Finance and Economics Discussion Series
Divisions of Research & Statistics and Monetary Affairs
Federal Reserve Board, Washington, D.C.

Using the "Chandrasekhar Recursions" for Likelihood Evaluation of DSGE Models

Edward P. Herbst

2012-35

Using the "Chandrasekhar Recursions" for Likelihood Evaluation of DSGE Models

Ed Herbst
Federal Reserve Board

May 17, 2012

Abstract

In likelihood-based estimation of linearized Dynamic Stochastic General Equilibrium (DSGE) models, the evaluation of the Kalman Filter dominates the running time of the entire algorithm. In this paper, we revisit a set of simple recursions known as the "Chandrasekhar Recursions" developed by Morf (1974) and Morf, Sidhu, and Kalaith (1974) for evaluating the likelihood of a Linear Gaussian State Space System. We show that DSGE models are ideally suited for the use of these recursions, which work best when the number of states is much greater than the number of observables. In several examples, we show that there are substantial benefits to using the recursions, with likelihood evaluation up to five times faster. This gain is especially pronounced in light of the trivial implementation costs – no model modification is required. Moreover, the algorithm is complementary with other approaches.

JEL Classification: C18, C63, E20
Keywords: Kalman Filter, Likelihood Estimation, Computational Techniques

1 Introduction

Dynamic Stochastic General Equilibrium (DSGE) are increasingly estimated by Central Banks and academic economists. In estimation, the model equilibrium conditions can be linked to the data using a Linear Gaussian State Space (LGSS) representation. As model complexity increases, so too does computational time. In maximum likelihood and Bayesian contexts, the log likelihood has to be evaluated hundreds of thousands or millions of times, usually in sequential fashion. In this enviroment, likelihood computation time dominates the running time of the entire algorithm. So it becomes crucial to construct efficient algorithms for likelihood evaluation and state filtering. To wit, considerable effort

Correspondence: Board of Governors of the Federal Reserve System, 20th Street and Consitution Avenue N.W., Washington, D.C. 20551; edward.p.herbst@frb.gov. I thank, without implication, participants at the Research Computing Seminar at the Fed Board and John Roberts for comments. The views expressed in this paper are those of the author and do not necessarily reflect the views of the Federal Reserve Board of Governors or the Federal Reserve System.

has been expended constructing elaborate filters tailored to DSGE models–see, for example, Strid and Walentin (2009).

The purpose of this paper is to report an old and simple algorithm for fast likelihood evaluation outlined in Morf (1974) and Morf, Sidhu, and Kalaith (1974) and show that it is ideally suited for DSGE models. The method, which we will call the "Chandrasekhar Recursions" (CR), is simple to implement and can yield considerable speed improvements.

This paper is closely related to Strid and Walentin (2009), who develop a "Block Kalman Filter" by exploiting the *a priori* known structure of the DSGE model to avoid some large matrix calculations. The algorithm must be applied on a case-by-case basis. We compare the algorithms in the example section. Moreover, in principle, one could use the CR and also exploit the block structure of the DSGE model. Finally, although the CR has not been used to aid the estimation of DSGE models, they have been employed in a general time series context before; see, for example, Klein, Melard, and Zahaf (1998).

The paper is structured as follows. Section 2 contains background information on the DSGE models and the Kalman Filter, Section 3 contains the derivation of the Chandrasekhar Recursions, Section 4 contains four examples, and Section 5 concludes.

2 The Kalman Filter and DSGE Models

The starting point for our analysis is the LGSS representation of an economic model. The state vector s_t is an $n_s \times 1$ collection of all the relevant variables in the economic model, while ϵ_t is $n_\epsilon \times 1$ vector of structural shocks driving the model. The state equation 1 is derived by first linearizing and then solving the model; see An and Schorfheide (2007) for details.

$$s_t = Ts_{t-1} + R\epsilon_t, \quad \epsilon_t \sim N(0,Q) \tag{1}$$

$$y_t = D + Zs_t + \eta_t, \quad \eta_t \sim N(0,H) \tag{2}$$

The vector y_t is an $n_y \times 1$ vector of observables, and η_t is an $n_\eta \times 1$ vector of measurement errors. This paper assumes that $E[\epsilon_t \eta_t'] = 0$ for ease of exposition. The objects (T,R,Q,D,Z,H) are matrix functions of a vector θ of structure parameters. We suppress the dependence for convenience. We make two assumptions, which are very often imposed in applications.

