# MAP 565
# Time series analysis : Lecture VI

François Roueff
http://perso.telecom-paristech.fr/~roueff/

Telecom ParisTech – École Polytechnique

January 20, 2015

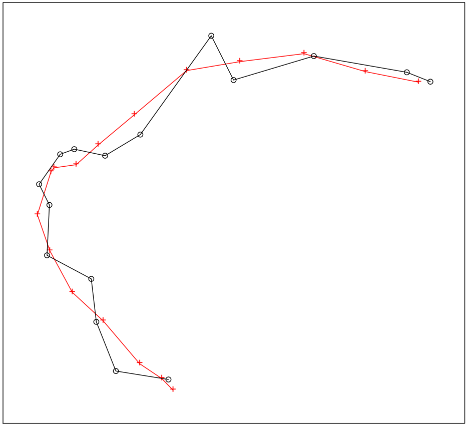
# Outline of the course

- ▷ Stochastic modeling
  - I Random processes.
  - II Spectral representation.
- ▷ Linear models
  - III Linear filtering, innovation process.
  - IV ARMA processes.
  - V Linear forecasting. $\swarrow$
- ▷ Statistical inference
  - VI Overview of goals and methods.
  - VII Asymptotic statistics in a dependent context.
- ▷ Non-linear models
  - VIII Standard models for financial time series.
  - IX Complements.

$\leftarrow$ : we are here.

## Outline of lectures VI

## State variables, observation variables



Noisy observations (black 'o') of state variables (red '+') evolving in the plan.

## Dynamic linear models (DLM)

- Simplest model for describing noisy observations of a multi-dimensional evolving system.
- Used in the 60's in Appolo's program.
- Many other applications since : radar, localization/tracking, econometrics ...

### Definition : DLM/linear state space models

A dynamic linear model is defined by

$$\begin{aligned}\text{a state equation : } \quad \mathbf{X}_t &= \Phi_t \mathbf{X}_{t-1} + \mathrm{A}_t \mathbf{u}_t + \mathbf{W}_t \;, \\ \text{an observation equation : } \quad \mathbf{Y}_t &= \Psi_t \mathbf{X}_t + \mathrm{B}_t \mathbf{u}_t + \mathbf{V}_t \;,\end{aligned}$$

where $\mathbf{Y}_t$ is the observed variable, $\mathbf{X}_t$ is the state variable, $\mathbf{u}_t$ is an exogenous (deterministic) input series variable. The matrices $\Phi_t$, $\Psi_t$, $\mathrm{A}_t$ and $\mathrm{B}_t$ are (known or unknown) parameters of the model, and $\left(\begin{bmatrix}\mathbf{W}_t & \mathbf{V}_t\end{bmatrix}\right)_t$ is a white noise sequence.

## Distribution assumptions

### Assumptions

(i) $(\mathbf{W}_t)_{t\in\mathbb{N}} \overset{\text{iid}}{\sim} \mathcal{N}(0, Q)$. matrix.

(ii) The initial state $\mathbf{X}_0 \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma_0)$.

(iii) $(\mathbf{V}_t)_{t\in\mathbb{N}} \overset{\text{iid}}{\sim} \mathcal{N}(0, R)$.

(iv) The initial state $\mathbf{X}_0$, the state noise $(\mathbf{W}_t)_{t\geq 1}$ and the observation noise $(\mathbf{V}_t)_{t\geq 1}$ are independent.

- ▷ Under this set of assumptions, the best predictor $\mathbb{E}\left[\mathbf{X}_t \middle| \mathbf{Y}_{1:s}\right]$ can be computed using the parameters $\Phi$, $\Psi$, $\mathrm{A}$, $\mathrm{B}$, $Q$, $\Sigma_0$ and $R$.
- ▷ The Gaussian assumption implies that

$$\mathbb{E}\left[\mathbf{X}_t \middle| \mathbf{Y}_{1:s}\right] = \operatorname{proj}\left(\mathbf{X}_t \middle| \operatorname{Span}\left(1, \mathbf{Y}_{1:s}\right)\right)$$

- ▷ On the other hand, the computation of $\operatorname{proj}\left(\mathbf{X}_t \middle| \operatorname{Span}\left(1, \mathbf{Y}_{1:s}\right)\right)$ does not require the Gaussian assumption.

## General goals

In the context of state space models, we call

- Filtering : the computation of $\mathbb{E}\left[\mathbf{X}_t \middle| \mathbf{Y}_{1:t}\right]$
- Forecasting : the computation of $\mathbb{E}\left[\mathbf{X}_s \middle| \mathbf{Y}_{1:t}\right]$ for $s > t$.
- Smoothing : the computation of $\mathbb{E}\left[\mathbf{X}_s \middle| \mathbf{Y}_{1:t}\right]$ for $s < t$.

