Computer algebra coordinate-free construction of slow manifold in stochastic or non-autonomous multiscale differential equations

A. J. Roberts*

31 May 2023 – July 11, 2023

Abstract

Provides a coded procedure to construct a slow manifold of a wide class of systems of non-autonomous or stochastic differential equations (herein abbreviated by S/ODEs). The method is coordinate-free. The methodology is based upon earlier research (Coullet and Spiegel 1983, Cox and Roberts 1991, Roberts 1997, Chao and Roberts 1996, Roberts 2008). Interpret all S/ODEs in the Stratonovich sense so the analysis applies to deterministic differential equations, both non-autonomous and autonomous. Cater for deterministic autonomous systems by simply omitting the 'noise'. For generality, this coded procedure caters for unstable modes, and for differential equation systems with a rational right-hand side.

Contents

*School of Mathematical Sciences, University of Adelaide, South Australia 5005, Australia. https://profajroberts.github.io/

1 Introduction

Inputs Write your S/ODE system in terms of variables $u_j(t)$. You must denote the slow modes by `u(1)` through to `x(nu)`, or the aliases `u1`,...). Each non-autonomous factor must be denoted by `w(.)` where the dot denotes almost any label you care to choose: simple numbers such as `w(1)` and/or `w(2)` are the usual choices (no aliases); but other labels may be used. Often `w(.)` is a Stratonovich white noise, a derivative of a Stratonovich Wiener process. Analyse deterministic, autonomous, systems by omitting any noise term `w()`

in the differential equations. Then, as in the example of the next Section 1.1, the input parameters to the procedure are the following:

- `dudt`, a comma separated list within `mat((...))` of the right-hand sides of the S/ODEs for the variables $u_j(t)$;
- `slowbasis`, a comma separated list within `mat(...)` of the (desired) basis for the slow subspace, corresponding to eigenvalues (near) zero;
- `fasteval`, a comma separated list within `mat((...))` of the fast eigenvalues of the system—best if real numbers (not complex), but maybe OK otherwise, especially by specifying `sign(??)=>-1` or whatever;
- `fastevec`, a comma separated list within `mat((...))` of the corresponding (generalised) eigenvectors that span the fast subspace;
- `toosmall`, an integer giving the desired order of error in the asymptotic approximation that is constructed. The procedure embeds the specified system in a family of systems parametrised by ε, and constructs a slow manifold, and evolution thereon, of the embedding system to the asymptotic error $\mathcal{O}\big(\varepsilon^{\texttt{toosmall}}\big)$ (as $\varepsilon \to 0$). Often the introduced artificial ε has a useful physical meaning, but strictly you should evaluate the output at $\varepsilon = 1$ to recover results for the specified system, and then reinterpret the results, and errors, in terms of your systems actual 'small' parameters.

  The code *also* truncates to errors $\mathcal{O}\big(\sigma^3\big)$ where σ characterises the magnitude of the non-autonomous/stochastic effects `w(j)`.

The right-hand side expressions for the time-derivatives must be multinomial in variables u_i and w_i. To cater for rational function right-hand sides, one also may include some $\dot{u}_i$ factors.

Outputs This procedure reports the embedded system it actually analyses, the number of iterations taken, the constructed time dependent coordinate transform (the original variables $\vec{u}$ as a function of the new variables $\vec{s}$), and the corresponding evolution in the new variables in terms of S/ODEs for $\vec{s}$.

- A plain report to the Terminal window in which *Reduce* is executing.
- A LaTeX source report written to the file `slowReport.tex` (with `slowReportHdr.tex` and `slowReportSys.tex`). Generate a pdf version by executing `pdflatex slowReport`.
- Global array `u_(i)` gives the parametrisation of the alow manifold $u_i(\vec{s},\sigma,\varepsilon)$.
- Global array `ff(i)` gives the corresponding evolution $\dot{U}_i =$ `ff(i)` as functions of $(\vec{s},\sigma,\varepsilon)$.

One may change the appearance of the output somewhat. For example, in order to group terms in output expressions, execute a `factor` command before executing the procedure `slownonauto(...)`.

Background The theoretical support for the results of the analysis of this procedure is centre/stable/unstable manifold theory (e.g., Carr 1981, Knobloch and Aulbach 1982, Haragus and Iooss 2011, Roberts 2015), and an embryonic backwards theory (Roberts 2022). This particular procedure is developed from that for human-efficient computer algebra (Roberts 1997), and extended to stochastic/non-autonomous systems (Chao and Roberts 1996, Roberts 2008).

