as shown in Figure 3.[2] For example, in this Duffing–van der Pol oscillator (1) the e^{+it} case has kernel

$$
\begin{aligned}
K_+ &= -\frac{(\Omega+\Omega'+\Omega\Omega')(\Omega+\Omega'+2)}{2(\Omega+2)(\Omega'+2)\Omega\Omega'} \\
&= \frac{2(4\omega^2-4\omega'-\omega'^2)(2+\omega')}{(2\omega+4+\omega')(2\omega-4-\omega')(2\omega+\omega')(2\omega-\omega')} \\
&\to \frac{1}{(\omega+2)(\omega-2)} \quad \text{as } \omega'\to 0\,;
\end{aligned}
$$

similarly, the e^{-it} case has kernel

$$
\begin{aligned}
K_- &= -\frac{(\Omega+\Omega'-\Omega\Omega')(\Omega+\Omega'-2)}{2(\Omega-2)(\Omega'-2)\Omega\Omega'} \\
&= \frac{2(4\omega^2+4\omega'-\omega'^2)(2-\omega')}{(2\omega-4+\omega')(2\omega+4-\omega')(2\omega+\omega')(2\omega-\omega')} \\
&\to \frac{1}{(\omega+2)(\omega-2)} \quad \text{as } \omega'\to 0\,.
\end{aligned}
$$

Take the inverse Fourier transforms of $\tilde\psi_\pm$ and we obtain forcing terms of the form $\sigma^2 a\psi_+(t)e^{+it} + \sigma^2 b\psi_-(t)e^{-it}$ where $\psi_\pm(t)$ are slowly varying due to the low-pass cutoff at frequency δ. Numerical simulations by `cpvdd.sce` *suggest* that on long time scales $\psi_\pm(t) \approx 0.87\psi_r(t) \pm i0.20\psi_i(t)$ where ψ_r and ψ_i are independent normal white noise with mean near zero. Figure 4 plots one realisation of $\psi_\pm(t)$ showing that it is both slowly varying on the oscillation time scale, and seems effectively white on long time scales (see the power

[2] There is an error of $\mathcal{O}(\delta^2)$ in a small mismatch of the domains of integration.

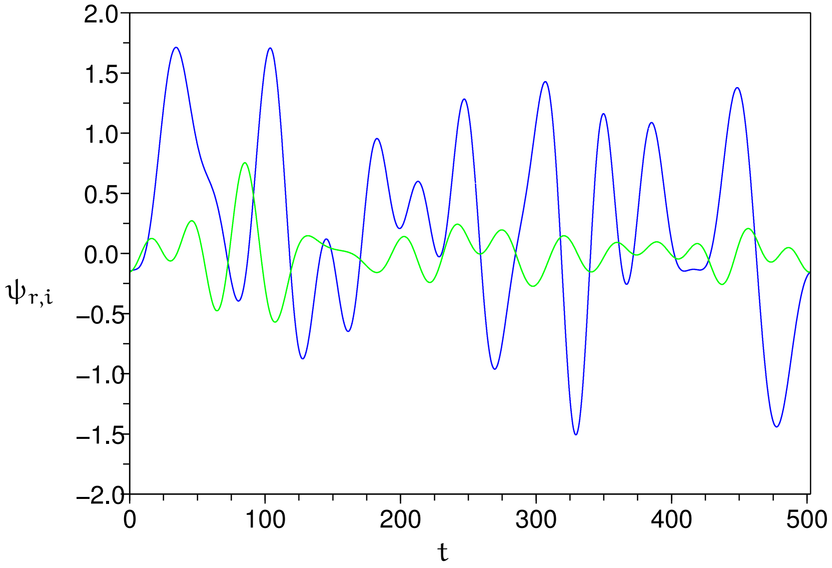

Figure 4: one realisation of the complex quadratically generated 'noise' $\psi_\pm(t) \approx 0.87\psi_r(t) \pm i0.20\psi_i(t)$ where the real part is the larger blue curve and the imaginary part is the smaller green curve. The resonant window size $\delta = 0.2$.

spectrum in Fihure 5). The magnitudes, here 0.87 and 0.20, do not *seem* to vary significantly with resonant window size δ. *Assume* the noises are independent of the other noise sources when considered over long times.

