

# Block Kalman filtering for large-scale DSGE models\*

Ingvar Strid<sup>†</sup> Karl Walentin<sup>‡</sup>

Sveriges Riksbank Working Paper Series

No. 224

June 2008

## Abstract

In this paper block Kalman filters for Dynamic Stochastic General Equilibrium models are presented and evaluated. Our approach is based on the simple idea of writing down the Kalman filter recursions on block form and appropriately sequencing the operations of the prediction step of the algorithm. It is argued that block filtering is the only viable serial algorithmic approach to significantly reduce Kalman filtering time in the context of large DSGE models. For the largest model we evaluate the block filter reduces the computation time by roughly a factor 2. Block filtering compares favourably with the more general method for faster Kalman filtering outlined by Koopman and Durbin (2000) and, furthermore, the two approaches are largely complementary.

---

\*We would like to thank Mattias Villani, Steve Lionel and Duncan Po. The views expressed in this paper are solely the responsibility of the authors and should not be interpreted as reflecting the views of the Executive Board of Sveriges Riksbank.

<sup>†</sup>Dept. of Economic Statistics and Decision Support, Stockholm School of Economics, P.O. Box 6501, SE-113 83 Stockholm, Sweden. [ingvar.strid@hhs.se](mailto:ingvar.strid@hhs.se).

<sup>‡</sup>Research Department, Sveriges Riksbank, SE-103 37 Stockholm, Sweden. [karl.walentin@riksbank.se](mailto:karl.walentin@riksbank.se).

# 1 Introduction

Dynamic Stochastic General Equilibrium (DSGE) models are nowadays routinely estimated using Bayesian methods. The size of some models, e.g. those developed at various central banks, is becoming larger and computational time is perceived as a concern [Adolfson, Lindé and Villani (2007)], [Azzin, Girardi and Ratto (2007)], [Christoffel, Coenen and Warne (2007)]. Bayesian estimation of linearised DSGE models using the Metropolis-Hastings algorithm and the Kalman filter for likelihood evaluation typically require at least, say, 100.000 draws from the posterior. For larger models several days of computing time may be needed and most of this time is spent on Kalman filtering.

This motivates the development of efficient Kalman filter algorithms and implementations tailored to the particular linear and Gaussian state-space model (LGSS) associated with a wide-class of linearly approximated DSGE models. Faster Kalman filtering, if possible, increases the quality and/or reduces the time of the likelihood-based analysis of DSGE models.

The purpose of this paper is to present and evaluate block Kalman filters for DSGE models which exploit the symmetry of the filter and the particular DSGE model block structure and sparse structure of some system matrices. Our DSGE model specific approach, which is straightforward and mainly attempts to reduce the time of the Kalman filter prediction step, is integrated with and compared to the general strategy for faster Kalman filtering outlined by Koopman and Durbin (2000). Their approach focuses on the updating step of the filter, implying that the two approaches are largely complementary.<sup>1</sup>

The usefulness of the block Kalman filter approach is illustrated using three well-known macroeconomic models and particular interest is devoted to the large-scale open-economy model developed at Sveriges Riksbank [Adolfson, Laséen, Lindé and Villani (2007)]. The evaluation exercise clarifies the interrelationship between model size, block structure, algorithm, implementation language, compiler, matrix library and Kalman filtering time for these three representative models. This exercise, we believe, provides a useful guide to the researcher who wants to obtain maximal computational efficiency in estimating linearised DSGE models.

Our main results are, first, that block filtering is the only algorithmic approach that, in itself, can deliver a significantly lower Kalman filtering wall-clock time for the large-scale DSGE model. Second, quicker execution of the updating step can largely be achieved without resorting to the univariate filtering approach of Koopman and Durbin (2000). Third, for smaller models the choice of implementation language appears much more important than the choice of Kalman filter algorithm.

The practical perspective of the paper guides the choice of programming language for implementations of the filters. The filters are programmed in Matlab, the dominant language among economists, and as Fortran Mex functions to be called from Matlab. The

---

<sup>1</sup>The terms *updating* and *prediction* are explained in section 4 where the Kalman filter is presented.

standard implementation of the Kalman filter is presumably close to the ideal application for Matlab. However, Matlab is not the ideal language if one wants to maximise the performance of Kalman filtering for DSGE models. In contrast, a language like Fortran (and presumably C) appears better suited for the implementations suggested in this paper.

The Kalman filter implementations that come with the paper are easy to use. From a user's perspective the only requirement is that the state vector of the economic model is ordered in a particular way. The Kalman filter interface is uniform across algorithms/implementations such that details of algorithms can largely be hidden from the user.

The paper proceeds as follows. In section 2 the computational problem is briefly described. In section 3 it is shown how to cast the state-space model in the form required for block filtering. In section 4 the Kalman filter is presented in a form suitable for the presentation of the block filters in section 5. Three example models and the setup of the computational experiment are described in section 6, and in section 7 results from tests of the filters are presented.

## 2 The computational kernel

For large DSGE models Kalman filtering time is dominated by the matrix multiplication associated with the prediction step of the Kalman filter

$$P^* = TPT^T \tag{1}$$

where  $P$  is an  $m \times m$  symmetric matrix and  $T$  is the  $m \times m$  state transition matrix. As an example, the open-economy DSGE model presented in Adolfson, Laséen, Lindé and Villani (2007) contains  $m = 65$  state variables. For this model, and an efficient Matlab implementation of the Kalman filter, on a standard desktop computer more than 60% of Kalman filtering time is spent on the above multiplication.

The univariate filtering approach [Koopman and Durbin (2000)] works on the updating step of the Kalman filter and is therefore, in itself, expected to provide only limited time gains when the state dimension is large relative to the observation dimension, which is often the case for DSGE models. However, the generality of the approach, its apparent success in significantly reducing Kalman filter time for many models and the unavailability of other approaches to faster Kalman filtering suggest it as a natural benchmark algorithm.

The block approach, on the other hand, aims mainly to reduce the time of the prediction step by exploiting the structure of the matrices  $T$  and  $P$  and therefore works complementary to univariate filtering. It should be noted that the matrix  $T$  is not sparse but that it has a specific block structure for DSGE models, e.g. for the open-economy model mentioned above 35% of the elements of  $T$  are nonzero.

The block filter performs the above multiplication on blocks of the matrices  $T$  and  $P$ . First, the symmetry of  $P$  (and  $P^*$ ) is “manually” taken into account. Second, a number

of multiplications involving zero submatrices are disposed of in the process.

### 3 State-space representation

A wide class of linearly approximated DSGE models can be cast in the general linear state space form

$$X_t = c + T(\theta) X_{t-1} + R(\theta) \epsilon_t \quad (2)$$

$$Y_t = d(\theta) + Z X_t + v_t \quad (3)$$

where [2] is the state transition equation and [3] is the observation equation. The state vector  $X_t$  has dimension  $m$ , the measurement vector  $Y_t$  has dimension  $N$  and the vector of fundamental innovations  $\epsilon_t$  has dimension  $g$ . The dimensions of the matrices  $T$ ,  $R$  and  $Z$  are  $m \times m$ ,  $m \times g$  and  $N \times m$ . The structural and auxiliary parameters of the DSGE model are collected in the vector  $\theta$ . The distributions of the fundamental innovations  $\epsilon_t$  and the measurement error  $v_t$  are  $N(0, Q)$  and  $N(0, H)$  respectively and  $\epsilon_t$  and  $v_s$  are assumed to be independent.

If Kalman filtering time is a serious concern it can be expected that at least some effort has been made to reduce the dimension of the state vector, i.e. to remove any variable  $X_{jt}$  for which the  $j^{\text{th}}$  column of  $T$  and the  $j^{\text{th}}$  column of  $Z$  are zero vectors (static endogenous variables which are not observed). Keeping these variables in the system obviously is contradictory to fast filtering although it could be convenient for other purposes. Since this reduction of the state dimension is easily performed manually prior to estimation it will not be considered further here.