We use the computer algebra package *Reduce* [`http://reduce-algebra.com/`] because it is both free and perhaps the fastest general purpose computer algebra system (Fateman 2003, e.g.).

1.1 A simple example: `examplesystem()`

Execute this example by invoking the command `examplesystem();` The classically basic non-trivial system of fast/slow S/ODEs where lowercase $w(t)$, called a *noise* within this document, often denotes the formal derivative dW/dt of a Stratonovich Wiener process $W(t,\omega)$. Alternatively, $w(t)$ represents an arbitrary deterministic time-dependent forcing, or some control, or some 'coloured' random process, or some other extrinsic input to the system. Parameter σ controls the strength of the so-called noise.

Use variables `u1,u2` to denote u_1, u_2, and use `w(j)` to denote the (Stratonovich) noise w_j. Hence this system is analysed for 'small' (u_1, u_2, σ) by executing the following defined procedure `examplesystem()`:

```
1 procedure examplesystem();
2     slownonauto(
3     mat(( -u1+u2-u1^2+w(1), u1-u2+u2^2+w(2) )),
4     mat((1,1)),
5     mat((-2)),
6     mat((-1,1)),
7     3 )$
```

The procedure `slownonauto` automatically multiplies the noise/non-autonomous factors by a parameter `sigma` so there is no need include the parameter σ in the specification of the problem—it will be done for you.

Further, the procedure uses the parameter `small`, often denoted by ε, to control truncation in nonlinearity. The fourth parameter in the above specifies to construct the normal form to errors $\mathcal{O}\big(\varepsilon^3\big)$.

Consequently, the procedure embeds the given system as the $\varepsilon = 1$ version of the following system that it actually analyses:

$$\dot{x}_1 = -\varepsilon x_1 y_1 \quad \text{and} \quad \dot{y}_1 = \sigma w_1 + \varepsilon\big(x_1^2 - 2y_1^2\big) - y_1$$

using analysis and theory based upon the subspace of equilibria $y_1 = \varepsilon = \sigma = 0$. The constructed coordinate transform and corresponding S/ODEs are the following.

This catches some errors during development

```
8 procedure testy(a,b,ind);
9     begin write "TESTY";
10    write a:=a;
11    write b:=b;
12    write indi:=ind;
13    rederr "procedure testy caught some failure";
14    end$
```

2 Header of the procedure

Need a couple of things established before defining the procedure: the `rlfi` package for a nicer version of the output via LaTeX; and operator names for the variables of the S/ODEs.

```
15 load_package rlfi;
16 operator u;
17 operator w;
```

Cater for rational function S/ODEs by allowing time dependence in these variables at specification. For rational function S/ODEs, users must multiply each S/ODE by a common denominator, and put on the right-hand side the nonlinear terms involving the time derivative.

```
18 depend u,t;
```

Now define the procedure as an operator so we can define procedures internally, and may be flexible with its arguments.

```
19 operator slownonauto;
20 for all dudt, slowbasis, fasteval, fastevec, toosmall let
21     slownonauto(dudt, slowbasis, fasteval, fastevec, toosmall)
22     = begin
```

2.1 Preamble to the procedure

Operators and arrays are always global, but we can make variables and matrices local, except for matrices that need to be declared `matrix`. So, move to implement all arrays and operators to have underscores, and almost all scalars and most matrices to be declared local here (for some reason `x/y/zrhs` must be global).

```
23 scalar maxiter, trace, nu, offdiag, jac, ok, res,
24 res0, res1, lengthresu;
```

Parameter `maxiter` is the maximum number of allowed iterations; this default may be changed.

```
25 maxiter:=9$
```

For optional trace printing of test cases: comment out second line when not needed.

```
26 trace:=0$
27 trace:=1; maxiter:=9;
```

Write an intro message.

```
28 write "Construct a non-autonomous slow manifold (version 11 Jul 2023)"$
```

Transpose the defining matrices so that vectors and lists are columns.

```
29 urhs_:= dudt;
30 urhs_ := tp urhs_;
31 if trace then write urhs_:=urhs_;
32 sbase_:=slowbasis;
33 sbase_ := tp sbase_;
34 if trace then write sbase_:=sbase_;
35 fevec_ := fastevec;
36 fevec_ := tp fevec_;
37 if trace then write fevec_:=fevec_;
38 feval_ := fasteval;
39 feval_ := tp feval_;
40 if trace then write feval_:=feval_;
```

