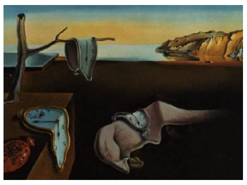

Journal of Statistical Software

August 2015, Volume 66, Issue 9. *http://www.jstatsoft.org/*

SSMMATLAB: A Set of MATLAB Programs for the Statistical Analysis of State Space Models

Víctor Gómez
Ministry of Finance and Public Administrations, Spain

Abstract

This article discusses and describes **SSMMATLAB**, a set of programs written by the author in MATLAB for the statistical analysis of state space models. The state space model considered is very general. It may have univariate or multivariate observations, time-varying system matrices, exogenous inputs, regression effects, incompletely specified initial conditions, such as those that arise with cointegrated VARMA models, and missing values. There are functions to put frequently used models, such as multiplicative VARMA models, VARMAX models in echelon form, cointegrated VARMA models, and univariate structural or ARIMA model-based unobserved components models, into state space form. There are also functions to implement the Hillmer-Tiao canonical decomposition and the smooth trend and cycle estimation proposed by Gómez (2001). Once the model is in state space form, other functions can be used for likelihood evaluation, model estimation, forecasting and smoothing. A set of examples is presented in the **SSMMATLAB** manual to illustrate the use of these functions.

Keywords: state space models, VARMAX models, cointegrated VARMA models, Kalman filter, unobserved components, MATLAB.

1. Introduction

This article describes **SSMMATLAB** (Gómez 2014), a set of programs written by the author in MATLAB (The MathWorks Inc. 2014) for the statistical analysis of time series that are assumed to follow state space models. The series can be univariate or multivariate and the state space model can be very general. It may have time-varying system matrices, exogenous inputs, regression effects, incompletely specified initial conditions, such as those that arise with cointegrated VARMA (vector autoregressive moving average) models, and missing values.

The motivation for **SSMMATLAB** is to provide the time series analyst with a set of programs written in MATLAB that will allow him to work with general state space models. Since many time series models can be put into state space form, special functions have been written for the most usual ones, such as multiplicative VARMA models, VARMAX models in echelon form, cointegrated VARMA models, univariate structural models, like those considered by Harvey (1989, Chapter 4) or Kitagawa and Gersch (1996), and ARIMA model-based (AMB) unobserved components models (Gómez and Maravall 2001b, Chapter 8). But if the user intends to work with more sophisticated state space models that are not available in standard commercial packages for time series analysis or econometrics, he can program his own model in **SSMMATLAB** and carry out model estimation, interpolation, forecasting and smoothing.

State space methods have been implemented in some statistical software packages, such as **STAMP** (Koopman, Harvey, Doornik, and Shephard 2009; Mendelssohn 2011), **REGCMPNT** (Bell 2011), R (Petris and Petrone 2011), State Space Models (**SSM**) toolbox for MATLAB (Peng and Aston 2011), SAS (Selukar 2011), EViews (Van den Bossche 2011), GAUSS (Aptech Systems, Inc. 2006), Stata (Drukker and Gates 2011), gretl (Lucchetti 2011), **RATS** (Doan 2011) and **SsfPack** (Pelagatti 2011). See the Special Volume 41 (Commandeur, Koopman, and Ooms 2011) of the *Journal of Statistical Software* for a discussion of these packages.

SSMMATLAB provides functions similar to the ones contained in previous packages for linear state space models. In addition, it provides functions for identification, estimation, forecasting and smoothing of VARMAX models, possibly in state space echelon form, and of cointegrated VARMA models. It provides also functions to design digital filters and to estimate smooth trends and cycles in an AMB approach. Moreover, the general functions in **SSMMATLAB** allow, with careful programming, to do at least all the things that the previous packages can do with linear state space models.

In Section 2, the state space model will be described. In Section 3, the functions to put into state space form multiplicative VARMA models, VARMAX models in echelon form, univariate structural models and AMB unobserved components models will be documented. In Section 4, the identification of VARMAX(p,q,r) models and VARMAX models in echelon form will be considered. Also in Section 4, the estimation of VARX models, the Hannan and Rissanen (1982) method to estimate VARMAX models, as well as the conditional and the exact methods to estimate VARMAX models will be described. The functions for likelihood evaluation, computation of recursive residuals, model estimation, forecasting and smoothing will be described in Section 5. Finally, in Section 6, reference will be made to some examples and case studies using **SSMMATLAB**.

2. The state space model

The state space model considered in **SSMMATLAB** is

$$\begin{aligned} \alpha_{t+1} &= W_t\beta + T_t\alpha_t + H_t\epsilon_t, && (1)\\ Y_t &= X_t\beta + Z_t\alpha_t + G_t\epsilon_t, \qquad t = 1,\ldots,n, && (2) \end{aligned}$$

where $\{Y_t\}$ is a multivariate process with $Y_t \in \mathbb{R}^p$, W_t, T_t, H_t, X_t, Z_t and G_t are time-varying deterministic matrices, $\beta \in \mathbb{R}^q$ is a constant bias vector, $\alpha_t \in \mathbb{R}^r$ is the state vector, and $\{\epsilon_t\}$ is a sequence of uncorrelated stochastic vectors, $\epsilon_t \in \mathbb{R}^s$, with zero mean and common covariance matrix $\sigma^2 I$. The initial state vector α_1 is specified as

$$\alpha_1 = c + W_0\beta + a + A\delta, \qquad (3)$$

where c has zero mean and covariance matrix $\sigma^2\Omega$, a is a constant vector, A is a constant matrix, and δ has zero mean and covariance matrix kI with $k \to \infty$ (diffuse). It is assumed that the vectors c and δ are mutually orthogonal and that α_1 is orthogonal to the $\{\epsilon_t\}$ sequence. The vector δ in (3) models uncertainty with respect to the initial conditions. For example, a multivariate random walk model, $Y_t = Y_{t-1} + A_t$, where $\{A_t\}$ is a zero mean normally distributed sequence with common covariance matrix Σ, can be put into state space form as

$$\begin{aligned} \alpha_{t+1} &= \alpha_t + L\epsilon_t, \\ Y_t &= \alpha_t, \end{aligned}$$

where $\Sigma = LL^\top$ is the Cholesky decomposition of Σ, $L\epsilon_t = A_{t+1}$, $\sigma^2 = 1$, and $\alpha_1 = \delta$.

The state space model (1) and (2) is very general. For example, it can be used in macroeconomics for analyzing time-varying parameter VARs as in Primiceri (2005) as well as for forecasting using mixed frequency data as in Aruoba, Diebold, and Scotti (2009).

It is not restrictive that the same term, ϵ_t, appears in both equations. To see this, suppose the state space model

$$\begin{aligned} \alpha_{t+1} &= W_t\beta + T_t\alpha_t + J_t u_t, & (4)\\ Y_t &= X_t\beta + Z_t\alpha_t + v_t, \qquad t = 1, \ldots, n, & (5) \end{aligned}$$

where W_t, T_t, J_t, X_t and Z_t are time-varying deterministic matrices,

$$\mathsf{E}\left\{ \begin{bmatrix} u_t \\ v_t \end{bmatrix} \begin{bmatrix} u_s^\top, v_s^\top \end{bmatrix} \right\} = \sigma^2 \begin{bmatrix} Q_t & S_t \\ S_t^\top & R_t \end{bmatrix} \delta_{ts},$$

δ_{ts} denotes the Kronecker delta, $u_t \in \mathbb{R}^s$, $v_t \in \mathbb{R}^p$, $\mathsf{E}(u_t) = 0$, $\mathsf{E}(v_t) = 0$ and α_1 is as before. To pass from the state space representation (4) and (5) to (1) and (2), let V_t be the symmetric covariance matrix

$$V_t = \mathsf{COV} \begin{bmatrix} J_t u_t \\ v_t \end{bmatrix} = \begin{bmatrix} J_t Q_t J_t^\top & J_t S_t \\ S_t^\top J_t^\top & R_t \end{bmatrix}.$$

Every symmetric matrix, M, satisfies the decomposition $M = M^{1/2}\left(M^{1/2}\right)^\top$, where $M^{1/2}$ is a square nonunique matrix. For example, let O be an orthogonal matrix such that $O^\top M O = D$, where D is a diagonal matrix. Then, we can take $M^{1/2} = OD^{1/2}$, where $D^{1/2}$ is the matrix obtained from D by replacing its nonzero elements with their square roots. This choice of $M^{1/2}$ has the advantage of being valid and numerically stable even if M is singular. It follows from this that we can take $(G_t^\top, H_t^\top)^\top = V_t^{1/2}$.

It could be useful to compare the state space model used in **SSMMATLAB** with the ones used in some other statistical software packages. For example, the state space model considered in MATLAB corresponds to a VARMAX model. It is of the form

$$\begin{aligned} \alpha_{t+1} &= T\alpha_t + Gu_t + Ka_t, \\ Y_t &= Z\alpha_t + Hu_t + a_t, \end{aligned}$$

where $\{A_t\}$ is an innovations sequence (uncorrelated, zero mean and with common covariance matrix) and $\{u_t\}$ is a sequence of exogenous variables. Since $Gu_t = \text{vec}(Gu_t) = (u_t^\top \otimes I)\text{vec}(G)$

and $Hu_t = \text{vec}(Hu_t) = (u_t^\top \otimes I)\text{vec}(H)$, if we define $\beta_h = \text{vec}(H)$, $\beta_g = \text{vec}(G)$, $\beta = (\beta_h^\top, \beta_g^\top)^\top$, $W_t = (0, u_t^\top \otimes I)$ and $V_t = (u_t^\top \otimes I, 0)$, we see that the previous state space model is as (4) and (5) but with some restrictions on it. This state space model does not encompass the structural model of Harvey (1989, Chapter 4) for example. Neither includes it state space models with time-varying coefficient matrices or multiplicative VARMA models, described in Section 3.1.

The state space model considered in the State Space Models (**SSM**) toolbox for MATLAB is of the form (4) and (5), but there is no $W_t\beta$ term and the errors u_t and v_t are uncorrelated. The state space models in **STAMP** and **REGCMPNT** are also as in the State Space Models (**SSM**) toolbox for MATLAB. In the state space model considered in Stata the system matrices are time invariant and the errors are uncorrelated.

To the best of this author's knowledge none of the software packages mentioned in Section 1 handles either VARMAX models in echelon form and Kronecker indices or the use of high pass and band pass filters in a model-based approach.

3. Putting some common models into state space form

Given that the state space model considered by **SSMMATLAB** is very general, it is advisable to have some functions that allow to put some of the most commonly used models in practice into state space form. In this section, we will document some functions that can be used for this purpose. The user can of course modify these functions or write his own functions in order to suit his needs, but in many cases these functions will be sufficient.

3.1. Multiplicative VARMA models

Theoretical introduction

Suppose a vector ARMA (VARMA) model given by

$$Y_t + \Phi_1 Y_{t-1} + \cdots + \Phi_p Y_{t-p} = A_t + \Theta_1 A_{t-1} + \cdots + \Theta_q A_{t-q}, \tag{6}$$

that can be written more compactly as

$$\Phi(B)Y_t = \Theta(B)A_t,$$

where $\Phi(B) = I + \Phi_1 B + \cdots + \Phi_p B^p$, $\Theta(B) = I + \Theta_1 B + \cdots + \Theta_q B^q$ and B is the backshift operator, $BY_t = Y_{t-1}$.

The model (6) is stationary if the roots of $\det[\Phi(z)]$ are all outside the unit circle and the model is invertible when the roots of $\det[\Theta(z)]$ are all outside the unit circle.

One possible state space representation is

$$T = \begin{bmatrix} -\Phi_1 & I & 0 & \cdots & 0 \\ -\Phi_2 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\Phi_{r-1} & 0 & 0 & \cdots & I \\ -\Phi_r & 0 & 0 & \cdots & 0 \end{bmatrix}, \qquad H = \begin{bmatrix} \Theta_1 - \Phi_1 \\ \Theta_2 - \Phi_2 \\ \vdots \\ \Theta_{r-1} - \Phi_{r-1} \\ \Theta_r - \Phi_r \end{bmatrix} \Sigma^{1/2}, \tag{7}$$

where $r = \max(p, q)$, $\Phi_i = 0$ if $i > p$, $\Theta_i = 0$ if $i > q$, $G = \Sigma^{1/2}$, $Z = [I, 0, \dots, 0]$ and $\mathsf{VAR}(A_t) = \Sigma^{1/2}\left(\Sigma^{1/2}\right)^\top$ is the Cholesky decomposition of $\mathsf{VAR}(A_t)$. This is the state space representation used in **SSMMATLAB**.