It will be assumed that the covariance matrices

$$P_{t|s} = E \left[ (X_t - E[X_t | Y_{1:s}]) (X_t - E[X_t | Y_{1:s}])^T | Y_{1:s} \right]$$

$s = t - 1, t, t + 1, t = 1, \dots, T$ , are not necessarily positive definite since many DSGE models are represented as state-space models with singular  $P_{t|s}$ .

The most general state-space model of interest to us here has four blocks. The state vector is partitioned as

$$X^T = [X_1^T \ X_2^T \ X_3^T \ X_4^T] \quad (4)$$

with dimensions  $m_j, j = 1, 2, 3, 4$ , such that  $g = m_1 + m_2 + m_3$  and  $m = g + m_4$ .

The first block contains exogenous AR(1) processes and the second block contains exogenous VAR(1) processes. Exogenous variables which appear in the observation equation, i.e. variables for which the corresponding column in  $Z$  contain non-zero entries, are collected in the third block and endogenous variables appear in the fourth block. The

block structure of a model is succinctly captured by the vector  $\mathbf{m}_4 = (m_1, m_2, m_3, m_4)$ . The four-block structure of the model is described by the matrices

$$T = \begin{bmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & A_3 & 0 \\ B_1 A_1 & B_2 A_2 & B_3 A_3 & C \end{bmatrix} \quad (5)$$

$$R = \begin{bmatrix} I_{m_1} & 0 & 0 \\ 0 & I_{m_2} & 0 \\ 0 & 0 & I_{m_3} \\ B_1 & B_2 & B_3 \end{bmatrix} \quad (6)$$

$$Q = \begin{bmatrix} Q_1 & 0 & 0 \\ 0 & Q_2 & 0 \\ 0 & 0 & Q_3 \end{bmatrix} \quad (7)$$

$$Z = [ 0 \quad 0 \quad Z_3 \quad Z_4 ] \quad (8)$$

where the matrices  $A_1$  and  $Q_1$  are assumed to be diagonal, the matrices  $Q_2$  and  $Q_3$  are symmetric and  $I_{m_i}$  is the  $m_i \times m_i$  identity matrix. In the block filters potential diagonality of  $H$  is not exploited, implying that the presence of correlated measurement errors does not present any additional issues.

The only requirement for application of the filters in this paper is that the LGSS model has the structure described by [5]-[8]. An unordered model, i.e. a model for which the state vector have not been cast in the form [4], described by the matrices  $\tilde{T}$ ,  $\tilde{R}$ ,  $\tilde{Q}$  and  $\tilde{Z}$  can be transformed to the required form by defining the  $m \times m$  reordering matrix  $M$ .

Assume that the original, unordered, state vector is given by  $\tilde{X}$ . Let  $M_{ij} = 1$  if the variable in place  $i$  in  $\tilde{X}$  obtains the new position  $j$  in  $X$  and  $M_{ij} = 0$  otherwise. The ordered model is then obtained utilizing the relationships

$$T = M^T \tilde{T} M \quad (9)$$

$$R = M^T \tilde{R} M \quad (10)$$

$$Q = \tilde{M}^T \tilde{Q} \tilde{M} \quad (11)$$

$$Z = \tilde{Z} M \quad (12)$$

where the  $g \times g$  matrix  $\tilde{M}$  is obtained as a submatrix of  $M$  through deletion of rows and columns associated with endogenous variables. It is thus easy to transform the state-space representation of a DSGE model to the required form.

Faster Kalman filtering in this context is essentially based on two approaches:

- Taking into account the symmetries in the filter, i.e. only calculating the upper (or lower) triangular part of covariance matrices  $P_{t|s}$ ,  $s = t - 1, t$ ,  $t = 1, \dots, T$ . This can be performed (i) using special matrix routines for symmetric matrices and/or (ii) manually, as in the block filter.
- Exploiting the DSGE model specific structure of  $T$  and  $Z$ . This goes beyond the symmetric filter and further motivates the blocked filters presented below.

Many DSGE models only consist of blocks 1 and 4. Therefore, in section 5.1, we first present the 2-block-filter which is suitable for models which do not contain an exogenous VAR block. In section 5.2 the more general 4-block filter is presented. A 3-block filter, which is obtained by merging blocks 3 and 4 of the 4-block filter into one block is presented in the Appendix. A notable example of a DSGE model, in addition to [Adolfson, Laséen, Lindé and Villani (2007)], that consists of more than two blocks is the New Area-Wide Model (NAWM) developed at the ECB [Christoffel, Coenen and Warne (2007)].

Before moving on we briefly comment on two issues. First, the discussion is restricted to time-invariant (constant coefficient) models. However, it would be straightforward to extend the filters to the time-varying case. In fact, if time variation is restricted to certain blocks of a model this could be an additional argument in favour of block filtering but this issue is not discussed further here.

Second, if the covariance matrix  $P$  has converged to its steady state solution avoiding updating it saves a lot of computational effort. For reasons of simplicity and transparency it will be assumed in our experiments that  $P$  has not converged during the time span of the filter. This is normally the case for large DSGE models and the relevant sample lengths. For smaller models convergence is presumably more common and steady state Kalman filtering can sometimes save a lot of computational effort.

## 4 Kalman filter

### 4.1 Kalman filter equations

The Kalman filter equations are included to make the paper self-contained (see e.g. Harvey (1989) ch. 3). Let

$$X_{t|s} = E[X_t | Y_{1:s}]$$

denote the optimal estimator of the state vector  $X_t$  based on the observations up to time  $s$

$$Y_{1:s} = (Y_1, Y_2, \dots, Y_s)$$

and let

$$P_{t|s} = E \left[ (X_t - E[X_t | Y_{1:s}]) (X_t - E[X_t | Y_{1:s}])^T | Y_{1:s} \right]$$

The *updating* (or *contemporaneous filtering*) equations are

$$X_{t|t} = X_{t|t-1} + K_t F_t^{-1} (Y_t - Z X_{t|t-1} - d) \quad (13)$$

and

$$P_{t|t} = P_{t|t-1} - K_t F_t^{-1} K_t^T \quad (14)$$

where

$$F_t = V(Y_t | Y_{1:t-1}) = Z K_t + H \quad (15)$$

is assumed to be positive definite and

$$K_t = P_{t|t-1} Z^T \quad (16)$$

The *prediction* equations are

$$X_{t+1|t} = T X_{t|t} + c \quad (17)$$

and

$$P_{t+1|t} = T P_{t|t} T^T + R Q R \quad (18)$$

As the state dimension of the LGSS model increases the matrix multiplication in [18] will dominate computational time, as previously discussed. For DSGE models the selector matrix  $Z$  is typically sparse, i.e. it contains relatively few non-zero entries, such that [15] and [16] are sparse-dense type of multiplications. This sparseness of  $Z$  can sometimes be exploited for more efficient computing.

## 4.2 Implementation issues

The usual (1-block) Kalman filter is implemented exactly as described above. In Matlab this can be done in essentially one way. Using Fortran the filter can be implemented using BLAS routines for symmetric matrices, e.g. *symm*, *syrk* and *syr2k*, calculating only the upper (or lower) part of the matrices  $P_{t|t-1}$  and  $P_{t|t}$ . This will be referred to as a *symmetric filter implementation*. The alternative is to implement the filter in correspondence with its implementation in Matlab, i.e. using the standard matrix multiplication, *gemm*, everywhere, a *non-symmetric filter implementation*.

It should be noted that the block filter, to some extent, reduces the rationale for applying symmetric routines since the block filter approach manually deals with the symmetry of the covariance matrices  $P_{t|s}$ . The added value of the block filter is of course that, in addition, the computations associated with the zero blocks of the state transition matrix,  $T$ , need not be performed.

The perspective of the paper is to take the set of matrix operation routines available in BLAS, as implemented in MKL or IMSL (or similar and/or overlapping libraries) as

given building blocks in constructing efficient Fortran implementations of the filters.<sup>2</sup> In other words there is no ambition to improve upon the design of the basic routines of these libraries.

