Robust Kalman Filtering

Peter Ruckdeschel

As already pointed out in Härdle, Klinke, and Müller (2000, Chapter 10), state-space models are very useful and flexible in the sense that various recursive methods for time-dependent situations can be formulated as general solutions of filtering, smoothing and prediction problems in state-space models.

The classically optimal solutions to these problems, the Kalman-type filters, smoothers and predictors are all based on second moments of the underlying distributions, however. This is clearly a robustness problem, i.e. small deviations from the model assumptions will cause large effects on the quality of the filter/smoother/predictor. As a way out for the filtering problem—at least for one type of "danger"—we suggest instead using robust Kalman filtering methods such as described in this tutorial.

All routines for robust Kalman filtering, which will be explained in the following, are part of the `kalman` quantlib. Notice that the quantlets for Kalman filtering explained in Härdle, Klinke, and Müller (2000, Chapter 10) are part of quantlib `times`.

1 State-Space Models and Outliers

`{X, Y} = kemitor2(T, x0, H, F, ErrY, ErrX)`
simulates observations and states of a time-invariant state-space model

For the definition and notation of a state-space model we refer to Härdle, Klinke, and Müller (2000, Section 10.1); we, too, will confine ourselves to discrete time; in particular we make the same assumptions on the vectors v_t and w_t and the initial state x_0 given in equations (2) and (3) there.

For our purposes, however, also the state is interesting, so we modified the quantlet `kemitor` by just adding an extra output `X` for it. This means that whenever you have used the command line

```
y = kemitor(T, x0, H, F, ey, ex)
```

you may as well use

```
erg = kemitor2(T, x0, H, F, ey, ex)
y=erg.Y
```

and additionally, you get the simulated states as a $T \times n$ matrix `x`, using

```
x=erg.X
```

With these slight modifications, the examples to `kemitor` stay valid. An example also using the state-simulations will be presented in the Subsection 1.2.

1.1 Outliers and Robustness Problems

```
{X, Ind} = epscontnorm(T, eps, mid, Cid, mcont, Ccont,
                                                    DirNorm)
```

simulates observations from an ε-contaminated multivariate normal distribution

A complication of the state-space model and the classical Kalman filter enters if we allow for deviations from the model assumptions. In the i.i.d. situation, this led to the development of robustness theory, for which we refer the reader e.g. to Huber (1981), Hampel et al. (1986) , Rieder (1994).

A common way to model this deviation are ε-contaminations used by the famous Huber (1964). The considered variable X no longer always comes from a fixed distribution P^{ideal}, but rather from a neighborhood around this fixed central/ideal distribution P^{ideal}, and $X \sim P^{\text{real}}$,

$$P^{\text{real}} = (1-\varepsilon)P^{\text{ideal}} + \varepsilon K \tag{1}$$

for some arbitrary distribution K.

In our context we will consider only multivariate normal distributions as central distributions, i.e. $P^{\text{ideal}} = \mathcal{N}_2(\mu_{\text{id}}, C_{\text{id}})$. As contaminating distributions we allow for Dirac-distributions, symmetric Dirac-distributions and again normal. This is done using the quantlet `epscontnorm`.

1.1.1 Example 1

How it works may be seen by the following example where we generate 500 observations from

$$0.9N_2\left(\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}2&1\\1&1\end{pmatrix}\right)+0.1N_2\left(\begin{pmatrix}3\\3\end{pmatrix},\begin{pmatrix}3&0\\0&0.2\end{pmatrix}\right)$$

and then plot them, in green those coming from the ideal, in red those from the contaminating distribution, see Figure 1.

```
library("xplore")
library("plot")
library("kalman")
randomize(0)
T = 500
eps = 0.1
mid=#(0,0)
Cid = #(2,1)~#(1,1)
mcont=#(3,3)
Ccont = #(3,0)~#(0,0.2)
erg=epscontnorm(T,eps,mid,Cid,mcont,Ccont,0)
color=2*erg.Ind+2
                    ; sets color to 2 (green) for "clean" data
                    ; and 4 (red) for contaminated data
data=erg.X
setmaskp(data,color, 3, 8)
disp = createdisplay(1,1)
show(disp,1,1,data)
```

rkalm01.xpl

In the situation of state-space models deviations can have quite different effects, depending on where they enter. In the time-series context there is a common

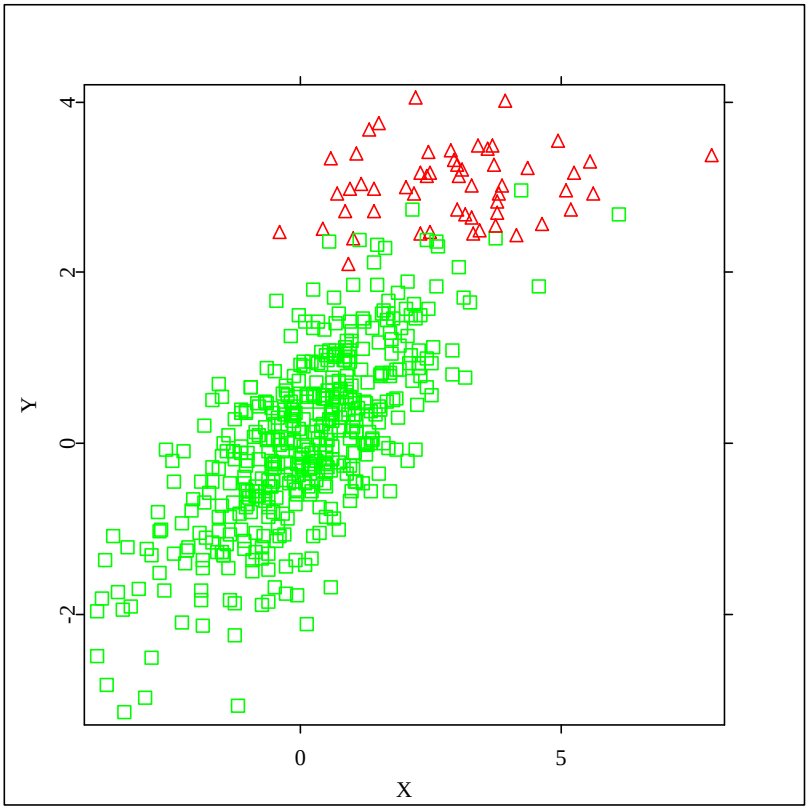

Figure 1: 500 observations from ε-contaminated Normal distribution; ideal observations are green squares, contaminated observations are red triangles.

terminology due to Fox (1972) distinguishing between **additive outliers** (AO) and **innovation outliers** (IO).

1.1.2 Additive Outliers

AO's enter in the observation-equation $y_t = Hx_t + v_t$, that is v_t is contaminated. As a consequence, singular observations will be erroneously large, but note that only a single observation is affected, as the error does not enter the state of the model.

An example for this can be seen in satellite navigation. The state of the system is the 3 dimensions of the satellite in space, whereas ground control receives a possibly noisy signal about this state. An AO could now be caused by a short defect in the observation device.

So the task of the estimator is to down-weight the influence of large obesrvations for the state estimation. This is what our robust Kalman filters rLS and rIC are designed for.

1.1.3 Innovation Outliers

Contrary to AO's, IO's enter in the state-equation $x_t = Fx_{t-1} + w_t$, where w_t is contaminated. Here an outlier will have effect also for the subsequent states and hence for the observations, too.

To return to our example from the AO's, an IO could be caused by an asteroid hitting the the satellite and thereby deviating it from its "should-be" track.

