State Space Modeling in Macroeconomics and Finance Using SsfPack for S+FinMetrics

Eric Zivot, Jiahui Wang and Siem Jan Koopman *

University of Washington, Seattle
Ronin Capital LLC, Chicago
Free University, Amsterdam

August 4, 2002
This version: May 29, 2003

Abstract

This paper surveys some common state space models used in macroeconomics and finance and shows how to specify and estimate these models using the SsfPack algorithms implemented in the S-PLUS module S+FinMetrics. Examples include recursive regression models, time varying parameter models, exact autoregressive moving average models and calculation of the Beveridge-Nelson decomposition, unobserved components models, stochastic volatility models, and term structure models.

1 Introduction

The first version of SsfPack[1] appeared in 1998 and was developed further during the years that the last author was working with Jim Durbin on their 2001 textbook on state space methods. The fact that SsfPack functions are now a part of the S-PLUS software is partly due to Jim Durbin. He convinced Doug Martin that SsfPack would be very beneficial to S-PLUS. Indeed the persuasive arguments of Jim Durbin has initiated the development of SsfPack functions for S-PLUS as part of the S+FinMetrics module. It is therefore an honor for us, the developers of SsfPack for S+FinMetrics, to contribute to this volume with the presentation of various applications in economics and finance that require the use of SsfPack for S+FinMetrics in empirical research.

State space modeling in macroeconomics and finance has become widespread over the last decade. Textbook treatments of state space models are given in Harvey (1989), Harvey (1993), Hamilton (1994), West & Harrison (1997), Kim & Nelson (1999), Shumway &

*Financial support from the Royal Netherlands Academy of Arts and Sciences, and from the Gary Waterman Distinguished Scholar Fund at the University of Washington is gratefully acknowledged.
Emails: ezivot@u.washington.edu, jwang@svolatility.com, koopman@ssfpack.com.
Updates to the paper, including data and programs used in this paper, are available at Eric Zivot's website http://faculty.washington.edu/ezivot/ezresearch.htm.

[1]Information about SsfPack can be found at http://www.ssfpack.com.

Stoffer (2000),Durbin & Koopman (2001), and Chan (2002). However, until recently there has not been much flexible software for the statistical analysis of general models in state space form. A modern set of state space modeling tools are available in SsfPack which is a suite of C routines for carrying out computations involving the statistical analysis of univariate and multivariate models in state space form. The routines allow for a variety of state space forms from simple time invariant models to complicated time varying models. Functions are available to put standard models like autoregressive moving average and spline models in state space form. General routines are available for filtering, smoothing, simulation smoothing, likelihood evaluation, forecasting and signal extraction. Full details of the statistical analysis is provided in Durbin & Koopman (2001), and the reader is referred to the papers by Koopman, Shephard & Doornik (1999) and Koopman, Shephard & Doornik (2001) for technical details on the algorithms used in the SsfPack functions.

The SsfPack 2.3 routines are implemented in Ox and the SsfPack 3.0 routines are in Insightful's new S-PLUS module S+FinMetrics[2]. The implementation of the SsfPack functions in Ox is described in Koopman et al. (1999). Its implementation in S+FinMetrics is described in Zivot & Wang (2003, Chapter 14). This paper gives a selected overview of state space modeling with some economic and financial applications utilizing the S+FinMetrics/SsfPack functions.

This paper is organized as follows. Section 2 deals with (i) the general state space model and its specific SsfPack state space representation, (ii) descriptions of some functions for putting common time series models into state space form and (iii) the process of simulating observations from a given state space model. Section 3 summarizes the main algorithms used for the analysis of state space models. These include the Kalman filter, smoothing and forecasting. Further the estimation of the unknown parameters in a state space model is described. The remaining part of the paper presents several applications of state space modeling in economics and finance. These include recursive estimation of the capital asset pricing model with fixed and time varying parameters (section 4), maximum likelihood estimation of autoregressive moving average models and unobserved component models together with trend-cycle decompositions based on these models including the Beveridge-Nelson decomposition (section 5), estimation of a stochastic volatility model using Monte Carlo simulation techniques (section 6) and the estimation and analysis of a simple affine term structure model (section 7).

2 Linear State Space Representation

Many dynamic time series models in economics and finance may be represented in state space form. Some common examples are autoregressive moving average (ARMA) models, time varying regression models, dynamic linear models with unobserved components, discrete versions of continuous time diffusion processes, stochastic volatility (SV) models, non-parametric and spline regressions. The linear Gaussian state space model is represented as the system of equa-

[2] Ox is an object-oriented matrix programming language developed by Doornik (1999); more information is available at http://www.nuff.ox.ac.uk/users/doornik. S+FinMetrics is an S-PLUS module for the analysis of economic and financial time series. It was conceived by the first two authors and Doug Martin, and developed at Insightful Corporation. Its use and functionality is described in detail in Zivot & Wang (2003); more information is available at http://www.insightful.com/products/default.asp.

tions

$$\underset{m\times 1}{\boldsymbol{\alpha}_{t+1}} = \underset{m\times 1}{\mathbf{d}_t} + \underset{m\times m}{\mathbf{T}_t}\ \underset{m\times 1}{\boldsymbol{\alpha}_t} + \underset{m\times r}{\mathbf{H}_t}\ \underset{r\times 1}{\boldsymbol{\eta}_t} \tag{1}$$

$$\underset{N\times 1}{\boldsymbol{\theta}_t} = \underset{N\times 1}{\mathbf{c_t}} + \underset{N\times m}{\mathbf{Z_t}}\ \underset{m\times 1}{\boldsymbol{\alpha_t}} \tag{2}$$

$$\underset{N\times 1}{\mathbf{y}_t} = \underset{N\times 1}{\boldsymbol{\theta}_t} + \underset{N\times N}{\mathbf{G}_t}\ \underset{N\times 1}{\boldsymbol{\varepsilon}_t} \tag{3}$$

where $t = 1, \ldots, n$ and

$$\boldsymbol{\alpha}_1 \sim N(\mathbf{a}, \mathbf{P}), \qquad \boldsymbol{\eta}_t \sim \text{iid } N(0, \mathbf{I}_r), \qquad \boldsymbol{\varepsilon}_t \sim \text{iid } N(\mathbf{0}, \mathbf{I}_N), \tag{4}$$

and it is assumed that $E[\boldsymbol{\varepsilon}_t \boldsymbol{\eta}_t'] = \mathbf{0}$. The initial mean vector $\mathbf{a}$ and initial variance matrix $\mathbf{P}$ are fixed and known but that can be generalized. The state vector $\boldsymbol{\alpha}_t$ contains unobserved stochastic processes and unknown fixed effects and the transition equation (1) describes the evolution of the state vector over time using a first order Markov structure. The measurement equation (3) describes the vector of observations $\mathbf{y}_t$ in terms of the state vector $\boldsymbol{\alpha}_t$ through the signal $\boldsymbol{\theta}_t$ and a vector of disturbances $\boldsymbol{\varepsilon}_t$. It is assumed that the innovations in the transition equation and the innovations in the measurement equation are independent, but this assumption can be relaxed. The time varying deterministic matrices $\mathbf{T}_t, \mathbf{Z}_t, \mathbf{H}_t, \mathbf{G}_t$ are called system matrices and are usually sparse selection matrices. The vectors $\mathbf{d}_t$ and $\mathbf{c}_t$ contain fixed components and may be used to incorporate known effects or known patterns into the model; otherwise they are equal to zero.

The state space model (1) - (3) may be compactly expressed as

$$\begin{pmatrix} \boldsymbol{\alpha}_{t+1} \\ \mathbf{y}_t \end{pmatrix} = \underset{(m+N)\times 1}{\boldsymbol{\delta}_t} + \underset{(m+N)\times m}{\Phi_t}\ \underset{m\times 1}{\cdot\ \boldsymbol{\alpha}_t} + \underset{(m+N)\times 1}{\mathbf{u}_t} \tag{5}$$

$$\boldsymbol{\alpha}_1 \sim N(\mathbf{a}, \mathbf{P}), \mathbf{u}_t \sim \text{iid } N(\mathbf{0}, \boldsymbol{\Omega}_t) \tag{6}$$

where

$$\boldsymbol{\delta}_t = \begin{pmatrix} \mathbf{d}_t \\ \mathbf{c}_t \end{pmatrix}, \qquad \Phi_t = \begin{pmatrix} \mathbf{T}_t \\ \mathbf{Z}_t \end{pmatrix}, \qquad \mathbf{u}_t = \begin{pmatrix} \mathbf{H}_t \boldsymbol{\eta}_t \\ \mathbf{G}_t \boldsymbol{\varepsilon}_t \end{pmatrix}, \qquad \boldsymbol{\Omega}_t = \begin{pmatrix} \mathbf{H}_t \mathbf{H}_t' & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_t \mathbf{G}_t' \end{pmatrix}$$

The initial value parameters are summarized in the $(m+1) \times m$ matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} \mathbf{P} \\ \mathbf{a}' \end{pmatrix} \tag{7}$$

For multivariate models, i.e. $N > 1$, it is assumed that the $N \times N$ matrix $\mathbf{G}_t \mathbf{G}_t'$ is diagonal. This restriction can be circumvented by including the disturbance vector $\boldsymbol{\varepsilon}_t$ in the state vector $\boldsymbol{\alpha}_t$.

2.1 Initial Conditions

The variance matrix $\mathbf{P}$ of the initial state vector $\boldsymbol{\alpha}_1$ is assumed to be of the form

$$\mathbf{P} = \mathbf{P}_* + \kappa \mathbf{P}_\infty \tag{8}$$

where $\mathbf{P}_\infty$ and $\mathbf{P}_*$ are symmetric $m \times m$ matrices with ranks k_∞ and k_*, respectively, and κ represents the diffuse prior that is typically taken as a large scalar value, say, 10^7. The matrix $\mathbf{P}_*$ captures the covariance structure of the stationary components in the initial state vector, and the matrix $\mathbf{P}_\infty$ is used to specify the initial variance matrix for nonstationary components and fixed unknown effects. When the ith diagonal element of $\mathbf{P}_\infty$ is negative, the corresponding ith column and row of $\mathbf{P}_*$ are assumed to be zero, and the corresponding row and column of $\mathbf{P}_\infty$ will be taken into consideration. Further, the algorithms implement an "exact diffuse prior" approach as described in Durbin & Koopman (2001, Chapter 5) and, with more algorithmic detail, in Koopman et al. (2001).

2.2 State Space Representation in `S+FinMetrics/SsfPack`

State space models in `S+FinMetrics/SsfPack` utilize the compact representation (5) with initial value information (7). The following two examples describe its functionality.

Example 1 *State space representation of the local level model*

Consider the simple local level model for the stochastic evolution of the logarithm of an asset price y_t

$$\alpha_{t+1} = \alpha_t + \eta_t^*, \qquad \eta_t^* \sim \text{iid } N(0, \sigma_\eta^2) \tag{9}$$

$$y_t = \alpha_t + \varepsilon_t^*, \qquad \varepsilon_t^* \sim \text{iid } N(0, \sigma_\varepsilon^2) \tag{10}$$

$$\alpha_1 \sim N(a, P), \qquad E[\varepsilon_t^* \eta_t^*] = 0. \tag{11}$$

In this model, the observed value y_t is the sum of the unobservables α_t and ε_t^*. The level α_t is the state variable and represents the underlying signal. The transition equation (9) shows that the state evolves according to a random walk . The component ε_t^* represents random deviations (noise) from the signal that are assumed to be independent from the innovations to α_t. The strength of the signal relative to the noise is measured by $q = \sigma_\eta^2/\sigma_\varepsilon^2$.

The state space form (5) of the local level model has time invariant parameters

$$\boldsymbol{\delta} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \Phi = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \Omega = \begin{pmatrix} \sigma_\eta^2 & 0 \\ 0 & \sigma_\varepsilon^2 \end{pmatrix} \tag{12}$$

with errors $\sigma_\eta \eta_t = \eta_t^*$ and $\sigma_\varepsilon \varepsilon_t = \varepsilon_t^*$. Since the state variable α_t is $I(1)$[3], the unconditional distribution of the initial state α_1 does not have a finite variance. In this case, it is customary to set $a = E[\alpha_1] = 0$ and $P = \text{var}(\alpha_1) = \kappa$ in (11) with $\kappa \to \infty$ to reflect that no prior information is available. Using (8), the initial variance is specified with $a = 0$, $P_* = 0$ and $P_\infty = 1$. Therefore, the initial state matrix (7) for the local level model has the form

$$\Sigma = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \tag{13}$$

where -1 implies that $P_\infty = 1$.

In `S+FinMetrics/SsfPack` , a state space model is specified by creating either a list variable with components giving the minimum components necessary for describing a particular state

[3]The short-hand notation $I(1)$ is for a nonstationary variable that needs to be differenced once to become stationary, see Hamilton (1994) for further details.

```
> sigma.e = 1
> sigma.n = 0.5
> a1 = 0
> P1 = -1
> ssf.ll.list = list(mPhi=as.matrix(c(1,1)),
+ mOmega=diag(c(sigma.n^2,sigma.e^2)),
+ mSigma=as.matrix(c(P1,a1)))
> ssf.ll.list
$mPhi:            $mOmega:          $mSigma:
     [,1]              [,1] [,2]         [,1]
[1,]    1         [1,] 0.25    0    [1,]   -1
[2,]    1         [2,] 0.00    1    [2,]    0

> ssf.ll = CheckSsf(ssf.ll.list)
> class(ssf.ll)
[1] "ssf"
> names(ssf.ll)
 [1] "mDelta"  "mPhi"    "mOmega"  "mSigma"  "mJPhi"
 [6] "mJOmega" "mJDelta" "mX"      "cT"      "cX"
[11] "cY"      "cSt"
> ssf.ll
$mPhi:            $mOmega:          $mSigma:          $mDelta:
     [,1]              [,1] [,2]         [,1]              [,1]
[1,]    1         [1,] 0.25    0    [1,]   -1         [1,]    0
[2,]    1         [2,] 0.00    1    [2,]    0         [2,]    0
$mJPhi:           $mJOmega:                           $mJDelta:
[1] 0             [1] 0                               [1] 0
$mX:              $cT:              $cX:              $cY:              $cSt:
[1] 0             [1] 0             [1] 0             [1] 1             [1] 1
attr(, "class"):
[1] "ssf"
```

Listing 1: Local level model in `S+FinMetrics/SsfPack`

space form or by creating an "`ssf`" object. To illustrate, consider listing 1 where a list variable is created that contains the state space parameters in (12)-(13), with $\sigma_\eta = 0.5$ and $\sigma_\varepsilon = 1$. In the list variable `ssf.ll.list`, the component names match the state space form parameters in (5) and (7). This naming convention, summarized in Table 1, must be used for the specification of any valid state space model. An "`ssf`" object may be created from the list variable `ssf.ll.list` using the function `CheckSsf` as done in listing 1. The function `CheckSsf` takes a list variable with a minimum state space form, coerces the components to matrix objects and returns the full parameterization of a state space model used in many of the `S+FinMetrics/SsfPack` state space modeling functions.

Example 2 *State space representation of a time varying parameter regression model*

Consider a Capital Asset Pricing Model (CAPM) with time varying intercept and slope

$$\begin{aligned} y_t &= \alpha_t + \beta_{M,t} x_{M,t} + \nu_t, \quad && \nu_t \sim \text{iid } N(0, \sigma_\nu^2), \\ \alpha_{t+1} &= \alpha_t + \xi_t, && \xi_t \sim \text{iid } N(0, \sigma_\xi^2), \\ \beta_{M,t+1} &= \beta_{M,t} + \varsigma_t, && \varsigma_t \sim \text{iid } N(0, \sigma_\varsigma^2), \end{aligned} \tag{14}$$

State Space Parameter	List Component Name
$\boldsymbol{\delta}$	`mDelta`
Φ	`mPhi`
Ω	`mOmega`
Σ	`mSigma`

Table 1: `S+FinMetrics/SsfPack` state space form list components

where y_t denotes the return on an asset in excess of the risk free rate, and $x_{M,t}$ denotes the excess return on a market index. In this model, both the abnormal excess return α_t and asset risk $\beta_{M,t}$ are allowed to vary over time following a random walk specification. Let $\boldsymbol{\alpha}_t = (\alpha_t, \beta_{M,t})'$, $\mathbf{x}_t = (1, x_{M,t})'$, $\mathbf{H}_t = \text{diag}(\sigma_\xi, \sigma_\varsigma)'$ and $G_t = \sigma_\nu$. Then the state space form (5) of (14) is

$$\begin{pmatrix} \boldsymbol{\alpha}_{t+1} \\ y_t \end{pmatrix} = \begin{pmatrix} \mathbf{I}_2 \\ \mathbf{x}_t' \end{pmatrix} \boldsymbol{\alpha}_t + \begin{pmatrix} \mathbf{H}_t \boldsymbol{\eta}_t \\ G_t \varepsilon_t \end{pmatrix}$$

and has parameters

$$\Phi_t = \begin{pmatrix} \mathbf{I}_2 \\ \mathbf{x}_t' \end{pmatrix}, \qquad \Omega = \begin{pmatrix} \sigma_\xi^2 & 0 & 0 \\ 0 & \sigma_\varsigma^2 & 0 \\ 0 & 0 & \sigma_\nu^2 \end{pmatrix} \tag{15}$$

