Kalman Filter: Sample Gauss Programs

R.G. Pierse

In representing examples of *GAUSS* code in these notes, the following conventions are adopted: all variables are in *italic* typeface and all *GAUSS* reserved words are in **bold** typeface. Roman typeface is used to represent strings and the names of procedures other than *GAUSS* intrinsic functions. The following procedures are written to be called using the *GAUSS* optimisation routine **optmum** to *minimise* minus the appropriate log-likelihood function.

1. Kalman Filter

The procedure **kalman** computes (minus) the log-likelihood function for the Kalman filter problem with *measurement equation* given by

$$\mathbf{y}_t = \mathbf{Z}_t\boldsymbol{\alpha}_t + \mathbf{X}_t\mathbf{d} + \boldsymbol{\varepsilon}_t$$

and transition equation

$$\boldsymbol{\alpha}_t = \mathbf{T}\boldsymbol{\alpha}_{t-1} + \mathbf{c} + \mathbf{R}\boldsymbol{\eta}_t$$

with

$$\text{var}(\boldsymbol{\varepsilon}_t) = \sigma^2\mathbf{H} \quad \text{and} \quad \text{var}(\boldsymbol{\eta}_t) = \sigma^2\mathbf{Q}$$

where $\mathbf{y}_t$ is an $n \times 1$ vector of observable variables, $\mathbf{Z}_t$ is an $n \times m$ matrix of variables, and $\mathbf{X}_t$ is an $n \times k$ matrix of exogenous variables, $\boldsymbol{\alpha}_t$ is an $m \times 1$ vector of state variables, $\mathbf{d}$ is a $k \times 1$ vector of coefficients on the exogenous variables and $\boldsymbol{\varepsilon}_t$ is an observational error with variance $\mathbf{H}$ a known $n \times n$ matrix and σ^2 a scaling factor. $\mathbf{T}$ is an $m \times m$ matrix, $\mathbf{c}$ is an $m \times 1$ vector, $\mathbf{R}$ is an $m \times g$ matrix and $\boldsymbol{\eta}_t$ is a $g \times 1$ vector of serially uncorrelated disturbances where $\mathbf{Q}$ is a known $g \times g$ matrix.

The user needs to specify a procedure **state** which takes as argument the vector of hyperparameters $\boldsymbol{\theta}$ and returns the six state matrices T, R, c, Q, d, H, the $T \times n$ data matrices $Y = (\mathbf{y}_1, \cdots, \mathbf{y}_T)'$, and the matrices Z and X. The

matrix Z may be time invariant in which case it should be of dimension $n \times m$. If however, $\mathbf{Z}_t$ varies with time then the matrix Z should be defined as

$$Z = [\mathbf{Z}_1 : \cdots : \mathbf{Z}_T]$$

and will be of dimension $n \times mT$. Similarly, if X is time invariant it should be of dimension $n \times k$, but if it varies with time then it should be defined as

$$X = [\mathbf{X}_1 : \cdots : \mathbf{X}_T]$$

and will be of dimension $n \times kT$.

The variance scale factor *s2* is a global variable and should be initialised before the procedure **kalman** is called. On return it will contain the maximum likelihood estimate $\widetilde{\sigma}^2$. The first *n1* observations are ignored in calculating the likelihood. There is a trap to set a penalty to the likelihood function if parameter values are encountered such that the prediction error variance $\mathbf{F}_t$ is singular. A diffuse prior is assumed for the initial value of the estimated variance of the Kalman filter predictor $\mathbf{P}_0$. If the problem is time invariant, then $\mathbf{P}_0$ can alternatively be initialised by the ergodic formula defined in procedure **pzero**().

