The Kalman filter in gretl: old syntax

Allin Cottrell and Riccardo "Jack" Lucchetti

June 21, 2016

This document—which originally appeared as Chapter 30 of the Gretl User's Guide—describes the syntax for using the Kalman filter that was available in gretl prior to version 2016c (June, 2016). We believe the new syntax is superior, but we are archiving the old documentation in case anyone needs it.

1 Preamble

The Kalman filter has been used "behind the scenes" in gretl for quite some time, in computing ARMA estimates. But user access to the Kalman filter is new and it has not yet been tested to any great extent. We have run some tests of relatively simple cases against the benchmark of SsfPack Basic. This is state-space software written by Koopman, Shephard and Doornik and documented in Koopman, Shephard and Doornik (1999). It requires Doornik's ox program. Both ox and SsfPack are available as free downloads for academic use but neither is open-source; see http://www.ssfpack.com. Since Koopman is one of the leading researchers in this area, presumably the results from SsfPack are generally reliable. To date we have been able to replicate the SsfPack results in gretl with a high degree of precision.

We welcome both success reports and bug reports.

2 Notation

It seems that in econometrics everyone is happy with $y = X\beta + u$, but we can't, as a community, make up our minds on a standard notation for state-space models. Harvey (1989), Hamilton (1994), Harvey and Proietti (2005) and Pollock (1999) all use different conventions. The notation used here is based on James Hamilton's, with slight variations.

A state-space model can be written as

$$\boldsymbol{\xi}_{t+1} = \mathbf{F}_t\boldsymbol{\xi}_t + \mathbf{v}_t \tag{1}$$

$$\mathbf{y}_t = \mathbf{A}_t'\mathbf{x}_t + \mathbf{H}_t'\boldsymbol{\xi}_t + \mathbf{w}_t \tag{2}$$

where (1) is the state transition equation and (2) is the observation or measurement equation. The state vector, $\boldsymbol{\xi}_t$, is $(r \times 1)$ and the vector of observables, $\mathbf{y}_t$, is $(n \times 1)$; $\mathbf{x}_t$ is a $(k \times 1)$ vector of exogenous variables. The $(r \times 1)$ vector $\mathbf{v}_t$ and the $(n \times 1)$ vector $\mathbf{w}_t$ are assumed to be vector white noise:

$$E(\mathbf{v}_t\mathbf{v}_s') = \mathbf{Q}_t \text{ for } t = s, \text{ otherwise } \mathbf{0}$$

$$E(\mathbf{w}_t\mathbf{w}_s') = \mathbf{R}_t \text{ for } t = s, \text{ otherwise } \mathbf{0}$$

The number of time-series observations will be denoted by T. In the special case when $\mathbf{F}_t = \mathbf{F}$, $\mathbf{H}_t = \mathbf{H}$, $\mathbf{A}_t = \mathbf{A}$, $\mathbf{Q}_t = \mathbf{Q}$ and $\mathbf{R}_t = \mathbf{R}$ for all t, the model is said to be *time-invariant*.

2.1 The Kalman recursions

Using this notation, and assuming for the moment that $\mathbf{v}_t$ and $\mathbf{w}_t$ are mutually independent, the Kalman recursions can be written as follows.

Initialization is via the unconditional mean and variance of $\boldsymbol{\xi}_1$:

$$\hat{\boldsymbol{\xi}}_{1|0} = E(\boldsymbol{\xi}_1)$$
$$\mathbf{P}_{1|0} = E\left\{[\boldsymbol{\xi}_1 - E(\boldsymbol{\xi}_1)]\,[\boldsymbol{\xi}_1 - E(\boldsymbol{\xi}_1)]'\right\}$$

Usually these are given by $\hat{\boldsymbol{\xi}}_{1|0} = \mathbf{0}$ and

$$\mathrm{vec}(\mathbf{P}_{1|0}) = [\mathbf{I}_{r^2} - \mathbf{F} \otimes \mathbf{F}]^{-1} \cdot \mathrm{vec}(\mathbf{Q}) \tag{3}$$

but see below for further discussion of the initial variance.