Since $\boldsymbol{\alpha}_t$ is $I(1)$ the initial state vector $\boldsymbol{\alpha}_1$ requires an infinite variance so it is customary to set $\mathbf{a} = \mathbf{0}$ and $\mathbf{P} = \kappa \mathbf{I}_2$ with $\kappa \to \infty$. Using (8), the initial variance is specified with $\mathbf{P}_* = \mathbf{0}$ and $\mathbf{P}_\infty = \mathbf{I}_2$. Therefore, the initial state matrix (7) for the time varying CAPM has the form

$$\Sigma = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{pmatrix}.$$

The state space parameter matrix Φ_t in (15) has a time varying system element $\mathbf{Z}_t = \mathbf{x}_t'$. In `S+FinMetrics/SsfPack`, the specification of this time varying element in Φ_t requires an index matrix $\mathbf{J}_\Phi$ and a data matrix $\mathbf{X}$ to which the indices in $\mathbf{J}_\Phi$ refer. The index matrix $\mathbf{J}_\Phi$ must have the same dimension as Φ_t. The elements of $\mathbf{J}_\Phi$ are all set to -1 except the elements for which the corresponding elements of Φ_t are time varying. The non-negative index value indicates the column of the data matrix $\mathbf{X}$ which contains the time varying values[4]. For example, in the time varying CAPM, the index matrix $\mathbf{J}_\Phi$ has the form

$$\mathbf{J}_\Phi = \begin{pmatrix} -1 & -1 \\ -1 & -1 \\ 1 & 2 \end{pmatrix}.$$

The specification of the state space form for the time varying CAPM requires values for the variances σ_ξ^2, σ_ς^2, and σ_ν^2 as well as a data matrix $\mathbf{X}$ whose rows correspond with $\mathbf{Z}_t = \mathbf{x}_t' = (1, r_{M,t})$. For example, let $\sigma_\xi^2 = (0.01)^2$, $\sigma_\varsigma^2 = (0.05)^2$ and $\sigma_\nu^2 = (0.1)^2$ and construct the data

[4]When there are time varying elements in $\mathbf{T}_t$, the initial values of these elements in the specification of Φ_t should not be set to $-1, 0$, or 1 due to the way `SsfPack` handles sparse matrices. We suggest setting these elements equal to 0.5 so that the sparse matrix operations know that there should be a non-trivial number there.

matrix $\mathbf{X}$ using the excess return data in the S+FinMetrics "timeSeries" excessReturns.ts. The state space form is then created as in listing 2. Notice in the specification of Φ_t the values associated with $\mathbf{x}_t'$ in the third row are set to zero. In the index matrix $\mathbf{J}_\Phi$, the (3,1) element is 1 and the (3,2) element is 2 indicating that the data for the first and second columns of $\mathbf{x}_t'$ come from the first and second columns of the component mX, respectively.

```
> X.mat = cbind(1,as.matrix(seriesData(excessReturns.ts[,"SP500"])))
> Phi.t = rbind(diag(2),rep(0,2))
> Omega = diag(c((.01)^2,(.05)^2,(.1)^2))
> J.Phi = matrix(-1,3,2)
> J.Phi[3,1] = 1
> J.Phi[3,2] = 2
> Sigma = -Phi.t
> ssf.tvp.capm = list(mPhi=Phi.t,
+ mOmega=Omega,
+ mJPhi=J.Phi,
+ mSigma=Sigma,
+ mX=X.mat)
> ssf.tvp.capm
$mPhi:                  $mOmega:
     [,1] [,2]                [,1]   [,2] [,3]
[1,]    1    0         [1,] 0.0001 0.0000 0.00
[2,]    0    1         [2,] 0.0000 0.0025 0.00
[3,]    0    0         [3,] 0.0000 0.0000 0.01
$mJPhi:                 $mSigma:
     [,1] [,2]               [,1] [,2]
[1,]   -1   -1          [1,]   -1    0
[2,]   -1   -1          [2,]    0   -1
[3,]    1    2          [3,]    0    0
$mX:
numeric matrix: 131 rows, 2 columns.
         SP500
  1 1  0.002839
 ...
131 1 -0.0007466
```

Listing 2: The CAPM model in S+FinMetrics/SsfPack

In the general state space model (5), it is possible that all of the system matrices $\boldsymbol{\delta}_t$, Φ_t and Ω_t have time varying elements. The corresponding index matrices $\mathbf{J}_\delta$, $\mathbf{J}_\Phi$ and $\mathbf{J}_\Omega$ indicate which elements of the matrices $\boldsymbol{\delta}_t$, Φ_t and Ω_t are time varying and the data matrix $\mathbf{X}$ contains the time varying components. The naming convention for these components is summarized in Table 2.

2.3 Model Specification

S+FinMetrics/SsfPack has functions for the creation of the state space representation of some common time series models. These functions and models are summarized in Table 3. For other models, the system matrices can be created within S-PLUS and the function CheckSsf.

A complete description of the underlying statistical models and use of these functions is given in Zivot & Wang (2003, Chapter 14). The use of some of these functions will be illustrated in

Parameter Index Matrix	List Component Name
$\mathbf{J}_\delta$	mJDelta
$\mathbf{J}_\Phi$	mJPhi
$\mathbf{J}_\Omega$	mJOmega
Time Varying Component Data Matrix	**List Component Name**
$\mathbf{X}$	mX

Table 2: S+FinMetrics/SsfPack time varying state space components

Function	Description
GetSsfReg	Create state space form for linear regression model
GetSsfArma	Create state space form for stationary and invertible ARMA model
GetSsfRegArma	Create state space form for linear regression model with ARMA errors
GetSsfStsm	Create state space form for structural time series model
GetSsfSpline	Create state space form for nonparametric cubic spline model

Table 3: S+FinMetrics/SsfPack functions for creating common state space models

the applications to follow.

2.4 Simulating Observations

Once a state space model has been specified, it is often interesting to draw simulated values from the model. Simulation from a given state space model is also necessary for Monte Carlo and bootstrap exercises. The S+FinMetrics/SsfPack function SsfSim may be used for such a purpose. The arguments expected from SsfSim are as illustrated in listing 3. The variable ssf represents either a list with components giving a minimal state space form or a valid "ssf" object, n is the number of simulated observations, mRan is user-specified matrix of disturbances, and a1 is the initial state vector.

Example 3 *Simulating observations from the local level model*

The code in listing 3 generates 250 observations on the state variable α_{t+1} and observations y_t in the local level model (9) - (11). The function SsfSim returns a matrix containing the simulated state variables α_{t+1} and observations y_t. These values are illustrated in Figure 1.

```
> set.seed(112)
> ll.sim = SsfSim(ssf.ll.list,n=250)
> class(ll.sim)
[1] "matrix"
> colIds(ll.sim)
[1] "state"    "response"
```

Listing 3: Simulating observations and states in S+FinMetrics/SsfPack

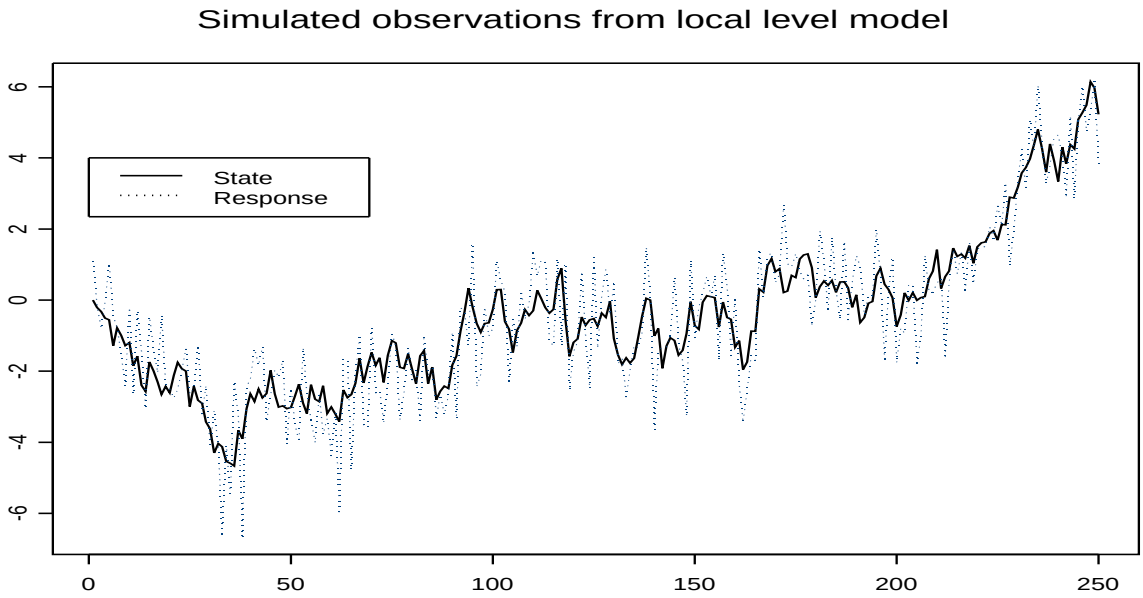

Figure 1: Simulated values from local level model created using the `S+FinMetrics` function `SsfSim`.

3 Overview of Algorithms

In this section we review the algorithms that are considered in `S+FinMetrics/SsfPack`. The most relevant equations are reproduced but the review does not aim to be complete and to provide the detail of implementation. The derivations of the algorithms can be found in, for example, Harvey (1989), Shumway & Stoffer (2000) and Durbin & Koopman (2001). Details of numerical implementation are reported in Koopman et al. (1999) and Koopman et al. (2001) unless indicated otherwise.

The `S+FinMetrics/SsfPack` functions for computing the algorithms described below are summarized in Table 4. All of the functions except `KalmanSmo` have an optional argument `task` which controls the task to be performed by the function. The values of the argument `task` with brief descriptions are given in Table 5.

3.1 Kalman Filter

The Kalman filter is a recursive algorithm for the evaluation of moments of the normally distributed state vector $\boldsymbol{\alpha}_{t+1}$ conditional on the observed data $\mathbf{Y}_t = (y_1, \ldots, y_t)$. Let $\mathbf{a}_t = E[\boldsymbol{\alpha}_t|\mathbf{Y}_{t-1}]$ denote the conditional mean of $\boldsymbol{\alpha}_t$ based on information available at time $t-1$ and let $\mathbf{P}_t = \text{var}(\boldsymbol{\alpha}_t|\mathbf{Y}_{t-1})$ denote the conditional variance of α_t. The filtering or updating equations of the Kalman filter compute $\mathbf{a}_{t|t} = E[\boldsymbol{\alpha}_t|\mathbf{Y}_t]$ and $\mathbf{P}_{t|t} = \text{var}(\boldsymbol{\alpha}_t|\mathbf{Y}_t)$, as well as the measurement equation innovation or one-step ahead prediction error $\mathbf{v}_t = \mathbf{y}_t - \mathbf{c}_t - \mathbf{Z}_t\mathbf{a}_t$ and prediction error variance $\mathbf{F}_t = \text{var}(\mathbf{v}_t)$. The prediction equations of the Kalman filter compute $\mathbf{a}_{t+1}$ and $\mathbf{P}_{t+1}$.

The function `KalmanFil` implements the Kalman filter recursions in a computationally efficient way. The output of `KalmanFil` is primarily used by other functions, but it can also

be used directly to evaluate the appropriateness of a given state space model through the analysis of the innovations $\mathbf{v}_t$. The function `SsfMomentEst` computes the filtered state and response estimates from a given state space model and observed data with the optional argument `task="STFIL"`. Predicted state and response estimates are computed using `SsfMomentEst` with the optional argument `task="STPRED"`.

When the initial conditions $\mathbf{a}_1$ and $\mathbf{P}_1$ can be properly defined, the Kalman filter can be used for prediction and filtering. For example, for a stationary ARMA process in state space, the initial conditions of the state vector are obtained from the unconditional mean and variance equations of the ARMA process. For nonstationary processes however the initial conditions can not be properly defined for the obvious reasons. The same applies to fixed unknown state elements. In such cases the applicable initial state element is taken as diffuse which implies that its mean is arbitrary, usually zero, and its variance is infinity.

3.2 Kalman Smoother

The Kalman filtering algorithm is a forward recursion which computes one-step ahead estimates $\mathbf{a}_{t+1}$ and $\mathbf{P}_{t+1}$ based on $\mathbf{Y}_t$ for $t = 1, \ldots, n$. The Kalman smoothing algorithm is a backward recursion which computes the mean and variance of specific conditional distributions based on the full data set $\mathbf{Y}_n = (y_1, \ldots, y_n)$. The state smoothing residuals are denoted $\mathbf{r}_t$ and the response smoothing residuals are denoted by $\mathbf{e}_t$.

The function `KalmanSmo` implements the Kalman smoother recursions in a computationally efficient way and it computes the smoothing residuals together with the diagonal elements of their variances. The output of `KalmanSmo` is primarily used by other functions for computing smoothed estimates of the state and disturbance vectors but it can also be used to compute score information for parameter estimation and to evaluate diagnostics for the detection of outliers and structural breaks .

3.3 Smoothed State, Response and Disturbance Estimates

The smoothed estimates of the state vector $\boldsymbol{\alpha}_t$ and its variance matrix are denoted $\hat{\boldsymbol{\alpha}}_t = E[\boldsymbol{\alpha}_t|\mathbf{Y}_n]$ (or $\mathbf{a}_{t|n}$) and $\mathrm{var}(\boldsymbol{\alpha}_t|\mathbf{Y}_n)$, respectively. The smoothed estimate $\hat{\boldsymbol{\alpha}}_t$ is the optimal estimate of $\boldsymbol{\alpha}_t$ using all available information $\mathbf{Y}_n$, whereas the filtered estimate $\mathbf{a}_{t|t}$ is the optimal estimate only using information available at time $t, \mathbf{Y}_t$. The computation of $\hat{\boldsymbol{\alpha}}_t$ and its variance from the Kalman smoother algorithm is described in Durbin & Koopman (2001). The smoothed estimate of the response $\mathbf{y}_t$ and its variance are computed using

$$\hat{\mathbf{y}}_t = \mathbf{c}_t + \mathbf{Z}_t\hat{\boldsymbol{\alpha}}_t, \qquad \mathrm{var}(\mathbf{y}_t|\mathbf{Y}_n) = \mathbf{Z}_t\mathrm{var}(\boldsymbol{\alpha}_t|\mathbf{Y}_n)\mathbf{Z}_t'. \tag{16}$$

Smoothed estimates of states and responses may be computed using the functions `SsfCondDens` and `SsfMomentEst` with the optional argument `task="STSMO"`. The function `SsfCondDens` only computes the smoothed states and responses whereas `SsfMomentEst` also computes the associated variances.

The smoothed disturbance estimates are the estimates of the measurement equations innovations $\boldsymbol{\varepsilon}_t$ and transition equation innovations $\boldsymbol{\eta}_t$ based on all available information $\mathbf{Y}_n$, and are denoted $\hat{\boldsymbol{\varepsilon}}_t = E[\boldsymbol{\varepsilon}_t|\mathbf{Y}_n]$ (or $\varepsilon_{t|n}$) and $\hat{\boldsymbol{\eta}}_t = E[\boldsymbol{\eta}_t|\mathbf{Y}_n]$ (or $\eta_{t|n}$), respectively. The computation of $\hat{\boldsymbol{\varepsilon}}_t$ and $\hat{\boldsymbol{\eta}}_t$ from the Kalman smoother algorithm is described in Durbin & Koopman (2001). These smoothed disturbance estimates can be useful for parameter estimation by maximum

likelihood and for diagnostic checking. The functions `SsfCondDens` and `SsfMomentEst`, with the optional argument `task="DSSMO"`, may be used to compute smoothed estimates of the measurement and transition equation disturbances. The function `SsfCondDens` only computes the smoothed states and responses whereas `SsfMomentEst` also computes the associated variances.

3.4 Missing Values and Forecasting

The Kalman filter prediction equation produces one-step ahead predictions of the state vector, $\mathbf{a}_{t+1} = E[\boldsymbol{\alpha}_{t+1}|\mathbf{Y}_t]$, along with prediction variance matrices $\mathbf{P}_{t+1}$. In the Kalman filter recursions, if there are missing values in $\mathbf{y}_t$ then $\mathbf{v}_t = \mathbf{0}$, $\mathbf{F}_t^{-1} = \mathbf{0}$ and $\mathbf{K}_t = \mathbf{0}$. This allows out-of-sample forecasts of $\boldsymbol{\alpha}_t$ and $\mathbf{y}_t$ to be computed from the updating and prediction equations. Out-of-sample predictions, together with associated mean square errors, can be computed from the Kalman filter prediction equations by extending the data set $\mathbf{y}_1, \ldots, \mathbf{y}_n$ with a set of missing values. When y_τ is missing, the Kalman filter reduces to the prediction step described above. As a result, a sequence of m missing values at the end of the sample will produce a set of h-step ahead forecasts for $h = 1, \ldots, m$. Forecasts with their variances based on a given state space model may be computed using the function `SsfMomentEst` with the optional argument `task="STPRED"`.

3.5 Simulation Smoothing

The joint simulation of state and response vectors $\boldsymbol{\alpha}_t$ and $\mathbf{y}_t$, $t = 1, \ldots, n$, or disturbance vectors $\boldsymbol{\eta}_t$ and $\boldsymbol{\varepsilon}_t$, $t = 1, \ldots, n$, conditional on the observations $\mathbf{Y}_n$ is called simulation smoothing. Simulation smoothing is useful for evaluating the appropriateness of a proposed state space model, for the Bayesian analysis of state space models using Markov chain Monte Carlo (MCMC) techniques and for the evaluation of the likelihood function using importance sampling techniques. Initial work on simulation smoothing has been developed by Fruhwirth-Schnatter (1994),Carter & Kohn (1994) and de Jong & Shephard (1995). The recent simulation smoothing method of Durbin & Koopman (2002) is used for the `S+FinMetrics/SsfPack` implementation. The function `SimSmoDraw` generates random draws from the distributions of the state and response variables (argument `task="STSIM"`) or from the distributions of the state and response disturbances (argument `task="DSSIM"`).

