Exponential smoothing

Marco Fattore

1 Introduction

In the previous lesson, we have discussed how to forecast a time series, when the time dependency is linear and deterministic. Actually, the behavior of real time series is more complex and trying to specify a mathematical function $f(t)$ capable to fit the data and forecast new observations would be rather artificial and ineffective. In particular, any deterministic model of time dependency would lead, at best, to *overfitting*, i.e. to building models that adapt to and fit the observed data, but fail to properly forecast the future. To get more flexible and effective forecasting procedures, we must explicitly base them on the local past history of the series, in such a way that the "memory" structure of the data generating process is adaptively taken into account. In this lesson, we introduce *Exponential smoothing*, a very simple, non-inferential approach to forecasting, which proves useful when dealing with short-term forecasting on short time series (where more sophisticated inferential procedures cannot be applied).

2 Formal development

Let $x_1, \ldots, x_t$ be an observed time series which oscillates around a fixed value, but where some persistencies occur and some correlation between the past and the future exists. In such a case, the intuitive idea is that recent observations are most useful than past ones, when forecasting the next future. We then want to forecast x_{t+1} as a weighted average of $x_1, \ldots, x_t$, where weights decrease as we move to the past.

2.1 A technical result

To develop the Exponential smoothing procedure, we need a simple technical result. Let λ be a real number, such that $|\lambda| < 1$; then the following equality holds:

$$\sum_{i=0}^{\infty} \lambda^i = \frac{1}{1-\lambda}. \tag{1}$$

Proof. Since $|\lambda| < 1$, $\sum_{i=0}^{\infty} \lambda^i$ is convergent, so that we can put $\sum_{i=0}^{\infty} \lambda^i = S < \infty$. We then have:

$$S = \sum_{i=0}^{\infty} \lambda^i = 1 + \sum_{i=1}^{\infty} \lambda^i = 1 + \sum_{i=0}^{\infty} \lambda^{i+1} = 1 + \lambda \sum_{i=0}^{\infty} \lambda^i = 1 + \lambda S, \tag{2}$$

so that $S = 1/(1-\lambda)$.

3 Exponential smoothing: the infinite case

Let us begin by considering the case of a time series starting at $t = -\infty$. This is clearly an abstraction, but it is very useful to set the stage and to understand the functioning of the exponential smoothing procedure.

Let $x_{-\infty}, \ldots, x_t$ be an observed and stationary time series; we want to forecast the next observation x_{t+1}, given the observed past, as a linear combination of the past observations, namely we put:

$$\hat{x}_{t+1|\leq t} = \sum_{i=0}^{\infty} \alpha_i x_{t-i} \tag{3}$$

where $\hat{x}_{t+1|\leq t}$ is the forecast for x_t, $\alpha_i \geq 0 \ \forall \ i$ and $\sum_{i=0}^{\infty} \alpha_i = 1$. In practice, $\hat{x}_{t+1|\leq t}$ is a weighted average of the past observations. To make things simple, we want coefficients α_i to depend upon a single parameter $0 < \lambda < 1$ and we choose the following parametrization:

$$\alpha_i = (1-\lambda)\lambda^i. \tag{4}$$

This choice amounts at averaging past observations with weights decreasing as we move to the past, following a power-law. Notice that forecasting by means of weighted averages makes this approach suitable for non-trended time-series only. In fact, weighted averages satisfy the *internality* condition, so that $\inf(x_{-\infty}, \ldots, x_t) \leq \hat{x}_{t+1|\leq t} \leq \sup(x_{-\infty}, \ldots, x_t)$. As a consequence, the exponential smoothing procedure cannot forecast values outside the range of past observations (and this is why we have restricted the procedure to stationary time-series only). With the above parametrization, we have:

$$\hat{x}_{t+1|\leq t} = (1-\lambda) \sum_{i=0}^{\infty} \lambda^i x_{t-i}. \tag{5}$$

This expression can be casted in two different and useful forms.