1. The system matrices (T,R,Q,D,Z,H) are time-invariant.
2. The process $\{s_t\}$ is stationary. This means that for our filtering problem, we can use the invariant distribution to initialize our state recursion.

The goal of the econometrician is to evaluate the log likelihood $p(Y|\theta)$ where $Y = [y_1', \ldots, y_T']'$. This is accomplished by using the Kalman Filter. The Kalman Filter computes the log likelihood of the model using the predictive error decomposition,

$$\mathscr{L}(Y|\theta) = -\frac{1}{2}\sum_{t=1}^{T}\left(n_y \ln(2\pi) + \ln(det(F_t)) + v_t' F_t^{-1} v_t\right) \tag{3}$$

Where, the forecast error v_t and the forecast error variance F_t are given by,

$$v_t = y_t - D - Z\hat{s}_{t|t-1}. \tag{4}$$

$$F_t = ZP_{t|t-1}Z' + H \tag{5}$$

The quantities $\hat{s}_{t|t-1}$ and $P_{t|t-1}$ are the predictive state mean and variance,

$$\hat{s}_{t|t-1} = E_{t-1}[s_t], \quad P_{t|t-1} = E_{t-1}[s_t s_t'].$$

The predictive moments of the state equation evolve according to the recursions,[1]

$$\hat{s}_{t+1|t} = T\hat{s}_{t|t-1} + K_t F_t^{-1} v_t. \tag{6}$$

$$P_{t+1|t} = TP_{t|t-1}T' - K_t F_t^{-1} K_t' + RQR'. \tag{7}$$

Equation 7 is often called the matrix Riccati difference equation because of its resemblance to the univariate Riccati differential equation. The initial conditionals $\hat{s}_{1|0}$ and $P_{1|0}$ are, in principle, hyper-parameters to be specified (or estimated) by the user, but this algorithm will use the unconditional distribution. Then $\hat{s}_{1|0} = 0$ and $P_{1|0}$ will be the unconditional variance, i.e., the one that solves the discrete Lyapunov Equation,

$$TP_{1|0}T' - P_{1|0} + RQR' = 0.$$

The formula for the K_t is

$$K_t = TP_{t|t-1}Z'. \tag{8}$$

This gain is usually written as $K_{g,t} = K_t F_t^{-1} = TP_{t|t-1}Z'F_t^{-1}$. Essentially it delivers the optimal extraction of information from the observations at time t.

We are interested in computing the likelihood, which we could do by iterating (in suitable order) over the above equations. However, this filtering procedure slows down as the number of states, n_s, grows. In particular, the performance is dominated by the updating of the state variance using Riccati equation (7). In that equation, we must perform the matrix multiplication,

$$\underbrace{T}_{n_s \times n_s} \underbrace{P_{t|t-1}}_{n_s \times n_s} \underbrace{T'}_{n_s \times n_s}.$$

This operation is $O(n_s^3)$, dominating all other matrix operations in the filter. Strid and Walentin (2009) report that, for a large DSGE models, 60% of filtering time is spent on the operation $TP_{t|t-1}T'$.

[1] We combine the standard prediction and updating equations because we are trying to avoid unecessary calculations.

3 The Chandrasekhar Recursions

The essence of the Chandrasekhar Recursions is the avoidance of direct computation of $P_{t|t-1}$ by instead looking at the *difference* in the state variances,

$$\Delta P_{t|t-1} = P_{t|t-1} - P_{t-1|t-2}.$$

$\Delta P_{t|t-1}$ is a real, symmetric, indefinite matrix.[2] Suppose that $rank(\Delta(P_t)) = \alpha$. Then $\Delta P_{t|t-1}$ can be decomposed into

$$\Delta P_{t|t-1} = W_{t-1} M_{t-1} W_{t-1}',$$

where W_t is $n_s \times \alpha$ matrix and M_t is $\alpha \times \alpha$ matrix . There is room for improvement over the standard algorithm if $\alpha < n_s$, since the matrix multiplications will take place on lower dimensional matrices. In general, however, this factorization is not unique. This seems like a difficult problem, because the factorization is not obvious, and, in our experience, even finding α consistently can be hard. Fortunately, the problem becomes much easier once it is known that the states are stationary.

