CPB Netherlands Bureau for Economic Policy Analysis

CPB Discussion Paper | 359

Exact Nonlinear and Non-Gaussian Kalman Smoother for State Space Models with Implicit Functions and Equality Constraints

Joris de Wind

Exact Nonlinear and Non-Gaussian Kalman Smoother for State Space Models with Implicit Functions and Equality Constraints*

Joris de Wind†

September 2017

Abstract

In this paper, I present a novel implementation of the exact nonlinear and non-Gaussian Kalman smoother that can also deal with implicit functions in the measurement and/or state equations as well as equality constraints. My approach has the additional advantage that it can be fully automated, on the basis of which I have developed a toolbox that can handle a wide class of discrete-time state space models. The toolbox is documented in an accompanying paper, while the technical details are presented in the current one.

1 Introduction

The Kalman filter and smoother are tools that are widely used in many applications ranging from engineering to econometrics, with common engineering applications such as guidance, navigation and control, and common econometrics applications in time series analysis such as trend-cycle decompositions and conditional forecasting. If a system is linear and Gaussian, the Kalman filter will deliver the best possible estimate of the current state of the system, while the Kalman smoother will update the past estimates in an optimal way. However, we are often also interested in systems that are nonlinear and/or non-Gaussian, for which we need a generalization of the Kalman filter and smoother.

In this paper, I present a new way to generalize the Kalman smoother to allow for general nonlinearities and non-Gaussian disturbances, implicit functions in the measurement and/or

*I am very grateful to Michal Andrle, Junior Maih, and Rob van Harrevelt for their comments and suggestions. Views expressed in this paper are my own and do not necessarily reflect those of the CPB.

†CPB Netherlands Bureau for Economic Policy Analysis, email: jorisdewind@gmail.com

state equations, and equations without disturbances (e.g. equality constraints or identities). My approach is related to the iterated Kalman smoother of Bell (1994) as well as Durbin and Koopman (1992), who basically employ the Kalman filter and smoother iteratively to solve a large-scale optimization problem for finding the mode of the state of the system. In particular, the iterative Kalman smoother exploits the fact that in the linear case the large-scale optimization problem is solved analytically by the Kalman filter and smoother. In the current paper, however, I exploit the equivalence between the large-scale optimization problem and the structure of dynamic models with forward-looking variables, which can be solved efficiently by the so-called stacked-time algorithm.[1] Moreover, relative to the aforementioned papers, I have also improved upon the setup of the optimization problem, wherefore my setup allows for a much wider class of state space models and my approach is also easier to implement.

As a matter of fact, my approach has the additional advantage that it can be fully automated, which enabled me to develop a toolbox that can handle a wide class of discrete-time state space models. The toolbox is documented in an accompanying paper (De Wind, 2017), while the technical details are presented in the next section.[2] The current paper also contains a separate section with two examples, one of which is a conditional forecasting exercise that is related to the 'observation procedure' of Sandee, Don, and van den Berg (1984).

2 Large-scale optimization problem for finding the mode of the state of the system

This section starts with a subsection about the linear case to set out some notational conventions and to build intuition. The results naturally generalize to the nonlinear case, which is presented in a separate subsection. This section concludes with a subsection with implementation details for the general nonlinear and non-Gaussian case. The implementation details are important in order to deal with implicit functions and equality constraints (i.e. equations without disturbances). It will become lucid from subsection 2.3 that the implementation details are also very important to minimize the programming burden.

2.1 Linear case

The goal is to find the mode of the state vector of a state space system, where the parameters of the system are known. Following the notation of Durbin and Koopman (2001), consider

[1] The stacked-time algorithm is explained in e.g. Juillard (1996) and Hollinger (1996). I provide a brief description of the stacked-time algorithm in the appendix.

[2] The toolbox is freely available at www.github.com/jorisdewind but requires a Matlab licence (though the toolbox is likely to work under Octave as well).

the linear Gaussian state space model[3]

$$\begin{aligned} y_t &= Z\alpha_t + \varepsilon_t & \varepsilon_t &\sim N(0, H) & & \\ \alpha_{t+1} &= T\alpha_t + \eta_{t+1} & \eta_{t+1} &\sim N(0, Q) & t &= 1, \ldots, n \\ & & \alpha_1 &\sim N(a_1, P_1) & & \end{aligned} \tag{1}$$

where y_t is a $p \times 1$ vector with observables and α_t is a $m \times 1$ state vector. For notational convenience and without loss of generality, the system matrices Z, H, T, and Q are considered to be time-invariant.[4] The first equation is called the observation or measurement equation and the second one is called the state equation. The state vector α_t captures the underlying dynamics of the state space system, but it is not directly observed.

The purpose of the Kalman filter and smoother is to estimate the state vector α_t based on the vector with observables y_t. The Kalman filter and smoother differ from each other in the amount of conditioning information that is used for the estimation. That is, the Kalman filter estimates the state vector α_t based on the available information up to time t exclusive (y^{t-1}), whereas the Kalman smoother estimates the state vector α_t based on all the available information in the sample period (y^n). The Kalman smoother basically updates the past predictions of the Kalman filter when new (future) information becomes available.

The Kalman filter delivers $a_{t+1} = \mathbb{E}[\alpha_{t+1}|y^t]$ and $P_{t+1} = \mathbb{V}ar[\alpha_{t+1}|y^t]$. The Kalman filter recursions are given by

$$\begin{aligned} \upsilon_t &= y_t - Za_t & F_t &= ZP_tZ' + H & & \\ K_t &= TP_tZ'F_t^{-1} & L_t &= T - K_tZ & t &= 1, \ldots, n \\ a_{t+1} &= Ta_t + K_t\upsilon_t & P_{t+1} &= TP_tL_t' + Q & & \end{aligned} \tag{2}$$

The Kalman smoother delivers $\hat{\alpha}_{t+1} = \mathbb{E}[\alpha_{t+1}|y^n]$ and $V_{t+1} = \mathbb{V}ar[\alpha_{t+1}|y^n]$. The Kalman smoother recursions are given by

$$\begin{aligned} r_{t-1} &= Z'F_t^{-1}\upsilon_t + L_t'r_t & N_{t-1} &= Z'F_t^{-1}Z + L_t'N_tL_t & & \\ \hat{\alpha}_t &= a_t + P_tr_{t-1} & V_t &= P_t - P_tN_{t-1}P_t & t &= n, \ldots, 1 \end{aligned} \tag{3}$$

where it should be noted that the Kalman smoother recursions run backwards, starting from $r_n = 0$ and $V_n = 0$. Moreover, the Kalman smoother can only be run after the Kalman filter.