1. The forecast $\hat{x}_{t+1|\leq t}$ can be expressed as a weighted average of the last observation x_t and the forecast $\hat{x}_{t|\leq t-1}$ of x_t, in fact:

$$\begin{aligned}
\hat{x}_{t+1|\leq t} &= (1-\lambda) \sum_{i=0}^{\infty} \lambda^i x_{t-i} = (1-\lambda)x_t + (1-\lambda) \sum_{i=1}^{\infty} \lambda^i x_{t-i} = \\
&= (1-\lambda)x_t + \lambda(1-\lambda) \sum_{i=1}^{\infty} \lambda^{i-1} x_{t-i} = (1-\lambda)x_t + \lambda(1-\lambda) \sum_{i=0}^{\infty} \lambda^i x_{t-1-i} = \\
&= (1-\lambda)x_t + \lambda \hat{x}_{t|\leq t-1}.
\end{aligned}$$

2. The forecast $\hat{x}_{t+1|\leq t}$ can then also be casted into the so-called *error-correction* form:

$$\hat{x}_{t+1|\leq t} = (1-\lambda)x_t + \lambda\hat{x}_{t|\leq t-1} = x_t - \lambda(x_t - \hat{x}_{t|\leq t-1}).$$

The last term is λ times the forecast error of x_t, based on its past. Putting $\lambda = 1-\alpha$, the last expression becomes:

$$\begin{aligned}\hat{x}_{t+1|\leq t} &= x_t - (1-\alpha)(x_t - \hat{x}_{t|\leq t-1}) = \hat{x}_{t|\leq t-1} + \alpha(x_t - \hat{x}_{t|\leq t-1}) = \\ &= \hat{x}_{t|\leq t-1} + (1-\lambda)(x_t - \hat{x}_{t|\leq t-1}).\end{aligned}$$

This is the final *error-correction* form, which recursively expresses the forecast of x_{t+1} as the forecast $\hat{x}_{t|\leq t-1}$, corrected by the forecast error, weighted by $1-\lambda$.

The above forms of the Exponential smoothing forecast show how the procedure adapts to the movements of the time series, trying to "chase" its dynamics over time. In other words, even if the "global" level of the time series does not change (recall that the series is assumed to be stationary), the *local* level can indeed change and the Exponential smoothing tries to "react" to this behavior, linking the short-term forecast to the varying local level of the series.

4 Exponential smoothing: the finite case

In practice, we deal only with finite sequences of observations, i.e. with finite time-series, which start from a fixed time t_1 in the past. In this case, we cannot apply Exponential smoothing in its orginal form (i.e. as an infinite weighted average); however, we can use its recursive formulation, by choosing a proper initial condition to start the computations.

Consider $\hat{x}_{t+1|\leq t} = (1-\lambda)x_t + \lambda\hat{x}_{t|\leq(t-1)}$ and substitute $\hat{x}_{t|\leq(t-1)}$ with its recursive espression $\hat{x}_{t|\leq(t-1)} = (1-\lambda)x_{t-1} + \lambda\hat{x}_{t-1|\leq(t-2)}$:

$$\hat{x}_{t+1|\leq t} = (1-\lambda)x_t + \lambda(1-\lambda)x_{t-1} + \lambda^2\hat{x}_{t-1|\leq(t-2)}. \tag{6}$$

Going on and substituting recursively the exponential smoothing relation into the above equality, in the end we get:

$$\hat{x}_{t+1|\leq t} = (1-\lambda)\sum_{i=0}^{t-1}\lambda^i x_{t-i} + \lambda^t\hat{x}_{1|\leq 0}. \tag{7}$$

Setting the initial condition $\hat{x}_{1|\leq 0} = S_0$, we can then get $\hat{x}_{t+1|\leq t}$ as:

$$\hat{x}_{t+1|\leq t} = (1-\lambda)\sum_{i=0}^{t-1}\lambda^i x_{t-i} + \lambda^t S_0. \tag{8}$$

Ideally, the term S_0 should be a "summary" of the (unobserved) past and, in practice, must be guessed in some way. The most common choices for S_0 are to simply put $S_0 = x_1$, or to put $S_0 = 1/q \sum_{i=1}^{q} x_i$, for a suitable number of initial observations q. Notice, however, that since $0 < \lambda < 1$, for sufficiently high values of t the term $\lambda^t S_0$ becomes negligible, so that on the long-run the effect of the initial condition gets lost.

The key point, to run the exponential smoothing procedure, is finally to set the value for parameter λ. This can be done numerically, following the procedure below:

1. We set S_0 as discussed above.
2. Given S_0, we can compute the forecast sequence $\hat{x}_{1|0} = S_0, \hat{x}_{2|1}, \hat{x}_{3|\leq 2}, \hat{x}_{4|\leq 3}, \ldots, \hat{x}_{t|\leq t-1}$.
3. From the forecast sequence, we can compute the squared sum of the forecasting errors, i.e. $E^2(\lambda) = \sum_{i=1}^{t} (x_i - \hat{x}_{i|\leq (i-1)})^2$. This expression depends upon the smoothing parameter and then we search for λ to minimize $E^2(\lambda)$.