This time the task of the estimator is to realize this and to adapt itself as fast as possible to the new situation. Of course, down-weighting the observations' influence will make detecting state deviations even harder, so simultaneous treatment of AO's and IO's will in general be less effective than either of the two alone. Additionally, for detecting a deviation from the "should-be" track, a single observation plus the estimation of the state based on the observations up to then is generally not sufficient. So for this task, it is better to drop strict recursivity in the sense that in stead of just one observation plus the estimation of the state based on the observations up to then one should rather base the estimation on the last p observations and smoothed/filtered/predicted values $x_{t-i|t}$, $i = -1, ..., p-2$. Simultaneous robust smoothers/filters designed for that purpose have not yet been implementated into XploRe up to now.

1.1.4 Other Types of Outliers

This distinction between AO and IO is by no means the only possible one—other types have been considered such as **patchy outliers** (PO) and **substitutive outliers** (SO) which will be mentioned later in this chapter.

1.2 Examples of AO's and IO's in XploRe

To give you an impression of how AO's and IO's affect data in state space models, we have simulated data from 10%-AO/IO-contaminated, normal setups.

1.2.1 Example 2

We realize AO-contamination by simulating v_t coming from an ε-contaminated $N_m(0, Q)$. An example of the effects of an AO-contaminated model is generated by the following XploRe instructions using the quantlets `epscontnorm` and `kemitor2` which generate a simulation of length 100 of a steady state model (i.e. $H = R = Q = F = 1$) under a convex contaminated v_t, with contamination-radius 0.1 and $K = N(10, 0.1)$;

First we set the system parameters:

```
library("xplore")
library("plot")
library("kalman")
randomize(0)
T = 100
mu=0
H = 1
F = 1
mid=0
Cid=1
mcont=10
Ccont=0.1
eps=0.1
```

Then we simulate data from this situation and apply the Kalman filter to this data.

```
ErrX = normal(T)
ErrY = epscontnorm(T,eps,mid,Cid,mcont,Ccont,0)
sim = kemitor2(T,mu,H,F,ErrY,ErrX)

state = (1:100)~(sim.X)
obs= (1:100)~(sim.Y)
ind = ErrY.Ind
```

Flags are set for the instances where we have contamination.

```
ind=(1:100)~ind
ind=paf(ind, ind[,2]==1)
```

Finally we plot the path of the state (blue) and the corresponding observations (green) and set red flags at the instances where we have contamination, see Figure 2.

```
setmaskp(ind,4, 3, 8)
state=setmask(state,"line","blue","thin")
obs=setmask(obs,"line","green","thin")
disp = createdisplay(1,1)
show(disp,1,1,state,obs,ind)*0
setgopt(disp,1,1, "title", "1-dim Steady State Model under A0")
setgopt(disp,1,1, "xlabel", "t")
setgopt(disp,1,1, "ylabel", "x, y")
```


rkalm02.xpl

1.2.2 Example 3

IO-contamination is realized by simulating w_t coming from an ε-contaminated $N_n(0, R)$. The effects of an IO-contaminated model may be seen by just interchanging the role of w_t and v_t in the first example. They will generate a simulation of length 100 of a steady state model (i.e. $H = R = Q = F = 1$) under a convex contaminated w_t, with radius 0.1, and $K = N(10, 0.1)$.

```
ErrY = normal(T)
ErrX = epscontnorm(T,eps,mid,Cid,mcont,Ccont,0)
```

```
sim = kemitor2(T,mu,H,F,ErrY,ErrX)

state = (1:100)~(sim.X)
obs= (1:100)~(sim.Y)
ind = ErrX.Ind
```

rkalm03.xpl

1.3 Problem Setup

To summarize we want to solve the following problem. In a given ideal normal state-space model, that is $x_0 \sim N_n(\mu, \Sigma)$, $v_t \sim N_m(0, Q)$, $w_t \sim N_n(0, R)$ all stochastically independent, with F, H, Q, R known and in "correct" dimensions, we want to find recursive estimates for x_t based on y_t and a preliminary estimate $x_{t|t-1}$ for x_t based on all observations y_{t-i}, $i = 1, \ldots, t-1$. The quality of this estimator is measured in terms of the mean squared error (MSE) $\mathrm{E}[|x_t - f(y_t, x_{t|t-1})|^2]$. For robustness reasons we want the influence of y_t on $f(y_t, x_{t|t-1})$ to be bounded in order to protect it from AO outliers.

Of course for this robustness, we pay a price, namely we cannot in general achieve the optimal MSE which is attained by the classical Kalman filter $x^0_{t|t}$. This price will be measured by the relative efficiency loss $[\mathrm{MSE}(f) - \mathrm{MSE}(x^0_{t|t})]/\mathrm{MSE}(x^0_{t|t})$.

2 Classical Method: Kalman Filter

`{filtX, KG, PreviousPs} = kfilter2(y, mu, Sig, H, F, Q, R)`
calculates the classical Kalman filter $x_{t|t}$ for a (multivariate) time series

For the definition and notation of the Kalman filter we refer to Härdle, Klinke, and Müller (2000, Section 10.2). As for our purposes, the state is more interesting, we have modified the quantlet `kfilter` by changing the output.

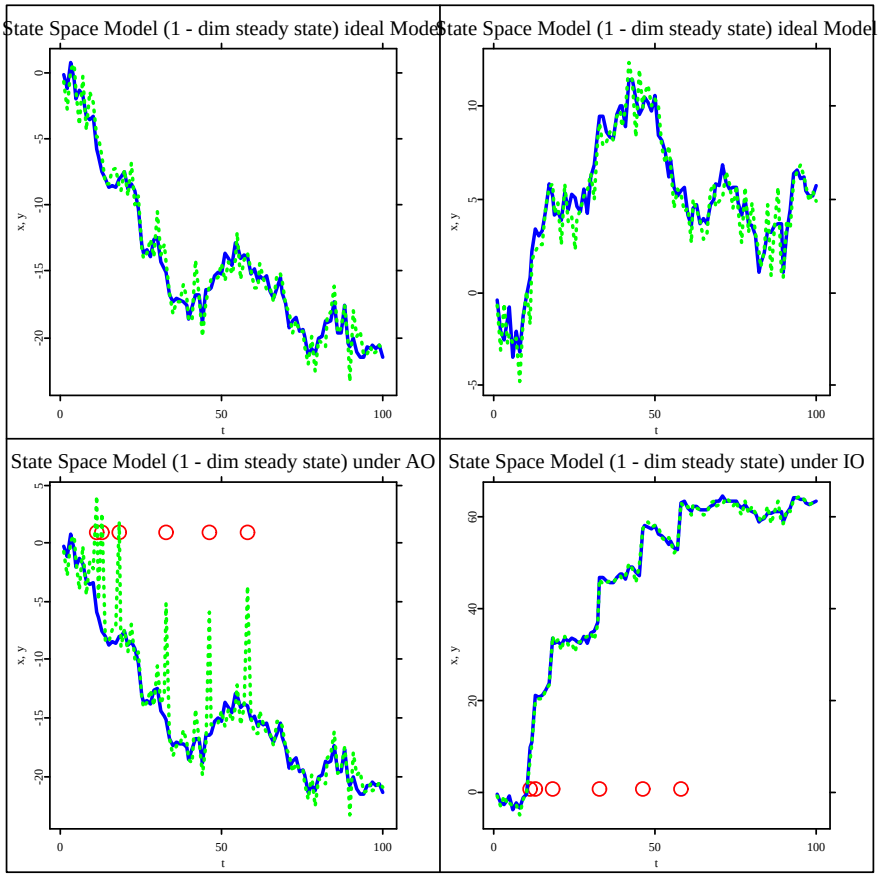

Figure 2: Examples 2 ,3 displayed simultaneously: AO's cause single peaks, IO's result in a level change

2.1 Features of the Classical Kalman Filter

At this point, we only recall the features that are—in our point of view—central for the Kalman filter:

- an easy, understandable structure with an initialization step, a prediction step and a correction step,
- the correction step is an easily evaluable function—it is linear !
- all information from the past useful for the future is captured in the value of $x_{t|t-1}$.
- the correction step is linear and thus not robust, as y enters unbounded;

These features—except for the last one—are to be preserved in our filtering procedures.

