Chapter 12: Kalman Filtering

The Kalman filter is a powerful tool for many econometric problems[1]. GROCER provides a relatively general program to estimate a model which can be put in a kalman filter form: this program is presented in part 1 of this chapter. Part 2 presents an application of current use: the time-varying parameter model[2]. Part 3 presents, as for every chapter, the specification of all these functions along with a few examples.

1. The Kalman filter

The Kalman filter applies when a model can be written in the following (state-space) form:

$y(t) = X(t)*B(t) + Z(t)*A + e(t)$

$B(t) = F(t) * B(t-1) + v(t)$

Where:

- y(t), X(t) are observed variables;
- B(t) is an unobserved variable;
- B(0) is assumed to be drawn from a normal law N(m,P);
- e and v are white noises, eventually correlated.

In GROCER, the following restrictions apply:

- dim(y(t)) = 1; nothing a priori prevents the Kalman filter to be applied to multivariate y(t) vectors, but the programming is then much more complicated;
- e and v are assumed non correlated of respective variances R and Q (this is not restrictive since one can always write the problem so that R and Q are uncorrelated[3]);
- F(t) is assumed not to depend on t: F(t)=F (this condition is verified for most practical problems).

Call $W(t/\tau)$ the expectation of variable W at date t knowing all information from date 0 to τ. If B(0), Q and R are known then the following quantities can be calculated:

- $B(t/t-1)$
- $\Sigma(t/t-1)=E([B(t)-B(t/t-1)]\ [B(t)-B(t/t-1)]')$
- $B(t/t)$
- $\Sigma(t/t)=E([B(t)-B(t/t)]\ [B(t)-B(t/t)]')$
- $B(t/T)$
- $\Sigma(t/T)=E([B(t)-B(t/T)]\ [B(t)-B(t/T)]')$

The first four quantities are calculated by a GROCER function called <u>filter</u> and the two last ones by a function called <u>smooth</u>.

In general however, parts of the matrices Q, R and m are unknown. They can be estimated by the maximisation of the log-likelihood: the log-likelihood is programmed in the function <u>filter_like</u>. Once the user has determined what are y, X, F, Q, R, B(1/0) and $\Sigma(1/0)$ in her problem, then she

[1] For a classical reference, see Harvey (1989), *Forecasting, Structural Time Series Models and the Kalman Filter*, Cambridge University Press.

[2] More applications are to come in the near future in GROCER.

[3] See Gourieroux et Monfort (1997), *Time Series and Dynamic Models*, Cambridge University Press.

has "only" to program a function which calculates the matrices R, Q and eventually F, B(1/0) from the vector of the parameters to estimate.

2. Time-varying parameters models

A time-varying model is a state-space model with $F=I_k$, where I_k is the (kxk) identity matrix and k is the number of parameters in X(t): this corresponds to an usual regression model, y(t)=x(t) b+ u(t) where b instead of being constant is assumed to evolve according to a random walk.

The time-varying model is programmed in GROCER in the function tvp. This function allows the user to cross two different options:

- on Q: Q can be either diagonal or arbitrary (definite positive)
- on the priors: B(1/0) and Σ(1/0); either B(1/0) is the ols result of the regression of y on X and the prior is diffuse (Σ(1/0) very large) or B(1/0) is to be estimated and Σ(1/0) is then equal to 0.

These methods correspond to the 4 following functions;

- tvp_param1: Q diagonal and diffuse prior
- tvp_param1a: Q arbitrary and diffuse prior
- tvp_param2: Q diagonal and Σ(0/0) = 0
- tvp_param2a: Q arbitrary and Σ(0/0) = 0

call to function tvp is made in one of the following way:

- tvp(endo,exo1,…,exon,'R=R0','Q=Q0','priorb0=b0','priorv0=v0','tvpmeth=1') for method 1
- tvp(endo,exo1,…,exon,'R=R0','Q=Q0','priorb0=b0','priorv0=v0','tvpmeth=1') for method 1a
- tvp(endo,exo1,…,exon,'R=R0','Q=Q0','priorb0=b0', 'tvpmeth=2') for method 2
- tvp(endo,exo1,…,exon,'R=R0','Q=Q0','priorb0=b0', 'tvpmeth=2a') for method 2a

where endo is the name of the endogenous variable; exo1, exo2,…,exon are exogenous variables; R0 (optional) is the starting value for R; Q0 (optional) is the starting value for Q; priorb0 is set to b0 (in the first two cases, this value is kept through the program; in the last two ones, it is used as a starting value for the program); priorv0 is set to v0 in the first two examples. Note that optional arguments, not given here, are set to their default values: the estimation method (which is by default maxlik) and the numerous options for this method (see chapter 5).