Before discussing the recursions for W_t and M_t, it's worth noting that the forecast error variance F_t, K_t and $K_{g,t}$ can be rewritten as recursions driven by $\Delta P_{t|t-1}$.

$$\begin{aligned} F_t &= F_{t-1} + Z \Delta P_{t|t-1} Z', && (9) \\ K_t &= K_{t-1} + T \Delta P_{t|t-1} Z', && (10) \\ K_{g,t} &= (K_{g,t-1} F_{t-1} + T \Delta P_{t|t-1} Z') F_t^{-1}. && (11) \end{aligned}$$

For verification of these recursions, see the Appendix. The heart of the algorithm lies in the recursions for M_t and W_t. Start with the initial state variance $P_{1|0}$. Recall from Assumption 2 that the system is stationary, so we can use the unconditional variance of the states, $E[s_t s_t']$ to initialize the state variance,

$$P_{1|0} = E_t[s_t s_t'].$$

This matrix has the property that it solves the discrete Lyapunov equation,

$$P_{1|0} = T P_{1|0} T' + RQR'.$$

Consider now the next-period state variance forecast, $P_{2|1}$. Using the Kalman Filter recursion (7), we can write this as,

$$\begin{aligned} P_{2|1} &= T P_{1|0} T' + RQR' - K_1 F_1^{-1} K_1' \\ &= P_{1|0} - K_1 F_1^{-1} K_1'. && (12) \end{aligned}$$

This means we can write the initial difference of state variances as,

$$\Delta P_{2|1} = -K_1 F_1^{-1} K_1',$$

[2] Indeed, looking at reduced rank decompositions of $P_{t|t-1}$ or differences thereof, has a wide literature known as Square Root Kalman Filtering. I present these related recursions because I think they are much easier to implement.

which suggests natural choices for M_1 and W_1,

$$W_t = K_1 = TP_{1|0}Z', \tag{13}$$
$$M_t = -F_1^{-1} = (ZP_{1|0}Z' + H)^{-1}. \tag{14}$$

Note that thse equations already satistify the initial conditions, since F_1 is positive definite. Moreover, since K_1 is $n_s \times n_y$ and F_1 is $n_y \times n_y$ it is easy to see that

$$rank(\Delta P_2) = \alpha \leq \min\{n_y, n_s\}.$$

For many economic models–in particular large scale DSGE models–$n_y << n_s$. In this case, by using the Chandrasekhar recursions, very roughly speaking, the algorithm has "replaced" the matrix multiplication $P_{t+1|t} = TP_{t|t-1}T' + \ldots$ with $W_tM_tW_t'$, which involves matrices of much lower dimension.

Finally, we must derive the recursions for M_t and W_t. To do this, we utilize the following Lemma.
Lemma. For $\Delta P_t = P_{t|t-1} - P_{t-1|t-2}$,

$$\Delta P_{t+1|t} = (T - K_{g,t}Z)(\Delta P_{t|t-1} - \Delta P_{t|t-1}Z'F_{t-1}^{-1}\Delta P_{t|t-1}Z')(T - K_{g,t}Z)' \tag{15}$$

Proof. See Appendix.
Thus, the Ricatti-type difference equation in P_{t+1} is replaced by another difference equation in terms of $\Delta P_{t+1|t}$. In fact, this new difference equation gives the Chandrasekhar Recursion its name, because it evokes the "so-called Chandrasekhar algorithm through which the matrix [Riccati Equation] is replaced by two coupled differential equations of lesser dimensionality [Aknouche and Hamdi (2007)]." Using the Lemma, substitute our decomposition $W_{t-1}M_{t-1}W_{t-1}'$ for $\Delta P_{t|t-1}$, to obtain

$$\Delta P_{t+1|t} = (T - K_{g,t}Z)(W_{t-1}M_{t-1}W_{t-1}' + W_{t-1}M_{t-1}W_{t-1}'Z'F_{t-1}^{-1}Z'W_{t-1}M_{t-1}W_{t-1}')(T - K_{g,t}Z)'. \tag{16}$$