3.6 Prediction Error Decomposition and Log-Likelihood

The prediction error decomposition of the log-likelihood function for the unknown parameters $\boldsymbol{\varphi}$ of a state space model may be conveniently computed using the output of the Kalman filter

$$\begin{aligned} \ln L(\boldsymbol{\varphi}|Y_n) &= \sum_{t=1}^{n} \ln f(\mathbf{y}_t|\mathbf{Y}_{t-1};\boldsymbol{\varphi}) \\ &= -\frac{nN}{2}\ln(2\pi) - \frac{1}{2}\sum_{t=1}^{n}\left(\ln|\mathbf{F}_t| + \mathbf{v}_t'\mathbf{F}_t^{-1}\mathbf{v}_t\right) \end{aligned} \tag{17}$$

where $f(\mathbf{y}_t|\mathbf{Y}_{t-1};\boldsymbol{\varphi})$ is a conditional Gaussian density implied by the state space model (1) - (2). The vector of prediction errors $\mathbf{v}_t$ and prediction error variance matrices $\mathbf{F}_t$ are computed from the Kalman filter recursions.

Function	Description	Tasks
KalmanIni	Initialize Kalman filter	All
KalmanFil	Kalman filtering and likelihood eval	All
KalmanSmo	Kalman smoothing	None
SsfCondDens	Conditional density/mean calculation	STSMO,DSSMO
SsfMomentEst	Moment estimation and smoothing	STFIL,STPRED,STSMO,DSSMO
SimSmoDraw	Simulation smoother draws	STSIM,DSSIM
SsfLoglike	Log-likelihood of state space model	None
SsfFit	Estimate state space model parameters	None

Table 4: General S+FinMetrics/SsfPack state space functions

Task	Description
KFLIK	Kalman filtering and loglikelihood evaluation
STFIL	State filtering
STPRED	State prediction
STSMO	State smoothing
DSSMO	Disturbance smoothing
STSIM	State simulation
DSSIM	Disturbance simulation

Table 5: Task argument to S+FinMetrics/SsfPack functions

The functions KalmanFil and SsfLoglike may be used to evaluate the prediction error decomposition of the log-likelihood function for a given set of parameters $\boldsymbol{\varphi}$. The S+FinMetrics function SsfFit, which evaluates (17) using SsfLoglike, may be used to find the maximum likelihood estimators of the unknown parameters $\boldsymbol{\varphi}$ using the S-PLUS optimization function nlminb[5].

4 The Capital Asset Pricing Model

This section illustrates the use of the S+FinMetrics/SsfPack state space modeling and analysis functions for an empirical analysis of the Capital Asset Pricing Model (CAPM).

4.1 Recursive Least Squares

Consider the typical CAPM regression model

$$y_t = \alpha + \beta_M x_{M,t} + \xi_t, \qquad \xi_t \sim \text{iid } N(0, \sigma_\xi^2)$$

where y_t denotes the return on an asset in excess of the risk free rate, and $x_{M,t}$ is the excess return on a market index. The state space representation is given by

$$\begin{pmatrix} \boldsymbol{\alpha}_{t+1} \\ y_t \end{pmatrix} = \begin{pmatrix} \mathbf{I}_k \\ \mathbf{x}_t' \end{pmatrix} \boldsymbol{\alpha}_t + \begin{pmatrix} \mathbf{0} \\ \sigma_\xi \xi_t \end{pmatrix} \tag{18}$$

[5]There are several optimization algorithms available in S-PLUS besides nlminb. Most notable are the functions nlmin, ms and optim (in the MASS library provided by Venables and Ripley).

with $\mathbf{x}_t = (1, x_{M,t})'$ and the state vector satisfies

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t = \boldsymbol{\beta} = (\alpha, \beta_M)'.$$

The state space system matrices are $\mathbf{T}_t = \mathbf{I}_k$, $\mathbf{Z}_t = \mathbf{x}_t'$, $\mathbf{G}_t = \sigma_\xi$ and $\mathbf{H}_t = 0$ and may be compared with the ones in Example 2. The monthly excess return data on Microsoft and the S&P 500 index over the period February, 1990 through December, 2000 are used as in listing 2. The state space form for the CAPM with fixed regressors may be created using the function `GetSsfReg` as in listing 4.

An advantage of analyzing the linear regression model in state space form is that recursive least squares (RLS) estimates of the regression coefficient vector $\boldsymbol{\beta}$ are readily computed from the Kalman filter. The RLS estimates are based on estimating the model

$$y_t = \boldsymbol{\beta}_t' \mathbf{x}_t + \xi_t, \qquad t = 1, \ldots, n \tag{19}$$

by least squares recursively for $t = 3, \ldots, n$ giving $n - 2$ least squares (RLS) estimates $(\hat{\boldsymbol{\beta}}_3, \ldots, \hat{\boldsymbol{\beta}}_T)$. If $\boldsymbol{\beta}$ is constant over time then the recursive estimates $\hat{\boldsymbol{\beta}}_t$ should quickly settle down near a common value. If some of the elements in $\boldsymbol{\beta}$ are not constant then the corresponding RLS estimates should show instability. Hence, a simple graphical technique for uncovering parameter instability is to plot the RLS estimates $\hat{\beta}_{it}$ $(i = 1, 2)$ and look for instability in the plots.

The RLS estimates are simply the filtered state estimates from the model (18), and may be computed using the function `SsfMomentEst` with the optional argument `task="STFIL"`:

The component `state.moment` contains the filtered state estimates $\mathbf{a}_{t|t}$ for $t = 1, \ldots, n$, which are equal to the RLS estimates of the linear regression coefficients, and the component `response.moment` contains the filtered response estimates $y_{t|t}$. The first column of the component `state.moment` contains the RLS estimates of α, and the second column contains the RLS estimates of β_M. The last row contains the full sample least squares estimates of α and β_M. The RLS estimates can be visualized using the generic `plot` method for objects of the class `SsfMomentEst`. The resulting plot is illustrated in Figure 2. Notice that the RLS estimates of β_M seem fairly constant whereas the RLS estimates of α do not.

4.2 Tests for constant parameters

Formal tests for structural stability of the regression coefficients, such as the CUSUM test of Brown, Durbin & Evans (1975), may be computed from the standardized 1-step ahead recursive residuals

$$w_t = \frac{v_t}{\sqrt{f_t}} = \frac{y_t - \hat{\boldsymbol{\beta}}_{t-1}' \mathbf{x}_t}{\sqrt{f_t}}$$

where f_t is an estimate of the recursive error variance

$$\sigma^2 \left[1 + \mathbf{x}_t' (\mathbf{X}_{t-1}' \mathbf{X}_{t-1})^{-1} \mathbf{x}_t\right]$$

and $\mathbf{X}_t$ is the $(t \times k)$ matrix of observations on $\mathbf{x}_s$ using data from $s = 1, \ldots, t$. These standardized recursive residuals result as a by-product of the Kalman filter recursions and may be extracted using the `S+FinMetrics/SsfPack` function `KalmanFil` as in listing 5. Diagnostic

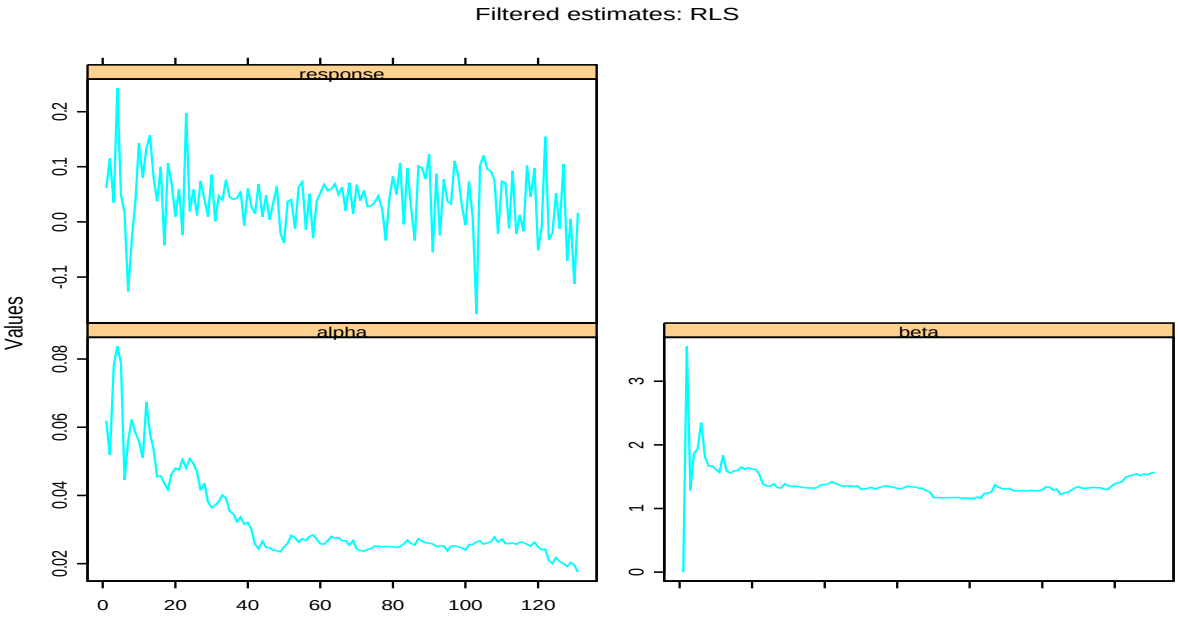

Figure 2: RLS estimates of CAPM for Microsoft using the Kalman filter.

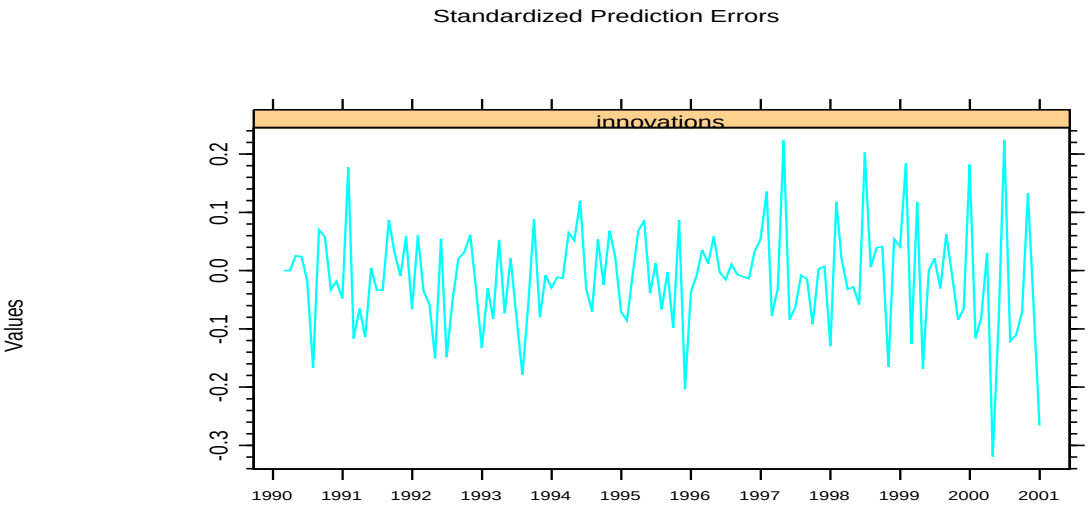

Figure 3: Standardized innovations, $w_t = v_t/\sqrt{f_t}$, computed from the RLS estimates of the CAPM for Microsoft.

```
> ssf.reg = GetSsfReg(X.mat)
> ssf.reg
$mPhi:            $mOmega:              $mSigma:         $mJPhi:
     [,1] [,2]         [,1] [,2] [,3]        [,1] [,2]        [,1] [,2]
[1,]    1    0    [1,]    0    0    0   [1,]   -1    0   [1,]   -1   -1
[2,]    0    1    [2,]    0    0    0   [2,]    0   -1   [2,]   -1   -1
[3,]    0    0    [3,]    0    0    1   [3,]    0    0   [3,]    1    2
> filteredEst.reg = SsfMomentEst(msft.ret,ssf.reg,task="STFIL")
> class(filteredEst.reg)
[1] "SsfMomentEst"
> names(filteredEst.reg)
[1] "state.moment"      "state.variance"    "response.moment"
[4] "response.variance" "task"              "positions"
> filteredEst.reg$state.moment
numeric matrix: 131 rows, 2 columns.
       state.1   state.2
  [1,] 0.06186 0.0001756
  [2,] 0.05179 3.5482887
  [3,] 0.07811 1.2844189
...
[131,] 0.01751 1.568
> ols.fit = OLS(MSFT~SP500,data=excessReturns.ts)
> coef(ols.fit)
 (Intercept) SP500
     0.01751 1.568
> colIds(filteredEst.reg$state.moment) = c("alpha","beta")
> plot(filteredEst.reg,main="Filtered estimates: RLS")
```

Listing 4: A recursive least squares analysis of the CAPM

```
> kf.reg = KalmanFil(msft.ret,ssf.reg)
> class(kf.reg)
[1] "KalmanFil"
> names(kf.reg)
 [1] "mOut"         "innov"        "std.innov"    "mGain"
 [5] "loglike"      "loglike.conc" "dVar"         "mEst"
 [9] "mOffP"        "task"         "err"          "call"
[13] "positions"

> plot(kf.reg)
```

Listing 5: Kalman filter for CAPM

plots of the standardized innovations may be created using the `plot` method for objects of class "`KalmanFil`". Selection 3 produces the graph shown in Figure 3.

The CUSUM test is based on the cumulated sum of the standardized recursive residuals

$$CUSUM_t = \sum_{j=k+1}^{t} \frac{\hat{w}_j}{\hat{\sigma}_w}$$

where $\hat{\sigma}_w$ is the sample standard deviation of $\hat{w}_j$ and k denotes the number of estimated coefficients. Under the null hypothesis that $\boldsymbol{\beta}$ in (19) is constant, $CUSUM_t$ has mean zero and vari-

ance that is proportional to $t-k-1$. Brown et al. (1975) show that approximate 95% confidence bands for $CUSUM_t$ are given by the two lines which connect the points $(k, \pm 0.948\sqrt{n-k-1})$ and $(n, \pm 0.948 \cdot 3\sqrt{n-k-1})$. If $CUSUM_t$ wanders outside of these bands, then the null of parameter stability may be rejected. The S-PLUS commands to compute $CUSUM_t$ are in listing 6. The CUSUM plot is given in Figure 4 and it indicates that the CAPM for Microsoft has stable parameters.

```
> w.t = kf.reg$std.innov[-c(1,2)]     # first two innovations are equal to zero
> cusum.t = cumsum(w.t)/stdev(w.t)
> nobs = length(cusum.t)
> tmp = 0.948*sqrt(nobs)
> upper = seq(tmp,3*tmp,length=nobs)
> lower = seq(-tmp,-3*tmp,length=nobs)
> tmp.ts = timeSeries(pos=kf.reg$positions[-c(1,2)],
+ data=cbind(cusum.t,upper,lower))
> plot(tmp.ts,reference.grid=F,
+ plot.args=list(lty=c(1,2,2),col=c(1,2,2)))
```

Listing 6: Computing the CUSUM test

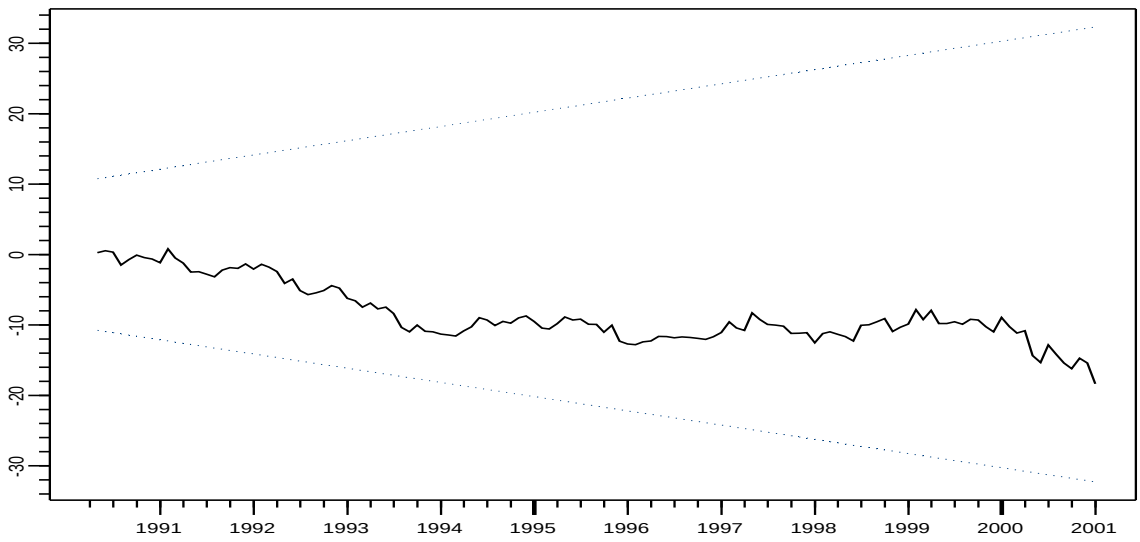

Figure 4: CUSUM test for parameter constancy in CAPM regression for Microsoft.

4.3 CAPM with Time Varying Parameters

Consider estimating the CAPM with time varying coefficients (14) subject to random walk evolution, using monthly data on Microsoft and the S&P 500 index over the period February, 1990 through December, 2000 contained in the S+FinMetrics "timeSeries" object excessReturns.ts.

