SUPPLEMENTARY RESULTS

Precision-based sampling for state space models that have no measurement error

Elmar Mertens*
Deutsche Bundesbank

June 2, 2023

Abstract

This online appendix provides results and descriptions that supplement our paper.

JEL Classification: C11, C32, C51

Keywords: State space models, signal extraction, Kalman filter and smoother, precision-based sampling, band matrix

*For correspondence: Deutsche Bundesbank, Wilhelm-Epstein-Strasse 14, 60431 Frankfurt am Main, Germany. elmar.mertens@bundesbank.de. Replication code is available at https://github.com/elmarmertens/ABCprecisionsampler. The views in this paper do not necessarily represent the views of the Deutsche Bundesbank or the Eurosystem. Any errors or omissions should be regarded as solely those of the author. Declarations of interest: none.

Contents

List of Tables

A Static representation of a model with VAR(p) state vector

For ease of exposition, the paper presents all state-variable dynamics as a VAR(1), while we also consider applications involving higher lag orders. For a recursive representation, as in equations (1) and (2) of the paper, the VAR(1) form is common, due to its Markovian character, and since higher order VAR's can easily be restated in companion form as VAR(1). However, in order to implement a precision-based sampler, it is important to explicitly state lag dependencies beyond the first lag when they exist. In principle, the static representation can also be constructed from the companion form of a VAR(p). But, as will be seen below, such a representation adds deterministic relationships to the state dynamics, so that the variance-covariance matrix of state shocks becomes singular and Assumption 1 of the paper is not satisfied.

Consider a state equation that is analogous to (1) in the paper, but featuring p lags:

$$x_t = \sum_{i=1}^{p} A_{i,t}\, x_{t-i} + B_t\, \varepsilon_t \tag{S.1}$$

A companion form representation, as used with the sampler of Durbin and Koopman (2002), proceeds by defining a companion vector, X_t, that comprises x_t and $p-1$ of its lags to obtain a system with VAR(1) dynamics:

$$X_t \equiv \begin{bmatrix} x_t \\ x_{t-1} \\ x_{t-2} \\ \vdots \\ x_{t-p+1} \end{bmatrix} = \underbrace{\begin{bmatrix} A_{1,t} & A_{2,t} & \dots & \dots & A_{p,t} \\ I & 0 & \dots & \dots & 0 \\ 0 & I & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & & I & 0 \end{bmatrix}}_{\mathcal{A}_t} X_{t-1} + \underbrace{\begin{bmatrix} B_t \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}}_{\mathcal{B}_t} \varepsilon_t \tag{S.2}$$

Of course, the larger p the larger the scale of the companion form and the larger the matrices that are to be handled by a recursive Kalman-filtering and smoothing algorithm.[1] Moreover, since $\mathcal{B}_t$ has not full row rank, it follows that $|\mathcal{B}_t\, \mathcal{B}_t'| = 0$ which also carries over to a block diagonal matrix $\boldsymbol{B}$ that consists of $\{\mathcal{B}_t\}_{t=1}^{T}$ as diagonal blocks. It follows that Assumption 1 of the paper would be violated if a static representation would be constructed from the companion form using $\mathcal{A}_t$ and $\mathcal{B}_t$, so that precision-based sampling would not applicable to this form.

Instead, a compact representation of (S.1) in static form is straightforward to construct, as shown also by Grant and Chan (2017a,b) and Chan, Poon, and Zhu (2023). Let $\boldsymbol{X}$ contain all values of $\{x_t\}_{t=1}^{T}$, and we can represent (S.1) in the form of (3) of the paper, $\boldsymbol{A}\,\boldsymbol{X} = \boldsymbol{X_0} + \boldsymbol{B}\,\boldsymbol{\varepsilon}$,

[1] The sampling approach of Durbin and Koopman (2002) avoids some of the companion form's increase in dimensionality with larger p, by focusing on projections of X_t on current and future y_t in its smoothing loop, instead of the analytically more convenient choices to project X_t onto X_{t+1} as in Carter and Kohn (1994). Nevertheless, the approach of Durbin and Koopman (2002) still needs to process the companion form matrices.

by utilizing the lower diagonal part of $\boldsymbol{A}$, which relates a given observation of x_t to its lags.[2] For ease of illustration, consider the case of $p = 2$, and we let:

$$\boldsymbol{A} = \begin{bmatrix} I & 0 & \dots & \dots & \dots & 0 \\ -A_{1,2} & I & 0 & \dots & & \vdots \\ -A_{2,3} & -A_{2,3} & I & \dots & \dots & \vdots \\ 0 & -A_{2,4} & -A_{2,4} & I & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & & -A_{2,T} & -A_{1,T} & I \end{bmatrix} \tag{S.3}$$

with $\boldsymbol{B}$ and $\boldsymbol{X_0}$ as illustrated in Section 2 of the paper.

B When measurement error affects only some observations

This section considers the mixed case when some observations are affected by measurement error and others not.[3] Specifically, we look at the following setting, written in static form:

$$\boldsymbol{Y_1} = \boldsymbol{C_1}\,\boldsymbol{X} + \boldsymbol{D_1}\boldsymbol{e_1}\,, \qquad \boldsymbol{e_1} \sim N(\boldsymbol{0}, \boldsymbol{I}) \tag{S.4}$$

$$\boldsymbol{Y_2} = \boldsymbol{C_2}\,\boldsymbol{X} \tag{S.5}$$

We maintain the state equation $\boldsymbol{A}\,\boldsymbol{X} = \boldsymbol{X}_0 + \boldsymbol{B}\,\boldsymbol{\varepsilon}$ as in (3) of the paper, and the prior is $\boldsymbol{X} \sim N(\boldsymbol{\mu_0}, \boldsymbol{P_0}^{-1})$. The only change is to have partitioned the measurement vector into two components, one afflicted by measurement error ($\boldsymbol{Y_1}$), whereas the other ($\boldsymbol{Y_2}$) is not.[4] Such a case might occur, for example, when data quality is assumed to have been changing over the sample due to use of historical long-range data as in Cogley and Sargent (2015), or when specific (sub-)periods display outliers or other features that a researcher might prefer to soak up with the measurement error term $\boldsymbol{e_1}$ while otherwise maintaining that observations are measured without error.

A convenient approach for deriving the posterior moments of $\boldsymbol{X}|(\boldsymbol{Y_1}, \boldsymbol{Y_2})$ is to proceed sequentially, and to build on elements from Algorithms 1 and 2 of the paper:

1. As in the case with measurement error, described in Section 2 of the paper, the posterior

[2] If desired, $\boldsymbol{X}$ may also including initial lags of x_t for $t = 0, -1, \dots, -p+1$ as illustrated in Section 2 of the paper for the case of $p = 1$.

[3] I would like to thank an anonymous reviewer for suggesting this extension.

[4] Of course, partitioning the measurement vector of the system written in static form does not pre-suppose a particular temporal order in which elements of $\boldsymbol{Y_1}$ occur relative to those of $\boldsymbol{Y_2}$, and neither have to occur as contiguous blocks over time.

after conditioning on $\boldsymbol{Y_1}$ is $\boldsymbol{X}|\boldsymbol{Y_1} \sim N(\boldsymbol{\mu_1}, \boldsymbol{P_1}^{-1})$ with

$$\boldsymbol{P_1} = \boldsymbol{P_0} + \boldsymbol{C_1}'(\boldsymbol{D_1}\boldsymbol{D_1}')^{-1}\boldsymbol{C_1}\,, \tag{S.6}$$

$$\text{and} \quad \boldsymbol{P_1}\,\boldsymbol{\mu_1} = \boldsymbol{P_0}\,\boldsymbol{\mu_0} + \boldsymbol{C_1}'(\boldsymbol{D_1}\boldsymbol{D_1}')^{-1}\boldsymbol{Y_1} \tag{S.7}$$

2. Apply Alorithm 2 of the paper, with $\boldsymbol{P_1}$ and $\boldsymbol{\mu_1}$ as prior precision and mean, and equation (S.5) as measurement equation. (This involves applying the QR decomposition to $\boldsymbol{C_2}$.)

C Likelihood computations

In principle, the precision-based methods described in the paper could also be used to compute the likelihood of the data vector (conditional on the state space matrices $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Omega}$, and $\boldsymbol{X}_0$). The likelihood can be computed based on

$$p(\boldsymbol{Y}) = |\boldsymbol{R_1}|^{-1} p(\boldsymbol{X}_1)\,, \tag{S.8}$$

$$\text{with} \quad \boldsymbol{X}_1 \sim N\left(\boldsymbol{Q}_1\boldsymbol{\mu_0},\ \tilde{\boldsymbol{\Sigma}}_{11}\right)\,, \tag{S.9}$$

$$\text{and} \quad \tilde{\boldsymbol{\Sigma}}_{11}^{-1} = \tilde{\boldsymbol{P}}_{11} - \tilde{\boldsymbol{P}}_{21}'\tilde{\boldsymbol{P}}_{22}^{-1}\tilde{\boldsymbol{P}}_{21}\,, \tag{S.10}$$

where the last line follows from the algebra for inverting partitioned matrices.

For faster computations based on sparse structures, the Choleski factorization of $\tilde{\boldsymbol{\Sigma}}_{11}^{-1}$ can be obtained directly as the bottom right block of the Choleski factorization of the following precision matrix (note the rearranged partitions relative to (18) in the paper). That means, we have:

$$\left(\begin{bmatrix} \tilde{\boldsymbol{P}}_{22} & \tilde{\boldsymbol{P}}_{21} \\ \tilde{\boldsymbol{P}}_{12} & \tilde{\boldsymbol{P}}_{11} \end{bmatrix}\right)^{-1} = \begin{bmatrix} \boldsymbol{L}_{11} & \boldsymbol{0} \\ \boldsymbol{L}_{12} & \boldsymbol{L}_{22} \end{bmatrix} \qquad \text{with} \quad \boldsymbol{L}_{22} = \tilde{\boldsymbol{\Sigma}}_{11}^{-1/2}\,. \tag{S.11}$$

However, even with the use of sparse routines, the computations are slower than use of the conventional forward recursions of the Kalman filter; see the replication set provided alongside this article.

D Additional details on the trend-inflation model

This section provides a few additional details about the implementation of the trend-inflation model in Section 6 of the paper. The model follows closely the baseline specification of Mertens (2016) with stochastic volatility (SV) affecting shocks to the common trend and a homoscedastic VAR(p) to describe deviations in all variables from the common trend.