Iteration then proceeds in two steps.[1] First we update the estimate of the state

$$\hat{\boldsymbol{\xi}}_{t+1|t} = \mathbf{F}_t \hat{\boldsymbol{\xi}}_{t|t-1} + \mathbf{K}_t \mathbf{e}_t \tag{4}$$

where $\mathbf{e}_t$ is the prediction error for the observable:

$$\mathbf{e}_t = \mathbf{y}_t - \mathbf{A}_t'\mathbf{x}_t - \mathbf{H}_t'\hat{\boldsymbol{\xi}}_{t|t-1}$$

and $\mathbf{K}_t$ is the gain matrix, given by

$$\mathbf{K}_t = \mathbf{F}_t \mathbf{P}_{t|t-1} \mathbf{H}_t \boldsymbol{\Sigma}_t^{-1} \tag{5}$$

with

$$\boldsymbol{\Sigma}_t = \mathbf{H}_t' \mathbf{P}_{t|t-1} \mathbf{H}_t + \mathbf{R}_t$$

The second step then updates the estimate of the variance of the state using

$$\mathbf{P}_{t+1|t} = \mathbf{F}_t \mathbf{P}_{t|t-1} \mathbf{F}_t' - \mathbf{K}_t \boldsymbol{\Sigma}_t \mathbf{K}_t' + \mathbf{Q}_t \tag{6}$$

2.2 Cross-correlated disturbances

The formulation given above assumes mutual independence of the disturbances in the state and observation equations, $\mathbf{v}_t$ and $\mathbf{w}_t$. This assumption holds good in many practical applications, but a more general formulation allows for cross-correlation. In place of (1)–(2) we may write

$$\boldsymbol{\xi}_{t+1} = \mathbf{F}_t \boldsymbol{\xi}_t + \mathbf{B}_t \boldsymbol{\varepsilon}_t$$
$$\mathbf{y}_t = \mathbf{A}_t'\mathbf{x}_t + \mathbf{H}_t'\boldsymbol{\xi}_t + \mathbf{C}_t \boldsymbol{\varepsilon}_t$$

where $\boldsymbol{\varepsilon}_t$ is a $(p \times 1)$ disturbance vector, all the elements of which have unit variance, $\mathbf{B}_t$ is $(r \times p)$ and $\mathbf{C}_t$ is $(n \times p)$.

The no-correlation case is nested thus: define $\mathbf{v}_t^*$ and $\mathbf{w}_t^*$ as modified versions of $\mathbf{v}_t$ and $\mathbf{w}_t$, scaled such that each element has unit variance, and let

$$\boldsymbol{\varepsilon}_t = \begin{bmatrix} \mathbf{v}_t^* \\ \mathbf{w}_t^* \end{bmatrix}$$

so that $p = r + n$. Then (suppressing time subscripts for simplicity) let

$$\mathbf{B} = \begin{bmatrix} \boldsymbol{\Gamma}_{r\times r} & \vdots & \mathbf{0}_{r\times n} \end{bmatrix}$$
$$\mathbf{C} = \begin{bmatrix} \mathbf{0}_{n\times r} & \vdots & \boldsymbol{\Lambda}_{n\times n} \end{bmatrix}$$

[1]For a justification of the following formulae see the classic book by Anderson and Moore (1979) or, for a more modern treatment, Pollock (1999) or Hamilton (1994). A transcription of R. E. Kalman's original paper (Kalman, 1960) is available at `http://www.cs.unc.edu/~welch/kalman/kalmanPaper.html`.

where $\mathbf{\Gamma}$ and $\mathbf{\Lambda}$ are lower triangular matrices satisfying $\mathbf{Q} = \mathbf{\Gamma}\mathbf{\Gamma}'$ and $\mathbf{R} = \mathbf{\Lambda}\mathbf{\Lambda}'$ respectively. The zero sub-matrices in the above expressions for $\mathbf{B}$ and $\mathbf{C}$ produce the case of mutual independence; this corresponds to the condition $\mathbf{BC}' = \mathbf{0}$.

In the general case p is not necessarily equal to $r + n$, and $\mathbf{BC}'$ may be non-zero. This means that the Kalman gain equation (5) must be modified as

$$\mathbf{K}_t = (\mathbf{F}_t\mathbf{P}_{t|t-1}\mathbf{H}_t + \mathbf{B}_t\mathbf{C}_t')\mathbf{\Sigma}_t^{-1} \tag{7}$$

Otherwise, the equations given earlier hold good, if we write $\mathbf{BB}'$ in place of $\mathbf{Q}$ and $\mathbf{CC}'$ in place of $\mathbf{R}$.

In the account of gretl's Kalman facility below we take the uncorrelated case as the baseline, but add remarks on how to handle the correlated case where applicable.

3 Intended usage

The Kalman filter can be used in three ways: two of these are the classic forward and backward pass, or filtering and smoothing respectively; the third use is simulation. In the filtering/smoothing case you have the data $\mathbf{y}_t$ and you want to reconstruct the states $\boldsymbol{\xi}_t$ (and the forecast errors as a by-product), but we may also have a computational apparatus that does the reverse: given artificially-generated series $\mathbf{w}_t$ and $\mathbf{v}_t$, generate the states $\boldsymbol{\xi}_t$ (and the observables $\mathbf{y}_t$ as a by-product).