We denote

$$\begin{aligned} \mathbf{X}_{s|t} &= \mathbb{E}\left[\mathbf{X}_s \middle| \mathbf{Y}_{1:t}\right] \\ \mathbf{Y}_{s|t} &= \mathbb{E}\left[\mathbf{Y}_s \middle| \mathbf{Y}_{1:t}\right] \\ \Sigma_{s|t} &= \mathbb{E}\left[\left(\mathbf{X}_s - \mathbf{X}_{s|t}\right)\left(\mathbf{X}_s - \mathbf{X}_{s|t}\right)^T\right] \\ &= \mathrm{Cov}\left(\mathbf{X}_s - \mathbf{X}_{s|t}\right) \end{aligned}$$

**Algorithm 1:** Kalman filter algorithm.

```
Data: Parameters $Q$, $R$ and $\mathrm{A}_t$, $\mathrm{B}_t$, $\Psi_t$ for $t = 1, \ldots, n$, initial conditions
      $\boldsymbol{\mu}$ and $\Sigma_0$, observations $\mathbf{Y}_t$ and exogenous input series $\mathbf{u}_t$, for
      $t = 1, \ldots, n$.
Result: Forecasting and filtering outputs $\mathbf{X}_{t|t-1}$, $\mathbf{X}_{t|t}$, and their
        autocovariance matrices $\Sigma_{t|t-1}$ and $\Sigma_{t|t}$ for $t = 1, \ldots, n$.
Initialization: set $\mathbf{X}_{0|0} = \boldsymbol{\mu}$ and $\Sigma_{0|0} = \Sigma_0$.
for $t = 1, 2, \ldots, n$ do
    Compute in this order
        $\mathbf{X}_{t|t-1} = \Phi_t \mathbf{X}_{t-1|t-1} + \mathrm{A}_t \mathbf{u}_t,$
        $\Sigma_{t|t-1} = \Phi_t \Sigma_{t-1|t-1} \Phi_t^T + Q,$
        $K_t = \Sigma_{t|t-1} \Psi_t^T [\Psi_t \Sigma_{t|t-1} \Psi_t^T + R]^{-1},$
        $\mathbf{X}_{t|t} = \mathbf{X}_{t|t-1} + K_t (\mathbf{Y}_t - \Psi_t \mathbf{X}_{t|t-1} - \mathrm{B}_t \mathbf{u}_t)\ ,$
        $\Sigma_{t|t} = [I - K_t \Psi_t] \Sigma_{t|t-1}\ .$
end
```

## Sketch of the proof

The matrix $K_t$ is called the Kalman gain matrix.

### Key point

One can recursively compute $\mathbf{X}_{t|t}$ and $\mathbf{X}_{t+1|t}$ by using that

$$\mathrm{Span}\left(1, \mathbf{Y}_{1:t}\right) = \mathrm{Span}\left(1, \mathbf{Y}_{1:t-1}\right) \overset{\perp}{\oplus} \mathrm{Span}\left(\mathbf{Y}_t - \mathbf{Y}_{t|t-1}\right) \ .$$

### Plus...

- Use the observation equation to derive $\mathbf{Y}_{t+1|t}$ from $\mathbf{X}_{t+1|t}$.
- Use the state equation to derive $\mathbf{X}_{t+1|t}$ from $\mathbf{X}_{t|t}$.
- The latter can be used to obtain $\mathbf{X}_{t+h|t}$, $h = 2, 3, 4, \ldots$

# Forecasting algorithm

```
Algorithm 2: Kalman forecasting algorithm.
Data: A forecasting lag $h$, parameters $Q$ and $\mathrm{A}_t$ for $t = n+1, \ldots, n+h$,
      and exogenous input series $\mathbf{u}_t$, for $t = n+1, \ldots, n+h$, Kalman
      filter output $\mathbf{X}_{n|n}$ and its error matrix $\Sigma_{n|n}$.
Result: Forecasting output $\mathbf{X}_{t|n}$ and their error matrices $\Sigma_{t|n}$ for
        $t = n+1, \ldots, n+h$
Initialization: set $k = 1$.
for $k = 1, 2, \ldots, h$ do
    Compute in this order
        $\mathbf{X}_{n+k|n} = \Phi_{n+k}\mathbf{X}_{n+k-1|n} + \mathrm{A}_{k+n}\mathbf{u}_{n+k}$ ,
        $\Sigma_{t|s} = \Phi_{n+k}\Sigma_{t-1|s}\Phi_{n+k}^T + Q$ .
end
```