Final remarks

1. We have discussed how to forecast x_{t+1}, given its past. However, we may want to forecast $x_{t+2}, x_{t+3} \ldots$ or, in general, ℓ steps ahead of t. A practical way to do this is to artificially add to the time series the forecasted values $\hat{x}_{t+1}, \ldots, \hat{x}_{t+\ell-1}$ and to compute $\hat{x}_{t+\ell}$ on this new series, as usual. Notice that, this way, we still express $\hat{x}_{t+\ell}$ as a weighted average of $x_1, \ldots, x_t$, so we are still using the information comprised in the observed series only.
2. Since the forecast errors depend upon S_0 and λ, we can set both parameters by numerically minmizing $E(S_0, \lambda)^2$ on them, jointly.
3. If the time series is long enough, it is reccomandable to perform the minimization of $E(S_0, \lambda)^2$ on a subset of data (say, the first 2/3 of the time series) and to test the forecasting effectiveness on the remaining data, so as to check for possible overfitting in the minimization process.

5 R codes

In the following, we provide some R code performing Exponential smoothing. The first procedure computes the Exponential smoothing forecast, for smoothing parameters set by the user and for two different initial conditions (see Figure 1 and 2). The second R procedure does the same, but optimizing over the smoothing parameter, as described in the main text (see Figure 3 and 4).

```
#####################################
#
# EXPONENTIAL SMOOTHING STEP-BY-STEP
#
#####################################



## GENERATING THE TIME SERIES ##

LNG<-70 # Length of the time series (must be greater than 20)
SERIES<-arima.sim(list(order=c(1,0,0), ar=.8), n=LNG)


## SETTING SMOOTHING PARAMETERS ##

LAMBDA<-.2 # Smoothing parameter (set it in (0,1) )
Q<-10 # Number of initial obsevations to average
S0a<-SERIES[1] # First kind of initialization
S0b<-mean(SERIES[1:Q]) # Second kind of initialization


## PERFORMING SMOOTHING ##

SMOOTHED<-matrix(0,nrow=length(SERIES), ncol=2)
colnames(SMOOTHED)<-c("S0a", "S0b")
SMOOTHED[1,"S0a"]<-S0a
SMOOTHED[1,"S0b"]<-S0b


for(h in 2:length(SERIES))
{
SMOOTHED[h,"S0a"]<-(1-LAMBDA)*SERIES[h-1]+LAMBDA*SMOOTHED[h-1,"S0a"]
SMOOTHED[h,"S0b"]<-(1-LAMBDA)*SERIES[h-1]+LAMBDA*SMOOTHED[h-1,"S0b"]
}


## COMPUTING POINTWISE SMOOTHING ERRORS ##

SMOOTHING.ERROR<-matrix(0,nrow=length(SERIES), ncol=2)
colnames(SMOOTHING.ERROR)<-c("S0a", "S0b")

SMOOTHING.ERROR[,"S0a"]<-SERIES-SMOOTHED[,"S0a"]
```

```
SMOOTHING.ERROR[,"S0b"]<-SERIES-SMOOTHED[,"S0b"]


## COMPUTING SUM OF SQUARED ERRORS ##

SQUARED.ERROR<-vector("numeric", length=2)
names(SQUARED.ERROR)<-c("S0a", "S0b")
SQUARED.ERROR["S0a"]<-sum(SMOOTHING.ERROR[,"S0a"]^2)
SQUARED.ERROR["S0b"]<-sum(SMOOTHING.ERROR[,"S0b"]^2)
MEAN.SQUARED.ERROR<-SQUARED.ERROR/nrow(SMOOTHING.ERROR)
names(MEAN.SQUARED.ERROR)<-c("S0a", "S0b")
SQUARED.ERROR
MEAN.SQUARED.ERROR



## ONE-STEP AHEAD FORECASTING ##

FORECAST<-vector("numeric", length=2)
names(FORECAST)<-c("S0a", "S0b")
FORECAST["S0a"]<-(1-LAMBDA)*SERIES[length(SERIES)]
+LAMBDA*SMOOTHED[nrow(SMOOTHED),"S0a"]
FORECAST["S0b"]<-(1-LAMBDA)*SERIES[length(SERIES)]
+LAMBDA*SMOOTHED[nrow(SMOOTHED),"S0b"]


## PLOTTING RESULTS ##

par(mfrow=c(2,2))

plot(SERIES, xlim=c(1,LNG+1), pch=20, type="o", main="SMOOTHED S0a")
points(SMOOTHED[,"S0a"], pch=20, type="o", col="red", cex=1.1)

plot(SERIES, xlim=c(1,LNG+1), pch=20, type="o", main="SMOOTHED S0b")
points(SMOOTHED[,"S0b"], pch=20, type="o", col="blue", cex=1.1)

plot(SERIES, xlim=c(LNG-20,LNG+1), type="o", main="FORECAST S0a (red point)")
points((LNG+1), FORECAST["S0a"], pch=20, type="o", col="red", cex=2)

plot(SERIES, xlim=c(LNG-20,LNG+1), type="o", main="FORECAST S0b (red point)")
points((LNG+1), FORECAST["S0b"], pch=20, col="red", cex=2)
```