The usefulness of the classical filter is well known; the usefulness of the Kalman filter as a simulation tool may be huge too. Think for instance of Monte Carlo experiments, simulation-based inference—see Gourieroux and Monfort (1996)—or Bayesian methods, especially in the context of the estimation of DSGE models.

4 Overview of syntax

Using the Kalman filter in gretl is a two-step process. First you set up your filter, using a block of commands starting with `kalman` and ending with `end kalman`—much like the `gmm` command. Then you invoke the functions `kfilter`, `ksmooth` or `ksimul` to do the actual work. The next two sections expand on these points.

5 Defining the filter

Each line within the `kalman ... end kalman` block takes the form

keyword value

where *keyword* represents a matrix, as shown below. (An additional matrix which may be useful in some cases is introduced later under the heading “Constant term in the state transition”.)

Keyword	Symbol	Dimensions
`obsy`	$\mathbf{y}$	$T \times n$
`obsymat`	$\mathbf{H}$	$r \times n$
`obsx`	$\mathbf{x}$	$T \times k$
`obsxmat`	$\mathbf{A}$	$k \times n$
`obsvar`	$\mathbf{R}$	$n \times n$
`statemat`	$\mathbf{F}$	$r \times r$
`statevar`	$\mathbf{Q}$	$r \times r$
`inistate`	$\hat{\boldsymbol{\xi}}_{1\|0}$	$r \times 1$
`inivar`	$\mathbf{P}_{1\|0}$	$r \times r$

For the data matrices **y** and **x** the corresponding *value* may be the name of a predefined matrix, the name of a data series, or the name of a list of series.[2]

For the other inputs, *value* may be the name of a predefined matrix or, if the input in question happens to be (1×1), the name of a scalar variable or a numerical constant. If the *value* of a coefficient matrix is given as the name of a matrix or scalar variable, the input is not "hard-wired" into the Kalman structure, rather a record is made of the *name* of the variable and on each run of a Kalman function (as described below) its value is re-read. It is therefore possible to write one `kalman` block and then do several filtering or smoothing passes using different sets of coefficients.[3] An example of this technique is provided later, in the example scripts 1 and 2. This facility to alter the values of the coefficients between runs of the filter is to be distinguished from the case of *time-varying* matrices, which is discussed below.

Not all of the above-mentioned inputs need be specified in every case; some are optional. (In addition, you can specify the matrices in any order.) The mandatory elements are **y**, **H**, **F** and **Q**, so the minimal `kalman` block looks like this:

```
kalman
  obsy y
  obsymat H
  statemat F
  statevar Q
end kalman
```

The optional matrices are listed below, along with the implication of omitting the given matrix.

Keyword	If omitted...
`obsx`	no exogenous variables in observation equation
`obsxmat`	no exogenous variables in observation equation
`obsvar`	no disturbance term in observation equation
`inistate`	$\hat{\boldsymbol{\xi}}_{1\|0}$ is set to a zero vector
`inivar`	$\mathbf{P}_{1\|0}$ is set automatically

It might appear that the `obsx` (**x**) and `obsxmat` (**A**) matrices must go together—either both are given or neither is given. But an exception is granted for convenience. If the observation equation includes a constant but no additional exogenous variables, you can give a $(1\times n)$ value for **A** without having to specify `obsx`. More generally, if the row dimension of **A** is 1 greater than the column dimension of **x**, it is assumed that the first element of **A** is associated with an implicit column of 1s.

Regarding the automatic initialization of $\mathbf{P}_{1|0}$ (in case no `inivar` input is given): by default this is done as in equation (3). However, this method is applicable only if all the eigenvalues of **F** lie inside the unit circle. If this condition is not satisfied we instead apply a diffuse prior, setting $\mathbf{P}_{1|0} = \kappa\mathbf{I}_r$ with $\kappa = 10^7$. If you wish to impose this diffuse prior from the outset, append the option flag `--diffuse` to the `end kalman` statement.[4]

5.1 Time-varying matrices

Any or all of the matrices `obsymat`, `obsxmat`, `obsvar`, `statemat` and `statevar` may be time-varying. In that case the *value* corresponding to the matrix keyword should be given in a special

[2]Note that the data matrices `obsy` and `obsx` have T rows. That is, the column vectors $\mathbf{y}_t$ and $\mathbf{x}_t$ in (1) and (2) are in fact the transposes of the t-dated rows of the full matrices.