Note the call to function kalman in tvp (presented here for method 1):

```
grocer_ik=eye(grocer_nvar,grocer_nvar);
grocer_func='[Q,R]=tvp_param1(grocer_param)';
rtvp=kalman(grocer_func,grocer_y,grocer_x,[],grocer_ik,grocer_param,...
'priorb0=grocer_priorb0','priorv0=grocer_priorv0','meth=grocer_optmeth',varargin(:))
```

In this call, grocer_y and grocer_x are the endogenous and exogenous variables respectively; grocer_ik is F value; grocer_param is either the vector of starting values given by the user, if it is the case, or default values calculated by tvp if it is not; grocer_priorb0 and grocer_priorv0 are either the starting values given by the user, if it is the case, or default values calculated by tvp if it is not; grocer_optmeth is the optimization program chosen by the user, if any, or 'maxlik' if the user has not entered it.

Not also that the parameters entered in the various functions tvp_param*** are not directly the non zero values of R and Q, but that R and Q are calculated from these parameters so as to be definite positive: R is the square of a scalar and, when Q is not diagonal, this is Q's Choleski decomposition which is calculated from the parameters. This choice is a convenient way to impose the definite positiveness of R and Q. It implies that the Hessian calculated by the kalman program is not the Hessian corresponding to the true parameters: this is why it must be adjusted after kalman's call.

The function tvp prints on screen, by default, the most important results of estimation: the log-likelihhod; the smoothed time-varying parameters; the Q and R matrices with their estimated Student t. It also graphs the time-varying parameters. As an example, here is what you obtain if you type the following commands (taken from the example tvp_d1 available in the sublibrary macros/grocer/kalman):

```
-->load('macros/grocer/bdexamples/cousa.dat')
// load the database cousa that contains the consumption and income series
-->rols=ols('con','inc','cte','noprint);
// estimate the model by ols to have a priori on b starting values and on R
-->b=rols('beta'):
-->sigu=sqrt(rols('sigu'))
--> rols('llike') // for the sake of comparison, provides the log-likehood of
the model with constant b

- 269.88499

-->r1=tvp('con','inc','cte','R=sigu','Q=0.1*eye(2,2)','priorb0=b',…
'priorv0=100000*eye(2,2)','tvpmeth=1');
// estimate the tvp model by

   CONVERGENCE CRITERIA MET: Change in Objective Function



tvp estimation results for dependent variable: con
estimation period: 1959a-1997a
number of observations: 39
number of variables: 2
log-likelihood: -255.88494

// log-likelihodd is indeed greater than that of the constant b model

variances of the observation equation: 2271.1105
(t-stat)                              : ( .0002461)

variances of the state equation:
variable variance  t-statistic p value
inc       .0000670 186.13496   0
cte       .0689863  .0061145    .9951214


variable coefficients
      inc       cte
1959a  .9871978 -676.1333
1960a  .9901299 -676.94773
1961a  .9800304 -660.50849
1962a  .9755539 -647.27125
1963a  .9772215 -651.59617
```

1964a	.9588415	-576.48952
1965a	.9520557	-549.37872
1966a	.9524031	-550.6153
1967a	.9373195	-505.64189
1968a	.9440639	-525.44132
1969a	.9482907	-534.77368
1970a	.9341449	-502.57422
1971a	.9279467	-488.4829
1972a	.9383706	-515.81446
1973a	.9151028	-429.31889
1974a	.9135153	-430.30979
1975a	.9167066	-431.8646
1976a	.9308574	-459.73302
1977a	.9407949	-476.79081
1978a	.9328711	-456.22816
1979a	.9280499	-449.57611
1980a	.9203613	-451.32067
1981a	.9099095	-442.35922
1982a	.9118304	-442.29108
1983a	.9322518	-464.93536
1984a	.9110876	-392.68122
1985a	.9241812	-412.79618
1986a	.9338550	-425.59896
1987a	.9442631	-433.21512
1988a	.9429989	-430.96375
1989a	.9453741	-432.78416
1990a	.9445371	-432.37484
1991a	.9406656	-435.15464
1992a	.9404473	-434.96914
1993a	.9537321	-438.05756
1994a	.9633265	-445.13974
1995a	.9626863	-444.46921
1996a	.9656700	-447.70861
1997a	.9709748	-453.45345

```
                    *
                *     *
```

```
// only the graph for the time-varying coefficient for inc has been copied in
the text, but the program provides also the graph for the time-varying
coefficient for cte
```

The results tlist (r1 here) contains more information that can be retrieved (see the precise specification in part 3 of this chapter). For instance, if you are interested in the filtered betas instead of the smoothed ones, then you can type:

```
-->r1('betat')
```

There is another demo program called tvp_d2: this program uses artificial data, built from random draws, that allow checking the proximity between the estimated parameters and the true ones.

