6 Further computational aspects

6.1 Introduction

In this chapter we will discuss a number of remaining computational aspects of the Kalman filter and smoother. We shall continue the treatment based on classical analysis on the assumption that results for linear estimation or Bayesian analysis can be obtained by application of Lemmas 2, 3 and 4. Two different ways of incorporating regression effects within the Kalman filter are described in Section 6.2. The standard Kalman filter recursion for the variance matrix of the filtered state vector does not rule out the possibility that this matrix becomes negative definite; this is obviously an undesirable outcome since it indicates the presence of rounding errors. The square root Kalman filter eliminates this problem at the expense of slowing down the filtering and smoothing processes; details are given in Section 6.3. The computational costs of implementing the filtering and smoothing procedures of Chapters 4 and 5 can become high for high-dimensional multivariate models, particularly in dealing with the initialisation problem. It turns out that by bringing the elements of the observation vectors in multivariate models into the filtering and smoothing computing operations one at a time, substantial gains in computational efficiency are achieved and the initialisation problem is simplified considerably. Methods based on this idea are developed in Section 6.4. The various algorithms presented in the Chapters 4 and 5 and this chapter need to be implemented efficiently on a computer. Different computer packages have been developed that implement the algorithms considered in this book. *SsfPack* is an example of such a package. Section 6.7 reviews the features of packages for state space methods and provides an illustration using *SsfPack*.

6.2 Regression estimation

6.2.1 Introduction

As for the structural time series model considered in Section 3.2, the general state space model can be extended to allow for the incorporation of explanatory variables and intervention variables into the model. To accomplish this generalisation we replace the measurement equation of the state space model (3.1) by

$$y_t = Z_t\alpha_t + X_t\beta + \varepsilon_t, \tag{6.1}$$

where $X_t = (x_{1,t}, \ldots, x_{k,t})$ is a $p \times k$ matrix of explanatory variables and β is a $k \times 1$ vector of unknown regression coefficients which we assume are constant over time and which we wish to estimate. We will not discuss here time-varying regression coefficients because they can be included as part of the state vector α_t in an obvious way as in Subsection 3.6.1 and are then dealt with by the standard Kalman filter and smoother. There are two ways in which the inclusion of regression effects with fixed coefficients can be handled. First, we may include the coefficient vector β in the state vector. Alternatively, and particularly on occasions where we wish to keep the dimensionality of the state vector as low as possible, we can use the augmented Kalman filter and smoother. Both solutions will be discussed in the next two sections. Different types of residuals exist when regression variables are included in the model. We show in Subsection 6.2.4 how to calculate them within the two different solutions.

6.2.2 Inclusion of coefficient vector in state vector

The state space model in which the constant coefficient vector β in (6.1) is included in the state vector has the form

$$y_t = [Z_t \quad X_t] \begin{pmatrix} \alpha_t \\ \beta_t \end{pmatrix} + \varepsilon_t,$$

$$\begin{pmatrix} \alpha_{t+1} \\ \beta_{t+1} \end{pmatrix} = \begin{bmatrix} T_t & 0 \\ 0 & I_k \end{bmatrix} \begin{pmatrix} \alpha_t \\ \beta_t \end{pmatrix} + \begin{bmatrix} R_t \\ 0 \end{bmatrix} \eta_t,$$

for $t = 1, \ldots, n$. In the initial state vector, β_1 can be taken as diffuse or fixed. In the diffuse case the model for the initial state vector is

$$\begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} \sim \mathrm{N}\left\{ \begin{pmatrix} a \\ 0 \end{pmatrix}, \kappa \begin{bmatrix} P_\infty & 0 \\ 0 & I_k \end{bmatrix} + \begin{bmatrix} P_* & 0 \\ 0 & 0 \end{bmatrix} \right\},$$

where $\kappa \to \infty$; see also Subsection 5.6.4 where we give the initial state vector for the regression model with ARMA errors. We attach suffixes to the β's purely for convenience in the state space formulation since $\beta_{t+1} = \beta_t = \beta$. The exact initial Kalman filter (5.19) and the Kalman filter (4.24) can be applied straightforwardly to this enlarged state space model to obtain the estimate of β. The enlargement of the state space model will not cause much extra computing because of the sparse nature of the system matrices.

6.2.3 Regression estimation by augmentation

Another method of estimating β is by augmentation of the Kalman filter. This technique is essentially the same as was used in the augmented Kalman filter of Section 5.7. We will give details of this approach on the assumption that the initial state vector does not contain diffuse elements. The likelihood function

in terms of β is constructed by applying the Kalman filter to the variables y_t, $x_{1,t}, \ldots, x_{k,t}$ in turn. Each of the variables $x_{1,t}, \ldots, x_{k,t}$ is treated in the Kalman filter as the observation vector with the same variance elements as used for y_t. Denote the resulting one-step ahead forecast errors by v_t^*, $x_{1,t}^*, \ldots, x_{k,t}^*$, respectively. Since the filtering operations are linear, the one-step ahead forecast errors for the series $y_t - X_t\beta$ are given by $v_t = v_t^* - X_t^*\beta$ where $X_t^* = (x_{1,t}^* \ldots x_{k,t}^*)$. It should be noted that the $k+1$ Kalman filters are the same except that the values for v_t and a_t in (4.24) are different. We can therefore combine these filters into an augmented Kalman filter where we replace vector y_t by matrix (y_t, X_t) to obtain the 'innovations' matrix (v_t^*, X_t^*); this is analogous to the augmented Kalman filter described in Section 5.7.

The one-step ahead forecast errors $v_1, \ldots, v_n$, with the corresponding variances $F_1, \ldots, F_n$, are independent of each other by construction. The sum of squared standardised forecast errors become therefore simply

$$\sum_{t=1}^{n} v_t' F_t^{-1} v_t = \sum_{t=1}^{n} (v_t^* - X_t^*\beta)' F_t^{-1} (v_t^* - X_t^*\beta),$$

and can be minimised with respect to β directly. We then obtain the *generalised least squares estimate* of β with its variance matrix. They are given by

$$\hat{\beta} = \left(\sum_{t=1}^{n} X_t^{*\prime} F_t^{-1} X_t^* \right)^{-1} \sum_{t=1}^{n} X_t^{*\prime} F_t^{-1} v_t^*, \qquad \mathrm{Var}(\hat{\beta}) = \left(\sum_{t=1}^{n} X_t^{*\prime} F_t^{-1} X_t^* \right)^{-1}. \tag{6.2}$$

In the case where the initial state vector contains diffuse elements we can extend the augmented Kalman filter for δ as shown in Section 5.7. However, we ourselves prefer to use the exact initial Kalman filter for dealing with δ. The equations for $P_{*,t}$ and $P_{\infty,t}$ of the exact initial Kalman filter are not affected since they do not depend on the data. The update for the augmented state vector is given by the equations

$$\begin{aligned} (v_t^*, x_{1,t}^*, \ldots, x_{k,t}^*) &= (y_t, x_{1,t}, \ldots, x_{k,t}) - Z_t(a_{a,t}, A_{x,t}), \\ (a_{a,t+1}, A_{x,t+1}) &= T_t(a_{a,t}, A_{x,t}) + K_t^{(0)}(v_t^*, x_{1,t}^*, \ldots, x_{k,t}^*), \end{aligned} \tag{6.3}$$

for $t = 1, \ldots, d$ with $(a_{a,1}, A_{x,1}) = (a, 0, \ldots, 0)$ and where $K_t^{(0)}$ is defined in Section 5.2. Note that F_t^{-1} in (6.2) must be replaced by zero or $F_{*,t}^{-1}$ depending on the value for $F_{\infty,t}$ in the exact initial Kalman filter. Overall, the treatment given in the previous section where we include β in the state vector, treat δ and β as diffuse and then apply the exact initial Kalman filter is conceptually simpler, though it may not be as efficient computationally for large models.