Here, we describe the static representation of the model, priors for parameters and initial values and further details on the MCMC steps.

For convenience, we restate the model equations as shown in the paper: Formally, the model takes the following from:

$$y_t = C_t x_t + y_0\,, \qquad x_t = \begin{bmatrix} \bar{y}_t \\ \tilde{y}_t \end{bmatrix}\,, \qquad \bar{y}_t = \bar{y}_{t-1} + \bar{\varepsilon}_t\,, \qquad \tilde{y}_t = \sum_{k=1}^{p} \tilde{A}_k \tilde{y}_{t-k} + \tilde{\varepsilon}_t \tag{23}$$

$$C_t = \begin{bmatrix} 1 & I \end{bmatrix} \qquad \bar{\varepsilon}_t \sim N(0, \bar{\sigma}_t^2)\,, \qquad \tilde{\varepsilon}_t \sim N(0, \tilde{\Sigma}) \tag{24}$$

where $\bar{y}_t$ is a scalar, $\tilde{y}_t$ is a $N_y \times 1$ column vector, and the VAR for $\tilde{y}_t$ is assumed stable eigenvalues of its companion form matrix are inside the unit circle). The constant y_0 is a $N_y \times 1$ vector of intercepts that captures constant differences in inflation rates from different price baskets, or constant biases in survey expectations.[5] The specification for C_t in (24) assumes all N_y variables are observable at time t; in case of missing data for a given t, the corresponding rows are to be omitted from C_t. As in the baseline version of the model in Mertens (2016), shocks to the trend are assumed to be affected by a scalar stochastic volatility (SV) process:

$$\log \bar{\sigma}_t^2 = \log \bar{\sigma}_{t-1}^2 + \eta_t\,, \qquad \eta_t = N(0, \sigma_\eta^2)\,. \tag{25}$$

[5] Only the sum of y_0 and the initial level of the trend component $\bar{y}_0$ is identified in this case.

D.1 State space representation of the model

For the precision-based sampler, we represent the model in static form after constructing a lag polynomial for x_t and then proceeding as described in Section A:

$$X_t = \sum_{k=1}^{p} A_k X_{t-k} + \begin{bmatrix} \bar{\varepsilon}_t \\ \tilde{\varepsilon}_t \end{bmatrix} \tilde{\varepsilon}_t \,, \qquad A_1 = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_1 \end{bmatrix} \,, \qquad A_i = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_i \end{bmatrix} \; \forall \, 1 < i \leq p \tag{S.12}$$

For the DK sampler, we set up a companion form (as discussed in Section A). The only twist is that we need to track only the current value of $\bar{y}_t$ but p lags (including the current value) of $\tilde{y}_t$:

$$X_t = \begin{bmatrix} \bar{y}_t \\ \tilde{y}_t \\ \tilde{y}_{t-1} \\ \vdots \\ \tilde{y}_{t-p+1} \end{bmatrix} = \mathcal{A} X_{t-1} + \begin{bmatrix} \bar{\varepsilon}_t \\ \tilde{\varepsilon}_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} \,, \qquad \mathcal{A} = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & \tilde{A}_1 & \tilde{A}_2 & \dots & \dots & \tilde{A}_p \\ 0 & I & 0 & \dots & \dots & 0 \\ \vdots & 0 & I & 0 & \dots & 0 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & & I & 0 \end{bmatrix} \,. \tag{S.13}$$

D.2 Priors

Here we describe priors for parameters and initial values of the model:

- Initial values for trend and cycle: $\bar{y}_0 \sim N(0, 10^8)$ and $\tilde{y}_k \sim N(0, 100 \cdot I) \quad \forall \; (-p) < k < 1$.
- Initial value of SV process: $\log \bar{\sigma}_0^2 \sim N(0, 100)$
- VAR parameters: We define the vector of VAR intercepts $a_0 = \left(\sum_i \tilde{A}_i \right) y_0$, and use a jointly-normal prior over $\{a_0, A_1, A_2 \ldots, A_p\}$ similar to the Minnesota prior. Under the prior, all coefficients are independent from each other, and mean zero. Only their prior variances differ. For an element of a_0, prior variance is 100. For a typical element of $\tilde{A}_i$ we have $\text{Var}\left([\tilde{A}_i]_{k,j} \right) = \theta_1 \cdot \theta_2 / i^{\theta_3}$ with $\theta_1 = 0.2^2$, $\theta_2 = 0.5^2$, and $\theta_3 = 4$.[6]
- Variance-covariance matrix for the shocks to the cyclical component: $\tilde{\Sigma} \sim \mathcal{W}^{-1}(I, N_y + 2)$.
- Variance of shocks to the SV process: $\bar{\sigma}^2 \sim IG(3/2, 1/2)$.

[6]The lag decay, θ_3 is set to a somewhat more aggressive value than typical to keep parameter draws from piling up too closely to values that would imply an unstable VAR.

D.3 MCMC steps

As described in the paper, the model is estimated with an MCMC sampler. Here we provide a few additional details. We collect all model parameters into the vector $\boldsymbol{\theta}$, and the paths for linear states and SV into $\boldsymbol{X}$ and $\bar{\boldsymbol{\sigma}}$; all observed data points are collected into $\boldsymbol{Y}$.[7] An MCMC sampler to estimate the model then iterates over the following blocks:

1. Draw linear states from $p(\boldsymbol{X}|\boldsymbol{\theta}, \bar{\boldsymbol{\sigma}}, \boldsymbol{Y})$ which can be done with the precision-based ABC sampler or the DK sampler, after adjusting the measurements by their variable-specific intercepts: $y_t^o = y_t - y_0$.

2. Model parameters $p(\boldsymbol{\theta}|\boldsymbol{X}, \bar{\boldsymbol{\sigma}}, \boldsymbol{Y})$ broken down into the following steps:[8]

   2 a) VAR parameters $p(a_0, A_1, A_2 \ldots, A_p|\bar{\boldsymbol{\sigma}}, \boldsymbol{\theta}^*, \boldsymbol{Y})$ which is a regression system with standard Bayesian update.

   2 b) Draw the variance-covariance matrix of the cyclical shocks $p(\tilde{\Sigma}|\bar{\boldsymbol{\sigma}}, \boldsymbol{\theta}^*, \boldsymbol{Y})$, which is an inverse-Wishart update based on the VAR residuals obtained from the previous step.

   2 c) Inverse-gamma update for $p(\bar{\sigma}^2|\bar{\boldsymbol{\sigma}}, \boldsymbol{\theta}^*, \boldsymbol{Y})$ based on the SV shocks implied by $\bar{\boldsymbol{\sigma}}$.

3. Draw the SV path $p(\boldsymbol{\theta}|\boldsymbol{X}, \bar{\boldsymbol{\sigma}}, \boldsymbol{Y})$ using the samplers of Kim, Shephard, and Chib (1998), and Omori, et al. (2007). These samplers rely on a mixture representation of the SV process, and we incorporate the advice of Del Negro and Primiceri (2015) regarding the proper order of steps to draw from the mixture states and other elements of model.[9]

[7] The model parameters comprise the cyclical VAR coefficients, $\tilde{A}_k$, the variable-specific intercepts y_0, the variance matrix $\tilde{\Sigma}$ and the SV-shock variance σ_η^2.

[8] For each step, we denote the remainder of the parameter vector by $\boldsymbol{\theta}^*$.

[9] The SV samplers of Kim, Shephard, and Chib (1998), and Omori, et al. (2007) employ an approximately linear state space representation of the SV process, draws of which can be obtained also with either Kalman- or precision-based samplers. For the implementation used here, a precision-based sampler is used, also when using a DK sampler in step 1. Our implementation also follows the advice of Del Negro and Primiceri (2015) for appropriately embedding the approximate sampler of Kim, Shephard, and Chib (1998) into a latent-variable setup like ours.

E Additional Results

The paper presented results for the application of a common trend model with VAR(p) deviations between data and trend, that were generated in single-threaded mode on a Windows client with Intel Xeon cores. The remainder of this appendix presents additional results that consider a) variations in computational environment and b) a different application, namely the case of a multivariate trend model (with N_y trends instead of a common trend, as described in Section 3.1 of the paper and a VAR(p) model with missing values. Results for the common trend case are shown in Section E.1, for the multivariate trend case in Section E.2, and for the missing-value VAR in Section E.3 below.

All simulations are carried out in MATLAB and replication code is available at `https://github.com/elmarmertens/ABCprecisionsampler`. To reliably measure execution times of the different samplers, the MATLAB function `timeit` is used.[10]. For each application, the variations in computational environments considered include multi-threaded, instead of single-threaded mode as well as alternative platforms (macOS running on Intel i7 chips with up to four threads, or macOS/Rosetta running on Apple Silicon M1 Pro chips with up to 8 threads). For each simulation, VAR parameters are randomly drawn from a Minnesota prior (centered at mean-zero coefficient values), and other model parameters (like shock loadings) are randomly drawn from standard normals.

Multi-threaded mode enables MATLAB to perform certain computations, like the column-wise processing of matrix multiplications, on different cores. Since multi-threading benefits relatively large matrix operations more strongly, it stands to boost more the precision-based calculations made in static form. In addition, more modern chip sets seem to generally do better equipped at boosting multi-threaded performance (for example, compare results between an older Intel i7 and a more recent Apple Silicon chip, both running on macOS).

Considered in isolation, multi-threading should be the preferred choice for improved execution times. But, the benefits from multi-threaded mode are from linear in the number of scores (due to the only intermittent use of parallelized matrix operations). Consequently, for applications involving multiple MCMC chains, it is generally more efficient to parallelize execution of the MCMC chains themselves (with each chain executed in single-threaded mode). Doing so achieves a more continuous computational load per thread as opposed to the use of multi-threaded matrix-operation while running one MCMC chain after another.[11] As such, it is also instructive to gauge the performance of different samplers in single-threaded mode.

[10]The `timeit` function measures a median execution time after conducting multiple calls designed to average out factors such as varying CPU loads, overhead due to just-in-time compilation etc. For details, see `https://www.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html` and and `https://www.mathworks.com/help/matlab/ref/timeit.html`.

[11]Of course, depending on available resources, a researcher might also be in a position to deploy multiple chains in parallel (say, distributed across different computers) and still execute operations within each chain in multithreaded mode. The optimal configuration then critically depends on problem dimensions and computational assets.