[3]Note that the timing convention of the vector with state disturbances η_t is shifted relative to Durbin and Koopman (2001).

[4]Moreover, also the selection matrix R of Durbin and Koopman (2001) is considered to be time-invariant and set to the identity matrix, but this can easily be generalized.

The Kalman filter and smoother recursions can be derived in many ways, see e.g. Durbin and Koopman (2001). A particularly appealing approach can be found in Bell (1994), whose approach generalizes very well to the nonlinear case. The key equation is the decomposition of the log-likelihood function

$$\log L\left(y^n|\Psi\right) = \log L\left(y^n, \alpha^n|\Psi\right) - \log L\left(\alpha^n|y^n, \Psi\right) \tag{4}$$

where the first term on the right-hand side is the complete data log-likelihood function and the second term on the right-hand side is the conditional density of the state vector. The parameters of the various system matrices are stacked in the vector Ψ.

It can be observed directly from decomposition (4) that the difference between the conditional density of the state vector and the complete data log-likelihood function does not depend on α^n, which implies that the partial derivatives with respect to α^n are equal to each other. Therefore, finding the mode of the state vector $\hat{\alpha}^n$ is equivalent to maximizing the complete data log-likelihood function, that is

$$\left.\frac{\partial \log L\left(\alpha^n|y^n, \Psi\right)}{\partial \alpha^n}\right|_{\alpha^n=\hat{\alpha}^n} = \left.\frac{\partial \log L\left(y^n, \alpha^n|\Psi\right)}{\partial \alpha^n}\right|_{\alpha^n=\hat{\alpha}^n} = 0 \tag{5}$$

This property can be exploited since the partial derivatives of the complete data log-likelihood function (as opposed to the conditional density of the state vector) are very easy to calculate. In particular, the complete data log-likelihood function can be written as

$$\begin{aligned} \log L\left(y^n, \alpha^n|\Psi\right) &= \text{constant} - \frac{1}{2}\left(\alpha_1 - a_1\right)' P_1^{-1}\left(\alpha_1 - a_1\right) \\ &\quad -\frac{1}{2}\sum_{t=1}^{n}\begin{pmatrix} \left(y_t - Z\alpha_t\right)' H^{-1}\left(y_t - Z\alpha_t\right) + \\ \left(\alpha_{t+1} - T\alpha_t\right)' Q^{-1}\left(\alpha_{t+1} - T\alpha_t\right) \end{pmatrix} \end{aligned} \tag{6}$$

and the first-order conditions with respect to α^n are given by[5]

$$\begin{aligned} Z'H^{-1}\left(y_1 - Z\hat{\alpha}_1\right) - P_1^{-1}\left(\hat{\alpha}_1 - a_1\right) + T'Q^{-1}\left(\hat{\alpha}_2 - T\hat{\alpha}_1\right) &= 0 \\ Z'H^{-1}\left(y_t - Z\hat{\alpha}_t\right) - Q^{-1}\left(\hat{\alpha}_t - T\hat{\alpha}_{t-1}\right) + T'Q^{-1}\left(\hat{\alpha}_{t+1} - T\hat{\alpha}_t\right) &= 0 \qquad t = 2, \ldots, n \\ \hat{\alpha}_{n+1} - T\hat{\alpha}_n &= 0 \end{aligned} \tag{7}$$

Bell (1994) has given a proof that the above first-order conditions can be solved analytically by the Kalman filter and smoother recursions (and, of course, it is easy to verify this numerically). In this paper, however, I propose to use the so-called stacked-time algorithm to solve the above first-order conditions, the benefits of which will become apparent from the nonlinear case. The stacked-time algorithm is often used to simulate dynamic models

[5]To be precise, also α_{n+1} is an instrument variable over which the complete data log-likelihood function is maximized.

with forward-looking variables (under the assumption of perfect foresight).[6]

The above first-order conditions constitute a second-order difference equation in $\hat{\alpha}_t$. In order to employ the stacked-time algorithm it is more convenient to rewrite the first-order conditions slightly so as to obtain explicit initial and terminal conditions.

- First, the terminal condition can be expressed explicitly in terms of the underlying state disturbance.
- Second, rather than starting the recursions from the unknown $\hat{\alpha}_1$ it is more convenient to initialize the system with a given $\hat{\alpha}_0$. By appropriately adjusting the state equation for the first period (see below), it is possible to simply set $\hat{\alpha}_0 = a_0 = a_1$.

Altogether, the system of equations can be rewritten as

$$\begin{aligned} \hat{\alpha}_0 &= a_0 \\ Z'H^{-1}\hat{\varepsilon}_t - Q_t^{-1}\hat{\eta}_t + T_{t+1}'Q_{t+1}^{-1}\hat{\eta}_{t+1} &= 0 \qquad t = 1, \ldots, n \\ \hat{\eta}_{n+1} &= 0 \\ y_t - Z\hat{\alpha}_t &= \hat{\varepsilon}_t \\ \hat{\alpha}_t - T_t\hat{\alpha}_{t-1} &= \hat{\eta}_t \end{aligned} \tag{8}$$

where $Q_t = Q$ and $T_t = T$ except for $t = 1$ in which case $Q_t = P_1$ and $T_t = I$.[7] The above system of equations can now simply be solved by the stacked-time algorithm.[8] Of course, it is easy to verify numerically that the Kalman smoother yields the same $\hat{\alpha}^n$.