6.2.4 Least squares and recursive residuals

By considering the measurement equation (6.1) we define two different types of residuals following Harvey (1989): recursive residuals and least squares residuals. The first type are defined as

$$v_t = y_t - Z_t a_t - X_t \hat{\beta}_{t-1}, \qquad t = d+1, \ldots, n,$$

where $\hat{\beta}_{t-1}$ is the maximum likelihood estimate of β given Y_{t-1}. The residuals v_t are computed easily by including β in the state vector with α_1 diffuse since the filtered state vector of the enlarged model in Subsection 6.2.2 contains $\hat{\beta}_{t-1}$. The augmentation method can of course also evaluate v_t but it needs to compute $\hat{\beta}_{t-1}$ at each time point which is not computationally efficient. Note that the residuals v_t are serially uncorrelated. The least squares residuals are given by

$$v_t^+ = y_t - Z_t a_t - X_t \hat{\beta}, \qquad t = d+1, \ldots, n,$$

where $\hat{\beta}$ is the maximum likelihood estimate of β based on the entire series, so $\hat{\beta} = \hat{\beta}_n$. For the case where the method of Subsection 6.2.2 is used to compute $\hat{\beta}$, we require two Kalman filters: one for the enlarged model to compute $\hat{\beta}$ and a Kalman filter for the constructed measurement equation $y_t - X_t\hat{\beta} = Z_t\alpha_t + \varepsilon_t$ whose 'innovations' v_t are actually v_t^+ for $t = d+1, \ldots, n$. The same applies to the method of Subsection 6.2.3, except that $\hat{\beta}$ is computed using equation (6.2). The least squares residuals are correlated due to the presence of $\hat{\beta}$ in these residuals, which is calculated from the whole sample.

Both sets of residuals can be used for diagnostic purposes. For these purposes the residuals v_t have the advantage of being serially uncorrelated whereas the residuals v_t^+ have the advantage of being based on the estimate $\hat{\beta}$ calculated from the whole sample. For further discussion we refer to Harvey (1989, Section 7.4.1).

6.3 Square root filter and smoother

6.3.1 Introduction

In this section we deal with the situation where, because of rounding errors and matrices being close to singularity, the possibility arises that the calculated value of P_t is negative definite, or close to this, giving rise to unacceptable rounding errors. From (4.24), the state variance matrix P_t is updated by the Kalman filter equations

$$\begin{aligned} F_t &= Z_t P_t Z_t' + H_t, \\ K_t &= T_t P_t Z_t' F_t^{-1}, \\ P_{t+1} &= T_t P_t L_t' + R_t Q_t R_t' \\ &= T_t P_t T_t' + R_t Q_t R_t' - K_t F_t K_t', \end{aligned} \tag{6.4}$$

where $L_t = T_t - K_t Z_t$. It can happen that the calculated value of P_t becomes negative definite when, for example, erratic changes occur in the system matrices over time. The problem can be avoided by using a transformed version of the Kalman filter called the *square root filter*. However, the amount of computation required is substantially larger than that required for the standard Kalman filter. The square root filter is based on orthogonal lower triangular transformations for which we can use Givens rotation techniques. The standard reference to square root filtering is Morf and Kailath (1975).

6.3.2 Square root form of variance updating

Define the partitioned matrix U_t by

$$U_t = \begin{bmatrix} Z_t\tilde{P}_t & \tilde{H}_t & 0 \\ T_t\tilde{P}_t & 0 & R_t\tilde{Q}_t \end{bmatrix}, \tag{6.5}$$

where

$$P_t = \tilde{P}_t\tilde{P}_t', \qquad H_t = \tilde{H}_t\tilde{H}_t', \qquad Q_t = \tilde{Q}_t\tilde{Q}_t',$$

in which the matrices $\tilde{P}_t$, $\tilde{H}_t$ and $\tilde{Q}_t$ are lower triangular matrices. It follows that

$$U_tU_t' = \begin{bmatrix} F_t & Z_tP_tT_t' \\ T_tP_tZ_t' & T_tP_tT_t' + R_tQ_tR_t' \end{bmatrix}. \tag{6.6}$$

The matrix U_t can be transformed to a lower triangular matrix using the orthogonal matrix G such that $GG' = I_{m+p+r}$. Note that a lower triangular matrix for a rectangular matrix such as U_t, where the number of columns exceeds the number of rows, is defined as a matrix of the form $[A \quad 0]$ where A is a square and lower triangular matrix. Postmultiplying by G we have

$$U_tG = U_t^*, \tag{6.7}$$

and $U_t^*U_t^{*\prime} = U_tU_t'$ as given by (6.6). The lower triangular rectangular matrix U_t^* has the same dimensions as U_t and can be represented as the partitioned matrix

$$U_t^* = \begin{bmatrix} U_{1,t}^* & 0 & 0 \\ U_{2,t}^* & U_{3,t}^* & 0 \end{bmatrix},$$

where $U_{1,t}^*$ and $U_{3,t}^*$ are lower triangular square matrices. It follows that

$$\begin{aligned} U_t^*U_t^{*\prime} &= \begin{bmatrix} U_{1,t}^*U_{1,t}^{*\prime} & U_{1,t}^*U_{2,t}^{*\prime} \\ U_{2,t}^*U_{1,t}^{*\prime} & U_{2,t}^*U_{2,t}^{*\prime} + U_{3,t}^*U_{3,t}^{*\prime} \end{bmatrix} \\ &= \begin{bmatrix} F_t & Z_tP_tT_t' \\ T_tP_tZ_t' & T_tP_tT_t' + R_tQ_tR_t' \end{bmatrix}, \end{aligned}$$

from which we deduce that

$$U^*_{1,t} = \tilde{F}_t,$$
$$U^*_{2,t} = T_t P_t Z_t' \tilde{F}_t'^{-1} = K_t \tilde{F}_t,$$

where $F_t = \tilde{F}_t \tilde{F}_t'$ and $\tilde{F}_t$ is lower triangular. It is remarkable to find that $U^*_{3,t} = \tilde{P}_{t+1}$ since

$$\begin{aligned} U^*_{3,t} U^{*\prime}_{3,t} &= T_t P_t T_t' + R_t Q_t R_t' - U^*_{2,t} U^{*\prime}_{2,t} \\ &= T_t P_t T_t' + R_t Q_t R_t' - K_t F_t K_t' \\ &= P_{t+1}, \end{aligned}$$

which follows from (6.4). Thus by transforming U_t in (6.5) to a lower triangular matrix we obtain $\tilde{P}_{t+1}$; this operation can thus be regarded as a square root recursion for P_t. The update for the state vector a_t can be easily incorporated using

$$\begin{aligned} a_{t+1} &= T_t a_t + K_t v_t, \\ &= T_t a_t + T_t P_t Z_t' \tilde{F}_t'^{-1} \tilde{F}_t^{-1} v_t \\ &= T_t a_t + U^*_{2,t} U^{*-1}_{1,t} v_t, \end{aligned}$$

where $v_t = y_t - Z_t a_t$. Note that the inverse of $U^*_{1,t}$ is easy to calculate since it is a lower triangular matrix.

6.3.3 Givens rotations

Matrix G can be any orthogonal matrix which transforms U_t to a lower triangular matrix. Many different techniques can be used to achieve this objective; for example, Golub and Van Loan (1996) give a detailed treatment of the Householder and Givens matrices for the purpose. We will give here a short description of the latter. The orthogonal 2×2 matrix

$$G_2 = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \tag{6.8}$$

with $c^2 + s^2 = 1$ is the key to Givens transformations. It is used to transform the vector

$$x = (x_1 \quad x_2),$$

into a vector in which the second element is zero, that is

$$y = xG_2 = (y_1 \quad 0),$$

by taking

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}, \qquad s = -\frac{x_2}{\sqrt{x_1^2 + x_2^2}}, \tag{6.9}$$

for which $c^2 + s^2 = 1$ and $sx_1 + cx_2 = 0$. Note that $y_1 = cx_1 - sx_2$ and $yG_2' = xG_2G_2' = x$.