E.1 Common trend model

The paper reports results of comparisons between precison-based sampling and the sampler of Durbin and Koopman (2002), henceforth "DK," as applied to a common trend model that is similar in structure to models used by Mertens (2016), Del Negro, et al. (2017; 2019), as described in Section 3.1 of the paper, with the defining feature for the common trend specification being that Λ is set to a unit vector. The comparisons shown in the paper were generated on a Windows virtual client with Intel Xeon CPU in single-threaded mode. To complement these results, Tables S.2– S.5 report additional comparisons generated on different computational platforms, with Table S.1 restating results shown in Table 1 of the paper. Further details on the computational environments are provided in the notes to each table.

Across different platforms, similar performance patterns emerge as described in the paper for a Windows machine with 32 Intel Xeon cores: The precision-based ABC sampler (Panels A) and the trend-cycle sampler (Panels C) outperform the DK sampler by a wide margin. Even when one-off computations like the QR-step are included in the times measured for the precision-based ABC sampler (Panels B) it performs typically better than the DK sampler or a noise-augmented precision-based sampler (Panels D). Nevertheless, depending on model configurations (p, N_y, T), relative performance can be sensitive to the choice of singe- or multi-threaded computations, even up to the point of overturning the performance ranking between selected sampler. For example, in single-threaded mode, and for small values of $p = 4$ or 8, the DK can even outperform the application of the noise-augmented short-cut of a conventional precision-based sampler (see Panel D of Table S.2) which is not the case with multi-threading.

A further example is the comparison between the precision-based ABC sample *inclusive of the QR step* and the DK sampler. As reported in Panel B of Table S.2), for some configurations (few lags, $p = 8$, and many observations, $N_y \times T$ high) the ABC sampler outperforms DK only when prepared one-off computations (the QR step in particular) are used (as in Panel A); otherwise, the additional overhead can lead to execution times that exceed those of the DK sampler by up to 25 percent (see Panel B). Having said that, for those cases with $p = 4$, a single draw from the DK sampler takes only a small fraction (often below a fifth) of a second so that, in absolute terms, the execution times between DK and ABC sampler, even with the QR step) remain very close.

E.2 Multivariate trend model

Here we provide simulation results for a multivariate trend model, with as many trend components as there are observables N_y, as described in Section 3.1 of the paper with $\Lambda = I$. Tables S.8– S.9 report simulation results obtained from different computational platforms, with details on the computational environments provided in the notes to each table.

For the multivariate trend model with N_y trends, precision-based samplers still perform generally well, when executed in multi-threaded mode. As before, the "noise-augmented"

short-cut reported in Panel D of each table is less effective than the exact implementations of Panels A (ABC sampler excl. QR step) and C (trend-cycle sampler). When executed in single-threaded mode, Panels A and C continue to show precision-based samplers outperforming DK, but only for models with many VAR lags ($p \geq 12$) in the process for the cyclical component. When computed on an Apple Silicon chip, the precision-based samplers do better for $p \geq 8$ compared to DK (in single-threaded mode). When multithreading on a Windows Xeon machine with 32 cores, precision-based sampling cuts execution times down to a third and less (compared to DK; see Table S.7), and even down to a fifth and less when executed on an Apple Silicon chip (see Table S.9) with less impressive gains on an Intel i7 (see Table S.11).

E.3 Missing-value VAR

As further illustration of the methods described in the paper, this (sub-)section considers the application of a VAR(p) model with missing values. Such applications may arise for various reasons, like the exclusion of extreme values — as in Carriero, et al. (2022) — or in case of mixed-frequency data as in Schorfheide and Song (2015), and Chan, Poon, and Zhu (2023). Specifically, in our simulations, a given percentage of randomly assigned data points is considered to be unobserved. We simulate cases with small and large shares of missing observations, ranging from 1% - 90% of all $N_y \cdot T$ data points are unobserved. Smaller shares of missing values correspond to application where observations are treated as missing due to rare occasions of limited data availability or in case of outlier observations, as considered, for example, by Carriero, et al. (2022). Subsequently presented tables report similar performance gains also for higher shares of missing observations, that could arise, for example, in a mixed-frequency VAR. Moreover, we consider similar variations in computational platforms as described in the introduction to this section. Tables S.21–S.20 report results generated on a Windows virtual client with Intel Xeon processor (for various shares of missing values as well as for single- and multi-threaded modes), Tables S.37– S.36 provide corresponding results generated on macOS with an Intel i7 processor.

As in the common trend and multivariate trend cases, each table compares execution inclusive and exclusive of the QR step involved in precision-based sampling. (In the missing-value case, the QR step would have to be evaluated only once by a researcher employing the sampler as part of a MCMC model estimation.) Furthermore, Panel C of each table reports results for using a variant of the DK sampler, that is specialized to the missing value case. Generally, as described in the paper, the measurement loadings of the missing-value application, C_t, are time varying, but consist solely of ones and zeros that pick specific rows of the state vector. The DK sampler can be adapted to this case by replacing matrix multiplications involving C_t by direct references to specific rows of objects pertaining to the state vector (like the transition matrix A), and Panel C of the tables documents that there are some gains from this approach relative to DK, but far below of what can be obtained with a precision-based sampler. Moreover, the per-

formance ranking between the standard DK sampler (in Panel D of each table) and the version specialized to the missing-value case is not always uniform. In single-threaded mode, the specialized version displays some weakness (and can even underperform) for models with large p and N_y (as the companion form grows), whereas the pattern is reversed in multi-threaded mode (which boosts larger matrix multiplications). All told, these results suggest that the most thorough perform optimization should account for the specifics of the computational environment as well as the nature of the underlying analytic problem.

Table S.1: Common trend model on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	14	10	35	20	27	12	30	31	25	24
8	7	14	14	13	13	14	15	12	12	13
12	10	10	11	10	11	13	8	9	9	8
24	6	6	7	3	3	5	5	5	2	2
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	29	26	70	44	62	30	70	70	69	61
8	25	31	31	30	34	37	33	28	28	28
12	21	24	24	21	23	30	19	22	20	20
24	11	12	13	6	6	10	11	11	5	6
PANEL C: Trend-Cycle PS as percentage of DK										
4	12	9	27	16	21	13	26	27	24	21
8	9	13	12	13	12	15	13	12	12	11
12	7	9	9	9	8	10	7	9	10	8
24	4	5	5	2	2	4	5	5	2	2
PANEL D: standard PS w/noise as percentage of DK										
4	26	43	96	54	66	47	72	75	65	55
8	18	43	40	38	42	42	38	32	34	40
12	31	29	29	31	29	36	21	25	27	27
24	16	17	17	8	10	14	15	16	8	11
PANEL E: DK in seconds										
4	0.02	0.05	0.06	0.16	0.20	0.05	0.13	0.23	0.44	0.78
8	0.04	0.11	0.27	0.42	0.61	0.12	0.42	1.02	1.57	2.35
12	0.06	0.27	0.49	0.76	1.18	0.22	1.00	1.77	2.95	4.50
24	0.27	0.78	1.57	5.84	7.68	1.00	2.82	5.99	22.36	30.71

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.2: Common trend VAR on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	16	32	50	53	58	15	45	51	55	55
8	17	26	27	28	28	24	25	26	27	26
12	12	19	18	18	16	18	19	17	17	15
24	11	10	9	5	5	10	9	7	4	3
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	46	79	109	111	114	40	106	118	125	119
8	41	57	56	55	54	54	55	57	58	53
12	28	40	35	34	31	39	40	37	34	30
24	21	18	16	9	8	22	18	14	8	7
PANEL C: Trend-Cycle PS as percentage of DK										
4	14	28	43	46	49	12	38	47	49	52
8	14	23	24	23	23	20	22	26	25	25
12	12	17	15	15	14	15	16	16	16	15
24	6	7	6	3	3	9	8	6	4	3
PANEL D: standard PS w/noise as percentage of DK										
4	50	130	172	206	213	52	145	173	200	215
8	57	99	121	128	139	78	100	116	130	141
12	53	81	88	98	99	61	83	86	98	98
24	45	51	52	34	35	45	53	52	34	34
PANEL E: DK in seconds										
4	0.01	0.02	0.04	0.08	0.13	0.05	0.09	0.18	0.31	0.50
8	0.02	0.07	0.16	0.30	0.48	0.08	0.28	0.65	1.18	2.00
12	0.04	0.15	0.37	0.69	1.25	0.16	0.59	1.52	2.80	5.05
24	0.14	0.66	1.81	5.95	11.03	0.57	2.58	7.60	24.11	46.06

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.3: Common trend VAR on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	18	28	39	42	40	18	28	35	47	39
8	18	19	20	22	25	16	19	20	21	22
12	19	13	15	16	15	14	14	15	14	14
24	9	9	7	8	7	6	8	8	7	7
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	48	82	100	102	97	46	74	86	114	92
8	45	56	49	51	54	40	47	47	46	49
12	39	32	35	35	32	36	31	34	30	30
24	20	18	15	15	13	15	17	17	15	14
PANEL C: Trend-Cycle PS as percentage of DK										
4	15	27	31	34	34	15	28	37	44	40
8	16	17	17	21	20	15	19	20	18	23
12	18	13	14	14	13	14	13	13	12	12
24	8	7	6	6	5	7	7	7	6	6
PANEL D: standard PS w/noise as percentage of DK										
4	44	80	98	137	135	48	89	124	151	137
8	42	58	65	87	100	42	66	76	78	100
12	46	41	55	68	67	42	50	54	61	62
24	25	34	35	44	45	24	30	38	45	47
PANEL E: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.07	0.13	0.19	0.35
8	0.02	0.04	0.09	0.16	0.25	0.06	0.20	0.44	0.81	1.13
12	0.02	0.09	0.19	0.35	0.64	0.09	0.43	0.88	1.78	2.94
24	0.09	0.33	0.92	1.75	3.21	0.43	1.56	3.79	7.04	12.64