2.2 Nonlinear case

The key advantage of the approach outlined in the previous subsection is that it does not rely on the linearity or Gaussianity of the state space model. No matter the type of nonlinearity or the distribution of the disturbances, the idea is to equate the partial derivatives of the complete data log-likelihood function to zero and solve the resulting difference equation with the stacked-time algorithm.[9] Although this approach is completely general, I focus in this section on the nonlinear case with additive Gaussian disturbances.

---

[6] See the appendix for a brief introduction to the stacked-time algorithm. A basic understanding of the stacked-time algorithm might help to gain a better intuition for the exact nonlinear and non-Gaussian Kalman smoother developed in this paper.

[7] Note that the time subscripts can be omitted from the Q_{t+1} and T_{t+1} in front of the $\hat{\eta}_{t+1}$ term.

[8] The stacked-time algorithm is implemented in software packages such as Dynare and Troll, see www.dynare.org and www.hendyplan.com/troll-software, respectively. Dynare is a freely available open source Matlab toolbox, while Troll is a commercial software package.

[9] This approach yields the state mode of the system, which in the nonlinear and/or non-Gaussian case is not equal to the state mean. In the remainder of this paper, $\hat{\alpha}^n$ is used to denote the state mode as opposed to the state mean. In the linear case, this distinction was non-existent.

Consider the nonlinear Gaussian state space model

$$\begin{aligned} y_t &= Z(\alpha_t) + \varepsilon_t & \varepsilon_t &\sim N(0, H) & & \\ \alpha_{t+1} &= T(\alpha_t) + \eta_{t+1} & \eta_{t+1} &\sim N(0, Q) & & t = 1, \ldots, n \\ & & \alpha_1 &\sim N(a_1, P_1) & & \end{aligned} \tag{9}$$

where $Z(\cdot)$ and $T(\cdot)$ are now vector-valued functions instead of matrices, i.e. $\mathbb{R}^m \to \mathbb{R}^p$ and $\mathbb{R}^m \to \mathbb{R}^m$, respectively.

In this case, the complete data log-likelihood function can be written as

$$\begin{aligned} \log L(y^n, \alpha^n | \Psi) &= \text{constant} - \frac{1}{2}(\alpha_1 - a_1)' P_1^{-1}(\alpha_1 - a_1) \\ &\quad - \frac{1}{2}\sum_{t=1}^{n} \begin{pmatrix} (y_t - Z(\alpha_t))' H^{-1}(y_t - Z(\alpha_t)) + \\ (\alpha_{t+1} - T(\alpha_t))' Q^{-1}(\alpha_{t+1} - T(\alpha_t)) \end{pmatrix} \end{aligned} \tag{10}$$

and the first-order conditions with respect to α^n are given by[10]

$$\begin{aligned} J_Z'(\hat{\alpha}_1) H^{-1}(y_1 - Z(\hat{\alpha}_1)) - P_1^{-1}(\hat{\alpha}_1 - a_1) + J_T'(\hat{\alpha}_1) Q^{-1}(\hat{\alpha}_2 - T(\hat{\alpha}_1)) &= 0 \\ J_Z'(\hat{\alpha}_t) H^{-1}(y_t - Z(\hat{\alpha}_t)) - Q^{-1}(\hat{\alpha}_t - T(\hat{\alpha}_{t-1})) + J_T'(\hat{\alpha}_t) Q^{-1}(\hat{\alpha}_{t+1} - T(\hat{\alpha}_t)) &= 0 \quad t = 2, \ldots, n \\ \hat{\alpha}_{n+1} - T(\hat{\alpha}_n) &= 0 \end{aligned} \tag{11}$$

where $J_Z(\hat{\alpha}_t)$ is the $p \times m$ Jacobian matrix of $Z(\cdot)$ evaluated at $\alpha_t = \hat{\alpha}_t$ and $J_T(\hat{\alpha}_t)$ is the $m \times m$ Jacobian matrix of $T(\cdot)$ evaluated at $\alpha_t = \hat{\alpha}_t$.

While Durbin and Koopman (1992) and Bell (1994) solve the above first-order conditions iteratively by linearization and applying the standard Kalman filter and smoother recursions at each iteration, my approach avoids the linearization step altogether and solves the system with the stacked-time algorithm.[11]

Like in the previous subsection, in order to employ the stacked-time algorithm it is more convenient to rewrite the above first-order conditions as a second-order difference equation with explicit initial and terminal conditions. In particular, the system of equations can be

[10] Again, to be precise, also α_{n+1} is an instrument variable over which the complete data log-likelihood function is maximized.

[11] I would like to emphasize that this is not the only advantage of my approach. As will become clear from the next subsection, in contrast to the approach of Durbin and Koopman (1992) and Bell (1994), my approach generalizes very nicely to state space models with implicit functions and equality constraints.

rewritten as

$$\begin{aligned} \hat{\alpha}_0 &= a_0 \\ J_Z'\left(\hat{\alpha}_t\right) H^{-1} \hat{\varepsilon}_t - Q_t^{-1} \hat{\eta}_t + J_T'\left(\hat{\alpha}_t\right) Q^{-1} \hat{\eta}_{t+1} &= 0 \qquad t = 1, \ldots, n \\ \hat{\eta}_{n+1} &= 0 \\ y_t - Z\left(\hat{\alpha}_t\right) &= \hat{\varepsilon}_t \\ \hat{\alpha}_t - T_t\left(\hat{\alpha}_{t-1}\right) &= \hat{\eta}_t \end{aligned} \tag{12}$$

with the same conventions as in the linear case, i.e. $Q_t = Q$ and $T_t = T(\cdot)$ except for $t = 1$ in which case $Q_t = P_1$ and $T_t = I$, and furthermore $\hat{\alpha}_0 = a_0 = a_1$. Several examples are given in section 3, but important implementation details are first given in the next subsection.

2.3 Implementation details for the general case

Although the above implementation seems quite natural, it can still be improved upon. In particular, the above implementation does not naturally extend to the case with implicit equations and also not to the case with equations without disturbances (e.g. equality constraints or identities). Therefore, in this subsection, I present a more general implementation, which in fact starts with a constrained optimization problem rather than an unconstrained one. In particular, the idea is to maximize the likelihood of the disturbances subject to the constraints implied by the model.