The general Givens matrix G is defined as the identity matrix I_q but with four elements $I_{ii}, I_{jj}, I_{ij}, I_{ji}$ replaced by

$$\begin{aligned} G_{ii} &= G_{jj} = c, \\ G_{ij} &= s, \\ G_{ji} &= -s, \end{aligned}$$

for $1 \leq i < j \leq q$ and with c and s given by (6.9) but now enforcing element (i, j) of matrix XG to be zero for all $1 \leq i < j \leq q$ and for any matrix X. It follows that $GG' = I$ so when Givens rotations are applied repeatedly to create zero blocks in a matrix, the overall transformation matrix is also orthogonal. These properties of the Givens rotations, their computational efficiency and their numerical stability make them a popular tool to transform nonzero matrices into sparse matrices such as a lower triangular matrix. More details and efficient algorithms for Givens rotations are given by Golub and Van Loan (1996).

6.3.4 Square root smoothing

The backward recursion (4.42) for N_{t-1} of the basic smoothing equations can also be given in a square root form. These equations use the output of the square root Kalman filter as given by:

$$\begin{aligned} U_{1,t}^* &= \tilde{F}_t, \\ U_{2,t}^* &= K_t\tilde{F}_t, \\ U_{3,t}^* &= \tilde{P}_{t+1}. \end{aligned}$$

The recursion for N_{t-1} is given by

$$N_{t-1} = Z_t'F_t^{-1}Z_t + L_t'N_tL_t,$$

where

$$\begin{aligned} F_t^{-1} &= (U_{1,t}^*U_{1,t}^{*\prime})^{-1}, \\ L_t &= T_t - U_{2,t}^*U_{1,t}^{*-1}Z_t. \end{aligned}$$

We introduce the lower triangular square matrix $\tilde{N}_t$ such that

$$N_t = \tilde{N}_t\tilde{N}_t',$$

and the $m \times (m + p)$ matrix

$$\tilde{N}_{t-1}^* = \begin{bmatrix} Z_t'U_{1,t}^{*-1\prime} & L_t'\tilde{N}_t \end{bmatrix},$$

from which it follows that $N_{t-1} = \tilde{N}_{t-1}^*\tilde{N}_{t-1}^{*\prime}$.

The matrix N^*_{t-1} can be transformed to a lower triangular matrix using some orthogonal matrix G such that $GG' = I_{m+p}$; compare Subsection 6.3.2. We have

$$\tilde{N}^*_{t-1}G = [\,\tilde{N}_{t-1} \quad 0\,],$$

such that $N_{t-1} = \tilde{N}^*_{t-1}\tilde{N}^{*\prime}_{t-1} = \tilde{N}_{t-1}\tilde{N}'_{t-1}$. Thus by transforming the matrix N^*_{t-1}, depending on matrices indexed by time t, to a lower triangular matrix we obtain the square root matrix of N_{t-1}. Consequently we have developed a backward recursion for N_{t-1} in square root form. The backwards recursion for r_{t-1} is not affected apart from the way in which F_t^{-1} and L_t are computed.

6.3.5 Square root filtering and initialisation

The square root formulation could be developed for the exact initial version of the Kalman filter of Section 5.2. However, the motivation for developing square root versions for filtering and smoothing is to avoid computational numerical instabilities due to rounding errors which are built up during the recursive computations. Since initialisation usually only requires a limited number of d updates, the numerical problems are not substantial during this process. Thus, although use of the square root filter may be important for $t = d, \ldots, n$, it will normally be adequate to employ the standard exact initial Kalman filter as described in Section 5.2 for $t = 1, \ldots, d$.

The square root formulation of the augmented Kalman filter of Section 5.7 is more or less the same as the usual Kalman filter because updating equations for F_t, K_t and P_t are unaltered. Some adjustments are required for the updating of the augmented quantities but these can be derived straightforwardly; some details are given by de Jong (1991). Snyder and Saligari (1996) have proposed a Kalman filter based on Givens rotations, such as the ones developed in Subsection 6.3.3, with the fortunate property that diffuse priors $\kappa \to \infty$ can be dealt with explicitly within the Givens operations. Their application of this solution, however, was limited to filtering only and it does not seem to provide an adequate solution for initial diffuse smoothing.

6.3.6 Illustration: local linear trend model

For the local linear trend model (3.2) we take

$$U_t = \begin{bmatrix} \tilde{P}_{11,t} & 0 & \sigma_\varepsilon & 0 & 0 \\ \tilde{P}_{11,t} + \tilde{P}_{21,t} & \tilde{P}_{22,t} & 0 & \sigma_\xi & 0 \\ \tilde{P}_{21,t} & \tilde{P}_{22,t} & 0 & 0 & \sigma_\zeta \end{bmatrix},$$

which is transformed to the lower triangular matrix

$$U^*_t = \begin{bmatrix} \tilde{F}_t & 0 & 0 & 0 & 0 \\ K_{11,t}\tilde{F}_t & \tilde{P}_{11,t+1} & 0 & 0 & 0 \\ K_{21,t}\tilde{F}_t & \tilde{P}_{21,t+1} & \tilde{P}_{22,t+1} & 0 & 0 \end{bmatrix}.$$

The zero elements of U_t^* are created row-wise by a sequence of Givens rotations applied to the matrix U_t. Some zero elements in U_t^* are already zero in U_t and they mostly remain zero within the overall Givens transformation so the number of computations can be limited somewhat.

6.4 Univariate treatment of multivariate series

6.4.1 Introduction

In Chapters 4 and 5 and in this chapter we have treated the filtering and smoothing of multivariate series in the traditional way by taking the entire observational vectors y_t as the items for analysis. In this section we present an alternative approach in which the elements of y_t are brought into the analysis one at a time, thus in effect converting the multivariate series into a univariate time series. This device not only offers significant computational gains for the filtering and smoothing of the bulk of the series but it also provides substantial simplification of the initialisation process when the initial state vector α_1 is partially or wholly diffuse.

This univariate approach to vector observations was suggested for filtering by Anderson and Moore (1979) and for filtering and smoothing longitudinal models by Fahrmeir and Tutz (1994). The treatment given by these authors was, however, incomplete and in particular did not deal with the initialisation problem, where the most substantial gains are made. The following discussion of the univariate approach is based on Koopman and Durbin (2000) who gave a complete treatment including a discussion of the initialisation problem.

6.4.2 Details of univariate treatment

Our analysis will be based on the standard model

$$y_t = Z_t\alpha_t + \varepsilon_t, \qquad \alpha_{t+1} = T_t\alpha_t + R_t\eta_t,$$

with $\varepsilon_t \sim \mathrm{N}(0, H_t)$ and $\eta_t \sim \mathrm{N}(0, Q_t)$ for $t = 1, \ldots, n$. To begin with, let us assume that $\alpha_1 \sim \mathrm{N}(a_1, P_1)$ and H_t is diagonal; this latter restriction will be removed later. On the other hand, we introduce two slight generalisations of the basic model: first, we permit the dimensionality of y_t to vary over time by taking the dimension of vector y_t to be $p_t \times 1$ for $t = 1, \ldots, n$; second, we do not require the prediction error variance matrix F_t to be nonsingular.