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.4: Common trend VAR on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	18	41	61	69	77	23	49	67	72	76
8	26	28	31	35	35	24	30	34	30	31
12	22	21	20	21	19	21	22	19	19	19
24	12	11	9	9	8	12	9	8	7	7
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	53	117	149	149	151	60	120	153	157	158
8	69	71	71	70	66	62	70	76	65	61
12	54	47	44	42	36	50	49	40	38	36
24	26	22	17	15	14	27	19	16	14	13
PANEL C: Trend-Cycle PS as percentage of DK										
4	15	36	48	55	63	18	47	65	67	73
8	19	24	25	26	28	22	30	34	30	30
12	22	21	19	19	17	20	21	18	18	18
24	11	9	7	6	6	11	9	8	7	6
PANEL D: standard PS w/noise as percentage of DK										
4	58	156	230	282	296	71	176	249	276	295
8	78	115	145	176	181	83	127	159	153	166
12	77	92	111	125	115	80	103	103	112	118
24	50	60	58	62	63	54	57	61	64	65
PANEL E: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.06	0.11	0.20	0.32
8	0.01	0.05	0.10	0.20	0.35	0.06	0.19	0.42	0.90	1.51
12	0.02	0.10	0.25	0.50	0.99	0.10	0.40	1.14	2.23	3.90
24	0.10	0.47	1.46	3.10	5.67	0.39	2.11	5.70	12.12	22.60

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.5: Common trend VAR on Apple Silicon (macOS, 8 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	10	17	15	18	13	6	11	13	17	11
8	10	10	8	9	8	7	8	7	6	7
12	8	5	6	6	5	6	5	4	5	5
24	4	4	3	3	3	3	3	2	2	2
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	22	34	31	36	25	11	24	28	34	22
8	18	19	15	16	15	14	15	14	12	12
12	14	10	10	10	9	11	9	8	9	8
24	6	6	5	4	4	5	5	4	4	3
PANEL C: Trend-Cycle PS as percentage of DK										
4	9	16	15	18	13	4	11	13	16	11
8	9	10	8	8	8	6	8	7	7	7
12	8	6	6	6	5	5	5	5	5	5
24	3	3	2	2	2	3	3	2	2	2
PANEL D: standard PS w/noise as percentage of DK										
4	22	40	41	51	41	11	27	36	49	35
8	22	29	27	33	34	16	23	25	24	30
12	20	19	22	26	24	15	17	18	22	23
24	12	14	14	15	16	10	12	13	14	15
PANEL E: DK in seconds										
4	0.01	0.02	0.05	0.07	0.15	0.10	0.12	0.20	0.28	0.66
8	0.02	0.06	0.16	0.26	0.44	0.11	0.29	0.66	1.31	1.93
12	0.04	0.16	0.33	0.56	0.99	0.18	0.68	1.51	2.48	4.00
24	0.16	0.54	1.40	2.80	4.78	0.68	2.31	5.78	11.20	21.08

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Apple M1 Pro (macOS, Rosetta 2) in multi-threaded mode (for matrix operations) using 8 threads.

Table S.6: Common trend VAR on Apple Silicon (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	13	33	39	40	38	17	32	38	39	37
8	16	20	18	17	16	17	19	18	15	14
12	14	13	11	10	8	14	12	9	8	8
24	9	6	4	4	3	7	4	3	3	2
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	36	70	76	75	68	39	68	75	75	68
8	34	39	33	30	26	37	38	33	26	24
12	28	24	19	16	13	29	23	16	14	13
24	16	9	7	6	5	13	7	6	5	4
PANEL C: Trend-Cycle PS as percentage of DK										
4	12	31	37	37	37	14	30	36	38	37
8	14	19	17	16	14	15	19	17	14	13
12	15	13	11	9	8	13	11	9	8	7
24	7	4	3	2	2	6	4	3	3	2
PANEL D: standard PS w/noise as percentage of DK										
4	36	107	148	169	172	48	107	147	172	173
8	55	90	94	98	99	54	86	94	87	91
12	55	65	66	68	63	52	63	58	61	62
24	35	35	32	32	31	35	32	32	32	31
PANEL E: DK in seconds										
4	0.01	0.02	0.03	0.06	0.10	0.03	0.06	0.13	0.23	0.40
8	0.02	0.05	0.13	0.26	0.48	0.06	0.20	0.51	1.16	2.10
12	0.03	0.12	0.33	0.72	1.45	0.10	0.48	1.51	3.17	5.85
24	0.11	0.67	2.21	4.98	9.57	0.45	2.98	8.92	19.99	40.30

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Apple M1 Pro (macOS, Rosetta 2) in single-threaded mode.

Table S.7: Multivariate trend model on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	37	56	64	58	54	56	44	51	50	45
8	31	51	36	41	43	42	42	32	34	39
12	22	23	26	34	33	24	21	26	30	29
24	15	19	9	9	11	11	16	7	9	11
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	66	96	98	106	92	81	96	101	92	80
8	63	83	59	64	72	63	70	55	58	64
12	42	42	48	53	52	43	35	45	47	48
24	20	27	13	14	16	21	26	11	13	16
PANEL C: Trend-Cycle PS as percentage of DK										
4	33	39	53	50	47	37	39	50	46	39
8	20	34	30	31	37	35	33	28	31	36
12	18	21	22	26	29	21	18	23	26	26
24	9	15	7	8	8	10	14	7	8	10
PANEL D: standard PS w/noise as percentage of DK										
4	49	68	76	75	68	65	71	73	73	64
8	42	62	49	55	57	54	50	42	49	52
12	42	29	41	44	46	35	28	36	42	42
24	19	23	11	13	14	15	24	10	12	15
PANEL E: DK in seconds										
4	0.01	0.05	0.10	0.18	0.29	0.05	0.21	0.39	0.71	1.20
8	0.03	0.10	0.29	0.46	0.65	0.12	0.43	1.17	1.84	2.69
12	0.06	0.28	0.50	0.78	1.17	0.24	1.11	2.05	3.19	4.91
24	0.25	0.71	3.50	5.39	7.89	1.01	2.84	13.76	21.94	30.87

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.8: Multivariate trend model on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	44	99	128	160	160	62	101	127	160	163
8	51	85	99	112	115	67	86	104	111	113
12	36	69	77	87	86	47	68	78	87	87
24	36	44	29	33	33	34	46	29	34	34
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	81	154	188	228	220	104	162	191	229	225
8	90	122	134	148	145	110	127	137	147	145
12	59	94	99	109	104	72	95	103	110	105
24	49	55	35	39	38	49	58	35	39	38
PANEL C: Trend-Cycle PS as percentage of DK										
4	38	77	112	142	144	45	89	116	146	150
8	42	73	88	99	103	56	77	94	101	105
12	32	60	67	77	77	40	61	71	81	81
24	26	37	25	29	29	30	41	27	31	31
PANEL D: standard PS w/noise as percentage of DK										
4	56	124	148	185	182	81	119	148	186	187
8	68	98	113	129	127	80	99	114	124	125
12	55	78	86	96	94	55	76	87	97	95
24	42	49	32	36	36	39	51	32	36	36
PANEL E: DK in seconds										
4	0.01	0.04	0.07	0.12	0.20	0.05	0.14	0.29	0.47	0.81
8	0.02	0.09	0.20	0.37	0.64	0.09	0.34	0.80	1.49	2.62
12	0.05	0.17	0.44	0.82	1.52	0.20	0.72	1.74	3.32	6.08
24	0.15	0.73	3.25	6.13	11.64	0.65	2.86	13.14	25.00	46.28

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.9: Multivariate trend model on Apple Silicon (macOS, 8 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	27	31	34	23	29	24	26	31	22	30
8	20	26	21	24	25	16	23	17	19	23
12	16	14	19	21	18	12	12	16	14	20
24	10	12	13	13	15	7	10	13	13	14
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	48	53	54	35	40	42	45	49	40	50
8	33	39	29	32	34	26	34	27	32	37
12	24	20	25	27	25	19	17	24	21	29
24	13	15	17	17	20	10	14	18	17	17
PANEL C: Trend-Cycle PS as percentage of DK										
4	26	32	34	23	28	22	25	30	21	29
8	19	25	21	23	24	15	22	17	19	23
12	15	14	18	21	18	12	12	15	13	18
24	9	11	11	12	13	7	10	11	12	13
PANEL D: standard PS w/noise as percentage of DK										
4	43	52	56	39	48	39	47	53	37	53
8	31	44	34	38	38	27	42	29	32	35
12	24	24	31	39	29	20	21	27	22	32
24	15	20	19	19	21	12	17	21	23	21
PANEL E: DK in seconds										
4	0.01	0.03	0.06	0.18	0.25	0.04	0.12	0.26	0.75	0.94
8	0.02	0.07	0.22	0.38	0.62	0.10	0.29	1.01	1.79	2.68
12	0.05	0.21	0.39	0.69	1.33	0.19	0.89	1.72	4.21	5.35
24	0.16	0.57	1.51	2.93	4.80	0.72	2.58	6.56	13.60	23.01

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Apple M1 Pro (macOS, Rosetta 2) in multi-threaded mode (for matrix operations) using 8 threads.

Table S.10: Multivariate trend model on Apple Silicon (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	34	86	105	122	116	44	84	108	124	119
8	50	73	72	77	74	45	71	71	70	68
12	41	51	53	55	50	40	52	49	51	51
24	28	29	27	27	27	28	27	28	27	27
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	60	129	142	157	144	75	126	146	174	157
8	75	96	88	92	87	71	93	92	91	84
12	59	65	62	63	57	59	65	60	62	61
24	36	34	31	30	30	36	33	33	31	29
PANEL C: Trend-Cycle PS as percentage of DK										
4	30	83	102	120	115	39	82	104	122	118
8	46	69	69	76	71	42	67	69	68	66
12	39	50	50	53	47	37	49	47	50	49
24	25	26	25	25	25	26	26	26	26	25
PANEL D: standard PS w/noise as percentage of DK										
4	49	115	131	147	139	62	111	134	149	141
8	62	89	84	88	84	60	87	85	81	77
12	52	63	61	63	56	51	62	56	58	59
24	35	35	31	30	30	34	31	32	32	30
PANEL E: DK in seconds										
4	0.01	0.02	0.05	0.09	0.17	0.04	0.08	0.19	0.35	0.67
8	0.02	0.06	0.17	0.35	0.67	0.06	0.24	0.69	1.52	2.91
12	0.03	0.14	0.42	0.89	1.84	0.12	0.57	1.83	3.85	7.26
24	0.12	0.74	2.52	5.61	10.83	0.49	3.26	10.03	23.22	45.74

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Apple M1 Pro (macOS, Rosetta 2) in single-threaded mode.

