SDE Stochastic Differential Equations

Nonlinear Filtering and Maximum Likelihood Parameter Estimation of Stochastic Differential Equations with Panel Data

■ Introduction

In 1991, LSDE (Linear Stochastic Differential Equations; Singer, 1991) was released as an SAS/IML program for the simulation, optimal filtering and maximum likelihood estimation of dynamic panel models.

This package implemented the linear continuous -discrete state space model with constant parameter matrices and uniform sampling intervals (Singer, 1991, 1993; Hamerle, Nagl and Singer, 1991).

Later, a missing data treatment was added to allow the estimation of irregularly spaced data (Singer, 1995).

Since at that time I was dissatisfied with IML, mainly because of the 2 -dimensional data structures (matrices), I ported the code to *Mathematica*.

The modules for the linear systems are similar to the older SAS/IML code. The data matrices (3-D tensors) have dimensions Y: (T+1) x p x N.

Later I developed nonlinear filters (EKF, SNF, UKF, GHF, Hermite filters, Monte Carlo filters etc.), which were firstly designed for time series data with data structure Y: (T+1) x p.

Application to a cross section of N time series leads to the data structure

Y: N x (T+1) x p.

In *Mathematica*, one can also form the list Y = {y1,y2,...,yN}, where each yn can have dimensions (Tn+1) x p. Thus, the time series of each panel unit n may have different sample size Tn+1 (cf. Singer, 2006e).

The different modules are explained in example notebooks (/SDE/applications/) and the statistical background is described in the articles (/SDE/papers/).

The modules are explained in the ■ reference section.

■ State Space Models

For details I refer to the articles in the folder /SDE/papers and the literature cited therein.
The formulae in the sequel are LaTeX pseudo code (omitting $, & etc.)

The linear continuous-discrete state space model (Jazwinski, 1970) is

dy_n(t) = A y_n(t) dt + B x_n(t) dt + G dw_n(t) (system model)

z_n(t_i) = H y_n(t_i) + D x_n(t_i) + \epsilon_{ni} (measurement model)

where \epsilon_{ni} \sim N(0,R) iid. is a discrete time white noise
and dw_n(t) /dt is a continuous time white noise process.

The system model can be written as a system of vector time series
at the times of measurement {t_0,t_1,...t_T}
(exact discrete model)

y_{n,i+1} = A*_i y_{ni} + B*_i x_{ni} + u_{ni} (system model)

z_{ni} = H_i y_{ni} + D_i x_{ni} + \epsilon_{ni} (measurement model)

with random initial condition

y_{n0} \sim N(\mu,\Sigma), i.i.d.

and the nonlinear parameter matrices

A*_i &=& \exp [A(t_{i+1}-t_i)]:=\exp (A \Deltat_i)

B*_i &=& A^{-1}(\exp (A \Deltat_i)-I) B

\Omega*_i &=& \int_{0}^{\Deltat_i} \exp (A s) G G' \exp (A's)ds.

The nonlinear continuous-discrete state space model (Jazwinski, 1970) reads

dy_n(t) = f(y_n(t),x_n(t) ,t) dt + g(y_n(t),x_n(t) ,t) dw_n(t) (system model)

z_n(t_i) = h(y_n(t_i),x_n(t_i) ,t) + \epsilon_{ni} (measurement model)

It cannot be solved explicitly, except in the linear case.

■ Filtering

The key tool for the estimation of the unknown parameters ψ in f, g, h and R
is the Kalman filter algorithm, which computes

1. recursive estimates of the states y(t)
2. the likelihood function of ψ, i.e. p(ψ; z_0,...z_T), the probability density of the
   data z as function of ψ.

In SDE, the approximate nonlinear filters

- EKF (extended Kalman filter)
- SNF (second order nonlinear filter)
- HNF (higher order nonlinear filter; HNF(K,L))
- UKF (unscented Kalman filter)
- GHF (Gauss-Hermite filter)
- GSF (Gaussian sum filters; using EKF, UKF and GHF updates)
- FIF (Functional integral filter; simulation of Ito sample paths)

are implemented.

Generalized Gauss-Hermite filters (using Hermite expansions of the filter density)
may be released later.

All filters can be used to compute the likelihood function of the data.
The maximum of p(ψ; z_0,...z_T) is computed by using a quasi Newton algorithm with
numerical score and secant updates for the model Hessian.

The modules for the linear system allow the computation of analytical score functions
and Fisher information matrices (Singer, 1993, 1995).

■ Data Structures and File Input

■ Linear Models

The older code ported from LSDE (old layer) used a data structure (tensor, nested list) with dimensions {T1,p,n}

```
Clear[y];
{T1, p, n} = {4, 3, 5};
Y = Array[y, {T1, p, n}];
Y // Dimensions
TableForm[Y, TableDirections → {Column, Column, Row},
              TableAlignments → Automatic,
              TableHeadings → None,
              TableSpacing → {6, 2, 4}]
```

```
{4, 3, 5}
```