Write the observation and disturbance vectors as

$$y_t = \begin{pmatrix} y_{t,1} \\ \vdots \\ y_{t,p_t} \end{pmatrix}, \qquad \varepsilon_t = \begin{pmatrix} \varepsilon_{t,1} \\ \vdots \\ \varepsilon_{t,p_t} \end{pmatrix},$$

and the observation equation matrices as

$$Z_t = \begin{pmatrix} Z_{t,1} \\ \vdots \\ Z_{t,p_t} \end{pmatrix}, \qquad H_t = \begin{pmatrix} \sigma^2_{t,1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma^2_{t,p_t} \end{pmatrix},$$

where $y_{t,i}$, $\varepsilon_{t,i}$ and $\sigma^2_{t,i}$ are scalars and $Z_{t,i}$ is a $(1\times m)$ row vector, for $i = 1, \ldots, p_t$. The observation equation for the univariate representation of the model is

$$y_{t,i} = Z_{t,i}\alpha_{t,i} + \varepsilon_{t,i}, \qquad i = 1, \ldots, p_t, \qquad t = 1, \ldots, n, \tag{6.10}$$

where $\alpha_{t,i} = \alpha_t$. The state equation corresponding to (6.10) is

$$\begin{aligned} &\alpha_{t,i+1} = \alpha_{t,i}, && i = 1, \ldots, p_t - 1, \\ &\alpha_{t+1,1} = T_t\alpha_{t,p_t} + R_t\eta_t, && t = 1, \ldots, n, \end{aligned} \tag{6.11}$$

where the initial state vector $\alpha_{1,1} = \alpha_1 \sim N(a_1, P_1)$.

Define

$$a_{t,1} = \mathrm{E}(\alpha_{t,1}|Y_{t-1}), \qquad P_{t,1} = \mathrm{Var}(\alpha_{t,1}|Y_{t-1}),$$

and

$$\begin{aligned} a_{t,i} &= \mathrm{E}(\alpha_{t,i}|Y_{t-1}, y_{t,1}, \ldots, y_{t,i-1}), \\ P_{t,i} &= \mathrm{Var}(\alpha_{t,i}|Y_{t-1}, y_{t,1}, \ldots, y_{t,i-1}), \end{aligned}$$

for $i = 2, \ldots, p_t$. By treating the vector series $y_1, \ldots, y_n$ as the scalar series

$$y_{1,1}, \ldots, y_{1,p_1}, y_{2,1}, \ldots, y_{n,p_n},$$

the filtering equations (4.24) can be written as

$$a_{t,i+1} = a_{t,i} + K_{t,i}v_{t,i}, \qquad P_{t,i+1} = P_{t,i} - K_{t,i}F_{t,i}K'_{t,i}, \tag{6.12}$$

where

$$v_{t,i} = y_{t,i} - Z_{t,i}a_{t,i}, \qquad F_{t,i} = Z_{t,i}P_{t,i}Z'_{t,i} + \sigma^2_{t,i}, \qquad K_{t,i} = P_{t,i}Z'_{t,i}F^{-1}_{t,i}, \tag{6.13}$$

for $i = 1, \ldots, p_t$ and $t = 1, \ldots, n$. This formulation has $v_{t,i}$ and $F_{t,i}$ as scalars and $K_{t,i}$ as a column vector. The transition from time t to time $t+1$ is achieved by the relations

$$a_{t+1,1} = T_t a_{t,p_t+1}, \qquad P_{t+1,1} = T_t P_{t,p_t+1}T'_t + R_t Q_t R'_t. \tag{6.14}$$

The values $a_{t+1,1}$ and $P_{t+1,1}$ are the same as the values a_{t+1} and P_{t+1} computed by the standard Kalman filter.

It is important to note that the elements of the innovation vector v_t are not the same as $v_{t,i}$, for $i = 1, \ldots, p_t$; only the first element of v_t is equal to $v_{t,1}$. The same applies to the diagonal elements of the variance matrix F_t and the variances $F_{t,i}$, for $i = 1, \ldots, p_t$; only the first diagonal element of F_t is equal to $F_{t,1}$. It should be emphasised that there are models for which $F_{t,i}$ can be zero, for example the case where y_t is a multinomial observation with all cell counts included in y_t. This indicates that $y_{t,i}$ is linearly dependent on previous observations for some i. In this case,

$$\begin{aligned} a_{t,i+1} &= \mathrm{E}(\alpha_{t,i+1}|Y_{t-1}, y_{t,1}, \ldots, y_{t,i}) \\ &= \mathrm{E}(\alpha_{t,i+1}|Y_{t-1}, y_{t,1}, \ldots, y_{t,i-1}) = a_{t,i}, \end{aligned}$$

and similarly $P_{t,i+1} = P_{t,i}$. The contingency is therefore easily dealt with.

The basic smoothing recursions (4.69) for the standard state space model can be reformulated for the univariate series

$$y_{1,1}, \ldots, y_{1,p_t}, y_{2,1}, \ldots, y_{n,p_n},$$

as

$$\begin{aligned} r_{t,i-1} &= Z'_{t,i}F_{t,i}^{-1}v_{t,i} + L'_{t,i}r_{t,i}, & N_{t,i-1} &= Z'_{t,i}F_{t,i}^{-1}Z_{t,i} + L'_{t,i}N_{t,i}L_{t,i}, \\ r_{t-1,p_{t-1}} &= T'_{t-1}r_{t,0}, & N_{t-1,p_{t-1}} &= T'_{t-1}N_{t,0}T_{t-1}, \end{aligned} \tag{6.15}$$

where $L_{t,i} = I_m - K_{t,i}Z_{t,i}$, for $i = p_t, \ldots, 1$ and $t = n, \ldots, 1$. The initialisations are $r_{n,p_n} = 0$ and $N_{n,p_n} = 0$. The equations for $r_{t-1,p_{t-1}}$ and $N_{t-1,p_{t-1}}$ do not apply for $t = 1$. The values for $r_{t,0}$ and $N_{t,0}$ are the same as the values for the smoothing quantities r_{t-1} and N_{t-1} of the standard smoothing equations, respectively.

The smoothed state vector $\hat{\alpha}_t = \mathrm{E}(\alpha_t|Y_n)$ and the variance error matrix $V_t = \mathrm{Var}(\alpha_t|Y_n)$, together with other related smoothing results for the transition equation, are computed by using the standard equations (4.39) and (4.43) with

$$a_t = a_{t,1}, \qquad P_t = P_{t,1}, \qquad r_{t-1} = r_{t,0}, \qquad N_{t-1} = N_{t,0}.$$

Finally, the smoothed estimators for the observation disturbances $\varepsilon_{t,i}$ of (6.10) follow directly from our approach and are given by

$$\begin{aligned} \widehat{\varepsilon}_{t,i} &= \sigma^2_{t,i}F_{t,i}^{-1}(v_{t,i} - K'_{t,i}r_{t,i}), \\ \mathrm{Var}(\widehat{\varepsilon}_{t,i}) &= \sigma^4_{t,i}F_{t,i}^{-2}(F_{t,i} + K'_{t,i}N_{t,i}K_{t,i}). \end{aligned}$$

Since the simulation smoothers of Subsection 4.9.1 and 4.9.2 rely fully on the application of the Kalman filter and smoother, the univariate treatment of multivariate observations does not have further consequences for simulation smoothing.