Table S.11: Multivariate trend model on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	44	69	95	102	113	38	59	97	105	115
8	46	59	71	81	84	35	64	69	73	75
12	36	42	54	64	59	34	47	51	57	65
24	24	33	33	40	46	29	29	34	42	45
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	90	146	176	177	188	83	112	171	184	172
8	87	104	120	128	128	71	108	113	114	113
12	66	72	86	95	84	61	77	79	85	85
24	41	49	46	54	58	39	44	48	54	56
PANEL C: Trend-Cycle PS as percentage of DK										
4	46	71	87	87	108	34	50	83	99	109
8	42	55	63	66	77	30	54	59	67	76
12	36	36	45	53	53	27	40	44	53	59
24	19	27	27	33	40	19	25	30	39	41
PANEL D: standard PS w/noise as percentage of DK										
4	66	91	144	142	154	65	87	133	147	153
8	60	85	104	107	108	57	91	95	96	105
12	45	60	78	85	76	52	66	69	76	82
24	34	46	43	50	55	34	38	44	53	54
PANEL E: DK in seconds										
4	0.01	0.02	0.04	0.07	0.11	0.04	0.11	0.17	0.29	0.51
8	0.02	0.06	0.11	0.18	0.31	0.07	0.20	0.47	0.87	1.46
12	0.03	0.12	0.22	0.39	0.76	0.12	0.42	1.01	1.84	3.09
24	0.09	0.39	0.99	1.88	3.21	0.38	1.65	3.95	7.58	13.81

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.12: Multivariate trend model on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	52	122	165	213	223	57	149	179	219	227
8	66	108	122	144	143	67	115	130	131	133
12	56	86	90	107	98	66	92	89	99	100
24	43	53	52	57	57	48	52	55	59	57
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	100	233	269	319	320	118	251	280	331	324
8	119	170	184	197	190	125	175	188	181	175
12	97	129	125	140	123	106	134	122	129	126
24	65	72	66	68	66	70	68	68	70	64
PANEL C: Trend-Cycle PS as percentage of DK										
4	50	119	148	192	209	49	136	163	208	213
8	62	95	105	125	133	58	102	117	120	126
12	53	77	80	94	90	54	82	81	91	93
24	36	46	45	49	50	40	46	50	54	53
PANEL D: standard PS w/noise as percentage of DK										
4	69	155	205	246	248	83	183	204	250	254
8	79	128	145	157	159	88	136	148	143	146
12	68	106	104	120	109	79	108	100	109	109
24	51	62	58	63	62	55	57	59	66	61
PANEL E: DK in seconds										
4	0.01	0.02	0.04	0.07	0.13	0.04	0.08	0.18	0.30	0.51
8	0.02	0.05	0.14	0.25	0.46	0.07	0.22	0.54	1.12	1.99
12	0.03	0.11	0.32	0.60	1.18	0.12	0.47	1.37	2.65	4.70
24	0.10	0.52	1.62	3.35	6.27	0.41	2.30	6.41	13.23	26.39

Notes: Based on simulated data. Panels A through D report execution times of precision-based samplers (PS) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK). Execution times (in seconds) of the DK sampler are reported in Panel E. Panels A and B report execution times for the generic ABC precision-based sampler, with Panel B reflecting the use of prepared one-off computations. Panel C provides results for a PS specicalized to the trend-cycle case (and prepared one-off computations). Panel D reflects calls to a standard precision-based sampler, when called with a minimal noise term added to the measurement equation. All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.13: VAR with 1% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	17	19	20	16	15	13	21	19	15	14
8	11	10	8	8	8	12	9	7	7	7
12	6	5	6	6	6	7	5	5	5	6
24	2	3	4	2	2	3	3	3	2	2
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	41	47	46	37	37	32	48	45	37	35
8	26	21	18	18	17	28	19	17	19	19
12	15	13	14	14	15	17	12	14	14	16
24	6	9	11	7	8	8	9	11	7	8
PANEL C: missing-variables DK as percentage of DK										
4	92	110	104	74	74	88	113	103	73	74
8	118	78	76	71	71	117	76	76	73	73
12	106	75	70	69	69	106	75	71	69	70
24	77	72	70	87	91	76	71	70	85	92
PANEL D: DK in seconds										
4	0.01	0.04	0.08	0.17	0.27	0.05	0.14	0.34	0.69	1.08
8	0.03	0.16	0.40	0.63	0.96	0.13	0.66	1.57	2.53	3.69
12	0.08	0.40	0.74	1.23	1.75	0.32	1.56	2.92	4.90	6.90
24	0.39	1.16	2.18	5.42	7.70	1.55	4.71	8.89	22.42	32.48

Notes: Based on simulated data with 1% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.14: VAR with 5% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	18	21	22	18	15	13	23	19	15	14
8	11	10	8	8	8	12	8	7	7	8
12	6	6	6	6	6	7	5	5	5	6
24	2	3	4	2	3	3	3	3	2	2
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	43	47	47	38	35	33	49	45	37	35
8	25	22	18	18	17	28	19	18	19	20
12	15	13	14	14	15	17	12	14	14	16
24	6	9	11	7	8	8	9	11	7	8
PANEL C: missing-variables DK as percentage of DK										
4	90	117	102	74	73	106	115	101	74	73
8	116	78	95	70	69	120	79	75	70	74
12	104	75	69	69	69	104	75	70	67	72
24	76	71	69	88	92	76	69	70	83	90
PANEL D: DK in seconds										
4	0.01	0.03	0.08	0.17	0.28	0.06	0.14	0.35	0.69	1.09
8	0.03	0.16	0.40	0.65	0.99	0.13	0.63	1.58	2.54	3.67
12	0.08	0.40	0.75	1.23	1.74	0.31	1.56	2.94	4.98	6.83
24	0.39	1.16	2.18	5.40	7.71	1.55	4.73	8.84	22.67	32.85

Notes: Based on simulated data with 5% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.15: VAR with 10% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	18	21	22	18	16	14	24	20	16	15
8	11	10	8	8	8	12	9	8	8	8
12	7	6	6	6	6	7	5	5	6	6
24	2	3	4	3	3	3	3	3	2	2
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	44	48	47	39	35	34	50	46	38	35
8	25	22	18	18	18	29	20	18	19	20
12	17	13	14	14	15	17	12	14	15	16
24	6	10	11	7	8	8	9	11	7	8
PANEL C: missing-variables DK as percentage of DK										
4	93	116	102	76	72	87	116	103	74	71
8	120	79	73	71	71	117	78	75	71	73
12	103	73	72	69	69	103	74	69	71	71
24	77	73	69	87	91	76	70	70	87	88
PANEL D: DK in seconds										
4	0.01	0.03	0.09	0.17	0.27	0.05	0.14	0.34	0.70	1.12
8	0.04	0.16	0.40	0.64	0.96	0.13	0.63	1.59	2.51	3.68
12	0.08	0.40	0.74	1.23	1.71	0.32	1.59	3.00	4.84	6.99
24	0.40	1.16	2.19	5.36	7.66	1.54	4.68	8.87	22.40	31.79

Notes: Based on simulated data with 10% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.16: VAR with 20% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	20	22	26	19	17	16	25	21	17	15
8	13	11	9	9	9	13	10	8	8	8
12	7	6	6	7	7	8	6	6	6	6
24	2	4	4	3	3	3	3	4	2	3
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	44	48	49	40	35	36	51	48	39	36
8	28	23	19	19	19	30	20	18	20	20
12	16	14	14	15	16	18	13	15	15	16
24	7	9	10	8	8	8	9	11	7	8
PANEL C: missing-variables DK as percentage of DK										
4	89	114	99	74	69	87	114	105	74	70
8	118	78	75	71	70	117	80	75	71	72
12	105	75	71	69	69	104	76	72	69	71
24	78	73	64	86	91	77	71	69	86	88
PANEL D: DK in seconds										
4	0.01	0.04	0.09	0.17	0.28	0.06	0.14	0.35	0.70	1.12
8	0.03	0.16	0.41	0.63	0.97	0.13	0.63	1.58	2.52	3.72
12	0.08	0.41	0.76	1.23	1.74	0.32	1.55	2.91	4.89	6.96
24	0.38	1.23	2.38	5.42	7.69	1.52	4.71	8.95	22.87	31.44

Notes: Based on simulated data with 20% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.17: VAR with 30% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	19	26	26	20	18	17	26	23	18	17
8	12	12	10	10	9	14	10	8	9	9
12	8	6	7	7	7	9	6	6	7	7
24	3	4	4	3	3	3	3	4	3	3
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	40	53	51	42	37	36	53	49	40	38
8	28	24	19	20	20	31	21	19	20	21
12	17	14	16	16	16	18	13	15	16	17
24	7	10	11	8	8	8	10	12	8	8
PANEL C: missing-variables DK as percentage of DK										
4	93	117	105	74	70	87	116	103	75	70
8	111	80	75	71	70	118	79	75	71	72
12	106	77	70	70	69	122	77	71	70	70
24	76	71	70	87	91	77	71	71	87	87
PANEL D: DK in seconds										
4	0.02	0.03	0.08	0.17	0.29	0.06	0.14	0.35	0.69	1.11
8	0.04	0.16	0.40	0.63	0.96	0.13	0.62	1.58	2.49	3.69
12	0.08	0.39	0.73	1.21	1.74	0.32	1.55	2.92	4.85	6.92
24	0.39	1.17	2.15	5.35	7.67	1.51	4.73	8.72	22.46	31.38

Notes: Based on simulated data with 30% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.18: VAR with 50% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	24	30	28	24	22	20	31	25	21	19
8	16	13	10	11	11	16	11	10	10	11
12	9	7	8	8	9	9	7	7	8	8
24	3	5	5	3	4	4	4	5	3	4
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	48	57	53	44	41	38	59	51	43	41
8	30	24	20	21	21	32	22	20	22	23
12	18	15	16	16	18	19	14	16	16	18
24	8	11	12	8	9	9	10	13	8	9
PANEL C: missing-variables DK as percentage of DK										
4	86	113	104	72	69	86	115	102	74	69
8	113	76	74	71	69	116	79	75	71	72
12	108	76	71	67	71	105	76	69	68	70
24	78	69	69	86	91	77	71	70	83	88
PANEL D: DK in seconds										
4	0.01	0.03	0.08	0.17	0.28	0.06	0.14	0.35	0.69	1.12
8	0.03	0.16	0.40	0.64	0.97	0.13	0.62	1.57	2.50	3.68
12	0.08	0.39	0.74	1.23	1.72	0.32	1.54	2.95	4.93	6.89
24	0.38	1.18	2.15	5.40	7.60	1.51	4.66	8.70	22.57	31.12