Rewriting with W_{t-1} removed from the inner product, we have,

$$\Delta(P_{t+1}) = (T - K_{g,t}Z)W_{t-1}(M_{t-1} + M_{t-1}W_{t-1}'Z'F_{t-1}^{-1}Z'W_{t-1}M_{t-1})W_{t-1}'(T - K_{g,t}Z)'. \tag{17}$$

From here it is easy to see that we can rewrite

$$\Delta P_{t+1|t} = W_tM_tW_t'. \tag{18}$$

Where W_t and M_t follow the recursions,

$$W_t = (T - K_tF_t^{-1}Z)W_{t-1}, \quad M_t = M_{t-1} + M_{t-1}W_{t-1}'Z'F_{t-1}^{-1}Z'W_{t-1}M_{t-1} \tag{19}$$

Combining the equations in 19, with the rewritten recursions for F_t and M_t,

$$F_t = F_{t-1} + ZW_{t-1}M_{t-1}W'_{t-1}Z' \quad (20)$$

$$K_t = K_{t-1} + TW_{t-1}M_{t-1}W'_{t-1}Z'. \quad (21)$$

These equations, used in conjunction with with Equations 4 and 6, can be used to iteratively compute the log likelihood in equation 3. We have elimated the state variance prediction, $P_{t|t-1}$, from all the calculations in the algorithm and hence avoid the most computationally intensive calculation. Pseudo code is reported below.

Initialization.

1. Solve the Lyapunov Equation for $\bar{P}$
2. Set $K_1 = T\bar{P}Z', \quad F_1 = Z\bar{P}Z' + H$.
3. Set $W_1 = K_1, \quad M_1 = -F_1^{-1}$.

Iteration. For each time period $t = 1, \ldots, T$.

1. Compute v_t, and evaluate the likelihood.
2. Compute $\hat{a}_{t+1|t}$.
3. Compute F_{t+1} using Equation 20, and F_{t+1}^{-1}.
4. Compute K_{t+1} using Equation 21,
5. Compute W_{t+1} using part one of Equation 19.
6. Compute M_{t+1} using part two of Equation 19.

Another advantage of this initialization is that it can shown that M_t will converge to a matrix as t gets large. This is analgous to the steady state of the system expressed in usual form. With a general initialization, though, one cannot show that M_t will converge to anything. Finally, note that we can recover $P_{i|i-1}$ by

$$P_{i|i-1} = P_{1|0} + \sum_{j=1}^{i} \Delta P_{j|j-1}, \quad i > 1. \quad (22)$$

3.1 Discussion

It is difficult to compute analytically the exact speed gain given by the Chandrasekhar Recursions given the differences between highly optimized linear algebra routines across architectures. Still, one can perform a crude assessment of the differences in the algorithms without resorting to purely empirical studies. We looked at all the matrix multiplications, including intermediate calculations, in the Kalman Filter and the Chandrasekhar Recursions, taking care to avoid unnecessary calculations, to gain insight into the differences in the two algorithms.

Table 1 lists the remaining matrix multiplication operations after the "common" operations have been canceled out. The two algorithms appear almost the mirror image of one another, with n_s and

n_y switched and a few additional operations for the Chandrasekhar recursions. Using naive linear algebraic calculations, the running time of two additional distinct operations for the Kalman Filter is $O(n_s^3)$ and $O(n_s^2 n_y)$. For the Chandrasekhar Recursions, the running times are $O(n_y^2 n_s)$ and $O(n_y^3)$. It is clear that if $n_s > n_y$, the Kalman Filter will have greater algorithmic complexity than the Chandrasekhar recursions. If $n_y < n_s$, the situation will be reversed.

Given that DSGE models feature more states than observables, the Chandrasekhar recursions seem a promising algorithm on this basis. However, the calculations in Table 1 are based on (1) a crude matrix multiplication accounting and (2) the naive matrix multiplication algorithmic complexity. Moreover, we have abstracted from matrix addition and transposes. We use four examples to give empirical guidance on the relative performance.

