Time series modelling with MATLAB: the SSpace toolbox

Pedregal DJ, Villegas MA, Villegas D, Trapero JR

Universidad de Castilla-La Mancha
ETSII (Ciudad Real)
PREDILAB
Diego.Pedregal@uclm.es

ITISE2018, Granada, September

Outline

Introduction

SSpace is a package that implements linear, non-linear and non-Gaussian State Space systems in a very flexible and powerful way.

It is mainly based on the work of Peter C Young and colaborators with other elements, like the books of Harvey (1989), Durbin and Koopman (2012), resembling many well-known alternative packages, but there are some unique interesting elements in it...

Download at
`https://bitbucket.org/predilab/sspace-matlab/`

To appear soon at the Journal of Statistical Software

Introduction

Statistical features:

1. The statistical framework is very general in the equations formulation and in the sense that all system matrices are multivariate and potentially time-varying.
2. Exact or diffuse initialisation of Kalman Filtering and state and disturbance smoothing.
3. Use of exact gradient in Maximum Likelihood estimation, when possible.
4. Kalman filtering and smoothing based on univariate treatment of multivariate systems.
5. Other estimation procedures, i.e. Concentrated Likelihood and minimum RMSE of several steps ahead are implemented.
6. ...

Introduction

Other advantages:

1. It is free.
2. Simple in its usage, since models are standard functions written in MATLAB.
3. A wide range of templates for specific models are available (Unobserved Components, ARIMA, Regression, Exponential Smoothing, Stochastic Volatility, etc.).
4. A full time series analysis may be done with a reduced number of coded functions with names that have been carefully selected following nemotechnic rules so that the user may remember them easily.
5. Minimizer may be chosen (fminunc).
6. We have a fully MATLAB version, we have coded core functions in C++ and finishing a R version.

Linear-Gaussian State Space system

State equation: $\alpha_{t+1} = T_t\alpha_t + \gamma_t + R_t\eta_t$

Observation equation: $y_t = Z_t\alpha_t + d_t + C_t\epsilon_t$

- η_t: vector of state noises $v \times 1$, $N(0; Q_t)$
- ϵ_t: vector of observed noises $h \times 1$, $N(0; H_t)$
- $S_t = E(\epsilon_t, \eta_t')$ covariance matrix $h \times v$
- $\gamma_t = f(\Gamma_t, u_t)$ and $d_t = g(D_t, u_t)$, with u_t of dimensions $k \times 1$
- y_t: output vector $m \times 1$
- α_t: state vector $n \times 1$
- Every term is potentially multivariate and time-varying
- Many different ways to write the same system

Non-Gaussian State Space system

State equation:

Observation equation:

$$\alpha_{t+1} = T_t\alpha_t + \gamma_t + R_t\eta_t$$
$$y_t \sim p(y_t \mid \theta_t) + d_t$$
$$\theta_t = Z_t\alpha_t$$

1. Exponential family distribution, where $p(y_t \mid \theta_t) = exp[y_t'\theta_t - b_t(\theta_t) + c_t(y_t)], -\infty < \theta_t < \infty$.
2. Stochastic Volatility models, i.e. $y_t = exp(\frac{1}{2}\theta_t)\epsilon_t + d_t$.
3. Observations generated by the relation $y_t = \theta_t + \epsilon_t, \epsilon_t \sim p(\epsilon_t)$, with $p(\bullet)$ being a distribution of the exponential family.

Non-Linear State Space system

State equation: $$\alpha_{t+1} = T_t(\alpha_t) + \gamma_t + R_t(\alpha_t)\eta_t$$

Observation equation: $$y_t = Z_t(\alpha_t) + d_t + C_t(\alpha_t)\epsilon_t$$

$$\eta_t \sim N(0, Q_t(\alpha_t)),\ \epsilon_t \sim N(0, H_t(\alpha_t))$$

The system is actually non-linear and/or state-dependent. At the moment the Extended Kalman Filter and state smoothing with exact initialisation are the implemented algorithms (Koopman and Lee, 2009).