Notes: Based on simulated data with 50% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.19: VAR with 70% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	26	34	31	27	24	22	35	30	24	24
8	16	15	12	13	12	17	13	11	12	13
12	10	8	9	9	10	11	8	8	9	10
24	4	5	6	4	4	4	5	5	4	4
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	51	60	58	48	43	42	62	56	46	44
8	32	27	22	23	22	32	24	22	24	24
12	19	16	17	18	18	21	15	18	18	19
24	8	11	13	9	10	9	10	12	9	10
PANEL C: missing-variables DK as percentage of DK										
4	89	115	101	71	69	86	112	102	74	69
8	114	79	74	71	69	113	81	74	71	72
12	105	78	69	69	68	103	74	71	67	71
24	76	73	70	89	92	76	67	62	85	91
PANEL D: DK in seconds										
4	0.01	0.03	0.08	0.17	0.28	0.05	0.14	0.34	0.68	1.11
8	0.03	0.15	0.40	0.62	0.96	0.14	0.62	1.58	2.48	3.68
12	0.08	0.39	0.75	1.22	1.74	0.32	1.57	2.93	4.94	7.01
24	0.38	1.16	2.20	5.33	7.54	1.54	5.07	9.87	22.90	31.06

Notes: Based on simulated data with 70% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.20: VAR with 90% of observations missing on Intel Xeon (Windows, 32 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	27	37	43	33	28	26	37	34	28	26
8	18	16	14	14	14	19	15	12	13	14
12	11	10	8	10	12	13	8	10	9	10
24	4	5	5	4	5	5	5	6	4	4
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	51	65	71	56	46	46	64	61	50	46
8	33	28	25	24	25	35	26	23	24	26
12	21	18	15	18	20	23	15	19	17	19
24	9	10	11	9	10	10	11	13	9	9
PANEL C: missing-variables DK as percentage of DK										
4	84	116	94	74	69	86	110	98	75	70
8	104	79	75	73	69	116	78	74	73	79
12	103	78	70	71	68	105	78	70	75	71
24	79	77	71	87	93	79	71	65	83	244
PANEL D: DK in seconds										
4	0.02	0.04	0.09	0.18	0.29	0.06	0.15	0.36	0.71	1.14
8	0.04	0.16	0.41	0.66	1.00	0.14	0.65	1.63	2.73	3.82
12	0.08	0.40	0.95	1.35	1.87	0.32	1.62	3.06	5.60	7.85
24	0.40	1.53	2.70	5.79	7.99	1.56	5.22	10.15	24.35	35.58

Notes: Based on simulated data with 90% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in multi-threaded mode (for matrix operations) using 32 threads.

Table S.21: VAR with 1% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	15	29	30	28	27	14	37	30	28	30
8	15	13	13	14	13	16	13	14	15	14
12	8	9	9	9	9	9	8	9	9	9
24	3	4	4	3	3	4	4	4	3	3
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	39	68	66	59	54	36	77	65	60	61
8	46	29	29	29	26	41	29	30	30	29
12	22	20	20	19	20	23	20	21	21	21
24	10	12	11	8	7	12	12	12	8	7
PANEL C: missing-variables DK as percentage of DK										
4	91	84	83	83	83	89	81	84	83	83
8	86	85	93	97	97	87	88	92	97	98
12	89	94	107	107	115	86	95	107	105	117
24	96	114	127	125	139	98	117	128	119	133
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.13	0.23	0.06	0.11	0.28	0.54	0.89
8	0.02	0.13	0.29	0.51	0.91	0.10	0.51	1.17	2.09	3.54
12	0.06	0.29	0.64	1.20	1.86	0.26	1.17	2.54	4.73	7.38
24	0.28	1.10	2.48	6.33	11.36	1.12	4.35	10.12	27.22	48.86

Notes: Based on simulated data with 1% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.22: VAR with 5% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	18	31	31	28	28	14	34	30	29	30
8	16	14	14	14	13	17	13	14	14	14
12	8	9	10	9	9	9	9	10	9	10
24	3	4	4	3	3	4	4	5	3	3
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	43	70	68	58	58	35	78	66	61	61
8	39	30	30	29	26	41	29	30	31	29
12	22	20	21	20	20	23	20	22	21	21
24	10	12	12	8	7	12	12	12	9	8
PANEL C: missing-variables DK as percentage of DK										
4	90	83	83	82	83	88	83	82	85	86
8	84	85	92	97	97	86	86	90	97	98
12	86	94	106	108	115	87	97	110	107	115
24	96	113	127	126	137	97	113	127	119	134
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.13	0.22	0.05	0.10	0.29	0.54	0.87
8	0.03	0.13	0.29	0.51	0.88	0.10	0.51	1.17	2.09	3.50
12	0.06	0.29	0.63	1.17	1.83	0.25	1.13	2.46	4.71	7.28
24	0.28	1.10	2.46	6.15	11.29	1.12	4.38	9.93	26.68	48.54

Notes: Based on simulated data with 5% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.23: VAR with 10% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	18	32	32	30	28	15	36	32	30	31
8	16	15	14	14	14	18	14	14	15	15
12	9	9	10	10	10	10	9	10	10	10
24	4	5	5	3	3	4	5	5	3	3
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	43	72	68	61	54	36	80	67	62	60
8	39	31	30	29	27	41	30	31	31	30
12	22	21	21	20	20	24	21	22	22	22
24	11	12	12	8	7	12	12	13	9	8
PANEL C: missing-variables DK as percentage of DK										
4	92	83	85	83	79	88	85	85	83	84
8	84	86	92	96	99	85	86	91	95	98
12	86	94	103	107	115	87	97	106	109	115
24	95	112	127	127	135	96	111	128	120	134
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.13	0.23	0.05	0.10	0.28	0.55	0.90
8	0.03	0.12	0.29	0.52	0.88	0.10	0.51	1.17	2.07	3.53
12	0.06	0.29	0.64	1.18	1.85	0.25	1.14	2.50	4.69	7.36
24	0.28	1.09	2.48	6.18	11.61	1.13	4.41	9.91	26.81	48.36

Notes: Based on simulated data with 10% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.24: VAR with 20% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	19	33	34	32	33	16	38	34	33	34
8	18	15	16	17	16	19	14	16	17	17
12	10	10	11	11	11	11	10	11	11	12
24	4	5	6	4	4	5	5	6	4	4
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	44	72	70	62	62	38	82	70	65	67
8	42	31	32	36	30	42	31	32	34	32
12	23	21	22	21	20	25	22	24	23	24
24	11	12	13	9	8	13	13	13	9	9
PANEL C: missing-variables DK as percentage of DK										
4	88	86	84	82	84	88	84	83	82	84
8	83	79	90	97	102	84	85	92	98	98
12	87	93	106	104	107	87	96	105	105	115
24	98	113	129	128	133	98	114	126	119	135
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.13	0.22	0.05	0.11	0.28	0.55	0.89
8	0.02	0.13	0.29	0.52	0.86	0.10	0.51	1.16	2.06	3.52
12	0.06	0.29	0.63	1.21	1.98	0.25	1.14	2.49	4.76	7.33
24	0.28	1.09	2.47	6.27	11.78	1.12	4.38	10.06	27.19	48.17

Notes: Based on simulated data with 20% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.25: VAR with 30% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	20	38	38	35	36	17	42	38	37	38
8	19	17	17	19	18	18	16	18	19	19
12	11	12	12	13	14	12	11	13	13	14
24	5	6	7	5	5	5	6	7	5	5
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	45	80	73	66	64	39	84	74	69	69
8	42	33	33	34	31	39	31	34	36	35
12	24	23	24	23	24	26	23	25	25	26
24	11	13	14	10	9	13	14	15	10	10
PANEL C: missing-variables DK as percentage of DK										
4	87	82	84	84	83	88	83	83	86	89
8	83	85	92	98	99	76	88	93	96	99
12	85	96	102	107	115	84	96	104	108	115
24	97	114	128	126	137	98	113	128	118	133
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.14	0.22	0.06	0.11	0.28	0.54	0.88
8	0.03	0.13	0.29	0.52	0.89	0.11	0.54	1.16	2.10	3.49
12	0.06	0.28	0.65	1.18	1.85	0.26	1.15	2.51	4.74	7.37
24	0.28	1.09	2.53	6.35	11.50	1.13	4.41	9.92	27.45	48.51

Notes: Based on simulated data with 30% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.26: VAR with 50% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	24	46	42	44	45	21	51	46	45	46
8	24	21	22	24	25	24	21	23	25	26
12	13	14	16	18	19	14	14	17	18	19
24	6	8	9	8	7	7	9	10	8	8
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	48	88	76	74	72	43	94	82	77	78
8	46	37	38	39	38	48	37	39	41	41
12	27	25	27	31	29	28	26	29	29	31
24	13	16	16	12	11	15	16	18	13	12
PANEL C: missing-variables DK as percentage of DK										
4	91	83	84	84	84	87	84	83	83	83
8	82	86	89	97	99	84	86	91	96	98
12	88	98	104	105	114	86	95	106	107	115
24	97	114	126	128	135	98	113	124	120	136
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.13	0.22	0.05	0.10	0.28	0.54	0.89
8	0.03	0.12	0.29	0.51	0.88	0.10	0.51	1.16	2.07	3.46
12	0.06	0.29	0.63	1.21	1.84	0.25	1.16	2.51	4.67	7.31
24	0.28	1.08	2.50	6.24	11.67	1.11	4.42	10.20	27.05	48.39

Notes: Based on simulated data with 50% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.27: VAR with 70% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	28	54	53	53	54	24	59	55	53	57
8	25	24	27	30	30	27	24	28	31	31
12	16	17	21	22	24	17	18	21	22	25
24	7	11	13	10	10	8	11	13	10	10
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	53	96	89	84	80	48	104	92	85	86
8	47	40	43	45	44	50	41	44	47	47
12	29	28	32	33	35	32	30	33	34	36
24	14	18	20	15	14	16	18	21	16	15
PANEL C: missing-variables DK as percentage of DK										
4	88	84	83	81	82	85	84	83	82	83
8	80	85	90	95	98	85	86	96	97	99
12	86	96	110	109	116	85	93	106	107	115
24	96	115	129	129	137	93	114	128	117	134
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.13	0.23	0.05	0.10	0.28	0.54	0.87
8	0.03	0.13	0.29	0.52	0.88	0.10	0.50	1.16	2.05	3.49
12	0.06	0.28	0.63	1.15	1.81	0.25	1.13	2.52	4.65	7.32
24	0.28	1.09	2.43	6.12	11.46	1.15	4.37	9.90	26.52	47.91