[3]Note, however, that the dimensions of the various input matrices are defined via the initial `kalman` set-up and it is an error if any of the matrices are changed in size.

[4]Initialization of the Kalman filter outside of the case where equation (3) applies has been the subject of much discussion in the literature—see for example de Jong (1991), Koopman (1997). At present gretl does not implement any of the more elaborate proposals that have been made.

form: the name of an existing matrix plus a function call which modifies that matrix, separated by a semicolon. Note that in this case you must use a matrix variable, even if the matrix in question happens to be 1×1.

For example, suppose the matrix $\mathbf{H}$ is time-varying. Then we might write

```
  obsymat H ; modify_H(&H, theta)
```

where `modify_H` is a user-defined function which modifies matrix `H` (and `theta` is a suitable additional argument to that function, if required).

The above is just an illustration: the matrix argument does not have to come first, and the function can have as many arguments as you like. The essential point is that the function must modify the specified matrix, which requires that it be given as an argument in "pointer" form (preceded by &). The function need not return any value directly; if it does, that value is ignored.

Such matrix-modifying functions will be called at each time-step of the filter operation, prior to performing any calculations. They have access to the current time-step of the Kalman filter via the internal variable `$kalman_t`, which has value 1 on the first step, 2 on the second, and so on, up to step T. They also have access to the previous n-vector of forecast errors, $\mathbf{e}_{t-1}$, under the name `$kalman_uhat`. When $t = 1$ this will be a zero vector.

5.2 Correlated disturbances

Defining a filter in which the disturbances $\mathbf{v}_t$ and $\mathbf{w}_t$ are correlated involves one modification to the account given above. If you append the `--cross` option flag to the `end kalman` statement, then the matrices corresponding to the keywords `statevar` and `obsvar` are interpreted not as $\mathbf{Q}$ and $\mathbf{R}$ but rather as $\mathbf{B}$ and $\mathbf{C}$ as discussed in section 2.1. Gretl then computes $\mathbf{Q} = \mathbf{BB}'$ and $\mathbf{R} = \mathbf{CC}'$ as well as the cross-product $\mathbf{BC}'$ and utilizes the modified expression for the gain as given in equation (7). As mentioned above, $\mathbf{B}$ should be $(r \times p)$ and $\mathbf{C}$ should be $(n \times p)$, where p is the number of elements in the combined disturbance vector $\boldsymbol{\varepsilon}_t$.

5.3 Constant term in the state transition

In some applications it is useful to be able to represent a constant term in the state transition equation explicitly; that is, equation (1) becomes

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\mu} + \mathbf{F}_t \boldsymbol{\xi}_t + \mathbf{v}_t \tag{8}$$

This is never strictly necessary; the system (1) and (2) is general enough to accommodate such a term, by absorbing it as an extra (unvarying) element in the state vector. But this comes at the cost of expanding all the matrices that touch the state ($\boldsymbol{\xi}$, $\mathbf{F}$, $\mathbf{v}$, $\mathbf{Q}$, $\mathbf{H}$), making the model relatively awkward to formulate and forecasts relatively expensive to compute.

As a simple illustration, consider a univariate model in which the state, s_t, is just a random walk with drift μ and the observed variable, y_t, is the state plus white noise:

$$s_{t+1} = \mu + s_t + v_t \tag{9}$$

$$y_t = s_t + w_t \tag{10}$$

Putting this into the standard form of (1) and (2) we get:

$$\begin{bmatrix} s_{t+1} \\ \mu \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_t \\ \mu \end{bmatrix} + \begin{bmatrix} v_t \\ 0 \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s_t \\ \mu \end{bmatrix} + w_t$$

In such a simple case the notational and computational burden is not very great; nonetheless it is clearly more "natural" to express this system in the form of (9) and (10) and in a multivariate model the gain in parsimony could be substantial.

For this reason we support the use of an additional named matrix in the `kalman` setup, namely `stconst`. This corresponds to $\boldsymbol{\mu}$ in equation (8); it should be an $r \times 1$ vector (or if $r = 1$ may be given as the name of a scalar variable). The use of `stconst` in setting up a filter corresponding to (9) and (10) is shown below.

```
matrix H = {1}
matrix R = {1}
matrix F = {1}
matrix Q = {1}
matrix mu = {0.05}

kalman
  obsy y
  obsymat H
  obsvar R
  statemat F
  statevar Q
  stconst mu
end kalman
```

5.4 Handling of missing values

It is acceptable for the data matrices, `obsy` and `obsx`, to contain missing values. In this case the filtering operation will work around the missing values, and the `ksmooth` function can be used to obtain estimates of these values. However, there are two points to note.