## Smoothing algorithm

```
Algorithm 3: Rauch-Tung-Striebel smoother algorithm.
Data: Parameters $\Phi_t$ for $t = 1, \dots, n$, and exogenous input series $\mathbf{u}_t$, for
      $t = n+1, \dots, n+h$, Kalman filter output $\mathbf{X}_{t|t}$, $\mathbf{X}_{t|t-1}$, and their
      error matrices $\Sigma_{t|t}$ and $\Sigma_{t|t-1}$ for $t = 1, \dots, n$.
Result: Smoothing outputs $\mathbf{X}_{t|n}$, and their autocovariance matrices $\Sigma_{t|n}$
        for $t = n-1, n-2 \dots, 1$.
for $t = n, n-1, \dots, 2$ do
    Compute in this order
        $J_{t-1} = \Sigma_{t-1|t-1} \Phi_t^T \Sigma_{t|t-1}^{-1}$ ,
        $\mathbf{X}_{t-1|n} = \mathbf{X}_{t-1|t-1} + J_{t-1} \left( \mathbf{X}_{t|n} - \mathbf{X}_{t|t-1} \right)$ ,
        $\Sigma_{t-1|n} = \Sigma_{t-1|t-1} + J_{t-1} \left( \Sigma_{t|n} - \Sigma_{t|t-1} \right) J_{t-1}^T$ .
end
```

## Key point of the proof

We have

$$\mathbb{E}\left[\mathbf{X}_{t-1} \middle| \mathbf{Y}_{1:n}\right] = \mathbb{E}\left[\widetilde{\mathbf{X}}_{t-1} \middle| \mathbf{Y}_{1:n}\right] ,$$

where

$$\begin{aligned}\widetilde{\mathbf{X}}_{t-1} &= \mathbb{E}\left[\mathbf{X}_{t-1} \middle| \mathbf{Y}_{1:t-1}, \mathbf{X}_t - \mathbf{X}_{t|t-1}, \mathbf{V}_{t:n}, \mathbf{W}_{t+1:n}\right] \\ &= \mathbb{E}\left[\mathbf{X}_{t-1} \middle| \mathbf{Y}_{1:t-1}, \mathbf{X}_t - \mathbf{X}_{t|t-1}\right] \\ &= \mathbb{E}\left[\mathbf{X}_{t-1} \middle| \mathbf{Y}_{1:t-1}\right] + \mathbb{E}\left[\mathbf{X}_{t-1} \middle| \mathbf{X}_t - \mathbf{X}_{t|t-1}\right] \\ &= \mathbf{X}_{t-1|t-1} + \mathbb{E}\left[\mathbf{X}_{t-1} \middle| \mathbf{X}_t - \mathbf{X}_{t|t-1}\right] \\ &= \mathbf{X}_{t-1|t-1} + J_{t-1}\left(\mathbf{X}_t - \mathbf{X}_{t|t-1}\right) ,\end{aligned}$$

Hence

$$\mathbf{X}_{t-1|n} = \mathbf{X}_{t-1|t-1} + J_{t-1}\left(\mathbf{X}_{t|n} - \mathbf{X}_{t|t-1}\right) .$$

## Concluding remarks

- ▷ A byproduct of the Kalman algorithm is the computation of $\mathbf{Y}_{t+1|t}$ and $\mathrm{Cov}\left(\mathbf{Y}_{t+1} - \mathbf{Y}_{t+1|t}\right)$. From this we can derive the likelihood of the observations $\mathbf{Y}_{1:n}$, defined as the density of $\mathbf{Y}_{1:n}$ applied to the observations $\mathbf{Y}_{1:n}$.
- ▷ The Kalman algorithm can be performed online ($O(1)$ operations at each new observation $\mathbf{Y}_t$).
- ▷ ARMA processes can be described using dynamic linear models but the representations are not unique and some have correlated errors.
- ▷ The Kalman algorithm can be adapted to the case with correlated errors.

## Basic (important) definitions

**Definition : Data set**

A data set is a collection of values, say $X_{1:n} = X_1, \ldots, X_n$. Time series data sets are usually sampled from recorded measurements.

**Definition : Model**

A model is a collection of probability distributions. The data set is assumed to be distributed according to one of them.

**Definition : Statistic**

A statistic is any value which can be computed from the data.

## Raw data set

A “real life” time series $X_1, \dots, X_n$ is usually presented as

- ▷ a list of real values $X_1, \dots, X_n$ in a data or spreadsheet file,
- ▷ a corresponding list of dates (days, years, seconds...)
- ▷ or, equivalently, a starting date (in some unit), and a frequency. For instance :
  - ▷ a date in years and frequency$= 12$ corresponds to monthly data,
  - ▷ a date in years and frequency$= 4$ corresponds to quarterly data,
  - ▷ a date in days and frequency$= 1$ corresponds to daily data,
  - ▷ $\vdots$
- ▷ Remarks :
  - ▷ There may be missing values (usually expressed as 'NA')
  - ▷ In the case of multivariate time series, each variable usually corresponds to a column (so each row corresponds to a date).