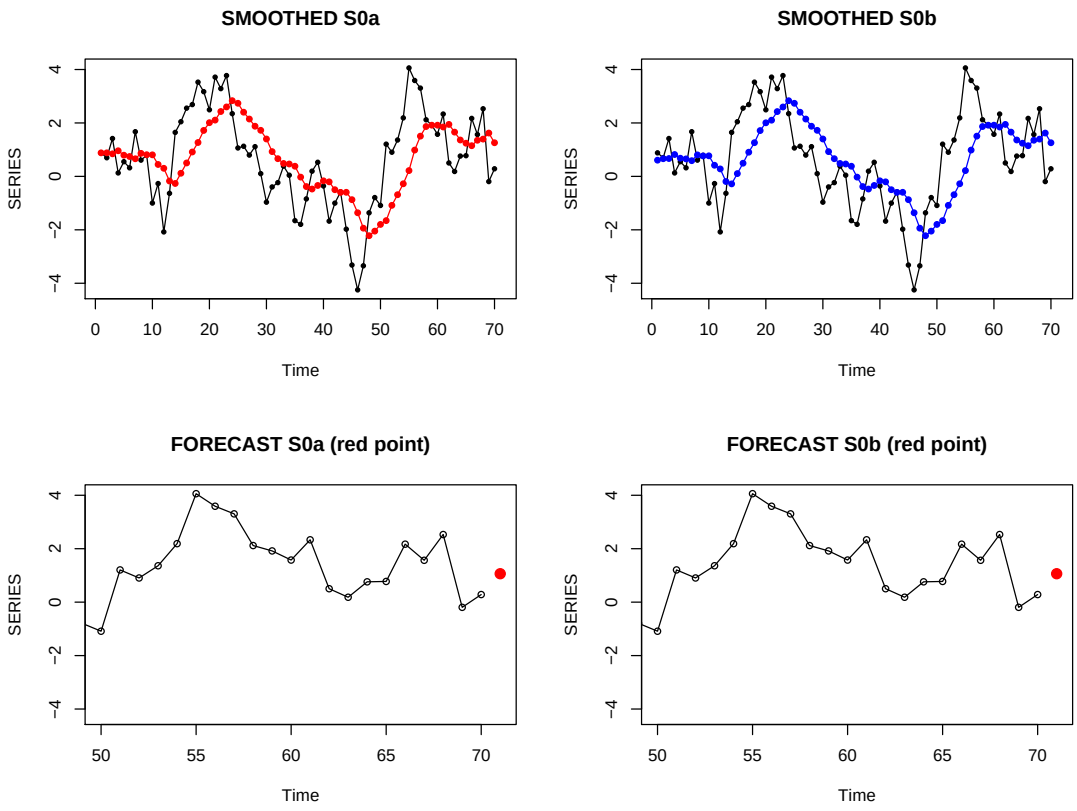

Figure 1: Upper panels: input time series (black) and the rolling forecasts (coloured); lower panels: forecast of the next observation (for two different initial conditions). Smoothing parameter = 0.8.

