Kalman 'secrets'

Allin Cottrell

August 8, 2022

1 Introduction

This note describes new functionality for state space models available in gretl 2022b but (deliberately) not yet documented. Both the underlying code and the user interface are still open to modification in light of testing, bug reports and users' suggestions. Note that as things stand the revised code is 100 percent backward-compatible: you should get exactly the same results as before from existing scripts, unless you engage any new options. No user interfaces have been changed other than to add new options.

To set the scene, here's a reminder of the basic linear, Gaussian state space model supported by gretl. (We also support certain extensions, which will be referenced where relevant.) The first equation is the observation or measurement equation, the second is the state transition equation.

$$\begin{aligned} y_t &= Z_t \alpha_t + \varepsilon_t \\ \alpha_{t+1} &= T_t \alpha_t + \eta_t \end{aligned}$$

y_t is $n \times 1$; α_t is $r \times 1$; Z_t and T_t are conformable matrices that may or may not be time-varying; and ε_t and η_t are Gaussian vector disturbances.

2 Long story short

There's a lot that can be said on this topic, but here's what is probably the bottom line for many users. If you have scripts where you set `diffuse = 1` on your Kalman bundle you can now try `diffuse = 2` instead. This invokes the new "exact initial" method for state space models with a diffuse initializer. Don't expect identical results from the new code, but to the extent results differ the new ones should be somewhat more accurate. (If results differ wildly you've probably found a bug; please report it!) You may also find that the new code is faster; it should be less likely to get hung up on numerical problems that delay or prevent convergence of ML estimation.

3 Variance of disturbances

For reference below, we note three common representations of the variance of the disturbances in the two equations above.

1. The "basic" case (`STD_VAR`): ε_t and η_t are assumed to be mutually uncorrelated, and we write their respective variance matrices as $V(\varepsilon_t)$ $(n \times n)$ and $V(\eta_t)$ $(r \times r)$. These variance matrices, and others noted below, may or may not be time-varying.

2. The "de Jong" case (`DJ_VAR`): write $\varepsilon_t = G_t \nu_t$ and $\eta_t = H_t \nu_t$, where G_t is $n \times p$, H_t is $r \times p$ and p is the length of the underlying disturbance vector ν_t. This formulation allows for correlation of the disturbances across the equations, if $H_t G_t'$ is non-zero.

3. The "Durbin–Koopman" case (DK_VAR): as in the first case assume that the disturbances are uncorrelated across the equations, but write $\eta_t = R_t\xi_t$ and $V(\eta_t) = R_t Q_t R_t'$, where R_t is a selection matrix and $Q_t = V(\xi_t)$. Let $m \leq r$ denote the dimension of ξ_t. Then Q_t is $m \times m$ and R_t is $r \times m$. This allows for the possibility that there are fewer disturbances to the state than elements of the state vector.

4 The "diffuse" case

To provide context for gretl's new state-space functionality we need one more piece of terminology. We describe a state space model as "diffuse" if the model is non-stationary and we have no prior information to pin down the variance of α_t, which is usually denoted by P_t. In that case there are two possible approaches. The "traditional" one is to ascribe a very large variance to the initial P_t, as in $P_0 = \kappa \times I_r$ where κ is, say, 10^7. This is what gretl has done, up to version 2022a. But while it seems to work fairly well in most cases it is nowadays generally deprecated in favor of one or other "exact initial" method. Such methods depend on derivation of the properties of the Kalman filter (and smoother) in the limit as the aforementioned "very large" variance tends to infinity.

5 New features

We can now identify the new features in gretl's support for state space models.

- While the disturbance variants labeled STD_VAR and DJ_VAR above have long been supported, the DK_VAR representation is now also available.
- Two variants of the "exact initial" treatment of the diffuse case are now implemented and available as options.

5.1 Disturbance variance options

As described in the *Gretl User's Guide*, a state space bundle is obtained via the `ksetup()` function, which takes four required arguments plus an optional fifth. The required arguments are `obsy` (y), `obsymat` (Z), `statemat` (T) and `statevar`.

If just four arguments are provided the STD_VAR variance case is assumed, and `statevar` is taken to be the $r \times r$ matrix $V(\eta)$. If there's a disturbance term in the observation equation its $n \times n$ variance matrix $V(\varepsilon)$ is then added to the bundle separately under the key `obsvar`.

If a fifth argument is supplied to `ksetup()`, to this point we have assumed the DJ_VAR variance case: `statemat` is interpreted as the $r \times p$ factor H and the last argument as G. That is still the handling of the 5-argument call, but it can now be inflected by appending a sixth, boolean argument. If that has a non-zero value it establishes the DK_VAR case: `statevar` is taken to be Q and the fifth argument taken to be R. Note that in this case `obsvar` should be added separately as in STD_VAR.

The following statements illustrate the three cases.

```
# STD_VAR
bundle kb1 = ksetup(y, Z', T, Veta)
kb1.obsvar = Veps # if wanted

# DJ_VAR
bundle kb2 = ksetup(y, Z', T, H, G)

# DK_VAR
bundle kb3 = ksetup(y, Z', T, Q, R, 1)
kb3.obsvar = Veps # if wanted
```

5.2 Exact initial options

It would be nice if there were one canonical approach to diffuse state space models that worked well for all cases but unfortunately that does not seem to be true. In libgretl we have implemented two such methods: the "univariate approach to multivariate observable" advocated by Durbin and Koopman (2012) and the augmented Kalman method set out by de Jong (1991) and de Jong and Chu-Chun-Lin (2003). The first of these is used in the KFAS package for R (Helske, 2017) and the second by the `sspace` command in Stata.[1] We refer to them via the labels `univariate` and `dejong` below.