Interestingly, the particular operation of main interest for Kalman filtering [1] has not yet, to our knowledge, been implemented as a separate routine in any of the mentioned libraries. Therefore we compare several ways of performing this operation based on routines in the matrix libraries, over a range of matrix sizes, in order to obtain a reasonably efficient implementation. Our symmetric filter implementations perform the type of multiplication in [1] via calls to *symm* and *syr2k*.

## 5 Block Kalman filter

### 5.1 2-block filter

In this section the Kalman filter is written down on 2-block form. The DSGE model is assumed to consist of  $m_1$  exogenous AR(1) shocks (block 1) and  $m_2$  endogenous variables (block 2). Many small and intermediate scale DSGE models have this structure. If the model contain exogenous VAR processes and/or AR/VAR variables which appear in the observation equation these are placed in the second block. In this case a simple adjustment to the filter is required (see below).

The model is thus assumed to be on the form

$$T = \begin{bmatrix} A & 0 \\ BA & C \end{bmatrix} \quad (19)$$

$$R = \begin{bmatrix} I_{m_1} \\ B \end{bmatrix} \quad (20)$$

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \quad (21)$$

$$Z = \begin{bmatrix} Z_1 & Z_2 \end{bmatrix} = \begin{bmatrix} 0 & Z_2 \end{bmatrix} \quad (22)$$

where  $A$  is diagonal.

The mean vectors  $X_{1t|s}$  and  $X_{2t|s}$  and the blocks  $P_{11t|s}$ ,  $P_{12t|s}$  and  $P_{22t|s}$  of the covariance matrices

$$P_{t|s} = \begin{bmatrix} P_{11t|s} & P_{12t|s} \\ & P_{22t|s} \end{bmatrix}$$

---

<sup>2</sup>The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations (see e.g. [www.netlib.org](http://www.netlib.org)). MKL (Intel Math Kernel Library) is a library of math routines and it includes BLAS routines optimised for Intel processors. IMSL (International Mathematics and Statistics Library) is a library of routines for numerical analysis, also containing the BLAS routines.

are propagated instead of  $X_{t|s}$  and  $P_{t|s}$ ,  $s = t - 1, t$  and  $t = 1, \dots, T$ .

Again, the purpose of the block filter approach is to (i) avoid calculation of  $P_{21} = P_{12}^T$ , (ii) to dispose of a set of unnecessary multiplications, i.e. those involving zero submatrices (here the upper-right submatrix of  $T$ ), (iii) to exploit the special structure of the lower-left submatrix of  $T$  and (iv) to exploit the diagonality of the matrix  $A$ . Whether this is useful of course depends on the sizes  $m_1$  and  $m_2$ . For DSGE models  $m$  is typically significantly larger than  $N$  and the shocks of the model constitute a significant part of the state vector.

The equations presented here are obtained by simply assuming the 2-block structure, [19]-[22], and writing down the Kalman filter recursions [13]-[18] while appropriately sequencing the operations of the prediction step, i.e. to minimise the number of matrix multiplications performed. Although the filtering equations are expressed in terms of the matrices  $A, B, C$  and  $Z_2$  it is seen above that these are easily extracted from the matrices  $T, R$  and  $Z$  such that the input to the Kalman filter can be kept the same as for the 1-block filter if desired.

### 5.1.1 Updating equations

The updating step of the filter consists of the following set of equations, which are obtained by writing out the updating equations of the Kalman filter, [13] and [14], on block form. The contemporaneous filtering covariance matrices are obtained as

$$P_{11t|t} = P_{11t|t-1} - P_{12t|t-1} Z_2^T F_t^{-1} Z_2 P_{12t|t-1}^T \quad (23)$$

$$P_{12t|t} = P_{12t|t-1} - P_{12t|t-1} Z_2^T F_t^{-1} Z_2 P_{22t|t-1} \quad (24)$$

$$P_{22t|t} = P_{22t|t-1} - P_{22t|t-1} Z_2^T F_t^{-1} Z_2 P_{22t|t-1} \quad (25)$$

and the means as

$$X_{1t|t} = X_{1t|t-1} + P_{12t|t-1} Z_2^T F_t^{-1} (Y_t - d - Z_2 X_{2t|t-1}) \quad (26)$$

$$X_{2t|t} = X_{2t|t-1} + P_{22t|t-1} Z_2^T F_t^{-1} (Y_t - d - Z_2 X_{2t|t-1}) \quad (27)$$

where

$$F_t = Z P_{t|t-1} Z^T + H = Z_2 P_{22t|t-1} Z_2^T + H \quad (28)$$

is assumed to be positive definite.

Let

$$F_t^{-1} = \tilde{F}_t^T \tilde{F}_t \quad (29)$$

and use the matrices

$$P_{12}^* = P_{12t|t-1} Z_2^T \tilde{F}_t^T \quad (30)$$

$$P_{22}^* = P_{22t|t-1} Z_2^T \tilde{F}_t^T \quad (31)$$

and the vector

$$v_t^* = \tilde{F}_t (Y_t - d - Z_2 X_{2t|t-1}) \quad (32)$$

for the matrix multiplications above and for the computation of the likelihood at time  $t$  (see below). In our implementations the matrix  $\tilde{F}_t$  is obtained as the inverse of the Cholesky factor of  $F_t$ , i.e. it is not the Cholesky factor of  $F_t^{-1}$ .

### 5.1.2 Prediction equations

The prediction step is given by the following set of equations which are obtained by writing out equations [17] and [18] on block form.

First

$$P_{11t+1|t} = AP_{11t|t}A^T + Q$$

which, due to the assumed diagonality of  $A$ , is obtained as

$$P_{11t+1|t} = P_{11t|t} \odot A_{11} + Q \quad (33)$$

where

$$A_{11} = \text{diag}(A) \text{diag}(A)^T$$

(here  $\odot$  denotes element-by-element multiplication, Hadamard product). Next

$$\begin{aligned} P_{12t+1|t} &= AP_{11t|t}A^TB^T + AP_{12t|t}C^T + QB^T = \\ &P_{11t+1|t}B^T + AP_{12t|t}C^T \end{aligned}$$

or

$$P_{12t+1|t} = P_{11t+1|t}B^T + L_t \quad (34)$$

where

$$L_t = (P_{12t|t}C^T) \odot A_{12}$$

and

$$A_{12} = \text{diag}(A) \mathbf{1}_{m_2}^T$$

where  $\mathbf{1}_{m_2}$  is a (column) vector of dimension  $m_2$  containing ones.

The block corresponding to the endogenous variables is obtained as

$$\begin{aligned} P_{22t+1|t} &= BAP_{11t|t}A^TB^T + BAP_{12t|t}C^T + BQB^T + \\ &+ CP_{12t|t}^T A^T B^T + CP_{22t|t}C^T = \\ &= BAP_{11t|t}A^TB^T + BQB^T + M_t + M_t^T + CP_{22t|t}C^T \end{aligned}$$

where

$$M_t = BL_t$$

or

$$P_{22t+1|t} = BP_{12t+1|t} + M_t^T + CP_{22t|t}C^T \quad (35)$$

As a computational detail, if we let

$$G_t = BP_{12t+1|t} + M_t = B(P_{12t+1|t} + L_t)$$

then

$$P_{22t+1|t} = \frac{1}{2}(G_t + G_t^T) + CP_{22t|t}C^T$$

The one-step ahead forecast of the state vector is given by

$$X_{1t+1|t} = AX_{1t|t} = \text{diag}(A) \odot X_{1t|t} \quad (36)$$

and

$$X_{2t+1|t} = BAX_{1t|t} + CX_{2t|t} = BX_{1t+1|t} + CX_{2t|t} \quad (37)$$

### 5.1.3 Likelihood

The log likelihood at time  $t$  is given by

$$\begin{aligned} \log L_t &= -N \log(2\pi) + \log(|F_t^{-1}|) - \\ &- \frac{1}{2} (Y_t - d - Z_2 X_{2t|t-1})^T F_t^{-1} (Y_t - d - Z_2 X_{2t|t-1}) = \\ &= -N \log(2\pi) + 2 \sum_{i=1}^N \log(\tilde{F}_{ii,t}) - \frac{v_t^{*T} v_t^*}{2} \end{aligned} \quad (38)$$

where  $\tilde{F}_{ii,t}$  is element  $(i, i)$  of the matrix  $\tilde{F}_t$  and the log likelihood is obtained as

$$\log L = \sum_{t=1}^T \log L_t$$

### 5.1.4 Theoretical gains from 2-block filtering

Consider the matrix multiplication [1] in a 2-block context

$$P^* = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} T_{11}^T & T_{21}^T \\ T_{12}^T & T_{22}^T \end{bmatrix} \quad (39)$$