y[1, 1, 1]	y[1, 1, 2]	y[1, 1, 3]	y[1, 1, 4]	y[1, 1, 5]
y[1, 2, 1]	y[1, 2, 2]	y[1, 2, 3]	y[1, 2, 4]	y[1, 2, 5]
y[1, 3, 1]	y[1, 3, 2]	y[1, 3, 3]	y[1, 3, 4]	y[1, 3, 5]
y[2, 1, 1]	y[2, 1, 2]	y[2, 1, 3]	y[2, 1, 4]	y[2, 1, 5]
y[2, 2, 1]	y[2, 2, 2]	y[2, 2, 3]	y[2, 2, 4]	y[2, 2, 5]
y[2, 3, 1]	y[2, 3, 2]	y[2, 3, 3]	y[2, 3, 4]	y[2, 3, 5]
y[3, 1, 1]	y[3, 1, 2]	y[3, 1, 3]	y[3, 1, 4]	y[3, 1, 5]
y[3, 2, 1]	y[3, 2, 2]	y[3, 2, 3]	y[3, 2, 4]	y[3, 2, 5]
y[3, 3, 1]	y[3, 3, 2]	y[3, 3, 3]	y[3, 3, 4]	y[3, 3, 5]
y[4, 1, 1]	y[4, 1, 2]	y[4, 1, 3]	y[4, 1, 4]	y[4, 1, 5]
y[4, 2, 1]	y[4, 2, 2]	y[4, 2, 3]	y[4, 2, 4]	y[4, 2, 5]
y[4, 3, 1]	y[4, 3, 2]	y[4, 3, 3]	y[4, 3, 4]	y[4, 3, 5]

where T1=T+1 is the number of time points t_0,...,t_T.
Thus, for each time point t, Y[[t]] is a (p x n) matrix, e.g..

```
Y[[1]] // MatrixForm
Y[[4]] // MatrixForm
```

	1	2	3	4	5
1	y[1, 1, 1]	y[1, 1, 2]	y[1, 1, 3]	y[1, 1, 4]	y[1, 1, 5]
2	y[1, 2, 1]	y[1, 2, 2]	y[1, 2, 3]	y[1, 2, 4]	y[1, 2, 5]
3	y[1, 3, 1]	y[1, 3, 2]	y[1, 3, 3]	y[1, 3, 4]	y[1, 3, 5]

	1	2	3	4	5
1	y[4, 1, 1]	y[4, 1, 2]	y[4, 1, 3]	y[4, 1, 4]	y[4, 1, 5]
2	y[4, 2, 1]	y[4, 2, 2]	y[4, 2, 3]	y[4, 2, 4]	y[4, 2, 5]
3	y[4, 3, 1]	y[4, 3, 2]	y[4, 3, 3]	y[4, 3, 4]	y[4, 3, 5]

The vector time series y2: T1 x p for unit n=2 is obtained as

```
(y2 = Y[[All, All, 2]]) // MatrixForm
 y2 // Dimensions
```

	1	2	3
1	y[1, 1, 2]	y[1, 2, 2]	y[1, 3, 2]
2	y[2, 1, 2]	y[2, 2, 2]	y[2, 3, 2]
3	y[3, 1, 2]	y[3, 2, 2]	y[3, 3, 2]
4	y[4, 1, 2]	y[4, 2, 2]	y[4, 3, 2]

```
{4, 3}
```

The **measurements** are called
Z: T1 x k x n,

the latent states are
Y: T1 x p x n

and the exgenous variables are

X: T1 x q x n.

The state space model is

dy_n(t) = A y_n(t) dt + B x_n(t) dt + G dw_n(t) (system model)

z_n(t_i) = H y_n(t_i) + D x_n(t_i) + \epsilon_{ni} (measurement model)

The masurements z_n(t_i):= z_{ni}: k x 1 are collected in the tensor Z: T1 x k x n with a dimension k (k = number of measured variables):

```
Clear[z];
{T1, k, n} = {T1, 1, n};
Z = Array[z, {T1, k, n}];
Z // Dimensions
TableForm[Z, TableDirections → {Column, Column, Row},
              TableAlignments → Left,
              TableHeadings → None,
              TableSpacing → {6, 2, 4}]
```

```
{4, 1, 5}
```

```
z[1, 1, 1]    z[1, 1, 2]    z[1, 1, 3]    z[1, 1, 4]    z[1, 1, 5]

z[2, 1, 1]    z[2, 1, 2]    z[2, 1, 3]    z[2, 1, 4]    z[2, 1, 5]

z[3, 1, 1]    z[3, 1, 2]    z[3, 1, 3]    z[3, 1, 4]    z[3, 1, 5]

z[4, 1, 1]    z[4, 1, 2]    z[4, 1, 3]    z[4, 1, 4]    z[4, 1, 5]
```

- **Nonlinear Models**

The nonlinear modules use a different data structure,
which is obtained from a collection of time series

Y = {y1,...,yN}: N x T1 x p,

where yn = {yn0,...,ynT}: T1 x p. Thus

yn=Y[[n]]: T1 x p

is a vector time series of the nth panel unit.
The nonlinear modules usually work on time series and the panel structure
is an N-fold replication of individual time series analysis
(but with cross restrictions between panel units).

The transformation of the old and new matrices is a simple transposition:

{T1, p, N} → {N, T1, p}
{ 2, 3, 1}.