## Example : US GNP data set

```
# Title:                 Gross National Product
# Source:                U.S. Department of Commerce
# Frequency:             Quarterly
DATE,VALUE
1947-01-01,238.1
1947-04-01,241.5
1947-07-01,245.6
1947-10-01,255.6
1948-01-01,261.7
1948-04-01,268.7
1948-07-01,275.3
1948-10-01,276.6
1949-01-01,271.3
1949-04-01,267.5
1949-07-01,268.9
```

$\vdots$

## First step : remove a trend

Consider a “real life” time series $X_1, \ldots, X_n$.
Recall the general decomposition of weakly stationary processes

$$X = \text{mean} + \text{deterministic process} + \text{purely non-det. process} .$$

- ▷ The mean is sometimes extended to a polynomial trend.
- ▷ The deterministic part is often modeled by a periodic harmonic process called the seasonal trend.

Then $X = D + Y$ where $D$ belongs to a finite dimensional space $V$ and $Y$ is a centered purely non-deterministic process.

Trends can be removed by

- ▷ Fitting the trend using least squares,
$$\widehat{D} = \operatorname*{argmin}_{d \in V} \sum_t |X_t - d_t|^2 .$$
- ▷ Or applying a well chosen FIR filter, such that $\mathrm{F}_\psi(D) = 0$ and thus
$$\mathrm{F}_\psi(X) = \mathrm{F}_\psi(Y) .$$

## Second step : choose a stochastic model

In time series analysis, one is interested in modeling the time dependence in the trend adjusted data $Y_1, \ldots, Y_n$.

This can be done by using

▷ a parametric model.

**Example**

$Y_1, \ldots, Y_n$ is the sample of a Gaussian ARMA$(p, q)$ model with (unknown) parameter $\vartheta = (\theta_1, \ldots, \theta_q, \phi_1, \ldots, \phi_p, \sigma^2)$.

▷ a non-parametric model.

**Example**

$Y_1, \ldots, Y_n$ is the sample of a centered stationary Gaussian process with (unknown) autocovariance $\gamma$ (or spectral density $f$).

## Third step : estimate parameters, test hypotheses

Once a model is fixed for $Y_1, \ldots, Y_n$, it can be used to

- ▷ Estimate a parameter of the model such as $\vartheta$, $\gamma(t)$, $\sigma^2$, $f$, …

  $\rightarrow$ Define an estimator, say $\widehat{\vartheta}_n$, which is a statistic based on the sample $Y_1, \ldots, Y_n$.

- ▷ Test hypotheses, for instance

$$H_0 = \{Y \text{ is white noise}\} \quad \text{against} \quad H_1 = \{Y \text{ is ARMA}(p,q)\}$$

  $\rightarrow$ Define a statistical test, say

$$\delta = \begin{cases} 1 & \text{if } T_n > t_n \ , \\ 0 & \text{otherwise} \ , \end{cases}$$

  where $T_n$ is statistic based on the sample $Y_1, \ldots, Y_n$ and $t_n$ is a threshold.

## Empirical estimation of the mean and the autocovariance

Consider a sample $X_1, \ldots, X_n$ assumed to be real valued and weakly stationary with mean $\mu$ and autocovariance $\gamma$.

They can be estimated using the empirical mean

$$\widehat{\mu}_n = \frac{1}{n} \sum_{k=1}^{n} X_k \ ,$$

and the empirical autocovariance

$$\widehat{\gamma}_n(h) = \frac{1}{n} \sum_{1 \leq k, k+h \leq n} (X_{k+h} - \widehat{\mu}_n)(X_k - \widehat{\mu}_n) \ .$$

The empirical autocorrelation is defined as

$$\widehat{\rho}_n(h) = \frac{\widehat{\gamma}_n(h)}{\widehat{\gamma}_n(0)} \ .$$

## Periodogram

The periodogram is defined, for all $\lambda \in \mathbb{R}$ by

$$I_n(\lambda) = \frac{1}{2\pi n} \left| \sum_{k=1}^{n} (X_k - \widehat{\mu}_n) \, \mathrm{e}^{-\mathrm{i}k\lambda} \right|^2 .$$

Then

- $\widehat{\gamma}_n$ is the sequence of Fourier coefficients of $I_n$,

$$\widehat{\gamma}_n(t) = \int_{\mathbb{T}} I_n(\lambda) \, \mathrm{e}^{\mathrm{i}t\lambda} \, \mathrm{d}\lambda \, , t \in \mathbb{Z} .$$

- Hence, by the Herglotz theorem, $\widehat{\gamma}_n$ is the autocovariance function of an MA process.