proc(1) = kalman(*theta*);
/* (c) Richard G. Pierse 1997 */
local *nt*, *m*, *n*, *k*, *T*, *R*, *c*, *d*, *Q*, *H*, *Y*, *Z*, *X*, *a*, *P*, *i*, *j*, *vt*, *Ft*, *yt*, *Zt*, *Xt*,
f, *ff*, *nobs*, *llf*, *zinvar*, *xinvar*, *big*, *n1*;
/* call user routine to set up state space system matrices */
{*T*, *R*, *c*, *Q*, *d*, *H*, *Y*, *Z*, *X*} = state(*theta*);
m = **rows**(*T*); *nt* = **rows**(*Y*); *n* = **cols**(*Y*); *k* = **cols**(*X*);
/* *zinvar* = 1 if *Zi* is time invariant */
zinvar = 0; **if cols**(*Z*) / *m* eq 1; *zinvar* = 1; **endif**;
/* *xinvar* = 1 if *Xi* is time invariant */
xinvar = 0; **if cols**(*X*) / *k* eq 1; *xinvar* = 1; **endif**;
a = **zeros**(*m*, 1); *big* = 0.9e7;
/* *big* scales initial *P* to approximate a diffuse prior */
P = **diagrv**(**eye**(*m*), *big****ones**(*m*, 1));

```
/* Cumulate log-likelihood function */
s2 = 0; f = 0;
R = R*Q*R';
n1 = 2; /* n1 is number of initial observations to ignore */
i = 1; do while i <= nt; /* loop over observations */
    jz = i; if zinvar eq 1; jz = 1; endif;
    jx = i; if xinvar eq 1; jx = 1; endif;
    /* Prediction equations */
    a = T * a + c;
    P = T*P*T' + R;
    /* Updating equations */
    Zt = Z[ ·, (jz-1)*m+1 : jz*m]; /* n × m */
    Xt = X[ ·, (jx-1)*k+1 : jx*k]; /* n × k */
    yt = Y[i, ·]'; vt = yt-Zt*a-Xt*d;
    Ft = inv(Zt*P*Zt'+H); ff = detl;
     if ff > 1.e-8;
        a = a+P*Zt'Ft*vt;
        P = P-P*Zt'Ft*Zt*P;
        if i > n1; /* skip 1st n1 observations */
            s2 = s2+vt'Ft*vt;
            f = f+ln(ff);
        endif;
    else;  /* Add penalty to likelihood function */
        f = f+1000; s2 = s2+100;
    endif;
i = i+1; endo;
nobs = nt-n1;
s2 = s2 / (nobs*n);
```

llf = − *nobs***n**(**ln**(2***pi**)+1)/2 − (*f* / 2)-*nobs***n****ln**(*s2*)/2;

retp(−*llf*);

endp;

proc(1)=pzero(*T*, *R*, *Q*);

/* Defines initial values for stationary P matrix

vec($\mathbf{P}_0$) = $(\mathbf{I} - \mathbf{T} \otimes \mathbf{T})^{-1}$ vec($\mathbf{RQR}'$). */

local *m*, *P*;

m = **rows**(*T*);

p = **inv**(**eye**(*m***m*)−(*T* ·*· *T*)) * **vec**(*R***Q***R*′); *P* = **reshape**(*p*,*m*,*m*)′;

retp(*P*);

endp;

The procedure **smooth** calculates smoothed estimators for the Kalman filter problem. It returns four matrices *as*, *af*, *Ps*, *Ys*. *as* is the $m \times T$ matrix of smoothed estimates $(\mathbf{a}_{1|T}, \mathbf{a}_{2|T}, \cdots, \mathbf{a}_{T|T})$, *af* is the $m \times T$ matrix of filtered estimates $(\mathbf{a}_{1|1}, \mathbf{a}_{2|2}, \cdots, \mathbf{a}_{T|T})$, *Ps* is the $m^2 \times T$ matrix of smoothed estimates $(\text{vec}(\mathbf{P}_{1|T}), \text{vec}(\mathbf{P}_{2|T}), \cdots, \text{vec}(\mathbf{P}_{T|T}))$ and *Ys* is the $T \times n$ matrix of smoothed predictors $(\widehat{\mathbf{y}}_1, \cdots, \widehat{\mathbf{y}}_T)'$ defined by

$$\widehat{\mathbf{y}}_t = \mathbf{Z}_t \mathbf{a}_{t|T} + \mathbf{X}_t \mathbf{d}\,.$$

proc(4) = smooth(*theta*); /* fixed interval smoothing algorithm */

/* (c) Richard G. Pierse 1997 */

local *n*, *m*, *nt*, *a*, *T*, *R*, *c*, *Q*, *d*, *H*, *Y*, *Z*, *i*, *j*, *vt*, *ft*, *yt*, *Ys*, *Zt*,

af, *as*, *P*, *Ps*, *Pp*, *Pstar*, *invar*, *big*;

/* set up state space system matrices */

{*T*, *R*, *c*, *Q*, *d*, *H*, *Y*, *Z*} = state(*theta*);

m = **rows**(*T*); *nt* = **rows**(*Y*); *n* = **cols**(*Y*);

/* *invar* = 1 if *Zi* is time invariant */

invar = 0; **if cols**(*Z*) / *m* eq 1; *invar* = 1; **endif**;

/* Pass once through Kalman filter storing state matrices *as* & *Ps* */

af = **zeros**(m, nt); as = af;

Ps = $zeros$(m*m, nt); ys = **zeros**(nt, n);

a = **zeros**(m, 1); big = 0.9e7;

/* big scales initial P to approximate a diffuse prior */

P = **diagrv**(**eye**(m), big * **ones**(m, 1));