Without any assumptions on the matrices  $P$  and  $T$ , performing the multiplication [39] naively involves  $8 + 8 = 16$  ordinary matrix multiplications for the submatrices of the partitioned matrices. Recognising the symmetry of  $P$  and the output matrix  $P^*$  this is reduced to 14 matrix multiplications, some of those involving symmetric input or output

matrices.<sup>3</sup> Further assuming that  $T_{11}$  is diagonal and that  $T_{12} = 0$  the number of multiplications is reduced to 2 Hadamard products plus 6 multiplications. Finally, also taking into account the special structure of  $T_{21}$  (see expression [19]) allows us to perform the multiplication [39] using 2 Hadamard products and 5 multiplications, as shown above in writing out the prediction equations for the 2-block Kalman filter.

For large matrices and  $m_1 \approx m_2$  these matrix multiplication counts would provide a good estimate of the time savings from 2-block filtering. In practise, e.g. for the DSGE models considered in the experiments of this paper, we expect the time savings to be smaller than indicated by the counts. The main value of this exercise is instead that it helps understanding the contribution of the different DSGE model specific assumptions in reducing the number of operations of the Kalman filter prediction step.

### 5.1.5 Moving exogenous variables

Assume that the DSGE model contain exogenous AR(1) shock processes which appear in the observation equation, i.e. exogenous variables for which the corresponding column in  $Z$  contain non-zero entries. For example, if the model contain unit root shocks this situation emerges. Further assume that the exogenous variables have been ordered such that the  $k$  exogenous variables that appear in the observation equation are the  $k$  last variables in the vector  $X_1$  such that

$$Z_1 = \begin{bmatrix} \bar{Z}_1 & \bar{Z}_2 \end{bmatrix} = \begin{bmatrix} 0 & \bar{Z}_2 \end{bmatrix}$$

where  $\bar{Z}_1$  has dimension  $N \times \tilde{m}_1$  where  $\tilde{m}_1 = m_1 - k$ . These  $k$  exogenous AR(1) variables are moved into the vector of endogenous variables forming the new blocks,  $\tilde{X}_1$  and  $\tilde{X}_2$  with dimensions  $\tilde{m}_1$  and  $\tilde{m}_2$  where  $\tilde{m}_2 = m_2 + k$ . The first vector thus contain exogenous variables which do not enter the observation equation and the second vector contain exogenous variables that appear in the observation equation and endogenous variables.

Let

$$\tilde{Z}_2 = \begin{bmatrix} \bar{Z}_2 & Z_2 \end{bmatrix}$$

and define  $\tilde{A} (\tilde{m}_1 \times \tilde{m}_1)$ ,  $\tilde{B} (\tilde{m}_2 \times \tilde{m}_1)$ ,  $\tilde{C} (\tilde{m}_2 \times \tilde{m}_2)$  via

$$T = \begin{bmatrix} A & 0 \\ BA & C \end{bmatrix} = \begin{bmatrix} \tilde{A} & 0 \\ \tilde{B}\tilde{A} & \tilde{C} \end{bmatrix}$$

where  $\tilde{B}$  is the lower left  $\tilde{m}_2 \times \tilde{m}_1$  matrix of  $R$ .

Further, let  $\tilde{Q} (\tilde{m}_1 \times \tilde{m}_1)$  be the upper-left submatrix of  $Q$ . Also, let  $R_{22}^*$  denote the  $(\tilde{m}_2 \times \tilde{m}_2)$  lower-right submatrix of  $RQR^T$ . The modified 2-block filter is now obtained

---

<sup>3</sup>Clearly there exist more sophisticated schemes which reduce the number of multiplications required, e.g. the Strassen algorithm for general matrix multiplication which would require  $7 + 7 = 14$  multiplications if no assumptions are imposed on  $P$  and  $T$ . However, we believe that the Strassen algorithm, or related algorithms, are of little practical interest in our context.

by replacing  $A, B, C, Q, P_{ij}, X_i$  and  $Z_2$  with  $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{Q}, \tilde{P}_{ij}, \tilde{X}_i$  and  $\tilde{Z}_2$  in the 2-block filter presented above and making the adjustment in the prediction step for  $\tilde{P}_{22}$

$$\begin{aligned}\tilde{P}_{22t+1|t} &= \tilde{B}\tilde{A}\tilde{P}_{11t|t}\tilde{A}^T\tilde{B}^T + \tilde{B}\tilde{A}\tilde{P}_{12t|t}\tilde{C}^T + R_{22}^* + \tilde{C}\tilde{P}_{12t|t}^T\tilde{A}^T\tilde{B}^T + \tilde{C}\tilde{P}_{22t|t}\tilde{C}^T = \\ &= \tilde{B}\tilde{P}_{12t+1|t} + \tilde{C}P_{12t|t}^T\tilde{A}^T\tilde{B}^T + \tilde{C}\tilde{P}_{22t|t}\tilde{C}^T + R_{22}^* - \tilde{B}\tilde{Q}\tilde{B}^T\end{aligned}\quad (40)$$

To aid understanding, note that if  $k = 0$

$$R_{22}^* = \tilde{B}\tilde{Q}\tilde{B}^T = BQB^T$$

A similar modification could be applied if the model contain an exogenous VAR block. The VAR block would be moved into the endogenous variables block independent of whether the variables appear in the observation equation or not. In practise the modification becomes interesting for models which contain at most a few - say one or two - exogenous variables in the observation equation. If many exogenous variables appear in the observation equation and/or the model contain many exogenous VAR variables the more general 4-block filter presented in the next section should instead be applied.

From a user's perspective the modification of the filter is not an issue. All that needs to be kept in mind is that exogenous variables which appear in the observation equation are placed in the second block.

### 5.1.6 2-block filter with a univariate filtering step

In this section we show how to use the approach of Koopman and Durbin (2000) with the two-block filter. In their univariate approach the observations are introduced one at a time in the updating step. In this way two matrix multiplications and the inversion of the matrix  $F_t$  can be avoided in the Kalman filter. For DSGE models the matrix  $Z$  is typically sparse and hence the multiplications in [15] and [16] can be performed efficiently, perhaps reducing the rationale for univariate filtering.

Define

$$X_{it|t,1} = E[X_{it,1}|Y_{1:t-1}] = X_{it|t-1} \quad (41)$$

$$P_{ijt|t,1} = E[(X_{it,1} - X_{it|t,1})(X_{jt,1} - X_{jt|t,1})|Y_{1:t-1}] = P_{ijt|t-1} \quad (42)$$

$$X_{it|t,k} = E[X_{it,k}|Y_{1:t-1}, y_{t,1}, \dots, y_{t,k-1}] \quad (43)$$

$$P_{ijt|t,k} = E[(X_{it,k} - X_{it|t,k})(X_{jt,k} - X_{jt|t,k})|Y_{1:t-1}, y_{t,1}, \dots, y_{t,k-1}] \quad (44)$$

for  $i = 1, 2, j = i, 2, k = 1, \dots, N$  and where  $y_{t,k}$  denotes the  $k^{th}$  element of the observation vector such that

$$X_{it|t} = X_{it|t,N+1} \quad (45)$$

$$P_{ijt|t} = P_{ijt|t,N+1} \quad (46)$$

The updating equations can then be written as

$$P_{ijt|t,k+1} = P_{ijt|t,k} - K_{it,k} F_{t,k}^{-1} K_{jt,k}^T \quad (47)$$

$$X_{it|t,k+1} = a_{it|t,k} + K_{it,k} F_{t,k}^{-1} v_{t,k} \quad (48)$$

where

$$v_{t,k} = y_{t,k} - Z_{2,k} X_{2t|t,k} \quad (49)$$

$$F_{t,k} = Z_{2,k} K_{2t,k} \quad (50)$$

$$K_{1t,k} = P_{12t,k} Z_{2,k}^T \quad (51)$$

$$K_{2t,k} = P_{22t,k} Z_{2,k}^T \quad (52)$$

Here  $Z_{2,k}$  denotes the  $k^{th}$  row of  $Z_2$  such that the vectors  $K_1$  and  $K_2$  have dimensions  $m_1$  and  $m_2$  respectively and  $F$  and  $v$  are scalars. The 2-block filter with univariate filtering is thus obtained by replacing [23]-[28] with [45]-[52]. The experiments using the univariate filtering approach in this paper only considers the case when the covariance matrix of the measurement errors,  $H$ , is diagonal.