Consider the nonlinear and non-Gaussian state space model

$$\begin{aligned} Z\left(y_t, \alpha_t, \varepsilon_t\right) &= 0 & \varepsilon_t &\sim p_\varepsilon\left(\varepsilon_t\right) \\ T\left(\alpha_t, \alpha_{t-1}, \eta_t\right) &= 0 & \eta_t &\sim p_\eta\left(\eta_t\right) \qquad t = 1, \ldots, n \\ & & \alpha_0 &\text{ is given} \end{aligned} \tag{13}$$

where the initial condition is already given explicitly. However, if one wishes to account for uncertainty about the initial state of the system, this can easily be incorporated in the above representation by appropriately adjusting the distribution of the state disturbance in the first period.

The idea is to maximize the likelihood of the disturbances subject to the constraints implied by the model, which can be formalized by the following Lagrangian

$$\mathcal{L}\left(\varepsilon^n, \eta^n, \alpha^n, \Lambda_Z^n, \Lambda_T^n\right) = \sum_{t=1}^{n} \begin{pmatrix} \log\left(p_\varepsilon\left(\varepsilon_t\right)\right) + \log\left(p_\eta\left(\eta_t\right)\right) + \\ \Lambda_{Z,t}' Z\left(y_t, \alpha_t, \varepsilon_t\right) + \Lambda_{T,t}' T\left(\alpha_t, \alpha_{t-1}, \eta_t\right) \end{pmatrix} \tag{14}$$

where $\Lambda_{Z,t}$ and $\Lambda_{T,t}$ are vectors with Lagrangian multipliers for the measurement and state equations, respectively. The first-order conditions with respect to $\Theta^n = \{\varepsilon^n, \eta^n, \alpha^n, \Lambda_Z^n, \Lambda_T^n\}$

are given by

$$
\begin{aligned}
\hat{\alpha}_0 &= \alpha_0 \\
\left.\frac{\partial \log\left(p_\varepsilon\left(\varepsilon_t\right)\right)}{\partial \varepsilon_t}\right|_{\Theta^n=\hat{\Theta}^n} + \hat{\Lambda}'_{Z,t} \left.\frac{\partial Z\left(y_t,\alpha_t,\varepsilon_t\right)}{\partial \varepsilon_t}\right|_{\Theta^n=\hat{\Theta}^n} &= 0 \\
\left.\frac{\partial \log\left(p_\eta\left(\eta_t\right)\right)}{\partial \eta_t}\right|_{\Theta^n=\hat{\Theta}^n} + \hat{\Lambda}'_{T,t} \left.\frac{\partial T\left(\alpha_t,\alpha_{t-1},\eta_t\right)}{\partial \eta_t}\right|_{\Theta^n=\hat{\Theta}^n} &= 0 \\
\hat{\Lambda}'_{Z,t} \left.\frac{\partial Z\left(y_t,\alpha_t,\varepsilon_t\right)}{\partial \alpha_t}\right|_{\Theta^n=\hat{\Theta}^n} + \hat{\Lambda}'_{T,t} \left.\frac{\partial T\left(\alpha_t,\alpha_{t-1},\eta_t\right)}{\partial \alpha_t}\right|_{\Theta^n=\hat{\Theta}^n} & \\
+ \hat{\Lambda}'_{T,t+1} \left.\frac{\partial T\left(\alpha_{t+1},\alpha_t,\eta_{t+1}\right)}{\partial \alpha_t}\right|_{\Theta^n=\hat{\Theta}^n} &= 0 \qquad t = 1,\ldots,n \\
\hat{\Lambda}_{T,n+1} &= 0 \\
Z\left(y_t,\hat{\alpha}_t,\hat{\varepsilon}_t\right) &= 0 \\
T\left(\hat{\alpha}_t,\hat{\alpha}_{t-1},\hat{\eta}_t\right) &= 0
\end{aligned}
\tag{15}
$$

The above system of equations can now simply be solved by the stacked-time algorithm, which yields the mode of the state vector $\hat{\alpha}^n$, i.e. the main output of the exact nonlinear and non-Gaussian Kalman smoother.

Based on this approach, I have developed a toolbox where the user only has to specify the specification of the state space model and the rest is taken care of automatically by the toolbox. In particular, the toolbox automatically sets up the optimization problem, takes analytical derivatives, and solves the resulting system of equations by the stacked-time algorithm. The toolbox is documented in an accompanying paper, see De Wind (2017).[12]

The key insight behind the toolbox is that the optimization problem is set up as a constrained one, which avoids substitutions that would otherwise be more difficult to generalize (or even impossible in case of implicit functions), and moreover equality constraints would require a Lagrangian approach anyway.[13]

[12]The toolbox is freely available at www.github.com/jorisdewind but requires a Matlab licence (though the toolbox is likely to work under Octave as well). The toolbox is developed on top of Dynare, which itself is an open source Matlab toolbox, see www.dynare.org.

[13]The treatment of nonlinear state space models in Durbin and Koopman (2001) does not take full account of equality constraints. For one thing, they do not allow for equality constraints in the measurement equations, since they assume that the covariance matrix H is non-singular and they take the inverse of this matrix. Yet, they do allow for equality constraints in the state equations via the selection matrix R, which consists of a subset of the columns of the identity matrix, but their treatment seems to be incorrect. As a matter of fact, Durbin and Koopman (2001) rewrite their state equations to get an explicit expression for the vector with state disturbances, which is then substituted out from the complete data log-likelihood function. Herewith, they also throw away the equality constraints, which they do not subsequently account for in their optimization problem (i.e. they should have set up a Lagrangian).

3 Examples

This section contains two examples. The first example is just for demonstration purposes, while the second example is a conditional forecasting exercise with a nonlinear macroeconomic model. The latter example is interesting on its own for practitioners at e.g. central banks and international institutions.