Neumann (2002) surveys several estimation strategies for time varying parameter models and concludes that the state space model with random walk specifications for the evolution of the time varying parameters generally performs very well. The parameters of the model are the variances of the innovations to the transition and measurement equations: $\boldsymbol{\sigma}^2 = (\sigma_\xi^2, \sigma_\varsigma^2, \sigma_\nu^2)'$. Since these variances must be positive the log-likelihood is parameterized using $\boldsymbol{\varphi} = (\ln(\sigma_\xi^2), \ln(\sigma_\varsigma^2), \ln(\sigma_\nu^2))'$, so that $\boldsymbol{\sigma}^2 = (\exp(\varphi_1), \exp(\varphi_2), \exp(\varphi_3))'$. The state space form for the CAPM with time varying coefficients requires a data matrix $\mathbf{X}$ containing the excess returns on the S&P 500 index and therefore the function `SsfFit` has $\boldsymbol{\varphi}$ and $\mathbf{X}$ as input and it returns the appropriate state space form. Listing 7 provides an example of an implementation for estimating the time varying CAPM.

Starting values for $\boldsymbol{\varphi}$ are specified by `tvp.start`. The maximum likelihood estimates for $\boldsymbol{\varphi}$ are computed using `tvp.mle` The `print` method gives estimates of $\boldsymbol{\varphi} = (\ln(\sigma_\xi^2), \ln(\sigma_\varsigma^2), \ln(\sigma_\nu^2))'$ and the `summary` method The estimates for the standard deviations σ_ξ, σ_ς and σ_ν as well as estimated standard errors, from the delta method[6], are: It appears that the estimated standard deviations for the time varying parameter CAPM are close to zero, suggesting a constant parameter model.

Given the estimated parameters, the filtered estimates of the time varying parameters α_t and $\beta_{M,t}$ may be computed using `SsfMomentEst`. The filtered moments, without standard error bands, may be visualized using the `plot` method for objects of class "`SsfMomentEst`" as illustrated in Figure 5. The filtered estimates of the parameters from the CAPM with time varying parameters look remarkably like the RLS estimates computed earlier.

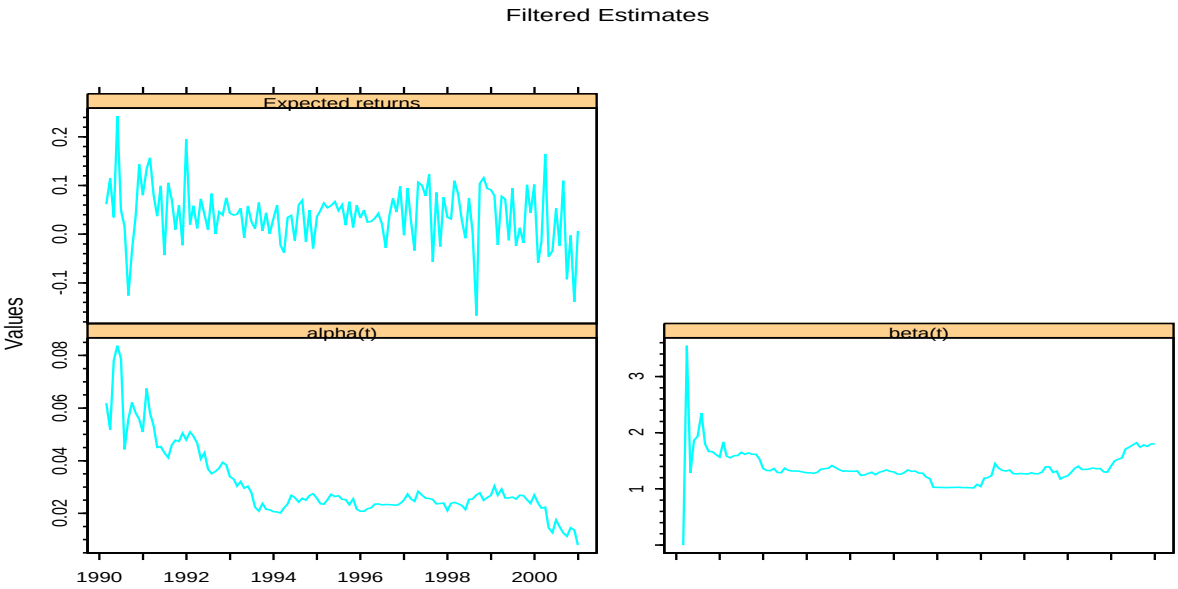

Figure 5: Filtered estimates of CAPM for Microsoft with time varying parameters.

The smoothed estimates of the time varying parameters α_t and $\beta_{M,t}$ as well as the expected returns may be extracted using `SsfCondDens` as in listing 7. The `plot` method is then used to visualize the smoothed estimates as illustrated in Figure 6. The smoothed state estimates appear

[6] If $\sqrt{n}(\hat{\theta} - \theta) \xrightarrow{d} N(\theta, V)$ and g is a continuous function then $\sqrt{n}(g(\hat{\theta}) - g(\theta)) \xrightarrow{d} N(0, \frac{\partial g}{\partial \theta'} V \frac{\partial g}{\partial \theta'}')$.

```
> tvp.mod = function(parm,mX=NULL) {
   parm = exp(parm)   # 3 x 1 vector containing log variances
   ssf.tvp = GetSsfReg(mX=mX)
   diag(ssf.tvp$mOmega) = parm
   CheckSsf(ssf.tvp)
}

> tvp.start = c(0,0,0)
> names(tvp.start) = c("ln(s2.alpha)","ln(s2.beta)","ln(s2.y)")

> tvp.mle = SsfFit(tvp.start,msft.ret,"tvp.mod",mX=X.mat)
Iteration  0 : objective =  183.2
...
Iteration  22 : objective =  -123
RELATIVE FUNCTION CONVERGENCE
> class(tvp.mle)
[1] "SsfFit"
> names(tvp.mle)
 [1] "parameters" "objective"  "message"    "grad.norm"  "iterations"
 [6] "f.evals"    "g.evals"    "hessian"    "scale"      "aux"
[11] "call"       "vcov"

> summary(tvp.mle)
Log-likelihood:  -122.979
131 observations
Parameters:
               Value Std. Error t value
ln(s2.alpha) -12.480     2.8020  -4.453
 ln(s2.beta)  -5.900     3.0900  -1.909
    ln(s2.y)  -4.817     0.1285 -37.480

> tvp2.mle = tvp.mle
> tvp2.mle$parameters = exp(tvp.mle$parameters/2)
> names(tvp2.mle$parameters) = c("s.alpha","s.beta","s.y")
> dg = diag(tvp2.mle$parameters/2)
> tvp2.mle$vcov = dg %*% tvp.mle$vcov %*% dg
> summary(tvp2.mle)
Log-likelihood:  -122.979
131 observations
Parameters:
           Value Std. Error t value
s.alpha 0.001951   0.002733  0.7137
 s.beta 0.052350   0.080890  0.6472
    s.y 0.089970   0.005781 15.5600

> smoothedEst.tvp = SsfCondDens(msft.ret,
+ tvp.mod(tvp.mle$parameters,mX=X.mat),
+ task="STSMO")
```

Listing 7: An analysis of the time-varying CAPM

quite different from the filtered state estimates shown in Figure 5, but this difference is primarily due to the erratic behavior of the first few filtered estimates. The function `SsfCondDens` does

not compute estimated variances for the smoothed state and response variables. If standard error bands for the smoothed estimates are desired, then `SsfMomentEst` with `task="STSMO"` must be used and the state variances are available in the component `state.variance`.

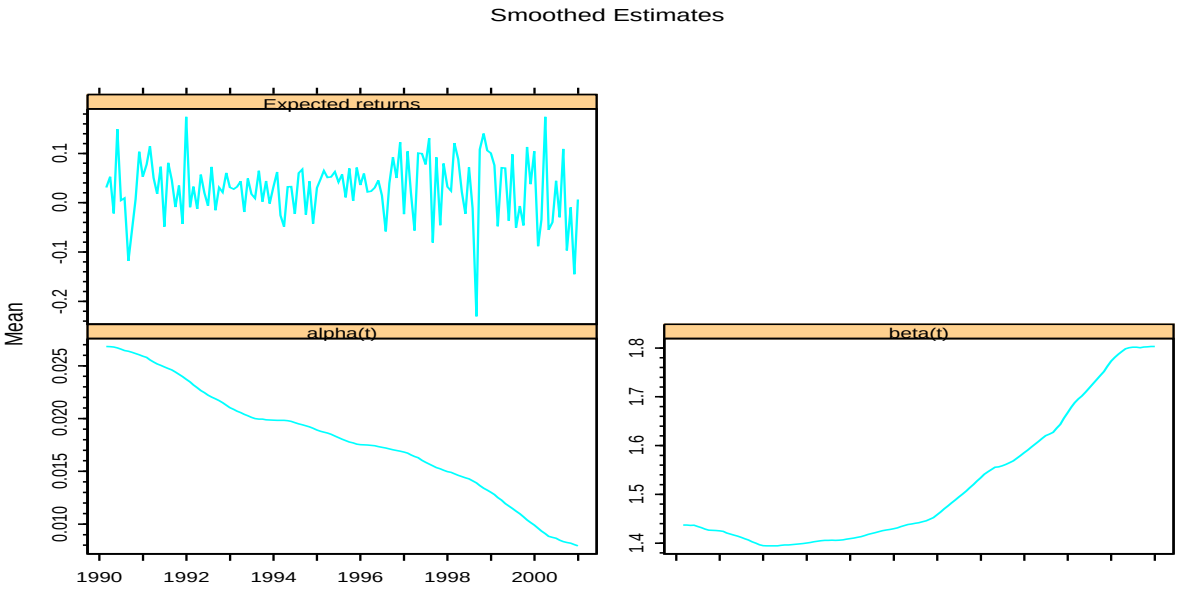

Figure 6: Filtered estimates of CAPM for Microsoft with time varying parameters.

5 Time Series Decompositions

In this section we use state space methods to compute some common trend-cycle decompositions of U.S. postwar quarterly real GDP. These decompositions are used to estimate the long-run trend in output as well as business cycles. We illustrate the well known Beveridge-Nelson decomposition as well as several unobserved components decompositions.

5.1 ARMA Modeling and Beveridge-Nelson Decompositions

Consider the problem of decomposing the movements in the natural logarithm of U.S. postwar quarterly real GDP into permanent and transitory (cyclical) components. Beveridge & Nelson (1981) proposed a definition for the permanent component of an $I(1)$ time series y_t with drift μ as the limiting forecast as horizon goes to infinity, adjusted for the mean rate of growth:

$$BN_t = \lim_{h \to \infty} E_t[y_{t+h} - \mu h]$$

where $E_t[\cdot]$ denotes expectation conditional on information available at time t. The transitory or cycle component is then defined as the gap between the present level of the series and its long-run forecast:

$$c_t = y_t - BN_t$$

This permanent-transitory decomposition is often referred to as the "BN decomposition". In practice, the BN decomposition is obtained by fitting an ARMA(p,q) model to Δy_t, where $\Delta = 1 - L$ is the difference operator, and then computing BN_t and c_t from the fitted model.

As shown recently by Morley (2002), the BN decomposition may be easily computed using the Kalman filter by putting the forecasting model for $\Delta y_t - \mu$ in state space form. In particular, suppose $\Delta y_t - \mu$ is a linear combination of the elements of the $m \times 1$ state vector $\boldsymbol{\alpha}_t$:

$$\Delta y_t - \mu = \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \boldsymbol{\alpha}_t$$

where z_i $(i = 1, \ldots, m)$ is the weight of the ith element of α_t in determining $\Delta y_t - \mu$. Suppose further that

$$\boldsymbol{\alpha}_{t+1} = \mathbf{T}\boldsymbol{\alpha}_t + \boldsymbol{\eta}_t^*, \qquad \boldsymbol{\eta}_t^* \sim \text{iid } N(0, \mathbf{V}),$$

such that all of the eigenvalues of $\mathbf{T}$ have modulus less than unity, and $\mathbf{T}$ is invertible. Then, Morley shows that

$$\begin{aligned} BN_t &= y_t + \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \mathbf{T}(\mathbf{I}_m - \mathbf{T})^{-1}\mathbf{a}_{t|t} \\ c_t &= y_t - BN_t = -\begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \mathbf{T}(\mathbf{I}_m - \mathbf{T})^{-1}\mathbf{a}_{t|t} \end{aligned} \tag{20}$$

where $\mathbf{a}_{t|t}$ denotes the filtered estimate of $\boldsymbol{\alpha}_t$.

To illustrate the process of constructing the BN decomposition for U.S. postwar quarterly real GDP over the period 1947:I to 1998:II, we follow Morley, Nelson & Zivot (2002) (hereafter MNZ) and consider fitting the ARMA(2,2) model

$$\begin{aligned} \Delta y_t - \mu &= \phi_1(\Delta y_{t-1} - \mu) + \phi_2(\Delta y_{t-2} - \mu) + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} \\ \varepsilon_t &\sim \text{iid } N(0, \sigma^2) \end{aligned}$$

where y_t denotes the natural log of real GDP multiplied by 100. In `S+FinMetrics/SsfPack`, the ARMA(p,q) model for a demeaned stationary variable y_t^* has a state space representation with transition and measurement equations

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &= \mathbf{T}\boldsymbol{\alpha}_t + \mathbf{H}\xi_t, \ \xi_t \sim N(0, \sigma_\varepsilon^2) \\ y_t^* &= \mathbf{Z}\boldsymbol{\alpha}_t \end{aligned}$$

and time invariant system matrices

$$\begin{aligned} \mathbf{T} &= \begin{pmatrix} \phi_1 & 1 & 0 & \cdots & 0 \\ \phi_2 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ \phi_{m-1} & 0 & 0 & & 1 \\ \phi_m & 0 & 0 & \cdots & 0 \end{pmatrix}, \ \mathbf{H} = \begin{pmatrix} 1 \\ \theta_1 \\ \vdots \\ \theta_{m-1} \\ \theta_m \end{pmatrix}, \\ \mathbf{Z} &= \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \end{pmatrix} \end{aligned} \tag{21}$$

where $\mathbf{d}, \mathbf{c}$ and $\mathbf{G}$ of the state space form (1)-(3) are all zero and $m = \max(p, q+1)$. The state vector $\boldsymbol{\alpha}_t$ has the form

$$\boldsymbol{\alpha}_t = \begin{pmatrix} y_t^* \\ \phi_2 y_{t-1}^* + \cdots + \phi_p y_{t-m+1}^* + \theta_1\xi_t + \cdots + \theta_{m-1}\xi_{t-m+2} \\ \phi_3 y_{t-1}^* + \cdots + \phi_p y_{t-m+2}^* + \theta_2\xi_t + \cdots + \theta_{m-1}\xi_{t-m+3} \\ \vdots \\ \phi_m y_{t-1}^* + \theta_{m-1}\xi_t \end{pmatrix} \tag{22}$$

The exact maximum likelihood estimates of the ARMA(2,2) parameters may be computed using the S+FinMetrics/SsfPack functions GetSsfArma and SsfFit. The function SsfFit requires as input a function which takes the unknown parameters $\boldsymbol{\varphi}$ and produces the state space form for the ARMA(2,2) as is illustrated in listing 8. Notice that the function arma22.mod parameterizes the error variance as $\sigma^2 = \exp(\gamma)$, $-\infty < \gamma < \infty$, to ensure that the estimated value of σ^2 is positive, and utilizes the S+FinMetrics/SsfPack function GetSsfArma to create the state space form for the ARMA(2,2) function. Starting values for the estimation are given by (conditional) MLE using S-PLUS function arima.mle. The data used for the estimation is in the "timeSeries" lny.ts and the demeaned first difference data is in the "timeSeries" dlny.ts.dm. The exact maximum likelihood estimates for $\boldsymbol{\varphi} = (\phi_1, \phi_2, \theta_1, \theta_2, \gamma)'$ are computed using SsfFit[7].

```
> arma22.mod = function(parm) {
  phi.1 = parm[1]
  phi.2 = parm[2]
  theta.1 = parm[3]
  theta.2 = parm[4]
  sigma2 = exp(parm[5])  # require positive variance
  ssf.mod = GetSsfArma(ar=c(phi.1,phi.2),ma=c(theta.1,theta.2),
  sigma=sqrt(sigma2))
  CheckSsf(ssf.mod)
}

> arma22.start = c(1.34,-0.70,-1.05,0.51,-0.08)
> names(arma22.start) = c("phi.1","phi.2","theta.1","theta.2","ln.sigma2")

> arma22.mle = SsfFit(arma22.start,dlny.ts.dm,"arma22.mod")
Iteration  0 : objective =  284.6686
...
Iteration  27 : objective =  284.651
RELATIVE FUNCTION CONVERGENCE

> summary(arma22.mle)
Log-likelihood:  284.651
205 observations
Parameters:
             Value Std. Error t value
    phi.1  1.34200    0.14480  9.2680
    phi.2 -0.70580    0.14930 -4.7290
  theta.1 -1.05400    0.18030 -5.8490
  theta.2  0.51870    0.19330  2.6830
ln.sigma2 -0.06217    0.09878 -0.6294
```

Listing 8: Estimation of an ARMA model

Given the maximum likelihood estimates $\boldsymbol{\varphi}$, the filtered estimate of the state vector may be computed using the function SsfMomentEst with optional argument task="STFIL". The BN decomposition (20) may then be computed as in listing 9. Figure 7 illustrates the results of

[7] One may also estimate the ARMA(2,2) model with SsfFit by maximizing the log-likelihood function concentrated with respect to the error variance σ^2.

the BN decomposition for U.S. real GDP. The BN trend follows the data very closely and, as a result, the BN cycle behaves much like the first difference of the data.

```
> ssf.arma22 = arma22.mod(arma22.mle$parameters)
> filteredEst.arma22 = SsfMomentEst(dlny.ts.dm,
+ ssf.arma22,task="STFIL")
> at.t = filteredEst.arma22$state.moment

> T.mat = ssf.arma22$mPhi[1:3,1:3]
> tmp = t(T.mat %*% solve((diag(3)-T.mat)) %*% t(at.t))
> BN.t = lny[2:nobs,] + tmp[,1]
> c.t = lny[2:nobs,] - BN.t
```

Listing 9: Implementation of Beveridge Nelson Decomposing using Morley (2002)

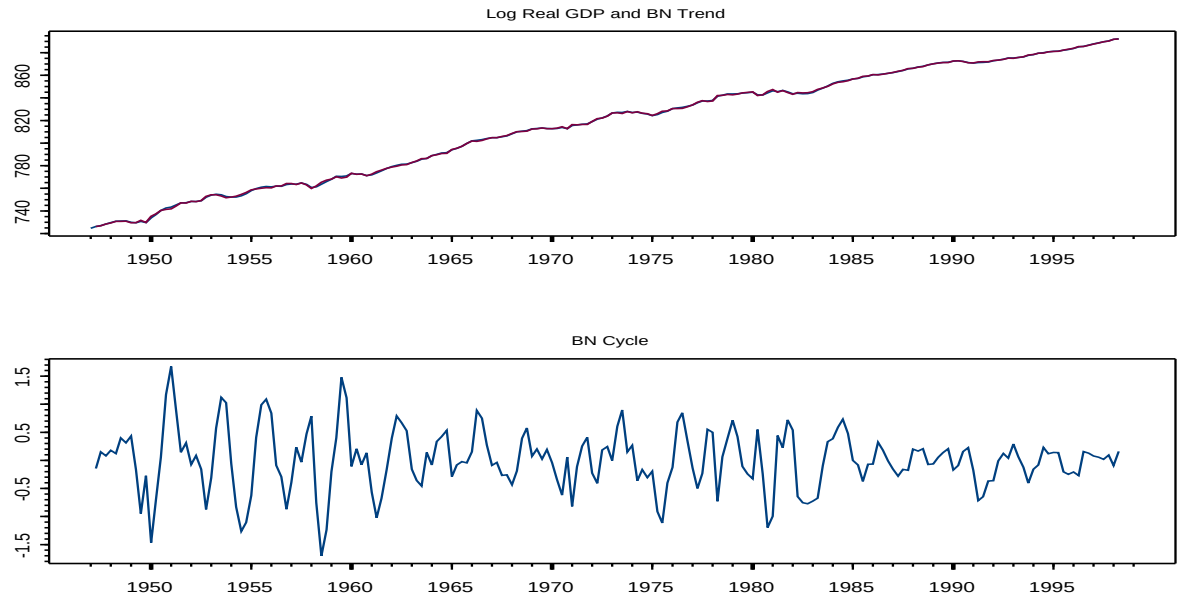

Figure 7: BN decomposition for U.S. postwar quarterly real GDP.