## 5.2 4-block filter

Assume that the matrices  $T$ ,  $Z$ ,  $R$  and  $Q$  have the structure [5]-[8] where  $A_1$  is a diagonal  $m_1 \times m_1$  matrix,  $A_2$  and  $A_3$  have dimensions  $m_2 \times m_2$  and  $m_3 \times m_3$  respectively, the matrices  $B_1, B_2, B_3$  and  $C$  have dimensions  $m_4 \times m_1, m_4 \times m_2, m_4 \times m_3$  and  $m_4 \times m_4$  respectively and  $Z_3$  and  $Z_4$  have dimensions  $N \times m_3$  and  $N \times m_4$ .

The block structure of the covariance matrices is described by

$$P_{t|s} = \begin{bmatrix} P_{11t|s} & P_{12t|s} & P_{13t|s} & P_{14t|s} \\ & P_{22t|s} & P_{23t|s} & P_{24t|s} \\ & & P_{33t|s} & P_{34t|s} \\ & & & P_{44t|s} \end{bmatrix}$$

Again, the block filter is obtained by simply writing out the Kalman filter recursions under the assumed state-space structure and appropriately sequencing the operations of the prediction step.

### 5.2.1 Updating equations

Define

$$\tilde{P}_{it} = \tilde{P}_{i3t|t-1} Z_3^T + \tilde{P}_{i4t|t-1} Z_4^T \quad (53)$$

for  $i = 1, 2, 3, 4$  such that

$$F_t = Z_3 \tilde{P}_{3t} + Z_4 \tilde{P}_{4t} + H \quad (54)$$

Note again that if sparseness of  $Z_3$  and  $Z_4$  is exploited [53] and [54] can be performed particularly efficiently.

The updating covariance matrix blocks are then obtained as

$$P_{ijt|t} = P_{ijt|t-1} - P_{it}P_{jt}^T \quad (55)$$

for  $i = 1, 2, 3, 4$  and  $j = i, \dots, 4$  where

$$P_{it} = \tilde{P}_{it}\tilde{F}_t^T$$

and

$$F_t^{-1} = \tilde{F}_t^T\tilde{F}_t$$

The means are given by

$$X_{it|t} = X_{it|t-1} + P_{it}v_t^* \quad (56)$$

where

$$v_t^* = \tilde{F}_t(Y_t - d - Z_2X_{2t|t-1})$$

and the likelihood is obtained as described previously for the 2-block filter.

### 5.2.2 Prediction equations

First, define

$$A_{11} = \text{diag}(A_1)\text{diag}(A_1)^T \quad (57)$$

and

$$A_{1i} = \text{diag}(A_1)1_{m_i}^T \quad (58)$$

for  $i = 2, 3, 4$  such that  $A_{1i}$  is an  $m_1 \times m_i$  matrix.

The prediction covariance matrix blocks are obtained via the following sequence of steps. The first row of blocks, i.e.  $P_{11}$  to  $P_{14}$ , is obtained via

$$P_{11t+1|t} = A_{11} \odot P_{11t|t} + Q_1 \quad (59)$$

$$P_{12t+1|t} = A_{12} \odot (P_{12t|t}A_2^T) \quad (60)$$

$$P_{13t+1|t} = A_{13} \odot (P_{13t|t}A_3^T) \quad (61)$$

$$P_{14t+1|t} = L_{1t} + \sum_{i=1}^3 P_{1it+1|t}B_i^T \quad (62)$$

where

$$L_{1t} = A_{14} \odot (P_{14t|t}C^T)$$

The second row of blocks,  $P_{22}$  to  $P_{24}$ , is obtained as

$$P_{22t+1|t} = A_2 P_{22t|t} A_2^T + Q_2 \quad (63)$$

$$P_{23t+1|t} = A_2 P_{23t|t} A_3^T \quad (64)$$

$$\begin{aligned} P_{24t+1|t} &= L_{2t} + \sum_{i=1}^3 P_{2it+1|t} B_i^T \\ &= L_{2t} + P_{12t+1|t}^T B_1^T + P_{22t+1|t} B_2^T + P_{23t+1|t} B_3^T \end{aligned} \quad (65)$$

where

$$L_{2t} = A_2 P_{24t|t} C^T$$

The third row of blocks,  $P_{33}$  and  $P_{34}$ , is obtained through

$$P_{33t+1|t} = A_3 P_{33t|t} A_3^T + Q_3 \quad (66)$$

$$P_{34t+1|t} = L_{3t} + \sum_{i=1}^3 P_{3it+1|t} B_i^T \quad (67)$$

where

$$L_{3t} = A_3 P_{34t|t} C^T$$

Finally, let

$$M_t = \sum_{i=1}^3 B_i (P_{i4t+1|t} + L_{it})$$

Then

$$P_{44t+1|t} = \frac{1}{2} (M_t + M_t^T) + C P_{44t|t} C^T \quad (68)$$

The prediction means are obtained as

$$X_{1t+1|t} = \text{diag}(A_1) \odot X_{1t|t} \quad (69)$$

$$X_{2t+1|t} = A_2 X_{2t|t} \quad (70)$$

$$X_{3t+1|t} = A_3 X_{3t|t} \quad (71)$$

and

$$X_{4t+1|t} = C X_{4t|t} + \sum_{i=1}^3 B_i X_{it+1|t} \quad (72)$$

### 5.2.3 Univariate filtering step: 4-block filter

The univariate filtering equations are analogous to the two-block case. The definitions [41]-[46] and the relations [47] and [48] now hold for  $i = 1, 2, 3, 4$ ,  $j = i, \dots, 4$   $k = 1, \dots, N$  where  $v$ ,  $F$  and  $K_i$  are instead defined as

$$v_{t,k} = y_{t,k} - Z_{3,k}X_{3t|t,k} - Z_{4,k}X_{4t|t,k} \quad (73)$$

$$F_{t,k} = Z_{3,k}K_{3t,k} + Z_{4,k}K_{4t,k} \quad (74)$$

$$K_{it,k} = P_{i3t,k}Z_{3,k}^T + P_{i4t,k}Z_{4,k}^T \quad (75)$$

Although the block filter and univariate filtering approaches are largely complementary in increasing Kalman filtering speed, as the number of blocks increase it is clear that in practical implementations the efficiency of univariate filtering is negatively affected by working on several smaller arrays.

## 6 Experimental setup

The computational experiment has two objectives. First, and most importantly, we want to assess the performance of the block Kalman filter approach to faster Kalman filtering. Second, and more broadly, we want to provide a fuller picture of the many factors that affect Kalman filtering performance for DSGE models in practise.