6.4.3 Correlation between observation equations

For the case where H_t is not diagonal, the univariate representation of the state space model (6.10) does not apply due to the correlations between the $\varepsilon_{t,i}$'s. In this situation we can pursue two different approaches. First, we can put the disturbance vector ε_t into the state vector. For the observation equation of (3.1) define

$$\bar{\alpha}_t = \begin{pmatrix} \alpha_t \\ \varepsilon_t \end{pmatrix}, \qquad \bar{Z}_t = \begin{pmatrix} Z_t & I_{P_t} \end{pmatrix},$$

and for the state equation define

$$\bar{\eta}_t = \begin{pmatrix} \eta_t \\ \varepsilon_t \end{pmatrix}, \qquad \bar{T}_t = \begin{pmatrix} T_t & 0 \\ 0 & 0 \end{pmatrix},$$
$$\bar{R}_t = \begin{pmatrix} R_t & 0 \\ 0 & I_{P_t} \end{pmatrix}, \qquad \bar{Q}_t = \begin{pmatrix} Q_t & 0 \\ 0 & H_t \end{pmatrix},$$

leading to

$$y_t = \bar{Z}_t \bar{\alpha}_t, \qquad \bar{\alpha}_{t+1} = \bar{T}_t \bar{\alpha}_t + \bar{R}_t \bar{\eta}_t, \qquad \bar{\eta}_t \sim \mathrm{N}(0, \bar{Q}_t),$$

for $t = 1, \ldots, n$. We then proceed with the same technique as for the case where H_t is diagonal by treating each element of the observation vector individually. The second approach is to transform the observations. In the case where H_t is not diagonal, we diagonalise it by the Cholesky decomposition

$$H_t = C_t H_t^* C_t',$$

where H_t^* is diagonal and C_t is lower triangular with ones on the diagonal. By transforming the observations, we obtain the observation equation

$$y_t^* = Z_t^* \alpha_t + \varepsilon_t^*, \qquad \varepsilon_t^* \sim \mathrm{N}(0, H_t^*),$$

where $y_t^* = C_t^{-1} y_t$, $Z_t^* = C_t^{-1} Z_t$ and $\varepsilon_t^* = C_t^{-1} \varepsilon_t$. Since C_t is a lower triangular matrix, it is easy to compute its inverse. The state vector is not affected by the transformation. Since the elements of ε_t^* are independent we can treat the series y_t^* as a univariate series in the above way.

These two approaches for correlated observation disturbances are complementary. The first method has the drawback that the state vector can become large. The second method is illustrated in Subsection 6.4.5 where we show that simultaneously transforming the state vector can also be convenient.

6.4.4 Computational efficiency

The main motivation for this 'univariate' approach to filtering and smoothing for multivariate state space models is computational efficiency. This approach

avoids the inversion of matrix F_t and two matrix multiplications. Also, the implementation of the recursions is more straightforward. Table 6.1 shows that the percentage savings in the number multiplications for filtering using the univariate approach compared to the standard approach are considerable. The calculations concerning the transition are not taken into account in the calculations for this table because matrix T_t is usually sparse with most elements equal to zero or unity.

Table 6.2 presents the considerable percentage savings in the number of multiplications for state smoothing compared to the standard multivariate approach. Again, the computations involving the transition matrix T_t are not taken into account in compiling these figures.

6.4.5 Illustration: vector splines

We now consider the application of the univariate approach to vector splines. The generalisation of smoothing splines of Hastie and Tibshirani (1990) to the multivariate case is considered by Fessler (1991) and Yee and Wild (1996). The vector spline model is given by

$$y_i = \theta(t_i) + \varepsilon_i, \qquad \mathrm{E}(\varepsilon_i) = 0, \qquad \mathrm{Var}(\varepsilon_i) = \Sigma_i, \qquad i = 1, \ldots, n,$$

where y_i is a $p \times 1$ vector response at scalar t_i, $\theta(\cdot)$ is an arbitrary smooth vector function and errors ε_i are mutually uncorrelated. The variance matrix Σ_i is assumed to be known and is usually constant for varying i. This is a

Table 6.1 Percentage savings for filtering using univariate approach.

State dim.	Obs. dim.	$p = 1$	$p = 2$	$p = 3$	$p = 5$	$p = 10$	$p = 20$
$m = 1$		0	39	61	81	94	98
$m = 2$		0	27	47	69	89	97
$m = 3$		0	21	38	60	83	95
$m = 5$		0	15	27	47	73	90
$m = 10$		0	8	16	30	54	78
$m = 20$		0	5	9	17	35	58

Table 6.2 Percentage savings for smoothing using univariate approach.

State dim.	Obs. dim.	$p = 1$	$p = 2$	$p = 3$	$p = 5$	$p = 10$	$p = 20$
$m = 1$		0	27	43	60	77	87
$m = 2$		0	22	36	53	72	84
$m = 3$		0	19	32	48	68	81
$m = 5$		0	14	25	40	60	76
$m = 10$		0	9	16	28	47	65
$m = 20$		0	5	10	18	33	51

generalisation of the univariate problem considered in Subsection 3.9.2. The standard method of estimating the smooth vector function is by minimising the generalised least squares criterion

$$\sum_{i=1}^{n} \{y_i - \theta(t_i)\}' \Sigma_i^{-1} \{y_i - \theta(t_i)\} + \sum_{j=1}^{p} \lambda_j \int \theta_j''(t)^2 \, dt,$$

where the non-negative smoothing parameter λ_j determines the smoothness of the jth smooth function $\theta_j(\cdot)$ of vector $\theta(\cdot)$ for $j = 1, \ldots, p$. Note that $t_{i+1} > t_i$ for $i = 1, \ldots, n-1$ and $\theta_j''(t)$ denotes the second derivative of $\theta_j(t)$ with respect to t. In the same way as for the univariate case in (3.41), we use the discrete model

$$\begin{aligned} y_i &= \mu_i + \varepsilon_i, \\ \mu_{i+1} &= \mu_i + \delta_i \nu_i + \eta_i, \qquad \mathrm{Var}(\eta_i) = \frac{\delta_i^3}{3}\Lambda, \\ \nu_{i+1} &= \nu_i + \zeta_i, \qquad \mathrm{Var}(\zeta_i) = \delta_i \Lambda, \qquad \mathrm{Cov}(\eta_i, \zeta_i) = \frac{\delta_i^2}{2}\Lambda, \end{aligned}$$

with vector $\mu_i = \theta(t_i)$, scalar $\delta_i = t_{i+1} - t_i$ and diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$. This model is the multivariate extension of the continuous-time representation of the integrated random walk model that is introduced in Subsection 3.2.1; see Harvey (1989, p. 487). In the case of $\Sigma_i = \Sigma$ and with the Cholesky decomposition $\Sigma = CDC'$ where matrix C is lower triangular and matrix D is diagonal, we obtain the transformed model

$$\begin{aligned} y_i^* &= \mu_i^* + \varepsilon_i^*, \\ \mu_{i+1}^* &= \mu_i^* + \delta_i \nu_i^* + \eta_i^*, \qquad \mathrm{Var}(\eta_i) = \frac{\delta_i^3}{3}Q, \\ \nu_{i+1}^* &= \nu_i^* + \zeta_i^*, \qquad \mathrm{Var}(\zeta_i) = \delta_i Q, \qquad \mathrm{Cov}(\eta_i, \zeta_i) = \frac{\delta_i^2}{2}Q, \end{aligned}$$

with $y_i^* = C^{-1} y_i$ and $\mathrm{Var}(\varepsilon_i^*) = D$ where we have used (3.42). Furthermore, we have $\mu_i^* = C^{-1}\mu_i$, $\nu_i^* = C^{-1}\nu_i$ and $Q = C^{-1}\Lambda C'^{-1}$. The Kalman filter smoother algorithm provides the fitted smoothing spline. The untransformed model and the transformed model can both be handled by the univariate strategy of filtering and smoothing discussed in this section. The advantage of the transformed model is that ε_i^* can be excluded from the state vector which is not possible for the untransformed model because $\mathrm{Var}(\varepsilon_i) = \Sigma_i$ is not necessarily diagonal; see the discussion in Subsection 6.4.3.

The percentage computational saving of the univariate approach for spline smoothing depends on the size p. The state vector dimension for the transformed model is $m = 2p$ so the percentage saving in computing for filtering is 30 if $p = 5$ and it is 35 if $p = 10$; see Table 6.1. The percentages for smoothing are 28 and 33, respectively; see Table 6.2.

6.5 Collapsing large observation vectors

6.5.1 Introduction

In cases where the dimensionality p of the observation vector y_t is large in relation to the dimensionality m of the state vector in the state space model (3.1), the Kalman filter may become computationally onerous. In particular, the computation of the inverse of the $p \times p$ innovation variance matrix, F_t^{-1} in the Kalman filter (4.24) for each t can become a computational burden. The univariate treatment of the Kalman filter, as discussed in the previous section, can alleviate a significant part of this computational problem since we update the state vector estimate for each individual element in y_t and hence the innovation variance is a scalar. Nevertheless, the computations required to update the state estimate for each element of the observation vector can become burdensome when p is as large as, say, 500.