Fudge these resonant integral terms by adding a fictitious forcing to the governing Duffing–van der Pol oscillator (1). Use `cr*p(r,0)` and `ci*p(i,0)` to denote the 'new' noises $\psi_\pm(t)$; remember they are complex conjugates.

```
                    ▷▷ hopf ◁◁+
fudge:=sigma^2*(a*(cr*p(r,0)+i*ci*p(i,0))*cis(+t)
                 +b*(cr*p(r,0)-i*ci*p(i,0))*cis(-t));
```

I do not yet know how this might generalise to other quadratic integral noise effects.

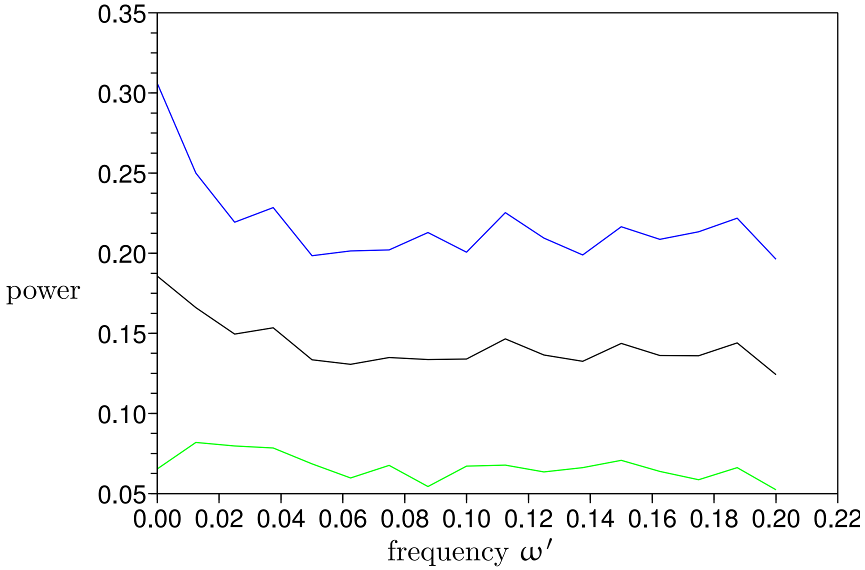

Figure 5: mean, with standard deviations, of the power spectrum of 100 realisations of the quadratic integral noises; on example is that in Figure 4. The spectrum is flat indicating white noise (except for a little rise at the lowest frequencies which may be numerical error, or may not).

3.3 Extend the inverse operator

Extend the inverse operator to handle integrating derivatives of slow noise when multiplying resonant forcing $\exp(\pm \mathrm{i}t)$. The terms need to separate into two parts: those integrable parts which go into the coordinate transform; and the non-integrable parts that go into the evolution of the complex amplitudes. The new operator `znon` extracts the non-integrable part.

First, do not change handling of $\exp(\pm \mathrm{i}t)$ terms independent of the stochastic forcing, flagged by `p` dependence.

```
                ▷▷ hopf ◁◁+
operator znon; linear znon;
let { znon(1,p)=>1, linv(1,p)=>0
```

Note: the following rules are not critical because they are not used in computing the residual. The rules may have shortcomings, but provided the iteration terminates, then the residual of the Duffing–van der Pol oscillator (1) is zero to the specified order and so we have constructed an asymptotic solution to that order. That is, the following rules should be correct enough *provided the iteration terminates.*

Linear noise Cater for factors of resonant terms, those in $\exp(\pm it)$, which are linear in the noise. The rules for `linv` are based upon ignoring the second derivative $\ddot f$ in the identity

$$\xi'_{tt} + \xi' = \pm i2\dot f e^{\pm it} + \ddot f e^{\pm it} \quad \text{when} \quad \xi' = f(t)e^{\pm it} . \tag{16}$$

Such ignorance is acceptable as `p()` varies slowly in time. Corrections due to the second derivative appear subsequently. Then for such resonant terms, the homological equation (19) becomes

$$+2i[\dot f'_+ + g']e^{it} - 2i[\dot f'_- + h']e^{-it} + \text{Res} = 0 . \tag{17}$$

Components in the residual in $\exp(\pm it)$ that can be integrated may be assigned to the coordinate transform ξ' via $f'_\pm$ whereas those components which cannot be integrated must be assigned to the evolution via g' and h'.