Three DSGE models of different sizes are used to evaluate the computational performance of the filter implementations, where the large model is of primary interest. For each model a benchmark problem is constructed which consists of evaluating the likelihood with the Kalman filter a number of times.<sup>4</sup>

The filters are coded in Matlab, in Fortran and as Fortran Mex functions to be called from Matlab. The default software setup consists of Matlab R2008a, Intel Visual Fortran (IVF) compiler 10.1.019 and the BLAS routines available in Matlab (MKL, *libmwbblas.lib*). The Fortran Mex functions are always compiled with the default *mexopts.bat* compiler options provided by the Mathworks.<sup>5</sup>

Results are presented for two computers: (i) AMD Opteron 275, 2.2 GHz and (ii) Dell 690, with Intel Xeon 5160 3.00 GHz. It can be expected that the block filter has better performance for machines with a relatively limited cache memory since it generally works

---

<sup>4</sup>Since Kalman filtering time is independent of the parameter point,  $\theta$ , at which the likelihood is evaluated, assuming convergence to the steady state solution does not occur, we simply choose an arbitrary point for evaluation. An alternative would be to use the filter in the Metropolis-Hastings algorithm.

<sup>5</sup>We have established that for this software setup execution in Fortran and execution of Fortran Mex files in Matlab yield similar execution times. Therefore no results from runs using “pure” Fortran are presented. However, in less up-to-date software and/or hardware environments there sometimes appears to be quite a significant cost of using Fortran routines via Matlab in comparison to standalone Fortran.

on smaller matrices. The Dell 690 is a high performance workstation with a large cache memory (4MB L2 cache) and it is therefore interesting to present results for this machine.

The Kalman filter implementations are categorized according to four criteria:

1. The block structure, i.e. the number of blocks of the filter.
2. Symmetric/non-symmetric. This refers to whether the filter is implemented using symmetric (*dsymm*, *dsyrk*, *dsyr2k*) or non-symmetric (*gemm*) BLAS routines.
3. Whether or not sparseness of  $Z$  is exploited in computations.
4. Whether or not the univariate filtering approach of Koopman and Durbin (2000) is applied.

This yields  $4 \times 2 \times 2 \times 2 = 32$  possible implementations and a subset of these are considered in our experiments. On a priori grounds we restrict the number of non-symmetric implementations considered.

The benchmark Fortran mex filter, referred to below, is the 1-block filter which utilizes symmetric BLAS routines but neither exploits sparseness of  $Z$  nor use univariate filtering. The benchmark Matlab filter is the 1-block Matlab implementation of the Kalman filter.

Concerning the Matlab implementations it should be noted that special routines for symmetric matrices are unavailable in Matlab and that exploitation of sparseness of  $Z$  and univariate filtering substantially *increase* Kalman filtering time for the three cases considered here. Therefore, results are presented for Matlab implementations which differ only along the block structure dimension.

Next the three example models are briefly described. The first two models have two blocks and the last model contains four blocks. If the  $(m + 1)$ -block filter is applicable for a model, then the  $m$ -block filter is also applicable. The standard, 1-block, Kalman filter is applied to all models.

## 6.1 A small scale model, An (2005)

The New Keynesian model used by An (2005) features price stickiness via quadratic adjustment costs in price setting. The block structure of the model is described generally by  $\mathbf{m}_4 = (2, 0, 1, 4)$  or in the “natural” 2-block form as  $\mathbf{m}_2 = (2, 5)$  and  $N = 3$  observed variables are used to estimate the model. The presence of an exogenous shock in the observation equation is due to the assumption of a unit root technology shock. The 2-block Kalman filter (with the modification described in section 5.1.5) is applied to this model.

## 6.2 An intermediate scale model, Smets and Wouters (2003)

Next the 2-block filter is applied to a slightly smaller version of the model presented by Smets and Wouters (2003). The block structure of the model is described by  $\mathbf{m}_4 = (8, 0, 0, 11)$  or  $\mathbf{m}_2 = (8, 11)$  and  $N = 5$  observed series are used for estimation. The 2-block filter is applied to the model.

## 6.3 A large scale model, Adolfson, Laséen, Lindé and Villani (2007)

The open economy DSGE model presented in Adolfson, Laséen, Lindé and Villani (2007) has the block structure  $\mathbf{m}_4 = (11, 14, 16, 24)$ . With two or three blocks the block dimensions are  $\mathbf{m}_2 = (11, 54)$  or  $\mathbf{m}_3 = (11, 14, 40)$ . The 2-, 3- and 4-block filters are applied to this model.

# 7 Computational performance

## 7.1 Small scale model

For the small scale model the benchmark problem consists of evaluating the likelihood  $R = 20\,000$  times on the AMD Opteron machine (using one processor) and 12 variants of 1- and 2-block Kalman filters, implemented as Matlab or Fortran mex files, are compared in the Matlab environment. The results are collected in Table 1 where the best performing algorithm/implementation has been given the normalized wall-clock time 100.

The best performing filter is the Fortran mex 1-block filter with a univariate filtering step and which exploits the sparseness of  $Z$ . This filter produces  $R = 20\,000$  likelihood evaluations in 11.5 seconds and it is more than 13 times faster than the benchmark Matlab filter and  $(188 - 100) / 188 = 47\%$  faster than the benchmark Fortran mex filter. The filter with a univariate filtering step is  $(188 - 117) / 188 = 38\%$  faster than the Fortran mex benchmark. As anticipated, for a model of this size and block structure the 2-block filter does not improve performance.

For small scale models the most important aspect thus appears to be the choice of implementation language and algorithmic refinements are of secondary importance. If Kalman filtering speed is an issue in the case of a small model, e.g. because of extensive simulation, univariate filtering appears to be the single most important algorithmic ingredient in increasing speed.

## 7.2 Intermediate scale model

For the intermediate scale model the benchmark problem consists of evaluating the likelihood  $R = 5\,000$  times on the AMD Opteron (using one processor) and, again, 12 variants

---

**Table 1** Kalman filtering time, small model

---

Language	Block	Symm.	Sparse Z	DK	Time
Matlab	1	-	No	No	1369
Matlab	2	-	No	No	2107
Fortran	1	No	No	No	151
Fortran	1	Yes	No	No	188
Fortran	1	Yes	Yes	No	153
Fortran	1	Yes	No	Yes	117
Fortran	1	Yes	Yes	Yes	100
Fortran	2	No	Yes	No	187
Fortran	2	Yes	No	No	265
Fortran	2	Yes	Yes	No	226
Fortran	2	Yes	No	Yes	179
Fortran	2	Yes	Yes	Yes	148

AMD Opteron, 2.2 Ghz, Matlab version 7.6, R2008a, Intel Visual Fortran 10.1.019.

---

of the Kalman filter are compared. To illustrate the sometimes complex interactions between algorithm, implementation and software two setups are compared using this model:

- i) “New”: Matlab R2008a, IVF 10.1, Matlab’s MKL library
- ii) “Old”: Matlab 7.0 (R14), Compaq Visual Fortran 6.6, IMSL library

The results are presented in Table 2. Note that the standardised times are not comparable across columns.

The results for the “new” setup, which are arguably those of main interest for the practitioner, are given in column (i) of Table 2. The best performing filter for this model and block structure is the 2-block Fortran mex filter with a univariate filtering step and exploitation of sparseness of  $Z$ . This filter implementation performs  $R = 5\,000$  Kalman filter likelihood evaluations in 11.0 seconds and it is  $(139 - 100)/139 = 28\%$  faster than the benchmark Fortran filter. The small performance gains come from exploiting sparseness of  $Z$  and univariate filtering whereas the block filter approach yields virtually no gain.

The results for the “old” setup, as given in column (ii) of the table, are quite different. The absolute performance of the best performing filter is now 12.6 seconds. Using this software setup, exploitation of the model’s block structure is the single most important factor in increasing Kalman filtering speed. Moving to the 2-block from the 1-block filter decreases filtering time by  $(186 - 128)/186 = 31\%$  whereas the univariate filtering approach, by itself, decreases time by  $(186 - 169)/186 = 9\%$ .