An alternative way to deal with the problem is to adopt the well-known matrix identity for the inverse of F_t,

$$F_t^{-1} = (Z_t P_t Z_t' + H_t)^{-1} = H_t^{-1} - H_t^{-1} Z_t \left(P_t^{-1} + Z_t' H_t^{-1} Z_t\right)^{-1} Z_t' H_t^{-1}, \quad (6.16)$$

which is valid when H_t and P_t are both nonsingular matrices; see, for example, Problem 2.9 of Chapter 1 in Rao (1973). Although H_t has the same large dimensionality as F_t, we can expect that matrix H_t has a convenient structure; for example, when H_t is a diagonal matrix. In cases where p is much larger than m, computing the inverse of P_t in (6.16) should be a much lighter operation than inverting F_t directly; for example, when, say, $p = 100$ and $m = 5$. Such dimensionalities are typical in the case of dynamic factors models which are discussed in Section 3.7. The drawback of using (6.16) for the computation of F_t^{-1} is the set of conditions required for H_t and P_t; they must be both nonsingular and H_t must have a convenient structure. Many models of interest do not have a nonsingular matrix H_t and/or it cannot be guaranteed that matrix P_t is nonsingular for all t. Furthermore, the univariate treatment cannot be employed once we adopt the identity (6.16) for inverting F_t.

However, a new approach has recently been developed by Jungbacker and Koopman (2008, Section 2) that collapses the original $p \times 1$ observation vector into a new observation vector of order $m \times 1$. The new observation equation can then be combined with the original state equation to provide a valid state space model. The Kalman filter and smoother for this model produces estimates which are identical to those produced by the original model. It is evident when p is very large relative to m, the use of this approach can lead to substantial computational savings; for example, when the analysis is applied to high-dimensional dynamic factor models where dimensionality of y_t can be as large as 500. Details of the approach are given in the next subsection together with illustrations.

6.5.2 Collapse by transformation

The idea behind collapsing by transformation is to transform the observation vector into a pair of vectors y_t^* and y_t^+ such that y_t^* is $m \times 1$ and y_t^+ is $(p-m) \times 1$, y_t^* depends on α_t and y_t^+ does not depend on α_t and y_t^* and y_t^+ are independent given α_t.We then replace the observation equation for y_t in the original state space model by a new observation equation for y_t^*; this gives a smaller model for analysis. We say that we have *collapsed* y_t into y_t^*.

Let A_t^* be the *projection matrix* $\left(Z_t' H_t^{-1} Z_t\right)^{-1} Z_t' H_t^{-1}$ and let $y_t^* = A_t^* y_t$. Then y_t^* is the generalised least squares estimate of α_t in the conditional distribution of α_t given y_t. Let $A_t^+ = B_t(I_p - Z_t A_t^*)$ where $(p-m) \times p$ matrix B_t is chosen such that A_t^+ has full rank $p-m$. Methods are available for calculating values of B_t with this property, but as will be seen below, we do not need to have an explicit value for B_t. It follows that $A_t^* Z_t = I_p$ and $A_t^+ Z_t = 0$. The original observation equation is given by

$$y_t = Z_t \alpha_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathrm{N}(0, H_t), \tag{6.17}$$

and the observation equation for the transformed variables is given by

$$\begin{pmatrix} y_t^* \\ y_t^+ \end{pmatrix} = \begin{bmatrix} A_t^* \\ A_t^+ \end{bmatrix} y_t = \begin{pmatrix} \alpha_t \\ 0 \end{pmatrix} + \begin{pmatrix} \varepsilon_t^* \\ \varepsilon_t^+ \end{pmatrix},$$

where $y_t^+ = A_t^+ y_t$, $\varepsilon_t^* = A_t^* \varepsilon_t$ and $\varepsilon_t^+ = A_t^+ \varepsilon_t$, for $t = 1, \ldots, n$. Since

$$\mathrm{Cov}(\varepsilon_t^*, \varepsilon_t^+) = \mathrm{E}(\varepsilon_t^* \varepsilon_t^{+\prime}) = A_t^* H_t (I_p - A_t^{*\prime} Z_t') B_t' = (A_t^* - A_t^*) H_t B_t' = 0,$$

the model equations for the transformed observations are given by

$$\begin{array}{rrcl} y_t^* = \alpha_t + \varepsilon_t^*, & \varepsilon_t^* & \sim & \mathrm{N}(0, H_t^*), \\ y_t^+ = \varepsilon_t^+, & \varepsilon_t^+ & \sim & \mathrm{N}(0, H_t^+), \end{array}$$

where ε_t^* and ε_t^+ are independent with variance matrices $H_t^* = A_t^* H_t A_t^{*\prime}$ and $H_t^+ = A_t^+ H_t A_t^{+\prime}$, respectively. All information about α_t is contained in the first equation and therefore we can discard the second equation and we can replace the original state space model by the collapsed model

$$\begin{array}{rrcl} y_t^* = \alpha_t + \varepsilon_t^*, & \varepsilon_t^* & \sim & \mathrm{N}(0, H_t^*), \\ \alpha_{t+1} = T_t \alpha_t + R_t \eta_t, & \eta_t & \sim & \mathrm{N}(0, Q_t), \end{array} \tag{6.18}$$

for $t = 1, \ldots, n$ where y_t^* is $m \times 1$. Analysis of this model by the methods of Chapter 4 gives exactly the same results for Kalman filtering and smoothing as analysis of the original model.

When p is substantially larger than m, the use of model (6.18) can therefore result in a significant saving in computation. When the original model is time-invariant in which the system matrices are constant over time, collapsing is particularly advantageous since we only need to compute the matrices A_t^* and H_t^* once.

6.5.3 A generalisation of collapsing by transformation

We now present a generalisation of the method of collapsing described in Subsection 6.5.2. Let

$$\bar{A}_t^* = C_t Z_t' H_t^{-1}, \tag{6.19}$$

where C_t is an arbitrary nonsingular and nonrandom matrix and let

$$\bar{A}_t^+ = B_t \left[I_p - Z_t \left(Z_t' H_t^{-1} Z_t \right)^{-1} Z_t' H_t^{-1} \right],$$

where B_t is the same as in Subsection 6.5.2. It follows that $\bar{A}_t^*$ is a generalisation of A_t^* of Subsection 6.5.2 where $\bar{A}_t^* = A_t^*$ when $C_t = \left(Z_t' H_t^{-1} Z_t \right)^{-1}$. We have $\bar{A}_t^+ Z_t = 0$ and consider the transformation

$$\begin{pmatrix} \bar{y}_t^* \\ \bar{y}_t^+ \end{pmatrix} = \begin{bmatrix} \bar{A}_t^* \\ \bar{A}_t^+ \end{bmatrix} y_t = \begin{pmatrix} Z_t^* \alpha_t \\ 0 \end{pmatrix} + \begin{pmatrix} \bar{\varepsilon}_t^* \\ \bar{\varepsilon}_t^+ \end{pmatrix},$$

where $Z_t^* = \bar{A}_t^* Z_t = C_t Z_t' H_t^{-1} Z_t$, $\bar{\varepsilon}_t^* = \bar{A}_t^* \varepsilon_t$ and $\bar{\varepsilon}_t^+ = \bar{A}_t^+ \varepsilon_t$, for $t = 1, \ldots, n$. Since,

$$\begin{aligned} \mathrm{E}(\bar{\varepsilon}_t^*, \bar{\varepsilon}_t^{+\prime}) &= \bar{A}_t^* H_t \left[I_p - H_t^{-1} Z_t \left(Z_t' H_t^{-1} Z_t \right)^{-1} Z_t' \right] B_t' \\ &= \left[C_t Z_t' - C_t Z_t' H_t^{-1} Z_t \left(Z_t' H_t^{-1} Z_t \right)^{-1} Z_t' \right] \;=\; 0, \end{aligned}$$

$\bar{\varepsilon}_t^*$ and $\bar{\varepsilon}_t^+$ are independent. We therefore can replace the original $p \times 1$ observation equation by the $m \times 1$ collapsed observation equation

$$\bar{y}_t^* = Z_t^* \alpha_t + \bar{\varepsilon}_t^*, \qquad \bar{\varepsilon}_t^* \sim \mathrm{N}(0, \bar{H}_t^*),$$

for $t = 1, \ldots, n$ where $\bar{H}_t^* = C_t Z_t' H_t^{-1} Z_t C_t'$. In particular, if we take $C_t = I_p$ then $\bar{A}_t = Z_t' H_t^{-1}$ and $\bar{H}_t^* = Z_t' H_t^{-1} Z_t$. This shows that the transformation is flexible and that C_t can be chosen for convenience (within the set of nonsingular matrices) depending on the details of the particular state space model under consideration.