4 Four Examples

We compare the algorithm using four different models: a small Real Business Cycle Model, the Generic State Space model of Chib and Ramamurthy (2010), the Smets and Wouters (2007) model and the model of Schmitt-Grohe and Uribe (2010). For each of the models, we calculate the "wall" (clock) time it takes to evaluate the likelihood at a particular point in the posterior 1000 times. We normalized these times, with the fastest algorithm being normalized to 100. This is a crude comparison, but it gives a sense of the actual user experience running the algorithms. We compare three algorithms, the standard Kalman Filter, the Block Kalman Filter of Strid and Walentin (2009), and the Chandrasekhar Recursions. We implement all the algorithms in Intel Fortran 11.1 and Matlab 2010a.

We wrote code for the standard KF and the CR recursions, and used the code provided by Strid and Walentin (2009) for the Block Kalman Filter. This algorithm, specific for DSGE models, uses *a priori* knowledge of the structure of T – it has large matrix of zeros where the exogenous variables load onto the endogenous variables – and Z, which is often quite sparse, to build a fast Kalman filter. It requires the user to prestructure the model in a particular way and apply it on a case-by-case basis. We did not benchmark this for the Generic State Space model, since it is not a DSGE model.

The Fortran code utilizes Intel's Math Kernel Library implementation of `BLAS` and `LAPACK` fast linear algebra routines, including `dsymm`, symmetric matrix multiplication. The Matlab code uses a standard distribution of `BLAS` and does not consider symmetric matrix multiplication, which might disadvantage it somewhat. Both programs utilize multithreaded `BLAS`, using four threads for matrix operations.[3] All calculations were on a 12-core Intel Xeon x5670 CPU (2.93GHz) with L1, L2, and L3 cache sizes of 32K, 256K, 12288K, respectively.

One other technical detail worth mentioning is method of computing the solution to the discrete Lyapunov Equation. For the Chandrasekhar Recursions, it is crucial that this solution be (at least approximately) correct. Repeating trials suggests that for large (n_s greater than 100), the Matlab routine `dlyap` does not provide a good solution. Instead, the Matlab implementation of the CR uses `lyap_symm`, distributed as part of Dynare [Adjemian, Bastani, Juillard, Mihoubi, Ratto, and Villemot

[3]The results were broadly the same when only serial computation was considered.

(2011)], which yields a much better solution.

The equilibriums for the RBC and SW models are computed with Sims (2002) GENSYS. GENSYS is widely used to compute equilibriums of Linear Rational Expectations Models in economics. The algorithm uses the Generalized (complex) Schur decomposition, which gives it the advantage that "controls" and "states" don't have to be specified ahead of time; i.e., redundant states can be included. This means that for our examples, it is possible to reduce the size of the state vector and thus the efficacy of the CR. Given that our examples are of the small and medium size, we think that they are illustrative of speed gains for larger models.

4.1 Real Business Cycle Model

The first example is a simple Real Business Cycle model. There are $n_y = 2$ observables, labor and output. In the GENSYS formulation of the model, there are $n_s = 12$ states. The model is driven by two shocks, neutral technology and demand.

Table 2 reports the timings associated with evaluating the likelihood 1000 times, with the fastest normalized to 100. The language gain associated with Fortran is substantial, with the likelihood evaluation between four and 8 times faster than their Matlab counterparts. Within languages, that the Chandrasekhar recursions are the fastest in both languages. While, the Block KF outperforms the standard KF in Matlab, that is not the case in Fortran. The slow performance in Fortran is consistent with the findings of Strid and Walentin (2009), who find that the additional overhead associated with the increased number of matrix multiplications outweighs the size gain for small models.

4.2 Generic State Space Model

The next example is the Generic State Space model used in Chib and Ramamurthy (2010). This is not a DSGE model. It has no meaningful economic interpretation. It is, however, a good example, because there are *fewer* states than observables. In the model $n_s = 5$ while $n_y = 10$. Stochastic singularity is avoided by measurement error. The data used is simulated, with length 200 periods.