To obtain initial conditions for the Kalman filter, the mean and the covariance matrix of the initial state vector are needed. If the series is stationary, the mean is obviously zero. As for the covariance matrix, letting $\mathsf{VAR}(\alpha_1) = V$, the matrix V satisfies the Lyapunov equation

$$V = TVT^\top + HH^\top,$$

where T and H are given by (7). In **SSMMATLAB**, this equation is solved in a numerically stable manner.

The VARMA models considered in **SSMMATLAB** can be multiplicative, i.e., they can be of the form

$$(I + \phi_1 B + \cdots + \phi_p B^p)(I + \Phi_1 B^s + \cdots + \Phi_P B^{Ps})Y_t = \\ (I + \theta_1 B + \cdots + \theta_q B^q)(I + \Theta_1 B^s + \cdots + \Theta_Q B^{Qs})A_t, \quad (8)$$

where s is the number of observations per year.

There also exists the possibility to incorporate regression variables into the model. More specifically, models of the form

$$Y_t = X_t\beta + U_t,$$

where U_t follows a VARMA model (8) and β is a vector of regression coefficients, can be handled in **SSMMATLAB**.

SSMMATLAB implementation

In **SSMMATLAB**, the matrix polynomials in (8) are given as three dimensional arrays in MATLAB. For example, the matrix polynomial

$$\Phi(z) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -.5 & .2 \\ 0 & -.7 \end{bmatrix} z$$

would be defined in MATLAB as

```
phi(:, :, 1) = eye(2); phi(:, :, 2) = [-.5  .2; 0. -.7];
```

Once the model (8) has been defined in MATLAB, we can use the following function to put this model into state space form.

```
function [str, ferror] = suvarmapqPQ(phi, th, Phi, Th, Sigma, freq)
```

Fixing of parameters If the user wants to fix some parameters in a VARMA model, he should proceed as follows. Assuming that the model has been defined and, therefore, the structure `str` exists, the appropriate parameters in the AR and MA matrix polynomials should be first set to their fixed values. Then, function `suvarmapqPQ` should be run. Finally, the corresponding parameters in the matrix polynomials `str.phin`, `str.thn`, `str.Phin` or `str.Thn` should be set to zero and function `fixvarmapqPQ` should be called. For example, the following sequence of commands can be used to fix the parameters `phi(1, 2, 2)` and `th(2, 1, 2)` to zero in a bivariate VARMA model.

```
phi(1, 2, 2) = 0.; th(2, 1, 2) = 0.;
[str, ferror] = suvarmapqPQ(phi, th, Phi, Th, Sigma, freq);
str.phin(1, 2, 2) = 0; str.thn(2, 1, 2) = 0;
[str, ferror] = fixvarmapqPQ(str);
```

Model estimation Once the model has been defined and, therefore, the structure `str` exists, it can be estimated. Before estimation, the user has to decide whether there are fixed parameters in the model or not. How to fix some parameters has been explained in the previous section. The parameters to estimate are in the array `str.xv`, and the fixed parameters are in `str.xf`. It is assumed that the values entered by the user for the parameters to be estimated are reasonable initial values. In any case, the estimation function checks at the beginning whether the model is stationary and invertible and issues a warning message if the model is nonstationary or noninvertible.

One method that usually provides good initial estimates for VARMA models is the Hannan and Rissanen (1982) method for univariate series or its generalization to multivariable series (Hannan and Kavalieris 1984, 1986). This method has been recently implemented in **SSMMATLAB** and will be described later.

It should be emphasized that in **SSMMATLAB**, the (1, 1) parameter in the covariance matrix of the innovations is always concentrated out of the likelihood.

During the estimation process, each time the log-likelihood is evaluated **SSMMATLAB** checks whether the model is stationary and invertible. In case any of these conditions is not satisfied, the variable in the corresponding matrix polynomial is multiplied by a small number so that all its roots are outside the unit circle. This guarantees that the solution will always be stationary and invertible.

The following function can be used for parameter estimation.

```
function  result = varmapqPQestim(y, str, Y)
```

After model estimation, the function `pr2varmapqPQ` can be used to set up the estimated model in VARMA form. For example, the following commands achieve this.

```
xvf = result.xvf; xf = result.xf;
[phif, thf, Phif, Thf, Lf, ferror] = pr2varmapqPQ(xvf, xf, str)
```

Recursive residuals As explained in Section 5.2, recursive residuals can be computed using function `scakff`. For example, the following commands can be used to compute recursive residuals after estimation of a VARMA model, assuming that `phif`, `thf`, `Phif` and `Thf` are the estimated matrix polynomials in the model, `Sigmaf` is the estimated covariance matrix of the innovations and `freq` is the number of observations per year.

```
Sigmaf = Lf * Lf';
[strf, ferror] = suvarmapqPQ(phif, thf, Phif, Thf, Sigmaf, freq);
[nalpha, mf] = size(strf.T);
i = [nalpha 0 0 0];
[ins, ferror] = mlyapunov(strf.T, strf.H * strf.H', .99);
X = Y; W = [];
```

```
T = strf.T; Z = strf.Z; G = strf.G; H = strf.H;
[Xt, Pt, g, M, initf, recrs] = scakff(y, X, Z, G, W, T, H, ins, i);
```

Forecasting As described in Section 5.4, forecasts can be obtained using function `ssmpred`. For example, the following commands can be used to obtain twelve forecasts after estimation of a bivariate regression model with VARMA errors. It is assumed that `phif`, `thf`, `Phif` and `Thf` are the estimated matrix polynomials in the model and `freq` is the number of observations per year. The variables `hb`, `Mb`, `A` and `P` are needed and they are in structure `result`. `hb` is the vector of regression estimates and `Mb` is the matrix of standard errors. `A` is the possibly augmented estimated state vector, $x_{t|t-1}$, obtained with the Kalman filter at the end of the sample and `P` is the matrix of standard errors.

```
[strf, ferror] = suvarmapqPQ(phif, thf, Phif, Thf, Sigmaf, freq);
T = strf.T; Z = strf.Z; G = strf.G; H = strf.H;
Xp = Y; Wp = [];
hb = result.h; Mb = result.H; A = result.A; P = result.P;
npr = 12;
m = 2;
[pry, mypr, alpr, malpr] = ssmpred(npr, m, A, P, Xp, Z, G, Wp, T, H, hb, ...
  Mb);
```

3.2. VARMA and VARMAX models in echelon form

Theoretical introduction

Suppose the s-dimensional VARMA model

$$\Phi(B)Y_t = \Theta(B)A_t, \tag{9}$$

where $\Phi(z) = \Phi_0 + \Phi_1 z + \cdots + \Phi_l z^l$, $\Theta(z) = \Theta_0 + \Theta_1 z + \cdots + \Theta_l z^l$, $\Theta_0 = \Phi_0$, and Φ_0 is lower triangular with ones in the main diagonal. We say that the VARMA model (9) is in `echelon form` if we can express the matrix polynomials $\Phi(z)$ and $\Theta(z)$ as follows

$$\phi_{ii}(z) = 1 + \sum_{j=1}^{n_i} \phi_{ii,j} z^j, \quad i = 1, \ldots, s, \tag{10}$$

$$\phi_{ip}(z) = \sum_{j=n_i-n_{ip}+1}^{n_i} \phi_{ip,j} z^j, \quad i \neq p, \tag{11}$$

$$\theta_{ip}(z) = \sum_{j=0}^{n_i} \theta_{ip,j} z^j, \quad i, p = 1, \ldots, s, \tag{12}$$

where $\Theta_0 = \Phi_0$ and

$$n_{ip} = \begin{cases} \min\{n_i + 1, n_p\} & \text{for} \quad i > p \\ \min\{n_i, n_p\} & \text{for} \quad i < p \end{cases} \qquad i, p = 1, \ldots, s.$$

Note that n_{ip} specifies the number of free coefficients in the polynomial $\phi_{ip}(z)$ for $i \neq p$. The numbers $\{n_i : i = 1, \ldots, s\}$ are called `Kronecker indices` and $l = \max\{n_i : i = 1, \ldots, s\}$.

The `state space echelon form` corresponding to the previous VARMA echelon form is

$$\begin{aligned} x_{t+1} &= Fx_t + KA_t, & (13) \\ Y_t &= Hx_t + A_t. & (14) \end{aligned}$$

It is described in more detail in the **SSMMATLAB** manual. Note that the A_t in the state space form (13) and (14) are the model innovations.

In a similar way, VARMAX models in echelon form can be handled in **SSMMATLAB**, as described in the manual.

SSMMATLAB implementation

As in the case of VARMA models, in **SSMMATLAB** the matrix polynomials of a VARMA or VARMAX model in echelon form are given as three dimensional arrays in MATLAB.

Once the Kronecker indices for model (9) or for a VARMAX model have been specified, we can use the following function to put this model into state space form, using `NaN` to represent the parameters that have to be estimated.

```
function [str, ferror] = matechelon(kro, s, m)
```

Fixing of parameters to zero The user can fix some parameters to zero in a VARMAX model after the structure `str` has been created using function `matechelon` or any other method. To this end, he can set the corresponding parameters of the appropriate matrix polynomials to zero and subtract the number of fixed parameters from `str.nparm`. For example, in the following lines of MATLAB code first some parameters are fixed to zero after the model has been estimated using the Hannan-Rissanen method. Then, in the next step, the model is re-estimated.

```
strv.gamma(:, :, 1) = 0.; strv.phi(:, :, 2:3) = zeros(1, 2);
strv.theta(:, :, 3) = 0.;
strv.nparm = strv.nparm - 4;
strv = mhanris(yd, xd, seas, strv, 0, 1);
```

3.3. Cointegrated VARMA models

Theoretical introduction

Cointegrated VARMA models can be handled in **SSMMATLAB**. The VARMA models can be ordinary, multiplicative, or in echelon form. The following discussion is valid for all these types of models. Let the k-dimensional VARMA model be given by

$$\Phi(B)Y_t = \Theta(B)A_t, \tag{15}$$

where B is the backshift operator, $BY_t = Y_{t-1}$, $\Phi(z) = \Phi_0 + \Phi_1 z + \cdots + \Phi_l z^l$, $\Theta(z) = \Theta_0 + \Theta_1 z + \cdots + \Theta_l z^l$, $\Theta_0 = \Phi_0$, Φ_0 is lower triangular with ones in the main diagonal, and

$\det[\Phi(z)] = 0$ implies $|z| > 1$ or $z = 1$. We assume that the matrix Π, defined by

$$\Pi = -\Phi(1),$$

has rank r such that $0 < r < k$ and that there are exactly $k - r$ roots in the model equal to one. Then, Π can be expressed (non uniquely) as

$$\Pi = \alpha\beta^\top,$$

where α and β are $k \times r$ of rank r. Let $\beta_\perp$ be a $k \times (k - r)$ matrix of rank $k - r$ such that

$$\beta^\top \beta_\perp = 0_{r\times(k-r)}$$

and define the matrix P as

$$P = [P_1, P_2] = [\beta_\perp, \beta].$$

Then, it is not difficult to verify that $Q = P^{-1}$ is given by

$$Q = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \begin{bmatrix} (\beta_\perp^\top \beta_\perp)^{-1}\beta_\perp^\top \\ (\beta^\top\beta)^{-1}\beta^\top \end{bmatrix}$$

and that if we further define $U_1 = P_1Q_1$ and $U_2 = P_2Q_2$, the following relations hold

$$U_1 + U_2 = I_k, \qquad U_1U_2 = U_2U_1 = 0. \tag{16}$$

Thus, we can write

$$I_k - zI_k = (I_k - U_1z)(I_k - U_2z) = (I_k - U_2z)(I_k - U_1z). \tag{17}$$

The error correction form corresponding to model (15) is

$$\Gamma(B)\nabla Y_t = \Pi Y_{t-1} + \Theta(B)A_t, \tag{18}$$

where $\nabla = I_k - BI_k$, $\Gamma(z) = \Gamma_0 + \sum_{i=1}^{l-1}\Gamma_i z^i$, and the Γ_i matrices are defined by $\Gamma_0 = \Phi_0$ and

$$\Gamma_i = -\sum_{j=i+1}^{l} \Phi_j, \qquad i = 1, \ldots, l - 1.$$

It follows from (18) that $\beta^\top Y_{t-1}$ is stationary because all the terms in this equation different from $\Pi Y_{t-1} = \alpha\beta^\top Y_{t-1}$ are stationary. Therefore, there are r cointegrated relations in the model given by $\beta^\top Y_t$.