In the printed expressions, may factor `small` (ε) and `sigma` (σ).

```
41 if trace then factor small,sigma;
```

The code cannot handle any cubic or higher order in noise amplitude `sigma`.

```
42 let sigma^3=>0;
```

The `rationalize` switch may make code much faster with complex numbers. The switch `gcd` seems to wreck convergence, so leave it off.

```
43 on div; off allfac; on revpri;
44 on rationalize;
```

2.2 Extract the differential equations

The number of differential equations is the number of rows in the matrix `dudt`.

```
write "no. of DEs ",nu:=part(length(urhs_),1);
write "no. of slow variables ",ns:=part(length(sbase_),2);
write "no. of fast variables ",nf:=part(length(fevec_),2);
if nu=ns+nf then else rederr "Abort: incompatible dimensions";
if nf=part(length(feval_),1) then
else rederr "different number of eigenvalues and eigenvectors";
if nu=part(length(sbase_),1) then
else rederr "dimension of slow basis vectors do not match the DEs";
if nu=part(length(fevec_),1) then
else rederr "dimension of fast eigenvectors do not match the DEs";
%for i:=1:nf do if numberp(feval_(i,1)) then
%    else rederr "Need all fast eigenvalues to be simple numbers";
```

Define aliases `ui` map to `u(i)`, except you *must* use the `u(i)`-form *within* any `df(,t)` on the right-hand side.

```
for i:=1:nu do set(mkid(u,i),u(i));
if trace then write "finished aliases";
```

Define some zero matrices and vectors.

```
matrix o_(nu,nu),o_s(ns,ns),o_f(nf,nf)
       ,ov_(nu,1),ov_s(ns,1),ov_f(nf,1);
```

Check on the basis matrix: putting the slow variables in the first n_s columns.

```
pp_:=o_;
for i:=1:nu do for j:=1:ns do pp_(i,j):=sbase_(i,j);
for i:=1:nu do for j:=1:nf do pp_(i,j+ns):=fevec_(i,j);
pp_inv:=1/pp_;
if trace then write "pp_inv := ",pp_inv;
```

Adjust system to suit the linear operator implicitly defined by the basis vectors and eigenvalues. Multiply all other terms, linear or nonlinear, by small.

```
ll_:=o_;
for i:=1:nf do ll_(i+ns,i+ns):=feval_(i,1);
ll_:=pp_*ll_*pp_inv;
if trace then write "ll_ :=",ll_;
u_:=ov_;
for i:=1:nu do u_(i,1):=u(i);
```

```
72 urhs_:=ll_*u_+small*(urhs_-ll_*u_);
```

Adjust the non-autonomous terms. Remove the `small` multiplication of noise terms, and instead multiply by `sigma` to empower independent control of the truncation in noise amplitude.

```
73 urhs_:=(urhs_ where w(~j)=>sigma*w(j,xyzzy)/small)$
74 urhs_:=(urhs_ where w(~j,xyzzy)=>w(j))$
```

Section 4 writes the resulting differential equations for information.

```
75 if trace then for i:=1:nu do
76     write "du(",i,")/dt = ",urhs_(i,1);
```

3 Setup LaTeX output using rlfi

Use inline math environment so that long lines, the norm, get line breaks. The command `\raggedright` in the LaTeX preamble appears the best option for the line breaking, but `\sloppy` would also work reasonably.

```
77 mathstyle math;
```

Define names for LaTeX formatting Define some names I use, so that rlfi translates them to Greek characters in the LaTeX.

```
78 defid small,name="\eps";%varepsilon;
79 defid alpha,name=alpha;
80 defid beta,name=beta;
81 defid gamma,name=gamma;
82 defid delta,name=delta;
83 defid epsilon,name=epsilon;
84 defid varepsilon,name=varepsilon;
85 defid zeta,name=zeta;
86 defid eta,name=eta;
87 defid theta,name=theta;
88 defid vartheta,name=vartheta;
89 defid iota,name=iota;
90 defid kappa,name=kappa;
91 defid lambda,name=lambda;
```

```
92 defid mu,name=mu;
93 defid nu,name=nu;
94 defid xi,name=xi;
95 defid pi,name=pi;
96 defid varpi,name=varpi;
97 defid rho,name=rho;
98 defid varrho,name=varrho;
99 defid sigma,name=sigma;
100 defid varsigma,name=varsigma;
101 defid tau,name=tau;
102 defid upsilon,name=upsilon;
103 defid phi,name=phi;
104 defid varphi,name=varphi;
105 defid chi,name=chi;
106 defid psi,name=psi;
107 defid omega,name=omega;
108 defid Gamma,name=Gamma;
109 defid Delta,name=Delta;
110 defid Theta,name=Theta;
111 defid Lambda,name=Lambda;
112 defid Xi,name=Xi;
113 defid Pi,name=Pi;
114 defid Sigma,name=Sigma;
115 defid Upsilon,name=Upsilon;
116 defid Phi,name=Phi;
117 defid Psi,name=Psi;
118 defid Omega,name=Omega;
```