2.2 Optimality of the Kalman Filter

The classical Kalman filter is characterized by an optimality property that coincides with various different notions of optimality in the case of a normal state-space model.

best linear filter:
: The Kalman filter is obtained as the linear filter minimizing the mean squared error (MSE). This is shown with Hilbert space theory, considering the closed linear spaces generated by the observations $\bar{y}_t := (\mu, y_1, \ldots, y_t)^T$ and orthogonal decompositions as follows:

$$\begin{aligned} \mathrm{lin}(\bar{y}_t) &= \mathrm{lin}(\bar{y}_{t-1}) \oplus \mathrm{lin}(\Delta y_t), \quad \Delta y_t = y_t - H x_{t|t-1} && (2) \\ x_{t|t} &= \mathrm{oP}(x_t|\bar{y}_{t-1}) + \mathrm{oP}(x_t|\Delta y_t) \\ &= x_{t|t-1} + \mathrm{oP}(\Delta x_t|\Delta y_t), \quad \Delta x_t = x_t - x_{t|t-1}, && (3) \end{aligned}$$

: where $\mathrm{oP}(\cdot|Z)$ denotes the orthogonal projection onto the closed linear space generated by Z and Δy_t is called the innovation induced by y_t. This decomposition will be the basis for the quantlet `rlsfil`.

conditional expectation (under normality assumptions):
: A very nice property is that under normality, classical Kalman filter and conditional expectation $x_{t|t} = \mathrm{E}[x_t|\bar{y}_t]$ coincide, so that $x_{t|t}$ is not only optimal among all linear filters based on $\bar{y}_t$, but among all $\bar{y}_t$-measurable filters.

posterior mode (under normality):
: Again under normality, $x_{t|t}$ coincides with the posterior mode of $\mathcal{L}(x_t|\bar{y}_t)$, which is the basis for robustifications done by Fahrmeir and Künstler (1999).

ML-Estimator in a Regression Model (under normality):
: Finally, under normality the Kalman filter is also the maximum likelihood estimator (MLE) for a certain regression model with random parameter which is the basis for the `rICfil`.

3 Robustifying Least Squares: the rLS filter

```
{filtX, KG, PreviousPs, clipInd}
                = rlsfil(y, mu, Sig, H, F, Q, R, bs, b)
```
calculates the rLS filter for a (multivariate) time series

```
{b, ctrl} = calibrLS(T, Sig, H, F, Q, R, e, N,
                                              eps, itmax, aus)
```
calibrates the rLS filter to a given efficiency loss in the ideal model.

3.1 Derivation

Instead of $M\Delta y$ we use a huberized version of it, that is

$$H_b(M\Delta y) = M\Delta y \min\{1, \frac{b}{\|M\Delta y\|}\}, \tag{4}$$

so that the correction is just as in the Kalman filter if it is not too large in absolute value, and if it is too large, the direction remains unchanged and it is projected to the ball with radius b in n dimensions.

This gives us

$$\begin{aligned} x_{0|0} &= \mu, & (5)\\ x_{t|t-1} &= Fx_{t-1|t-1} & (6)\\ x_{t|t} &= x_{t|t-1} + H_b(\hat{M}_t(y_t - Hx_{t|t-1})), & (7) \end{aligned}$$

where we have not yet specified how to choose M and b, but (c.f. Ruckdeschel (1999))

- under strict normality the optimal M is $\hat{M}$ (Kalman gain);
- under strict normality rLS is SO-optimal; SO stands for *s*ubstitutive outlier, and means that instead of corrupting v_t, contamination effects y_t directly, replacing it by an arbitrarily distributed variable $\tilde{y}_t$ with some low probability.
- strict normality of all $x_{t|t}, x_{t|t-1}, \Delta x_t$ gets lost during the history of $x_{t|t}$ for growing t;
- Δx_t is "nearly" normal and $\hat{M}$ cannot be improved significantly;

We already note at this point that the rLS has preserved the crucial features from of the Kalman filter

- an easy, understandable structure with an initialization step, a prediction step and a correction step,
- the correction step is an easily evaluable function—it is "almost linear" !
- all information from the past useful for the future is captured in the value of $x_{t|t-1}$.
- and: the correction step is now bounded in influence, as Δy enters bounded;

So in the implementation, we use $\hat{M}$.

3.2 Calibration

As to the choice of b we propose an assurance criterion: *How much efficiency in the ideal model relative to the optimal procedure* ,i.e. the Kalman filter, *am I ready to pay in order to get robustness under deviations from the ideal model?* This efficiency loss is quantified as the relative degradation in MSE in the ideal model one obtains if one uses robust versions instead of the classical Kalman filter.

Of course the lower the clipping b, the higher the relative efficiency loss, so due to this monoticity we may solve

$$\mathrm{E}\,\|\Delta x - H_b(\hat{M}\Delta y)\|^2 \stackrel{!}{=} (1+\delta)\,\mathrm{tr}\,\Sigma_{t|t} \tag{8}$$

in b, uniquely for a given efficiency loss δ.

We note here that

- Calibration, that is finding b to a given δ, can be done beforehand. So this relatively time-expansive step is not really a hindrance to the application.
- Assumptions in our implementation:
  - We are assuming strict normality, which is not quite the case, but using exact terms makes things more complicated and do not really improve the quality, c.f. Ruckdeschel (1999)
  - for $n = 1$ we solve (8) using analytic terms,
  - for $n > 1$ we use MC-Simulation.
- In most time-invariant situations, the sequence of $\Sigma_{t|t-1}$ (and hence also $\Sigma_{t|t-1}$) stabilizes due to asymptotic stationarity. Thus, once $\Sigma_{t|t-1}$ do not change for more than `eps`, we can stop calibration and use the last calibrated b for all subsequent times t. For details as to this stabilization we refer to Anderson and Moore (1979) and Moore and Anderson (1980).

3.3 Examples

All examples stem—at least for the state space model specification—from Petr Franěk (see Härdle, Klinke, and Müller 2000, Chapter 10). They have been slightly modified to better demonstrate the features of our methods.

3.3.1 Example 5

As first example we have taken Petr Franěk's data `kalman` together with his specifications for the system matrices and by manipulating observations 50, 60, 90 entered three artificial outliers; the result is saved in `kalmanao`.

```
library("xplore")
library("plot")
library("kalman")

serie = read("kalmanao.dat")
y = serie[,2]
mu = 10
Sig = 0
H = 1
F = 1
Q = 9
R = 9
T=dim(y)

e=0.05
N=100
eps=0.01
itmax=15
aus=4
```

The rLS filter is then calibrated to an efficiency loss of 5%

```
ergLS=calibrLS(T,Sig,H,F,Q,R,e,N,eps,itmax,aus)
b=ergLS.b
```

Next we filter the data with the Kalman filter and the rLS filter.

```
res = kfilter2(y,mu,Sig, H,F,Q,R)
fx = res.filtX
res= rlsfil(y,mu,Sig, H,F,Q,R,b)
frx = res.filtX
```

The results are then displayed, the classical Kalman filter in red, the rLS filter in green and the observations from `kalmanao` in blue. Additionally we set flags on the time instances where the rLS filter clips $\hat{M}_t \Delta y_t$.

```
origy= serie[,1]~serie[,2]
origy= setmask(origy, "line", "blue", "thin")
fx = serie[,1]~fx
```

```
fx = setmask(fx, "line", "red", "thin")
frx = serie[,1]~frx
frx = setmask(frx, "line", "green", "thin")
clip=serie[,1]~(res.clipInd)
clip=paf(clip,clip[,2]==1)
clip[,2]=0
setmaskp(clip,4, 3, 4)

disp = createdisplay(1,1)
show(disp,1,1, origy,fx,frx,clip)
setgopt(disp,1,1, "title", "KalmanData1 + AO's in t=50,60,90")
setgopt(disp,1,1, "xlabel", "t")
setgopt(disp,1,1, "ylabel", "y, rLS, Kalman")
```


rkalm04.xpl

The graphical result of this example is displayed in Figure 3.