Considering (16) and (17), the following relation between the autoregressive polynomials in (15) and (18) holds

$$\begin{aligned} \Phi(z) &= \Gamma(z)(I_k - zI_k) - \Pi z \\ &= [\Gamma(z)(I_k - U_2z) - \Pi z]\,(I_k - U_1z) \end{aligned}$$

because $\Pi U_1 = 0$. Thus, defining $\Phi^*(z) = \Gamma(z)(I_k - U_2z) - \Pi z$ and $D(z) = I_k - U_1z$, we can write $\Phi(z)$ as

$$\Phi(z) = \Phi^*(z)D(z) \tag{19}$$

and the model (15) as

$$\Phi^*(B)D(B)Y_t = \Theta(B)A_t. \tag{20}$$

Since both $U_1 = \beta_\perp(\beta_\perp^\top\beta_\perp)^{-1}\beta_\perp^\top$ and $U_2 = \beta(\beta^\top\beta)^{-1}\beta^\top$ are idempotent and symmetric matrices of rank $k-r$ and r, respectively, the eigenvalues of these two matrices are all equal to one or zero. In particular,

$$\det(I_k - U_1 z) = (1-z)^{k-r}$$

and therefore, the matrix polynomial $D(z) = I_k - U_1 z$ in (19) is a "differencing" matrix polynomial because it contains all the unit roots in the model. This implies in turn that the matrix polynomial $\Phi^*(z)$ in (19) has all its roots outside the unit circle and the series $D(B)Y_t$ in (20) is stationary. Thus, the matrix polynomial $\Phi^*(z)$ can be inverted so that the following relation holds

$$D(B)Y_t = [\Phi^*(B)]^{-1}\,\Theta(B)A_t.$$

Premultiplying the previous expression by $\beta_\perp^\top$, we can see that there are $k-r$ linear combinations of Y_t that are $I(1)$ given by $\beta_\perp^\top Y_t$. In a similar way, premultiplying by $\beta^\top$, it follows as before that there are r linear combinations of Y_t that are $I(0)$ given by $\beta^\top Y_t$.

The series $D(B)Y_t$ can be considered as the "differenced series", and the notable feature of (20) is that the model followed by $D(B)Y_t$ is stationary. Therefore, we can specify and estimate a stationary VARMA model if we know the series $D(B)Y_t$.

It is shown in the **SSMMATLAB** manual how the matrix U_1 can be parameterized.

***SSMMATLAB** implementation*

In **SSMMATLAB** there are two ways to handle cointegrated VARMA models. The first one parameterizes model (18) in terms of the matrix polynomials $\Gamma(z)$ and $\Theta(z)$ and the matrices α and $\beta_\perp$. The second one parameterizes model (20) in terms of the matrix polynomials $\Phi^*(z)$ and $\Theta(z)$ and the matrix $\beta_\perp$. The advantage of the latter parametrization is that we can specify a stationary VARMA model in echelon form for the "differenced" series by directly specifying $\Phi^*(z)$ and $\Theta(z)$. There is no need for a reverse echelon form considered by some authors (Lütkepohl 2007).

Once the cointegration rank has been identified, the matrix $\beta_\perp$ and the differencing matrix polynomial $D(z)$ can be estimated in **SSMMATLAB** using the following function, that also gives the "differenced" series.

```
function [D, DA, yd, ferror] = mdfestim1r(y, x, prt, nr)
```

If a model such as (20) has been identified for the "differenced" series, $D(B)Y_t$, the following function can be used in **SSMMATLAB** to obtain the matrix polynomial $\Gamma(z)$ and the matrices α and $\beta_\perp$ corresponding to the error correction model

```
function [Pi, Lambda, alpha, betap, ferror] = mid2mecf(phi, D, DAf)
```

If a model in error correction form (18) has been identified, the following function can be used in **SSMMATLAB** to obtain the matrix polynomial $\Phi^*(z)$, the matrix $\beta_\perp$ and the differencing matrix polynomial, $D(z)$, corresponding to the model for the "differenced" series.

```
function  [phi, D, DA, ferror] = mecf2mid(Lambda, alpha, betap)
```

Estimating the number of unit roots in the model The number of unit roots in the model can be obtained in **SSMMATLAB** using a generalization to multivariate series of the criterion based on different rates of convergence proposed by Gómez (2013) for univariate series. The following function can be used in **SSMMATLAB** for that purpose.

```
function  [D, nr, yd, DA, ferror] = mcrcregr(y, x)
```

Model estimation Before estimating a model parameterized in terms of the matrix polynomials $\Gamma(z)$ and $\Theta(z)$ and the matrices α and $\beta_{\perp}$ corresponding to the error correction model (18), we have to put the model into state space form. This can be done in **SSMMATLAB** using the following function.

```
function [str, ferror] = suvarmapqPQe(Lambda, alpha, betap, th, Th, ...
  Sigma, freq)
```

Once we have model (18) in state space form, we can estimate it in **SSMMATLAB** using the following function.

```
function  [result, ferror] = varmapqPQestime(y, str, Y, constant)
```

If the model is parameterized in terms of the matrix polynomials $\Phi^{*}(z)$ and $\Theta(z)$ and the matrix $\beta_{\perp}$ corresponding to the model for the "differenced" series (20), we can put the model into state space form in **SSMMATLAB** using first the functions `suvarmapqPQ`, described earlier, and then the following function.

```
function [str, ferror] = aurirvarmapqPQ(str, nr, DA)
```

After having set model (20) in state space form, we can estimate it in **SSMMATLAB** using the following function.

```
function  [result, ferror] = varmapqPQestimd(y, str, Y, constant)
```

In the **SSMMATLAB** manual, it is described how to set up the estimated model and how to forecast with it if desired.

3.4. Univariate structural models

Theoretical introduction

Univariate structural models are models in which the observed univariate process, $\{y_t\}$, is assumed to be the sum of several unobserved components. In its general form, the model is

$$y_t = p_t + s_t + u_t + v_t + e_t,$$

where p_t is the trend, s_t is the seasonal, u_t is the cyclical, v_t is the autoregressive, and e_t is the irregular component. Each of these components follows an ARIMA model. All the models described later in this section can be handled in **SSMMATLAB**.

The trend component is usually specified as

$$\begin{aligned} p_{t+1} &= p_t + b_t + c_t, \\ b_{t+1} &= b_t + d_t, \end{aligned}$$

where $\{c_t\}$ and $\{d_t\}$ are two mutually and serially uncorrelated sequences of random variables with zero mean and variances σ_c^2 and σ_d^2. The idea behind the previous model is to make the slope and the intercept stochastic in a linear equation, $p_t = p + b(t-1)$, and to let them vary according to a random walk. In fact, if $\sigma_c^2 = 0$ and $\sigma_d^2 = 0$ we get the deterministic linear trend $p_t = p_1 + b_1(t-1)$.

There are basically two specifications for the seasonal component. The first one is called "stochastic dummy seasonality" and, according to it, s_t follows the model

$$S(B)s_t = r_t,$$

where $S(B) = 1 + B + \cdots + B^{f-1}$, B is the backshift operator, $By_t = y_{t-1}$, f is the number of observations per year and $\{r_t\}$ is an uncorrelated sequence of random variables with zero mean and variance σ_r^2. The idea behind this model is that the seasonal component is periodic and its sum should be approximately zero in one year. The other representation is called "trigonometric seasonality" and in this case s_t follows the model

$$s_t = \sum_{i=1}^{[f/2]} s_{i,t},$$

where $[x]$ denotes the greatest integer less than or equal x, f is, as before, the number of observations per year, s_{it} follows the model

$$\begin{bmatrix} s_{i,t+1} \\ s^*_{i,t+1} \end{bmatrix} = \begin{bmatrix} \cos\omega_i & \sin\omega_i \\ -\sin\omega_i & \cos\omega_i \end{bmatrix} \begin{bmatrix} s_{i,t} \\ s^*_{i,t} \end{bmatrix} + \begin{bmatrix} j_{i,t} \\ j^*_{i,t} \end{bmatrix}, \tag{21}$$

$\omega_i = 2\pi i/f$, and $\{j_{i,t}\}$ and $\{j^*_{i,t}\}$ are two mutually and serially uncorrelated sequences of random variables with zero mean and common variance σ_i^2. If f is even, $\omega_{f/2} = 2\pi[f/2]/f = \pi$ and the model followed by the component $s_{f/2,t}$ in (21), corresponding to the frequency $\omega_{f/2}$, collapses to $s_{f/2,t+1} = -s_{f/2,t} + j_{f/2,t}$. In **SSMMATLAB**, it is assumed that all seasonal components have a common variance, $\sigma_i^2 = \sigma_s^2$, $i = 1, 2, \ldots, [f/2]$. This representation of the seasonal component has its origin in the observation that, from the theory of difference equations, we know that the solution of the equation $S(B)s_t = 0$ is the sum of $[f/2]$ deterministic harmonics, each one corresponding to a seasonal frequency ω_i.

If the cyclical component, u_t, is present, it can be modeled in two different ways. The first one corresponds to that proposed by Harvey (1993), namely

$$\begin{bmatrix} u_{t+1} \\ u^*_{t+1} \end{bmatrix} = \rho \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u_t \\ u^*_t \end{bmatrix} + \begin{bmatrix} k_t \\ k^*_t \end{bmatrix}, \tag{22}$$

where $0 < \rho < 1$, $\theta \in [0, \pi]$ is the cyclical frequency, and $\{k_t\}$ and $\{k^*_t\}$ are two mutually and serially uncorrelated sequences of random variables with zero mean and common variance σ_k^2. The ρ factor ensures that the cycle is stationary.

It can be shown that the initial conditions for the cycle (22) satisfy

$$\left[\begin{array}{c} u_1 \\ u_1^* \end{array} \right] \sim \left[\left(\begin{array}{c} 0 \\ 0 \end{array} \right), \sigma_u^2 I_2 \right],$$

where I_2 is the unit matrix of order two and $(1-\rho^2)\sigma_u^2 = \sigma_k^2$. Here, the notation $\sim$ refers to the first two moments of the distribution of u_1 and u_1^*, meaning that these two variables have zero mean, are uncorrelated and have common variance σ_u^2.

The second way to model the cycle has its origin in the model-based interpretation of a band-pass filter derived from a Butterworth filter based on the sine function. See Gómez (2001) for details. The model for the cycle is in this case

$$(1 - 2\rho\cos\theta B + \rho B^2)u_t = (1 - \rho\cos\theta B)k_t, \tag{23}$$

where ρ and θ are as described earlier and $\{k_t\}$ is an uncorrelated sequence of random variables with zero mean and variance σ_k^2. The previous model can be put into state space form as

$$\left[\begin{array}{c} u_{t+1} \\ u_{t+1|t} \end{array} \right] = \left[\begin{array}{cc} 0 & 1 \\ -\rho & 2\rho\cos\theta \end{array} \right] \left[\begin{array}{c} u_t \\ u_{t|t-1} \end{array} \right] + \left[\begin{array}{c} 1 \\ \rho\cos\theta \end{array} \right] k_t. \tag{24}$$

To obtain the initial conditions in this case, we can consider that, clearly, u_1 and $u_{1|0}$ have zero mean and their covariance matrix, V, satisfies the Lyapunov equation

$$V = AVA^\top + bb^\top \sigma_k^2,$$

where

$$A = \left[\begin{array}{cc} 0 & 1 \\ -\rho & 2\rho\cos\theta \end{array} \right], \qquad b = \left[\begin{array}{c} 1 \\ \rho\cos\theta \end{array} \right].$$

The matrix V is obtained in **SSMMATLAB** by solving the previous Lyapunov equation in a numerically safe manner.

It is to be noticed that the variables $s_{i,t}^*$, $u_{i,t}^*$ and $u_{t|t-1}$ in (21), (22) and (23) are auxiliary variables used to define the state space forms for the components of interest. In addition, it can be shown that the cycle specified as in (23) can be obtained from (22) if we let $\{k_t^*\}$ be deterministic and equal to zero while maintaining $\{k_t\}$ stochastic and without change.