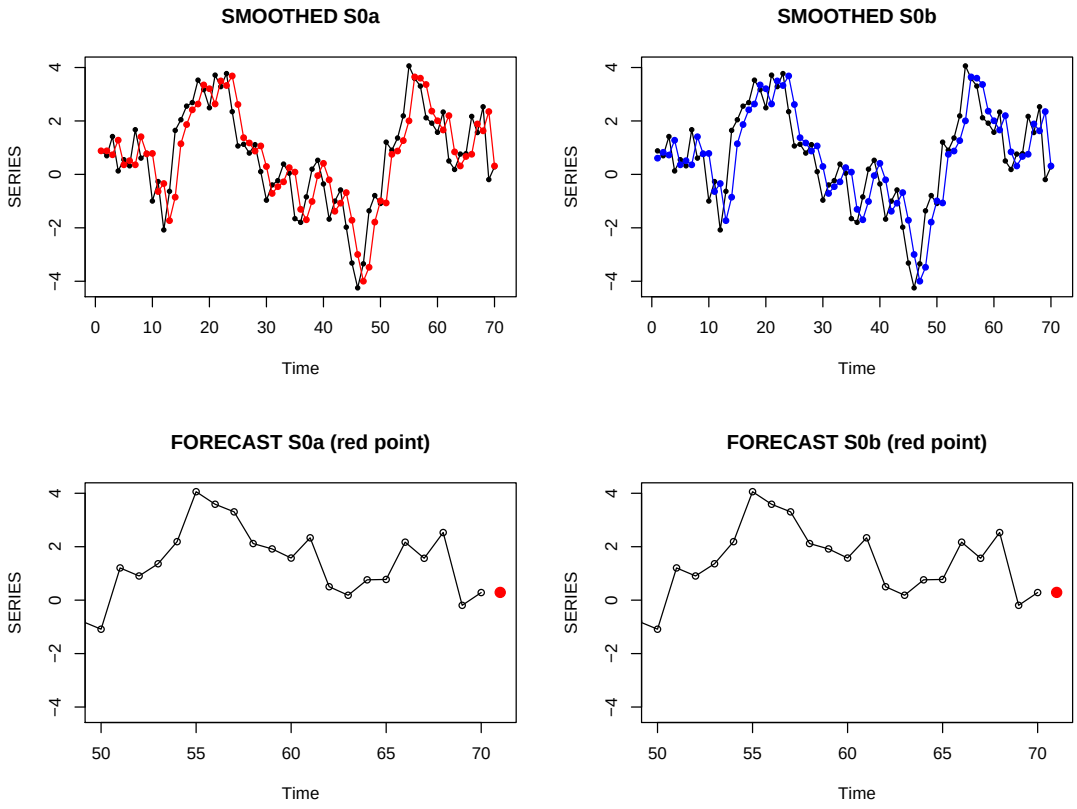

Figure 2: Upper panels: input time series (black) and the rolling forecasts (coloured); lower panels: forecast of the next observation (for two different initial conditions). Smoothing parameter = 0.2.

```
###################################
#
# OPTIMIZED EXPONENTIAL SMOOTHING
#
###################################

# REMARK: WE COULD USE NUMERICAL OPTIMIZATION FUNCTIONS
# AVAILABLE IN R. FOR DIDACTICAL PURPOSES, HERE WE INSTEAD
# DEFINE A GRID OF LAMBDA VALUES, AND SEARCH THE BEST VALUE
# IN IT. WE BUILD A FUNCTION TO COMPUTE THE SMMOTHING ERROR
# AND APPLY IT TO THE GRID.

EXPONENTIAL.SMOOTHING<-function(LAMBDA, S0a, S0b)
{
SMOOTHED<-matrix(0,nrow=length(SERIES), ncol=2)
colnames(SMOOTHED)<-c("S0a","S0b")
SMOOTHED[1,"S0a"]<-S0a
SMOOTHED[1,"S0a"]<-S0b

for(h in 2:length(SERIES))
{
SMOOTHED[h,"S0a"]<-(1-LAMBDA)*SERIES[h-1]+LAMBDA*SMOOTHED[h-1,"S0a"]
SMOOTHED[h,"S0b"]<-(1-LAMBDA)*SERIES[h-1]+LAMBDA*SMOOTHED[h-1,"S0b"]
}

SMOOTHING.ERROR<-matrix(0,nrow=length(SERIES), ncol=2)
colnames(SMOOTHING.ERROR)<-c("S0a","S0b")
SMOOTHING.ERROR[,"S0a"]<-SERIES-SMOOTHED[,"S0a"]
SMOOTHING.ERROR[,"S0b"]<-SERIES-SMOOTHED[,"S0b"]
SQUARED.ERROR<-vector("numeric", length=2)
names(SQUARED.ERROR)<-c("S0a","S0b")
SQUARED.ERROR["S0a"]<-sum(SMOOTHING.ERROR[,"S0a"]^2)
SQUARED.ERROR["S0b"]<-sum(SMOOTHING.ERROR[,"S0b"]^2)
FORECAST<-vector("numeric", length=2)
names(FORECAST)<-c("S0a","S0b")
FORECAST["S0a"]<-(1-LAMBDA)*SERIES[length(SERIES)]+LAMBDA*SMOOTHED[nrow(SMOOTHED),"S0a"]
FORECAST["S0b"]<-(1-LAMBDA)*SERIES[length(SERIES)]+LAMBDA*SMOOTHED[nrow(SMOOTHED),"S0b"]
list("SMOOTHED"=SMOOTHED, "SMOOTHING.ERROR"=SMOOTHING.ERROR,
    + "FORECAST"=FORECAST, "SQUARED.ERROR"=SQUARED.ERROR)
}


LNG<-70 # Length of the time series (must be greater than 20)
```

```
SERIES<-arima.sim(list(order=c(1,0,0), ar=.6), n=LNG)

Q<-10 # Number ofinitial obsevations to average
S0a<-SERIES[1] # First kind of initialization
S0b<-mean(SERIES[1:Q]) # Second kind of initialization


LAMBDA.GRID<-seq(from=0, to=1, length.out=1000) #1000 values, comprising also 0 and 1.
EXP.SMOOTHING<-lapply(LAMBDA.GRID, function(x) EXPONENTIAL.SMOOTHING(x, S0a, S0b))


SQUARED.ERROR.GRID<-lapply(EXP.SMOOTHING, function(x) x$SQUARED.ERROR)
SQUARED.ERROR.GRID<-matrix(unlist(SQUARED.ERROR.GRID), ncol=2, byrow=TRUE)
colnames(SQUARED.ERROR.GRID)<-c("S0a","S0b")

QUALE.OPTIM<-vector("numeric", length=2)
names(QUALE.OPTIM)<-c("S0a","S0b")
QUALE.OPTIM["S0a"]<-which(SQUARED.ERROR.GRID[,"S0a"]==min(SQUARED.ERROR.GRID[,"S0a"]))