## Moment estimation (or $Z$-estimation)

Let $\boldsymbol{\theta} \in \mathbb{R}^q$ be an unknown parameter of the model.

The moment estimation of $\boldsymbol{\theta}$ is based on two ingredients :

- ▷ the inversion of a $\mathbb{R}^q \to \mathbb{R}^q$ mapping

$$\boldsymbol{\theta} \mapsto \mathrm{M}(\boldsymbol{\theta}) = \mathbb{E}[g(X)]$$

- ▷ empirical estimates of $\mathbb{E}[g(X)]$ from $X_{1:n}$, say $\widehat{g}_n$.

One then defines

$$\widehat{\theta}_n = \mathrm{M}^{-1}(\widehat{g}_n) \ .$$

## Example : AR(p) moment estimation

Let $X_{1:n}$ be a $n$-sample of a Gaussian AR($p$) model

$$X_t = \sum_{k=1}^{p} \phi_k X_{t-1} + Z_t \ ,$$

where $Z \sim \mathrm{WN}(0, \sigma^2)$ and set $\boldsymbol{\theta} = (\phi_1, \ldots, \phi_p, \sigma^2) \in \mathbb{R}^p \times \mathbb{R}_+^*$.

The moment estimation of $\boldsymbol{\theta}$ follows the following steps :

Step 1 Estimate $\gamma(k)$ for $k = 0, 1, \ldots, p$ with the empirical autocovariance function $\widehat{\gamma}_n$.

Step 2 Solve the Yule-Walker equations with $\widehat{\gamma}_n$ in place of $\gamma$.

One obtains an estimator $\widehat{\boldsymbol{\theta}}_n = (\widehat{\boldsymbol{\phi}}_n, \widehat{\sigma}_n^2)$, which, by construction, satisfies

$$1 - \sum_{k=1}^{n} \widehat{\boldsymbol{\phi}}_{k,n} z^k \neq 0 \quad \text{for all} \quad z \in \mathbb{C},\ |z| \leq 1 \ .$$

## Contrast estimation (or $M$-estimation)

Let $\boldsymbol{\theta} \in \mathbb{R}^q$ be an unknown parameter of the model.

Contrast estimation of $\boldsymbol{\theta}$ is based on two ingredients :

- ▷ a $\mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ mapping

$$(\boldsymbol{\theta}, \vartheta) \mapsto \mathrm{M}(\boldsymbol{\theta}, \vartheta) = \mathbb{E}[g(\vartheta, X)]$$

  such that

$$\boldsymbol{\theta} = \operatorname*{argmin}_{\vartheta} \mathrm{M}(\boldsymbol{\theta}, \vartheta)$$

- ▷ an empirical estimate of $\mathbb{E}[g(\vartheta, X)]$ from $X_{1:n}$ for all $\vartheta$, say $\widehat{g}_n(\vartheta)$.

One then defines

$$\widehat{\theta}_n = \operatorname*{argmin}_{\vartheta} \widehat{g}_n(\vartheta) \ .$$

## Maximum likelihood estimator (MLE)

A crucial example of contrast estimation is the Maximum Likelihood Estimator (MLE).

Dominated parametric model : Assume that $X_{1:n}$ admits a probability density $p(\cdot|\boldsymbol{\theta})$ on $\mathbb{R}^n$, parameterized by the unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^p$. That is, for any $g : \mathbb{R}^n \to \mathbb{R}$,

$$\mathbb{E}\left[g(X_{1:n})\right] = \int g(x_{1:n})\; p(x_{1:n}|\boldsymbol{\theta})\; \mathrm{d}x_1 \ldots \mathrm{d}x_n \;.$$

Then define a contrast estimator as above by setting

$$\widehat{g}_n(\vartheta) = -\log p(X_{1:n}|\vartheta) \;.$$

(called the negated log-likelihood function).

The MLE is thus defined as

$$\widehat{\theta}_n = \underset{\vartheta}{\operatorname{argmin}} - \log p(X_{1:n}|\vartheta) = \underset{\vartheta}{\operatorname{argmax}}\, p(X_{1:n}|\vartheta) \;.$$

## Example : Gaussian AR(1) likelihood function

Let $X_{1:n}$ be a $n$-sample of a Gaussian AR(1) model

$$X_t = \phi X_{t-1} + Z_t \ ,$$

where $Z \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$ and set $\boldsymbol{\theta} = (\phi, \sigma^2) \in (-1, 1) \times \mathbb{R}_+^*$.