1. Specify the model

For a local level model:

- State equation: $L_{t+1} = L_t + \eta_t$
- Observation equation: $y_t = L_t + \epsilon_t$

We have to fit that model in the general framework:

- State equation: $\alpha_{t+1} = T_t\alpha_t + \gamma_t + R_t\eta_t$
- Observation equation: $y_t = Z_t\alpha_t + d_t + C_t\epsilon_t$

$\alpha_t = L_t$; $T_t = 1$; $R_t = 1$; $Z_t = 1$; $C_t = 1$; γ_t and d_t do not exist. $S_t = 0$ and Q_t and H_t are two unknown scalar values that have to be positive

2. Translate

Translate model to a MATLAB function (e.g. using templates).

```
model= SampleSS(p)
model.T= []; model.Gam= []; model.R= []; model.Z= []; model.D= [];
model.C= []; model.Q= []; model.H= []; model.S= [];
```

For the local level model: $L_{t+1} = L_t + \eta_t$ $y_t = L_t + \epsilon_t$

```
model= example1(p)
model.T= 1; model.Gam= []; model.R= 1; model.Z= 1; model.D= [];
model.C= 1; model.Q= 10.^p(1); % This makes variance always positive
model.H= 10.^p(2); model.S= 0;
```

3. Set up the model

```
1 sys= SSmodel('y', y, 'model', @example1);
```

Using duplets it is possible to select the input and output data, the model filename, additional inputs to the function that implements the model, initial values for parameters, fix values for parameters, diffuse or exact initialisatoin of algorithms, objective cost function and analytical or exact score in Maximum Likelihood estimation.

3. Set up the model

Inputs to the system (and to SSmodel):

```
                    y: [1x100 double]
                    u: []
                model: @example1
          user_inputs: {}
                   p0: [2x1 double]
                    p: [2x1 double]
                   a1: NaN
                   P1: NaN
    OBJ_FUNCTION_NAME: {@llik}
             gradient: 1
               Nsimul: 300
```

3. Set up the model

Outputs of operations on models:

```
          table: []
      obj_value: 0
           llik: 0
           covp: []
            mat: [1x1 struct]
              a: 740.0149
              P: 0.7781
           yfit: []
              F: []
           yfor: []
           Ffor: []
              v: []
             Fv: []
            eta: []
            eps: []
           Veta: []
           Veps: []
```

4. Estimation, validation, filtering, ...

```
sys= SSestim(sys);
sys= SSvalidate(sys);
```

```
sys= SSfilter(sys); % Just for filtering estimates
sys= SSsmooth(sys); % For state smoothing estimates
sys= SSdisturb(sys); % If disturbance smoothing is required
```

Results are stored in sys output structure. It stores parametes with covariance matrix, optimal states and covariances, fitted output values and covariances, forecasted values and covariances, innovations, disturbances estimates with covariances. Further statistical diagnostics are advisable.

Example 1: VARMA

The standard VARMA template is:

```
model= SampleVARMA(p)
Sigma = ;       % Covariance matrix of perturbance
DIFFpoly= ;     % Difference operator of output(s)
ARpoly = ;      % (V)AR polynomial
MApoly = ;      % (V)MA polynomial
D= ;            % Inputs
```

```
model= SampleVARMA(p)
Sigma = 10.^p(1);                % Covariance matrix of perturbance
DIFFpoly= 1;                     % Difference operator of output(s)
ARpoly = [1 p(2) p(3)]';         % (V)AR polynomial
MApoly = [1 p(4)]';              % (V)MA polynomial
D= p(5);                         % Inputs
```

Example 1: VARMA

```
model= SampleVARMA(p, u)
Sigma = 10.^p(1);                          % Covariance matrix of perturbance
DIFFpoly= [1 -1]';                         % Difference operator of output(s)
ARpoly = conv([1 p(2) p(3)]', [1 zeros(1, 11) p(5)]'); % (V)AR
MApoly = [1 p(4) -p(4)]';                  % (V)MA polynomial
D= p(5)*u;                                 % Inputs (row vector)
```

```
...
D= filter(1, [1 -p(5)], u)';               % Inputs
```

```
...
D= p(5)./(exp(-p(6)*u));                   % Inputs
```

Example 1: VARMA

```
1  model= SampleVARMA(p, u, y)
2  ...
3  D= p(5)*u;                                  % Inputs
4  ind= find(y> 0);
5  D(1, ind)= p(6)*u(1, ind);
```

Example 2: Unobserved Components (DHR)

```
function model= SampleDHR(p, N)
m= ;
% Trend model
I= eye(m); O= zeros(m);
TT = [I I;O I]; ZT = [I O]; RT = [I O; O I];
QT = blkdiag(varmatrix(), varmatrix());

% Seasonal/cyclical DHR components
Periods = ;
Rho = ;
Qs = repmat(varmatrix(), [], []);

% Linear Regression effects
D= [];

% IRREGULAR COMPONENT (often empty if VAR component is included)
H = varmatrix();
```