3.3.2 Example 5

As a second example we took the state-space model underlying Petr Franěk's Example 2.

```
library("xplore")
library("plot")
library("kalman")

mu = #(20,0)
Sig = #(0,0)~#(0,0)
H  = #(0.3,-0.3)~#(1,1)
F  = #(1,0)~#(1,0)
R  = #(0,0)~#(0,9)
mid=#(0,0)
Qid= #(9,0)~#(0,9)
```

According to this model we simulated 50 states and observations from a 10%-AO-contamination using quantlets `epscontnorm` and `kemitor2`. The contaminating distribution is $N_2\left(\begin{pmatrix}25\\30\end{pmatrix}, \begin{pmatrix}0.9 & 0\\0 & 0.9\end{pmatrix}\right)$.

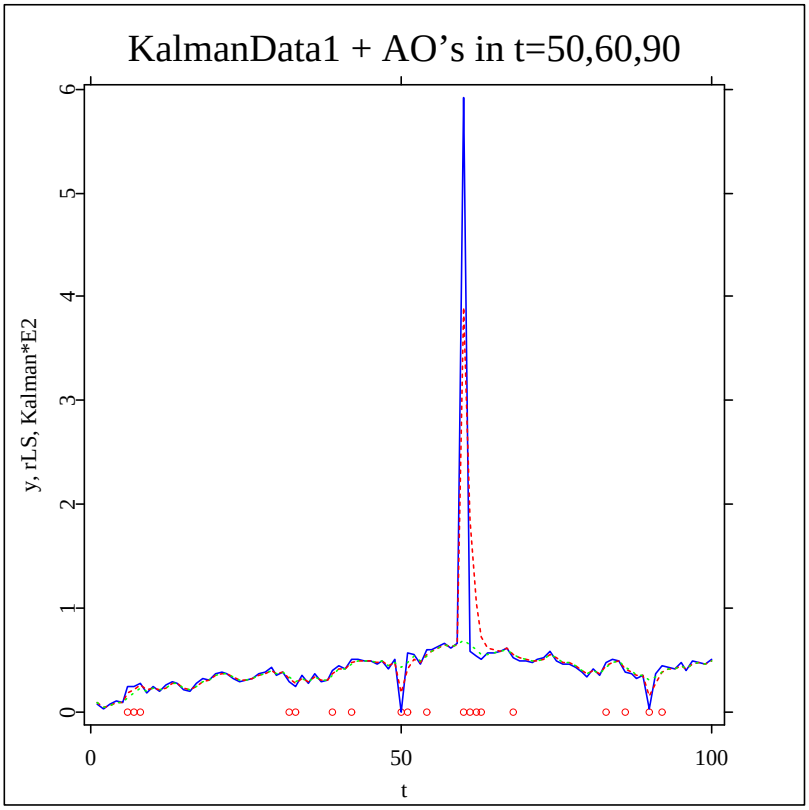

Figure 3: Actual observations $\hat{=}$ solid line, classical Kalman filter $\hat{=}$ dashed and a bit lighter line, the rLS filter $\hat{=}$ dotted line; clipped instances $\hat{=}$ circles on top of the graphic.

```
T=50
e=0.1

mcont=#(25,30)
Qcont=0.1.*Qid
AOr=0.1
```

```
randomize(0)
ErrX = epscontnorm(T,0,mid,R,mcont,Qcont,0)
ErrY = epscontnorm(T,AOr,mid,Qid,mcont,Qcont,0)
sim = kemitor2(T,mu,H,F,(ErrY.X),(ErrX.X))
y=sim.Y
Xact=sim.X
```

The rLS filter is then calibrated to an efficiency loss of 10%.

```
N=10000
eps=0.01
itmax=15
aus=4

ergLS=calibrLS(T,Sig,H,F,Qid,R,e,N,eps,itmax,aus)

b=ergLS.b
```

The simulated data are filtered by the classical Kalman filter and the rLS filter

```
res = kfilter2(y,mu,Sig, H,F,Qid,R)
fx = res.filtX
res= rlsfil(y,mu,Sig, H,F,Qid,R,b)
frx = res.filtX
```

Next, the filtered values are prepared for graphical output: the classical Kalman filter is to be displayed in red, the rLS filter in green and the actual simulated states in blue.

```
i=(1:T)
Xact1 = i~(Xact[,1])
Xact1 = setmask(Xact1, "line", "blue", "thin")
fx1 = i~(fx[,1])
fx1 = setmask(fx1, "line", "red", "thin")
frx1= i~(frx[,1])
frx1 = setmask(frx1, "line", "green", "thin")
```

Additionally we set green flags on the time instances where the rLS filter clips $\hat{M}_t \Delta y_t$ on bottom of the graphics and red ones on top where an AO-outlier occurred.

```
ym1=max(vec((Xact[,1])~(fx[,1])~(frx[,1]))) ;top of graphics
ym2=min(vec((Xact[,1])~(fx[,1])~(frx[,1]))) ;bottom of graphics
flagInd=i~(ErrY.Ind)
flagInd=paf(flagInd,flagInd[,2]==1)
flagInd[,2]=ym1*((ym1>0)*1.1+(ym1<0)*0.9)
flagclip=i~(res.clipInd)
flagclip=paf(flagclip,flagclip[,2]==1)
flagclip[,2]= ym2*((ym2<0)*1.1+(ym2>0)*0.9)
setmaskp(flagInd,4, 3, 4)
setmaskp(flagclip,2, 4, 4)
```

Finally, all the results of the 1-st coordinate are displayed in the left half of a graphic window.

```
disp = createdisplay(1,2)
show(disp,1,1,Xact1,fx1,frx1,flagInd,flagclip)
setgopt(disp,1,1, "title", "simulated Model under A0
                                                  -- 1st coord.")
setgopt(disp,1,1, "xlabel", "t")
setgopt(disp,1,1, "ylabel", "x, rLS, Kalman")
```


rkalm05.xpl

Then the same is done for the second coordinate, replacing `[,1]` by `[,2]` on all instances, and the results are then plotted on the right half of the graphic-window. All the program may be seen in
`rkalm05.xpl`.

3.3.3 Example 6

To show that robustification does not go for free, we have taken Petr Franěk's data set `kalman2` as it was, together with his specification of the state space model. So the data stems from the ideal situation and the classical Kalman filter in this case should do better than the rLS.

```
library("xplore")
library("plot")
library("kalman")

serie = read("kalman2.dat")
y = serie[,2]
```

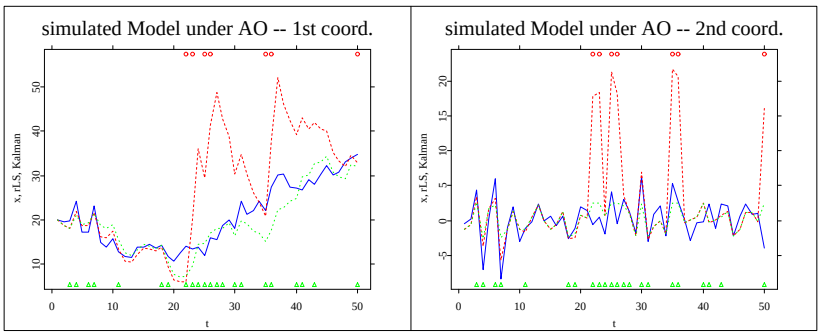

Figure 4: Simulated data according to model from Example 2: actual states $\hat{=}$ solid line, the classical Kalman filter $\hat{=}$ dashed / a bit lighter, the rLS filter $\hat{=}$ dotted line; clipped instances are marked by circles on bottom of the graphic, the AO-instances by circles on top.

```
mu = #(0,0)
Sig = #(0,0)~#(0,0)
H = #(1,0)'
F = #(0.5,1)~#(-0.3,0)
R = #(1,0)~#(0,0)
Q = 4

e=0.05
N=10000
eps=0.01
itmax=15
aus=4
```

Here the rLS filter is calibrated to an efficiency loss of 5%.

```
ergLS=calibrLS(T,Sig,H,F,Q,R,e,N,eps,itmax,aus)
b=ergLS.b

res= kfilter2(y,mu,Sig, H,F,Q,R)
fx = res.filtX
```

```
fy=(H*fx')'
res= rlsfil(y,mu,Sig, H,F,Q,R,b)
frx = res.filtX
fry=(H*frx')'
```

As we do not have any information about the actual states in this example, we examine the filtered observations $y_{t|t}$ generated by the Kalman filter and the rLS filter and compare them to the actual observations y_t. The filtered observations are simply calculated as $y_{t|t} = Hx_{t|t}$.