3. 2. The kalman functions and their specifications.

filter_____Kalman filter

CALLING SEQUENCE

[betat,betaf,sigmatt,sigmatf]=filter(param,func,y,x,F,z)

PARAMETERS

INPUT:
* param = a vector of parameters (sqrt of variances)
* func = the function which transforms the parameters into the matrix of variances (Q and R) and, if necessary, into the matrices of prior values
* y = (nx1) data vector

* x = (nxk) data matrix
* F = transition matrix
* z = (nxl) data matrix

OUTPUT:
* betat = vector of filtered beta at date t with the information available at date t (beta(t|t))
* betaf = vector of filtered beta at date t with the information available at date t-1 (beta(t|t-1)
* ferror = vector of errors at date t (=y-x*beta(t|t))
* sigmatt = vector of filtered variances at date t with the information available at date t (sigma(t|t))
* sigmatf = vector of filtered variances at date t with the information available at date t-1 (sigma(t|t-1))

DESCRIPTION

Generates model filtered betas and variance, given values for Q and R in a Kalman model:

y(t) = X(t)*B(t) + Z(t)*A + e(t), e(t) = N(0,R)
B(t) = Z(t) * B(t-1) + v(t), v(t) = N(0,Q)

Example:

Filter is a used by the function kalman:

[betat,betaf,sigmatt,sigmatf]=filter(grocer_param,grocer_func,grocer_y,grocer_x,grocer_F)

filter_like_____Kalman filter log-likelihood

CALLING SEQUENCE

[llik]=filter_like(param,func,y,x,F,z)

PARAMETERS

INPUT:
* param = a vector of parameters (sqrt of variances)
* func = the function which transforms the parameters into the matrix of variances (Q and R)
* y = (nx1) data vector
* x = (nxk) data matrix
* F = transition matrix
* z = (nxl) data matrix

OUTPUT:
* llik = - 2*log-likehood (+n*log(2*%pi))

DESCRIPTION

generate model loglikelihood in a kalman model:

y(t) = X(t)*B(t) + Z(t)*A + e(t), e(t) = N(0,R)
B(t) = Z(t) * B(t-1) + v(t), v(t) = N(0,Q)

Example:

Filter_like is used by the function kalman:

oresult = maxlik(filter_like,grocer_param,grocer_func,grocer_y,grocer_x,grocer_F, varargin(:))

kalman_____Kalman filter estimation

CALLING SEQUENCE

[rkalman]=kalman(func,y,x,z,F,param,varargin)

PARAMETERS

INPUT:
* func = the function which transforms the parameters into the matrix of variances (Q and R)
* y = (nobs x 1) dependent variable vector
* x = (nobs x 1) explanatory variable matrix
* z = (nxl) data matrix of exogenous variables (or [] if there are no exogenous variables in the model) * F = the transfer matrix
* param = a vector of parameters (sqrt of variances)
* varargin = optional arguments which can be:
  - 'priorb0=x' where x is (k x 1) vector with prior b0 values (default = zeros(k,1), diffuse)
  - 'priorv0=x' where x = (k x k) matrix with prior variance for Q (default = eye(k)*1e+5, a diffuse prior)
  - 'meth=x' where x is either 'maxlik' (default) or 'optim' according to the optimization program used
  - any option to maxlik (see maxlik() for a list)

OUTPUT:
rkalman = a results tlist with
  - rkalman('meth') = 'kalman'
  - rkalman('Q') = estimated Q
  - rkalman('R') = estimated R
  - rkalman('priorb0') = B(0/0)
  - rkalman('priorv0') = sigma(0/0)
  - rkalman('betat') = B(t/t)
  - rkalman('betaf') = B(t/t-1)
  - rkalman('betas') = B(t/T)
  - rkalman('sigmatt') = sigma(t/t)
  - rkalman('sigmatf') = sigma(t/t-1)
  - rkalman('sigmats') = sigma(t/T)
  - rkalman('param') = estimated parameters
  - rkalman('vcov') = variance-covariance matrix of estimated parameters
  - rkalman('tstat') = Student's t of estimated parameters
  - rkalman('y') = y
  - rkalman('x') = x
  - rkalman('yhat') = X(t)*B(t)
  - rkalman('resid') = y-X*B(t)
  - rkalman('like') = log-likelihood
  - rkalman('nobs') = # of observations
  - rkalman('nvar') = # of exogenous variables