Thus we use the permutation {2,3,1}, i.e. T1 goes to dim 2, p goes to dim 3 and N goes to dim 1:

```
Ynew = Transpose[Y, {2, 3, 1}]
Ynew // Dimensions
{n, T1, p}
```

```
{{{y[1, 1, 1], y[1, 2, 1], y[1, 3, 1]},
  {y[2, 1, 1], y[2, 2, 1], y[2, 3, 1]},
  {y[3, 1, 1], y[3, 2, 1], y[3, 3, 1]},
  {y[4, 1, 1], y[4, 2, 1], y[4, 3, 1]}},
 {{y[1, 1, 2], y[1, 2, 2], y[1, 3, 2]},
  {y[2, 1, 2], y[2, 2, 2], y[2, 3, 2]},
  {y[3, 1, 2], y[3, 2, 2], y[3, 3, 2]},
  {y[4, 1, 2], y[4, 2, 2], y[4, 3, 2]}},
 {{y[1, 1, 3], y[1, 2, 3], y[1, 3, 3]},
  {y[2, 1, 3], y[2, 2, 3], y[2, 3, 3]},
  {y[3, 1, 3], y[3, 2, 3], y[3, 3, 3]},
  {y[4, 1, 3], y[4, 2, 3], y[4, 3, 3]}},
 {{y[1, 1, 4], y[1, 2, 4], y[1, 3, 4]},
  {y[2, 1, 4], y[2, 2, 4], y[2, 3, 4]},
  {y[3, 1, 4], y[3, 2, 4], y[3, 3, 4]},
  {y[4, 1, 4], y[4, 2, 4], y[4, 3, 4]}},
 {{y[1, 1, 5], y[1, 2, 5], y[1, 3, 5]},
  {y[2, 1, 5], y[2, 2, 5], y[2, 3, 5]},
  {y[3, 1, 5], y[3, 2, 5], y[3, 3, 5]},
  {y[4, 1, 5], y[4, 2, 5], y[4, 3, 5]}}}
```

```
{5, 4, 3}
```

```
{5, 4, 3}
```

Indeed, the dimensions of the new data matrix are {N,T1,p}= {5,4,3}.

The reverse operation is {N,T1,p}->{T1,p,N}:

```
Yold = Transpose[Ynew, {3, 1, 2}]
Yold // Dimensions
{T1, p, n}
```

```
{{{y[1, 1, 1], y[1, 1, 2], y[1, 1, 3], y[1, 1, 4], y[1, 1, 5]},
  {y[1, 2, 1], y[1, 2, 2], y[1, 2, 3], y[1, 2, 4], y[1, 2, 5]},
  {y[1, 3, 1], y[1, 3, 2], y[1, 3, 3], y[1, 3, 4], y[1, 3, 5]}},
 {{y[2, 1, 1], y[2, 1, 2], y[2, 1, 3], y[2, 1, 4], y[2, 1, 5]},
  {y[2, 2, 1], y[2, 2, 2], y[2, 2, 3], y[2, 2, 4], y[2, 2, 5]},
  {y[2, 3, 1], y[2, 3, 2], y[2, 3, 3], y[2, 3, 4], y[2, 3, 5]}},
 {{y[3, 1, 1], y[3, 1, 2], y[3, 1, 3], y[3, 1, 4], y[3, 1, 5]},
  {y[3, 2, 1], y[3, 2, 2], y[3, 2, 3], y[3, 2, 4], y[3, 2, 5]},
  {y[3, 3, 1], y[3, 3, 2], y[3, 3, 3], y[3, 3, 4], y[3, 3, 5]}},
 {{y[4, 1, 1], y[4, 1, 2], y[4, 1, 3], y[4, 1, 4], y[4, 1, 5]},
  {y[4, 2, 1], y[4, 2, 2], y[4, 2, 3], y[4, 2, 4], y[4, 2, 5]},
  {y[4, 3, 1], y[4, 3, 2], y[4, 3, 3], y[4, 3, 4], y[4, 3, 5]}}}
```

```
{4, 3, 5}
```

```
{4, 3, 5}
```

The main advantage of the 3-dimensional list (tensor) structure is the simple picking of subtensors, e.g the data of unit nn=3 at time tt=2:

```
nn = 3; tt = 2;
Yold[[tt, All, nn]]
Ynew[[nn, tt, All]]
```

```
{y[2, 1, 3], y[2, 2, 3], y[2, 3, 3]}
```

```
{y[2, 1, 3], y[2, 2, 3], y[2, 3, 3]}
```

or the data for panel units {1,3} at time 1

```
nn = {1, 3}; tt = 1;
Yold[[tt, All, nn]] // MatrixForm
Ynew[[nn, tt, All]] // MatrixForm
```

	1	2
1	y[1, 1, 1]	y[1, 1, 3]
2	y[1, 2, 1]	y[1, 2, 3]
3	y[1, 3, 1]	y[1, 3, 3]

	1	2	3
1	y[1, 1, 1]	y[1, 2, 1]	y[1, 3, 1]
2	y[1, 1, 3]	y[1, 2, 3]	y[1, 3, 3]

- **2-Dimensional Data and File Input of Data Sets**

Usually, data are 2-dimensional data structures, where the dimensions are
layed out as rows and columns. For example, the tensor indices
{n,t,p}, n=1,...,N; t=0,...T; p=1,...P
may be distributed as N rows and T*P columns

```
Clear[y];
{T1, P, n} = {4, 3, 5}; (* write n since N is protected *)
Y1 = Array[y, {n, T1, P}];

flatten[Y1, {2, 3}] // MatrixForm
(*Map[Flatten,Y1]//MatrixForm*)(*the same*)
flatten[Y1, {2, 3}] // Dimensions
```

	1	2	3	4	5	6
1	y[1, 1, 1]	y[1, 1, 2]	y[1, 1, 3]	y[1, 2, 1]	y[1, 2, 2]	y[1, 2, 3]
2	y[2, 1, 1]	y[2, 1, 2]	y[2, 1, 3]	y[2, 2, 1]	y[2, 2, 2]	y[2, 2, 3]
3	y[3, 1, 1]	y[3, 1, 2]	y[3, 1, 3]	y[3, 2, 1]	y[3, 2, 2]	y[3, 2, 3]
4	y[4, 1, 1]	y[4, 1, 2]	y[4, 1, 3]	y[4, 2, 1]	y[4, 2, 2]	y[4, 2, 3]
5	y[5, 1, 1]	y[5, 1, 2]	y[5, 1, 3]	y[5, 2, 1]	y[5, 2, 2]	y[5, 2, 3]

```
{5, 12}
```

Thus, the panel units n=1,..,5 are the rows,
and each row n is the vector y[[n,t,p]], t=1,...,4; p= 1,...3.
Note that the index p runs faster than the index t.
Such data set may be obtained by joining the data sets
Yt: N x P for the several time points t, i.e. [Y0|Y1|...|YT].
This would be a natural structure from a cross sectional point of view,
i.e. in each row we have the variables of statistical unit n as column entries.