Table 3 shows the timings associated with evaluating the likelihood 1000 times, with the fastest normalized to 100.[4] The language gain is quite substantial, with Fortran being roughly four times faster for both algorithms. This large difference is mostly driven by the length of sample size. Concentrating on within-language performance, we see that there is a slight drop in speed when using the CR in Fortran. Relative to the standard KF, the CR about 17% percent slower. On the other hand, they are about 10% faster in Matlab. In both cases, the speed difference is small compared to the other examples.

4.3 Smets-Wouters Model

The Smets and Wouters (2007) is a medium-scale macroeconomic model that has become a benchmark for many issues related to the estimation and identification of DSGE models. In this formulation of the

[4]We again note that we do not use the Block KF because it is not a DSGE model.

model, there are $n_s = 50$ states and only $n_y = 7$ observables.[5] This example includes some redundant states, I but it is consistent with the conventional way of writing and estimating such models in economics. 50 states are not unusual for a medium-scale DSGE model, especially the kind used in central banks. For this reason, this example serves as a good benchmark.

Table 4 reports the timings associated with evaluating the likelihood 1000 times, with the fastest normalized to 100. It quite apparent that the language gain associated with Fortran relative Matlab is large, with the running time approximately doubled for each algorithm in Matlab compared to Fortran. Within a language, we see that the Chandrasekhar Recursions dominate both the Block and the standard KF. For Matlab, the CR offer algorithmic speed gains of 61% and 72%, for the Block KF and Standard KF respectively.

This speed gain is accomplished by eliminating of the matrix operation $TP_{t|t-1}T'$. Indeed, inspection of the Matlab Profiler indicates that this operation is about 45% of all filtering time for the Standard KF. For the Block KF, about 35% of the filtering time is spent on similar, slightly smaller matrix multiplication corresponding to the non-exogenous sub-state vector (which has $n_s - n_y = 43$ states.)

4.4 Schmitt-Grohe and Uribe (2010) Model

The final model comes from Schmitt-Grohe and Uribe (2010). The paper estimates a DSGE model augmented with "news" shocks. Specifically, they construct a real business cycle model with investment adjustment consts, capacity utilization, habits, and Jaimovich and Rebelo (2009) preferences. Each of the seven structural shocks is driven by three innovations. One of these innovations is unanticipated, while the other two are anticipated four and quarters ahead, respectively. The process for a given exogenous shock, z_t, looks like:

$$z_t = \rho_z z_{t-1} + \epsilon_t^{z,0} + \epsilon_{t-4}^{z,4} + \epsilon_{t-8}^{z,8}.$$

Writing this process recursively requires at least an additional eight states. In total, there are $n_s = 98$ states in the model and $n_y = 7$ observables, using quarterly data from 1965 - 2006, yielding 207 observations. Moreover, given the news structure, this model is not easily converted into a form used by Strid and Walentin. The comparison is thus restricted to the standard Kalman Filter and the Chandrasekhar Recursions.

Table 4 reports the timings associated with evaluating the likelihood 1000 times, with the fastest normalized to 100. Once again, the gain from using Fortran is substantial, regardless of the algorithm used. Once again, the Chandrasekhar Recursions dominate the Kalman Filter irrespective of language. Much like the Smets-Wouters model, much of the gain comes from eliminating the matrix operation $TP_{t|t-1}T$. Overall, the CR posts algorithmic gains of about 38% and 68% for Matlab and Fortran, respectively.

[5] We set the moving average coefficients on the markup shocks and automatic stabilizer on the government spending shock to zero. This way we can use the two block (AR), sparse Block Kalman Filter.

5 Conclusion

For DSGE models, the biggest bottleneck in computation is the evaluation of the log likelihood. Within this evaluation, the prediction of the state variance is the slowest operation, taking about half of all filtering time for large models. This paper has presented the Chandrasekhar recursions, a simple, fast algorithm for evaluating the likelihood of a Linear Gaussian State Space System. The CR algorithm works by eliminating the predicted state variance from the Kalman Filter equations altogether. In this way, it is ideally suited for DSGE models. The price paid to use this algorithm is relatively small. The system must be stationary and time invariant, assumptions that are typically satisfied for DSGE models.