5.2 Unobserved Components Decompositions: Clark Model

Harvey (1985) and Clark (1987) provide an alternative to the BN decomposition of an $I(1)$ time series with drift into permanent and transitory components based on unobserved components structural time series models. For example, Clark's model for the natural logarithm of postwar real GDP specifies the trend as a pure random walk, and the cycle as a stationary AR(2) process:

$$\begin{aligned} y_t &= \tau_t + c_t \\ \tau_t &= \mu + \tau_{t-1} + v_t \\ c_t &= \phi_1 c_{t-1} + \phi_2 c_{t-2} + w_t \end{aligned} \tag{23}$$

where the roots of $\phi(z) = 1 - \phi_1 z - \phi_2 z^2 = 0$ lie outside the complex unit circle[8]. For identification purposes, Clark assumes that the trend innovations and cycle innovations are uncorrelated and normally distributed:

$$\begin{pmatrix} v_t \\ w_t \end{pmatrix} \sim \text{ iid } N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_w^2 \end{pmatrix}\right)$$

The Clark model may be put in state-space form (5) with

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &= \begin{pmatrix} \tau_{t+1} \\ c_{t+1} \\ c_t \end{pmatrix}, \boldsymbol{\delta} = \begin{pmatrix} \mu \\ 0 \\ 0 \\ 0 \end{pmatrix}, \Phi = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \phi_1 & \phi_2 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \\ \mathbf{u}_t &= \begin{pmatrix} \eta_t^* \\ 0 \end{pmatrix}, \boldsymbol{\eta}_t^* = \begin{pmatrix} v_{t+1} \\ w_{t+1} \\ 0 \end{pmatrix}, \Omega = \begin{pmatrix} \sigma_v^2 & 0 & 0 & 0 \\ 0 & \sigma_w^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

Since the trend component is nonstationary, it is given a diffuse initialization. The initial covariance matrix $\mathbf{P}_*$ of the stationary cycle is determined from

$$\text{vec}(\mathbf{P}_*) = (\mathbf{I}_4 - (\mathbf{F} \otimes \mathbf{F})^{-1})\text{vec}(\mathbf{V}_w)$$

where

$$\mathbf{F} = \begin{pmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{pmatrix}, \ \mathbf{V}_w = \begin{pmatrix} \sigma_w^2 & 0 \\ 0 & 0 \end{pmatrix}$$

The initial value parameter matrix (7) is then

$$\Sigma = \begin{pmatrix} -1 & 0 & 0 \\ 0 & p_{11} & p_{12} \\ 0 & p_{21} & p_{22} \\ 0 & 0 & 0 \end{pmatrix}$$

where p_{ij} denotes the (i, j) element of $\mathbf{P}_*$.

The exact maximum likelihood estimates of the Clark model parameters, based on the prediction error decomposition of the log-likelihood function, may be computed using the `S+FinMetrics/SsfPack` `SsfFit`. The function `SsfFit` requires as input a function which takes the unknown parameters $\boldsymbol{\varphi}$ and produces the state space form for the Clark model; an example is given in listing 10. Notice that the state variances are parameterized as $\sigma_v^2 = \exp(\gamma_v)$ and $\sigma_w^2 = \exp(\gamma_w)$, $-\infty < \gamma_v, \gamma_w < \infty$, to ensure positive estimates. Starting values for the parameters are based on values near the estimates of the Clark model from MNZ.

The data used for the estimation is in the "`timeSeries`" `lny.ts`, and is the same data used to compute the BN decomposition earlier. The maximum likelihood estimates and asymptotic

[8]Harvey's model differs from Clark's model in that the stationary AR(2) cycle is restricted to have complex roots. The function `GetSsfStsm` in Table 3 may be used to easily construct the state space form for Harvey's model.

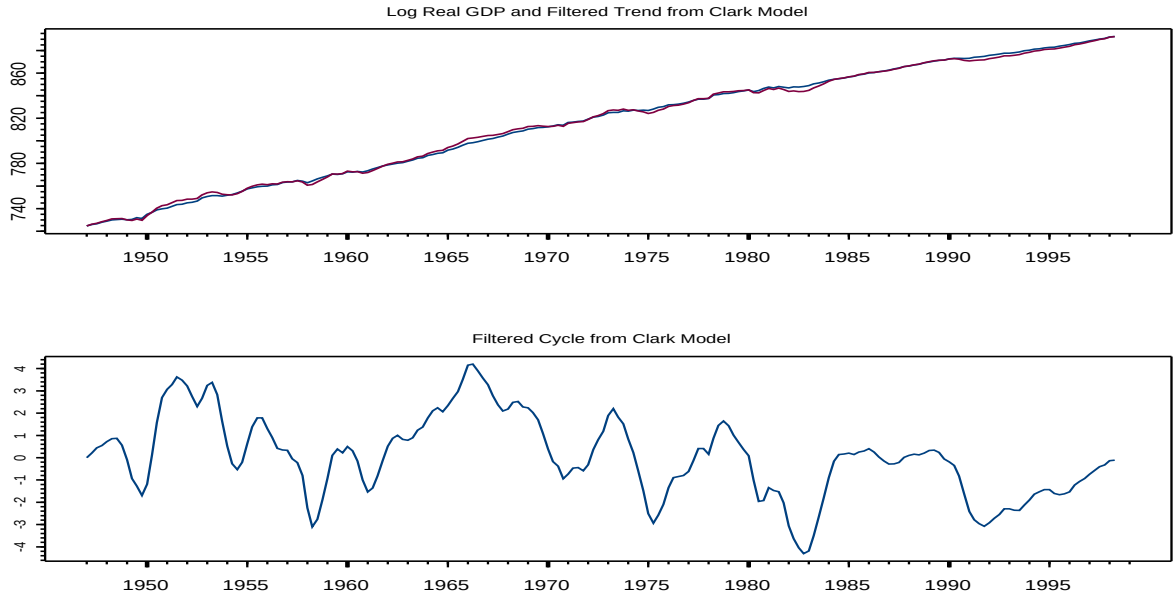

Figure 8: Filtered estimates of the trend and cycle from the Clark model estimated to U.S. real GDP.

standard errors of the parameters $\boldsymbol{\varphi} = (\mu, \gamma_v, \gamma_w, \phi_1, \phi_2)'$ using `SsfFit` are[9] The maximum likelihood estimates for the Clark model parameters are almost identical to those found by MNZ[10]. Estimates of the error component standard deviations are also produced.

The filtered estimates of the trend, $\tau_{t|t}$, and cycle, $c_{t|t}$, given the estimated parameters may be computed using the function `SsfMomentEst` with the optional argument `task="STFIL"` as in listing 10. The filtered estimates are in the `state.moment` component and the variances of the filtered estimates are in the `state.variance` component. The filtered trend estimate is in the first column of the `state.moment` component and the filtered cycle is in the second column. These filtered estimates are illustrated in Figure 8. The filtered trend estimate is fairly smooth and is quite similar to a linear trend. The filtered cycle estimate is large in amplitude and has a period of about eight years. In comparison to the BN decomposition, the trend-cycle decomposition based on the Clark model gives a much smoother trend and longer cycle, and attributes a greater amount of the variability of log output to the transitory cycle.

The smoothed estimates of the trend, $\tau_{t|n}$, and cycle, $c_{t|n}$, along with estimated standard errors, given the estimated parameters, may be computed using the function `SsfMomentEst` with the optional argument `task="STSMO"`. The smoothed cycle estimates with 95% standard error bands are illustrated in Figure 9.

The Clark model assumes that the unobserved trend evolves as a random walk with drift

[9]In the estimation, no restrictions were imposed on the AR(2) parameters ϕ_1 and ϕ_2 to ensure that the cycle is stationary. The function `SsfFit` uses the `S-PLUS` optimization algorithm `nlminb`, which performs minimization of a function subject to box constraints. Box constraints on ϕ_1 and ϕ_2 may be used to constrain their estimated values to be near the appropriate stationary region.

[10]MNZ estimate the Clark model in GAUSS using the prediction error decomposition with the variance of the initial state for the nonstationary component set to a large positive number. The state space representation of the Clark model here utilizes an exact initialization of the Kalman filter.

```
Clark.mod = function(parm) {
  mu = parm[1]
  phi1 = parm[2]
  phi2 = parm[3]
  sigma2.v = exp(parm[4])
  sigma2.w = exp(parm[5])
  bigV = diag(c(sigma2.v,sigma2.w))
  Omega = matrix(0,4,4)
  Omega[1:2,1:2] = bigV
  a1 = matrix(0,3,1)
# solve for initial variance of stationary part
  bigF = matrix(c(phi1,1,phi2,0),2,2)
  vecV = c(sigma2.w,0,0,0)
  vecP = solve(diag(4)-kronecker(bigF,bigF))%*%vecV
  P.ar2 = matrix(vecP,2,2)
  Sigma = matrix(0,4,3)
  Sigma[1,1] = -1
  Sigma[2:3,2:3] = P.ar2
# create state space list
  ssf.mod = list(mDelta=c(mu,0,0,0),
  mPhi=rbind(c(1,0,0),c(0,phi1,phi2),c(0,1,0),c(1,1,0)),
  mOmega=Omega,
  mSigma = Sigma)
  CheckSsf(ssf.mod)
}

> Clark.start=c(0.81,1.53,-0.61,-0.74,-0.96)
> names(Clark.start) = c("mu","phi.1","phi.2",
+ "ln.sigma2.v","ln.sigma2.w")

> Clark.mle = SsfFit(Clark.start,lny.ts,"Clark.mod")
> summary(Clark.mle)
Log-likelihood:  287.524
206 observations
Parameters:
              Value Std. Error t value
         mu  0.8119    0.05005  16.220
      phi.1  1.5300    0.10180  15.030
      phi.2 -0.6097    0.11450  -5.326
ln.sigma2.v -0.7441    0.30100  -2.472
ln.sigma2.w -0.9565    0.42490  -2.251

> Clark.sd = sqrt(exp(coef(Clark.mle)[4:5]))
> names(Clark.sd) = c("sigma.v","sigma.w")
> Clark.sd
 sigma.v sigma.w
  0.6893  0.6199

> ssf.Clark = Clark.mod(Clark.mle$parameters)
> filteredEst.Clark = SsfMomentEst(lny.ts,ssf.Clark,task="STFIL")
```

Listing 10: Estimating Clark's model

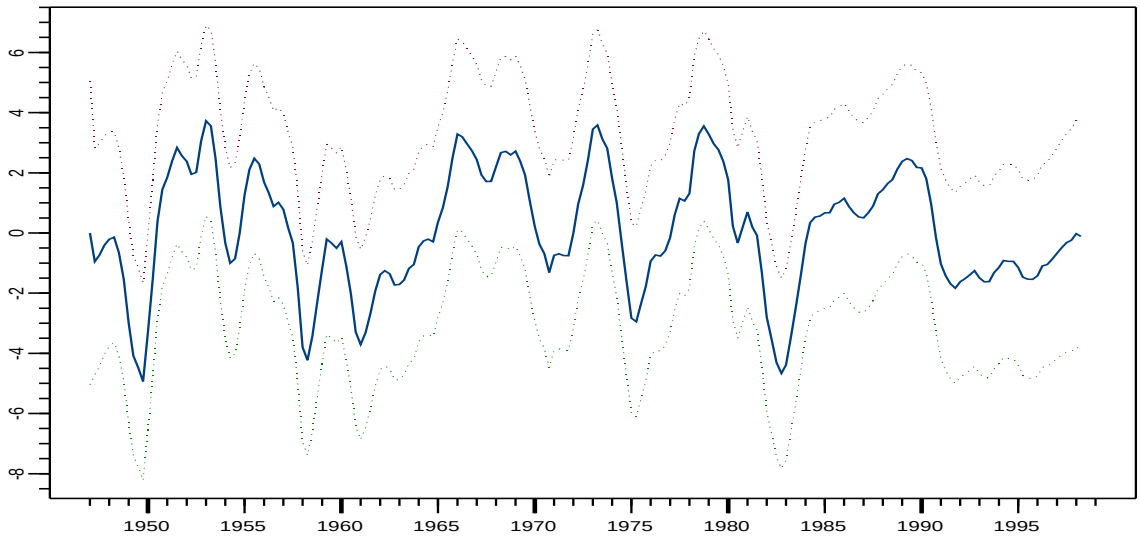

Figure 9: Smoothed cycle estimate, $c_{t|n}$, with 95% error bands from Clark model for U.S. real GDP.

. That is, the unobserved trend component is nonstationary. If the variance of the drift innovation, σ_v^2, is zero then the trend becomes deterministic. A number of statistics have been proposed to test the null hypothesis that $\sigma_v^2 = 0$, see Harvey & Streibel (1997) and Harvey (2001) for reviews. For the Clark model, a Lagrange multiplier (LM) test of the null hypothesis $\sigma_v^2 = 0$, against the alternative that $\sigma_v^2 > 0$, can be formulated as

$$\eta = T^{-1} \sum_{i=1}^{T} \left[\sum_{t=1}^{i} e_t \right]^2 > c$$

where e_t is the standardized innovation at time t from the model assuming that τ_0 is fixed, and c is the desired critical value. Harvey & Streibel (1997) show that the statistic η has a second-level Cramér-von Mises distribution. Table I(b) of Harvey (2001) gives $c = 0.149$ as the 5% critical value.

To compute the standardized innovations assuming that τ_0 is fixed, Harvey & Streibel (1997) suggest the following procedures. Start by estimating the unrestricted model by maximum likelihood and compute the smoothed estimates of τ_t. Then the standardized innovations e_t, assuming that τ_0 is fixed, are computed from the Kalman filter algorithm by setting $\sigma_v^2 = 0$ and initializing τ_0 at the smoothed estimate of $\tau_1 - \mu$. The commands to compute the LM test statistic are listed in 11. Since the test statistic is greater than the 5% critical value of 0.149, the null hypothesis of a deterministic trend is rejected.

5.3 Unobserved Components Decompositions: MNZ Model

Recently, Morley et al. (2002) have shown that the apparent difference between BN decomposition and the Clark model trend-cycle decomposition is due to the assumption of independence

```
> n = nrow(lny.ts)
> ssf.Clark0 = ssf.Clark
> ssf.Clark0$mOmega[1,1] = 0
> ssf.Clark0$mSigma[1,1] = 0
> ssf.Clark0$mSigma[4,1] = smoothedEst.Clark$state.moment[1,1] -
> Clark.mle$parameters["mu"]
> kf.Clark0 = KalmanFil(lny.ts,ssf.Clark0)
> test.stat = sum(cumsum(kf.Clark0$std.innov)^2)/n^2
> test.stat
[1] 56.71
```

Listing 11: Computing the LM test for the null of $\sigma_v^2 = 0$.

between trend and cycle innovations in the Clark model. In particular, they show that the independence assumption is actually an overidentifying restriction in the Clark model, and once this assumption is relaxed to allow correlated components the difference between the decompositions disappears.