For the variables names I use, as operators, define how they appear in the LaTeX, and also define that their arguments appear as subscripts.

```
119 defindex w(down);
120 defindex u(down);
121 defindex s(down);
122 defindex ff(down);
```

First use these for the specified dynamical system, later use them for the normal form equations.

```
123 defid ff,name="\dot u";
```

The 'Ornstein–Uhlenbeck' operator is to translate into a LaTeX command, see the preamble, that typesets the convolution in a reasonable manner. The definition of the LaTeX command is a bit dodgy as convolutions of convolutions are not printed in the correct order; however, convolutions commute so it does not matter.

```
124 defid ou,name="\ou";
125 defindex ou(arg,arg,arg);
```

Write the LaTeX dynamical system Because of the way rfli works, to get good quality output to the LaTeX document, I need to write the algebraic expressions to a file, then read them back in again. While being read back in, I send the output to the LaTeX file. In this convoluted way I avoid extraneous output lines polluting the LaTeX.

Temporarily use these arrays for the right-hand sides of the dynamical system.

```
126 clear ff;
127 array ff(nu);
```

Write expressions to the file `scratchfile.red` for later reading. Prepend the expressions with an instruction to write a heading, and surround the heading with anti-math mode to cancel the math environment that rlfi puts in.

```
128 out "scratchfile.red"$
129 write "off echo;"$  % do not understand why needed in 2021??
130 write "write ""\)
131 \paragraph{Specified dynamical system}
132 \(""$";
133 for i:=1:nu do write "ff(",i,"):=urhs_(",i,",1);";
134 write "end;";
135 shut "scratchfile.red";
```

Then switch on LaTeX output before writing to file as this LaTeX file is to be input from the main LaTeX file and hence does not need a header. The header here gets sent to the 'terminal' instead. Then write to `slowReportSys.tex` the expressions we stored in `scratchfile.red` as nice LaTeX.

```
136 write "Ignore the following five lines of LaTeX"$
137 on latex$
```

```
138 out "slowReportSys.tex"$
139 in "scratchfile.red"$
140 shut "slowReportSys.tex"$
141 off latex$
```

4 Delayed write of text info

Because it is messy to interleave LaTeX and plain output, I delay writing anything much in plain text until here.

Write the delayed warning message about equations.

```
142 write "
143 ===== Inspect these printed equations.  Answers
144 are rubbish if not appropriate for little small."$
```

Write the plain text versions of the dynamical system.

```
145 write "no. of variables ",nu:=nu;
146 for i:=1:nu do write "du(",i,")/dt = ",urhs_(i,1);
```

5 Represent the noise

The 'noises' `w` depend upon time. But we find it useful to discriminate upon the notionally fast time fluctuations of a noise/non-autonomous process, and the notionally ordinary time variations of the dynamic variables u_i. 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 u_i.

```
147 depend w,tt;
148 depend tt,t,ttf;
```

In the construction, convolutions of the noise arise, both backwards over history and also forwards in time to anticipate the noise (Roberts 2008, 2022).