R=R*Q*R';

i = 1; **do while** i <= nt;

    j = i; **if** $invar$ eq 1; j = 1; **endif**;

    /* Prediction equations */

    a = T * a + c;

    P = T * P * T' + R;

    af[·, i] = a; Ps[·, i] = **vec**(P);

    /* Updating equations */

    Zt = Z[·, (j-1)*m+1 : j*m]; /* $n \times m$ */

    yt = Y[i, ·]'; vt = yt−Zt*a−d;

    Ft = **inv**(Zt*P*Zt'+H);

    a = a+P*$Zt'Ft$*vt;

    P = P−P*$Zt'Ft$*Zt*P;

    af[·, i] = a; Ps[·, i] = **vec**(P);

i = i+1; **endo**;

as = af;

/* Now execute smoothing recursion from nt−1,...,1 */

i = nt−1; **do while** i >= 1;

    P = **reshape**(Ps[·,i], m, m)'; Pp = T*P*T'+R;

    $Pstar$ = P*T' **invpd**(Pp);

    as[·, i] = as[·, i]+$Pstar$*(as[·, i+1]−T*as[·, i]−c);

    Ps[·, i] = **vec**(P+$Pstar$*(**reshape**(Ps[·,i+1], m, m)'−Pp)*$Pstar'$);

i = i−1; **endo**;

/* Finally compute smoothed predictor *Ys* */
i = 1; **do while** *i* <= *nt*;
    j = *i*; **if** *invar* eq 1; *j* = 1; **endif**;
    Zt = *Z*[·, (*j*−1)**m*+1 : *j***m*]; /* *n* × *m* */
    Ys[*i*, ·] = (*Zt***as*[·, *i*] + *Xt***d*)′;
i = *i*+1; **endo**;
retp(*as*, *af*, *Ps*, *Ys*);
endp;

2. Hamilton Filter

The procedure **hamilton** computes the log-likelihood for the Hamilton regime switching model defined by

$$y_t = \mathbf{x}_t'\boldsymbol{\beta}_{s_t} + \varepsilon_t \quad , \quad \text{var}(\varepsilon_t) = \sigma^2_{s_t}$$

where y_t is a scalar variable which is a linear function of the $k \times 1$ vector of variables $\mathbf{x}_t$ with coefficients that depend on the state s_t in period t. There are n states, with s_t taking one of the n possible values, $1, \cdots, n$. Defining $\boldsymbol{\alpha}_t$ as the $n \times 1$ vector with ith element equal to one when $s_t = i$ and all other elements equal to zero, this model can be rewritten as

$$y_t = \mathbf{x}_t'\mathbf{B}\boldsymbol{\alpha}_t + \varepsilon_t$$

where $\mathbf{B} = [\boldsymbol{\beta}_1 : \cdots : \boldsymbol{\beta}_n]$ is of dimension $k \times n$.

The state vector $\boldsymbol{\alpha}_t$ follows the first order Markov process

$$\boldsymbol{\alpha}_t = \boldsymbol{\Pi}\boldsymbol{\alpha}_{t-1} + \boldsymbol{\eta}_t$$

where $\boldsymbol{\eta}_t$ is a disturbance uncorrelated with $\boldsymbol{\alpha}_{t-1}$ or ε_t, and $\boldsymbol{\Pi}$ is the matrix of transition probabilities satisfying the condition that that the columns sum to one, so that there are only $n(n-1)$ free parameters in $\boldsymbol{\Pi}$.

The parameters to be estimated are the nk coefficients $\mathbf{B}$, the $n(n-1)$ transition probabilities in $\boldsymbol{\Pi}$ and the n variances σ_i^2. These are collected in the vector $\boldsymbol{\theta}$ that is passed to **hamilton**. The filter defines $\mathbf{a}_t$, which is the best estimator of $\boldsymbol{\alpha}_t$, given the information set available at time t.

```
proc(1) = hamilton(theta);
/* (c) Richard G. Pierse 1997 */
local Beta, P, s2, y, X, ns, nt, u, v, a, llf, t, fys, fy;
{Beta, P, s2, y, X} = state(theta);
ns = cols(Beta); nt = rows(y);
u = y*ones(1, ns)-X*Beta; /* nt × ns */
v = 1/sqrt(2*pi*s2') .* exp(-(u^2) ./ (2*s2')); /* nt × ns */
/* Initialise a from ergodic probabilities */
A = (eye(ns)-P) | ones(1, ns);
A = invpd(A'A)*A'; a = A[ ., ns+1];
llf = 0;
t = 1; do while t <= nt; /* loop over observations */
    fys = v[t, .]' .* a; /* ns × 1 */
    fy = v[t, .] * a;
    if fy > 1.e-8;
        llf = llf + ln(fy); /* cumulate llf */
        a = P * fys / fy; /* update a */
    else; llf = llf-1000; /* penalty to likelihood fn */
    endif;
t = t+1; endo;
retp(-llf);
endp;
```