The MNZ model is simply Clark's model (23) where the trend and cycle innovations are allowed to be correlated with correlation coefficient ρ_{vw}:

$$\begin{pmatrix} v_t \\ w_t \end{pmatrix} \sim \text{ iid } N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_v^2 & \rho_{vw}\sigma_v\sigma_w \\ \rho_{vw}\sigma_v\sigma_w & \sigma_w^2 \end{pmatrix}\right)$$

The new state space system matrix Ω becomes

$$\Omega = \begin{pmatrix} \sigma_v^2 & \rho_{vw}\sigma_v\sigma_w & 0 & 0 \\ \rho_{vw}\sigma_v\sigma_w & \sigma_w^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

An `S-PLUS` function, to be passed to `SsfFit`, to compute the new state space form is given in listing 12. No restrictions are placed on the correlation coefficient ρ_{vw} in the function `MNZ.mod`. A box constraint $-0.999 < \rho_{vw} < 0.999$ will be placed on ρ_{vw} during the estimation. Starting values for the parameters are based on values near the estimates of the Clark model from MNZ. Box constraints on the AR parameters ϕ_1 and ϕ_2, to encourage stationarity, and the correlation coefficient ρ_{vw}, to enforce validity. The ML estimates are almost identical to those reported by MNZ. Notice that the estimated value of ρ_{vw} is -0.91 and that the estimated standard deviation of the trend innovation is much larger than the estimated standard deviation of the cycle innovation.

The filtered estimates of the trend, $\tau_{t|t}$, and cycle, $c_{t|t}$, given the estimated parameters are computed using the `SsfMomentEst` function and are illustrated in Figure 10. Notice that the filtered estimates of trend and cycle , when the correlation between the error components is estimated, are identical to the estimated trend and cycle from the BN decomposition. The smoothed estimates of trend and cycle are much more variable than the filtered estimates. Figure 11 shows the smoothed estimate of the cycle. For more discussion see Proietti (2002).

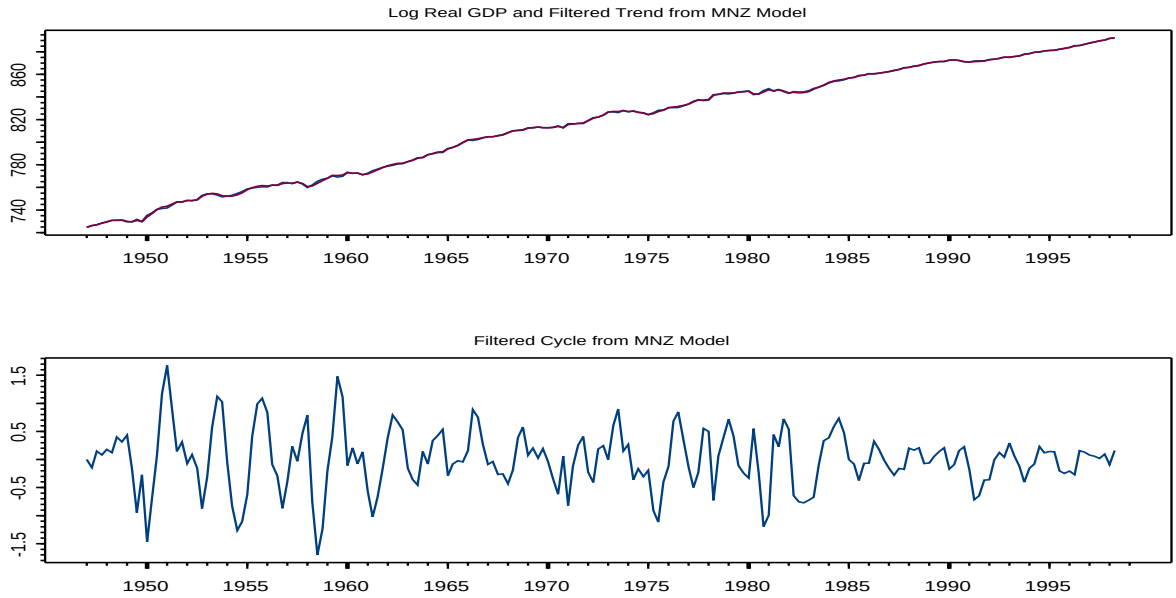

Figure 10: Filtered estimates from Clark model with correlated components for U.S. real GDP.

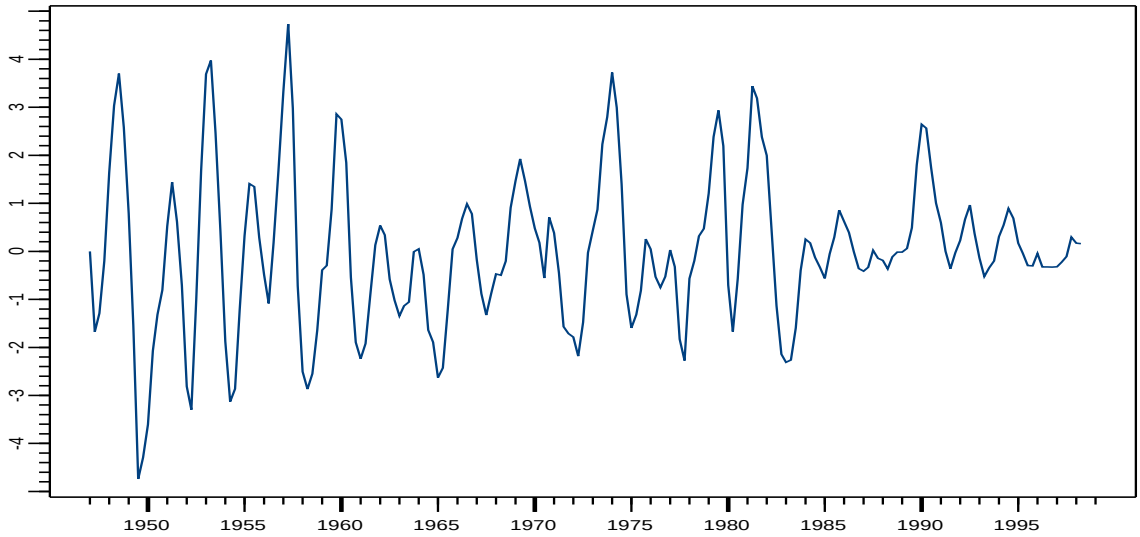

Figure 11: Smooth cycle estimate, $c_{t|n}$, from MNZ model for U.S. real GDP.

```
MNZ.mod = function(parm) {
  delta = parm[1]
  phi1 = parm[2]
  phi2 = parm[3]
  sigma.v = exp(parm[4])
  sigma.w = exp(parm[5])
  rho.vw = parm[6]
  sigma.vw = sigma.v*sigma.w*rho.vw
  bigV = matrix(c(sigma.v^2,sigma.vw,sigma.vw,sigma.w^2),2,2)
  Omega = matrix(0,4,4)
  Omega[1:2,1:2] = bigV
  a1 = matrix(0,3,1)
# solve for initial variance of stationary part
  bigF = matrix(c(phi1,1,phi2,0),2,2)
  vecV = c(sigma.w^2,0,0,0)
  vecP = solve(diag(4)-kronecker(bigF,bigF))%*%vecV
  P.ar2 = matrix(vecP,2,2)
  Sigma = matrix(0,4,3)
  Sigma[1,1] = -1
  Sigma[2:3,2:3] = P.ar2
  ssf.mod= list(mDelta=c(delta,0,0,0),
  mPhi=rbind(c(1,0,0),c(0,phi1,phi2),c(0,1,0),c(1,1,0)),
  mOmega=Omega,
  mSigma = Sigma)
  CheckSsf(ssf.mod)
}

> MNZ.start=c(0.81,1.34,-0.70,0.21,-0.30,-0.9)
> names(MNZ.start) = c("mu","phi.1","phi.2",
+ "ln.sigma.v","ln.sigma.w","rho")

> MNZ.mle = SsfFit(MNZ.start,lny.ts,"MNZ.mod",
+ lower=low.vals,upper=up.vals)
> summary(MNZ.mle)
Log-likelihood:  285.57
206 observations
Parameters:
             Value Std. Error t value
     delta  0.8156    0.08651  9.4280
     phi.1  1.3420    0.14550  9.2250
     phi.2 -0.7059    0.15050 -4.6890
ln.sigma.v  0.2125    0.13100  1.6230
ln.sigma.w -0.2895    0.38570 -0.7505
       rho -0.9062    0.12720 -7.1260

> MNZ.sd = exp(coef(MNZ.mle)[4:5])
> names(MNZ.sd) = c("sigma.v","sigma.w")
> MNZ.sd
 sigma.v sigma.w
   1.237  0.7487
```

Listing 12: Estimating MNZ model.

6 The Stochastic Volatility Model

Financial returns data can be characterized by noise processes with volatility clustering and non-Gaussian features. In the financial and econometrics literature much attention is devoted to the empirical modeling and analysing of volatility since volatility is important for the pricing of financial securities and their associated derivatives. Much applied work on this topic is focused on generalized autoregressive conditional heteroskedasticity (GARCH) models and its long list of variants. Although most of these models are relatively straightforward to estimate, they are not necessarily convincing in empirical analyses and do not necessarily produce satisfactory forecasts. In this section we focus on the stochastic volatility model that describes volatility as a stochastic process with its own independent source of randomness. Such descriptions are in nature expressed in continuous time but they can also be formulated in discrete time. The resulting model can then be regarded as the discrete time analogue of the continuous time model used in papers on option pricing, see Hull & White (1987). The discrete SV model is intrinsically a nonlinear model. The parameters can be estimated by using approximating methods or by using exact methods based on simulation which are subject to Monte Carlo error. Both estimation approaches will be illustrated in the next sections.

6.1 Quasi-Maximum Likelihood Estimation

Let r_t denote the continuously compounded return on an asset between times $t-1$ and t. Following Harvey, Ruiz & Shephard (1994), hereafter HRS, a simple stochastic volatility (SV) model has the form

$$\begin{aligned} r_t &= \sigma_t \varepsilon_t, \qquad \varepsilon_t \sim \text{iid } N(0,1) \\ h_t &= \ln \sigma_t^2 = \gamma + \phi h_{t-1} + \eta_t, \qquad \eta_t \sim \text{iid } N(0, \sigma_\eta^2) \\ E[\varepsilon_t \eta_t] &= 0 \end{aligned} \tag{24}$$

Defining $y_t = \ln r_t^2$, and noting that $E[\ln \varepsilon_t^2] = -1.27$ and $\text{var}(\ln \varepsilon_t^2) = \pi^2/2$ an unobserved components state space representation for y_t has the form

$$\begin{aligned} y_t &= -1.27 + h_t + \xi_t, \qquad \xi_t \sim \text{iid } (0, \pi^2/2) \\ h_t &= \gamma + \phi h_{t-1} + \eta_t, \qquad \eta_t \sim \text{iid } N(0, \sigma_\eta^2) \\ E[\xi_t \eta_t] &= 0 \end{aligned}$$

If ξ_t were iid Gaussian then the parameters $\boldsymbol{\varphi} = (\gamma, \sigma_\eta^2, \phi,)'$ of the SV model could be efficiently estimated by maximizing the prediction error decomposition of the log-likelihood function constructed from the Kalman filter recursions. However, since $\xi_t = \ln \varepsilon_t^2$ is not normally distributed the Kalman filter only provides minimum mean squared error linear estimators of the state and future observations. Nonetheless, HRS point out that even though the exact log-likelihood cannot be computed from the prediction error decomposition based on the Kalman filter, consistent estimates of $\boldsymbol{\varphi}$ can still be obtained by treating ξ_t as though it were iid $N(0, \pi^2/2)$ and maximizing the quasi log-likelihood function constructed from the prediction error decomposition.

The state space representation of the SV model has system matrices

$$\boldsymbol{\delta} = \begin{pmatrix} \gamma \\ -1.27 \end{pmatrix}, \qquad \Phi = \begin{pmatrix} \phi \\ 1 \end{pmatrix}, \qquad \Omega = \begin{pmatrix} \sigma_\eta^2 & 0 \\ 0 & \pi^2/2 \end{pmatrix}$$

Assuming that $|\phi| < 1$, the initial value matrix has the form

$$\Sigma = \begin{pmatrix} \sigma_\eta^2/(1-\phi^2) \\ \gamma/(1-\phi) \end{pmatrix}$$

If $\phi = 1$ then use

$$\Sigma = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

```
sv.mod = function(parm) {
     g = parm[1]
     sigma2.n = exp(parm[2])
     phi = exp(parm[3])/(1+exp(parm[3]))
     ssf.mod = list(mDelta=c(g,-1.27),
     mPhi=as.matrix(c(phi,1)),
     mOmega=matrix(c(sigma2.n,0,0,0.5*pi^2),2,2),
     mSigma=as.matrix(c((sigma2.n/(1-phi^2)),g/(1-phi))))
     CheckSsf(ssf.mod)
}

> parm.hrs = c(-0.3556,log(0.0312),log(0.9646/0.0354))
> nobs = 1000
> set.seed(179)
> e = rnorm(nobs)
> xi = log(e^2)+1.27
> eta = rnorm(nobs,sd=sqrt(0.0312))
> sv.sim = SsfSim(sv.mod(parm.hrs),
+ mRan=cbind(eta,xi),a1=(-0.3556/(1-0.9646)))

> sv.start = c(-0.3,log(0.03),0.9)
> names(sv.start) = c("g","ln.sigma2","exp(phi)/(1+exp(phi))")

> sv.mle = SsfFit(sv.start,sv.sim[,2],"sv.mod")
Iteration  0 : objective =  5147.579
...
Iteration  32 : objective =  2218.26
RELATIVE FUNCTION CONVERGENCE
> sv.mle
Log-likelihood:  2218
1000 observations
Parameter Estimates:
          g ln.sigma2 exp(phi)/(1+exp(phi))
 -0.4815574 -3.561574              2.963182
```

Listing 13: Estimating the SV model.

The function to obtain the state space form of the SV model given a vector of parameters, assuming $|\phi| < 1$, is given in listing 13. The logit transformation is used to impose the restriction $0 < \phi < 1$.

The analysis reported in listing 13 starts with simulating $T = 1000$ observations from the SV model using the parameters $\gamma = -0.3556$, $\sigma_\eta^2 = 0.0312$ and $\phi = 0.9646$. The simulated

squared returns, r_t^2, and latent squared volatility, σ_t^2, are shown in Figure 12. To estimate the underlying parameters of the simulated realisations, the starting values of $\boldsymbol{\varphi} = (\gamma, \sigma_\eta^2, \phi,)'$ are chosen close to the true values. The quasi-maximum likelihood (QML) estimates are obtained using `SsfFit`. The QML estimates of σ_η^2 and ϕ are 0.02839 and 0.95088, respectively[11]. These values are fairly close to the true values.

The filtered and smoothed estimates of log-volatility and volatility may computed using `SsfMomentEst` and `SsfCondDens`. One disadvantage of the QML approach is that the variances for log-volatility computed from the Kalman filter and smoother are not valid.

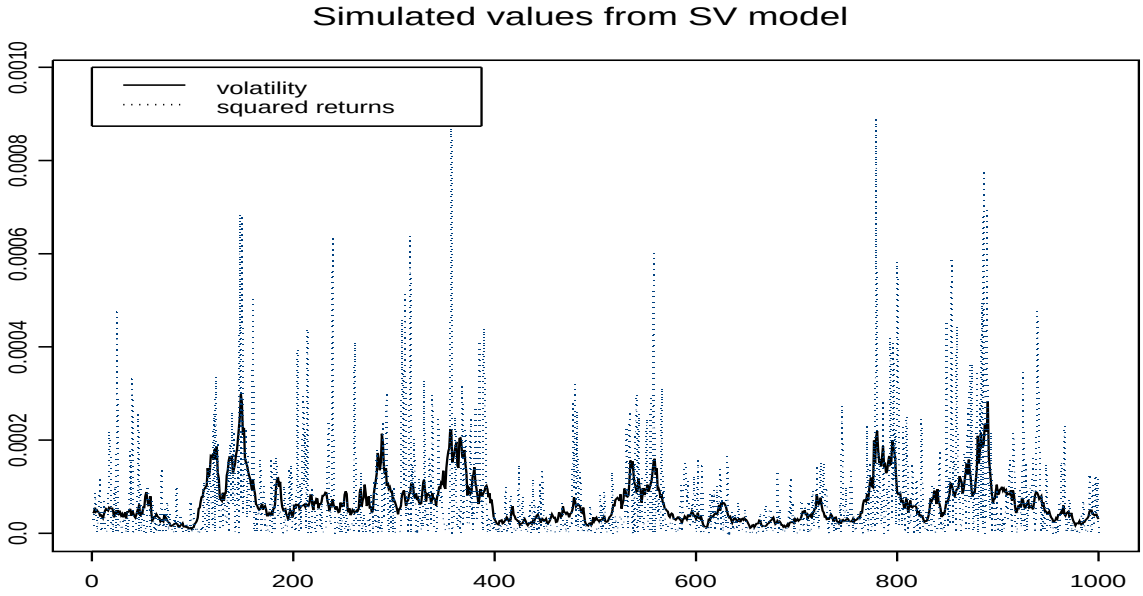

Figure 12: Simulated data from SV model.