First, gretl's default behavior is to skip missing observations when constructing matrices from data series. To change this, use the `set` command thus:

```
set skip_missing off
```

Second, the handling of missing values is not yet quite right for the case where the observable vector $\mathbf{y}_t$ contains more than one element. At present, if any of the elements of $\mathbf{y}_t$ are missing the entire observation is ignored. Clearly it should be possible to make use of any non-missing elements, and this is not very difficult in principle, it's just awkward and is not implemented yet.

5.5 Persistence and identity of the filter

At present there is no facility to create a "named filter". Only one filter can exist at any point in time, namely the one created by the last `kalman` block.[5] If a filter is already defined, and you give a new `kalman` block, the old filter is over-written. Otherwise the existing filter persists (and remains available for the `kfilter`, `ksmooth` and `ksimul` functions) until either (a) the gretl session is terminated or (b) the command `delete kalman` is given.

6 The `kfilter` function

Once a filter is established, as discussed in the previous section, `kfilter` can be used to run a forward, forecasting pass. This function returns a scalar code: 0 for successful completion, or 1 if numerical problems were encountered. On successful completion, two scalar accessor variables

[5]This is not quite true: more precisely, there can be no more than one Kalman filter *at each level of function execution.* That is, if a gretl script creates a Kalman filter, a user-defined function called from that script may also create a filter, without interfering with the original one.

become available: `$kalman_lnl`, which gives the overall log-likelihood under the joint normality assumption,

$$\ell = -\frac{1}{2}\left[nT\log(2\pi) + \sum_{t=1}^{T}\log|\mathbf{\Sigma}_t| + \sum_{t=1}^{T}\mathbf{e}_t'\mathbf{\Sigma}_t^{-1}\mathbf{e}_t\right]$$

and `$kalman_s2`, which gives the estimated variance,

$$\hat{\sigma}^2 = \frac{1}{nT}\sum_{t=1}^{T}\mathbf{e}_t'\mathbf{\Sigma}_t^{-1}\mathbf{e}_t$$

(but see below for modifications to these formulae for the case of a diffuse prior). In addition the accessor `$kalman_llt` gives a $(T \times 1)$ vector, element t of which is

$$\ell_t = -\frac{1}{2}\left[n\log(2\pi) + \log|\mathbf{\Sigma}_t| + \mathbf{e}_t'\mathbf{\Sigma}_t^{-1}\mathbf{e}_t\right]$$

The `kfilter` function does not require any arguments, but up to five matrix quantities may be retrieved via optional pointer arguments. Each of these matrices has T rows, one for each time-step; the contents of the rows are shown in the following listing.

1. Forecast errors for the observable variables: $\mathbf{e}_t'$, n columns.
2. Variance matrix for the forecast errors: $\mathrm{vech}(\mathbf{\Sigma}_t)'$, $n(n+1)/2$ columns.
3. Estimate of the state vector: $\hat{\boldsymbol{\xi}}_{t|t-1}'$, r columns.
4. MSE of estimate of the state vector: $\mathrm{vech}(\mathbf{P}_{t|t-1})'$, $r(r+1)/2$ columns.
5. Kalman gain: $\mathrm{vec}(\mathbf{K}_t)'$, rn columns.

Unwanted trailing arguments can be omitted, otherwise unwanted arguments can be skipped by using the keyword `null`. For example, the following call retrieves the forecast errors in the matrix `E` and the estimate of the state vector in `S`:

```
matrix E S
kfilter(&E, null, &S)
```

Matrices given as pointer arguments do not have to be correctly dimensioned in advance; they will be resized to receive the specified content.

Further note: in general, the arguments to `kfilter` should all be matrix-pointers, but under two conditions you can give a pointer to a series variable instead. The conditions are: (i) the matrix in question has just one column in context (for example, the first two matrices will have a single column if the length of the observables vector, n, equals 1) and (ii) the time-series length of the filter is equal to the current gretl sample size.