First, when the mesoscopic fluctuations, or constant Fourier amplitudes, are not derivatives of time, then their integral is a Wiener process, or linearly growth, both of which are unbounded and hence must be assigned to the evolution.

```
                    ▷▷ hopf ◁◁+
, znon(p(~m,0),p)=>p(m,0), linv(p(~m,0),p)=>0
```

Second, when the mesoscopic fluctuations are differentiated, just integrate and assign to the shape of the stochastic oscillation. Further iterations will chain the corrections.

```
                    ▷▷ hopf ◁◁+
, znon(p(~m,~n),p)=>0 when n>0
, linv(p(~m,~n),p)=>-p(m,n-1)/(i*2*meso) when n>0
```

The above division by small parameter `meso` leads to lower orders having additional terms through higher order effects. Its effect is to implicitly change the definition of the complex amplitudes a and b. Since I have not precisely defined the meaning of these amplitudes, then this implicit change need not be a real issue. In pitchfork bifurcations we analogously had to abandon strict control over the meaning of order parameters to avoid fast time memory integrals in the slow evolution.

Quadratic noise Factors of resonant terms which are quadratic in the noise are complicated. Most of these involve integration by parts.

First, squares of mesoscopic fluctuations, or constant Fourier amplitudes, have non-zero mean and so cannot be integrated into the shape; thus assign to the evolution.

```
                        ▷▷ hopf ◁◁+
, znon(p(~m,0)^2,p)=>p(m,0)^2
, linv(p(~m,0)^2,p)=>0
```

Second, mesoscopic products $\phi_m\dot\phi_m$ are integrated into the shape, and thus are not assigned to the evolution. Again note the division by the mesoscale parameter whenever a term is integrated into the oscillation shape.

```
                        ▷▷ hopf ◁◁+
, znon(p(~m,1)*p(~k,0),p)=>0 when m=k
, linv(p(~m,1)*p(~k,0),p)=>-p(m,0)^2/(i*4*meso) when m=k
```

Third, products which involve ϕ_k undifferentiated and ϕ_m differentiated are assigned to the evolution (unless $m = k$ and a first derivative as handled above).

```
                        ▷▷ hopf ◁◁+
, znon(p(~m,~n)*p(~k,0),p)=>p(m,n)*p(k,0)
  when (abs(m)>abs(k))or(m=k and n neq 1)or(m+k=0)
```

```
, linv(p(~m,~n)*p(~k,0),p)=>0
  when (abs(m)>abs(k))or(m=k and n neq 1)or(m+k=0)
```

Fourth, squares of mesoscale derivatives get integrated by parts, possibly contributing to the evolution and definitely contributing something to a shape change. (The commented lines are the poor alternative of assigning to evolution.) Try to move derivatives to the lowest resonant frequency in order to seek a canonical form.

```
                         ▷▷ hopf ◁◁+
, znon(p(~m,~n)^2,p)=>-znon(p(m,n+1)*p(m,n-1),p) when n>0
, linv(p(~m,~n)^2,p)=>-p(m,n)*p(m,n-1)/(i*2*meso)
  -linv(p(m,n+1)*p(m,n-1),p) when n>0
```

Fifth, integrate products to drive all products of derivatives of mesoscale noise to a cannonical form of a mesoscale noise times derivatives of a mesoscale noise.

```
                         ▷▷ hopf ◁◁+
, znon(p(~m,~n)*p(~k,~l),p)=>-znon(p(m,n+1)*p(k,l-1),p)
  when (m=k and n>l and l>0)or(abs(m)<abs(k) and l>0)
, linv(p(~m,~n)*p(~k,~l),p)=>-p(m,n)*p(k,l-1)/(i*2*meso)
  -linv(p(m,n+1)*p(k,l-1),p)
  when (m=k and n>l and l>0)or(abs(m)<abs(k) and l>0)
```