These filtered observations are then displayed, as usual.

```
origy = serie[,1]~serie[,2]
origy = setmask(origy, "line", "blue", "thin")

fy = serie[,1]~fy
fy = setmask(fy, "line", "red", "thin")
fry = serie[,1]~fry
fry = setmask(fry, "line", "green", "thin")
flags = serie[,1]~(res.clipInd)
flags=paf(flags,flags[,2]==1)
setmaskp(flags,4, 3, 4)

disp = createdisplay(1,1)
show(disp,1,1,origy,fy,fry,flags)
setgopt(disp,1,1, "title", "KalmanData2 in Observation Space")
setgopt(disp,1,1, "xlabel", "t")
setgopt(disp,1,1, "ylabel", "y, y-rLS, y-Kalman")
```

 rkalm06.xpl

The graphical result of this example is displayed in Figure 5.

3.4 Possible Extensions

Alternatively to using `rlsbnorm` and `rlsbnorm1` although the assumption of normality cannot be exactly met, one could use

Simulation of a bundle of paths and then MC-Integration:

One can quite easily replace the routines `rlsbnorm` and `rlsbnorm1` by

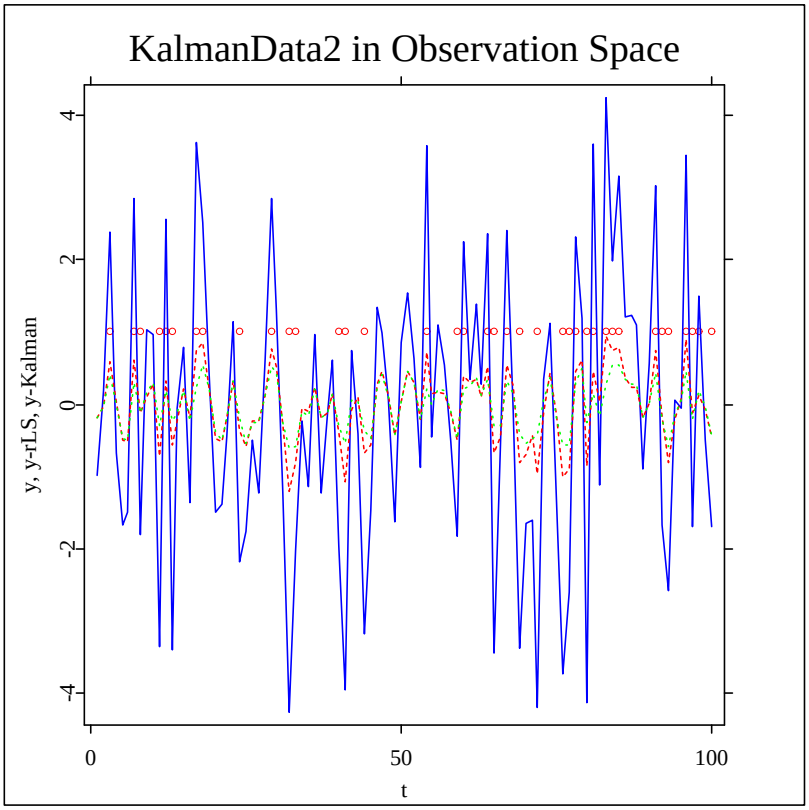

Figure 5: actual observations $\hat{=}$ solid line, the classical Kalman filter dashed and a bit lighter line, the rLS filter $\hat{=}$ dotted line; the clipped instances are marked by circles.

MC-Routines calculating $\mathrm{E}\,\|\Delta x - H_b(\hat{M}\Delta y)\|^2$ ob basis of a large sample of $(\Delta y_t, \Delta x_t)$, that is evolving according to recursions (2), simultaneously to $x_{t|t}$.

Numerical Integration:

Even keeping the normality assumption, in order to speed up the calibration step, one could try and evaluate $\mathrm{E}\,\|\Delta x - H_b(\hat{M}\Delta y)\|^2$ for low dimension n by numerical integration routines instead of using MC-integration.

4 Robustified Regression: the rIC filter

```
{filtX, KG, PreviousPs, clipInd}
      = rICfil(y, mu, Sig, H, F, Q, R, cliptyp, As, bs)
```
calculates the rIC filter for a (multivariate) time series

```
{A, b, ctrl} = calibrIC(T, Sig, H, F, Q, R,
         typ, A0, b0, e, N, eps, itmax, expl, fact0, aus)
```
calibrates the rIC filter to a given efficiency loss in the ideal model.

The idea to think of the filter problem as a "regression" problem stems from Boncelet and Dickinson (1984) and Cipra and Romera (1991), where we write "regression" because the parameter in this model is stochastic, whereas one component of the observation is deterministic, and thus this regression is not covered by the robustness theory for common regression. The robustification however then uses techniques of optimal influence curves for regression to be found in chapter VII in Rieder (1994); instead of taking M-estimators to get a robust regression estimator with a prescribed influence curve, we use a one-step procedure corresponding to a Hampel-Krasker influence curve.

4.1 Filtering Problem as a Regression Problem

The model suggested by Boncelet and Dickinson (1984) and Cipra and Romera (1991) uses the innovational representation (2) and reads

$$\begin{pmatrix} x^0_{t|t-1} \\ y_t \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n \\ H \end{pmatrix} x_t + \begin{pmatrix} \xi_t \\ v_t \end{pmatrix}, \quad \begin{pmatrix} \xi_t \\ v_t \end{pmatrix} \sim N_{n+m}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_{t|t-1} & 0 \\ 0 & Q \end{pmatrix} \right), \tag{9}$$

where $x^0_{t|t-1}$ denotes the classical Kalman filter, which in this procedure has to be calculated aside, too. As already indicated, assuming normality, the classical Kalman filter is the ML-Estimator for this model, with scores function

$$\Lambda_x \begin{pmatrix} x^0_{t|t-1} \\ y_t \end{pmatrix} = \Lambda \begin{pmatrix} x^0_{t|t-1} - x \\ y_t - Hx \end{pmatrix}$$

with

$$\Lambda \begin{pmatrix} s \\ u \end{pmatrix} := \Lambda_1 + \Lambda_2 = \Sigma^{-1}_{t|t-1} s + H' Q^{-1} u \tag{10}$$

and Fisher information

$$\mathcal{I} = \mathrm{E}[\Lambda\Lambda'] = \Sigma^{-1}_{t|t}. \tag{11}$$

4.2 Robust Regression Estimates

To get a robust estimator for model (9) we use an influence curve (IC) of Hampel-Krasker form:

$$\psi(s,u) = A\Lambda(s,u) \min\{1, \frac{b}{\|A\Lambda(s,u)\|}\} \tag{12}$$

where A is a Lagrange multiplyer guaranteeing that the coorelation condition $\mathrm{E}[\psi\Lambda'] = \mathbf{I}_n$ is fullfilled; due to symmetry of $\mathcal{L}(\Lambda)$, ψ of form (12) is centered automatically; furthermore b bounds the influence of y_t on $x_{t|t}$.