QUALE.OPTIM["S0b"]<-which(SQUARED.ERROR.GRID[,"S0b"]==min(SQUARED.ERROR.GRID[,"S0b"]))
LAMBDA.OPTIM<-c(LAMBDA.GRID[QUALE.OPTIM["S0a"]], LAMBDA.GRID[QUALE.OPTIM["S0b"]])
LAMBDA.OPTIM #Values of the optimal LAMBDAs for the two initial conditions

## PLOTTING TOTAL SQUARED ERRORS ##

windows()
par(mfrow=c(1,2))
plot(LAMBDA.GRID,SQUARED.ERROR.GRID[,"S0a"], type="o", ylim=c(min(SQUARED.ERROR.GRID),
 + max(SQUARED.ERROR.GRID)), xlab="LAMBDA", ylab="Squared error", main="S0a")
plot(LAMBDA.GRID,SQUARED.ERROR.GRID[,"S0b"], type="o",  ylim=c(min(SQUARED.ERROR.GRID),
 +max(SQUARED.ERROR.GRID)), xlab="LAMBDA", ylab="Squared error", main="S0b")


## EXTRACTING AND PLOTTING ONE-STEP AHEAD FORECAST AS A FUNCTION OF LAMBDA ##

FORECAST.GRID<-lapply(EXP.SMOOTHING, function(x) x$FORECAST)
FORECAST.GRID<-matrix(unlist(FORECAST.GRID), ncol=2, byrow=TRUE)
colnames(FORECAST.GRID)<-c("S0a","S0b")

windows()
par(mfrow=c(1,2))
plot(FORECAST.GRID[,"S0a"], type="o", ylim=c(min(FORECAST.GRID),
 + max(FORECAST.GRID)), xlab="LAMBDA", ylab="Forecast", main="S0a")
plot(FORECAST.GRID[,"S0b"], type="o",  ylim=c(min(FORECAST.GRID),
```

```
 + max(FORECAST.GRID)), xlab="LAMBDA", ylab="Forecast", main="S0b")


## EXTRACTING OPTIMAL SOLUTIONS (for S=a and S0b) ##

SOLUTION.S0a<-EXP.SMOOTHING[[QUALE.OPTIM["S0a"]]]
SOLUTION.S0a$SMOOTHED<-SOLUTION.S0a$SMOOTHED[,"S0a"]
SOLUTION.S0a$SMOOTHING.ERROR<-SOLUTION.S0a$SMOOTHING.ERROR[,"S0a"]
SOLUTION.S0a$FORECAST<-SOLUTION.S0a$FORECAST["S0a"]
SOLUTION.S0a$SQUARED.ERROR<-SOLUTION.S0a$SQUARED.ERROR["S0a"]

SOLUTION.S0b<-EXP.SMOOTHING[[QUALE.OPTIM["S0b"]]]
SOLUTION.S0b$SMOOTHED<-SOLUTION.S0b$SMOOTHED[,"S0b"]
SOLUTION.S0b$SMOOTHING.ERROR<-SOLUTION.S0b$SMOOTHING.ERROR[,"S0b"]
SOLUTION.S0b$FORECAST<-SOLUTION.S0b$FORECAST["S0b"]
SOLUTION.S0b$SQUARED.ERROR<-SOLUTION.S0b$SQUARED.ERROR["S0b"]


## PLOTTING SMOOTHING AND FORECAST OF THE OPTIMAL SOLUTIONS ##

windows()
par(mfrow=c(2,2))

plot(SERIES, xlim=c(1,LNG+1), type="o", pch=20, main="SMOOTHED S0a")
points(SOLUTION.S0a$SMOOTHED, type="o", pch=20, col="red")

plot(SERIES, xlim=c(1,LNG+1), type="o", pch=20, main="SMOOTHED S0b")
points(SOLUTION.S0b$SMOOTHED, type="o", pch=20, col="blue")

plot(SERIES, xlim=c(LNG-20,LNG+1), type="o", pch=20, main="FORECAST S0a (red point)")
points((LNG+1), SOLUTION.S0a$FORECAST, type="o", pch=20, col="red", cex=2)

plot(SERIES, xlim=c(LNG-20,LNG+1), type="o", pch=20, main="FORECAST S0b (red point)")
points((LNG+1), SOLUTION.S0a$FORECAST, type="o", pch=20, col="red", cex=2)
```