3.1 Nonlinear state space model

Consider the following nonlinear Gaussian state space model

$$\begin{aligned}
y_t &= \cos\left(x_t^2 + \varepsilon_t\right)\exp\left(z_t\right) && \varepsilon_t \sim N\left(0, h_y^2\right) \\
x_t &= \left(1-\rho_x\right)\mu_x + \rho_x x_{t-1} + \eta_t^x && \eta_t^x \sim N\left(0, q_x^2\right) \\
z_t &= \left(1-\rho_z\right)\mu_z + \rho_z z_{t-1} + \eta_t^z && \eta_t^z \sim N\left(0, q_z^2\right) \qquad t = 1, \ldots, n \\
& x_0 \text{ and } z_0 \text{ are given}
\end{aligned} \tag{16}$$

The goal is to estimate the state variables given the data (and for given parameter values). The exact nonlinear Kalman smoother solves this estimation problem in an optimal manner. In order to follow the procedure outlined in subsection 2.3, define $\alpha_t = (x_t, z_t)'$ and $\eta_t = (\eta_t^x, \eta_t^z)'$. Then, the functions $Z(\cdot)$ and $T(\cdot)$ can be written as

$$\begin{aligned}
Z\left(y_t, \alpha_t, \varepsilon_t\right) &= y_t - \cos\left(x_t^2 + \varepsilon_t\right)\exp\left(z_t\right) \\
T\left(\alpha_t, \alpha_{t-1}, \eta_t\right) &= \begin{pmatrix} x_t - \left(1-\rho_x\right)\mu_x - \rho_x x_{t-1} - \eta_t^x \\ z_t - \left(1-\rho_z\right)\mu_z - \rho_z z_{t-1} - \eta_t^z \end{pmatrix}
\end{aligned} \tag{17}$$

so that the needed derivatives are given by

$$\begin{aligned}
\frac{\partial Z\left(y_t, \alpha_t, \varepsilon_t\right)}{\partial \alpha_t} &= \begin{pmatrix} 2x_t \sin\left(x_t^2 + \varepsilon_t\right)\exp\left(z_t\right) & -\cos\left(x_t^2 + \varepsilon_t\right)\exp\left(z_t\right) \end{pmatrix} \\
\frac{\partial Z\left(y_t, \alpha_t, \varepsilon_t\right)}{\partial \varepsilon_t} &= \sin\left(x_t^2 + \varepsilon_t\right)\exp\left(z_t\right) \\
\frac{\partial T\left(\alpha_t, \alpha_{t-1}, \eta_t\right)}{\partial \alpha_t} &= I_2 \\
\frac{\partial T\left(\alpha_{t+1}, \alpha_t, \eta_{t+1}\right)}{\partial \alpha_t} &= -\begin{pmatrix} \rho_x & 0 \\ 0 & \rho_z \end{pmatrix} \\
\frac{\partial T\left(\alpha_t, \alpha_{t-1}, \eta_t\right)}{\partial \eta_t} &= -I_2
\end{aligned} \tag{18}$$

Moreover, the disturbances are in this example independently normally distributed, with

the following log probability density functions

$$\begin{aligned}\log\left(p_{\varepsilon}\left(\varepsilon_t\right)\right) &= \text{constant} - \frac{1}{2}\left(\frac{\varepsilon_t}{h_y}\right)^2 \\ \log\left(p_{\eta}\left(\eta_t\right)\right) &= \text{constant} - \frac{1}{2}\left(\frac{\eta_t^x}{q_x}\right)^2 - \frac{1}{2}\left(\frac{\eta_t^z}{q_z}\right)^2\end{aligned} \tag{19}$$

so that the needed derivatives are given by

$$\begin{aligned}\frac{\partial \log\left(p_{\varepsilon}\left(\varepsilon_t\right)\right)}{\partial \varepsilon_t} &= -\left(\frac{\varepsilon_t}{h_y^2}\right) \\ \frac{\partial \log\left(p_{\eta}\left(\eta_t\right)\right)}{\partial \eta_t} &= -\begin{pmatrix}\frac{\eta_t^x}{q_x^2} & \frac{\eta_t^z}{q_z^2}\end{pmatrix}\end{aligned} \tag{20}$$

We have now all the ingredients to apply the procedure outlined in subsection 2.3. In particular, I have manually calculated all the derivatives that are part of the system of equations (15), so that we can now simply employ the stacked-time algorithm. Nevertheless, I would like to emphasize that it is also possible to make use of my toolbox, described in De Wind (2017), to run the exact nonlinear Kalman smoother automatically (which among other things avoids the need to manually take derivatives).

Numerical example with measurement error. I have generated artificial data with the following parameter configuration: $\rho_x = 0.95$, $\mu_x = 1$, $\rho_z = -0.75$, $\mu_z = -1$, $h_y = 0.25$, $q_x = 0.1$, and $q_z = 0.05$. The number of periods is set to 200 and the state vector is initialized at its unconditional mean, that is $x_0 = \mu_x$ and $z_0 = \mu_z$.

The observable data and latent state variables are plotted in figure 1 together with the smoothed state variables. It is clear from the figure that the state variable x_t is very well identified and estimated very well over the whole sample period, while the state variable z_t is considerably more difficult to track. Of course, that is due to the model (with three disturbances and only one measurement per time period) since the exact nonlinear Kalman smoother delivers the best possible estimate of the state variables given the data.

Numerical example without measurement error. I have also applied the exact nonlinear Kalman smoother to the same artificial data but then absent of measurement error, i.e. $h_y = 0$. One of the advantages of my implementation of the exact nonlinear Kalman smoother is that it naturally extends to the case without measurement error. In this particular example, simply drop the first-order condition with respect to ε^n from the system of equations (15), which is the *only* adjustment that is needed.

The observable data and latent state variables are plotted in figure 2 together with the smoothed state variables. The latent state variables are exactly the same as in the previous example, while the observable data are a little bit smoother. Now the state variable x_t is

Figure 1: Numerical example with measurement error

Figure 2: Numerical example without measurement error

captured almost perfectly, while the state variable z_t is sometimes still difficult to track. Again, that is due to the model since the exact nonlinear Kalman smoother delivers the best possible estimate of the state variables given the data.