Example 2: Unobserved Components (DHR)

```
function model= SampleDHR(p, N)
m= 3;
% Trend model
I= eye(m); O= zeros(m);
TT = [I I;O I]; ZT = [I O]; RT = [I O; O I];
QT = blkdiag(varmatrix(p(1:6)), varmatrix(p(7:12)));

% Seasonal/cyclical DHR components
Periods = repmat([4 2], 3, 1);
Rho = ones(3, 2);
Qs = repmat(varmatrix(p(19:24)), 1, 2);

% Linear Regression effects
D= [];

% IRREGULAR COMPONENT (often empty if VAR component is included)
H = varmatrix(p(13:18));
```

Example 2: Variance of seasonal components

```
...
Qs= [varmatrix(p(19:24)) varmatrix(p(19:24))];
Qs= [varmatrix(p(19:24)) varmatrix(p(25:30))];
Qs= [diag(10.^p(19:21)) diag(10.^p(19:21))];

% Next code produces a time varying covariance matrix with a sudden
    change in observation 100
...
Hc1= varmatrix(p(13:18));
H=   repmat(Hc1, [1 1 N]);
Hc2= varmatrix(p(25:30));
H(:, :, 100:N)= repmat(Hc2, [1 1 N-99]);
```

Example 3: Comparison with other software

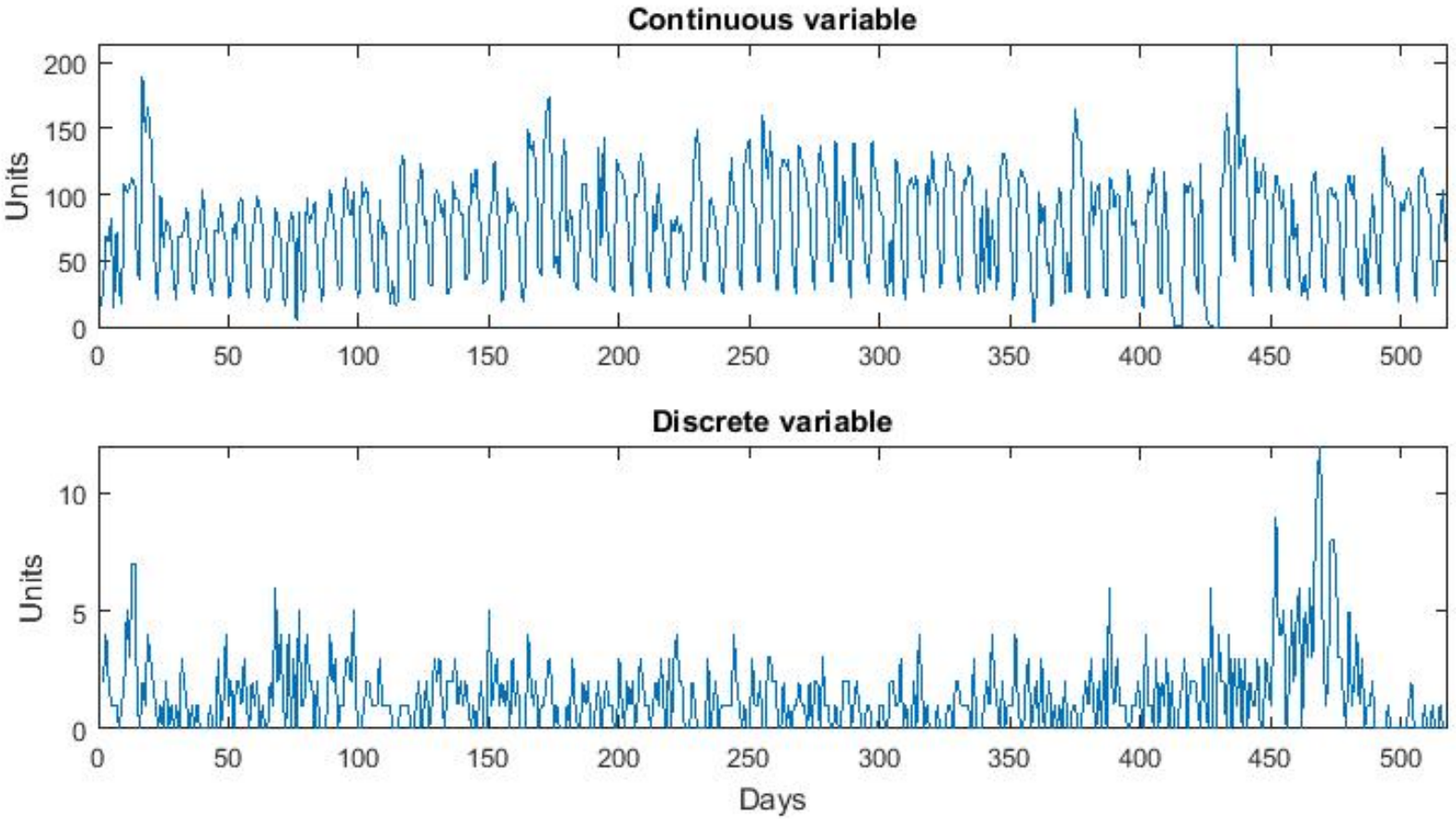
(Airline data example of Box and Jenkins)
(All variances are 1e-6)