DESCRIPTION
Maximum likelihood estimation of a kalman model:

y(t) = X(t)*B(t) + Z(t)*A + e(t), e(t) = N(0,R)
B(t) = Z(t) * B(t-1) + v(t), v(t) = N(0,Q)

Example:
rtvp=kalman(grocer_func,grocer_y,grocer_x,grocer_ik,grocer_param,'priorb0=grocer_priorb0','priorv0=grocer_priorv0','meth=grocer_optmeth',varargin(:))

This example is taken from tvp. Here grocer_func is one of the 4 following cases: 1) '[Q,R]=tvp_param1(grocer_param)' 2) '[Q,R]=tvp_param1a(grocer_param)' 3) '[Q,R,grocer_priorb0]=tvp_param2(param)' 4) '[Q,R,grocer_priorb0]=tvp_param2a(param)'; grocer_y is the vector of endogenous variables, grocer_x the matrix of exogenous variables; grocer_ik is the (kxk) identity matrix; grocer_param is a vector of starting values, either given b the user or calculated as the ols regression of y on x; grocer_priorb0, grocer_priorv0 and grocer_optmeth are either the values given by the user or the default values; lastly, varargin are the option for maxlik, if any, given by he user.

smoothing_____Kalman smoother

CALLING SEQUENCE

[betas,sigmas]=smoothing(F,betat,betaf,sigmat,sigmaf, begsmooth)

PARAMETERS

INPUT:
* F = transition matrix
* betat = vector of filtered beta at date t with the information available at date t (beta(t|t))
* betaf = vector of filtered beta at date t with the information available at date t-1 (beta(t|t-1)
* sigmat = vector of filtered variances at date t with the information available at date t (sigma(t|t))
* sigmat = vector of filtered variances at date t with the information available at date t-1 (sigma(t|t-1))
* begsmooth = first observation where to calculate smoothed values
---
OUTPUT:
* betas = vector of smoothed beta at date t with the information available at date T (beta(t|T)
* sigmat = vector of smoothed variances at date t with the information available at date T (sigma(t|T))

DESCRIPTION
Provides smoothed values from a Kalman filtering.

Example:
<u>smoothing</u> is a used by the function <u>kalman</u>:
[betas,sigmats]=smooth(grocer_F,betat,betaf,sigmatt,sigmatf)

tvp_____Time varying parameters estimation

CALLING SEQUENCE

[rtvp]=tvp(namey,varargin)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* varargin = arguments which can be:
  . a time series
  . a real (nx1) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
  . the string 'noprint' if the user doesn't want to print the results of the regression
  . 'priorb0=x' where x is (k x 1) vector with prior b0 values (default = ols(y,x))
  . 'priorv0=x' where x = (k x k) matrix with prior variance for Q (default = eye(k)*1e+5, a diffuse prior)
  . 'tvpmeth=x' where x = 1 (Q is diagonal and only Q and R are estimated),
    x=1a (no constraint on Q and only Q and R are estimated),
    x=2 (prirov0=0, Q is diagonal and priorb0, Q and R are estimated),
    x=2a (priorv0=0, no constraint on Q, and priorb0, Q and R are estimated)
    default = 1
  . 'Q=x' where x is a (kxk) initial value for Q
  . 'R=x' where x is an initial value for R
  . 'optmeth=x' where x is the optimization program used (optim or -default- maxlik)

OUTPUT:
rtvp = a results tlist with
  - rtvp('meth') = 'tvp'
  - rtvp('Q') = estimated Q
  - rtvp('R') = estimated R
  - rtvp('betat') = B(t/t)
  - rtvp('betaf') = B(t/t-1)
  - rtvp('betas') = B(t/T)
  - rtvp('sigmatt') = sigma(t/t)
  - rtvp('sigmatf') = sigma(t/t-1)
  - rtvp('sigmats') = sigma(t/T)
  - rtvp('param') = estimated parameters
  - rtvp('vcov') = variance-covariance matrix of
    estimated paramters
  - rtvp('tstat') = Student's t of estimated parameters
  - rtvp('y') = y
  - rtvp('x') = x
  - rtvp('yhat') = X(t)*B(t)
  - rtvp('resid') = y-X*B(t)
  - rtvp('like') = log-likelihood
  - rtvp('nobs') = # of observations
  - rtvp('nvar') = # of exogenous variables