3.2 Conditional forecasting

Consider the following dynamic version of the IS-LM model

$$\begin{aligned}
Y_t &= C_t + I_t + G_t \\
Y_t^d &= Y_t - T_t \\
T_t &= t_0 + t_1 Y_t + \varepsilon_t^T \\
C_t &= c_0 + c_1 Y_{t-1}^d + c_2 Y_t^d + c_3 \mathbb{E}_t \left[Y_{t+1}^d \right] + c_4 R_t + c_5 R_t^2 + \varepsilon_t^C \\
I_t &= i_0 + i_1 Y_{t-1} + i_2 Y_t + i_3 \mathbb{E}_t \left[Y_{t+1} \right] + i_4 R_t + i_5 R_t^2 + \varepsilon_t^I \\
M_t^d &= m_0 + m_1 Y_t + m_2 R_t + m_3 R_t^2 + \varepsilon_t^{MD} \\
M_t^d &= M_t^s
\end{aligned} \tag{21}$$

The first equation is the aggregate resource constraint which equates domestic output Y_t to the sum of consumption C_t, investment I_t, and government spending G_t. The second equation defines disposable income Y_t^d as the difference between income Y_t and taxes T_t. The third equation determines the level of taxes as a function of income, implying that taxes are determined endogenously as opposed to government spending which is an exogenous variable. The fourth equation determines consumption as a function of lagged, current, and expected future disposable income as well as the interest rate R_t. The fifth equation determines investment as a function of lagged, current, and expected future output and also the interest rate. The sixth equation determines money demand M_t^d which depends on output and the interest rate. The seventh equation determines the money market equilibrium, where money supply M_t^s is determined exogenously. Finally, the ε_t^X variables are the residuals of the behavioral equations.

It is straightforward to run a deterministic simulation (assuming perfect foresight) for the purpose of making a forecast for given trajectories of the exogenous variables, i.e. just invoke the stacked-time algorithm.[14] However, we are often also interested in making a forecast conditional on trajectories for one or more of the endogenous variables (or we might have a ragged-edge dataset). The exact nonlinear and non-Gaussian Kalman smoother can be employed for making such *conditional* forecasts and it turns out that the novel implementation introduced in this paper suits perfectly in this setting. In fact, for solving a forward-looking model we need to use the stacked-time algorithm and the same is true for

[14] Of course, there also exist alternative approaches for solving models with forward-looking variables that do not rely on the assumption of perfect foresight.

the conditional forecasting approach proposed in this paper, but it turns out that for the joint problem we only need to run the stacked-time algorithm once. As a matter of fact, relative to the problem of solving a forward-looking model we just need to augment the system with some extra difference equations in order to make a conditional forecast. This will be elaborated on in the remainder of this section.[15]

Suppose we want to make a forecast conditional on the assumption that the government budget closes for the next eight periods, that is $T_t = G_t$ for $t = 1, \ldots, 8$. Of course, the tax equation in the above IS-LM model will yield non-zero residuals but since taxes also appear in the other model equations it is very likely that the other residuals will be non-zero too. The idea behind the exact nonlinear Kalman smoother is to maximize the likelihood of the residuals subject to the constraints implied by the model, which delivers the most likely conditional forecast.

Assuming independently normally distributed residuals with variance q_X^2 for residual ε_t^X we get the following first-order conditions with respect to the residuals[16]

$$\begin{aligned} \frac{\varepsilon_t^T}{q_T^2} &= \lambda_t^T & \qquad \frac{\varepsilon_t^C}{q_C^2} &= \lambda_t^C \\ \frac{\varepsilon_t^I}{q_I^2} &= \lambda_t^I & \qquad \frac{\varepsilon_t^{MD}}{q_{MD}^2} &= \lambda_t^{MD} \end{aligned} \tag{22}$$

where the Lagrangian multipliers are labeled by the left-hand sides of the associated model equations, which are considered to be the constraints of the optimization problem, apart from the last equation which is labeled by the interest rate. Note that the above first-order conditions have a particularly nice form in this example because of the additive Gaussian residuals.

We also need to calculate the first-order conditions with respect to the model variables

[15]The conditional forecasting approach described in this section is related to the 'observation procedure' of Sandee, Don, and van den Berg (1984), and can be considered as a generalization of their approach. In fact, they start off from a similar constrained optimization problem, yet employ a conjugate gradient algorithm that cannot be generalized to the case with non-Gaussian disturbances. Moreover, they do not deal with models with forward-looking variables, although the setup of their optimization problem could in principle handle such models. Of course, such an extension would require an alternative numerical solution technique such as the stacked-time algorithm.

[16]The underlying optimization problem is similar to expression (14), although for this example it is not necessary to distinguish between measurement and state equations.

(apart from the exogenous and conditioning variables), which are given by[17]

$$\begin{aligned}
\partial Y_t: \quad & -\lambda_t^Y + \lambda_t^{YD} + t_1\lambda_t^T + i_1\lambda_{t+1}^I + i_2\lambda_t^I + i_3\lambda_{t-1}^I + m_1\lambda_t^{MD} = 0 \\
\partial Y_t^d: \quad & -\lambda_t^{YD} + c_1\lambda_{t+1}^C + c_2\lambda_t^C + c_3\lambda_{t-1}^C = 0 \\
\partial C_t: \quad & \lambda_t^Y - \lambda_t^C = 0 \\
\partial I_t: \quad & \lambda_t^Y - \lambda_t^I = 0 \\
\partial M_t^d: \quad & -\lambda_t^{MD} - \lambda_t^R = 0 \\
\partial R_t: \quad & (c_4 + 2c_5R_t)\,\lambda_t^C + (i_4 + 2i_5R_t)\,\lambda_t^I + (m_2 + 2m_3R_t)\,\lambda_t^{MD} = 0
\end{aligned} \tag{23}$$

Altogether, the IS-LM model equations (21), the first-order conditions with respect to the residuals (22), and the first-order conditions with respect to the endogenous model variables (23) should be put together as a system of equations, which can then be solved by the stacked-time algorithm. Of course, the system of equations is not complete without initial and terminal conditions. The initial conditions are simply the most recent data points for the backward-looking variables and zeros for the Lagrangian multipliers.