In *Mathematica* notation this reads

```
<< LinearAlgebra`MatrixManipulation`

AppendRows[Y1[[All, 1, All]], Y1[[All, 2, All]], Y1[[All, 3, All]], Y1[[All, 4, A
AppendRows[Y1[[All, 1, All]], Y1[[All, 2, All]], Y1[[All, 3, All]], Y1[[All, 4, A
```

	1	2	3	4	5	6
1	y[1, 1, 1]	y[1, 1, 2]	y[1, 1, 3]	y[1, 2, 1]	y[1, 2, 2]	y[1, 2, 3]
2	y[2, 1, 1]	y[2, 1, 2]	y[2, 1, 3]	y[2, 2, 1]	y[2, 2, 2]	y[2, 2, 3]
3	y[3, 1, 1]	y[3, 1, 2]	y[3, 1, 3]	y[3, 2, 1]	y[3, 2, 2]	y[3, 2, 3]
4	y[4, 1, 1]	y[4, 1, 2]	y[4, 1, 3]	y[4, 2, 1]	y[4, 2, 2]	y[4, 2, 3]
5	y[5, 1, 1]	y[5, 1, 2]	y[5, 1, 3]	y[5, 2, 1]	y[5, 2, 2]	y[5, 2, 3]

```
{5, 12}
```

Alternatively, one may have the panel data for each variable p seperately, i.e. Y1[[All,All,p]]: N x T1.
From this the data set [Y1|Y1|...|YP] may be obtained.

```
AppendRows[Y1[[All, All, 1]], Y1[[All, All, 2]], Y1[[All, All, 3]]] // MatrixFor
AppendRows[Y1[[All, All, 1]], Y1[[All, All, 2]], Y1[[All, All, 3]]] // Dimension
```

	1	2	3	4	5	6
1	y[1, 1, 1]	y[1, 2, 1]	y[1, 3, 1]	y[1, 4, 1]	y[1, 1, 2]	y[1, 2, 2]
2	y[2, 1, 1]	y[2, 2, 1]	y[2, 3, 1]	y[2, 4, 1]	y[2, 1, 2]	y[2, 2, 2]
3	y[3, 1, 1]	y[3, 2, 1]	y[3, 3, 1]	y[3, 4, 1]	y[3, 1, 2]	y[3, 2, 2]
4	y[4, 1, 1]	y[4, 2, 1]	y[4, 3, 1]	y[4, 4, 1]	y[4, 1, 2]	y[4, 2, 2]
5	y[5, 1, 1]	y[5, 2, 1]	y[5, 3, 1]	y[5, 4, 1]	y[5, 1, 2]	y[5, 2, 2]

```
{5, 12}
```

Here, the index t runs faster.

It happens very often, that N vector time series Yn: T1 x p are joined together underneath, i.e [Y1|Y2|...|YN]' : N*T1 x p

```
(Y2 = AppendColumns[Y1[[1, All, All]], Y1[[2, All, All]], Y1[[3, All, All]], Y1[[
 Y2 // Dimensions
```