The procedure **hsmooth** calculates smoothed estimators for the Hamilton filter problem. It returns two matrices *as*, and *af*. *as* is the $n \times T$ matrix of smoothed estimates $(\mathbf{a}_{1|T}, \mathbf{a}_{2|T}, \cdots, \mathbf{a}_{T|T})$, *af* is the $n \times T$ matrix of filtered estimates $(\mathbf{a}_{1|1}, \mathbf{a}_{2|2}, \cdots, \mathbf{a}_{T|T})$.

```
proc(2) = hsmooth(theta); /* smoothing algorithm */
/* (c) Richard G. Pierse 1997 */
```

local *Beta*, *P*, *s2*, *y*, *X*, *ns*, *nt*, *u*, *v*, *a*, *af*, *as*, *t*, *fys*, *fy*;
{*Beta*, *P*, *s2*, *y*, *X*} = state(*theta*);
ns = **cols**(*Beta*); *nt* = **rows**(*y*);
u = *y****ones**(1, *ns*)−*X***Beta*; /* *nt* × *ns* */
v = 1/**sqrt**(2***pi****s2*′) ·* **exp**(−(*u*^2) ·/ (2**s2*′)); /* *nt* × *ns* */
/* Initialise *a* from ergodic probabilities */
A = (**eye**(*ns*)−*P*) | **ones**(1, *ns*);
A = **invpd**(*A*′*A*) * *A*′; *a* = *A*[·, *ns*+1];
af = **zeros**(*ns*, *nt*); *as* = *af*;
/* Pass once through filter saving *a* */
t = 1; **do while** *t* <= *nt*;
    fys = *v*[*t*, ·]′ ·* *a*;
    fy = *v*[*t*, ·] * *a* ;
    af[·, *t*] = *fys* / *fy*; /* save *a* */
t = *t* + 1; **endo**;
as[·, *nt*] = *af*[·, *nt*];
/* Now execute smoothing recursion from *nt*−1,...,1 */
t = *nt*−1; **do while** *t* >= 1;
    as[·, *t*]=*af*[·, *t*] ·* *P*′(*as*[·, *t*+1] ·/ (*P* * *af*[·, *t*]));
t = *t*-1; **endo**;
retp(*as*, *af*);
endp;

The user needs to specify a procedure **state** which takes as argument the vector of unrestricted parameters $\boldsymbol{\theta}$ and returns the parameter matrices B, and P, the vector of variances *s2*, the $T \times 1$ data vector y and the $T \times k$ matrix X. Note that the matrix P must satisfy the conditions that

$$0 \leq P_{ij} \leq 1 \quad \text{and} \quad \sum_{i=1}^{n} P_{ij} = 1 \quad , \quad \forall j \, .$$

state imposes these restrictions by transforming the relevant $n * (n-1)$ unrestricted elements of *theta*. Similarly, the restriction that the variances *s2* cannot be negative is imposed by squaring the relevant *theta* parameters. Note that this procedure could easily be adapted to impose zero restrictions on some of the elements of B, and P with the length of the vector of unrestricted parameters *theta* being reduced accordingly.

This procedure treats the data matrices y and X and the parameters k (the number of columns in X) and ns (the number of states) as *global variables* that have been defined before the optimisation is started.

proc(5) = state(*theta*);
/* define state matrices from theta ordered *Beta*, *P*, *s2* */
/* *y*, *X*, *k*, *ns* are all global variables */
local *Beta*, *P*, *s2*, *n1*, *n2*;
n1 = 1; *n2* = *k* * *ns*;
Beta = **reshape**(*theta*[*n1* : *n2*], *k*, *ns*);
P = **zeros**(*ns*, *ns*);
n1 = *n2*+1; *n2* = *n2*+*ns**(*ns*−1);
P[1 : *ns*−1, ·] = **reshape**(*theta*[*n1* : *n2*], *ns*−1, *ns*);
P[*ns*, ·] = **ones**(1, *ns*);
/* Transform *P* to lie in interval [0,1] */
P = *P*^2; *P* = *P* ·/ (**sumc**(*P*)′);
n1 = *n2*+*1*; *n2* = *n2*+*ns*;
/* Square *s2* to ensure *s2* ≥ 0 */
s2 = *theta*[*n1* : *n2*]^2;
retp(*Beta*, *P*, *s2*, *y*, *X*);
endp;