Then, one can show that

$$\log p(x_{1:n}|\boldsymbol{\theta}) = \log p(x_1|\boldsymbol{\theta})\cancel{\log p(x_1|\boldsymbol{\theta})} - \frac{n-1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=2}^{n}(x_t - \phi\, x_{t-}$$

Discarding the first term, one gets

$$\widehat{\phi}_n = \operatorname*{argmin}_{\vartheta} \sum_{t=2}^{n} (X_t - \vartheta\, X_{t-1})^2 \quad \text{and} \quad \widehat{\sigma}_n^2 = \frac{1}{n-1}\sum_{t=2}^{n}(X_t - \widehat{\phi}_n X_{t-1})^2 \ .$$

## Remarks

- ▷ In the previous example, the Gaussian and AR($p$) assumptions are essential to obtain a closed form for the likelihood and the MLE. Numerical algorithms have to be considered in most of the other cases.
- ▷ Gaussian likelihood can be considered for non-Gaussian models. This is called Quasi-ML estimation.
- ▷ In "good" cases, the MLE is shown to be consistent and asymptotically normal,

$$\sqrt{n}(\widehat{\theta}_n - \theta) \Longrightarrow \mathcal{N}(0, \mathcal{I}^{-1}(\theta)) \,,$$

where $n\mathcal{I}(\theta)$ is the asymptotic equivalent of the Fisher information matrix ,