	1	2	3
1	y[1, 1, 1]	y[1, 1, 2]	y[1, 1, 3]
2	y[1, 2, 1]	y[1, 2, 2]	y[1, 2, 3]
3	y[1, 3, 1]	y[1, 3, 2]	y[1, 3, 3]
4	y[1, 4, 1]	y[1, 4, 2]	y[1, 4, 3]
5	y[2, 1, 1]	y[2, 1, 2]	y[2, 1, 3]
6	y[2, 2, 1]	y[2, 2, 2]	y[2, 2, 3]
7	y[2, 3, 1]	y[2, 3, 2]	y[2, 3, 3]
8	y[2, 4, 1]	y[2, 4, 2]	y[2, 4, 3]
9	y[3, 1, 1]	y[3, 1, 2]	y[3, 1, 3]
10	y[3, 2, 1]	y[3, 2, 2]	y[3, 2, 3]
11	y[3, 3, 1]	y[3, 3, 2]	y[3, 3, 3]
12	y[3, 4, 1]	y[3, 4, 2]	y[3, 4, 3]
13	y[4, 1, 1]	y[4, 1, 2]	y[4, 1, 3]
14	y[4, 2, 1]	y[4, 2, 2]	y[4, 2, 3]
15	y[4, 3, 1]	y[4, 3, 2]	y[4, 3, 3]
16	y[4, 4, 1]	y[4, 4, 2]	y[4, 4, 3]
17	y[5, 1, 1]	y[5, 1, 2]	y[5, 1, 3]
18	y[5, 2, 1]	y[5, 2, 2]	y[5, 2, 3]
19	y[5, 3, 1]	y[5, 3, 2]	y[5, 3, 3]
20	y[5, 4, 1]	y[5, 4, 2]	y[5, 4, 3]

```
{20, 3}
```

```
Y2 // InputForm
```

```
{{y[1, 1, 1], y[1, 1, 2], y[1, 1, 3]}, {y[1, 2, 1], y[1, 2, 2], y[1, 2, 3]},
 {y[1, 3, 1], y[1, 3, 2], y[1, 3, 3]}, {y[1, 4, 1], y[1, 4, 2], y[1, 4, 3]},
 {y[2, 1, 1], y[2, 1, 2], y[2, 1, 3]}, {y[2, 2, 1], y[2, 2, 2], y[2, 2, 3]},
 {y[2, 3, 1], y[2, 3, 2], y[2, 3, 3]}, {y[2, 4, 1], y[2, 4, 2], y[2, 4, 3]},
 {y[3, 1, 1], y[3, 1, 2], y[3, 1, 3]}, {y[3, 2, 1], y[3, 2, 2], y[3, 2, 3]},
 {y[3, 3, 1], y[3, 3, 2], y[3, 3, 3]}, {y[3, 4, 1], y[3, 4, 2], y[3, 4, 3]},
 {y[4, 1, 1], y[4, 1, 2], y[4, 1, 3]}, {y[4, 2, 1], y[4, 2, 2], y[4, 2, 3]},
 {y[4, 3, 1], y[4, 3, 2], y[4, 3, 3]}, {y[4, 4, 1], y[4, 4, 2], y[4, 4, 3]},
 {y[5, 1, 1], y[5, 1, 2], y[5, 1, 3]}, {y[5, 2, 1], y[5, 2, 2], y[5, 2, 3]},
 {y[5, 3, 1], y[5, 3, 2], y[5, 3, 3]}, {y[5, 4, 1], y[5, 4, 2], y[5, 4, 3]}}
```

```
SetDirectory["/SDE/data/"]
SetOptions[TableForm, TableDirections → {Column, Row}, TableHeadings → None]
Export["dataset", Y2, "CSV"]
!! "dataset"
```

```
/SDE/data
```

```
{TableAlignments → Automatic,
 TableDepth → ∞, TableDirections → {Column, Row},
 TableHeadings → None, TableSpacing → {1, 0, 1}}
```

```
dataset
```

```
"y[1, 1, 1]","y[1, 1, 2]","y[1, 1, 3]"
"y[1, 2, 1]","y[1, 2, 2]","y[1, 2, 3]"
"y[1, 3, 1]","y[1, 3, 2]","y[1, 3, 3]"
"y[1, 4, 1]","y[1, 4, 2]","y[1, 4, 3]"
"y[2, 1, 1]","y[2, 1, 2]","y[2, 1, 3]"
"y[2, 2, 1]","y[2, 2, 2]","y[2, 2, 3]"
"y[2, 3, 1]","y[2, 3, 2]","y[2, 3, 3]"
"y[2, 4, 1]","y[2, 4, 2]","y[2, 4, 3]"
"y[3, 1, 1]","y[3, 1, 2]","y[3, 1, 3]"
"y[3, 2, 1]","y[3, 2, 2]","y[3, 2, 3]"
"y[3, 3, 1]","y[3, 3, 2]","y[3, 3, 3]"
"y[3, 4, 1]","y[3, 4, 2]","y[3, 4, 3]"
"y[4, 1, 1]","y[4, 1, 2]","y[4, 1, 3]"
"y[4, 2, 1]","y[4, 2, 2]","y[4, 2, 3]"
"y[4, 3, 1]","y[4, 3, 2]","y[4, 3, 3]"
"y[4, 4, 1]","y[4, 4, 2]","y[4, 4, 3]"
"y[5, 1, 1]","y[5, 1, 2]","y[5, 1, 3]"
"y[5, 2, 1]","y[5, 2, 2]","y[5, 2, 3]"
"y[5, 3, 1]","y[5, 3, 2]","y[5, 3, 3]"
"y[5, 4, 1]","y[5, 4, 2]","y[5, 4, 3]"
```

This data set can be input from a file using

```
(Y2in = Import["dataset", "CSV"]) // MatrixForm
Y2in // Dimensions
```

	1	2	3
1	y[1, 1, 1]	y[1, 1, 2]	y[1, 1, 3]
2	y[1, 2, 1]	y[1, 2, 2]	y[1, 2, 3]
3	y[1, 3, 1]	y[1, 3, 2]	y[1, 3, 3]
4	y[1, 4, 1]	y[1, 4, 2]	y[1, 4, 3]
5	y[2, 1, 1]	y[2, 1, 2]	y[2, 1, 3]
6	y[2, 2, 1]	y[2, 2, 2]	y[2, 2, 3]
7	y[2, 3, 1]	y[2, 3, 2]	y[2, 3, 3]
8	y[2, 4, 1]	y[2, 4, 2]	y[2, 4, 3]
9	y[3, 1, 1]	y[3, 1, 2]	y[3, 1, 3]
10	y[3, 2, 1]	y[3, 2, 2]	y[3, 2, 3]
11	y[3, 3, 1]	y[3, 3, 2]	y[3, 3, 3]
12	y[3, 4, 1]	y[3, 4, 2]	y[3, 4, 3]
13	y[4, 1, 1]	y[4, 1, 2]	y[4, 1, 3]
14	y[4, 2, 1]	y[4, 2, 2]	y[4, 2, 3]
15	y[4, 3, 1]	y[4, 3, 2]	y[4, 3, 3]
16	y[4, 4, 1]	y[4, 4, 2]	y[4, 4, 3]
17	y[5, 1, 1]	y[5, 1, 2]	y[5, 1, 3]
18	y[5, 2, 1]	y[5, 2, 2]	y[5, 2, 3]
19	y[5, 3, 1]	y[5, 3, 2]	y[5, 3, 3]
20	y[5, 4, 1]	y[5, 4, 2]	y[5, 4, 3]

```
{20, 3}
```