The reader not familiar to the notion of influence curve may recur to section 5 and look up some of the references given there.

Instead of replacing the ML-equation by an M-Equation to be solved for $\psi \stackrel{!}{=} 0$, we use a one-step with same asymptotic properties:

$$x_{t|t} = x_{t|t-1} + \psi_{x_{t|t-1}} \begin{pmatrix} x^0_{t|t-1} \\ y_t \end{pmatrix} = \tag{13}$$

$$= x_{t|t-1} + \psi \begin{pmatrix} x^0_{t|t-1} - x_{t|t-1} \\ y_t - H x_{t|t-1} \end{pmatrix}. \tag{14}$$

We note the following properties:

- Setting $b = \infty$, $\psi = \hat{\psi} = \Sigma_{t|t}\Lambda$ reproduces the classical Kalman filter.
- There is quite a numerical problem determining A, which will be roughly explained in Section 5.
- As in the rLS the time expensive calibration step—here to find (A, b)—can be done beforehand.
- We calibrate the IC at the ideal model, so whenever we have a situation, where the sequence of $\Sigma_{t|t-1}$ stabilizes, we may, after a sufficient stabilization, stop calibration and use the last calibrated A, b for all subsequent times t. For details as to this stabilization we again refer to Anderson and Moore (1979) and Moore and Anderson (1980).

and again we already note that the rIC has preserved the crucial features from of the Kalman filter

- an easy, understandable structure with an initialization step, a prediction step and a correction step,
- the correction step is an easily evaluable function—it is "almost linear" !
- all information from the past useful for the future is captured in the values of $x_{t|t-1}$ and $x^0_{t|t-1}$.
- and: the correction step is now bounded in influence, as Δy enters bounded.

4.3 Variants: Separate Clipping

As already seen in (10), Λ is decomposed into a sum of two independent variables Λ_1 and Λ_2. They may be interpreted as estimating v_t and ξ_t, thus they represent in some sense the sensitive point to AO and IO respectively. Instead of simultaneous clipping of both summands, just clipping the "IO-part", i.e. Λ_1, or "AO-part", i.e. Λ_2, separately will therefore lead to a robustified version specialized to IO's or AO's. For the AO-specialization we get

$$\tilde{\psi}(s,u) = A\left(\Lambda_1 + \Lambda_2(s,u)\min\left\{1, \frac{b}{\|A\Lambda_2(s,u)\|}\right\}\right) \tag{15}$$

As to the IO-variant we have to admit that the robustification against IO's that is possible with rIC-sep-IO is limited. Here you should better take into account more information on the process history up to that point. Encouraging results however have been obtained in a situation with an unfavorable signal-to-noise ratio—in one dimension the quotient of R/Q.

4.4 Criterion for the Choice of b

As in the rLS case, we propose the assurance criterium for the choice of b: We adjust the procedure to a given relative efficiency loss in the ideal model. This loss is quantified in this case as the relative degradation of the "asymptotic" variance of our estimator which is in our situation just $\mathrm{E}\,\|\psi\|^2$, which in the ideal model gives again the MSE.

Of course the lower the clipping b, the higher the relative efficiency loss, so that we may solve

$$\mathrm{E}\,\|\psi\|^2 \stackrel{!}{=} (1+\delta)\,\mathrm{tr}\,\Sigma_{t|t} = (1+\delta)\,\mathrm{E}\,\|\hat{\psi}\|^2 \tag{16}$$

in b for a given efficiency loss δ, which is monotonous in δ.

4.5 Examples

For better comparability the examples for the rIC will use the same setups as those for rLS. So we just write down the modifications necessary to get from the rLS- to the rIC-example.

4.5.1 Example 7

As the first example is one-dimensional, `calibrIC` uses a simultaneous Newton procedure to determine A, b, so neither a number of grid points nor a MC-sample size is needed and parameter `N` is without meaning, as well as `fact` and `expl`. Nevertheless you are to transmit them to `calibrIC` and, beside the rLS setting we write

```
fact=1.2
expl=2
A0=0
b0=-1
typ= 0  ; rIC-sim
```

Next we calibrate the influence curve ψ to `e` $= 0.05$.

```
ergIC=calibrIC(T,Sig,H,F,Q,R,typ,A0,b0,e,N,eps,itmax,
                                                expl,fact,aus)

A=ergIC.A
b=ergIC.b
```

Calling `rICfil` is then very much as calling `rlsfil`—just with some extra parameters:

```
res= rICfil(y,mu,Sig,H,F,Q,R,typ,A,b)
frx = res.filtX
```


`rkalm07.xpl`

The graphical output is then done just as in Example 4.

4.5.2 Example 8

The second example goes through analogously with the following modifications with respect to Example 5:

```
N=300
eps=0.01
itmax=15
```

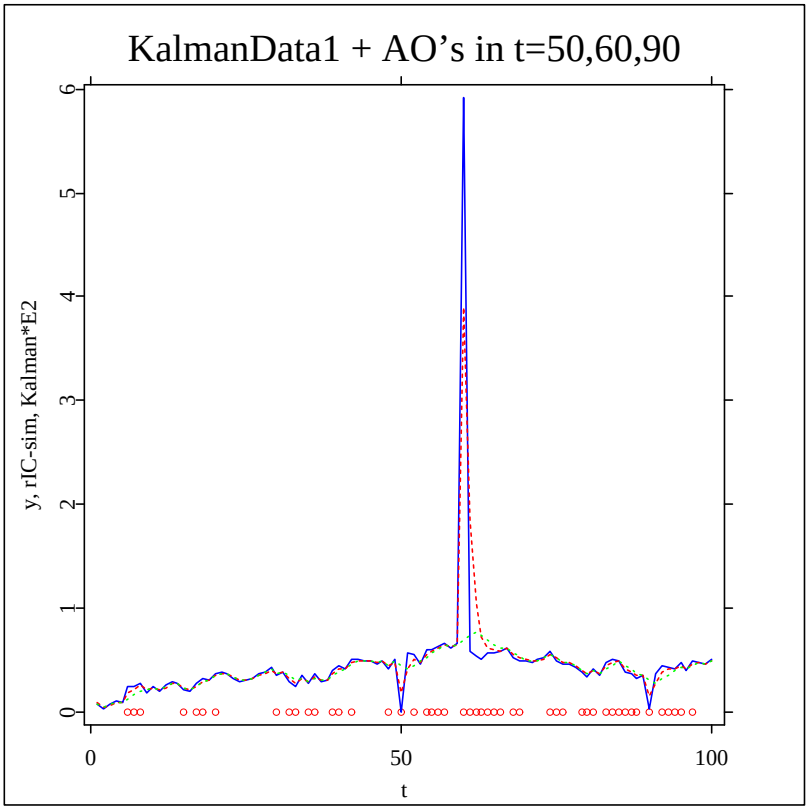

Figure 6: Actual observations $\hat{=}$ solid line, the classical Kalman filter $\hat{=}$ dashed / a bit lighter line, the rIC filter $\hat{=}$ dotted line; the clipped instances are marked by circles on top of the graphic.

```
aus=4
fact=1.2
expl=2
A0=0
b0=-1
typ= 0   ; rIC-sim
```

Note that as we are in 2 dimensions, integration along the directions is 1-dimensional and is done by a Romberg procedure; so the N might even be a bit too large. The next modifications are straightforward:

```
ergIC=calibrIC(T,Sig,H,F,Qid,R,typ,A0,b0,e,N,eps,itmax,
                                                expl,fact,aus)

A=ergIC.A
b=ergIC.b

res = kfilter2(y,mu,Sig, H,F,Qid,R)
fx = res.filtX
res= rICfil(y,mu,Sig,H,F,Qid,R,typ,A,b)
frx = res.filtX
```

rkalm08.xpl

The graphical result is displayed in Figure 7.