Why is this comparison interesting? To us it shows that the payoff from applying a more complex algorithm is much smaller using an up-to-date Fortran compiler and BLAS library. In the “new” setup the total gain is  $(139 - 100)/139 = 28\%$  whereas in the “old”

---

**Table 2** Kalman filtering time, intermediate scale model

---

Language	Block	Symm.	Sparse Z	DK	(i) Time	(ii) Time
Matlab	1	-	No	No	441	359
Matlab	2	-	No	No	586	417
Fortran	1	No	No	No	121	207
Fortran	1	Yes	No	No	139	186
Fortran	1	Yes	Yes	No	121	162
Fortran	1	Yes	No	Yes	120	169
Fortran	1	Yes	Yes	Yes	105	149
Fortran	2	No	Yes	No	106	128
Fortran	2	Yes	No	No	137	128
Fortran	2	Yes	Yes	No	121	109
Fortran	2	Yes	No	Yes	118	122
Fortran	2	Yes	Yes	Yes	100	100

AMD Opteron, 2.2 Ghz

---

setup it is  $(186 - 100)/186 = 46\%$ .

From a practical point of view, as was the case for the small-scale model, the choice of implementation language is much more important than the particularities of the algorithm. The non-symmetric Fortran 1-block filter is  $441/121 = 3.6$  times faster than the corresponding Matlab implementation, a *language gain*. The *algorithm gain* of 28% is marginal in comparison. Compared with the small-scale model example the relative performance of Matlab thus becomes better as the time spent on matrix computations increases.

### 7.3 Large scale model

For the large scale model the benchmark problem consists of evaluating the likelihood  $R = 1000$  times on the AMD Opteron and the Intel Xeon and 20 variants of the Kalman filter are compared. Results are presented in Table 3. Note again that the reported standardised times are not comparable across columns. The inclusion of the result for the Matlab implementation of the Kalman filter in Dynare merely serves the purpose of validating that our benchmark Matlab implementation is sufficiently efficient.<sup>6</sup>

On both computers the best performing filter is a non-symmetric filter which exploits the block structure and sparseness of  $Z$ . The time of 1000 likelihood evaluations using this filter is 29.3 seconds on the Opteron and 14.8 seconds on the Xeon. Since the relative performance of the filter versions are quite similar for the Opteron and Xeon the discussion

---

<sup>6</sup>The timing result for Dynare refers to the Kalman filter implementation *DiffuselikelihoodH1.m*.

is largely restricted to results for the latter computer.<sup>7</sup>

First we look at the contribution of each algorithmic feature in isolation, using the 1-block symmetric Fortran mex implementation as the point of reference. Exploitation of sparseness of  $Z$  leads to a speed gain of  $(199-174)/199 = 13\%$ , the gain from univariate filtering is  $(199-188)/199 = 6\%$  and the gain from block filtering is  $(199-128)/199 = 36\%$ . The corresponding numbers on the Opteron are 12%, 6% and 44%. This shows that the block filter approach is the single most important algorithmic ingredient in increasing Kalman filtering speed for this model. It can also be noted that the gain from sub-optimally employing a two-block structure, 17%, is larger than that from univariate filtering or sparse- $Z$ -exploitation.

The best performing filter yields a total *algorithm gain* of  $(184-100)/100 = 46\%$  on the Xeon and 52% on the Opteron. This illustrates the complementarity of block filtering and the other approaches in increasing speed. Of practical interest is also the *language gain*, i.e. the gain from using Fortran instead of Matlab, which is  $(280-184)/280 = 34\%$  on the Xeon and 25% on the Opteron.

Finally, concerning Matlab two things are noted. First, the Matlab block filter implementations yield time gains, although limited, when compared to the benchmark 1-block Matlab implementation. This was not the case for the smaller models considered previously. Second, using “automatic” parallelisation by enabling multithreading in Matlab and executing using the four available processors on the Opteron yields an execution time for the best performing multithreaded implementation which is *larger* than for the best performing serial implementation reported here.

## 7.4 Discussion

The Kalman filter consists of a series of operations on matrices and is therefore very suitable for implementation using Matlab. In fact, it is presumably very close to the ideal application for Matlab. Matrix operations in Matlab are performed using high performance routines, the MKL implementation of BLAS and LAPACK.<sup>8</sup> However, for small models, and hence operations on small matrices, Fortran performs much better than Matlab since the relative time spent on matrix operations is smaller.

The key result to emerge from our experiments above is that the block filter approach is the only serial algorithmic approach that has the potential to significantly reduce Kalman filtering time for large scale DSGE models. Our experiments indicate that block filtering pays off exactly when computational time is becoming a concern.

The univariate filtering approach only yields small time gains for the large model. Furthermore, simply exploiting the sparseness of  $Z$  apparently reduces the rationale for

---

<sup>7</sup>Tests have been carried out using other, lower-performing, hardware and the relative performance of filter implementations is similar to that reported here for the Opteron and Xeon.

<sup>8</sup>Previous versions of Matlab use the ATLAS (Automatically Tuned Linear Algebra Software) library.

---

**Table 3** Kalman filtering time, large model

---

Language	Block	Symm.	Sparse Z	DK	Opteron Time	Xeon Time
Dynare						374
Matlab	1	-	No	No	279	280
Matlab	2	-	No	No	257	282
Matlab	3	-	No	No	233	212
Matlab	4	-	No	No	233	217
Fortran	1	No	No	No	210	184
Fortran	1	Yes	No	No	224	199
Fortran	1	Yes	Yes	No	198	174
Fortran	1	Yes	No	Yes	211	188
Fortran	1	Yes	Yes	Yes	188	164
Fortran	2	No	Yes	No	158	141
Fortran	2	Yes	No	No	187	173
Fortran	2	Yes	Yes	No	165	150
Fortran	2	Yes	No	Yes	183	169
Fortran	2	Yes	Yes	Yes	163	146
Fortran	3	Yes	No	No	148	140
Fortran	3	Yes	Yes	No	133	126
Fortran	4	No	Yes	No	100	100
Fortran	4	Yes	No	No	125	128
Fortran	4	Yes	Yes	No	109	110
Fortran	4	Yes	Yes	Yes	119	119

Matlab R2008a, IVF 10.1, Matlab BLAS library.

---

univariate filtering in the context of DSGE models. However, since the different approaches are largely complementary in reducing filtering time they should be combined for maximal performance.

The univariate filtering approach and the sparse  $Z$ -exploitation is not suitable for implementation in Matlab. Although not reported here in detail we have established, for the examples considered above, that attempts to implement these features using Matlab significantly *increases* computational time. The reason, in simple terms, is that the cost of introducing loops and operating on smaller arrays outweighs any computational savings.

The block filter approach can be implemented successfully in Matlab, as the experimental results for the large model above show. However, the gains from implementing a block filter in Matlab are smaller and, more importantly, the other approaches can not be implemented successfully.

The practical view adopted in this paper essentially means that we focus on Kalman filtering wall-clock time in a practically relevant and easy-to-use software and hardware environment. Abstract performance measurement, e.g. counting matrix multiplications saved, may be of limited value if computing time is the ultimate concern.

Some implications of this perspective are the following: First, for relatively small DSGE models the payoff from using a more refined Kalman filter algorithm are small and simply using the right implementation language appears to be much more important. Second, to be able to compete with recent Matlab versions in the case of large models an up-to-date Fortran compiler and high-performing implementation of BLAS must be used. Third, applying matrix routines for symmetric matrices, as implemented in state-of-the-art matrix libraries (i.e. Intel MKL), does not payoff for models of the sizes considered in this paper. Non-symmetric implementations based on *gemm* are both easier to code and apparently more efficient in practise.<sup>9</sup>

More generally, the size of the time gains will depend on the model size and block structure and on software and hardware characteristics. Naturally, the value of the block filter approach increases with model size and the extent to which the block structure of a model is exploitable. Our work is projected on the assumption that DSGE models will continue to be valuable tools for central banks, as well as other institutions, and that these models will grow both in size and complexity. Our experiments also indicate that the gains from using more complex Kalman filter algorithms becomes smaller in practise when better performing software (compiler, matrix libraries) and hardware is used.

---

<sup>9</sup>Performance of BLAS routines for symmetric matrices is discussed in Goto and van de Geijn (2007). The empirical results in that paper confirm our own experiences, suggesting that the application of symmetric routines becomes interesting for matrix sizes above, say, 200.