You can reshape the matrix to the required tensor structure by writing

```
Y3 = Shape[Y2in, {n, T1, p}]
Y3 // Dimensions
```

```
{{{y[1, 1, 1], y[1, 1, 2], y[1, 1, 3]},
  {y[1, 2, 1], y[1, 2, 2], y[1, 2, 3]},
  {y[1, 3, 1], y[1, 3, 2], y[1, 3, 3]},
  {y[1, 4, 1], y[1, 4, 2], y[1, 4, 3]}},
 {{y[2, 1, 1], y[2, 1, 2], y[2, 1, 3]},
  {y[2, 2, 1], y[2, 2, 2], y[2, 2, 3]},
  {y[2, 3, 1], y[2, 3, 2], y[2, 3, 3]},
  {y[2, 4, 1], y[2, 4, 2], y[2, 4, 3]}},
 {{y[3, 1, 1], y[3, 1, 2], y[3, 1, 3]},
  {y[3, 2, 1], y[3, 2, 2], y[3, 2, 3]},
  {y[3, 3, 1], y[3, 3, 2], y[3, 3, 3]},
  {y[3, 4, 1], y[3, 4, 2], y[3, 4, 3]}},
 {{y[4, 1, 1], y[4, 1, 2], y[4, 1, 3]},
  {y[4, 2, 1], y[4, 2, 2], y[4, 2, 3]},
  {y[4, 3, 1], y[4, 3, 2], y[4, 3, 3]},
  {y[4, 4, 1], y[4, 4, 2], y[4, 4, 3]}},
 {{y[5, 1, 1], y[5, 1, 2], y[5, 1, 3]},
  {y[5, 2, 1], y[5, 2, 2], y[5, 2, 3]},
  {y[5, 3, 1], y[5, 3, 2], y[5, 3, 3]},
  {y[5, 4, 1], y[5, 4, 2], y[5, 4, 3]}}}
```

```
{5, 4, 3}
```

Numerical data can be read as well:

```
Y4 = Shape[60 // Range // N, {n * T1, P}]
```

```
{{1., 2., 3.}, {4., 5., 6.}, {7., 8., 9.}, {10., 11., 12.},
 {13., 14., 15.}, {16., 17., 18.}, {19., 20., 21.}, {22., 23., 24.},
 {25., 26., 27.}, {28., 29., 30.}, {31., 32., 33.}, {34., 35., 36.},
 {37., 38., 39.}, {40., 41., 42.}, {43., 44., 45.}, {46., 47., 48.},
 {49., 50., 51.}, {52., 53., 54.}, {55., 56., 57.}, {58., 59., 60.}}
```

```
Y4[[1, 1]] = "miss";  (* missing data code *)
```

```
Y4
```

```
{{miss, 2., 3.}, {4., 5., 6.}, {7., 8., 9.}, {10., 11., 12.},
 {13., 14., 15.}, {16., 17., 18.}, {19., 20., 21.}, {22., 23., 24.},
 {25., 26., 27.}, {28., 29., 30.}, {31., 32., 33.}, {34., 35., 36.},
 {37., 38., 39.}, {40., 41., 42.}, {43., 44., 45.}, {46., 47., 48.},
 {49., 50., 51.}, {52., 53., 54.}, {55., 56., 57.}, {58., 59., 60.}}
```

```
SetDirectory["/SDE/data/"]
SetOptions[TableForm, TableDirections → {Column, Row}, TableHeadings → None]
Export["dataset1", Y4, "Table"]
!! "dataset1"
```

```
/SDE/data
```

```
{TableAlignments → Automatic,
 TableDepth → ∞, TableDirections → {Column, Row},
 TableHeadings → None, TableSpacing → {1, 0, 1}}
```

```
dataset1
```

```
miss   2.   3.
4.   5.   6.
7.   8.   9.
10.   11.   12.
13.   14.   15.
16.   17.   18.
19.   20.   21.
22.   23.   24.
25.   26.   27.
28.   29.   30.
31.   32.   33.
34.   35.   36.
37.   38.   39.
40.   41.   42.
43.   44.   45.
46.   47.   48.
49.   50.   51.
52.   53.   54.
55.   56.   57.
58.   59.   60.
```

```
Import["dataset1", "Table"] // MatrixForm
```

	1	2	3
1	miss	2.	3.
2	4.	5.	6.
3	7.	8.	9.
4	10.	11.	12.
5	13.	14.	15.
6	16.	17.	18.
7	19.	20.	21.
8	22.	23.	24.
9	25.	26.	27.
10	28.	29.	30.
11	31.	32.	33.
12	34.	35.	36.
13	37.	38.	39.
14	40.	41.	42.
15	43.	44.	45.
16	46.	47.	48.
17	49.	50.	51.
18	52.	53.	54.
19	55.	56.	57.
20	58.	59.	60.

■ Reference

Mathematica code is written in Courier.
The functons have 4 parameters (lists): data, parameters, system and miscellaneous.

▪ BFGS

Purpose: ML estimation of the parameter vector Theta with the BFGS algorithm,
using the BFGS secant update as model Hessian.