6.1 Likelihood under the diffuse prior

There seems to be general agreement in the literature that the log-likelihood calculation should be modified in the case of a diffuse prior for $\mathbf{P}_{1|0}$. However, it is not clear to us that there is a well-defined "correct" method for this. At present we emulate `SsfPack` (see Koopman *et al.* (1999) and section 1). In case $\mathbf{P}_{1|0} = \kappa\mathbf{I}_r$, we set $d = r$ and calculate

$$\ell = -\frac{1}{2}\left[(nT-d)\log(2\pi) + \sum_{t=1}^{T}\log|\mathbf{\Sigma}_t| + \sum_{t=1}^{T}\mathbf{e}_t'\mathbf{\Sigma}_t^{-1}\mathbf{e}_t - d\log(\kappa)\right]$$

and

$$\hat{\sigma}^2 = \frac{1}{nT-d}\sum_{t=1}^{T}\mathbf{e}_t'\mathbf{\Sigma}_t^{-1}\mathbf{e}_t$$

7 The `ksmooth` function

This function returns the $(T \times r)$ matrix of smoothed estimates of the state vector—that is, estimates based on all T observations: row t of this matrix holds $\hat{\boldsymbol{\xi}}'_{t|T}$. This function has no required arguments but it offers one optional matrix-pointer argument, which retrieves the variance of the smoothed state estimate, $\mathbf{P}_{t|T}$. The latter matrix is $(T \times r(r+1)/2)$; each row is in transposed vech form. Examples:

```
matrix S = ksmooth()  # smoothed state only
matrix P
S = ksmooth(&P)       # the variance is wanted
```

These values are computed via a backward pass of the filter, from $t = T$ to $t = 1$, as follows:

$$
\begin{aligned}
\mathbf{L}_t &= \mathbf{F}_t - \mathbf{K}_t \mathbf{H}'_t \\
\mathbf{u}_{t-1} &= \mathbf{H}_t \boldsymbol{\Sigma}_t^{-1} \mathbf{e}_t + \mathbf{L}'_t \mathbf{u}_t \\
\mathbf{U}_{t-1} &= \mathbf{H}_t \boldsymbol{\Sigma}_t^{-1} \mathbf{H}'_t + \mathbf{L}'_t \mathbf{U}_t \mathbf{L}_t \\
\hat{\boldsymbol{\xi}}_{t|T} &= \hat{\boldsymbol{\xi}}_{t|t-1} + \mathbf{P}_{t|t-1} \mathbf{u}_{t-1} \\
\mathbf{P}_{t|T} &= \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1} \mathbf{U}_{t-1} \mathbf{P}_{t|t-1}
\end{aligned}
$$

with initial values $\mathbf{u}_T = 0$ and $\mathbf{U}_T = 0$.[6]

This iteration is preceded by a special forward pass in which the matrices $\mathbf{K}_t$, $\boldsymbol{\Sigma}_t^{-1}$, $\hat{\boldsymbol{\xi}}_{t|t-1}$ and $\mathbf{P}_{t|t-1}$ are stored for all t. If $\mathbf{F}$ is time-varying, its values for all t are stored on the forward pass, and similarly for $\mathbf{H}$.

8 The `ksimul` function

This simulation function takes up to three arguments. The first, mandatory, argument is a $(T \times r)$ matrix containing artificial disturbances for the state transition equation: row t of this matrix represents $\mathbf{v}'_t$. If the current filter has a non-null $\mathbf{R}$ (`obsvar`) matrix, then the second argument should be a $(T \times n)$ matrix containing artificial disturbances for the observation equation, on the same pattern. Otherwise the second argument should be given as `null`. If $r = 1$ you may give a series for the first argument, and if $n = 1$ a series is acceptable for the second argument.

Provided that the current filter does not include exogenous variables in the observation equation (`obsx`), the T for simulation need not equal that defined by the original `obsy` data matrix: in effect T is temporarily redefined by the row dimension of the first argument to `ksimul`. Once the simulation is completed, the T value associated with the original data is restored.

The value returned by `ksimul` is a $(T \times n)$ matrix holding simulated values for the observables at each time step. A third optional matrix-pointer argument allows you to retrieve a $(T \times r)$ matrix holding the simulated state vector. Examples:

```
matrix Y = ksimul(V)    # obsvar is null
Y = ksimul(V, W)        # obsvar is non-null
matrix S
Y = ksimul(V, null, &S) # the simulated state is wanted
```

The initial value $\boldsymbol{\xi}_1$ is calculated thus: we find the matrix $\mathbf{T}$ such that $\mathbf{TT}' = \mathbf{P}_{1|0}$ (as given by the `inivar` element in the `kalman` block), multiply it into $\mathbf{v}_1$, and add the result to $\boldsymbol{\xi}_{1|0}$ (as given by `inistate`).

If the disturbances are correlated across the two equations the arguments to `ksimul` must be revised: the first argument should be a $(T \times p)$ matrix, each row of which represents $\boldsymbol{\varepsilon}'_t$ (see section 2.1), and the second argument should be given as `null`.