The autoregressive component, v_t, is assumed to follow an autoregressive model, i.e.,

$$(1 + \phi_1 B + \cdots + \phi_p B^p)v_t = w_t,$$

where the polynomial $\phi(B) = 1 + \phi_1 B + \cdots + \phi_p B^p$ has all its roots outside the unit circle and $\{w_t\}$ is an uncorrelated sequence of random variables with zero mean and variance σ_w^2.

In **SSMMATLAB** only one cycle at a time can be specified in the structural model. The reason for this is that cycles are usually difficult to specify and to estimate. Thus, if one believes that there are several cycles in the model, one can specify one cycle and let the autoregressive component take account of the other cycles by specifying a sufficiently high autoregressive order.

There exists the possibility to incorporate regression variables into structural models. More specifically, models of the form

$$y_t = Y_t\beta + w_t,$$

where w_t follows a structural model and β is a vector of regression coefficients, can be handled in **SSMMATLAB**. It is also possible to incorporate interventions that affect some component. For example, an impulse to accommodate a sudden change in the slope of the series that takes place at one observation only. This type of intervention can be modeled by defining a proper W_t matrix in Equation 1. This procedure will be illustrated in Case Study 3.

SSMMATLAB *implementation*

The following function can be used in **SSMMATLAB** to put a univariate structural model into state space form.

```
function [str, ferror] = suusm(comp, y, Y, npr)
```

Fixing of parameters If the user wants to fix some parameters in a structural model, he should set the corresponding elements in the arrays `comp.level`, `comp.slope`, `comp.seas`, `comp.cycle`, `comp.cyclep`, `comp.arp` or `comp.irreg` to zero instead of `NaN`.

Model estimation Once the model has been defined and, therefore, the structure `str` exists, it can be estimated. Before estimation, the user has to decide whether to fix some parameters in the model or not. How to fix some parameters has been explained in the previous paragraph. The parameters to be estimated are in the array `str.xv`, and the fixed parameters are in `str.xf`. It is assumed that the values entered by the user for the parameters to be estimated are reasonable initial values.

Initial values that can be used for the standard deviations are all equal to .1, except the slope standard deviation that is usually smaller and can be set to 0.005. Initial values that can be used for the autoregressive parameters are all equal to .1. Initial values for the cycle ρ and frequency parameters can be .9 and a frequency that can be considered reasonable by the user.

If the user has not selected a variance to be concentrated out using the field `comp.conout`, the program will select the biggest variance to that effect. After calling function `suusm`, the index for the parameter to be concentrated out is in the field `str.conc`.

The following function can be used for parameter estimation.

```
function  [result, str] = usmestim(y, str)
```

It is to be noticed that, even if the user has selected a variance to be concentrated out using the field `comp.conout`, the program will always check whether the selected variance is the biggest one. To this end, a preliminary estimation is performed in `usmestim`. After it, if the biggest estimated variance does not correspond to the initially selected parameter to be concentrated out, the program will change this parameter and will make the necessary adjustments in structure `str`. Therefore, structure `str` can change after calling `usmestim`. The actual estimation is performed after the previous check.

After model estimation, function `pr2usm` can be used to set up the estimated structural model. For example, the following commands achieve this.

```
xvf = result.xvf; xf = result.xf;
[X, Z, G, W, T, H, ins, ii, ferror] = pr2usm(xvf, xf, str);
```

Recursive residuals As explained in Section 5.2, recursive residuals can be computed using functions `scakff` or `scakfff`. For example, the following command can be used to compute recursive residuals after estimation of a structural model, assuming that `X`, `Z`, `G`, `W`, `T` and `H` are the estimated matrices and `ins` and `ii` contain the initial conditions.

```
[Xt, Pt, g, M, initf, recrs] = scakff(y, X, Z, G, W, T, H, ins, ii);
```

Forecasting As described in Section 5.4, forecasts can be obtained using function `ssmpred`. For example, the following commands can be used to obtain ten forecasts after estimating a structural model, assuming that `X`, `Z`, `G`, `W`, `T`, `H`, `ins` and `ii` are as in the previous paragraph. Note that the regression matrices, `X` and `W`, if they are time-varying, should have been extended to account for the forecast horizon. The variables `hb`, `Mb`, `A` and `P` are needed and they are in structure `result`. `hb` is the vector of regression estimates and `Mb` is the matrix of standard errors. `A` is the possibly augmented estimated state vector, $x_{t|t-1}$, obtained with the Kalman filter at the end of the sample and `P` is the matrix of standard errors.

```
hb = result.h; Mb = result.M; A = result.A; P = result.P;
npr = 10;
if ~isempty(X)
 Xp = X(end - npr + 1:end, :);
end
if ~isempty(W)
 Wp = W(end - npr + 1:end, :);
end
m = 1;
[pry, mypr, alpr, malpr] = ssmpred(npr, m, A, P, Xp, Z, G, Wp, T, H, hb, Mb);
```

Smoothing As described in Section 5.5, smoothing can be performed using function `scakfs`. For example, assuming that ten forecasts have been previously obtained after estimating a structural model and that the series forecasts are in array `pry` and the state vector forecasts are in array `alpr`, the following commands can be used to estimate the trend using smoothing, extend it with the forecasts, and display both the extended original and trend series. It is further assumed that `X`, `Z`, `G`, `W`, `T` and `H` are the estimated matrices and `ins` and `ii` contain the initial conditions.

```
npr = 10;
X = str.X; W = str.W;
if ~isempty(X)
 X = X(1:end - npr, :);
end
if ~isempty(W)
 W = W(1:end - npr, :);
end
[Xt, Pt, g, M] = scakfs(y, X, Z, G, W, T, H, ins, ii);

% example with constant slope
trend = Xt(:, 1) + X * g(end);
```

```
% forecast of trend.
% Xp is the regression matrix corresponding to the forecast horizon
trendp = alpr(1, :)' + Xp * g(end);
t = 1:ny + npr; plot(t, [y; pry'], t, [trend; trendp])
pause
closefig
```

The following function can be used after smoothing to select a desired smoothed component. This function works with structural as well as with AMB unobserved components models. These last models will be introduced in Section 3.5.

```
function Cc = dispcomp(KKP, str, comp, varargin)
```

For example, the following lines can be used to select the smoothed cycle and to plot it after the smoothed components have been obtained using function `scakfs`.

```
[KKP, PT, a, b] = scakfs(y, X, Z, G, W, T, H, ins, ii);
Cc = dispcomp(KKP, str, 'cycle', datei, 'PR Smoothed Cycle');
cyc = Cc(:, 1);
```

3.5. AMB unobserved components models

Theoretical introduction

The ARIMA model-based (AMB) method to decompose a given time series that follows an ARIMA model into several unobserved components that also follow ARIMA models is described in, for example, Gómez and Maravall (2001b, Chapter 8).

This approach was originally proposed by Hillmer and Tiao (1982). The idea is based on a partial fraction expansion of the pseudospectrum of an ARIMA model specified for the series at hand, $\{y_t\}$. According to this decomposition, terms with denominators originating peaks at the low frequencies should be assigned to the trend component, terms with denominators originating peaks at the seasonal frequencies should be assigned to the seasonal component, and the other terms should be grouped into a so-called "stationary component". This last component can in turn be decomposed into an irregular (white noise) plus some other, usually moving average, component. For example, consider the model

$$\nabla\nabla_4 y_t = a_t,$$

where $\nabla = 1 - B$, B is the backshift operator, $By_t = y_{t-1}$, and $\{a_t\}$ is a white noise sequence with zero mean and $\mathsf{VAR}(a_t) = \sigma^2$. Given that $(1-z)(1-z^4) = (1-z)^2(1+z+z^2+z^3)$, the pseudoespectrum is

$$\begin{aligned} f(x) &= \frac{\sigma^2}{2\pi}\frac{1}{|1-e^{-ix}|^4|1+e^{-ix}+e^{-2ix}+e^{-3ix}|^2} \\ &= \frac{A(x)}{|1-e^{-ix}|^4} + \frac{B(x)}{|1+e^{-ix}+e^{-2ix}+e^{-3ix}|^2}, \end{aligned}$$

where $A(x)$ and $B(x)$ are polynomial functions of $\cos(x)$ to be determined. To see this, consider that, setting $y = e^{-ix} + e^{ix}$ as the new variable, any pseudospectrum can be written as a quotient of polynomials in $y = 2\cos(x)$.

In the previous decomposition of $f(x)$, the first term on the right hand side becomes infinite at the zero frequency and should be assigned to the trend, whereas the second term becomes infinite at the seasonal frequencies, π and $\pi/2$, and should, therefore, be assigned to the seasonal component. However, both the seasonal and the trend components are not identified because it is possible that one may subtract some positive quantity from each of the terms on the right hand side and at the same time add it as a new term in the decomposition of $f(x)$, so that we would obtain

$$f(x) = \frac{\widetilde{A}(x)}{|1 - e^{-ix}|^4} + \frac{\widetilde{B}(x)}{|1 + e^{-ix} + e^{-2ix} + e^{-3ix}|^2} + k,$$

where $\widetilde{A}(x)$ and $\widetilde{B}(x)$ are new polynomial functions in $\cos(x)$ and k is a positive constant. This positive constant gives rise to a new white noise component.

To identify the components, the so-called `canonical decomposition` is performed. According to this decomposition, a positive constant, as big as possible, is subtracted from each term on the right hand side. In this way, the components are made as smooth as possible and become identified. The resulting components are called `canonical components`. The canonical decomposition does not always exist and this constitutes a flaw in the procedure. However, there are simple solutions to this problem.

It can be shown that the trend and seasonal components, p_t and s_t, corresponding to the previous example are of the form

$$\nabla^2 p_t = (1 + \alpha B)(1 + B)b_t,$$

and

$$(1 + B + B^2 + B^3)s_t = (1 + \beta_1 B + \beta_2 B^2 + \beta_3 B^3)c_t,$$

where $\{b_t\}$ and $\{c_t\}$ are two uncorrelated white noises and the polynomial $1+\beta_1 z+\beta_2 z^2+\beta_3 z^3$ has at least one root in the unit circle. In addition, the equality $y_t = p_t + c_t + i_t$ holds, where $\{i_t\}$ is white noise.

If logs of the series, y_t, are taken, then the procedure is applied to the transformed series. Thus, in order to obtain the multiplicative components one has to exponentiate the components obtained from the decomposition of $\log(y_t)$. This may cause problems with the estimated trend because usually the annual trend sums are lower than the annual sums of the original series, a phenomenon due to geometric means being smaller than arithmetic means. For this reason, some kind of "bias" correction is usually applied to the estimated trend. This problem is also present in structural models.

SSMMATLAB *implementation*

Before performing the canonical decomposition, it is necessary to select the roots in the autoregressive polynomial that should be assigned to the trend and the seasonal components. The following function can be used in **SSMMATLAB** for that purpose.

```
function [phir, phis, thr, ths, phirst] = arima2rspol(phi, Phi, th, Th, ...
  freq, dr, ds)
```

Once the model has been decomposed into its canonical components, one can put the unobserved components model into state space form and perform forecasting and smoothing in the same way as that previously described for structural models.

To put the model into state space form, the following **SSMMATLAB** function can be used.

```
function [X, Z, G, W, T, H, ins, ii, strc, ferror] = sucdm(comp, y, Y, ...
  stra, npr)
```

The following lines of MATLAB code illustrate how to first decompose an ARIMA model into its canonical components and then how to smooth these components. Finally, the original series as well as the trend-cycle are displayed. The model is given by the polynomials `phi`, `Phi`, `th` and `Th`. The regular and seasonal differences are one and one, respectively. The standard deviation of the residuals is `sconp`. The number of observations per year is `freq`.

```
s = freq; dr = 1; ds = 1; sconp = .5;
Sigma = sconp^2;
[str, ferror] = suvarmapqPQ(phi, th, Phi, Th, Sigma, s);

[phir, phis, thr, ths, phirst] = arima2rspol(phi, Phi, th, Th, s, dr, ds);

[compcd, ierrcandec] = candec(phir, phis, thr, ths, phirst, s, dr, ds, ...
  sconp);

npr = 0; Y = [];
[X, Z, G, W, T, H, ins, ii, strc, ferror] = sucdm(compcd, y, Y, str, npr);

[KKP, PT, a, b] = scakfs(y, X, Z, G, W, T, H, ins, ii);

Cc = dispcomp(KKP, strc, 'trendcycle');
trend = Cc(:, 1);
vnames = strvcat('PR', 'PR trend');
figure
tsplot([y trend], datei, vnames);
pause
```

Estimation of smooth trends and cycles

In the AMB approach, it is not usually possible to directly estimate cycles. This is due to the fact that the majority of the ARIMA models fitted in practice do not have autoregressive components with complex roots that may give rise to cyclical components. Trend components given by the AMB approach are for this reason also called "trend-cycle" components. For similar reasons, it is also usually not possible to estimate smooth trends using only the unobserved components given by the canonical decomposition.