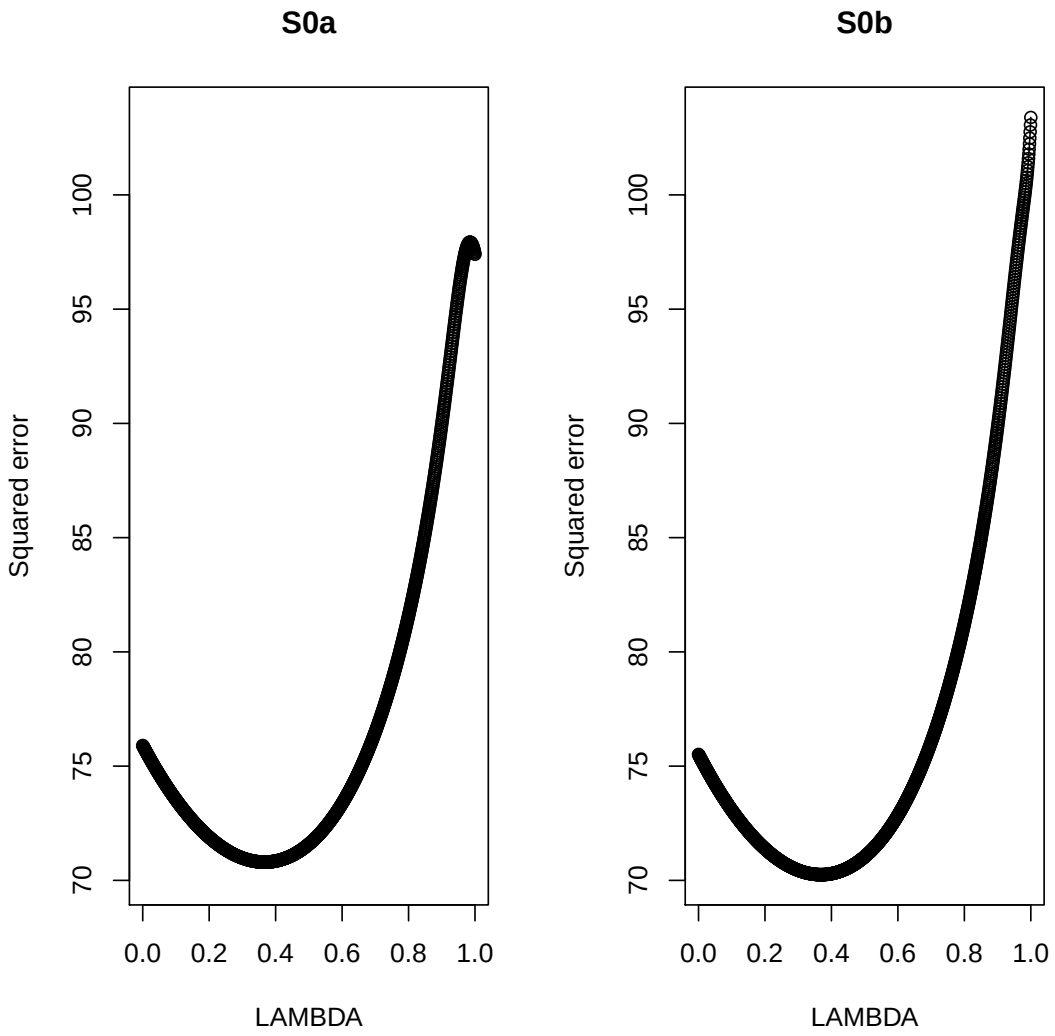

Figure 3: Squared error of the rolling forecasts, as the smoothing parameter changes in $[0, 1]$, for two different initial conditions (minima at 0.366 and 0.368, respectively).