For any non-zero parameter μ, define the 'Ornstein–Uhlenbeck' convolution

$$\mathrm{e}^{\mu t}\star\phi = \begin{cases} \int_{-\infty}^{t} \exp[\mu(t-\tau)]\phi(\tau)\,d\tau\,, & \mu<0\,, \\ \int_{t}^{+\infty} \exp[\mu(t-\tau)]\phi(\tau)\,d\tau\,, & \mu>0\,, \end{cases} \tag{1}$$

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

$$\mathrm{e}^{\mu t}\star 1 = \frac{1}{|\mu|}\,, \tag{2}$$

$$\frac{d}{dt}\mathrm{e}^{\mu t}\star\phi = -\operatorname{sgn}\mu\,\phi + \mu\mathrm{e}^{\mu t}\star\phi\,, \tag{3}$$

$$E[\mathrm{e}^{\mu t}\star\phi] = \mathrm{e}^{\mu t}\star E[\phi]\,, \tag{4}$$

$$E[(\mathrm{e}^{\mu t}\star\phi)^2] = \frac{1}{2|\mu|}\,, \tag{5}$$

$$\mathrm{e}^{\mu t}\star\mathrm{e}^{\nu t}\star = \begin{cases} \frac{1}{|\mu-\nu|}\left[\mathrm{e}^{\mu t}\star + \mathrm{e}^{\nu t}\star\right], & \mu\nu<0\,, \\ \frac{-\operatorname{sgn}\mu}{\mu-\nu}\left[\mathrm{e}^{\mu t}\star - \mathrm{e}^{\nu t}\star\right], & \mu\nu>0 \;\&\; \mu\neq\nu\,. \end{cases} \tag{6}$$

Also remember that although with $\mu<0$ the convolution $\mathrm{e}^{\mu t}\star$ integrates over the past, with $\mu>0$ the convolution $\mathrm{e}^{\mu t}\star$ integrates into the future—both over a time scale of order $1/|\mu|$.

The operator `ou(f,tt,mu)` represents the convolution $\mathrm{e}^{\mu t}\star f$ as defined by (1): called `ou` because it is an Ornstein–Uhlenbeck process when f is a stochastic white noise. The operator `ou` is 'linear' over fast time `tt` as the convolution only arises from solving PDEs in the operator $\partial_t - \mu$. Code its derivative (3) and its action upon autonomous terms (2):

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

Also code the transform (6) that successive convolutions at different rates may be transformed into several single convolutions.

```
152     , ou(ou(~r,tt,~nu),tt,~mu) =>
153       (ou(r,tt,mu)+ou(r,tt,nu))/abs(mu-nu) when (mu*nu<0)
154     , ou(ou(~r,tt,~nu),tt,~mu) =>
```

```
155       -sign(mu)*(ou(r,tt,mu)-ou(r,tt,nu))/(mu-nu)
156       when (mu*nu>0)and(mu neq nu)
157     };
```

The above properties are *critical*: they must be correct for the results to be correct. Currently, they are only coded for *real* rates μ, ν.

Second, identify the resonant parts, some of which must go into the evolution `gg(i)`, and some into the transform. It depends upon the exponent of `yz` compared to the decay rate of this mode, here `r`.

```
158 clear reso_; operator reso_; linear reso_;
159 let { reso_(~a,yz,~r)=>1 when df(a,yz)*yz=r*a
160     , reso_(~a,yz,~r)=>0 when df(a,yz)*yz neq r*a
161     };
```

Lastly, the remaining terms get convolved at the appropriate rate to solve their respective homological equation by the operator `zres_`.

```
162 depend yz,ttf;
163 clear zres_; operator zres_; linear zres_;
164 let zres_(~a,ttf,~r)=>ou(sign(df(a,yz)*yz/a-r)
165     *sub(yz=1,a),tt,df(a,yz)*yz/a-r);
```

6 Operators to solve noisy homological equation

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 knowably bounded integrable. 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, or in different scenarios. These transforms are not critical: changing the transforms may change intermediate computations, but as long as the iteration converges, the computer algebra results will be algebraically correct.

```
166 clear zint_; operator zint_; linear zint_;
167 clear znon_; operator znon_; linear znon_;
```

First, avoid obvious secularity.

```
168 zrules:= { zint_(w(~i),tt)=>0,    znon_(w(~i),tt)=>w(i)
169     , zint_(1,tt)=>0,         znon_(1,tt)=>1
170     , zint_(w(~i)*~r,tt)=>0, znon_(w(~i)*~r,tt)=>w(i)*r
```

Second, by (3) a convolution may be split into an integrable part, and a part in its argument which in turn may be integrable or not.

```
171     , zint_(ou(~r,tt,~mu),tt)
172       =>ou(r,tt,mu)/mu+zint_(r,tt)/abs(mu)
173     , znon_(ou(~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.