Hopefully finished with the linear and quadratic rules.

```
                         ▷▷ hopf ◁◁+
};
```

3.4 Control truncation of asymptotic solution

Use parameter `eps` as the main control of the truncation of the asymptotic solution. For now, use it to control the nonlinearity and the bifurcation parameter, and the noise (need noise controlled by eps for some as yet unappreciated reason as otherwise does not converge). Scaling the noise magnitude sigma with eps will generate some quadratic noise interactions at a low order; scaling the noise with eps^2 will restrict low order analysis to just linear noise. Use parameter meso $=$ del^2 to measure the mesoscopic window around the resonant frequencies.

```
                    ▷▷ hopf ◁◁+
    beta:=bet*eps^2;
    sigma:=sig*eps^2;
    meso:=del^2;
    let { eps^3=>0, sig^3=>0, del^4=>0 };
```

3.5 Solve by iteration

Iterate to a solution of the stochastic differential equation, to the order of error specified above, using the residual of the Duffing–van der Pol oscillator (1) to drive corrections to the asymptotic expansion.

```
                    ▷▷ hopf ◁◁+
    it:=1$
    repeat begin
    ◁◁ compute residuals ▷▷
    ◁◁ update xform ▷▷
    showtime;
    end until res=0 or (it:=it+1)>9;
    write xamp:=coeffn(x,cis(+t),1);
    write xbmp:=coeffn(x,cis(-t),1);
```

Use the residual to drive corrections to the 'coordinate' transform and the resultant normal form. Explore different oscillatory dynamics (including other Hopf bifurcations) simply by changing this residual computation (provided the linear dynamics are simply $x_{tt} + x = 0$). Trace print the length of the residual simply to indicate how close any iteration is to a solution.

```
                    ▷▷ compute residuals ◁◁
write res:=df(x,t,t)+x -(beta+sigma*phi)*df(x,t)
    +eps^2*(x^3+x^2*df(x,t)) +fudge;
write lengthres:=length(res);
```

Get gcd to do its simplification on the residual.

```
                   ▷▷ compute residuals ◁◁+
write res:=combin(res);
write lengthres:=length(res);
```

Now use the residual to update the normal form dynamics and the coordinate transform. Note at one level of approximation the homological equation is

$$\frac{\partial^2\xi'}{\partial t^2}+\xi'+\left(2ig'+\frac{\partial g'}{\partial t}\right)e^{it}+\left(-2ih'+\frac{\partial h'}{\partial t}\right)e^{-it}+\mathrm{Res}=0\,. \qquad (18)$$

This allows amplitude evolution $\dot a = g$ and $\dot b = h$ to have fast time fluctuations as done by Coullet & Spiegel (1983); however, I am not convinced they have correctly accounted in their analysis for the equivalent of the g'_t and h'_t derivatives in the above. Recall the intent here is to derive models for long term dynamics which *only* resolve slow dynamics. Thus we must carefully maintain that $\dot a = g$ and $\dot b = h$ only have 'slow' fluctuations. A benefit is that the homological equation simplifies as the terms g'_t and h'_t will be small as they are slowly varying. Consequently assume the homological equation is

$$\frac{\partial^2\xi'}{\partial t^2}+\xi'+2ig'e^{it}-2ih'e^{-it}+\mathrm{Res}=0\,. \qquad (19)$$

Updates for the evolution come from the non-integrable parts of the $\exp(\pm it)$ terms in Res, and every other term in the residual contributes to updates for the shape x of the stochastic oscillations.

```
                      ▷▷ update xform ◁◁
g:=g+i/2*znon((ca:=coeffn(res,cis(+t),1)),p);
```

```
h:=h-i/2*znon((cb:=coeffn(res,cis(-t),1)),p);
x:=x+linv(res-ca*cis(t)-cb*cis(-t),cis)
    +cis(+t)*linv(ca,p)-cis(-t)*linv(cb,p);
```