- rtvp('tR') = t-stat of estimated R variance
- rtvp('tQ') = t-stat of estimated Q variance
- rtvp('tpriorb0') = t-stat of estimated priorb0 (if method 1a or 2a are used)
- rtvp('param') = estimated parameters in a vector form
- rtvp('tstat') = their t-stat
- rtvp('tvpmeth') = method used in tvp
- rtvp('namey') = name of the endogenous variable
- rtvp('namex') = name of the exogenous variable
- rtvp('prests') = boolean indicating the presence or absence of a time series in the regression
- rtvp('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION
Time-varying parameter maximum likelihood estimation of the linear regression model

$y(t) = X(t)*B(t) + e(t), e(t) = N(0,R)$

$B(t) = B(t-1) + v(t), \quad v(t) = N(0,Q)$

NOTE: the methods '1a' and '2a' are not very robust; so I recommend to use them with much caution!

tvp_param1_____transformation of tvp parameters into kalman compatible ones

CALLING SEQUENCE

[Q,R]=tvp_param1(param)

PARAMETERS

INPUT:
* param = vector of parameters
* k = size of the vectors of parameters

OUTPUT:
* Q = variance of the state equation
* R = variance of the observation equation

DESCRIPTION
In a time-varying estimation, transforms the vectors parameters into their matrix and vectors counterpart in the corresponding Kalman filter.
NOTE: Q is assumed diagonal and priorb0 is not estimated.

Example:
This function should have no other use than providing the transformation of the parameters that must be estimated into the matrices used in the kalman filter.

tvp_param1a_____transformation of tvp parameters into kalman compatible ones

CALLING SEQUENCE

[Q,R]=tvp_param1a(param)

PARAMETERS

INPUT:
* param = vector of parameters
* k = size of the vectors of parameters

OUTPUT:
* Q = variance of the state equation
* R = variance of the observation equation
* b0 = (k x 1) vector of intial conditions for b

NOTE: Q is not assumed diagonal and priorb0 is not estimated

Example:
This function should have no other use than providing the transformation of the parameters that must be estimated into the matrices used in the kalman filter.

tvp_param1d_____transformation of tvp parameters into kalman compatible ones

CALLING SEQUENCE

[Q,R]=tvp_param1d(param)

DESCRIPTION
in a time-varying estimation, transform the vectors parameters into their matrix and vectors counterpart in the corresponding Kalman filter

PARAMETERS

INPUT:
* param = vector of parameters
* k = size of the vectors of parameters

OUTPUT:
* Q = variance of the state equation
* R = variance of the observation equation
* b0 = (k x 1) vector of intial conditions for b

DESCRIPTION
In a time-varying estimation, transform the vectors parameters into their matrix and vectors counterpart in the corresponding Kalman filter.
NOTE: Q is not assumed diagonal and priorb0 is not estimated

Example:
This function should have no other use than providing the transformation of the parameters that must be estimated into the matrices used in the kalman filter.

tvp_param2_____transformation of tvp parameters into kalman compatible ones

CALLING SEQUENCE

[Q,R]=tvp_param2(param)

PARAMETERS

INPUT:
* param = vector of parameters
* k = size of the vectors of parameters

OUTPUT:
* Q = variance of the state equation
* R = variance of the observation equation
* b0 = (k x 1) vector of intial conditions for b

DESCRIPTION
In a time-varying estimation, transforms the vectors parameters into their matrix and vectors counterpart in the corresponding Kalman filter.
NOTE: Q is assumed diagonal and priorb0 is estimated

Example:
This function should have no other use than providing the transformation of the parameters that must be estimated into the matrices used in the kalman filter.

tvp_param2a_____transformation of tvp parameters into kalman compatible ones

CALLING SEQUENCE

[Q,R]=tvp_param2a(param)

PARAMETERS

INPUT:
* param = vector of parameters
* k = size of the vectors of parameters

OUTPUT:
* Q = variance of the state equation
* R = variance of the observation equation

DESCRIPTION
In a time-varying estimation, transforms the vectors parameters into their matrix and vectors counterpart in the corresponding Kalman filter.

NOTE: Q is not assumed diagonal and priorb0 is estimated

Example:
This function should have no other use than providing the transformation of the parameters that must be estimated into the matrices used in the kalman filter.

tvp_param2ad_____transformation of tvp parameters into kalman compatible ones

CALLING SEQUENCE

[Q,R]=tvp_param2ad(param)

PARAMETERS

INPUT:
* param = vector of parameters
* k = size of the vectors of parameters

OUTPUT:
* Q = variance of the state equation
* R = variance of the observation equation

DESCRIPTION
In a time-varying estimation, transforms the vectors parameters into their matrix and vectors counterpart in the corresponding Kalman filter.
NOTE: Q is not assumed diagonal and priorb0 is estimated

Example:
This function should have no other use than providing the transformation of the parameters that must be estimated into the matrices used in the kalman filter.