Definition:

```
(*************************************************************************)
BFGS[data_,
     {THETA0_,F0_},
     {system_,{Lik_,LikN_,Score_,Hesse_}},
     {misc_,KMAX_,{EPS1_,EPS2_},OPTION_,PRINT_}]
(*************************************************************************)
```

Input Parameters:

data: `{{Z1,X1},...,{ZN,XN}}.`
Zn: T1n x k is the endogenous time series of the nth panel unit.
Xn: T1n x q is the exogenous time series of the nth panel unit.
THETA0: u**.** Initial value of the parameter vector.
F0: u x u**.** Initial value of the BFGS update. Use `IdentityMatrix[u]`
or some approximation of the Fisher information matrix
(e.g. negative Hessian).
system: vector of system functions corresponding to the likelihood
function LikN. For example, using EKF,
`system={f_,Df_,g_,h_,Dh_,R_,mue_,sigma_,dt_}.`
Lik: Name of likelihood function (don`t change).

LikN: Name of individual likelihood function (EKF, UKF, etc).
Score: Name of score function (don`t change).
Hesse: Name of model Hessian (`Hesse` or `FisherInformation`)
misc: miscellanous parameters of LikN (e.g. {miss,option}:
miss = missing data code; option = 1: compute likelihood).
KMAX: maximal number of iterations.
EPS1: convergence bound `Max[Abs[S]]<EPS1`
EPS2: convergence bound `Max[Abs[STEP]]<EPS2`

Output Parameters:

Theta: Final estimate (ML estimate after convergence) after KMAX iterations.
COV: Asymptotic covariance matrix (`=Inverse[F]`).
LIKI: sequence of likelihood values.
Thetai: sequence of parameter values.
Scorei: sequence of score (gradient) values.
k: maximum number of iterations
ReturnCode:

```
If[Max[Abs[S   ]]<EPS1,Print["ReturnCode = 1"];ReturnCode = 1];
If[Max[Abs[STEP]]<EPS2,Print["ReturnCode = 2"];ReturnCode = 2];
If[Max[Abs[S   ]]<EPS1 &&
   Max[Abs[STEP]]<EPS2,Print["ReturnCode = 3"];ReturnCode = 3];
If[LIKNEU - LIKALT < 0,Print["ReturnCode = 4"];ReturnCode = 4;Break[]];
End of iteration if no improvement in step reduction.
```

Global settings:

```
FDIGITS=Round[-Log[10,$MachineEpsilon]];
TYP=1;
YMAX=1000.; (*reset value in EKF etc. to control filter divergencies*)
ANALGRAD=False;
FACTOR=1.0;
MAXSTEP=0.1; (*use other values to tune performance*)
CSEED=-999;
SCALING=1;
STEPREDUCE=4; (*use other values to tune performance*)
FISHER=False;
PRINT={1,1,1,1,1,1,1,1} (*print all*)
OVERFLOW=False;
SHIFT$=10^-8;
```

You can use a scaling vector `SCALING={s1,s2,...,su}`
if some parameters strongly deviate from 1. For example, if
`ThetaTrue={2,0.01}`, one may define `SCALING={1,0.01}` and
use `Theta=ThetaTrue/SCALING.` After convergence, you can
transform back to the original `ThetaTrue=Theta*SCALING.`

- **EKF**

Purpose: Extended Kalman filter. Approximate computation of filtered states, filter errors and likelihood.

Definition:

```
(*************************************************************)
EKF[{Z_,X_},
     Theta_,
    {f_,Df_,g_,h_,Dh_,R_,mue_,sigma_,dt_},
    {miss_,option_:1}]
(*************************************************************)
```

Input Parameters:

{Z,X}: Z: T1 x k is an endogenous time series. X: T1 x q is an exogenous time series.
THETA: u. Value of the parameter vector.
{f_,Df_,g_,h_,Dh_,R_,mue_,sigma_,dt_}: vector of system functions.
miss: missing data code
option: = 1 computation of likelihood only
= 2: computation of filtered states , filter error and likelihood : {YF,PF,lik}
= 4 computation of likelihood and innovation: {lik,inno}
= 5 computation of likelihood (sum) and t th contribution : {lik,likvec}

Output Parameters:

option: = 1 computation of likelihood only: `lik`
= 2: computation of filtered states , filter error and likelihood : `{YF,PF,lik}`
= 4 computation of likelihood and innovation: `{lik,inno}`
= 5 computation of likelihood (sum) and conditional likelihood vector: `{lik,likvec}`

- **UKF**

Purpose: Unscented Kalman filter. Approximate computation of filtered states, filter errors and likelihood.

Definition:

```
(*************************************************************)
UKF[{Z_,X_},
     Theta_,
    {f_,g_,h_,R_,mue_,sigma_,dt_,kappa_,messupdate_:MessUpUKF,
     timeupdate_:TimeUpUKF1},
    {miss_,option_:1}]:=
(*************************************************************)
```

Input Parameters:

{Z,X}: Z: T1 x k is an endogenous time series. X: T1 x q is an exogenous time series.
THETA: u. Value of the parameter vector.
{f_,g_,h_,R_,mue_,sigma_,dt_,kappa_,messupdate_:MessUpUKF,timeupdate_:TimeUpUKF1}: vec
miss: missing data code
option: = 1 computation of likelihood only
= 2: computation of filtered states , filter error and likelihood : {YF,PF,lik}
= 4 computation of likelihood and innovation: {lik,inno}
= 5 computation of likelihood (sum) and t th contribution : {lik,likvec}

Output Parameters:

option: = 1 computation of likelihood only: `lik`
= 2: computation of filtered states , filter error and likelihood : `{YF,PF,lik}`
= 4 computation of likelihood and innovation: `{lik,inno}`
= 5 computation of likelihood (sum) and conditional likelihood vector: `{lik,likvec}`

- **to be continued, see definitions in Kalman.exakt.impl.nb etc**

Bibliography

Hamerle, A. and Singer, H. (1993). Kalman Filtering and maximum likelihood estimation of stochastic differential equations using panel data, in J. Oud and R. van Blokland-Vogelesang (eds), Advances in longitudinal and multivariate analysis in the behavioral sciences, ITS, Nijmegen, pp. 3–26.