## 8 Conclusion

The Kalman filter can be viewed as the key computational kernel in Bayesian estimation of linearised DSGE models and Kalman filtering time is becoming a concern in the context of large-scale DSGE models. This view is reinforced by efforts to parallelise the estimation of DSGE models.[Azzin, Girardi and Ratto (2007)] However, if the parallel computing avenue is opened it is our perspective that the parallel algorithms should be based on as close to optimal as possible serial algorithms.

The block Kalman filter is based on the simple idea of writing down the Kalman filter recursion on block form and appropriately sequencing the operations of the prediction step. The block Kalman filter presented in this paper appear to be the only viable serial algorithmic approach to significantly reduce Kalman filtering time for large-scale DSGE models. For the large-scale example model Kalman filtering time decreased by roughly a factor 2 and most of this time decrease was attributed to block filtering. The approaches directed at the updating step of the Kalman filter, univariate filtering [Koopman and Durbin (2000)] and sparse-Z-exploitation, apparently provide more limited gains for large-scale DSGE models. However, due to complementarities in decreasing Kalman filtering time the algorithmic features should be combined for maximal efficiency.

From a practical perspective, and not surprisingly, we note that the choice of implementation language and matrix library, i.e. the particular implementation of BLAS, are very important for performance. Our experimental results, using up-to-date software and optimised implementations, provide a useful guide to what is important for faster Kalman filtering for DSGE models in practise.

## References

- Adolfson, M., Laséen, S., Lindé, J. and Villani, M. (2007), ‘Bayesian estimation of an open economy dsge model with incomplete pass-through’, *Journal of International Economics* **72**, 481–511.
- Adolfson, M., Lindé, J. and Villani, M. (2007), ‘Bayesian inference in dsge models - some comments’, *Econometric Reviews* **26**, 173–185.
- An, S. (2005), ‘Bayesian estimation of dsge models: Lessons from second-order approximations’, *Working Paper, University of Pennsylvania* .
- Azzin, L., Girardi, R. and Ratto, M. (2007), ‘Paralellization of matlab codes under windows platform for bayesian estimation: a dynare application’, *Manuscript, European Commission* .
- Christoffel, K., Coenen, G. and Warne, A. (2007), ‘Conditional versus unconditional forecasting with the new area-wide model of the euro area’, *Mimeo, ECB* .
- Goto, K. and van de Geijn, R. (2007), ‘High performance implementation of the level-3 blas’, *ACM Transactions on Mathematical Software* **35**(1).
- Harvey, A. (1989), *Forecasting, Structural Time Series Models and the Kalman filter*, Cambridge University Press.
- Koopman, S. and Durbin, J. (2000), ‘Fast filtering and smoothing for multivariate state space models’, *Journal of Time Series Analysis* **21**, 281–296.
- Smets, F. and Wouters, R. (2003), ‘An estimated stochastic dynamic general equilibrium model of the’, *Journal of the European Economic Association* **20**, 891–910.

## 9 Appendix

### 9.1 3-block filter

The 3-block filter is obtained by merging blocks 3 and 4 in the 4-block filter into one block. The size of this block is then  $\tilde{m}_3 = m_3 + m_4$ . Define the  $\tilde{m}_3 \times m_1$  matrix

$$\tilde{B}_1 = \begin{bmatrix} 0 \\ B_1 \end{bmatrix}$$

the  $\tilde{m}_3 \times m_2$  matrix

$$\tilde{B}_2 = \begin{bmatrix} 0 \\ B_2 \end{bmatrix}$$

the  $\tilde{m}_3 \times \tilde{m}_3$  matrix

$$\tilde{C} = \begin{bmatrix} A_3 & 0 \\ B_3 A_3 & C \end{bmatrix}$$

and the  $N \times \tilde{m}_3$  matrix

$$\tilde{Z}_3 = [ Z_3 \quad Z_4 ]$$

and let  $R_3$  be the  $\tilde{m}_3 \times \tilde{m}_3$  lower-right submatrix of  $RQR$ . The matrices  $T$  and  $Z$  then have the structures

$$T = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & 0 \\ \tilde{B}_1 A_1 & \tilde{B}_2 A_2 & \tilde{C} \end{bmatrix} \quad (76)$$

$$Z = [ 0 \quad 0 \quad \tilde{Z}_3 ] \quad (77)$$

The block form of the covariance matrix is then given by

$$P_{t|s} = \begin{bmatrix} P_{11t|s} & P_{12t|s} & P_{13t|s} \\ & P_{22t|s} & P_{23t|s} \\ & & P_{33t|s} \end{bmatrix}$$

#### 9.1.1 Filtering equations

Let

$$\tilde{P}_{it} = P_{i3t|t-1} \tilde{Z}_3^T$$

such that

$$F_t = \tilde{Z}_3 \tilde{P}_{3t} + H$$

and, further, let

$$P_{it} = \tilde{P}_{it} \tilde{F}_t^T$$

where

$$F_t^{-1} = \tilde{F}_t^T \tilde{F}_t$$

The filtering covariance matrices are obtained from

$$P_{ijt|t} = P_{ijt|t-1} - P_{it}P_{jt}^T \quad (78)$$

and the filtered state as

$$X_{it|t} = X_{it|t-1} + P_{it}v_t^* \quad (79)$$

where

$$v_t^* = \tilde{F}_t (Y_t - d - Z_3 X_{3t|t-1})$$

for  $i = 1, 2, 3$  and  $j = i, 3$ .

### 9.1.2 Prediction equations

The first row of blocks,  $P_{11}$ ,  $P_{12}$  and  $P_{13}$  is obtained via

$$P_{11t+1|t} = P_{11t|t} \odot A_{11} + Q_1 \quad (80)$$

$$P_{12t+1|t} = A_{12} \odot (P_{12t|t} A_2^T) \quad (81)$$

$$P_{13t+1|t} = P_{11t+1|t} \tilde{B}_1^T + P_{12t+1|t} \tilde{B}_2^T + \tilde{L}_{1t} \quad (82)$$

where

$$\tilde{L}_{1t} = \tilde{A}_{13} \odot (P_{13t|t} C^T)$$

and

$$\tilde{A}_{13} = \text{diag}(A_1) \mathbf{1}_{\tilde{m}_3}^T$$

The second row of covariance matrix blocks,  $P_{22}$  and  $P_{23}$ , are obtained as

$$P_{22t+1|t} = A_2 P_{22t|t} A_2^T + Q_2 \quad (83)$$

and

$$P_{23t+1|t} = P_{12t+1|t} \tilde{B}_1^T + P_{22t+1|t} \tilde{B}_2^T + \tilde{L}_{2t} \quad (84)$$

where

$$\tilde{L}_{2t} = A_2 P_{23t|t} \tilde{C}^T$$

Finally

$$P_{33t+1|t} = \tilde{M}_t + \tilde{C} P_{33t|t} \tilde{C}^T + \tilde{R}_3 \quad (85)$$

where

$$\tilde{M}_t = \frac{1}{2} (M_t + M_t^T) \quad (86)$$

and

$$M_t = \tilde{B}_1 (P_{13t+1|t} + \tilde{L}_{1t}) + \tilde{B}_2 (P_{23t+1|t} + \tilde{L}_{2t}) \quad (87)$$

and

$$\tilde{R}_3 = R_3 - \tilde{B}_1 Q_1 \tilde{B}_1^T - \tilde{B}_2 Q_2 \tilde{B}_2^T$$

To aid understanding, note that if the model contains no exogenous variables that appear in the observation equation and no exogenous VAR processes, then  $\tilde{R}_3 = 0$ . It can also be mentioned that the *dsyr2k* BLAS routine performs exactly the computation required for obtaining the symmetric matrix  $\tilde{M}_t$  in the symmetric case, i.e. when only the upper (or lower) part of  $P_{33}$  is calculated. Equations [86] and [87] shows how it is implemented using non-symmetric routines, e.g. how it is implemented in the Matlab versions of the filter.