A particularly convenient choice for C_t is to take $C_t' C_t = \left(Z_t' H_t^{-1} Z_t \right)^{-1}$ such that

$$Z_t^* = \bar{A}_t^* Z_t = C_t Z_t' H_t^{-1} Z_t = C_t (C_t' C_t)^{-1} = C_t C_t^{-1} C_t'^{-1} = C_t'^{-1},$$

and

$$\bar{H}_t^* = C_t Z_t' H_t^{-1} Z_t C_t' = C_t (C_t' C_t)^{-1} C_t' = I_p.$$

Kalman filtering and smoothing applied to the collapsed model for $\bar{y}_t^*$ with $\bar{H}_t^* = I_p$ can directly be based on the univariate treatment of Section 6.4.

The results into this section were first obtained by Jungbacker and Koopman (2008) in a context more general than considered here. For example, they show how the collapse of y_t can take place to the lowest possible dimension for $\bar{y}_t^*$.

Table 6.3 Percentage savings for Kalman filtering.

State dim.	Obs. dim.	$p = 10$	$p = 50$	$p = 100$	$p = 250$	$p = 500$
$m = 1$		50	82	85	89	92
$m = 5$		23	79	87	93	94
$m = 10$		–	68	82	92	95
$m = 25$		–	33	60	81	90
$m = 50$		–	–	33	67	81

6.5.4 Computational efficiency

To obtain some insight into the size of the computational gains due to collapsing, we apply the Kalman filter for different values of p and m, with and without collapsing the observation vector y_t to y_t^*. The percentage savings achieved by the Kalman filter for the collapsed observation vector compared to the Kalman filter for the original observation vector are reported in Table 6.3 for different dimensions p and m. The computations associated with the transformation are taken into account as well as the computations associated with the transition equation. The percentage savings of the collapsing method are considerable by any means but particularly when $p >> m$.

6.6 Filtering and smoothing under linear restrictions

We now consider how to carry out filtering and smoothing subject to a set of time-varying linear restrictions on the state vector of the form

$$R_t^* \alpha_t = r_t^*, \qquad t = 1, \ldots, n, \tag{6.20}$$

where the matrix R_t^* and the vector r_t^* are known and where the number of rows in R_t^* can vary with t. Although linear restrictions on the state vector can often be easily dealt with by respecifying the elements of the state vector, an alternative is to proceed as follows. To impose the restrictions (6.20) we augment the obervation equation as

$$\begin{pmatrix} y_t \\ r_t^* \end{pmatrix} = \begin{bmatrix} Z_t \\ R_t^* \end{bmatrix} \alpha_t + \begin{pmatrix} \varepsilon_t \\ 0 \end{pmatrix}, \qquad t = 1, \ldots, n. \tag{6.21}$$

For this augmented model, filtering and smoothing will produce estimates a_t and $\hat{\alpha}_t$ which are subject to the restrictions $R_t^* a_t = r_t^*$ and $R_t^* \hat{\alpha}_t = r_t^*$; for a discussion of this procedure, see Doran (1992). Equation (6.21) represents a multivariate model whether y_t is univariate or not. This can, however, be treated by a univariate approach to filtering and smoothing based on the devices discussed in Section 6.4.

6.7 Computer packages for state space methods

6.7.1 Introduction

The use of state space methods in empirical work is enhanced with the implementations of state space methods in statistical and econometric computer packages. The program *STAMP* (Structural Time Series Analyser, Modeller and Predictor) firstly appeared in 1982 and was developed by Simon Peters and Andrew Harvey at the London School of Economics. It has been widely acknowledged as one of the first statistical software packages primarily based on state space methods. The *STAMP* program includes options for maximum likelihood estimation, diagnostic checking, signal extraction and forecasting procedures applied to the structural time series models which we discuss in Section 3.2 and 3.3. *STAMP* has made these functions available in a user-friendly menu system. The more recent versions of the program include extended facilities for analysing univariate and multivariate models, automatic outlier and break detection, and forecasting. The current version is developed by Koopman, Harvey, Doornik and Shephard (2010) and the latest information can be obtained from the Internet at `http://stamp-software.com/`.

Software implementations of state space methods have increased with the progress in computing capabilities and the modernisation of computer software generally. Many well-known statistical and econometric software packages currently have options for the use of state space methods. A complete overview of available computer packages with options for using state space methods is provided in a special issue of the *Journal of Statistical Software* and is edited by Commandeur, Koopman and Ooms (2011). It shows that many software tools for state space models are currently available for the analysis of time series. An example of a computer package for state space analysis is *SsfPack* of Koopman, Shephard and Doornik (1999, 2008) which we discuss next in more detail.

6.7.2 *SsfPack*

SsfPack is a suite of C routines for carrying out computations involving the statistical analysis of univariate and multivariate models in state space form. *SsfPack* allows for a range of different state space forms from simple time-invariant models to complicated time-varying models. Functions are specified which put standard models such as ARIMA and spline models into state space form. Routines are available for filtering, smoothing and simulation smoothing. Ready-to-use functions are provided for standard tasks such as likelihood evaluation, forecasting and signal extraction. The headers of these routines are documented by Koopman, Shephard and Doornik (1999); a more detailed documentation of the updated version 3.0 is presented in Koopman, Shephard and Doornik (2008). *SsfPack* can be used for implementing, fitting and analysing linear, Gaussian, nonlinear and/or non-Gaussian state space models relevant to many areas of time series analysis. A Gaussian illustration is given in Subsection 6.7.5.

6.7.3 The basic *SsfPack* functions

A list of *SsfPack* functions is given in Table 6.4. The functions are grouped into functions which put specific univariate models into state space form, functions which perform the basic filtering and smoothing operations and functions which execute specific important tasks for state space analysis such as likelihood evaluation. Table 6.4 consists of three columns : the first column contains the function name, the second column gives reference to the section number of the *SsfPack* documentation of Koopman, Shephard and Doornik (1999) and the third column describes the function with references to equation or section numbers in this book. This part of the package is freely available from `http://www.ssfpack.com`.

6.7.4 The extended *SsfPack* functions

In version 3.0 of *SsfPack* by Koopman, Shephard and Doornik (2008), further modifications of the computer routines are developed for the exact initial Kalman filter of Section 5.2, the exact initial state smoothing of Section 5.3, the exact initial disturbance smoothing of Section 5.4 and the computation of the diffuse likelihood functions defined in Section 7.2. The additional functions are listed in Table 6.5 in the same way as in Table 6.4.

Table 6.4 Functions in *SsfPack Basic* with section reference to documentation and with short description.