The `univariate` approach handles a vector observable by "unpacking" it and substituting scalar calculations for matrix ones so far as possible. Durbin and Koopman claim it is faster than the alternatives, and this seems to be borne out in our testing so far. It is also able to deal in a straightforward way with incomplete observations (where some but not all elements of y_t are missing at time t): it can utilize any non-missing elements while ignoring the missing ones. However, it runs into complications if (a) the variance matrix of the observation disturbances is not diagonal and/or (b) the disturbances are correlated between the state and observation equations. Case (a) can be handled at the cost of some extra preliminary computation—transforming y and Z to induce a diagonal variance matrix—and this is automatically carried out by gretl if needed. Handling case (b) is more bothersome, requiring augmentation of the state; this is supported by KFAS but not by gretl.

The `dejong` approach has no problem with the variance cases (a) and (b) mentioned above. However, it's not clear how incomplete observations can be handled, and at present observations with any missing elements are ignored.

In short, there are cases where `univariate` is probably better, and cases that are not handled by `univariate` but where `dejong` works fine. Hence our decision to implement both methods.

Table 1 sets out the various cases that arise via combination of "code" (where `legacy` indicates the Kalman code as of gretl 2022a) and "diffuse status" (i.e. whether the model is diffuse, and if so how it is handled). Note that although the primary virtue of `univariate` and `dejong` is their handling of the exact diffuse case, these methods should be able to handle the non-diffuse case and the traditional "κ-diffuse" case (with their "exact" mode turned off internally).

	non-diffuse	κ-diffuse	exact diffuse
code	`diffuse=0`	`diffuse=1`	`diffuse=2`
`legacy`	1	2	NA
`univariate`	4	5	6
`dejong`	7	8	9

Table 1: Cross-tabulation of code-path and diffuse status. Numbers in cells are used for reference in the text.

The case used depends on various points, the primary one being the `diffuse` integer member of the state space bundle, which defaults to 0 but can be set to 1 or 2.

- `diffuse=0`: case 1 is the default (for backward compatibility) but case 4 or 7 can be selected, by adding `univariate=1` or `dejong=1` to the bundle.
- `diffuse=1`: case 2 is the default but case 5 or 8 can be selected as above.
- `diffuse=2`: if the `DJ_VAR` variance representation is selected the case is 9; otherwise it defaults to 6 but can be switched to 9 via `dejong=1`.

[1] See https://www.stata.com/manuals/tssspace.pdf.

For cases in the same column—namely {1,4,7}, {2,5,8} and {6,9}—results from `kfilter()`, `ksmooth()` and `kdsmooth()` should in principle be the same across the code-paths but in practice there are bound to be slight differences due to the differing algorithms employed. And note that slight differences at that level may be somewhat amplified by iterated filtering as in ML estimation. That said, substantial within-column differences probably signal a bug in one or more of the code-paths.

Differences across columns 2 and 3 are of course expected; see section 2 above.

6 Testing

Here are some questions we're interested in, for anyone willing to test the new code.

1. Are there any instances where results from cases 4 or 7 differ substantially from case 1? Or where results are similar but case 1 is substantially faster than the others?
2. Are there any instances where results from cases 5 or 8 differ substantially from case 2? Or where results are similar but case 2 is substantially faster than the others?
3. Are there any instances where results from cases 6 and 9 differ substantially?
4. Is there a systematic speed difference between cases 6 and 9?

If the answers to questions 1 and 2 turn out to be negative, we should be able to scrap the legacy code-path in a future gretl release. Answers to questions 3 and 4 will be helpful in debugging the new methods, and in deciding whether `univariate` or `dejong` should be the default when they are both applicable.

As an aid for testing, some of the switches that can be set via script can also be set "globally" via environment variables. These are shown in Table 2.

`KALMAN_UNIVAR`	Equivalent to `univariate=1` in a script. Ignored if `DJ_VAR` is in force.
`KALMAN_DEJONG`	Equivalent to `dejong=1` in a script.
`KALMAN_EXACT`	If `diffuse=1` is found in a script, it will be automatically "promoted" to exact diffuse.
`KALMAN_TRACE`	Shows the Kalman-related functions being called (0=off, 1=on). Useful if you're in any doubt as to which code-path is being followed.
`KALMAN_DEBUG`	Debugging info to `stderr` (0=none, 1=some, 2=more).

Table 2: Environment variables that can be used to influence state space models

References

Durbin, J. and S. J. Koopman (2012) *Time Series Analysis by State Space Methods*, Oxford: Oxford University Press, second edn.

Helske, J. (2017) 'KFAS: Exponential family state space models in R', *Journal of Statistical Software* 78(10): 1–39. URL https://doi.org/10.18637/jss.v078.i10.

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

de Jong, P. and S. Chu-Chun-Lin (2003) 'Smoothing with an unknown initial condition', *Journal of Time Series Analysis* 24(2): 141–148.