To estimate smooth trends and cycles within the AMB approach, one possibility is to incorporate fixed filters into the approach in the manner proposed by Gómez (2001).

The filters considered in **SSMMATLAB** for smoothing trends are two-sided versions of Butterworth filters. Butterworth filters are low-pass filters and they are of two types. The first one is based on the sine function (BFS), whereas the second one is based on the tangent function (BFT). See, for example, Otnes and Enochson (1978).

The squared gain of a BFS is given by

$$|G(x)|^2 = \frac{1}{1+\left(\frac{\sin(x/2)}{\sin(x_c/2)}\right)^{2d}}, \tag{25}$$

where x denotes angular frequency and x_c is such that $|G(x_c)|^2 = 1/2$. These filters depend on two parameters, d and x_c. If x_c is fixed, the effect of increasing d is to make the fall of the squared gain sharper. BFSs are autoregressive filters of the form $H(B) = 1/\theta(B)$, where B is the backshift operator, $By_t = y_{t-1}$, $\theta(B) = \theta_0 + \theta_1 B + \cdots + \theta_d B^d$ and $|G(x)|^2 = H(e^{-ix})H(e^{ix})$. Thus, if $\{y_t\}$ is the input series, the output series, $\{z_t\}$, is given by the recursion

$$\theta_0 z_t + \theta_1 z_{t-1} + \cdots + \theta_d z_{t-d} = y_t.$$

To start the recursion at $t = 1$ say, some initial values, $z_{1-d}, \ldots, z_0$, are needed.

The BFSs used in **SSMMATLAB** are of the form $H_s(B,F) = H(B)H(F) = 1/[\theta(B)\theta(F)]$, where F is the forward operator, $Fy_t = y_{t+1}$ and $H(B) = 1/\theta(B)$ is a BFS.

It can be shown that $H_s(B,F)$ can be given a model-based interpretation. It is the Wiener-Kolmogorov filter to estimate the signal in the signal plus noise model

$$y_t = s_t + n_t, \tag{26}$$

under the assumption that the signal s_t follows the model $\nabla^d s_t = b_t$, where $\{b_t\}$ is a white noise sequence with zero mean and unit variance and $\{b_t\}$ is independent of the white noise sequence $\{n_t\}$. The estimator of s_t is given by

$$\hat{s}_t = H_s(B,F)z_t = \nu_0 y_t + \sum_{k=1}^{\infty} \nu_k (B^k + F^k) y_t. \tag{27}$$

The weights ν_k in (27) can be obtained from the signal extraction formula

$$H_s(B,F) = 1/[1 + \lambda(1-B)^d(1-F)^d], \tag{28}$$

where $\lambda = \mathsf{VAR}(n_t)$. The frequency response function, $\hat{H}_s(x)$, of the filter $H_s(B,F)$ is obtained from (28) by replacing B and F with e^{-ix} and e^{ix}, respectively. After some manipulation, it is obtained that

$$\hat{H}_s(x) = \frac{1}{1+\left(\frac{\sin(x/2)}{\sin(x_c/2)}\right)^{2d}}, \tag{29}$$

where $\lambda = [2\sin(x_c/2)]^{-2d}$. Thus, the gain, $|\hat{H}_s(x)|$, of $H_s(B,F)$ coincides with the squared gain of a BFS. See Gómez (2001) for details.

For BFT, the squared gain function is given by (25), but replacing the sine function by the tangent function. The filter is of the form $H(B) = (1+B)^d/\theta(B)$, where $\theta(B) = \theta_0 + \theta_1 B + \cdots + \theta_d B^d$ and $|G(x)|^2 = H(e^{-ix})H(e^{ix})$.

To design a BFS in **SSMMATLAB**, the following function can be used.

```
function [compf, ferror] = dsinbut(D, Thetap, Thetas, Di, Thetac, Lambda)
```

The following function can be applied in **SSMMATLAB** to design a BFT.

```
function [compf, ferror] = dtanbut(D, Thetap, Thetas, Di, Thetac, Lambda)
```

To plot the gain function of a BFS or BFT, the following function can be used in **SSMMATLAB**.

```
function ggsintanbut(D, Thetap, Thetas, d, thc)
```

The following lines of MATLAB code can be used to first specify a BFS that coincides with the Hodrick-Prescott filter and then to plot the gain function of the filter. The filter is specified giving the parameters `Lambda` and `Di`.

```
Lambda = 1600; Di = 2;
[compbst, ferror] = dsinbut([], [], [], Di, [], Lambda);
figure
ggsintanbut([], [], [], compbst.Di, compbst.Thetac)
pause
```

To estimate cycles in **SSMMATLAB**, one can use band-pass filters derived from BFT. These are two-sided filters that can be obtained by estimating signals which follow the model $(1 - 2\cos xB + B^2)^d s_t = (1 - B^2)^d b_t$ in the signal plus noise model (26). Details regarding the design of these band-pass filters and their model-based interpretation can be found in Gómez (2001).

To design a band-pass filter in **SSMMATLAB**, the following function can be used.

```
function [compf, ferror] = dbptanbut(D, Omegap1, Omegap2, Omegas2, ...
  Di, Thetac, Lambda)
```

To plot the gain function of a band-pass filter in **SSMMATLAB**, the following function can be used.

```
function ggbptanbut(D, omp1, omp2, oms2, d, alph, lambda)
```

The following MATLAB code lines illustrate how to first design a band-pass filter to be applied to quarterly data to estimate a cycle with frequencies in the business cycle frequency band (periods between a year and a half and eight years). Then, it is shown how the gain of the designed filter can be plotted. Frequencies are expressed divided by π.

```
D(1) = .1; D(2) = .1; xp1 = .0625; xp2 = .3; xs = .4;
[compbp, ferror] = dbptanbut(D, xp1, xp2, xs);
figure
ggbptanbut(D, xp1, xp2, xs, compbp.Di, compbp.Alph, compbp.Lambda)
pause
closefig
```

All of the previously described filters are fixed filters. However, they can be incorporated into the AMB approach as described in Gómez (2001). See the **SSMMATLAB** manual in Gómez (2014) for more details.

The following function can be used in **SSMMATLAB** to set up a state space model for an unobserved components model, where the components are obtained in the manner previously described given information from both the canonical decomposition of an ARIMA model and a designed BFS or BFT.

```
function [X, Z, G, W, T, H, ins, ii, strc, ferror] = sucdmpbst(comp, ...
  compf, y, Y, stra, npr)
```

If instead of a low-pass filter (BFS or BFT), as in the previous function, a band-pass filter based on BFT is applied, the following function can be used to set up the appropriate state space model.

```
function [X, Z, G, W, T, H, ins, ii, strc, ferror] = sucdmpbp(comp, ...
  compf, y, Y, stra, npr)
```

Once we have a state space model in which the trend-cycle given by the AMB approach has been further decomposed into a smooth trend and a cycle by means of a fixed filter of the type BFS, BFT or band-pass filter based on BFT, we can use the Kalman filter to smooth the components. This can be done in **SSMMATLAB** by using the function `scakfs`. For example, the following lines of MATLAB code can be used to first design a low-pass filter of the BFS type, the Hodrick-Prescott filter, and then to estimate the unobserved components. Finally, the smooth trend and the cycle, estimated as the difference between the trend-cycle and the smooth trend, are plotted.

```
Lambda = 1600; Di = 2;
[compbst, ferror] = dsinbut([], [], [], Di, [], Lambda);

[X, Z, G, W, T, H, ins, ii, strc, ferror] = sucdmpbst(compcd, compbst, ...
  y, Y, str, npr);

[KKP, PT, a, b] = scakfs(y, X, Z, G, W, T, H, ins, ii);

Cc = dispcomp(KKP, strc, {'trend', 'cycle'}, datei, 2, 'PR bpcycle');
bptrend = Cc(:, 1);
bpcycle = Cc(:, 2);
```

4. Identification and estimation of VAR(MA)X models

In this section, we will describe a number of tools available in **SSMMATLAB** for the identification and estimation of VARX and VARMAX models.

4.1. VARX identification and estimation

The VARX models considered in **SSMMATLAB** are of the form

$$Y_t = \sum_{j=1}^{p} \Pi_j Y_{t-j} + \sum_{j=0}^{p} \Gamma_j Z_{t-j} + A_t. \tag{30}$$

These models are important because every VARMAX model can be approximated to any degree of accuracy by a VARX model with a sufficiently high order.

Although a VARX model can be put into state space form, VARX models are estimated in **SSMMATLAB** using OLS. The reason why these models are included in **SSMMATLAB** is that they usually constitute a good starting point when analyzing multivariate models, like VARMAX models. To estimate a VARX model in **SSMMATLAB**, the following function can be used.

```
function res = varx_est(y, nlag, x, test, xx)
```

When estimating a VARX model, sometimes only the residuals are desired. In this case, the following function can be used in **SSMMATLAB**.

```
function resid = varx_res(y, nlag, x)
```

To identify the order of a possibly nonstationary VARX model, the likelihood ratio criterion can be used. The following function can be applied in **SSMMATLAB** for this purpose.

```
function [lagsopt, initres] = lratiocrx(y, maxlag, minlag, prt, x)
```

When there are no exogenous variables, that is, when the model is a VAR model, the following function can be used in **SSMMATLAB** for model estimation.

```
function res = var_est(y, nlag, test, x)
```

If only the residuals are desired when estimating a VAR model, the following function can be used in **SSMMATLAB**.

```
function   resid = var_res(y, nlag, x)
```

To determine the optimal lag length of a possibly nonstationary VAR model using the likelihood ratio criterion, the following function can be used in **SSMMATLAB**.

```
function [lagsopt, initres] = lratiocr(y, maxlag, minlag, prt, x)
```

In the case of a possibly nonstationary VAR model, the following function can be used in **SSMMATLAB** to determine the optimal lag length using the AIC or BIC criterion.

```
function lagsopt = infcr(y, maxlag, minlag, crt, prt, x)
```

4.2. Multivariate residual diagnostics

To estimate the covariances and the autocorrelations, as well as the portmanteau statistics of a multivariate time series, the following function can be used in **SSMMATLAB**.

```
function str = mautcov(y, lag, ic, nr)
```

4.3. Identification of VARMAX(p, q, r) models

The following function can be used in **SSMMATLAB** to identify VARMAX(p, q, r) models. It applies a sequence of likelihood ratio tests to obtain the orders.

```
function [lagsopt, ferror] = lratiopqr(y, x, seas, maxlag, minlag, prt)
```

4.4. Identification of VARMAX models in echelon form

The following two functions can be used in **SSMMATLAB** to identify the Kronecker indices for VARMAX models in echelon form. The first one identifies and estimates in a preliminary step a VARMAX(p, q, r) model, whereas the second one starts by identifying and estimating a VARMAX(p, p, p) model. Both functions use a sequence of likelihood ratio tests on each equation to determine the Kronecker indices.

```
function [order, kro, scm] = varmaxscmidn(y, x, seas, maxorder, hr3, prt)
function [order, kro] = varmaxkroidn(y, x, seas, maxorder, hr3, ct, prt)
```

4.5. The Hannan-Rissanen method to estimate VARMAX models

Although state space models can be directly estimated using regression techniques, like subspace methods, these methods involve the estimation of a large number of parameters as soon as the dimension of the state vector increases. For this reason, the approach adopted in **SSMMATLAB** is to use the Hannan-Rissanen method, that applies regression techniques only and is based on the VARMAX specification of the model. Even though it does not use state space models, it usually gives very good starting values when estimating a VARMAX model in state space echelon form by maximum likelihood.