It should be noted that this method is entirely consistent with other fast filtering techniques, such as Durbin and Koopman (2000). These additional speed gains are largely orthogonal to the ones presented here. Given the ease of implementation and apparent speed gains, the Chandrasekhar Recursions should become part of applied macroeconomists' computational toolkit.

References

ADJEMIAN, S., H. BASTANI, M. JUILLARD, F. MIHOUBI, M. RATTO, AND S. VILLEMOT (2011): "Dynare: Reference Manual, Version 4," Dynare Working Papers 1, CEPREMAP.

AKNOUCHE, A., AND F. HAMDI (2007): "Periodic Chandrasekhar recursions," *arXiv:0711.385v1*.

AN, S., AND F. SCHORFHEIDE (2007): "Bayesian Analysis of DSGE Models," *Econometric Reviews*, 26(2-4), 113–172.

CHIB, S., AND S. RAMAMURTHY (2010): "Tailored Randomized Block MCMC Methods with Application to DSGE Models," *Journal of Econometrics*, 155(1), 19–38.

DURBIN, J., AND S. KOOPMAN (2000): "Fast Filtering and Smoothing For Multivariate State Space Models," *Journal of Time Series Analysis*, 21, 281–296.

JAIMOVICH, N., AND S. REBELO (2009): "Can News about the Future Drive the Business Cycle?," *American Economic Review*, 9(4), 1097–1118.

KLEIN, A., G. MELARD, AND T. ZAHAF (1998): "Computation of the Exact Information Matrix of Gaussian Dynamic Regression Time Series Models," *The Annals of Statistics*, 26, 1636–1650.

MORF, M. (1974): "Fast Algorithms for Multivariate Systems," Ph.D. thesis, Stanford University.

MORF, M., G. SIDHU, AND T. KALAITH (1974): "Some new algorithms for recursive estimation in constant, linear, discrete-time systems," *IEEE Transactions on Automatic Control*, 19, 315–323.

SCHMITT-GROHE, S., AND M. URIBE (2010): "What's News in Business Cycles?," *Working Paper*.

SIMS, C. A. (2002): "Solving Linear Rational Expectations Models," *Computational Economics*, 20, 1–20.

SMETS, F., AND R. WOUTERS (2007): "Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach," *American Economic Review*, 97, 586–608.

STRID, I., AND K. WALENTIN (2009): "Block Kalman Filtering for Large-Scale DSGE Models," *Computational Economics*, 33, 277 – 304.

6 Tables

Table 1: Noncommon Matrix Multiplications

Algorithm	
Standard KF	(2x) $(n_s \times n_s)(n_s \times n_s)$, $(n_s \times n_s)(n_s \times n_y)$
Chandrasekhar Recursions	(3x) $(n_s \times n_y)(n_y \times n_y)$, (2x) $(n_y \times n_y)(n_y \times n_y)$

Table 2: RBC Model: Wall Time, Fastest Normalized to One Hundred

Method	Language	Wall Time
Standard KF	Matlab	982
Block KF	Matlab	616
Chandrasekhar Recursions	Matlab	410
Standard KF	Fortran	123
Block KF	Fortran	151
Chandrasekhar Recursions	Fortran	100

Table 3: Generic State Space Model: Wall Time, Fastest Normalized to One Hundred.

Method	Language	Wall Time
Standard KF	Matlab	499
Chandrasekhar Recursions	Matlab	451
Standard KF	Fortran	100
Chandrasekhar Recursions	Fortran	120

Table 4: Smets-Wouters Model: Wall Time, Fastest Normalized to One Hundred

Method	Language	Wall Time
Standard KF	Matlab	643
Block KF	Matlab	451
Chandrasekhar Recursions	Matlab	174
Standard KF	Fortran	249
Block KF	Fortran	214
Chandrasekhar Recursions	Fortran	100

Table 5: News Model: Wall Time, Fastest Normalized to One Hundred

Method	Language	Wall Time
Standard KF	Matlab	1263
Chandrasekhar Recursions	Matlab	787
Standard KF	Fortran	309
Chandrasekhar Recursions	Fortran	100

7 Appendix

7.1 Verification of recursions for F_t, K_t, and $K_{g,t}$.

For F_t:

$$\begin{aligned} F_t &= ZP_{t|t-1}Z' + H \\ &= ZP_{t|t-1}Z' + H + F_{t-1} - F_{t-1} \\ &= F_{t-1} + ZP_{t|t-1}Z' - ZP_{t-1|t-2}Z' \\ &= F_{t-1} + Z\Delta P_{t|t-1}Z'. \end{aligned}$$

K_t and $K_{g,t}$ are similar.