```
174     , zint_(ou(~r,tt,~mu)^2,tt)=>ou(~r,tt,~mu)^2/(2*mu)
175                                 +zint_(r*ou(r,tt,mu),tt)/abs(mu)
176     , znon_(ou(~r,tt,~mu)^2,tt)=>znon_(r*ou(r,tt,mu),tt)/abs(mu)
```

Fourth, different products of convolutions may be similarly separated using integration by parts. Why is the 2nd one here not acted upon, unless double tilde is used??

```
177     , zint_(ou(~r,tt,~~mu)*ou(~s,tt,~~nu),tt)
178       =>ou(r,tt,mu)*ou(s,tt,nu)/(mu+nu)
179       +zint_(sign(mu)*r*ou(s,tt,nu)+sign(nu)*s*ou(r,tt,mu),tt)
180       /(mu+nu) when mu+nu neq 0
181     , znon_(ou(~r,tt,~~mu)*ou(~s,tt,~~nu),tt)=>
182       +znon_(sign(mu)*r*ou(s,tt,nu)+sign(nu)*s*ou(r,tt,mu),tt)
183       /(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.

```
184     , zint_(ou(~r,tt,~~mu)*ou(~s,tt,~~nu),tt)=>
185       ou(ou(r,tt,-nu),tt,-nu)*ou(s,tt,nu)
186       +zint_(ou(ou(r,tt,-nu),tt,-nu)*s,tt)
```

```
187       when (mu+nu=0)and(nu>0)
188     , znon_(ou(~r,tt,~~mu)*ou(~s,tt,~~nu),tt)=>
189       znon_(ou(ou(r,tt,-nu),tt,-nu)*s,tt)
190       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.

```
191     }; % let/zrules
```

7 Initialise approximate transform

Truncate asymptotic approximation of the coordinate transform depending upon the parameter `toosmall`. Use the 'instant evaluation' property of a loop index to define the truncation so that Reduce omits small terms on the fly.

```
192 for j:=toosmall:toosmall do let small^j=>0;
```

Variables `u` are operators in the specification of the equations. We now want them to map to the approximation to the coordinate transform, so point them to arrays storing the normal form expressions. Need to clear the mapping to the array before exiting.

```
193 clear u_;
194 let u(~j)=>u_(j,1);
195 array u_(nu);
```

Express the normal form in terms of new evolving variables s_i, denoted by operators `s(i)`. The expressions for the normal form S/ODEs are stored in `ff`.

```
196 clear s;
197 operator s;
198 depend s,t;
199 let df(s(~i),t)=>ff(i);
```

The first linear approximation is then $\vec{u} \approx V_s \vec{s}$ such that $\dot{s}_i \approx 0$, in `ff(i)`.

```
ss_:=ov_s;
for j:=1:ns do ss_(j,1):=s(j);
u_:=sbase_*ss_;
if trace then write u_:=u_;
for j:=1:ns do ff(j):=0;
```

8 Iterative updates

We iterate to a solution of the governing S/ODEs to residuals of some order of error. The number of iterations are limited by a maximum.

```
write "===== Starting the constructive iteration";
res_:=-df(u_,t)+urhs_;
write lengthres := tp map(length(~q),res_);

for iter:=1:maxiter do begin
  if trace then write "
  ITERATION = ",iter,"
  -------------";
```

Compute the current residual vector.

```
   if trace then write res_:=res_;
   mode_:=pp_inv*res_;
   if trace then write mode_:=mode_;
```

8.1 Fast modes

Modify the slow manifold parametrisation in view of the fast modes.

```
   du_:=ov_;
   for j:=1:nf do begin
     dyi_:=-zres_(mode_(j+ns,1),ttf,-feval_(j,1));
     if trace then write dyi_:=dyi_;
     testing:=(dyi_ where znon_(~a,~b)=>testy(a,b,-1));
     du_(j+ns,1):=dyi_;
   end;
```

8.2 Slow modes

Update the s_i evolution `ff(i)` and parts of the $\vec{u}$ parametrisation.

```
223   for j:=1:ns do begin
224     ff(j):=ff(j)+(znon_(mode_(j,1),tt) where zrules);
225     if trace then write "ds",j,"/dt = ",ff(j);
226     testing:=(ff(j) where znon_(~a,~b)=>testy(a,b,0));
227     dxi_:= (zint_(mode_(j,1),tt) where zrules);
228     if trace then write dxi_:=dxi_;
229     du_(j,1):=dxi_;
230     testing:=(dxi_ where zint_(~a,~b)=>testy(a,b,1));
231   end;
```

Update the slow manifold parametrisation

```
232   u_:=u_+pp_*du_;
233   if trace then write u_:=u_;
```

Terminate the iteration loop once all residuals are zero, or the maximum number of iterations has been done.