$$\mathcal{I}_n(\theta) = \mathrm{cov}_\theta \left(\partial_\theta \log p \left(X_{1:n} | \theta\right)\right) \sim n\,\mathcal{I}(\theta) \quad \text{as } n \to \infty \,.$$

```
################################################
#      Dynamic linear models in action         #
################################################
require(astsa)
require(plotrix)
# Setting
set.seed(1)
# wait after each plot
dx <-  0.5 # seconds
# number of points
n <- 2**10
# number of plots
nbpl <- 2**4
# std var of accelaration
sig <- 1
# std var of obs. noise
sigo <- 1
# 95% scale factor
scf <- qchisq(0.95,2)
# initial position, phi matrix
x <- as.matrix(rnorm(4,sd=sig))
phi <- matrix(c(1,0,0,0,0,1,0,0,1,0,1,0,0,1,0,1),nrow=4)
# display matrix
print(phi)
# random accelaration
acc <- matrix(rnorm(2*n,sd=sig),nrow=2)
acc <- rbind(matrix(rep(0,2*n),nrow=2),acc)
# generate state variables
for (k in 1:n){
  x <- cbind(x, phi %*% x[,k]+acc[,k])
}
# generate observation variables
y <- x[1:2,2:(n+1)] + matrix(rnorm(2*n,sd=sigo),nrow=2)
# plot setting
ylimites <- c(min(c(x[2,1:nbpl+1],y[2,1:nbpl]))-1,
              max(c(x[2,1:nbpl+1],y[2,1:nbpl]))+1)
```

```
xlimites <- c(min(c(x[1,1:nbpl+1],y[1,1:nbpl]))-1,
              max(c(x[1,1:nbpl+1],y[1,1:nbpl]))+1)
dev.new(width=10, height=5)
# displays state and obs variables
plot(x[1,2:nbpl+1],x[2,2:nbpl+1], pch='+',type='o',col=2,
     xlim=xlimites, ylim=ylimites,xlab='',ylab='',
     asp=1, xaxt='n',yaxt='n',
     main='State variables (positions) and observations')
lines(y[1,1:nbpl],y[2,1:nbpl], pch=8, type='o',col=1)
legend("topleft", c(expression(paste('State var.',
    X[t],sep=' ')), expression(paste('Obs. var.',
        Y[t],sep=' '))), col=c(2,1), text.col='black',
       lty=1, pch=c('+','*'))
# Filter and smooth (Ksmooth0 does both)
# convention:
# X_t = \phi X_{t-1} +  W_t\;,
# Y_t = A X_t + V_t\;,
# W_t iid N(0,cQ^T cQ), V_t iid N(0,cR^T cR),
# X_0 ~ N(mu0, Sigma0)
# observations y are such that time=row number
# needs 0 and identity matrix blocks
zz <- matrix(rep(0,4),nrow=2)
ii <- diag(1,nrow=2)
ks <- Ksmooth0(num=n, y=t(y), A=cbind(ii,zz), mu0 = rep(0,4),
                Sigma0 = sig**2*diag(1,nrow=4), Phi = phi,
                cQ = sig*rbind(cbind(zz,zz),cbind(zz,ii)),
                cR = sigo*ii)
########################################################################
#   Successive plots: filtering/forecasting/smoothing                  #
########################################################################
# BEGIN filter plots
for (k in (1:nbpl)){
  plot(x[1,2:(k+1)],x[2,2:(k+1)], pch='+', type='o',col=2,
       xlim=xlimites,ylim=ylimites,asp=1,xlab='',ylab='',
       xaxt='n',yaxt='n')
  lines(y[1,1:k],y[2,1:k],type='o',pch=8,col=1)
  lines(ks$xf[1,1,1:k],ks$xf[2,1,1:k],col=3)
  points(ks$xf[1,1,1:k],ks$xf[2,1,1:k],pch='x',col=3)
  points(ks$xf[1,1,k],ks$xf[2,1,k],pch='x',col=3)
  draw.circle(ks$xf[1,1,k],ks$xf[2,1,k],
              radius=sqrt(scf*ks$Pf[1,1,k]), border=3,lwd=2)
  legend("topleft", c(expression(paste('State var.',
      X[t],sep=' ')), expression(paste('Obs. var.',
          Y[t],sep=' ')), expression(paste('Kalman Filt.',
              hat(X)['t| t'],sep=' '))), col=c(2,1,3),
         text.col= 'black',lty=1, pch=c('+','*','x'),
         title=paste('Variables, t=',k-1,sep=' '))
  Sys.sleep(dx)
}
# All of them without state
plot(y[1,1:k],y[2,1:k], pch=8,type='o',col=1,xlim=xlimites,
     ylim=ylimites, asp=1,xlab='',ylab='',xaxt='n',yaxt='n')
lines(ks$xf[1,1,1:k],ks$xf[2,1,1:k],col=3)
points(ks$xf[1,1,1:k],ks$xf[2,1,1:k],pch='x',col=3)
points(ks$xf[1,1,k],ks$xf[2,1,k],pch='x',col=3)
legend("bottomleft", c(expression(paste('Obs. var.',
    Y[t],sep=' ')), expression(paste('Kalman Filt.',
        hat(X)['t | t'],sep=' '))), col=c(1,3),
       text.col= 'black',lty=1, pch=c(8,'x'),
       title=paste('Variables, t=',k-1,sep=' '))
# add confidence regions
for (k in (1:nbpl)){
  draw.circle(ks$xf[1,1,k],ks$xf[2,1,k],
              radius=sqrt(scf*ks$Pf[1,1,k]), border=3,lwd=2)
}
# All of them without obs
plot(x[1,2:(k+1)],x[2,2:(k+1)], type='o',col=2,
     xlim=xlimites,ylim=ylimites,asp=1,
     xlab='',ylab='',xaxt='n',yaxt='n')
lines(ks$xf[1,1,1:k],ks$xf[2,1,1:k],col=3)
points(ks$xf[1,1,1:k],ks$xf[2,1,1:k],pch='x',col=3)
points(ks$xf[1,1,k],ks$xf[2,1,k],pch='x',col=3)
```

```
legend("bottomleft", c(expression(paste('State var.',
    X[t],sep=' ')), expression(paste('Kalman Filt.',
        hat(X)['t|t'],sep=' '))), col=c(2,3),
       text.col= 'black',lty=1, pch=c('+','x'),
       title=paste('Variables, t=',k-1,sep=' '))
# add confidence regions
for (k in (1:nbpl)){