Models in state space form		
`AddSsfReg`	§3.3	adds regression effects (3.30) to state space.
`GetSsfArma`	§3.1	puts ARMA model (3.18) in state space.
`GetSsfReg`	§3.3	puts regression model (3.30) in state space.
`GetSsfSpline`	§3.4	puts cubic spline model (3.46) in state space.
`GetSsfStsm`	§3.2	puts structural time series model of §3.2 in state space.
`SsfCombine`	§6.2	combines system matrices of two models.
`SsfCombineSym`	§6.2	combines symmetric system matrices of two models.
General state space algorithms		
`KalmanFil`	§4.3	provides output of the Kalman filter in §4.3.2.
`KalmanSmo`	§4.4	provides output of the basic smoothing algorithm in §4.4.4.
`SimSmoDraw`	§4.5	provides a simulated sample.
`SimSmoWgt`	§4.5	provides output of the simulation smoother.
Ready-to-use functions		
`SsfCondDens`	§4.6	provides mean or a draw from the conditional density (4.81).
`SsfLik`	§5.1	provides loglikelihood function (7.4).
`SsfLikConc`	§5.1	provides concentrated loglikelihood function.
`SsfLikSco`	§5.1	provides score vector information (2.63).
`SsfMomentEst`	§5.2	provides output from prediction, forecasting and §5.3 smoothing.
`SsfRecursion`	§4.2	provides output of the state space recursion (3.1).

Table 6.5 Additional functions in *SsfPack Extended* with section reference to documentation and with short description.

General state space algorithms		
`KalmanInit`	§8.2.2	provides output of the exact initial
`KalmanFilEx`	§8.2.2	Kalman filter in §5.2.
`KalmanSmoEx`	§8.2.4	provides output of exact initial smoothing.
`KalmanFilSmoMeanEx`	§9.2	provides output to facilitate the augmented Kalman filter and smoother of §5.7.
Ready-to-use functions		
`SsfCondDensEx`	§9.4	provides the mean from the conditional density based on §§5.2 and 5.3.
`SsfLikEx`	§9.1.2	provides diffuse loglikelihood function (7.4).
`SsfLikConcEx`	§9.1.2	provides concentrated diffuse loglikelihood function.
`SsfLikScoEx`	§9.1.3	provides score vector information, see §7.3.3.
`SsfMomentEstEx`	§9.4	provides output from prediction, forecasting and based on §§5.2 and 5.3.
`SsfForecast`	§9.5	provides output of Kalman filter for forecasting, see §4.11.
`SsfWeights`	§9.6	provides output for filter and smoothing weights, see §4.8.

For all tasks involving the (exact initial) Kalman filter and smoother, the univariate treatment of multivariate series as described in Section 6.4 is implemented for the *SsfPack Extended* functions when the state space form represents a multivariate time series model. Furthermore, all computations involving unity and zero values are treated specifically such that they can handle sparse system matrices in a computationally efficient manner. These amendments lead to computationally faster and numerically more stable calculations. We will not discuss the *SsfPack* functions further but an example of *Ox* code, which utilises the link with the *SsfPack* library, is given in Subsection 6.7.5. This part of the package is a commercial product; more information can be obtained from `http://www.ssfpack.com`.

6.7.5 Illustration: spline smoothing

In the *Ox* code below we consider the continuous spline smoothing problem which aims at minimising (3.46) for a given value of λ. The aim of the program is to fit a spline through the Nile time series of Chapter 2 (see Subsection 2.2.5 for details). To illustrate that the *SsfPack* functions can deal with missing observations we have treated two parts of the data set as missing. The continuous spline model is easily put in state space form using the function `GetSsfSpline`. The smoothing parameter λ is chosen to take the value 2500 (the function requires the input of $\lambda^{-1} = 0.004$). We need to compute an estimator for the unknown scalar value of

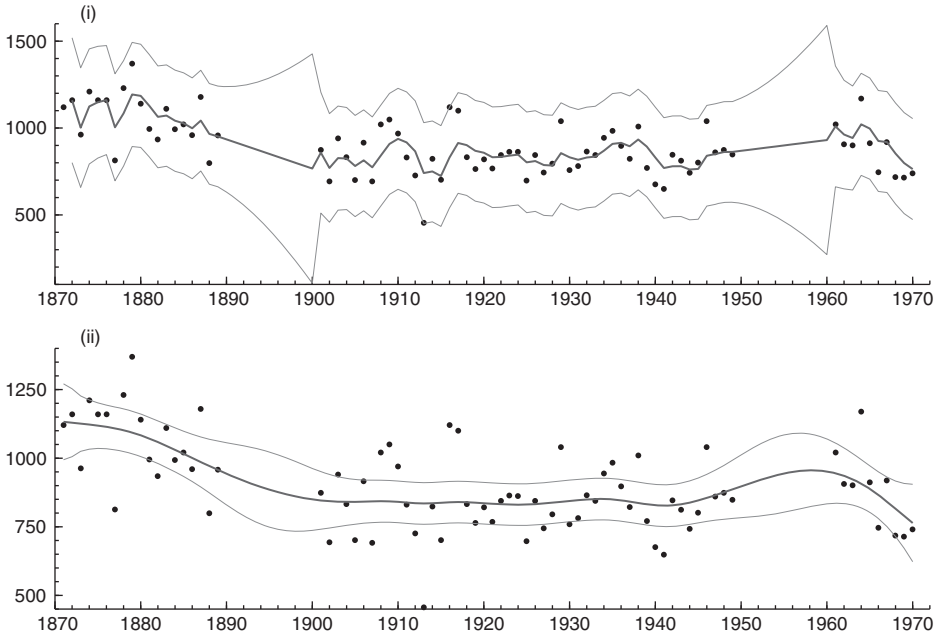

Fig. 6.1 Output of *Ox* program `spline.ox`: (i) Nile data (dots) with filtered estimate of spline function with 95% confidence interval; (ii) Nile data (dots) with smoothed estimate of spline function with 95% confidence interval.

σ_ζ^2 in (3.47) which can be obtained using the function `SsfLikEx`. After rescaling, the estimated spline function using filtering (`ST_FIL`) and smoothing (`ST_SMO`) is computed using the function `SsfMomentEst`. The output is presented in Fig. 6.1 and shows the filtered and smoothed estimates of the spline function. The two diagrams illustrate the point that filtering can be interpreted as extrapolation and that when observations are missing, smoothing is in fact interpolation.

```
 spline.ox
#include <oxstd.h>
#include <oxdraw.h>
#include <oxfloat.h>
#include <packages/ssfpack/ssfpack.h>
main()
{
  decl mdelta, mphi, momega, msigma, myt, mfil, msmo, cm, dlik, dvar;

  myt = loadmat("Nile.dat")';
  myt[][1890-1871:1900-1871] = M_NAN;  // set 1890..1900 to missing
  myt[][1950-1871:1960-1871] = M_NAN;  // set 1950..1960 to missing

  GetSsfSpline(0.004, <>, &mphi, &momega, &msigma); // SSF for spline
  SsfLikEx(&dlik, &dvar, myt, mphi, momega);         // need dVar
```

```
  cm = columns(mphi);                        // dimension of state
  momega *= dvar;                   // set correct scale of Omega
  SsfMomentEstEx(ST_FIL, &mfil,  myt, mphi, momega);
  SsfMomentEstEx(ST_SMO, &msmo, myt, mphi, momega);

  // note that first filtered estimator does not exist
  DrawTMatrix(0, myt, {"Nile"}, 1871, 1, 1);
  DrawTMatrix(0, mfil[cm][1:], {"Pred +/- 2SE"}, 1872, 1, 1, 0, 3);
  DrawZ(sqrt(mfil[2*cm+1][1:]), "", ZMODE_BAND, 2.0, 14);
  DrawTMatrix(1, myt, {"Nile"}, 1871, 1, 1);
  DrawTMatrix(1, msmo[cm][], {"Smooth +/- 2SE"}, 1871, 1, 1, 0, 3);
  DrawZ(sqrt(msmo[2*cm+1][]), "", ZMODE_BAND, 2.0, 14);
  ShowDrawWindow();
}
```