```
234   res_:=-df(u_,t)+urhs_;
235   write lengthres := tp map(length(~q),res_);
236   showtime;
237   if res_=ov_ then write "Number of iterations ",
238     iter:=1000000+iter;
239 end;
```

9 Output results

Only proceed to print if terminated successfully.

```
240 if res_=ov_
241   then write "===== SUCCESS: converged to an expansion"
242   else begin
243       out "slowReport.tex"; on latex;
244       write "Failed-to-converge";
245       off latex; shut "slowReport.tex";
246       rederr "===== FAILED TO CONVERGE: I EXIT";
247   end;
```

9.1 Plain text version

Print the resultant coordinate transform: but only print to one lower power in `small` and `sigma` in order to keep output relatively small.

```
248 write "Parametrisation of time-dependent slow manifold"$
249 write "(to one order lower in both small and sigma)"$
250 if trace then write u_:=u_
251 else begin u_:=sigma*small*u_;
252      write u_:=u_/small/sigma; end;
```

Print the S/ODEs of the parametrisation of the slow manifold:

```
253 write "The slow manifold S/ODEs";
254 for i:=1:ns do write "ds(",i,")/dt = ",ff(i);
```

9.2 LaTeX version

Include order of error to make printing more robust. But we cannot use `small^toosmall` in the following as that is set to zero (for the asymptotics), so we hard code that `small` appears as `varepsilon` ε. Further, to avoid `sigma^3` being replaced by zero, introduce `sigma_` that maps to σ.

```
255 clear order_;  operator order_;
256 defid order_,name="0";
257 defindex order_(arg,arg);
258 defid sigma_,name="\sigma";
```

As before, we have to write expressions to file for later reading so they get printed without extraneous dross in the LaTeX source. First open up the temporary file `scratchfile.red` again.

```
259 out "scratchfile.red"$
260 write "off echo;"$ % do not understand why needed in 2021??
```

Write the time-dependent coordinate transform to file, with a heading, and with an anti-math environment to cancel the auto-math of rlfi. For some reason we have to keep these writes short as otherwise it generates spurious fatal blank lines in the LaTeX.

```
261 write "write ""\)
```

```
\paragraph{Time dependent slow manifold parametrisation}
\("";";
for i:=1:nu do write "u_",i,":=u_(",i,
    ",1) +order_(varepsilon^",toosmall-1+trace,",sigma_^",2+trace,");";
```

Write the resultant model differential equations to file, with a heading, and with an anti-math environment to cancel the auto-math of rlfi.

```
write "write ""\)
\paragraph{Result slow manifold DEs}
\("";";
for i:=1:ns do write "ff(",i,"):=ff(",i,
    ") +order_(varepsilon^",toosmall,",sigma_^3);";
write "end;";
```

Shut the temporary output file.

```
shut "scratchfile.red";
```

Get expressions from file and write the main LaTeX file. But first *redefine* how these names get printed, namely as the slow parameters time derivatives.

```
defindex s_(down);
defid ff,name="\dot s";
```

Penultimately, write the header information file that is to be included in the report via an `\input` in the modified `on latex`.

```
out "slowReportHdr.tex"$
write "\title{A slow manifold of your dynamical system}"$
write "\author{A. J. Roberts, University of Adelaide\\
\texttt{http://orcid.org/0000-0001-8930-1552}}
\date{\now, \today}
\def\ou\big(#1,#2,#3\big)%
    {{{\rm e}^{\if#31\else#3\fi t}\star}\left(#1\right)}
\def\eps{\varepsilon} \def\_{_}
\maketitle
Generally, the lowest order, most important,
terms are near the end of each expression.
\input{slowReportSys}
"$
shut "slowReportHdr.tex"$
```

Finally write to the main LaTeX file so switch on latex after starting to write to the file. Then write expressions in `scratchfile.red` to `slowReport.tex` as nice LaTeX. Switch off latex, to get the end of the document, and finish writing.

```
289 out "slowReport.tex"$
290 on latex$
291 in "scratchfile.red"$
292 off latex$
293 shut "slowReport.tex"$
```

10 Fin

That's all folks, so end the procedure, after clearing the mapping from operators to the stored expressions.