3.6 Post-processing cleans expressions

Unscale the parameters.

```
                    ▷▷ hopf ◁◁+
clear beta; clear sigma;
factor beta,sigma;
procedure unscale(x);
  begin scalar bs;
    bs:=append(
      (if eps^2 neq 0 then {bet=beta/eps^2} else {}),
      (if sig neq 0 then {sig=sigma/eps} else {}));
    return sub(bs,x);
  end;
x:=unscale(x)$
g:=unscale(g);
h:=unscale(h)$
```

See if x looks simpler like this.

```
                    ▷▷ hopf ◁◁+
factor doo,dom;
xs:=(x where {xint(~a,om)=>dom*a,xint(~a,oo)=>doo*a})$
lengthxs:=length(xs);
```

Finish via factorising the denominator of the transform—the zeros indicate where the resonances occur in the stochastic forcing. The major resonances are for frequency $\Omega = 0, \pm 2$. Higher order analysis generates resonances at $\Omega = \pm 4, \pm 6, \ldots$. This must be very much like the Mathieu equation.

```
                    ▷▷ hopf ◁◁+
write resonance0 :=factorize(den(coeffn(coeffn(xs,doo,0),dom,0
write resonanceom:=factorize(den(coeffn(xs,dom,1)));
write resonanceoo:=factorize(den(coeffn(xs,doo,1)));
```

Finished.

```
                    ▷▷ hopf ◁◁+
end;
```

References

Arnold, L. (2003), *Random Dynamical Systems*, Springer Monographs in Mathematics, Springer. 2

Arnold, L. & Imkeller, P. (1998), 'Normal forms for stochastic differential equations', *Probab. Theory Relat. Fields* **110**, 559–588. doi:10.1007/s004400050159. 2, 8, 11

Arnold, L. & Xu Kedai (1993), Simultaneous normal form and center manifold reduction for random differential equations, *in* C. Perello, C. Simo & J. Sola-Morales, eds, 'Equadiff-91', pp. 68–80. 11

Arnold, L., Sri Namachchivaya, N. & Schenk-Hoppé, K. R. (1996), 'Toward an understanding of stochastic Hopf bifurcation: a case study', *Intl. J. Bifurcation & Chaos* **6**, 1947–1975. 3

Chao, X. & Roberts, A. J. (1996), 'On the low-dimensional modelling of Stratonovich stochastic differential equations', *Physica A* **225**, 62–80. doi:10.1016/0378-4371(95)00387-8. 2

Coullet, P. H. & Spiegel, E. A. (1983), 'Amplitude equations for systems with competing instabilities', *SIAM J. Appl. Math.* **43**, 776–821. 27

Cox, S. M. & Roberts, A. J. (1994), Initialisation and the quasi-geostrophic slow manifold, Technical report, http://arXiv.org/abs/nlin.CD/0303011. 12

Cox, S. M. & Roberts, A. J. (1995), 'Initial conditions for models of dynamical systems', *Physica D* **85**, 126–141. doi:10.1016/0167-2789(94)00201-Z. 12

Murdock, J. (2003), *Normal forms and unfoldings for local dynamical systems*, Springer Monographs in Mathematics, Springer. 2

Roberts, A. J. (2006*a*), Create useful low-dimensional models of complex dynamical systems, Technical report, http://www.sci.usq.edu.au/staff/robertsa/Modelling/. 12

Roberts, A. J. (2006*b*), 'Resolving the multitude of microscale interactions accurately models stochastic partial differential equations', *LMS J. Computation and Maths* **9**, 193–221. http://www.lms.ac.uk/jcm/9/lms2005-032. 2

Roberts, A. J. (2007), Computer algebra derives normal forms of stochastic differential equations, Technical report, http://www.sci.usq.edu.au/staff/robertsa/CA/cadnfsde.pdf. 1

Sri Namachchivaya, N. & Leng, G. (1990), 'Equivalence of stochastic averaging and stochastic normal forms', *J. Appl. Mech.* **57**, 1011–1017. 3

Sri Namachchivaya, N. & Lin, Y. K. (1991), 'Method of stochastic normal forms', *Int. J. Nonlinear Mechanics* **26**, 931–943. 3