Theoretical introduction

Suppose that the process $\{Y_t\}$ follows the VARMAX model in echelon form

$$\Phi_0 Y_t + \cdots + \Phi_r Y_{t-r} = \Omega_0 Z_t + \cdots + \Omega_r Z_{t-r} + \Theta_0 A_t + \cdots + \Theta_r A_{t-r}, \tag{31}$$

where $\Phi_0 = \Theta_0$ is a lower triangular matrix with ones in the main diagonal. Equation 31 can be rewritten as

$$Y_t = (I_k - \Phi_0)V_t - \sum_{j=1}^{r} \Phi_j Y_{t-j} + \sum_{j=0}^{r} \Omega_j Z_{t-j} + \sum_{j=1}^{r} \Theta_j A_{t-j} + A_t, \tag{32}$$

where $V_t = Y_t - A_t$ and A_t in (32) is uncorrelated with Z_s, $s \leq t$, Y_u, A_u, $u \leq t-1$, and

$$V_t = \Phi_0^{-1}\left(-\sum_{j=1}^{r} \Phi_j Y_{t-j} + \sum_{j=0}^{r} \Omega_j Z_{t-j} + \sum_{j=1}^{r} \Theta_j A_{t-j}\right).$$

Applying the vec operator to (32), it is obtained that

$$\begin{aligned} Y_t &= -\sum_{j=1}^{r}(Y_{t-j}^\top \otimes I_k)\mathrm{vec}(\Phi_j) + \sum_{j=0}^{r}(Z_{t-j}^\top \otimes I_k)\mathrm{vec}(\Omega_j) - (V_t^\top \otimes I_k)\mathrm{vec}(\Theta_0 - I_k) \\ &\quad + \sum_{j=1}^{r}(A_{t-j}^\top \otimes I_k)\mathrm{vec}(\Theta_j) + A_t \\ &= [W_{1,t}, W_{2,t}, W_{3,t}]\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} + A_t \\ &= W_t\alpha + A_t, \end{aligned} \tag{33}$$

where $W_{1,t} = [-Y_{t-1}^\top \otimes I_k, \ldots, -Y_{t-r}^\top \otimes I_k]$, $W_{2,t} = [Z_t^\top \otimes I_k, \ldots, Z_{t-r}^\top \otimes I_k]$, $W_{3,t} = [-V_t^\top \otimes I_k, A_{t-1}^\top \otimes I_k, \ldots, A_{t-r}^\top \otimes I_k]$, $\alpha_1 = [\mathrm{vec}^\top(\Phi_1), \ldots, \mathrm{vec}^\top(\Phi_r)]^\top$, $\alpha_2 = [\mathrm{vec}^\top(\Omega_0), \ldots, \mathrm{vec}^\top(\Omega_r)]^\top$, $\alpha_3 = [\mathrm{vec}^\top(\Theta_0 - I_k), \mathrm{vec}^\top(\Theta_1), \ldots, \mathrm{vec}^\top(\Theta_r)]^\top$, $W_t = [W_{1,t}, W_{2,t}, W_{3,t}]$ and $\alpha = [\alpha_1^\top, \alpha_2^\top, \alpha_3^\top]^\top$.

The parameter restrictions given by the echelon form (31) can be incorporated into Equation 33 by defining a selection matrix, R, containing zeros and ones such that

$$\alpha = R\beta, \tag{34}$$

where β is the vector of parameters that are not restricted in the matrices Φ_i, Ω_i or Θ_i, $i = 0, 1, \ldots, r$. Using (34), Equation 33 can be rewritten as

$$\begin{aligned} Y_t &= W_t R\beta + A_t \\ &= X_t\beta + A_t, \end{aligned} \tag{35}$$

where $X_t = W_t R$. Notice that, as mentioned earlier, X_t is uncorrelated with A_t in (35) and that if we knew X_t, we could estimate β by OLS. The idea behind the Hannan-Rissanen method is to estimate β in (35) after we have replaced the unknown innovations in X_t with those estimated using a VARX model.

The Hannan-Rissanen method is described in more detail in the **SSMMATLAB** manual.

SSMMATLAB *implementation*

As mentioned earlier, the Hannan-Rissanen method usually gives very good starting values when estimating a VARMAX model in state space echelon form by maximum likelihood. The following function can be used in **SSMMATLAB** to estimate a VARMAX model using this method.

```
function [str, ferror] = estvarmaxkro(y, x, seas, kro, hr3, finv2, ...
  mstainv, nsig, tsig)
```

For example, the following lines of MATLAB code can be used to estimate a transfer function model with one input series. Both, the input and the output series, are differenced prior to estimation. All polynomials have degree two.

```
yd = diferm(y, 1); xd = diferm(x, 1);
kro = 2; hr3 = 0; finv2 = 1;
strv = estvarmaxkro(yd, xd, seas, kro, hr3, finv2);
```

As described earlier, if there are parameters that are not significant after estimation, it is possible to fix them to zero and estimate the model again. The following lines can be used in the previous example to fix some parameters to zero and re-estimate the model. The estimation is performed using function `mhanris`, that will be described later in this section.

```
strv.gamma(:, :, 1) = 0.; strv.phi(:, :, 2:3) = zeros(1, 2);
strv.theta(:, :, 3) = 0.;
strv.nparm = strv.nparm - 4;
strv = mhanris(yd, xd, seas, strv, 0, 1);
```

Note how the number of parameters to estimate, contained in the field `nparm` of the structure `strv`, is decreased according to the number of parameters fixed.

To re-estimate a VARMAX model in **SSMMATLAB** after having fixed some parameters to zero, the following function can be used.

```
function str = mhanris(y, x, seas, str, hr3, finv2, mstainv, nsig, tsig)
```

Sometimes a VARMAX model is given as a multiplicative VARMA model with exogenous inputs. In this case, the following function can be used in **SSMMATLAB** to estimate the models in this form.

```
function [str, ferror] = estvarmaxpqrPQR(y, x, seas, ordersr, orderss, ...
  hr3, finv2, mstainv, nsig, tsig)
```

4.6. The conditional method to estimate VARMAX models

When a VARMAX model has been estimated using the Hannan-Rissanen method, sometimes it is convenient to iterate in the third stage to obtain better parameter estimates. This constitutes the so-called conditional method. See, for example, Reinsel (1997) or Lütkepohl (2007).

The following function can be used in **SSMMATLAB** to estimate a VARMAX model using the conditional method.

```
function  [xvf, str, ferror] = mconestim(y, x, str)
```

Conditional residuals After estimating a VARMAX model using the conditional method, the conditional residuals are in the field `residcon` of the structure `str` given as output by function `mconestim`. For example, the following lines of MATLAB code can be used to plot the conditional residuals of a bivariate model and their simple and partial correlograms after estimation.

```
[xvf, strc, ferror] = mconestim(yd, xd, strv);
s = 2;
freq = 1;
lag = 16; cw = 1.96;
rlist = {'resid1', 'resid2'};
```

```
dr = 0; ds = 0;
for i = 1:s
   c0 = sacspacdif(strc.residcon(:, i), rlist{i}, dr, ds, freq, lag, cw);
   pause
end
closefig
```

Forecasting The procedure to obtain some forecasts after estimating a VARMAX model using the conditional or the exact method will be described at the end of the next section.

4.7. The exact ML method to estimate VARMAX models

After a VARMAX model has been estimated using the Hannan-Rissanen or the conditional method, the user may be interested in estimating the model using the exact maximum likelihood (ML) method.

The following function can be used in **SSMMATLAB** to estimate a VARMAX model using the exact ML method.

```
function  [xvf, str, ferror] = mexactestimc(y, x, str, Y)
```

Recursive residuals After estimating a VARMAX model using the exact ML method, the following function can be used to obtain the recursive residuals.

```
function [ff, beta, e, f, str, stx, recrs] = exactmedfvc(beta, y, x, ...
  str, Y, chb)
```

For example, the following lines of MATLAB code illustrate the estimation of a bivariate VARMAX model using the exact ML method. Then, some diagnostic statistics based on the recursive residuals are computed. Finally, the recursive residuals and their simple and partial correlograms are plotted.

```
s = 2;
Y = eye(s);
[xvfx, strx, ferror] = mexactestimc(yd, xd, strc, Y);

chb = 2;
[ff, beta, e, f, str, stx, recrs] = exactmedfvc(xvfx, yd, xd, strx, Y, chb);

lag = 12; ic = 1; nr = strv.nparm-s;
str = mautcov(recrs, lag, ic, nr);
disp('sample autocorrelations signs:')
disp(str.sgnt)
pause

disp('p-values of Q statistics:')
disp(str.pval)
pause
```

```
lag = 16; cw = 1.96;
rlist = {'resid1', 'resid2'};
dr = 0; ds = 0;
for i = 1:s
   c0 = sacspacdif(recrs(:, i), rlist{i}, dr, ds, freq, lag, cw);
   pause
end
closefig
```

Forecasting To obtain some forecasts after estimating a VARMAX model using the conditional or the exact method, the observed series, y_t, that is assumed to follow the state space model in echelon form

$$\begin{aligned} \alpha_{t+1} &= F\alpha_t + Bx_t + Ka_t \\ y_t &= Y_t\beta + H\alpha_t + Dx_t + a_t, \end{aligned}$$

is first expressed as

$$y_t = Y_t\beta + V_t + U_t,$$

where V_t is the exogenous part, that depends on the inputs x_t and their initial condition only, and U_t is the endogenous part, that depends on the innovations a_t and their initial condition only. Then, the forecasts can be obtained separately by forecasting V_t and U_t, that are uncorrelated. To this end, one can use functions `ssmpredexg` and `ssmpred`, respectively. The latter is described in Section 5.4.2, whereas the former is as follows.

```
function [ypr, mypr, alpr, malpr] = ssmpredexg(n, x, stx, sts)
```

It is to be noted that if the inputs are stochastic, a model for them must be provided by the user. This model will be used in function `ssmpredexg` to obtain the input forecasts. If the inputs are not stochastic, the user must provide the forecasts.

For example, the following lines of MATLAB code can be used to first estimate a regression model with errors following a bivariate VARMAX model in echelon form and then to obtain eight forecasts.

```
s = 2;
Y = eye(s);
[xvfx, strx, ferror] = mexactestimc(yd, xd, strc, Y);
conp = strx.sigma2c;
npr = 8;
if (npr > 0)
  chb = 1;
  [ff, beta, e, f, str, stx, recrs] = exactmedfvc(xvfx, yd, xd, strx, ...
    Y, chb);
  A = stx.A; P = stx.P; Z = stx.Z; G = stx.G; T = stx.T; H = stx.H;
  hb = stx.hb; Mb = stx.Mb;
  Xp = Y;
  Wp = [];
  cw = 1.96;
  s = 2;
  [pry, mypr, alpr, malpr] = ssmpred(npr, s, A, P, Xp, Z, G, Wp, T, H, ...
    hb, Mb);
  spry = zeros(s, npr);
  xdx = xd;
  hr3 = 0;  finv2 = 1;
  [strv, ferror] = estvarmaxpqrPQR(xd, [], freq, [1 1 0], [0 1 0], hr3, ...
    finv2);
  sts.T = strv.Fs; sts.Z = strv.Hs; H = strv.Ks; Sg = strv.sigmar2;
  [R, p] = chol(Sg); L = R'; sts.H = H*L; sts.G = L;
  [prx, mxpr, glpr, mglpr] = ssmpredexg(npr, xdx, stx, sts);
  pry = pry + prx; mypr = mypr*conp + mxpr;
  for i = 1:npr
    spry(:, i) = sqrt(diag(mypr(:, :, i)));
  end
  opry = pry; ospry = spry;
  tname = 'var1';
  out.pry = pry(1, :); out.spry = spry(1, :);
  out.opry = opry(1, :); out.ospry = ospry(1, :); out.y = yd(:, 1);
  out.yor = yd(:, 1); out.ny = length(yd(:, 1)); out.npr = npr;
  out.cw = cw; out.tname = tname;
  lam = 1;
  out.lam = lam; out.s = freq;
  pfctsusm(out);
  tname = 'var2';
  out.pry = pry(2, :); out.spry = spry(2, :);
  out.opry = opry(2, :); out.ospry = ospry(2, :); out.y = yd(:, 2);
  out.yor = yd(:, 2); out.ny = length(yd(:, 2)); out.npr = npr; out.cw = cw;
  out.tname = tname;
  lam = 1;
  out.lam = lam; out.s = freq;
  pfctsusm(out);
end
```

5. Further functions

5.1. The Kalman filter and likelihood evaluation

Theoretical introduction

As described in the **SSMMATLAB** manual, the Kalman filter can be used for likelihood evaluation. Assuming $\beta = 0$ in (1) and (2) and $\delta = 0$, $a = 0$ in (3), the Kalman filter is given for $t = 1, \dots, n$ by the recursions

$$\begin{aligned} E_t &= Y_t - Z_t\hat{\alpha}_{t|t-1}, & \Sigma_t &= Z_tP_tZ_t^\top + G_tG_t^\top, \\ K_t &= (T_tP_tZ_t^\top + H_tG_t^\top)\Sigma_t^{-1}, & \hat{\alpha}_{t+1|t} &= T_t\hat{\alpha}_{t|t-1} + K_tE_t, \\ P_{t+1} &= (T_t - K_tZ_t)P_tT_t^\top + (H_t - K_tG_t)H_t^\top, \end{aligned} \tag{36}$$

initialized with $\hat{\alpha}_{1|0} = a$ and $P_1 = \Omega$. More details are given in the **SSMMATLAB** manual.