However, the terminal conditions are a bit more involved.[18] Typically, we want to provide terminal conditions such that the forward-looking variables are back in the steady state after a very large number of simulation periods, but this is in conflict with the timing of the Lagrangian multipliers that should be zero beyond the conditioning period (in this example from period nine onwards).[19] Of course, it is not directly possible to provide terminal conditions for different variables in different time periods, but we can solve this issue by introducing a dummy variable as elaborated on below.[20]

In particular, we can augment the system with another equation introducing a dummy variable D_t that sets taxes equal to the conditioning values T_t^{cond} in the conditioning period, while treating taxes endogenously beyond the conditioning period. In case taxes are endogenous, the conditions in (23) should be augmented with the first-order condition with respect to taxes, which is given by $-\lambda_t^{YD} - \lambda_t^T = 0$.[21] Hence, the system of equations is

[17] It should be noted that it is also possible to make use of my toolbox which avoids the need to manually take derivatives, but for the current example the derivations are straightforward anyway.

[18] At this point, it is strongly recommended to read the appendix to obtain a basic understanding of the stacked-time algorithm as well as the terminal conditions.

[19] The larger the number of simulation periods the more accurate is the assumption that the model is back in its steady state in the terminal period. It depends on the model (in particular its eigenvalues) how many simulation periods are sufficient to solve the model accurately. In the current example only eighteen simulation periods are sufficient, although often we need a much larger number of simulation periods.

[20] This dummy variable approach is incorporated in my toolbox in an automatic fashion.

[21] Of course, in case of an unconditional forecast, we would only have needed the equations in (21) and, in fact, the equations in (22) and (23) would have been redundant, but nevertheless would have implied that all future residuals and Lagrangian multipliers are zero (which is exactly what we want for an unconditional forecast).

completed by

$$D_t \left(T_t - T_t^{cond}\right) + (1 - D_t) \left(-\lambda_t^{YD} - \lambda_t^T\right) = 0 \tag{24}$$

and the terminal conditions are simply the steady-state values for the forward-looking variables and zeros for the Lagrangian multipliers.

As an illustration, figure 3 plots the unconditional forecast for output together with the conditional forecasts conditioning on $T_t = G_t$ for two, four, and eight conditioning periods, respectively. For the chosen parameterization, taxes are higher in the conditioning path than what the model endogenously implies, so that the conditional forecasts for output are lower the longer the conditioning period.[22]

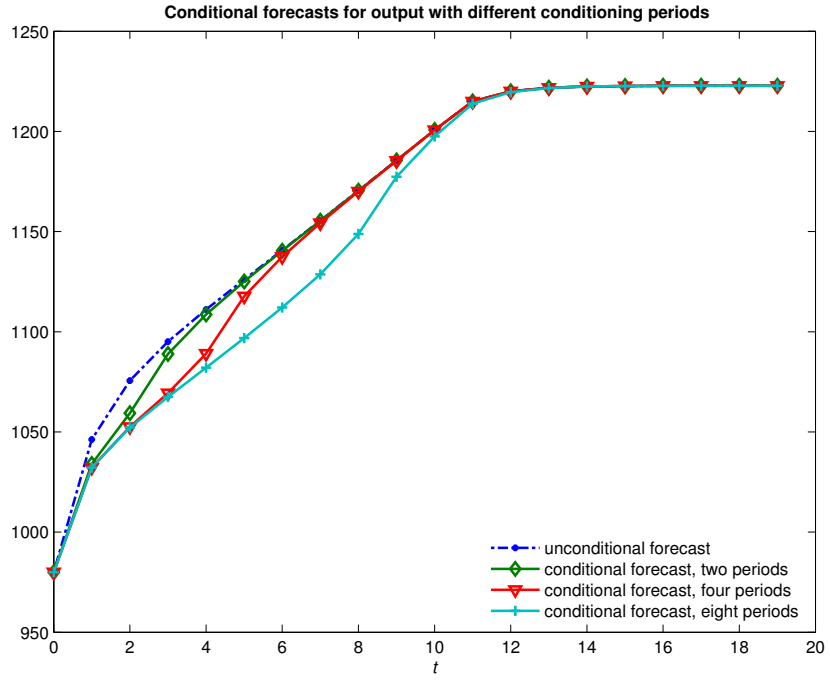

Figure 3: Example conditional forecasting with a nonlinear macroeconomic model

4 Concluding remarks

I have introduced a novel implementation of the exact nonlinear and non-Gaussian Kalman smoother, which can handle very general state space models including ones with implicit

[22]Note that it does not make sense to condition taxes over the full simulation period because in that case the assumption that the economy goes back to the steady state does not make sense.

functions in the measurement and/or state equations as well as equality constraints. My approach has the additional advantage that it can be fully automated, which enabled me to develop a Matlab toolbox that can handle a wide class of discrete-time state space models. The toolbox is documented in an accompanying paper, see De Wind (2017).

I have demonstrated my approach on a simple nonlinear state space model as well as for conditional forecasting with a nonlinear macroeconomic model. The latter example has turned out to be particularly nice, because for solving forward-looking models we already need to use the stacked-time algorithm which can be combined efficiently with the stacked-time algorithm that is needed for the conditional forecasting approach proposed in this paper.[23]

The conditional forecasting example is interesting on its own for practitioners at e.g. central banks and international institutions. As a matter of fact, this example is related to an old yet still very relevant paper by Sandee, Don, and van den Berg (1984), who already wrote that the Kalman filter would be a natural alternative for their 'observation procedure,' but rejected the possibility because of computational costs.[24] My conditional forecasting approach can be considered as a non-Gaussian generalization of their approach, with the key distinction that my approach can deal with models with forward-looking variables and is absent of approximation errors.

This paper might result in many future applications since my novel implementation makes the exact nonlinear and non-Gaussian Kalman smoother much easier to use (whether or not using my toolbox). Moreover, the methodology that I propose in this paper is very general and it is, for example, not hard to extend the methodology to account for mixed frequencies and missing observations (i.e. by using a similar dummy variable trick as in the conditional forecasting example).