6.2 Simulated Maximum Likelihood Estimation

We will consider here the estimation of the parameters of the SV model by exact maximum likelihood methods using Monte Carlo importance sampling techniques. For this purpose, we will use the following reparameterization of the SV model

$$\begin{aligned} r_t &= \sigma \exp\left(\frac{1}{2}\theta_t\right)\varepsilon_t,\ \varepsilon_t \sim N(0,1) \\ \theta_t &= \phi\theta_{t-1} + \eta_t,\ \eta_t \sim N(0,\sigma_\eta^2) \end{aligned} \tag{25}$$

where σ represents average volatility. The likelihood function of this SV model can be constructed using simulation methods developed by Shephard & Pitt (1997) and Durbin & Koopman (1997). The nonlinear relation between log-volatility θ_t and the observation equation of

[11]Currently, `SsfFit` does not compute the "sandwhich" covariance matrix estimator required for the quasi-MLE.

r_t does not allow the computation of the exact likelihood function by linear methods such as the Kalman filter. However, for the SV model (25) we can express the likelihood function as

$$L(\boldsymbol{\psi}) = p(\mathbf{y}|\boldsymbol{\psi}) = \int p(\mathbf{y}, \boldsymbol{\theta}|\boldsymbol{\psi}) d\boldsymbol{\theta} = \int p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta}|\boldsymbol{\psi}) d\boldsymbol{\theta}, \tag{26}$$

where

$$\mathbf{y} = (r_1, \ldots, r_n)', \qquad \boldsymbol{\psi} = (\sigma, \phi, \sigma_\eta^2)', \qquad \boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)'.$$

An efficient way of evaluating such expressions is by using importance sampling; see Ripley (1987, Chapter 5). A simulation device is required to sample from an importance density $\tilde{p}(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi})$ which we prefer to be as close as possible to the true density $p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi})$. An obvious choice for the importance density is the conditional Gaussian density since in this case it is relatively straightforward to sample from $\tilde{p}(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi}) = g(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi})$ using simulation smoothers such as the ones developed by de Jong & Shephard (1995) and Durbin & Koopman (2002). For constructing of the likelihood function using this approach, the following three steps are important.

1. The likelihood function (26) is obtained by writing

$$L(\boldsymbol{\psi}) = \int p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi}) \frac{p(\boldsymbol{\theta}|\boldsymbol{\psi})}{g(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi})} g(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi}) d\boldsymbol{\theta} = \tilde{E}\{p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi}) \frac{p(\boldsymbol{\theta}|\boldsymbol{\psi})}{g(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi})}\}, \tag{27}$$

   where $\tilde{E}$ denotes expectation with respect to the importance density $g(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi})$. Expression (27) can be simplified using a suggestion of Durbin & Koopman (1997). This leads to

$$L(\boldsymbol{\psi}) = L_g(\boldsymbol{\psi}) \tilde{E}\{\frac{p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi})}{g(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi})}\}, \tag{28}$$

   and is a convenient expression that we will use in the calculations. The likelihood function of the approximating Gaussian model $L_g(\boldsymbol{\psi})$ can be calculated via the Kalman filter. The conditional density functions $p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi})$ and $g(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi})$ are obviously easy to compute for given values of $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$. It follows that the likelihood function of the SV model is equivalent to the likelihood function of an approximating Gaussian model, multiplied by a correction term. This correction term only needs to be evaluated via simulation.

2. An obvious estimator for the likelihood of the SV model is

$$\hat{L}(\boldsymbol{\psi}) = L_g(\boldsymbol{\psi}) \bar{w}, \tag{29}$$

   where

$$\bar{w} = \frac{1}{M} \sum_{i=1}^{M} w_i, \qquad w_i = \frac{p(\mathbf{y}|\boldsymbol{\theta}^i, \boldsymbol{\psi})}{g(\mathbf{y}|\boldsymbol{\theta}^i, \boldsymbol{\psi})}, \tag{30}$$

   and $\boldsymbol{\theta}^i$ denotes a draw from the importance density $g(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi})$. The accuracy of this estimator depends on M, the number of simulation samples. In practice, we usually work with the log of the likelihood function to manage the magnitude of density values. The log

transformation of $\hat{L}(\boldsymbol{\psi})$ introduces bias for which we can correct up to order $O(M^{-3/2})$. We obtain

$$\ln \hat{L}(\boldsymbol{\psi}) = \ln L_g(\boldsymbol{\psi}) + \ln \bar{w} + \frac{s_w^2}{2M\bar{w}^2}, \tag{31}$$

with $s_w^2 = (M-1)^{-1} \sum_{i=1}^{M} (w_i - \bar{w})^2$.

3. Denote

$$p(\mathbf{y}|\boldsymbol{\theta}) = \prod_{t=1}^{T} p_t, \qquad g(\mathbf{y}|\boldsymbol{\theta}) = \prod_{t=1}^{T} g_t,$$

where $p_t = p(y_t|\theta_t)$ and $g_t = g(y_t|\theta_t)$. The importance density is based on the linear Gaussian model

$$y_t = h_t + u_t, \qquad u_t \sim N(c_t, d_t), \tag{32}$$

where c_t and d_t are chosen such that the first and second derivatives of p_t and g_t are equal for $t = 1, \ldots, n$. These conditions lead to n nonlinear equations which we solve by a Newton-Raphson scheme of optimisation . This involves a sequence of Kalman filter smoothers. Convergence is usually fast.

Once the maximum likelihood estimates are obtained, smoothed estimates of volatility may be computed using Monte Carlo integration with importance sampling based on the weights w_i in (30).

The `S-PLUS` code to estimate the SV model (25) by maximizing the simulated log-likelihood function (31) is somewhat involved so we do not list all of the details here[12]. The key computations involve evaluating (32) and (31). For (32), the linear Gaussian model is an AR(1) model with autoregressive coefficient ϕ and heteroskedastic errors. This may be constructed using the `SsfPack` function `GetSsfArma` and then modifying the state space representation to allow for time varying variances. The solution for c_t and d_t involves a loop in which `SsfCondDens` is called repeatedly until convergence is achieved. To evaluate (31), the term $\ln L_g(\boldsymbol{\psi})$ is computed using `SsfLoglike` and the weights w_i are computed by drawing $\boldsymbol{\theta}^i$ using `SimSmoDraw`. As noted in Durbin & Koopman (2002), antithetic variates may be used to improve the efficiency of $\bar{w}$.

To illustrate, consider estimating the SV model (25) for daily log returns on the UK/US spot exchange rate over the period 10/1/1981 through 6/28/1985. The MLEs for the elements of $\boldsymbol{\psi} = (\sigma, \phi, \sigma_\eta^2)'$ are found to be $\hat{\sigma} = 0.6352$, $\hat{\phi} = 0.9744$ and $\hat{\sigma}_\eta^2 = 0.0278$. Figure 13 shows the absolute log returns along with the smoothed volatility estimates and 95% error bands.

7 Term Structure Models

This section illustrates how some common affine term structure models may be expressed in state space form, estimated and evaluated using the Kalman filter and smoothing algorithms in `S+FinMetrics/SsfPack`. The notation and examples are taken from Duan & Simonato (1999).

[12]The code is contained in the file `ssStochasticVolatility.ssc`, available on Eric Zivot's web page, and replicates the Ox code in the files `ssfnong.ox` and `sv_mcl_est.ox` written by Siem Jan Koopman.

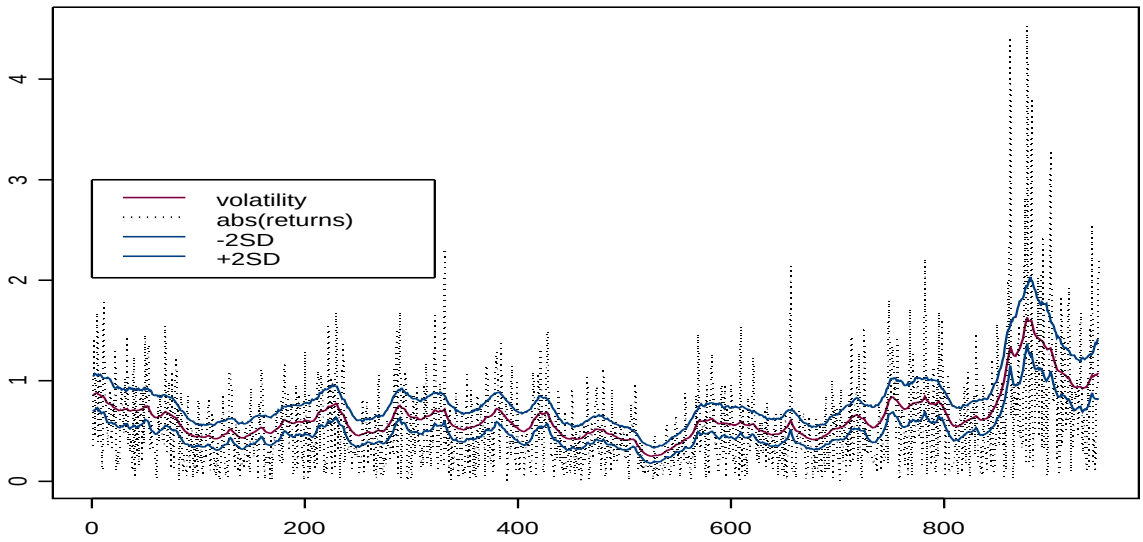

Figure 13: Smoothed volatility from maximum simulated likelihood estimates of SV model fit to daily UK/US log returns.

7.1 Affine Term Structure Models

Traditionally the study of the term structure of interest rates focuses on either the cross sectional aspect of the yield curve, or the time series properties of the interest rate. Recently, researchers have utilized state space models and Kalman filtering techniques to estimate affine term structure models by combining both time series and cross sectional data. For simple models, the state space representation is often linear and Gaussian and analysis is straightforward. For more general models, the unobserved state variables generally influence the variance of the transition equation errors making the errors non-Gaussian. In these cases, non-standard state space methods are necessary.

Duffie & Kan (1996) show that many of the theoretical term structure models, such as the Vasicek (1977) Ornstein-Uhlenbeck model, Cox, Ingersoll & Ross (1985) square root diffusion model and its multi-factor extensions (for example, see Chen & Scott (1993), Longstaff & Schwartz (1992) two-factor model, and Chen (1996) three factor model, are special cases of the class of affine term structure models. The class of affine term structure models is one in which the yields to maturity on default-free pure discount bonds and the instantaneous interest rate are affine (constant plus linear term) functions of m unobservable state variables $\mathbf{X}_t$, which are assumed to follow an affine diffusion process

$$d\mathbf{X}_t = \mathbf{U}(\mathbf{X}_t; \Psi)dt + \Sigma(\mathbf{X}_t; \Psi)d\mathbf{W}_t, \tag{33}$$

where $\mathbf{W}_t$ is an $m \times 1$ vector of independent Wiener processes ; Ψ is a $p \times 1$ vector of model specific parameters; $\mathbf{U}(\cdot)$ and $\Sigma(\cdot)$ are affine functions in $\mathbf{X}_t$ such that (33) has a unique solution. In general, the functions $\mathbf{U}(\cdot)$ and $\Sigma(\cdot)$ can be obtained as the solution to some ordinary differential equations. Only in special cases are closed form solutions available. In this class of models, the price at time t of a default-free pure discount bond with time to

maturity τ has the form

$$P_t(\mathbf{X}_t; \Psi, \tau) = A(\Psi, \tau) \exp\{-\mathbf{B}(\Psi, \tau)' \mathbf{X}_t\} \tag{34}$$

where $A(\tau, \Psi)$ is a scalar function and $\mathbf{B}(\tau, \Psi)$ is an $m \times 1$ vector function. The time-t continuously compounded yield-to-maturity on a pure discount bond with time to maturity τ is defined as

$$Y_t(\mathbf{X}_t; \Psi, \tau) = -\frac{\ln P_t(\mathbf{X}_t; \Psi, \tau)}{\tau}, \tag{35}$$

which, using (34), has the affine form

$$Y_t(\mathbf{X}_t; \Psi, \tau) = -\frac{\ln A(\Psi, \tau)}{\tau} + \frac{\mathbf{B}(\Psi, \tau)' \mathbf{X}_t}{\tau} \tag{36}$$

7.2 State Space Representation

Although (36) dictates an exact relationship between the yield $Y_t(\tau)$ and the state variables $\mathbf{X}_t$, in econometric estimation it is usually treated as an approximation giving rise to the measurement equation

$$Y_t(\tau) = -\frac{\ln A(\Psi, \tau)}{\tau} + \frac{\mathbf{B}(\Psi, \tau)' \mathbf{X}_t}{\tau} + \epsilon_t(\tau), \tag{37}$$

where ϵ_t is a normally distributed measurement error with zero mean and variance σ_τ^2. For any time to maturity τ, the above equation can be naturally treated as the measurement equation of a state space model, with $\mathbf{X}_t$ being the unobserved state variable. To complete the state space representation, the transition equation for $\mathbf{X}_t$ over a discrete time interval h needs to be specified. Defining $\Phi(\mathbf{X}_t; \Psi, h) = \text{var}(\mathbf{X}_{t+h} | \mathbf{X}_t)$, Duan & Simonato (1999) show that the transition equation for $\mathbf{X}_t$ has the form

$$\mathbf{X}_{t+h} = \mathbf{a}(\Psi, h) + b(\Psi, h)\mathbf{X}_t + \Phi(\mathbf{X}_t; \Psi, h)^{1/2} \boldsymbol{\eta}_{t+h} \tag{38}$$

where $\boldsymbol{\eta}_t \sim$ iid $N(\mathbf{0}, \mathbf{I}_m)$, and $\Phi(\mathbf{X}_t; \Psi, h)^{1/2}$ represents the Cholesky factorization of $\Phi(\mathbf{X}_t; \Psi, h)$.

In general, the state space model defined by (37) and (38) is non-Gaussian because the conditional variance of $\mathbf{X}_{t+h}$ in (38) depends on $\mathbf{X}_t$. Only for the special case in which $\Sigma(\cdot)$ in (33) is not a function of $\mathbf{X}_t$, is the conditional variance term $\Phi(\mathbf{X}_t; \Psi, h)$ also not a function of $\mathbf{X}_t$ and the state space model is Gaussian[13]. See Lund (1997) for a detailed discussion of the econometric issues associated with estimating affine term structure models using the Kalman filter. Although the quasi-maximum likelihood estimator of the model parameters based on the modified Kalman filter is inconsistent, the Monte Carlo results in Duan & Simonato (1999) and de Jong (2000) show that the bias is very small even for the moderately small samples likely to be encountered in practice.

[13]To estimate the non-Gaussian state space model, Duan & Simonato (1999) modify the Kalman filter recursions to incorporate the presence of $\Phi(\mathbf{X}_t; \Psi, h)$ in the conditional variance of $\boldsymbol{\eta}_{t+h}$. The `S+FinMetrics/SsfPack` functions `KalmanFil` and `SsfLoglike` can be modified to accommodate this modification.

7.3 Estimation of Vasicek's model

The data used for the following example are in the `S+FinMetrics` "`timeSeries`" `fama.bliss`, and consist of four monthly yield series over the period April, 1964 to December, 1997 for the U.S. Treasury debt securities with maturities of 3, 6, 12 and 60 months, respectively. This data was also used by Duan & Simonato (1999). All rates are continuously compounded rates expressed on an annual basis. These rates are displayed in Figure 14.

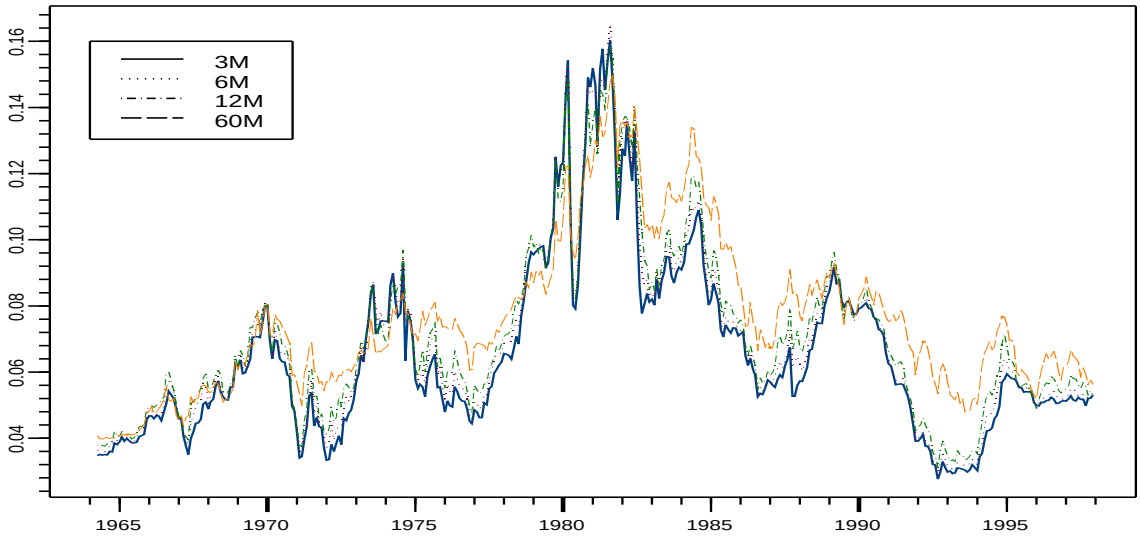

Figure 14: Monthly yields on U.S. treasury debt securities.

In the Vasicek (1977) model, the state variable driving the term structure is the instantaneous (short) interest rate, r_t, and is assumed to follow the mean-reverting diffusion process

$$dr_t = \kappa(\theta - r_t)dt + \sigma dW_t, \ \kappa \geq 0, \sigma > 0 \tag{39}$$

where W_t is a scalar Wiener process, θ is the long-run average of the short rate, κ is a speed of adjustment parameter, and σ is the volatility of r_t. Duan & Simonato (1999) show that the functions $A(\cdot), B(\cdot), a(\cdot), b(\cdot)$ and $\Phi(\cdot)$ have the form

$$\begin{aligned}
\ln A(\Psi,\tau) &= \gamma(B(\Psi,\tau)-\tau) - \frac{\sigma^2 B^2(\Psi,\tau)}{4\kappa}, \ B(\Psi,\tau) = \frac{1}{\kappa}(1-\exp(-\kappa\tau)) \\
\gamma &= \theta + \frac{\sigma\lambda}{\kappa} - \frac{\sigma^2}{2\kappa^2} \\
a(\Psi,h) &= \theta(1-\exp(-\kappa h)), \ b(\Psi,h) = \exp(-\kappa h) \\
\Phi(X_t;\Psi,h) &= \Phi(\Psi,h) = \frac{\sigma^2}{2\kappa}(1-\exp(-2\kappa h))
\end{aligned}$$

where λ is the risk premium parameter. The model parameters are $\Psi = (\kappa, \theta, \sigma, \lambda)'$. Notice that for the Vasicek model, $\Phi(X_t;\Psi,h) = \Phi(\Psi,h)$ so that the state variable r_t does not influence the conditional variance of transition equation errors, the state space model is Gaussian.