[6] See I. Karibzhanov's exposition at `http://karibzhanov.com/help/kalcvs.htm`.

9 Example 1: ARMA estimation

As is well known, the Kalman filter provides a very efficient way to compute the likelihood of ARMA models; as an example, take an ARMA(1,1) model

$$y_t = \phi y_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1}$$

One of the ways the above equation can be cast in state-space form is by defining a latent process $\xi_t = (1 - \phi L)^{-1} \varepsilon_t$. The observation equation corresponding to (2) is then

$$y_t = \xi_t + \theta \xi_{t-1} \tag{11}$$

and the state transition equation corresponding to (1) is

$$\begin{bmatrix} \xi_t \\ \xi_{t-1} \end{bmatrix} = \begin{bmatrix} \phi & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \xi_{t-1} \\ \xi_{t-2} \end{bmatrix} + \begin{bmatrix} \varepsilon_t \\ 0 \end{bmatrix}$$

The gretl syntax for a corresponding `kalman` block would be

```
matrix H = {1; theta}
matrix F = {phi, 0; 1, 0}
matrix Q = {s^2, 0; 0, 0}

kalman
    obsy y
    obsymat H
    statemat F
    statevar Q
end kalman
```

Note that the observation equation (11) does not include an "error term"; this is equivalent to saying that $V(\mathbf{w}_t) = 0$ and, as a consequence, the `kalman` block does not include an `obsvar` keyword.

Once the filter is set up, all it takes to compute the log-likelihood for given values of ϕ, θ and σ^2 is to execute the `kfilter()` function and use the `$kalman_lnl` accessor (which returns the total log-likelihood) or, more appropriately if the likelihood has to be maximized through `mle`, the `$kalman_llt` accessor, which returns the series of individual contribution to the log-likelihood for each observation. An example is shown in script 1.

10 Example 2: local level model

Suppose we have a series $y_t = \mu_t + \varepsilon_t$, where μ_t is a random walk with normal increments of variance σ_1^2 and ε_t is a normal white noise with variance σ_2^2, independent of μ_t. This is known as the "local level" model in Harvey's (1989) terminology, and it can be cast in state-space form as equations (1)-(2) with $\mathbf{F} = 1$, $\mathbf{v}_t \sim N(0, \sigma_1^2)$, $\mathbf{H} = 1$ and $\mathbf{w}_t \sim N(0, \sigma_2^2)$. The translation to a `kalman` block is

```
kalman
   obsy y
   obsymat 1
   statemat 1
   statevar s2
   obsvar s1
end kalman --diffuse
```

The two unknown parameters σ_1^2 and σ_2^2 can be estimated via maximum likelihood. Script 2 provides an example of simulation and estimation of such a model. For the sake of brevity, simulation is carried out via ordinary gretl commands, rather than the state-space apparatus described above.

Example 1: ARMA estimation

```
function void arma11_via_kalman(series y)
    /* parameter initalization */
    phi = 0
    theta = 0
    sigma = 1

    /* Kalman filter setup */
    matrix H = {1; theta}
    matrix F = {phi, 0; 1, 0}
    matrix Q = {sigma^2, 0; 0, 0}

    kalman
        obsy y
        obsymat H
        statemat F
        statevar Q
    end kalman

    /* maximum likelihood estimation */
    mle logl = ERR ? NA : $kalman_llt
        H[2] = theta
        F[1,1] = phi
        Q[1,1] = sigma^2
        ERR = kfilter()
        params phi theta sigma
    end mle -h
end function

# ------------------------ main ---------------------------

open arma.gdt        # open the "arma" example dataset
arma11_via_kalman(y) # estimate an arma(1,1) model
arma 1 1 ; y --nc    # check via native command
```

The example contains two functions: the first one carries out the estimation of the unknown parameters σ_1^2 and σ_2^2 via maximum likelihood; the second one uses these estimates to compute a smoothed estimate of the unobservable series μ_t under the name muhat. A plot of μ_t and its estimate is presented in Figure 1.

By appending the following code snippet to the example in Table 2, one may check the results against the R command StructTS.

```
foreign language=R --send-data
  y <- gretldata[,"y"]
  a <- StructTS(y, type="level")
  a
  StateFromR <- as.ts(tsSmooth(a))
  gretl.export(StateFromR)
end foreign

append @dotdir/StateFromR.csv

ols muhat 0 StateFromR --simple
```

References

Anderson, B. and J. Moore (1979) *Optimal Filtering*, Upper Saddle River, NJ: Prentice-Hall.