Finally, my approach just yields point estimates, i.e. the mode of the state of the system, but it is also possible to calculate the Hessian of the complete data log-likelihood function, which could potentially be used as an input for a simulation algorithm. Furthermore, a next step would be to work out the details for diffuse initialization.

References

Bell, B. M. (1994): "The Iterated Kalman Smoother as a Gauss-Newton Method," *SIAM Journal on Optimization*, 4, pp. 626–636.

[23] Of course, there also exist alternative approaches for solving models with forward-looking variables. It would be interesting to develop an approach that combines my exact nonlinear and non-Gaussian Kalman smoother with an alternative way to solve for the forward-looking variables that does not require assuming perfect foresight.

[24] At the CPB Netherlands Bureau for Economic Policy Analysis a simplified version of the 'observation procedure' of Sandee, Don, and van den Berg (1984) is being used for virtually all forecasting exercises.

de Wind, J. (2017): "SMOOTHIES: A Toolbox for the Exact Nonlinear and Non-Gaussian Kalman Smoother," CPB Discussion Paper No. 360.

Durbin, J. and S. J. Koopman (1992): "Filtering, smoothing and estimation for time series models when the observations come from exponential family distributions," unpublished manuscript.

——— (2001): *Time Series Analysis by State Space Methods*, Oxford University Press.

Hollinger, P. (1996): "The Stacked-Time Simulator in TROLL: A Robust Algorithm for Solving Forward-Looking Models," unpublished manuscript.

Juillard, M. (1996): "DYNARE: a program for the resolution and simulation of dynamic models with forward variables through the use of a relaxation algorithm," CEPREMAP Working Papers, 9602, CEPREMAP.

Sandee, J., F. J. H. Don, and P. J. C. M. van den Berg (1984): "Adjustment of projections to recent observations," *European Economic Review*, 26, pp. 153–166.

A Stacked-time algorithm

A basic understanding of the stacked-time algorithm might help to gain a better intuition for the exact nonlinear and non-Gaussian Kalman smoother developed in this paper, and in particular the discussion about the terminal conditions in the conditional forecasting example.

The purpose of the stacked-time algorithm is to solve nonlinear dynamic models under the assumption of perfect foresight. The basic idea behind the algorithm is to solve a model with n dynamic equations in n endogenous variables simultaneously for T time periods, yielding a stacked-time system of nT equations in nT unknowns. This approach is computationally feasible since the stacked-time system is very sparse, which can exploited by the algorithm.[25]

Consider the following generic form for a dynamic model with one lag and lead[26]

$$f\left(y_{t-1}, y_t, y_{t+1}\right) = 0 \tag{25}$$

where the vector with lags y_{t-1} (i.e. the predetermined state variables) is given and the vector with leads y_{t+1} (i.e. the expectations) is unknown. The idea behind the stacked-time algorithm is to solve for the leads under the assumption of perfect foresight and using the above model equation one period forward, giving the joint system of equations

$$\begin{aligned} f\left(y_{t-1}, y_t, y_{t+1}\right) &= 0 \\ f\left(y_t, y_{t+1}, y_{t+2}\right) &= 0 \end{aligned} \tag{26}$$

where, in fact, we have just shifted the problem of solving for the leads since now we need an extra condition for y_{t+2} instead of y_{t+1}. The stacked-time algorithm iterates on the idea of forwarding the model equation (25) for a large number of time periods T, which yields the following stacked-time system of nT equations in nT unknowns

$$\begin{aligned} f\left(y_0, y_1, y_2\right) &= 0 \\ f\left(y_1, y_2, y_3\right) &= 0 \\ &\dots \\ f\left(y_{T-1}, y_T, y_{T+1}\right) &= 0 \end{aligned} \tag{27}$$

[25] Every block of n equations corresponds to a particular time period and hence depends only on model variables around that time period, resulting in a band-block-diagonal Jacobian matrix of partial derivatives that includes many zeros and hence is called sparse.

[26] Models with multiple lags and leads can be rewritten as models with just one lag and lead by introducing auxiliary variables (like rewriting an AR(p) as VAR(1)). Furthermore, note that in general not all variables enter the model with a lag or a lead, so that models are in general more sparse than the generic form given in expression (25).

which can be written alternatively with the $f(\cdot)$ functions stacked in $F(\cdot)$, giving us

$$F(y_0, y_1, y_2, \ldots, y_T, y_{T+1}) = 0 \tag{28}$$

Of course, we still have the problem that we need an extra (terminal) condition for the leads in the last period, that is y_{T+1}. If the number of time periods T is large enough we are typically willing to assume that the model goes back to its steady state or balanced growth path in the terminal period. Even though the terminal conditions are often chosen according to the steady state or balanced growth path, it is possible to make any assumption about the terminal period.

In principle, we can use a standard Newton method to solve the above stacked-time system of nT equations in nT unknowns, but for computational efficiency we need to exploit the sparsity of the band-block-diagonal Jacobian matrix of partial derivatives of the $F(\cdot)$ function, as mentioned in footnote 25. The stacked-time algorithm is implemented in software packages such as Dynare and Troll.[27] The standard Dynare implementation makes use of the sparse-matrix capabilities that are standard in Matlab.[28]

Finally, even though the stacked-time algorithm is often applied to solve dynamic models, it is a general-purpose algorithm that can be used to solve general systems of difference equations. Indeed, in the current paper the stacked-time algorithm is applied to solve the difference equations that arise from the first-order conditions of the large-scale constrained optimization problem that is equivalent to the nonlinear and non-Gaussian Kalman smoothing problem.

[27] See www.dynare.org and www.hendyplan.com/troll-software, respectively. Dynare is a freely available open source Matlab toolbox, while Troll is a commercial software package.

[28] There are also more advanced versions of the stacked-time algorithm, available in both Dynare and Troll, but that is beyond the scope of this appendix and not necessary for the understanding of the current paper.

Publisher:

CPB Netherlands Bureau for Economic Policy Analysis
P.O. Box 80510 | 2508 GM The Hague
T (088) 984 60 00

September 2017