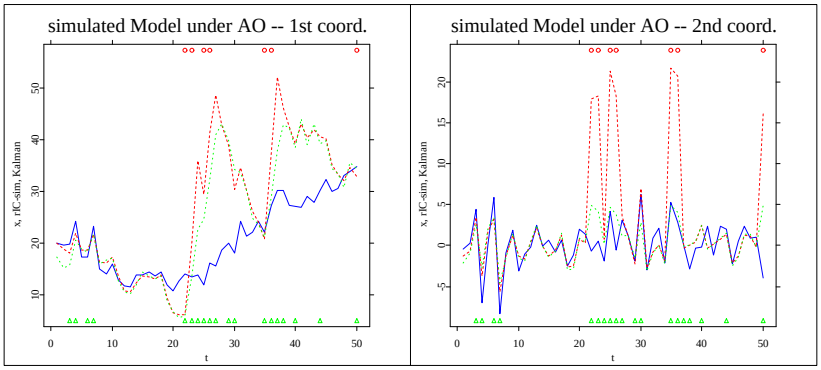

Figure 7: Simulated data according to Example 2 from Petr Franěk: The actual states ≙ solid line, the classical Kalman filter ≙ dashed / a bit lighter line, the rIC filter ≙ dotted line; the clipped instances are marked by circles on bottom of the graphic, the AO-instances by circles on top.

4.5.3 Example 9

Again, as in the third rLS-example, it is shown in the next example that we really loose some efficiency in the ideal model, using the rIC filter instead of the Kalman filter; the following modifications are to be done with respect to Example 6:

```
e=0.05
N=300
eps=0.01
itmax=15
aus=4
fact=1.2
expl=2
A0=0
b0=-1
typ= 1   ; rIC-sep-A0

ergIC=calibrIC(T,Sig,H,F,Q,R,typ,A0,b0,e,N,eps,itmax,
                                              expl,fact,aus)

A=ergIC.A
b=ergIC.b

res = kfilter2(y,mu,Sig, H,F,Q,R)
fx = res.filtX
res= rICfil(y,mu,Sig,H,F,Q,R,typ,A,b)
frx = res.filtX
fry=(H*frx')'
```


rkalm09.xpl

All this produces the graphics in Figure 8.

4.6 Possible Extensions

As sort of an outlook, we only want to mention here the possibility of using different norms to assess the quality of our procedures. The most important norms besides the euclidean are in our context those derived from the Fisher information of the ideal model (information-standardization) and the asymptotic

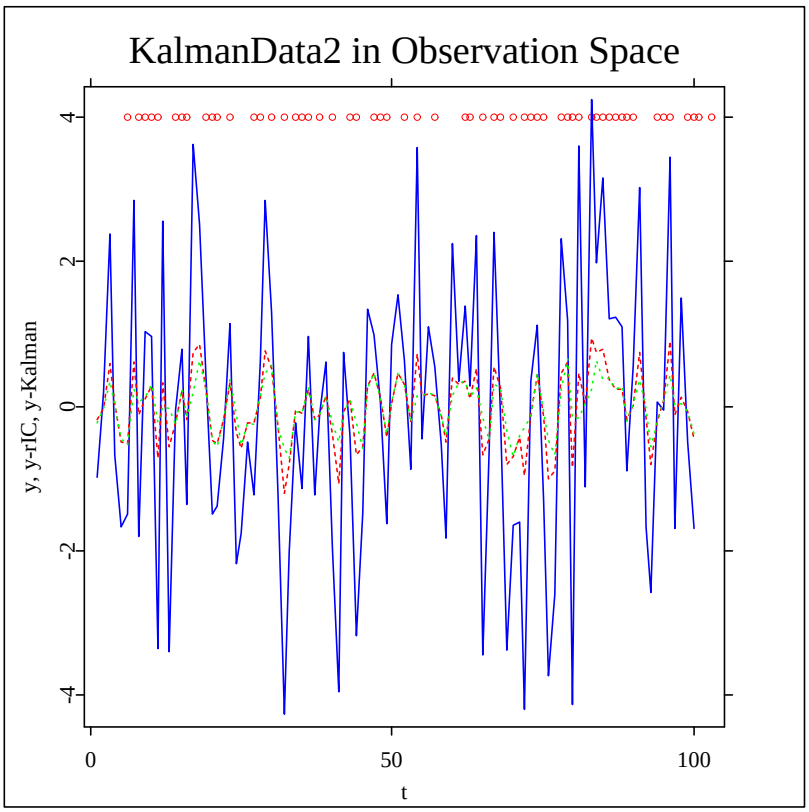

Figure 8: Actual observations $\hat{=}$ solid line, the classical Kalman filter $\hat{=}$ dashed / a bit lighter line, the rIC filter $\hat{=}$ dotted line; the clipped instances are marked by circles.

Covariance of ψ itself (self-standardization). Generally speaking these two have some nice properties compared to the euclidean norm; so among others, optimal influence curves in this norm stay invariant under smooth transformation in the parameter space, c.f. Rieder (1994). In the context of normal scores, they even lead to a drastic simplification of the calibration problem even in higher dimensions, c.f. Ruckdeschel (1999). Nevertheless the use of this norm

has to be justified by the application, and in the XploRe quantlib `kalman`, they have not yet been included.

5 Excursion: Generating Multivariate Robust Influence Curves for Normal Scores

`{A, b, V, ctrl} = ICerz(e, FI, A0, b0, N, eps, itmax, expl, fact0, aus)`
generates—if possible—a (simultaneously clipped) IC to a given efficiency loss for normal scores

`{A, b, V, ctrl} = ICerzsep(e, S1, S2, A0, b0, N, eps, itmax, expl, fact0, aus)`
generates—if possible—a (separately clipped) IC to a given efficiency loss for normal scores, with the same output list as `ICerz`.

Coming from the local i.i.d. asymptotic setup, we have applied estimators to the regression model (9) that have proven to be optimal there; so at this point we want to give a short abridge of the theory behind it and of how these optimal IC's may be obtained, numerically.

5.1 Definition of IC

In the local i.i.d. asymptotic setup, we consider a parametric family $\{P_\theta, \quad \theta \in \Theta\}$ and want to estimate the true θ_0 based on observations $x_1, \ldots x_n$. To do so we only allow for asymptotically linear estimators S_n for this parameter θ, i.e.,

$$\sqrt{n}(S_n - \theta) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \psi_\theta(x_i) + \mathrm{o}_{P_\theta}(1) \tag{17}$$

for some L_2-Variable ψ. As is shown in Rieder (1994), to get local Fisher consistency of S_n—i.e. S_n has to converge to $\theta_0 + h/\sqrt{n}$ in $P_{\theta_0 + h/\sqrt{n}}$-probability for all h—we must necessarily have

$$\mathrm{E}_\theta[\psi_\theta] = 0, \qquad \mathrm{E}_\theta[\psi_\theta \Lambda_\theta'] = \mathbf{I}_n. \tag{18}$$

Optimality results also to be looked up in Rieder (1994) show that for the i.i.d. setup,

- the IC minimizing the trace of the covariance in the ideal model subject to a bias bound in a neighbourhood around the ideal situation is just of Hampel-Kraker form; [in short this is $\min \mathrm{E}\,\|\psi\|^2$ subject to $\|\psi\| \leq b[P]$].
- for b not too small, there exists exactly one Hampel-Krasker IC ψ and as $\mathcal{L}(\Lambda)$ is continuous, A also is unique.
- to each δ not too big there is exactly one pair (A, b) fullfilling (16) and (18).