Notes: Based on simulated data with 70% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.28: VAR with 90% of observations missing on Intel Xeon (Windows, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	30	60	63	62	64	25	66	62	70	66
8	31	28	32	35	35	30	28	32	28	39
12	18	20	24	27	30	20	21	25	48	28
24	8	12	15	13	13	10	13	18	16	12
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	54	107	98	93	92	48	110	98	93	101
8	52	44	48	50	48	54	44	48	41	56
12	31	31	35	37	41	34	33	38	49	38
24	15	20	22	18	17	18	21	24	18	17
PANEL C: missing-variables DK as percentage of DK										
4	86	83	82	82	83	85	83	83	96	80
8	79	82	90	97	98	83	81	94	71	108
12	85	92	106	107	116	86	94	108	103	113
24	95	116	125	127	138	95	115	134	123	128
PANEL D: DK in seconds										
4	0.01	0.03	0.07	0.13	0.22	0.05	0.10	0.28	0.65	1.04
8	0.03	0.13	0.29	0.52	0.88	0.10	0.52	1.17	3.34	3.78
12	0.06	0.29	0.64	1.15	1.78	0.25	1.10	2.45	6.03	8.43
24	0.28	1.07	2.47	6.13	11.17	1.10	4.31	9.75	37.86	53.49

Notes: Based on simulated data with 90% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2021b) with the `timeit` function on an Intel(R) Xeon(R) Gold 6320 CPU @ 2.1 GHz (Windows) in single-threaded mode.

Table S.29: VAR with 1% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	13	21	26	30	35	10	21	30	36	37
8	11	15	15	17	19	11	16	17	17	16
12	10	10	13	12	11	9	12	11	11	11
24	5	5	5	4	5	5	5	5	5	5
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	33	51	57	67	81	24	52	70	81	81
8	28	34	33	39	43	25	38	40	39	36
12	28	26	29	30	25	24	27	27	30	28
24	21	23	19	18	18	22	22	21	20	18
PANEL C: missing-variables DK as percentage of DK										
4	92	88	83	80	88	90	93	82	80	86
8	103	82	86	92	89	92	81	86	88	96
12	92	89	92	97	99	91	88	105	91	86
24	154	99	107	111	117	87	96	105	108	107
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.08	0.13	0.21	0.34
8	0.02	0.05	0.10	0.17	0.28	0.07	0.20	0.43	0.84	1.47
12	0.03	0.10	0.21	0.38	0.72	0.11	0.41	1.01	1.82	3.26
24	0.10	0.37	1.03	1.94	3.27	0.40	1.75	4.19	7.99	14.63

Notes: Based on simulated data with 1% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.30: VAR with 5% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	13	28	27	31	36	10	23	31	39	39
8	12	15	15	18	20	10	16	18	18	19
12	11	10	12	12	12	9	12	12	12	12
24	5	5	5	5	4	5	6	6	5	5
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	31	54	59	69	82	26	54	69	81	79
8	29	35	35	41	43	24	38	39	38	40
12	29	26	29	30	25	25	28	27	30	28
24	22	23	19	19	16	23	22	21	20	17
PANEL C: missing-variables DK as percentage of DK										
4	90	94	83	82	90	92	95	81	80	87
8	109	84	89	91	88	97	81	88	84	109
12	92	90	91	95	98	92	92	86	91	97
24	88	99	105	111	103	87	95	104	107	106
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.08	0.14	0.21	0.35
8	0.02	0.05	0.10	0.17	0.28	0.07	0.20	0.44	0.86	1.33
12	0.03	0.10	0.21	0.38	0.72	0.11	0.41	1.03	1.82	3.36
24	0.10	0.37	1.03	1.91	3.72	0.40	1.74	4.13	7.95	15.04

Notes: Based on simulated data with 5% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.31: VAR with 10% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	14	23	27	34	38	10	24	33	40	39
8	12	15	16	19	21	10	17	19	18	19
12	11	11	13	13	12	11	12	11	12	13
24	5	6	5	5	5	5	6	6	6	6
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	32	55	58	71	80	25	54	70	83	80
8	29	35	36	41	44	25	39	41	39	41
12	31	25	29	30	30	26	28	27	31	32
24	22	23	20	19	18	22	22	21	20	19
PANEL C: missing-variables DK as percentage of DK										
4	91	94	83	82	89	91	96	81	81	87
8	96	83	91	95	87	96	81	89	87	86
12	124	90	92	94	98	92	89	87	93	98
24	88	94	103	108	116	88	94	104	107	114
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.08	0.14	0.21	0.35
8	0.02	0.05	0.11	0.18	0.28	0.07	0.20	0.43	0.85	1.32
12	0.03	0.10	0.21	0.38	0.71	0.11	0.43	1.03	1.80	2.86
24	0.10	0.38	1.03	1.94	3.28	0.41	1.74	4.13	7.91	13.64

Notes: Based on simulated data with 10% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.32: VAR with 20% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	15	28	32	38	41	13	27	35	42	41
8	14	19	18	21	23	12	19	20	18	18
12	14	13	15	14	13	11	13	11	13	14
24	7	7	6	6	6	6	6	7	6	6
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	34	59	66	77	83	27	56	73	86	81
8	31	38	36	43	46	27	40	41	37	37
12	28	26	30	32	32	25	28	25	30	30
24	23	24	20	19	19	23	23	22	18	19
PANEL C: missing-variables DK as percentage of DK										
4	90	95	83	80	89	90	96	81	80	86
8	94	82	88	94	89	95	79	88	95	96
12	94	92	89	95	97	93	88	91	95	86
24	89	95	104	110	116	88	95	98	91	102
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.08	0.14	0.21	0.36
8	0.02	0.05	0.11	0.18	0.28	0.07	0.20	0.44	0.93	1.51
12	0.03	0.10	0.21	0.38	0.72	0.11	0.43	1.17	2.05	3.30
24	0.10	0.38	1.03	1.96	3.28	0.41	1.73	4.40	9.41	15.17

Notes: Based on simulated data with 20% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.33: VAR with 30% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	16	30	35	42	42	14	30	37	48	47
8	16	21	21	23	24	13	20	22	22	24
12	15	14	15	16	15	12	14	14	16	17
24	7	7	6	6	7	7	7	7	8	8
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	36	61	68	82	86	30	60	76	91	88
8	32	40	39	44	49	27	42	43	43	45
12	30	27	31	31	30	26	31	30	34	35
24	22	23	20	19	20	23	24	23	22	21
PANEL C: missing-variables DK as percentage of DK										
4	88	93	83	81	86	90	97	82	80	87
8	94	83	91	92	85	95	81	87	86	86
12	93	89	89	94	100	92	89	86	92	99
24	84	105	104	108	115	88	95	104	109	113
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.09	0.04	0.08	0.13	0.21	0.35
8	0.02	0.05	0.11	0.18	0.29	0.07	0.20	0.44	0.85	1.31
12	0.03	0.10	0.21	0.38	0.71	0.11	0.42	1.03	1.79	2.85
24	0.11	0.39	1.03	1.97	3.31	0.41	1.74	4.13	8.02	13.51

Notes: Based on simulated data with 30% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.34: VAR with 50% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	19	34	41	49	50	15	31	45	55	54
8	18	25	22	26	30	15	24	27	27	29
12	17	16	18	20	19	14	16	17	19	21
24	9	9	8	8	9	8	9	11	10	10
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	40	67	75	89	92	31	62	83	99	96
8	34	43	40	48	54	29	46	49	47	50
12	32	29	32	35	33	28	34	32	34	41
24	24	22	22	21	22	21	25	24	25	23
PANEL C: missing-variables DK as percentage of DK										
4	90	93	81	80	83	90	93	81	79	84
8	93	84	86	93	88	95	80	87	84	86
12	93	88	90	94	97	92	88	84	92	98
24	88	93	103	109	116	88	94	105	109	112
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.09	0.04	0.08	0.13	0.21	0.35
8	0.02	0.05	0.11	0.18	0.28	0.07	0.20	0.44	0.86	1.31
12	0.03	0.10	0.21	0.38	0.71	0.11	0.42	1.05	1.82	2.85
24	0.10	0.38	1.03	1.96	3.30	0.41	1.74	4.19	7.94	14.00

Notes: Based on simulated data with 50% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.35: VAR with 70% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	20	39	48	53	55	17	33	48	62	61
8	19	27	27	31	35	17	26	30	29	33
12	23	19	22	23	22	17	19	20	22	23
24	10	10	10	11	12	9	10	11	12	13
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	42	70	82	93	104	33	66	87	108	103
8	36	47	45	53	61	32	49	52	51	55
12	38	32	36	38	37	30	35	36	39	44
24	29	27	24	24	25	25	27	27	27	27
PANEL C: missing-variables DK as percentage of DK										
4	87	92	82	79	82	89	92	77	78	82
8	128	82	87	93	85	92	79	87	87	87
12	92	87	87	93	97	91	88	86	92	96
24	90	92	104	109	117	88	93	106	108	112
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.08	0.14	0.21	0.35
8	0.02	0.05	0.11	0.17	0.28	0.07	0.20	0.44	0.85	1.31
12	0.03	0.10	0.21	0.39	0.71	0.11	0.43	1.03	1.80	2.89
24	0.10	0.38	1.03	1.97	3.27	0.41	1.76	4.08	7.95	13.65

Notes: Based on simulated data with 70% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.36: VAR with 90% of observations missing on Intel i7 (macOS, 4 threads)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	23	44	51	59	61	19	41	55	69	69
8	23	31	30	37	41	19	30	34	35	38
12	21	20	23	26	23	19	22	23	26	28
24	11	12	12	13	15	10	12	14	15	16
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	44	74	86	102	105	35	72	93	114	111
8	39	51	49	56	64	34	51	54	55	59
12	37	34	40	44	38	33	39	38	41	45
24	27	28	26	26	28	26	28	29	30	29
PANEL C: missing-variables DK as percentage of DK										
4	88	91	81	79	81	88	94	80	78	81
8	97	82	86	90	86	92	78	86	84	86
12	91	87	88	93	86	90	87	85	92	97
24	89	94	103	108	116	88	93	104	106	114
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.08	0.13	0.21	0.34
8	0.02	0.05	0.11	0.17	0.28	0.07	0.20	0.44	0.85	1.31
12	0.03	0.10	0.21	0.38	0.80	0.11	0.42	1.03	1.81	2.86
24	0.10	0.38	1.03	1.94	3.27	0.41	1.75	4.14	7.97	14.07

Notes: Based on simulated data with 90% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in multi-threaded mode (for matrix operations) using 4 threads.