Gourieroux, C. and A. Monfort (1996) *Simulation-Based Econometric Methods*, Oxford: Oxford University Press.

Hamilton, J. D. (1994) *Time Series Analysis*, Princeton, NJ: Princeton University Press.

Harvey, A. C. (1989) *Forecasting, structural time series models and the Kalman filter*, Cambridge: Cambridge University Press.

Harvey, A. C. and T. Proietti (2005) *Readings in Unobserved Component Models*, Oxford: Oxford University Press.

de Jong, P. (1991) 'The diffuse Kalman filter', *The Annals of Statistics* 19: 1073–1083.

Kalman, R. E. (1960) 'A new approach to linear filtering and prediction problems', *Transactions of the ASME–Journal of Basic Engineering* 82(Series D): 35–45.

Koopman, S. J. (1997) 'Exact initial Kalman filtering and smoothing for nonstationary time series models', *Journal of the American Statistical Association* 92: 1630–1638.

Koopman, S. J., N. Shephard and J. A. Doornik (1999) 'Statistical algorithms for models in state space using SsfPack 2.2', *Econometrics Journal* 2: 107–160.

Pollock, D. S. G. (1999) *A Handbook of Time-Series Analysis, Signal Processing and Dynamics*, New York: Academic Press.

Example 2: Local level model

```
function matrix local_level (series y)
    /* starting values */
    scalar s1 = 1
    scalar s2 = 1

    /* Kalman filter set-up */
    kalman
       obsy y
       obsymat 1
       statemat 1
       statevar s2
       obsvar s1
    end kalman --diffuse

    /* ML estimation */
    mle ll = ERR ? NA : $kalman_llt
        ERR = kfilter()
        params s1 s2
    end mle

    return s1 ~ s2
end function

function series loclev_sm (series y, scalar s1, scalar s2)
    /* return the smoothed estimate of \mu_t */
    kalman
       obsy y
       obsymat 1
       statemat 1
       statevar s2
       obsvar s1
    end kalman --diffuse
    series ret = ksmooth()
    return ret
end function

/* -------------------- main script -------------------- */

nulldata 200
set seed 202020
setobs 1 1 --special
true_s1 = 0.25
true_s2 = 0.5
v = normal() * sqrt(true_s1)
w = normal() * sqrt(true_s2)
mu = 2 + cum(w)
y = mu + v

matrix Vars = local_level(y)           # estimate the variances
muhat = loclev_sm(y, Vars[1], Vars[2]) # compute the smoothed state
```

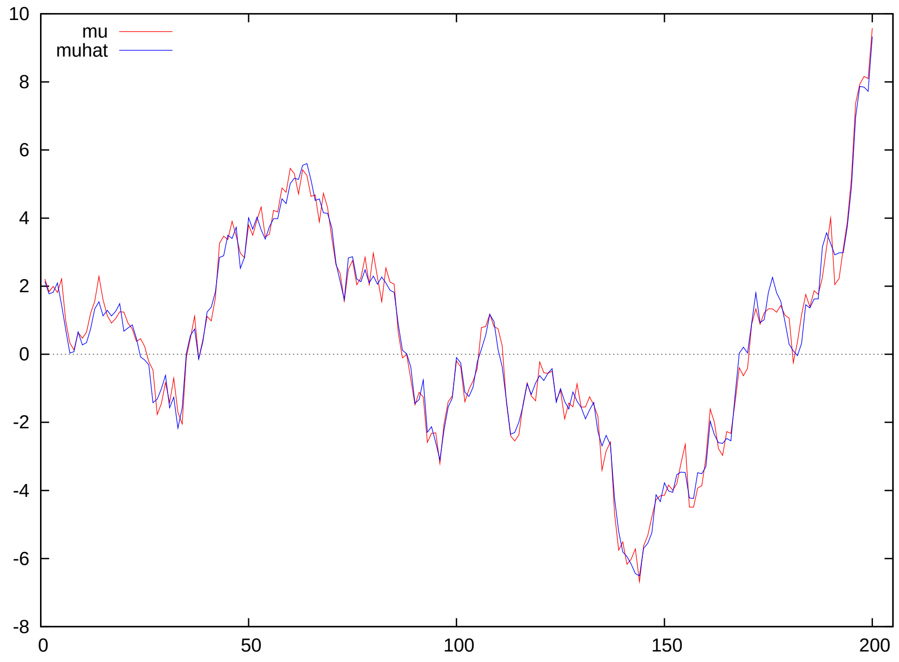

Figure 1: Local level model: μ_t and its smoothed estimate