```
294 clear u(~j);
295 return Finished_slow_manifold_of_system$
296 end$
```

11 Override some rlfi procedures

Now setup the rlfi package to write a LaTeX version of the output. It is all a bit tricky and underhand. We override some stuff from `rlfi.red`.[1]

First, change `name` to get Big delimiters, not left-right delimiters, so LaTeX can break lines.

```
297 deflist('((!( !\!b!i!g!() (!) !\!b!i!g!)) (!P!I !\!p!i! )
298       (!p!i !\!p!i! ) (!E !e) (!I !i) (e !e) (i !i)),'name)$
```

Override the procedure that prints annoying messages about multicharacter symbols. It ends the output of one expression. This is mainly a copy from `rlfi.red` with the appropriate if-statement deleted.

```
299 symbolic procedure prinlaend;
```

[1] Find it in reduce-algebra/trunk/packages/misc/rlfi.red

```
<<terpri();
  prin2t "\)\par";
  if !*verbatim then
      <<prin2t "\begin{verbatim}";
        prin2t "REDUCE Input:">>;
  ncharspr!*:=0;
  if ofl!* then linelength(car linel!*)
    else laline!*:=cdr linel!*;
  nochar!*:=append(nochar!*,nochar1!*);
  nochar1!*:=nil >>$
```

Similarly, hardcode at the beginning of expression output that the mathematics is in inline mode.

```
symbolic procedure prinlabegin;
<<if !*verbatim then
      <<terpri();
        prin2t "\end{verbatim}">>;
  linel!*:=linelength nil . laline!*;
  if ofl!* then linelength(laline!* + 2)
    else laline!*:=car linel!* - 2;
  prin2 "\(" >>$
```

Override the procedure that outputs the LaTeX preamble upon the command `on latex`.

```
symbolic procedure latexon;
<<!*!*a2sfn:='texaeval;
  !*raise:=nil;
  prin2t "\documentclass[11pt,a5paper]{article}";
  prin2t "\usepackage[a5paper,margin=13mm]{geometry}";
  prin2t "\usepackage{parskip,time} \raggedright";
  prin2t "\begin{document}\input{slowReportHdr}";
  if !*verbatim then
      <<prin2t "\begin{verbatim}";
        prin2t "REDUCE Input:">>;
  put('tex,'rtypefn,'(lambda(x) 'tex)) >>$
```

End the file when input to Reduce

```
end;
```

References

Carr, J. (1981), *Applications of centre manifold theory*, Vol. 35 of *Applied Math. Sci.*, Springer–Verlag.
http://books.google.com.au/books?id=93BdN7btysoC

Chao, X. & Roberts, A. J. (1996), 'On the low-dimensional modelling of Stratonovich stochastic differential equations', *Physica A* **225**, 62–80.

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

Cox, S. M. & Roberts, A. J. (1991), 'Centre manifolds of forced dynamical systems', *J. Austral. Math. Soc. B* **32**, 401–436.

Fateman, R. (2003), 'Comparing the speed of programs for sparse polynomial multiplication', *ACM SIGSAM Bulletin* **37**(1), 4–15.
http://www.cs.berkeley.edu/~fateman/papers/fastmult.pdf

Haragus, M. & Iooss, G. (2011), *Local Bifurcations, Center Manifolds, and Normal Forms in Infinite-Dimensional Dynamical Systems*, Springer.

Knobloch, H.-W. & Aulbach, B. (1982), The role of center manifolds in ordinary differential equations, *in* M. Gregus, ed., 'Proceedings of the fifth Czechoslovak conference on differential equations and their applications', BSB B.G. Teubner Verlagsgesellschaft, Leipzig, pp. 179–189.
http://dml.cz/bitstream/handle/10338.dmlcz/702285/Equadiff_05-1982-1_40.pdf

Roberts, A. J. (1997), 'Low-dimensional modelling of dynamics via computer algebra', *Computer Phys. Comm.* **100**, 215–230.

Roberts, A. J. (2008), 'Normal form transforms separate slow and fast modes in stochastic dynamical systems', *Physica A* **387**, 12–38.

Roberts, A. J. (2015), *Model emergent dynamics in complex systems*, SIAM, Philadelphia.
http://bookstore.siam.org/mm20/

Roberts, A. J. (2022), Backwards theory supports modelling via invariant manifolds for non-autonomous dynamical systems, Technical report, [http://arxiv.org/abs/1804.06998].