  draw.circle(ks$xf[1,1,k],ks$xf[2,1,k],
              radius=sqrt(scf*ks$Pf[1,1,k]), border=3,lwd=2)
}
# END filter plot
# BEGIN prediction plot
for (k in (1:nbpl)){
  plot(x[1,2:(k+1)],x[2,2:(k+1)], pch='+',col=2,type='o',
       xlim=xlimites,ylim=ylimites,xlab='',ylab='',
       asp=1,xaxt='n',yaxt='n')
  if (k>1)
    lines(y[1,1:(k-1)],y[2,1:(k-1)],type='o',pch=8,col=1)
  lines(ks$xp[1,1,1:k],ks$xp[2,1,1:k],col=3)
  points(ks$xp[1,1,1:k],ks$xp[2,1,1:k],pch='x',col=3)
  points(ks$xp[1,1,k],ks$xp[2,1,k],pch='x',col=3)
  draw.circle(ks$xp[1,1,k],ks$xp[2,1,k],
            radius=sqrt(scf*ks$Pp[1,1,k]), border=3,lwd=2)
  legend("bottomleft", c(expression(paste('State var.',
      X[t+1],sep=' ')), expression(paste('Obs. var.',
          Y[t],sep=' ')), expression(paste('Forecast.',
              hat(X)['t+1 | t'],sep=' '))),col=c(2,1,3),
         text.col= 'black',lty=1, pch=c('+','*','x'),
         title=paste('Variables, t=',k-1,sep=' '))
  Sys.sleep(dx)
}
# END prediction plot
# BEGIN smoothing plot
for (k in (nbpl:1)){
  plot(x[1,(k+1):(n+1)],x[2,(k+1):(n+1)], pch='+',col=2,
       type='o',xlim=xlimites,ylim=ylimites,xlab='',ylab='',
      asp=1,xaxt='n',yaxt='n')
  lines(y[1,],y[2,],type='o',pch=8,col=1)
  lines(ks$xs[1,1,k:n],ks$xs[2,1,k:n],col=3)
  points(ks$xs[1,1,k:n],ks$xs[2,1,k:n],pch='x',col=3)
  points(ks$xs[1,1,k],ks$xs[2,1,k],pch='x',col=3)
  draw.circle(ks$xs[1,1,k],ks$xs[2,1,k],
            radius=sqrt(scf*ks$Ps[1,1,k]),border=3,lwd=2)
    legend("bottomleft", c(expression(paste('State var.',
        X[t],sep=' ')),expression(paste('Obs. var.',
            Y[t],sep=' ')),expression(paste('Forecast.',
                hat(X)['t | n'],sep=' '))), col=c(2,1,3),
           text.col= 'black',lty=1, pch=c('+','*','x'),
           title=paste('Variables, t=',k-1,'n=',n,sep=' '))
  Sys.sleep(dx)
}
# END smoothing plot
# BEGIN plot of emp. errors
eo <- apply(x[1:2,2:(n+1)]-y,2,
            function(x){norm(as.matrix(x),'F')})
ef <- apply(x[1:2,2:(n+1)]-ks$xf[1:2,1,],2,
            function(x){norm(as.matrix(x),'F')})
es <- apply(x[1:2,2:(n+1)]-ks$xs[1:2,1,],2,
            function(x){norm(as.matrix(x),'F')})
ep <- apply(x[1:2,2:(n+1)]-ks$xp[1:2,1,],2,
            function(x){norm(as.matrix(x),'F')})
# smooth error functions in time
avnb <- min(floor(n/8),2**5)
for (term in c('o','f','s','p')){
  eval(parse(text=paste('sme',term,
               ' <- ts(filter(e',term,
               ', rep(1,avnb)/avnb))',sep='')))
}
# plot smoothed errors
ts.plot(smeo,smef,smes,smep,col=c(2,3,4,5))
legend('topright', c('Obs', 'Filt.', 'Smooth.',
                     'Forecast.'), col=c(2,3,4,5),
       text.col= 'black',lty='solid', title='Errors')
####################################################################
#   Parameter Estimation via maximum likelihood noisy AR(1)        #
####################################################################
rm(list = ls())
graphics.off()
set.seed(1)
# Generate causal AR(1)+noise
n <- 2**8; nbpl <- 2**6
# AR coeff
phi <- .6
# std var of innovations
sig <- 1
# std var of obs. noise
sigo <- 1
# initial x
x <- rnorm(1,sd=sig/sqrt(1-phi**2))
# innov. noise
eps <- rnorm(n,sd=sig)
# generate state variables
for (k in 1:n){
  x <- c(x, phi *x[k]+eps[k])
}
# generate observation variables
y <- x[2:(n+1)] + sigo*rnorm(n,sd=sigo)
# plot setting
dev.new(width=10, height=5)
# displays state and obs variables
ts.plot(cbind(x[2:(nbpl+1)],y[1:nbpl]), col=c(2,1),
     main='State variables (positions) and observations')
legend("bottomleft", c(expression(paste('State var.',
    X[t],sep=' ')), expression(paste('Obs. var.',
        Y[t],sep=' '))), col=c(2,1), text.col='black', lty=1)
# max number of observations
maxn <- length(y)
# obs. matrix (known)
```

```
A0=1
# Obs std dev (known)
cR0 = sigo
cQ0 = sig
# Likelihood function
Linn <- function(para){
    kf=Kfilter0(num=n, y=yest, A=A0, mu0 = 0,
        Sigma0 = sig**2/(1-para**2), Phi=para, cQ=cQ0, cR=cR0)
    return(kf$like/n)
}
# Initial Parameters
init.par = 0.0
# take first n observations
n <- 2**3
yest <- y[1:n]
# Likelihood maximisation
est <- optim(init.par, Linn, NULL, method="BFGS",
   hessian=TRUE, control=list(trace=1,REPORT=1))
# Maximizing parameter
lpar <- c(est$par)
ln <- c(n)
# increase n
for (n in seq(from=n+2, to=maxn, by=2)){
    yest <- y[1:n]
    print(n)
    init.par <- est$par
    est <- optim(init.par, Linn, NULL, method="BFGS",
                 hessian=TRUE, control=list(trace=1,REPORT=1))
    lpar <- c(lpar,est$par)
    ln <- c(ln,n)
}
# parameter estimates VS n
plot(ln,lpar,type='l',ylim=c(0,1),xlab='Nb of observations',ylab='Estimates')
abline(phi,0,col=2)
```