5.2 General Algorithm

In general, for b given, A is determined by the implicite equation

$$A^{-1} \stackrel{!}{=} \mathrm{E}\left[\Lambda\Lambda' \min\left\{1, \frac{b}{\|A\Lambda\|}\right\}\right]. \tag{19}$$

5.2.1 Arbitrary Dimension n

As clipping is done continuously, and by the integration $\mathrm{E}[\cdot]$, we achieve an arbitrary smoothness of (19) in A and b.

As we know that for $b = \infty$, $A = \mathcal{I}^{-1}$, for b not too small we can use $\mathcal{I}^{-1}$ as a starting value for the fixed point iteration

$$A_{i+1}^{-1} \stackrel{!}{=} \mathrm{E}\left[[\Lambda\Lambda' \min\left\{1, \frac{b}{\|A_i\Lambda\|}\right\}\right]. \tag{20}$$

Proofs for local convergence may be found in Ruckdeschel (1999).

For smaller b, we first solve (19) for a larger b' and then take $A(b')$ as a starting value for (20); as a criterium whether (20) converges or not we take the development of the size of A which is controlled by the parameter `expl`, the stepsize from b to b' is controlled by `fact`.

Once we have determined for given b $A(b)$, we control whether $\mathrm{E}\,\|\psi\|^2$ is smaller or larger than $(1+\delta)\,\mathrm{tr}\,\mathcal{I}^{-1}$. To determine the pair $(b, A(b))$ for given δ, we use a bisection algorithm, as $\mathrm{E}\,\|\psi\|^2$ is strictly dicreasing in b.

5.2.2 One Dimension

In dimension $n = 1$, the problem is only 2-dimensional, so it pays off, using a simultaneous Newton procedure to determine $(b, A(b))$. This is done by the auxiliary routines `abinfonewton` and `absepnewton` for the simultaneously and the separately clipped case, respectively.

5.3 Polar Representation and Explicite Calculations

In the case of $\Lambda \sim \mathcal{N}_n(0, \mathcal{I})$, $n > 1$, we have some nice properties, which make calculations in (20) easier.

5.3.1 Polar Representation

We write Λ as

$$\Lambda = \mathcal{I}^{\frac{1}{2}} \tilde{\Lambda} = \mathcal{I}^{\frac{1}{2}} Y u \tag{21}$$

with $\tilde{\Lambda} \sim N_n(0, \mathbf{I}_n)$, $u = \|\tilde{\Lambda}\|$, $Y = \tilde{\Lambda}/u$. Then $Y \sim \text{ufo}(\mathcal{S}_{n-1})$, $u^2 \sim \chi^2_n$ and u, Y independent.

Now we have to solve

$$\mathcal{I}^{-\frac{1}{2}} A^{-1} \mathcal{I}^{-\frac{1}{2}} \stackrel{!}{=} \mathrm{E}[YY' r(Y)] \tag{22}$$

$$(1+\delta) \operatorname{tr} \Sigma_{t|t} \stackrel{!}{=} \operatorname{tr} A \mathcal{I}^{\frac{1}{2}} \mathrm{E}[YY' s(Y)] \mathcal{I}^{\frac{1}{2}} A \tag{23}$$

with

$$r(Y) = \mathrm{E}\left[u \min\left\{ u, \frac{b}{\|A\mathcal{I}^{\frac{1}{2}} Y\|} \right\} | Y \right] \tag{24}$$

$$s(Y) = \mathrm{E}\left[\min\left\{ u^2, \frac{b^2}{\|A\mathcal{I}^{\frac{1}{2}} Y\|^2} \right\} | Y \right] \tag{25}$$

5.3.2 Explicit Calculations for the Absolute Value

For given Y, $c = b/\|A\mathcal{I}^{\frac{1}{2}} Y\|$ is constant, and calculation of r, s using the proposition that clipped moments of u can be calculated by means of (higher) clipped moments of the standard normal—c.f. Ruckdeschel (1999).

5.4 Integrating along the Directions

For the remaining integration along the directions Y we do

- $n = 1$: nothing has to be done; (A, b) are calculated simultaneously by a Newton procedure.
- $n = 2$: 2×2-valued integration along the unit-circle; done by a Romberg procedure.
- $n > 2$: $n \times n$-valued integration along the surface of the n-dim unit ball; done by MC-integration.

For simultaneous clipping we additionally have (c.f. Ruckdeschel (1999)) the interesting proposition that clipping only effects the spectrum of $\mathcal{I}$ but not the invariant spaces. This is used to transform integration from $YY'r$, $YY's$ to $Y'Yr$, $Y'Ys$, thus reducing the problem from n^2 to n dimensions.

5.5 Short Description of the Auxiliary Routines Used

The further quantlets in the quantlib `kalman` just being auxiliary routines for the ones described up to now, we confine ourselves to shortly giving a survey of these routines in form of a table; note that in the table, `AIh`$\hat{=}A\mathcal{I}^{\frac{1}{2}}$, and u, x stand for random variables, $u^2 \sim \chi^2_n$ and $x \sim N(0, 1)$.

quantlet	input	output	function
itera	(A0,FI,b,N, eps,itmax, expl)	(A,V,ctrl)	Fixed-Point-Iteration (20) for sim. clipping; also decides if there was convergence or not
iteras	(A0,S1,S2,b, N,eps,itmax, expl)	(A,V,ctrl)	Fixed-Point-Iteration (20) for sep. clipping; also decides if there was convergence or not
numint2	(aIh, b, N)	(r,s)	$n = 2$: (reduced problem) Romberg-integration
numint2m	(aIh, b, N)	(r,s)	$n = 2$: (full problem) Romberg-integration
stointp	(aIh, b, N)	(r,s)	$n > 2$: (reduced problem) MC-integration
stointpm	(aIh, b, N)	(r,s)	$n > 2$: (full problem) MC-integration
ewinn	(c,n)	y	calculates $r(c)$
ew2inn	(c,n)	y	calculates $s(c)$
betrnormF	(t,n)	y	calculates $\mathrm{E}[(u \leq t)]$.
betrnormE	(t,n)	y	calculates $\mathrm{E}[u(u \leq t)]$.
betrnormV	(t,n)	y	calculates $\mathrm{E}[u^2(u \leq t)]$.
nmomnorm	(t,n)	y	calculates $\mathrm{E}[x^n(x \leq t)]$.

References

Anderson, B. D. O. and Moore, J. B. (1979). *Optimal filtering*, Prentice-Hall, Inc. Englewood Cliffs, NJ.

Boncelet, C. G. jun. and Dickinson, B. W. (1984). A variant of Huber robust regression, *SIAM J. Sci. Stat. Comput.* **5**: 720–734.

Cipra, T. and Romera, R. (1991). Robust Kalman filter and its application in time series analysis, *Kybernetika* **27**: 481–494.

Fahrmeir, L. and Künstler R. (1999). Penalized Likelihood Smoothing in Robust State Space Models *Metrika* **49**: 173–191.

Fox, A. J. (1972). Outliers in Time Series, *Journal of the Royal Statistical Society, Series B* **43**: 350–363.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (1986). *Robust statistics, The approach based on influence functions,* Wiley, New York.

Härdle, W., Klinke, S., and Müller, M. (2000). *XploRe Learning Guide*, Springer.

Huber, P. J. (1964). Robust estimation of a location parameter, *Annals of Statistics* **35**: 73–101.

Huber, P. J. (1981). *Robust statistics,* Wiley, New York.

Moore, J. B. and Anderson, B. D. O. (1980). Coping with singular transition matrices in estimation and control stability theory, *Int. J. Control* **31**: 571-586.

Rieder, H. (1994). *Robust asymptotic statistics*, Springer, New York.

Ruckdeschel, P. (1999). Algorithms for Robust Kalman Filtering, PHD-thesis, unpublished manuscript.