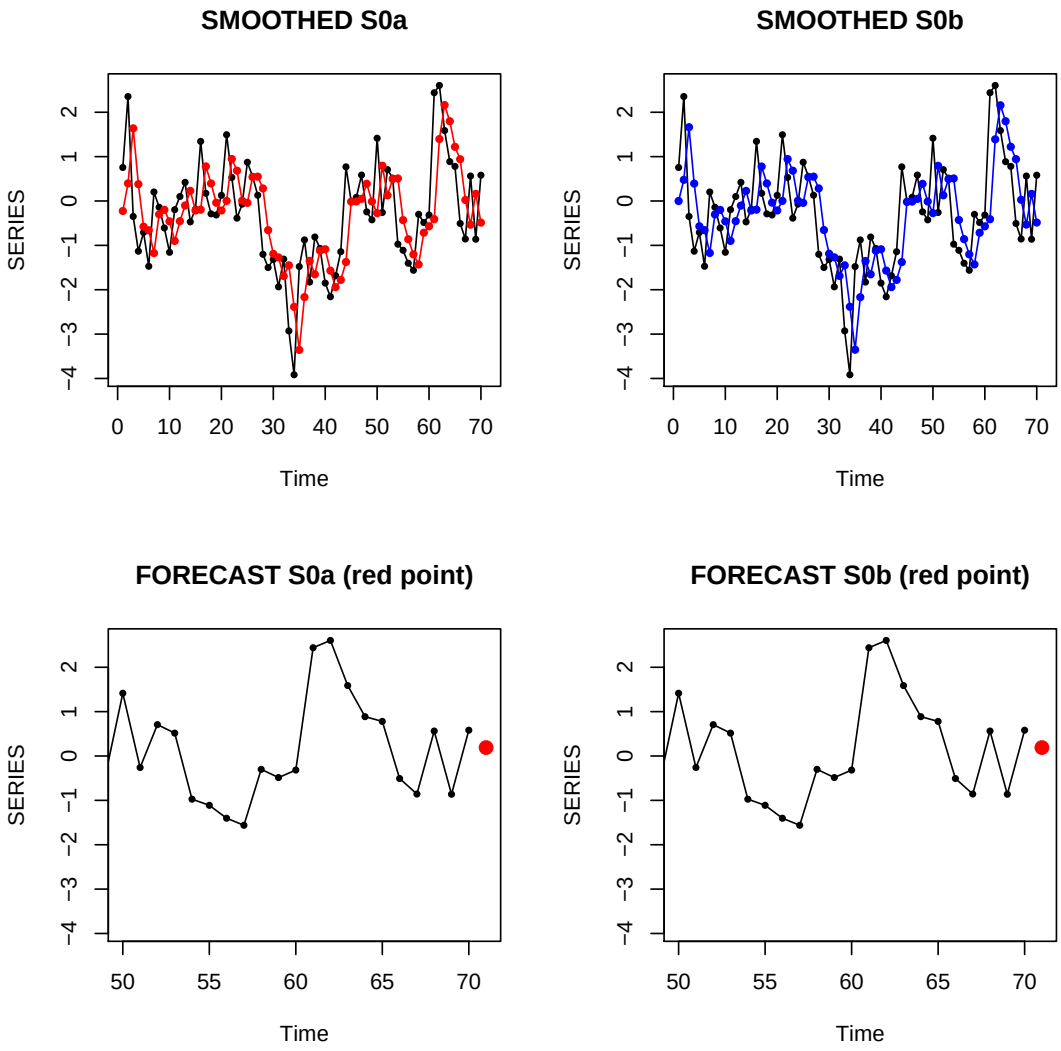

Figure 4: Upper panels: input time series (black) and the rolling forecasts (coloured); lower panels: forecast of the next observation (for two different initial conditions). Optimal smoothing parameters, for the initial conditions S0a and S0b are equal to 0.366 and 0.368, respectively.