Hamerle, A., Nagl, W. and Singer, H. (1991). Problems with the estimation of stochastic differential equations using structural equations models, Journal of Mathematical Sociology 16, 3: 201– 220.

Hamerle, A., Nagl, W. and Singer, H. (1993). Identification and estimation of continuous time dynamic systems with exogenous variables using panel data, Econometric Theory 9: 283– 295.

Jazwinski, A. (1970). Stochastic Processes and Filtering Theory, Academic Press, New York.

Singer, H. (1991). LSDE - A program package for the simulation, graphical display, optimal filtering and maximum likelihood estimation of Linear Stochastic Differential Equations, User‘s guide, Meersburg.

Singer, H. (1992a). Zeitkontinuierliche Dynamische Systeme, Campus-Verlag, Frankfurt a.M./New York.

Singer, H. (1992b). Dynamic structural equations in discrete and continuous time, in G. Haag, U. Mueller and K. Troitzsch (eds), Economic Evolution and Demographic Change, Springer, Berlin, Heidelberg, New York, pp. 306– 320.

Singer, H. (1992c). Continuous time dynamical systems with discrete data, errors of measurement and individual specific effects, in F. Faulbaum (ed.), Advances in Statistical Software, G. Fischer, pp. 153– 159.

Singer, H. (1992d). The aliasing phenomenon in visual terms, Journal of Mathematical Sociology 17, 1: 39– 49.

Singer, H. (1993). Continuous-time dynamical systems with sampled data, errors of measurement and unobserved components, Journal of Time Series Analysis 14, 5: 527– 545.

Singer, H. (1995). Analytical score function for irregularly sampled continuous time stochastic processes with control variables and missing values, Econometric Theory 11: 721– 735.

Singer, H. (1996). Continuous-time dynamic models for panel data, in U. Engel and J. Reinecke (eds), Analysis of Change, de Gruyter, Berlin, New York, pp. 113– 133.

Singer, H. (1998). Continuous Panel Models with Time Dependent Parameters, Journal of Mathematical Sociology 23: 77– 98.

Singer, H. (1999). Finanzmarktökonometrie. Zeitstetige Systeme und ihre Anwendung in Ökonometrie und empirischer Kapitalmarktforschung, Physica-Verlag, Heidelberg.

Singer, H. (2002). Parameter Estimation of Nonlinear Stochastic Differential Equations: Simulated Maximum Likelihood vs. Extended Kalman Filter and Itô-Taylor Expansion, Journal of Computational and Graphical Statistics 11,4: 972– 995.

Singer, H. (2003a). Nonlinear Continuous-Discrete Filtering and ML Estimation using Kernel Density Estimates and Functional Integrals, Journal of Mathematical Sociology.

Singer, H. (2003b). Simulated Maximum Likelihood in Nonlinear Continuous-Discrete State Space Models: Importance Sampling by Approximate Smoothing, Computational Statistics 18,1: 79– 106.

Singer, H. (2004c). A Survey of Estimation Methods for Stochastic Differential Equations, in C. van Dijkum, J. Blasius, H. Kleijer and B. Van Hilten (eds), Sixth International Conference on Social Science Methodology, Netherlands Institute for the Social Sciences (SISWO), CD ROM, ISBN 90-6706-176-x.

Singer, H. (2006a). Continuous-Discrete Unscented Kalman Filtering, Diskussionsbeiträge Fachbereich Wirtschaftswissenschaft 384, FernUniversität in Hagen. http://www. fernuni-hagen.de/ FBWIWI/ forschung/ beitraege/ pdf/ db384.pdf.

Singer, H. (2006b). Generalized Gauss-Hermite Filtering for Multivariate Diffusion Processes, Diskussionsbeitr äge Fachbereich Wirtschaftswissenschaft 402, FernUniversit ät in Hagen. http://www. fernuni-hagen.de/ FBWIWI/ forschung/ beitraege/ pdf/ db402.pdf.

Singer, H. (2006c). Moment Equations and Hermite Expansion for Nonlinear Stochastic Differential Equations with Application to Stock Price Models, Computational Statistics 21,3: in print.

Singer, H. (2006d). Nonlinear Continuous Time Modeling Approaches in Panel Research, Diskussionsbeitr äge Fachbereich Wirtschaftswissenschaft 398, FernUniversit ät in Hagen. http://www. fernuni-hagen.de/ FBWIWI/ forschung/ beitraege/ pdf/ db398.pdf.

Singer, H. (2007a). Stochastic Differential Equation Models with Sampled Data, in K. van Montfort, H. Oud and A. Satorra (eds), Longitudinal Models in the Behavioral and related Sciences, The European Association of Methodology (EAM) Methodology and Statistics series, vol. II, Erlbaum publishers.

Singer, H. (2007b). Nonlinear Continuous Time Modeling Approaches in Panel Research, Statistica Neerlandica.

Singer, H. (2008a). Generalized Gauss-Hermite Filtering, Advances in Statistical Analysis 92, 2: 179– 195.

Singer, H. and Hautzinger, M. (1988). Kausalattributionen, Depressivit ät und kritische Lebensereignisse als Stochastischer Prozeß, in D. Kammer and M. Hautzinger (eds), Kognitive Depressionsforschung, Huber, Bern, Stuttgart, Toronto, pp. 42– 56.