7.2 Verification of the difference equation for $\Delta P_{t|t-1}$.

To show that Chandrasekhar recursions work for the special case discussed above, we show how to write the recursions for F_t and K_t in terms of $\Delta(P_t)$.

Proof. Using the definition of F_t and adding and subtracting F_{t-1}, A similar algebraic manipulation

can be used for K_t. Note that we can also write $K_{g,t}$ as

$$K_{g,t} = (K_{g,t-1}F_{t-1} + T\Delta(P_t)Z')F_t^{-1}. \tag{23}$$

Proof of Lemma. From the Kalman Filter, we have that

$$P_{t+1|t} = TP_{t|t-1}T' + RQR - K_{g,t}F_tK_{g,t}'.$$

Subtracting $P_{t|t-1}$ from both sides, we have that

$$\Delta P_{t+1|t} = T\Delta P_{t|t-1}T' - K_{g,t}F_tK_{g,t}' + K_{g,t-1}F_{t-1}K_{g,t-1}.$$

Using the recursions for F_t shown in the previous lemma, we have

$$\Delta P_{t+1|t} = T\Delta P_{t|t-1}T' - K_{g,t}(F_{t-1} + Z\Delta P_{t-1|t}Z')K_{g,t}' + K_{g,t-1}F_{t-1}K_{g,t-1}.$$

Grouping like terms and completing the square for $(T - K_{g,t}Z)$, we have

$$\begin{aligned}\Delta P_{t+1|t} &= (T - Kg,tZ)\Delta P_{t|t-1}(T - K_{g,t}Z)' + K_{g,t}Z\Delta P_{t|t-1}T' + T\Delta P_{t|t-1}Z'K_{g,t}' \\ &- 2K_{g,t}\Delta P_{t|t-1}K_{g,t}' - K_{g,t}F_{t-1}K_{g,t}' + K_{g,t-1}F_{t-1}K_{g,t-1}'.\end{aligned} \tag{24}$$

Note that we can write the final product in the equation, using 23, 20, and tediously expanding terms, as,

$$\begin{aligned}K_{g,t-1}F_{t-1}K_{g,t-1} &= (K_{g,t}F_t - T\Delta P_{t|t-1}Z')F_t^{-1}(K_{g,t}F_t - T\Delta P_{t|t-1}Z') \\ &= (K_{g,t}(F_{t-1} + Z\Delta P_{t|t-1}Z') - T\Delta P_{t|t-1}Z')F_t^{-1}(K_{g,t}(F_{t-1} + Z\Delta P_{t|t-1}Z')T\Delta P_{t|t-1}Z') \\ &= (K_{g,t}F_{t-1} + (K_{g,t}Z - T)\Delta P_{t|t-1}Z')F_{t-1}^{-1}(K_{g,t}F_{t-1} + (K_{g,t}Z - T)\Delta P_{t|t-1}Z') \\ &= (T - K_{g,t}Z)\Delta P_{t|t-1}Z'F_{t-1}^{-1}Z\Delta P_{t|t-1}(T - K_{g,t}Z) + K_{g,t}F_{t-1}K_{g,t} \\ &+ K_{g,t}Z\Delta P_{t|t-1}(K_{g,t}Z - T)' + (K_{g,t}Z - T)\Delta P_{t|t-1}Z'K_{g,t}' \\ &= (T - K_{g,t}Z)\Delta P_{t|t-1}Z'F_{t-1}^{-1}Z\Delta P_{t|t-1}(T - K_{g,t}Z) + K_{g,t}F_{t-1}K_{g,t} \\ &- K_{g,t}Z\Delta P_{t|t-1}T' - T\Delta P_{t|t-1}Z'K_{g,t}' + 2K_{g,t}\Delta P_{t|t-1}K_{g,t}'.\end{aligned} \tag{25}$$

Combining 24 and 25, we have verified the lemma.