SSMMATLAB implementation

The MATLAB function used in **SSMMATLAB** for likelihood evaluation is

```
function [e, f, hb, Mb, A, P, qyy, R] = scakfle2(y, X, Z, G, W, T, H, ...
  ins, i, chb)
```

For parameter estimation, we first use function `scakfle2` to obtain the residual vector e and the constant f. Then, we multiply e by f to get $F = ef$, the vector of nonlinear functions that has to be minimized. Using the notation of the Kalman filter equations (36), $e_t = \Sigma_t^{-1/2}E_t$, $e = (e_1^\top, \dots, e_n^\top)^\top$ and $f = \prod_{t=1}^n |\Sigma_t|^{1/(2np)}$. More details are given in the **SSMMATLAB** manual.

Missing values

If the series has missing values, these should be replaced with the symbol `NaN` in MATLAB. The algorithms in **SSMMATLAB** are designed to take account of the missing values. For example, for a univariate series that follows an ARIMA model, each time the Kalman filter encounters a missing value, it skips this observation, sets $K_t = 0$ and continues filtering.

5.2. Recursive residuals

Theoretical introduction

Recursive residuals can be of two types, depending on whether one considers the estimated regression parameters fixed, together with the other parameters of the model, or not. More details are given in the **SSMMATLAB** manual.

SSMMATLAB implementation

When the estimated regression parameters are not considered fixed, in **SSMMATLAB** a square root information filter is applied to obtain the recursive residuals. The following function can be used for that purpose.

```
function [KKP, PT, hd, Md, initf, recrs, recr, srecr] = scakff(y, X, Z, ...
  G, W, T, H, ins, i)
```

It is to be noted that this function also provides the filtered state estimates, that is, the estimate of the state α_t based on the observations $Y_1, \dots, Y_t$, as well as their MSE. When the estimated regression parameters are considered fixed, together with the other parameters of the model, the following function can be used in **SSMMATLAB** to obtain the recursive residuals.

```
function [KKP, PT, recrs, recr, srecr, t1, A1, P1, KG] = scakfff(y, X, ...
  Z, G, W, T, H, ins, i, g)
```

5.3. Maximum likelihood parameter estimation

Theoretical introduction

Once the state space model has been defined and assuming that reasonable initial parameter values are available, the model can be estimated. It is to be emphasized that in **SSMMATLAB** we always concentrate one parameter out of the likelihood in the covariance matrix of the errors of the state space model. As shown in Section 5.1, this allows for the transformation of the log-likelihood maximization problem into a minimization of a nonlinear sum of squares function. In **SSMMATLAB**, the optimization method used is that of Levenberg-Marquardt (Levenberg 1944; Marquardt 1963). This method has been proved in practice to be a reliable method for minimizing a nonlinear sum of squares function.

***SSMMATLAB** implementation*

The following function can be used in **SSMMATLAB** for parameter estimation.

```
function [x, fjac, ff, g, iter, conf] = marqdt(info, x, varargin)
```

5.4. Forecasting

Theoretical introduction

Let the forecast or, equivalently, the orthogonal projection of α_{n+h} onto the sample $Y = (Y_1^\top, \ldots, Y_n^\top)^\top$ be $\hat{\alpha}_{n+h|n}$, where $h \geq 1$. Then, the h-period-ahead forecasts and their mean squared error, $\widehat{P}_{n+h}$, can be recursively obtained by

$$\begin{aligned} \hat{\alpha}_{n+h|n} &= v_{n+h} + U_{n+h}\hat{\gamma}_{n+1} \\ \widehat{P}_{n+h} &= \left(P_{n+h} + U_{n+h}\Pi_{n+1}U_{n+h}^\top\right)\hat{\sigma}^2, \end{aligned}$$

where $\hat{\gamma}_{n+1}$ and Π_{n+1} are the GLS estimator of γ based on Y and its MSE and for $h > 1$

$$\begin{aligned} (-U_{n+h}, v_{n+h}) &= (0, -W_{n+h-1}, 0) + T_{n+h-1}(-U_{n+h-1}, v_{n+h-1}) \\ P_{n+h} &= T_{n+h-1}P_{n+h-1}T_{n+h-1}^\top + H_{n+h-1}H_{n+h-1}^\top, \end{aligned}$$

$v_{n+1} = x_{n+1|n}$.

***SSMMATLAB** implementation*

In **SSMMATLAB**, the following function can be used for forecasting.

```
function [ypr, mypr, alpr, malpr] = ssmpred(n, p, A, P, X, Z, G, W, T, ...
  H, g, M)
```

5.5. Smoothing

Theoretical introduction

For smoothing, the Bryson-Frazier recursions are used, as described in the **SSMMATLAB** manual.

***SSMMATLAB** implementation*

The following function can be used in **SSMMATLAB** for smoothing of the state vector.

```
function [KKP, PT, hd, Md] = scakfs(y, X, Z, G, W, T, H, ins, i)
```

If it is of interest to smooth a general vector of the form $Y_t = U_t\beta + C_t\alpha_t + D_t\epsilon_t$, the following function can be used.

```
function [KKP, PT, hd, Md] = smoothgen(y, X, Z, G, W, T, H, ins, i, ...
  mucd, U, C, D)
```

6. Examples and case studies

In the **SSMMATLAB** manual in Gómez (2014) some examples are given on how to use **SSMMATLAB** in practice. These include estimation, computation of recursive residuals, forecasting and smoothing using univariate ARMA and ARMAX models, VARMA and VARMAX

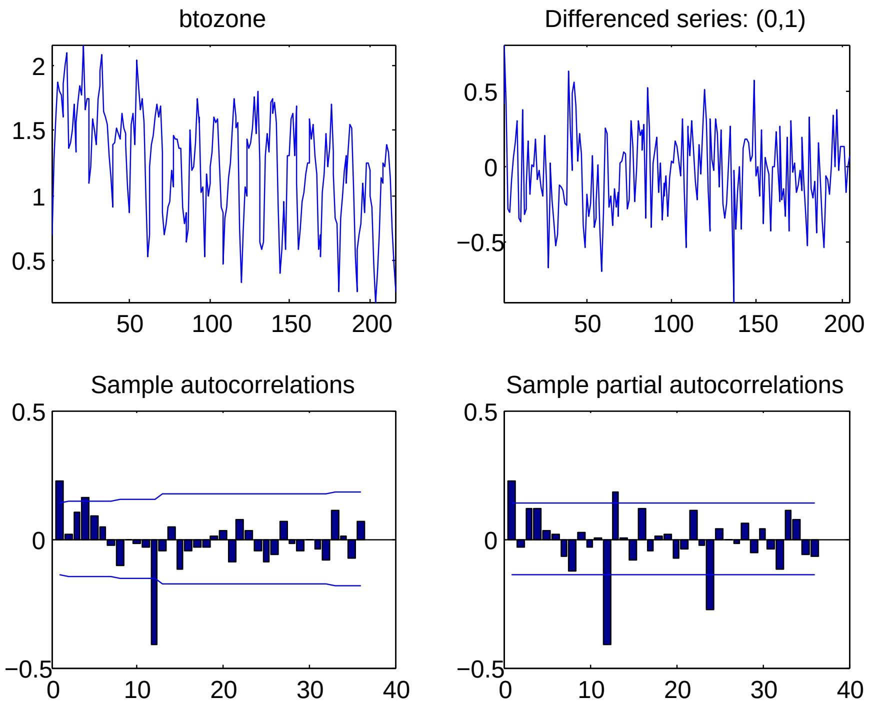

Figure 1: Ozone series: original, seasonally differenced, sample autocorrelations and sample partial autocorrelations.

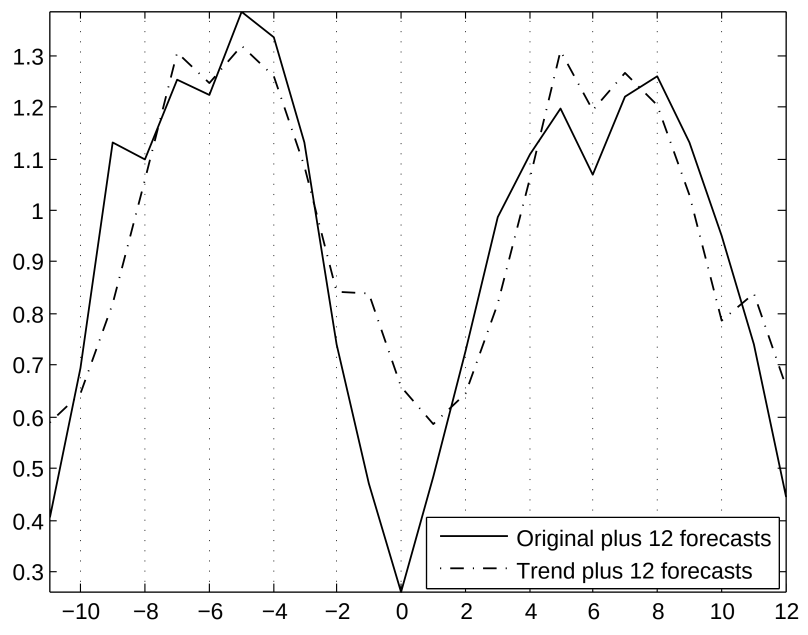

Figure 2: Ozone series: original and trend plus twelve forecasts.

models, AMB unobserved components models and univariate structural models. In addition, five case studies illustrate the use of **SSMMATLAB** to analyze sophisticated state space models that cannot be dealt with standard commercial packages.

As an illustration, we will present in this section one example and one case study. The example is Example 5 in the **SSMMATLAB** manual. In this example, the series of ozone levels used by Box and Tiao (1975) to introduce Intervention Analysis is considered. Unlike these authors, a structural model instead of an ARIMA model is specified. The model considered has a deterministic level, a seasonal component that is modeled as trigonometric seasonality and an autoregressive component of order one. In addition, there are three intervention variables corresponding to the interventions in Box and Tiao (1975). The MATLAB script file `usm2_d.m` contains the instructions for putting the model into state space form, and for model estimation, computation of recursive residuals, forecasting and smoothing of the trend. In Figure 1, one can see the original series together with the seasonally differenced series and its sample autocorrelations and partial autocorrelations. In Figure 2, the original and the trend series are displayed together with twelve forecasts of both series.

The case study is Case Study 4 in the **SSMMATLAB** manual. The purpose of the analysis is to first estimate the business cycle of the US Industrial Production Index for the periods 1946.Q1 through 2011.Q3. Then, to obtain bootstrap samples of the estimated cycle. The estimated cycle can be used as business cycle indicator while studying the cyclical comovements of different series. The cycle is estimated using two different methods. The first one consists of fitting a structural model that includes a cycle. The second one applies the AMB methodology described in Section 6.4 of the **SSMMATLAB** manual.

Two script files are used. In the first one, `USIPIstscl_d.m`, a structural model that includes a cycle is first fitted to the data and the cycle is estimated. Then, bootstrap samples of this

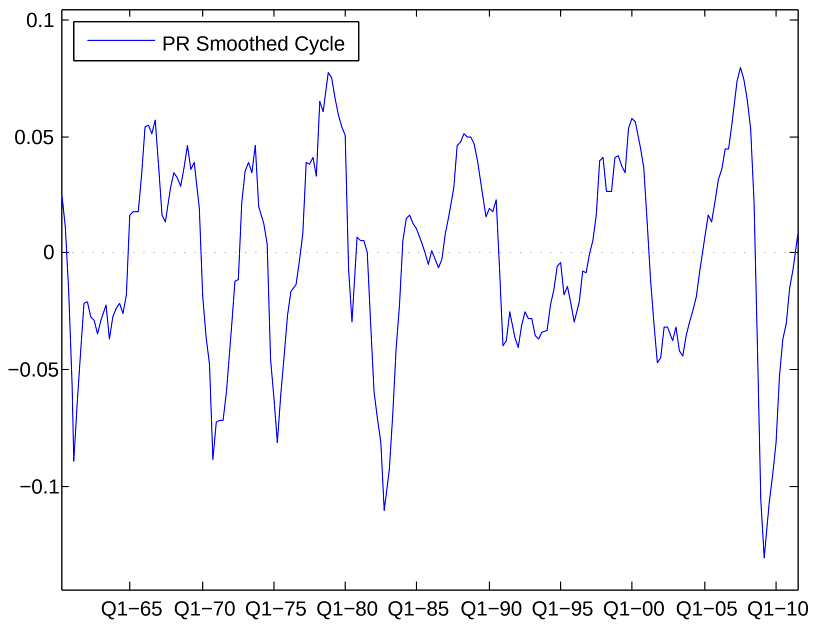

Figure 3: US Industrial Production cycle estimated using a structural model.