Variances	STAMP	SSpace	Variance	SsfPack	SSpace
Trend	297.84	297.75	Trend	238	237.97
Slope	0.00	0.00	Slope	0	0.00
Seas	3.55	3.55	Seas1	11	11.11
Irregular	237.93	237.82	Seas2	5	5.26
			Seas3	0	0.00
			Seas4	2	2.30
			SEas5	1	1.23
			Seas6	0	0.00
			Irr.	327	329.78
LogLik	413.82	216.08	LogLik	223.46	223.27

Example 4: Forecast comparisons

- Daily data on sales demand of 261 products of a retailer in Spain
- Absolute error of 90 runs of 14 days ahead with daily data
- 36 % of time series are modeled as poisson processes
- Automatic identification techniques based on BIC:
  - NAIVE: random walk
  - AR up to 32th order
  - ARIMA (Hyndman & Khandakar, 2008)
  - Exponential Smoothing (ES)
  - Unobserved Components (UC)
  - Poisson Unobserved Components (UCp)
  - Mean combination of ARIMA, ES, UC and UCp
  - Median combination of ARIMA, ES, UC and UCp

Example 4: Forecast comparisons



Forecast comparisons

Table of Mean of Mean Absolute Errors for continuous time series (168 series):

	1	2	3	4	7	14
NAIVE	0.4695	1.0833	1.7437	2.4153	4.0597	8.3784
AR	0.3484	0.7198	1.0967	1.4816	2.6518	5.5116
ARIMA	0.3210	0.6634	1.0133	1.3695	2.4550	5.0598
ES	0.3308	0.6849	1.0455	1.4117	2.5241	5.2015
UC	0.3164	0.6539	0.9989	1.3500	2.4225	4.9901
Mean	0.3209	0.6635	1.0127	1.3691	2.4563	5.0740
Median	0.3216	0.6655	1.0163	1.3728	2.4590	5.0774

Example 4: Forecast comparisons

Table of Mean of Mean Absolute Errors for Discrete series (93 series):

	1	2	3	4	7	14
NAIVE	1.0128	2.0892	3.1674	4.2484	7.3912	14.9254
AR	0.8304	1.6736	2.5218	3.3707	5.9269	11.9996
ARIMA	0.8285	1.6634	2.5020	3.3425	5.8658	11.8557
ES	0.8338	1.6762	2.5220	3.3694	5.9136	11.9533
UC	0.8147	1.6331	2.4542	3.2784	5.7589	11.6495
Poisson	0.8056	1.6161	2.4284	3.2456	5.7119	11.5940
Mean	0.8089	1.6242	2.4428	3.2642	5.7368	11.6101
Median	0.8104	1.6259	2.4445	3.2660	5.7378	11.6103

Conclusion

- SSpace is a flexible and easy toolbox for full SS system analysis
- `https://bitbucket.org/predilab/sspace-matlab/`
- Documentation also available
- Eight examples in increasing order of complexity
- You may 'buy' it by pieces:
  - Just filtering, smoothing and the general template (SSfilter, SSsample)
  - Previous and additional templates
  - Part of the previous and optimization

Thank you for your attention!

e-mail: diego.pedregal@uclm.es
blog: www.uclm.es/profesorado/diego/

This work has been supported by the Spanish Ministerio de Economía y Competitividad, under Research Grant no. DPI2015-64133-R (MINECO/FEDER/UE)