The state space representation of the Vasicek model has system matrices

$$\begin{aligned} \boldsymbol{\delta} &= \begin{pmatrix} a(\Psi,h) \\ -\ln A(\Psi,\tau_1)/\tau_1 \\ \vdots \\ -\ln A(\Psi,\tau_4)/\tau_4 \end{pmatrix}, \ \Phi = \begin{pmatrix} b(\Psi,h) \\ B(\Psi,\tau_1)/\tau_1 \\ \vdots \\ B(\Psi,\tau_4)/\tau_4 \end{pmatrix} \\ \Omega &= \text{diag}(\Phi(\Psi,h), \sigma^2_{\tau_1}, \ldots, \sigma^2_{\tau_4}) \end{aligned} \tag{40}$$

and initial value matrix

$$\Sigma = \begin{pmatrix} \theta \\ \sigma^2/2\kappa \end{pmatrix}$$

based on the stationary distribution of the short rate in (39). Notice that this a multivariate state space model.

A function to compute the state space form of the Vasicek model for a given set of parameters Ψ, number of yields $\tau_1, \ldots, \tau_N$, and sampling frequency h is given in listing 14. Notice that the exponential transformation is used for those parameters that should be positive, and, since the data in `fama.bliss` are monthly, the default length of the discrete sampling interval, h, is set to 1/12.

An implementation of the Vasicek model is provided in listing 15. Specific starting values for the parameters

$$\boldsymbol{\varphi} = (\ln\kappa, \ln\theta, \ln\sigma, \lambda, \ln\sigma_{\tau_1}, \ln\sigma_{\tau_2}, \ln\sigma_{\tau_3}, \ln\sigma_{\tau_4})'$$

and the maturity specification for the yields are given and maximum likelihood estimates for the parameters are obtained using `SsfFit`. The maximum likelihood estimates and asymptotic standard errors for the model parameters

$$\boldsymbol{\theta} = (\kappa, \theta, \sigma, \lambda, \sigma_{\tau_1}, \sigma_{\tau_2}, \sigma_{\tau_3}, \sigma_{\tau_4})'$$

computed using the delta method are ????

These results are almost identical to those reported by Duan & Simonato (1999). All parameters are significant at the 5% level except the measurement equation standard deviation for the six month maturity yield. The largest measurement equation error standard deviation is for the sixty month yield, indicating that the model has the poorest fit for this yield. The short rate is mean reverting since $\hat{\kappa} > 0$, and the long-run average short rate is $\hat{\theta} = 5.74\%$ per year. The estimated risk premium parameter, $\hat{\lambda} = 0.3477$, is positive indicating a positive risk premium for bond prices.

The smoothed estimates of the short-rate and the yields are computed using `SsfCondDens` with `task="STSMO"`. Figure 15 gives the smoothed estimate of the instantaneous short rate r_t from (39). The differences between the actual and smoothed yield estimates are displayed in Figure 16. The model fits well on the short end of the yield curve but poorly on the long end.

As another check on the fit of the model, the presence of serial correlation in the standardized innovations is tested using the Box-Ljung modified Q-statistic (computed using the `S+FinMetrics` function `autocorTest`) The null of no serial correlation is easily rejected for the standardized innovations of all yields.

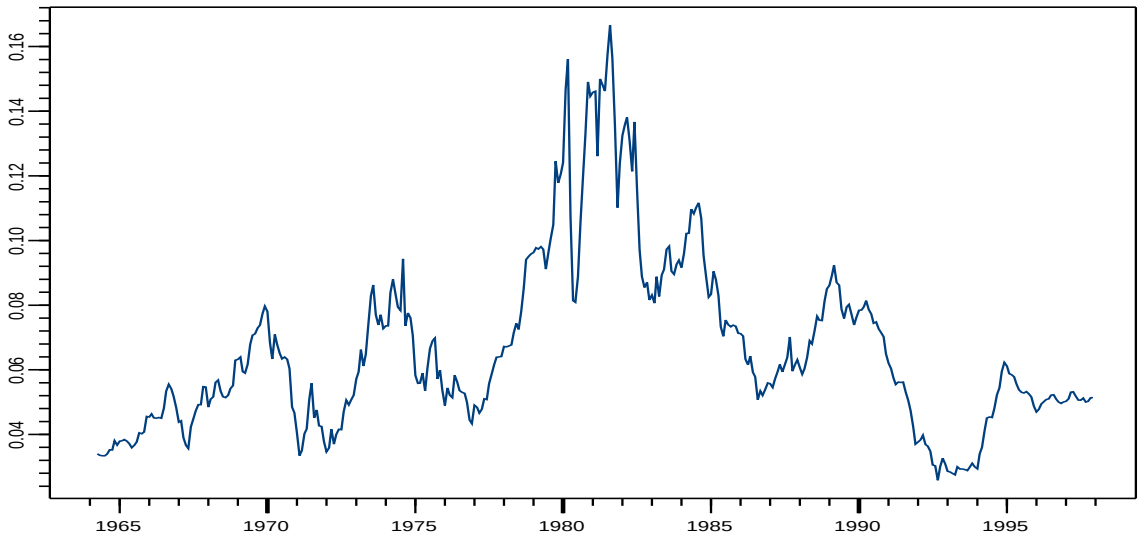

Figure 15: Smoothed estimate of short rate r_t from (39).

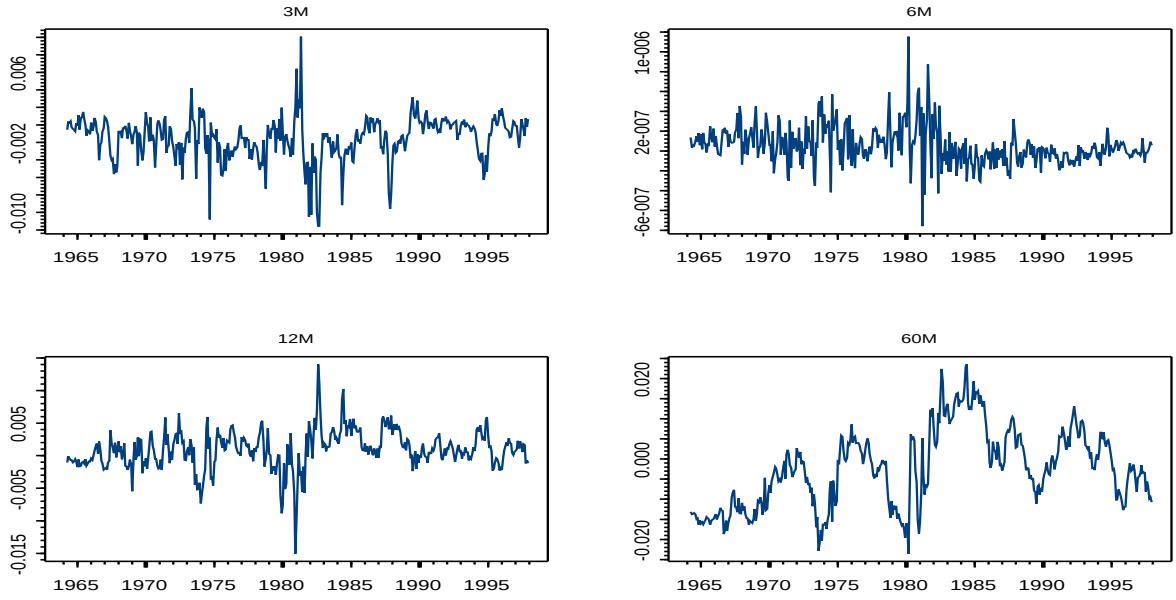

Figure 16: Smoothed estimated of yields from Vasicek term structure model.

```
vasicek.ssf = function(param, tau=NULL, freq=1/52)
{
## 1. Check for valid input.
  if (length(param) < 5)
    stop("param must have length greater than 4.")
  N = length(param) - 4
  if (length(tau) != N)
    stop("Length of tau is inconsistent with param.")
## 2. Extract parameters and impose constraints.
  Kappa = exp(param[1])  ## Kappa > 0
  Theta = exp(param[2])  ## Theta > 0
  Sigma = exp(param[3])  ## Sigma > 0
  Lamda = param[4]
  Var   = exp(param[1:N+4])  ## meas eqn stdevs
## 3. Compute Gamma, A, and B.
  Gamma = Theta + Sigma * Lamda / Kappa - Sigma^2 / (2 * Kappa^2)
  B = (1 - exp(-Kappa * tau)) / Kappa
  lnA = Gamma * (B - tau) - Sigma^2 * B^2 / (4 * Kappa)
## 4. Compute a, b, and Phi.
  a = Theta * (1 - exp(-Kappa * freq))
  b = exp(-Kappa * freq)
  Phi = (Sigma^2 / (2 * Kappa)) * (1 - exp(-2 * Kappa * freq))
## 5. Compute the state space form.
  mDelta = matrix(c(a, -lnA/tau), ncol=1)
  mPhi = matrix(c(b, B/tau), ncol=1)
  mOmega = diag(c(Phi, Var^2))
## 6. Duan and Simonato used this initial setting.
  A0 = Theta
  P0 = Sigma * Sigma / (2*Kappa)
  mSigma = matrix(c(P0, A0), ncol=1)
## 7. Return state space form.
  ssf.mod = list(mDelta=mDelta, mPhi=mPhi, mOmega=mOmega, mSigma=mSigma)
  CheckSsf(ssf.mod)
}
```

Listing 14: Computing the state space form for Vasicek model.

8 Conclusion

This paper provides an overview of the `SsfPack` state space functions available in `S+FinMetrics`. The functions can be used to implement the state space methods and algorithms presented in, for example, Durbin & Koopman (2001). We also illustrate the use of the `SsfPack` functions through several examples from macroeconomics and finance. With powerful easy-to-use software, a wide variety of state space models are straightforward to estimate and analyze.

```
> start.vasicek = c(log(0.1), log(0.06), log(0.02), 0.3, log(0.003),
+ log(0.001), log(0.003), log(0.01))
> names(start.vasicek) = c("ln.kappa","ln.theta","ln.sigma","lamda",
+ "ln.sig.3M","ln.sig.6M","ln.sig.12M","ln.sig.60M")
> start.tau = c(0.25, 0.5, 1, 5)

> ans.vasicek = SsfFit(start.vasicek, fama.bliss, vasicek.ssf,
+ tau=start.tau, freq=1/12, trace=T,
+ control=nlminb.control(abs.tol=1e-6, rel.tol=1e-6,
+ x.tol=1e-6, eval.max=1000, iter.max=500))
Iteration  0 : objective =  -6347.453
...
Iteration  37 : objective =  -6378.45

> ssf.fit = vasicek.ssf(ans.vasicek$parameters,tau=start.tau,freq=1/12)

Log-likelihood:  -6378.45
1620 observations
Parameters:
             Value Std. Error t value
  kappa 0.11880000  0.0106300 11.1700
  theta 0.05729000  0.0269000  2.1300
  sigma 0.02139000  0.0007900 27.0800
  lamda 0.34800000  0.1500000  2.3200
 sig.3M 0.00283500  0.0001011 28.0500
 sig.6M 0.00001773  0.0001148  0.1544
sig.12M 0.00301700  0.0001083 27.8600
sig.60M 0.00989800  0.0003703 26.7300

> autocorTest(KalmanFil(fama.bliss,ssf.fit)$std.innov)

Test for Autocorrelation: Ljung-Box
Null Hypothesis: no autocorrelation
Test Statistics:
                 3M        6M       12M       60M
Test Stat   80.9471  282.4316  756.3304 3911.7736
  p.value    0.0000    0.0000    0.0000    0.0000

Dist. under Null: chi-square with 26 degrees of freedom
   Total Observ.: 405
```

Listing 15: An analysis of the Vasicek model.

References

Beveridge, S. & Nelson, C. R. (1981), 'A new approach to decomposition of economic time series into permanent and transitory components with particular attention to measurement of the business cycle', *J. Monetary Economics* **7**, 151–174.

Brown, R. L., Durbin, J. & Evans, J. M. (1975), 'Techniques of testing the constancy of regression relationships over time', *J. Royal Statistical Society B* **37**, 141–92.

Carter, C. K. & Kohn, R. (1994), 'On Gibbs sampling for state space models', *Biometrika* **81**, 541–53.

Chan, N. H. (2002), *Time series: application to finance*, John Wiley & Sons, New York.

Chen, L. (1996), *Stochastic Mean and Stochastic Volatility - A Three-Factor Model of the Term Structure of Interest Rates and Its Application to the Pricing of Interest Rate Derivatives*, Blackwell Publishers, Oxford.

Chen, R. & Scott, L. (1993), 'Maximum likelihood estimation for a multifactor equilibrium model of the term structure of interest rates', *J. Fixed Income* **3**, 14–31.

Clark, P. K. (1987), 'The cyclical component of u.s. economic activity', *The Quarterly Journal of Economics* **102**, 797–814.

Cox, J. C., Ingersoll, J. E. & Ross, S. A. (1985), 'A theory of the term structure of interest rates', *Econometrica* **53**, 385–407.

de Jong, F. (2000), 'Time series and cross-section information in affine term - structure models', *J. Business and Economic Statist.* **18**, 300–314.

de Jong, P. & Shephard, N. (1995), 'The simulation smoother for time series models', *Biometrika* **82**, 339–50.

Doornik, J. A. (1999), *Ox: an object-oriented matrix language*, Timberlake Consultants Press, London.

Duan, J. C. & Simonato, J. G. (1999), 'Estimating exponential-affine term structure models by kalman filter', *Review of Quantitative Finance and Accounting* **13**, 111–135.

Duffie, D. & Kan, R. (1996), 'A yield-factor model of interest rates', *Mathematical Finance* **6**, 379–406.

Durbin, J. & Koopman, S. J. (1997), 'Monte Carlo maximum likelihood estimation of non-Gaussian state space model', *Biometrika* **84**, 669–84.

Durbin, J. & Koopman, S. J. (2001), *Time Series Analysis by State Space Methods*, Oxford University Press, Oxford.

Durbin, J. & Koopman, S. J. (2002), 'A simple and efficient smoother for state space time series analysis', *Biometrika* **89**, 603–616.

Fruhwirth-Schnatter, S. (1994), 'Applied state space modeling of Non-Gaussian time series using integration-based Kalman filtering', *Statistical Computing* **4**, 259–269.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton University Press, Princeton.

Harvey, A. C. (1985), 'Trends and cycles in macroeconomic time series', *J. Business and Economic Statist.* **3**, 216–227.

Harvey, A. C. (1989), *Forecasting, Structural Time Series Models and the Kalman Filter*, Cambridge University Press, Cambridge.

Harvey, A. C. (1993), *Time Series Models*, 2nd edn, Harvester Wheatsheaf, Hemel Hempstead.

Harvey, A. C. (2001), 'Testing in unobserved components models', *J. Forecasting* **20**, 1–19.

Harvey, A. C., Ruiz, E. & Shephard, N. (1994), 'Multivariate stochastic variance models', *Rev. Economic Studies* **61**, 247–64.

Harvey, A. C. & Streibel, M. (1997), Testing for nonstationary unobserved components. Unpublished manuscript, Department of Economics, Cambridge University.

Hull, J. & White, A. (1987), 'The pricing of option on assets with stochastic volatilities', *J. Finance* **42**, 281–300.

Kim, C. J. & Nelson, C. R. (1999), *State Space Models with Regime Switching*, MIT Press, Cambridge, Massachusetts.

Koopman, S. J., Shephard, N. & Doornik, J. A. (1999), 'Statistical algorithms for models in state space form using SsfPack 2.2', *Econometrics Journal* **2**, 113–66. http://www.ssfpack.com/.

Koopman, S. J., Shephard, N. & Doornik, J. A. (2001), Ssfpack 3.0beta: Statistical algorithms for models in state space. Unpublished manuscript, Departement of Economics and Business Administration, Free University, Amsterdam.

Longstaff, F. & Schwartz, E. (1992), 'Interest rate volatility and the term structure: A two-factor general equilibrium model', *J. Finance* **47**, 1259–1282.

Lund, J. (1997), Non-linear Kalman filtering techniques for term - structure models. Discussion paper, The Aarhus School of Business, Denmark.

Morley, J. C. (2002), 'A state-space approach to calculating the Beveridge-Nelson decomposition', *Economic Letters* **75**, 123–127.

Morley, J. C., Nelson, C. R. & Zivot, E. (2002), 'Why are Beveridge-Nelson and unobserved components decompositions of GDP so different?', *Review of Economics and Statistics* **forthcoming**.

Neumann, T. (2002), Time-varying coefficient models: A comparison of alternative estimation strategies. Unpublished manuscript, DZ Bank, Germany.

Proietti, T. (2002), Some reflections on trend-cycle decompositions with correlated components. Unpublished manuscript, European University Institute, Florence.

Ripley, B. D. (1987), *Stochastic Simulation*, Wiley, New York.

Shephard, N. & Pitt, M. K. (1997), 'Likelihood analysis of non-Gaussian measurement time series', *Biometrika* **84**, 653–67.

Shumway, R. H. & Stoffer, D. S. (2000), *Time Series Analysis and Its Applications*, Springer-Verlag, New York.

Vasicek, O. (1977), 'An equilibrium characterization of the term structure', *J. Financial Economics* **5**, 177–188.

West, M. & Harrison, J. (1997), *Bayesian Forecasting and Dynamic Models*, 2nd edn, Springer-Verlag, New York.

Zivot, E. & Wang, J. (2003), *Modeling Financial Time Series with S-PLUS*, Springer-Verlag, New York.

Index