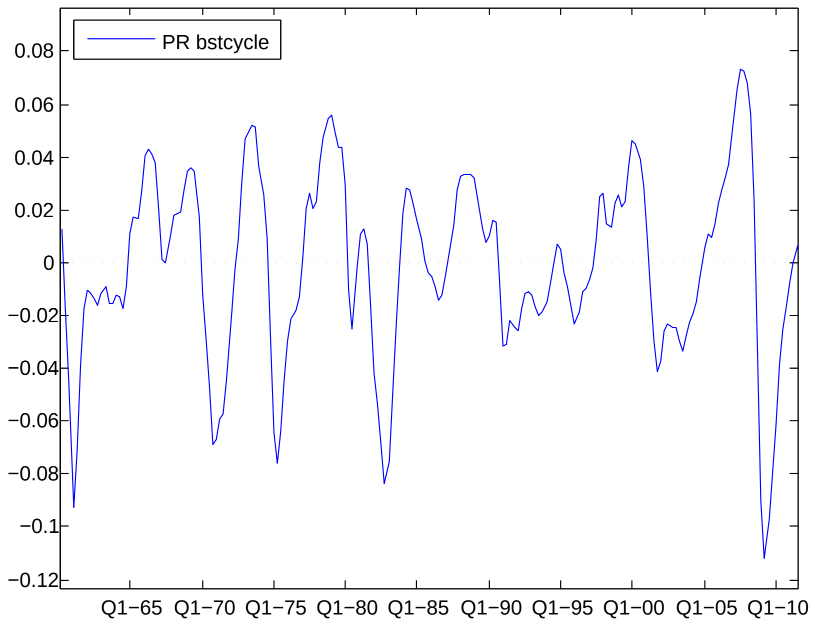

Figure 4: US Industrial Production cycle estimated using a band-pass filter within the AMB approach.

estimated cycle are obtained using the algorithm proposed by Stoffer and Wall (2004).

In the second file, `USIPIcdstcl_d.m`, the procedure proposed by Gómez (2001) is applied. Prior to the analysis, an ARIMA model was identified using the program TRAMO of Gómez

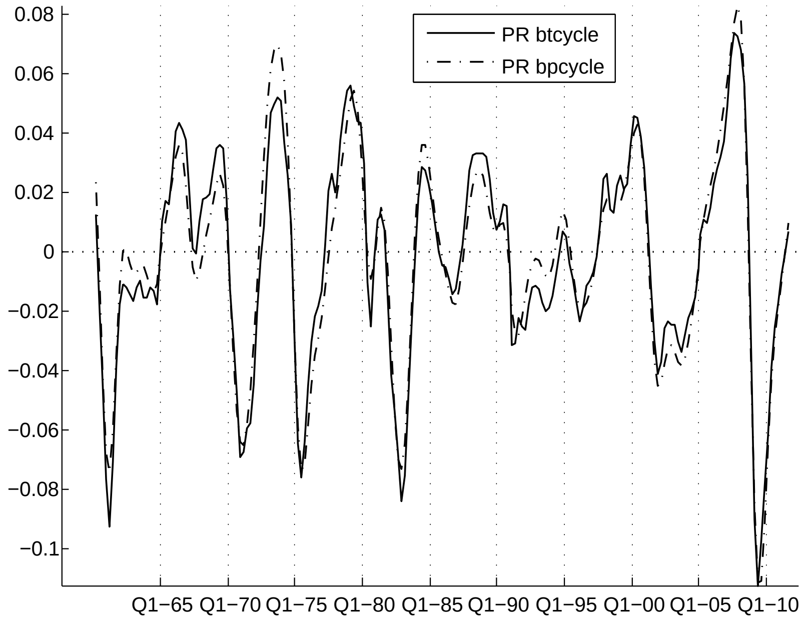

Figure 5: US Industrial Production cycles estimated using a band-pass and a low-pass filter (Hodrick-Prescott with $\lambda = 1600$) within the AMB approach.

and Maravall (2001a). In the script file, first the identified model is estimated by exact maximum likelihood and the models for the unobserved components, trend-cycle, seasonal and irregular, are obtained by means of the canonical decomposition. Then, based on the trend-cycle model and the model corresponding to the band-pass filter, models for the two subcomponents of the trend-cycle, the cycle and the smooth trend, are obtained as explained in Gómez (2001). After putting the new model into state space form, the Kalman filter and smoother are applied to estimate the smooth trend, the cycle, the seasonal and the irregular. Finally, bootstrap samples of the estimated cycle are obtained using the same method as in the case of the structural model.

The cycle estimated with the structural model is displayed in Figure 3. In Figure 4, the cycle estimated with a band-pass filter within the AMB approach is shown. It is seen that this cycle is smoother than the cycle estimated with the structural model. Finally, in Figure 5, two cycles are displayed. They correspond to two filters applied within the AMB approach. The first filter is the band-pass filter mentioned previously and the second one is a low-pass filter well known to economists, the Hodrick-Prescott filter corresponding to the parameter $\lambda = 1600$. Again, the cycle obtained with the band-pass filter is smoother than the one obtained with the low-pass filter. This is due to the fact that the band-pass filter extracts the components corresponding to the business cycle frequencies better than the low-pass filter.

References

Aptech Systems, Inc (2006). *GAUSS Mathematical and Statistical System 8.0*. Aptech Systems, Inc., Black Diamond, Washington. URL `http://www.Aptech.com/`.

Aruoba SB, Diebold FX, Scotti C (2009). "Real-Time Measurement of Business Conditions." *Journal of Business & Economic Statistics*, **27**(4), 417–427.

Bell WR (2011). "**REGCMPNT** – A Fortran Program for Regression Models with ARIMA Component Errors." *Journal of Statistical Software*, **41**(7), 1–23. URL http://www.jstatsoft.org/v41/i07/.

Box GEP, Tiao GC (1975). "Intervention Analysis with Applications to Economic and Environmental Problems." *Journal of the American Statistical Association*, **70**(349), 70–79.

Commandeur JJF, Koopman SJ, Ooms M (2011). "Statistical Software for State Space Methods." *Journal of Statistical Software*, **41**(1), 1–18. URL http://www.jstatsoft.org/v41/i01/.

Doan T (2011). "State Space Methods in **RATS**." *Journal of Statistical Software*, **41**(9), 1–16. URL http://www.jstatsoft.org/v41/i09/.

Drukker DM, Gates RB (2011). "State Space Methods in Stata." *Journal of Statistical Software*, **41**(10), 1–25. URL http://www.jstatsoft.org/v41/i10/.

Gómez V (2001). "The Use of Butterworth Filters for Trend and Cycle Estimation in Economic Time Series." *Journal of Business & Economic Statistics*, **19**(3), 365–373.

Gómez V (2013). "A Strongly Consistent Criterion to Decide between $I(1)$ and $I(0)$ Processes Based on Different Convergence Rates." *Communications in Statistics – Simulation and Computation*, **42**(8), 1848–1864.

Gómez V (2014). "**SSMMATLAB**." URL http://www.sepg.pap.minhap.gob.es/sitios/sepg/en-GB/Presupuestos/Documentacion/Paginas/SSMMATLAB.aspx.

Gómez V, Maravall A (2001a). "Programs **TRAMO** and **SEATS**, Instructions for the User (Beta Version: June 1997)." *Working Paper 97001*, Dirección General De Presupuestos, Ministry of Finance, Madrid, Spain.

Gómez V, Maravall A (2001b). "Seasonal Adjustment and Signal Extraction in Economic Time Series." In GC Tiao, D Peña, RS Tsay (eds.), *A Course in Time Series Analysis*, chapter 8. John Wiley & Sons, New York.

Hannan EJ, Kavalieris L (1984). "Multivariate Linear Time Series Models." *Advances in Applied Probability*, **16**(3), 492–561.

Hannan EJ, Kavalieris L (1986). "Regression, Autoregression Models." *Journal of Time Series Analysis*, **7**(1), 27–49.

Hannan EJ, Rissanen J (1982). "Recursive Estimation of Mixed Autoregressive-Moving Average Order." *Biometrika*, **69**(1), 81–94.

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

Harvey AC (1993). *Time Series Models*. 2nd edition. Harvester Wheatsheaf, Hemel Hempstead.

Hillmer S, Tiao G (1982). "An ARIMA-Model-Based Approach to Seasonal Adjustment." *Journal of the American Statistical Association*, **77**(377), 63–70.

Kitagawa G, Gersch W (1996). *Smoothness Priors Analysis of Time Series*. Springer-Verlag, New York.

Koopman SJ, Harvey AC, Doornik JA, Shephard N (2009). ***STAMP** 8.2: Structural Time Series Analyser, Modeler, and Predictor*. Timberlake Consultants, London.

Levenberg K (1944). "A Method for the Solution of Certain Non-Linear Problems in Least Squares." *The Quarterly of Applied Mathematics*, **2**, 164–168.

Lucchetti R (2011). "State Space Methods in gretl." *Journal of Statistical Software*, **41**(11), 1–22. URL http://www.jstatsoft.org/v41/i11/.

Lütkepohl H (2007). *New Introduction to Multiple Time Series Analysis*. Springer-Verlag, Berlin.

Marquardt D (1963). "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." *SIAM Journal on Applied Mathematics*, **11**(2), 431–441.

Mendelssohn R (2011). "The **STAMP** Software for State Space Models." *Journal of Statistical Software*, **41**(2), 1–18. URL http://www.jstatsoft.org/v41/i02/.

Otnes RK, Enochson L (1978). *Applied Time Series Analysis*, volume 1. John Wiley & Sons, New York.

Pelagatti MM (2011). "State Space Methods in Ox/**SsfPack**." *Journal of Statistical Software*, **41**(3), 1–25. URL http://www.jstatsoft.org/v41/i03/.

Peng JY, Aston JAD (2011). "The State Space Models Toolbox for MATLAB." *Journal of Statistical Software*, **41**(6), 1–26. URL http://www.jstatsoft.org/v41/i06/.

Petris G, Petrone S (2011). "State Space Models in R." *Journal of Statistical Software*, **41**(4), 1–25. URL http://www.jstatsoft.org/v41/i04/.

Primiceri GE (2005). "Time Varying Structural Vector Autoregressions and Monetary Policy." *The Review of Economic Studies*, **72**(3), 821–852.

Reinsel GC (1997). *Elements of Multivariate Time Series Analysis*. Springer-Verlag, New York.

Selukar R (2011). "State Space Modeling Using SAS." *Journal of Statistical Software*, **41**(12), 1–13. URL http://www.jstatsoft.org/v41/i12/.

Stoffer DS, Wall KD (2004). "Resampling in State Space Models." In AC Harvey, SJ Koopman, N Shephard (eds.), *State Space and Unobserved Components: Theory and Applications*. Cambridge University Press, Cambridge.

The MathWorks Inc (2014). *MATLAB – The Language of Technical Computing, Version R2014b*. Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Van den Bossche FAM (2011). "Fitting State Space Models with EViews." *Journal of Statistical Software*, **41**(8), 1–16. URL http://www.jstatsoft.org/v41/i08/.

Affiliation:

Víctor Gómez
Ministerio de Hacienda y Administraciones Públicas
Subdirección Gral. de Análisis y P.E.
Alberto Alcocer 2, 1-P, D-34
28046, Madrid, Spain
E-mail: VGomez@sepg.minhap.es
URL: http://www.sepg.pap.minhap.gob.es/sitios/sepg/en-GB/Presupuestos/Documentacion/Paginas/SSMMATLAB.aspx

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 66, Issue 9 *Submitted:* 2013-07-09
August 2015 *Accepted:* 2014-10-31