Table S.37: VAR with 1% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	11	30	42	49	63	11	33	52	59	66
8	14	20	23	24	28	13	24	28	26	26
12	14	15	15	15	15	12	17	15	15	15
24	6	6	5	5	5	6	7	6	6	5
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	30	72	90	98	119	30	82	111	119	126
8	36	46	49	50	53	34	54	58	51	51
12	37	35	34	34	28	34	39	34	34	32
24	26	21	17	24	13	27	21	19	16	15
PANEL C: missing-variables DK as percentage of DK										
4	87	92	99	101	102	91	93	96	98	100
8	92	107	115	121	126	90	103	110	113	119
12	104	120	128	137	138	102	116	121	130	137
24	120	144	152	160	166	119	136	151	160	164
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.07	0.13	0.22	0.35
8	0.02	0.05	0.11	0.21	0.36	0.07	0.20	0.45	0.99	1.65
12	0.03	0.11	0.26	0.53	1.03	0.11	0.43	1.22	2.42	4.15
24	0.10	0.50	1.53	3.22	5.91	0.41	2.29	6.14	13.03	24.19

Notes: Based on simulated data with 1% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.38: VAR with 5% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	13	31	43	49	64	12	38	54	62	69
8	15	21	23	24	28	14	25	30	28	27
12	14	15	15	16	15	12	18	16	16	16
24	6	6	5	5	5	7	7	6	6	5
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	32	73	91	100	120	30	84	112	120	128
8	37	47	50	50	52	35	55	60	52	51
12	38	36	34	33	28	34	40	34	34	33
24	26	21	17	15	14	27	22	19	17	15
PANEL C: missing-variables DK as percentage of DK										
4	93	91	95	100	101	90	93	97	99	102
8	88	109	111	120	125	93	103	113	114	120
12	102	120	129	137	131	101	116	121	130	138
24	124	149	152	158	166	119	136	151	159	164
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.09	0.04	0.07	0.13	0.22	0.35
8	0.02	0.05	0.11	0.22	0.38	0.07	0.20	0.45	0.98	1.66
12	0.03	0.11	0.27	0.56	1.08	0.11	0.43	1.22	2.42	4.15
24	0.10	0.50	1.62	3.36	5.90	0.41	2.29	6.15	13.08	24.10

Notes: Based on simulated data with 5% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.39: VAR with 10% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	14	33	45	55	68	12	40	56	65	71
8	17	22	26	28	29	14	26	30	27	28
12	15	17	17	17	16	14	19	17	17	17
24	7	7	6	6	6	8	7	7	6	6
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	33	75	93	104	124	32	87	114	124	129
8	38	47	53	54	53	34	55	60	53	52
12	37	36	36	34	32	35	40	36	35	34
24	26	22	17	15	14	28	22	19	17	15
PANEL C: missing-variables DK as percentage of DK										
4	90	91	98	100	102	96	92	96	100	102
8	92	107	111	121	124	88	103	111	113	122
12	101	119	129	137	137	101	115	121	130	137
24	125	145	151	159	166	119	134	151	159	163
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.07	0.13	0.22	0.35
8	0.02	0.05	0.11	0.21	0.37	0.07	0.21	0.46	0.99	1.66
12	0.03	0.11	0.27	0.55	1.04	0.11	0.44	1.22	2.42	4.16
24	0.10	0.50	1.59	3.30	6.01	0.42	2.33	6.15	13.07	24.16

Notes: Based on simulated data with 10% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.40: VAR with 20% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	15	38	55	62	75	15	44	61	69	77
8	18	26	30	31	35	18	30	35	31	32
12	19	20	20	20	19	15	21	19	20	20
24	8	8	7	7	7	9	9	8	8	7
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	35	80	108	116	130	34	89	118	127	136
8	39	52	56	58	59	40	60	65	57	57
12	38	39	40	37	36	35	43	38	38	37
24	27	23	19	16	15	29	23	21	19	17
PANEL C: missing-variables DK as percentage of DK										
4	89	93	100	103	104	89	93	96	99	102
8	93	107	115	121	127	93	103	113	114	120
12	102	119	131	138	137	100	115	120	130	137
24	120	145	153	160	167	120	136	152	160	164
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.07	0.13	0.22	0.35
8	0.02	0.05	0.11	0.21	0.36	0.07	0.20	0.44	0.98	1.66
12	0.03	0.10	0.26	0.53	1.04	0.11	0.43	1.22	2.41	4.15
24	0.10	0.50	1.54	3.23	5.88	0.41	2.30	6.16	13.02	24.09

Notes: Based on simulated data with 20% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.41: VAR with 30% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	17	42	60	69	82	17	50	67	79	89
8	20	30	35	36	39	19	33	39	37	37
12	20	22	22	23	22	17	24	22	24	23
24	10	10	8	8	8	10	10	10	10	10
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	36	85	111	123	139	36	94	127	136	149
8	41	56	61	61	63	42	63	72	62	62
12	42	40	41	39	37	37	47	41	41	40
24	29	25	20	18	17	30	25	23	20	19
PANEL C: missing-variables DK as percentage of DK										
4	87	92	97	101	102	91	92	96	99	101
8	91	108	113	122	126	92	103	112	114	120
12	101	118	127	137	138	102	116	120	130	137
24	122	150	150	158	166	119	135	151	159	166
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.09	0.04	0.07	0.13	0.22	0.35
8	0.02	0.05	0.11	0.21	0.37	0.07	0.20	0.45	0.98	1.65
12	0.03	0.10	0.27	0.55	1.03	0.11	0.43	1.22	2.41	4.16
24	0.10	0.50	1.60	3.38	5.89	0.41	2.30	6.18	13.09	23.77

Notes: Based on simulated data with 30% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.42: VAR with 50% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	19	52	75	87	100	19	56	84	97	106
8	27	38	43	46	49	23	42	50	47	49
12	25	28	30	32	30	22	31	30	30	31
24	13	13	12	12	13	13	14	15	14	14
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	41	94	129	142	157	39	103	145	159	167
8	49	64	70	73	75	44	73	80	72	74
12	44	46	49	48	44	42	55	48	46	49
24	28	26	24	22	21	31	29	27	25	23
PANEL C: missing-variables DK as percentage of DK										
4	87	90	97	102	101	90	90	95	98	100
8	92	108	116	121	123	91	103	111	113	120
12	103	117	128	137	138	101	115	120	129	136
24	122	145	151	161	167	118	140	150	157	165
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.09	0.04	0.07	0.13	0.23	0.35
8	0.02	0.05	0.11	0.21	0.37	0.07	0.20	0.46	1.00	1.65
12	0.03	0.10	0.26	0.53	1.03	0.11	0.43	1.25	2.48	4.18
24	0.10	0.50	1.53	3.21	5.88	0.41	2.29	6.25	13.32	24.01

Notes: Based on simulated data with 50% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.43: VAR with 70% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	22	60	85	98	120	20	64	97	113	126
8	28	44	54	58	63	27	48	60	56	59
12	30	35	38	39	37	27	37	37	38	38
24	15	17	17	17	17	17	17	19	19	18
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	43	104	140	152	184	42	114	159	175	191
8	53	71	83	84	91	49	77	92	83	85
12	49	53	55	56	51	47	61	56	55	57
24	34	32	29	27	26	36	33	32	30	28
PANEL C: missing-variables DK as percentage of DK										
4	88	90	96	100	102	88	88	94	98	99
8	91	108	116	121	126	89	99	112	113	119
12	102	118	128	138	138	102	114	120	129	138
24	121	145	152	161	168	117	134	150	157	158
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.07	0.13	0.22	0.34
8	0.02	0.05	0.11	0.21	0.36	0.07	0.21	0.45	1.00	1.65
12	0.03	0.10	0.26	0.53	1.03	0.11	0.44	1.24	2.45	4.28
24	0.10	0.50	1.52	3.21	5.90	0.42	2.34	6.21	13.21	24.93

Notes: Based on simulated data with 70% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

Table S.44: VAR with 90% of observations missing on Intel i7 (macOS, 1 thread)

	N_y									
	$T = 200$					$T = 800$				
p	5	10	15	20	25	5	10	15	20	25
PANEL A: ABC-PS (excl. QR) as percentage of DK										
4	26	71	99	115	140	24	77	114	129	146
8	35	51	64	68	76	31	58	71	66	68
12	32	39	44	48	45	32	45	43	44	48
24	19	21	21	21	22	21	22	23	22	24
PANEL B: ABC-PS (incl. QR) as percentage of DK										
4	47	113	152	171	201	44	125	172	191	210
8	57	79	92	93	101	54	87	101	93	94
12	53	58	65	65	60	52	68	62	61	64
24	38	36	33	30	31	41	37	36	33	33
PANEL C: missing-variables DK as percentage of DK										
4	88	89	96	101	101	88	90	94	97	100
8	91	107	116	121	127	89	103	112	114	119
12	99	118	129	138	138	102	115	121	129	136
24	123	144	151	161	167	119	135	150	157	164
PANEL D: DK in seconds										
4	0.01	0.02	0.03	0.05	0.08	0.04	0.07	0.12	0.22	0.34
8	0.02	0.05	0.11	0.21	0.36	0.06	0.20	0.44	0.98	1.68
12	0.03	0.10	0.26	0.53	1.03	0.11	0.43	1.22	2.49	4.19
24	0.10	0.50	1.53	3.23	5.88	0.41	2.29	6.19	13.51	24.05

Notes: Based on simulated data with 90% of all observations missing. Panels A and B report the execution time of a typical call to the precision-based sampler (PS) for different choices of lag length (p), number of VAR variables (N_y) and observations (T) as percentage of the execution time of the Durbin-Koopmann's disturbance smoothing sampler (DK) whose execution time (in seconds) is reported in Panel D. Execution times for the precision-based sampler reported in Panel A reflect the use of prepared one-off computations (incl. the QR decomposition of measurement loadings $\boldsymbol{C}$) outside the measured times. These time are relevant for MCMC applications where $\boldsymbol{C}$ does not change between sampling steps. Panel B considers calls to the precision-based sampler that encompass all computations. Panel C considers a version of the DK sampler that is specialized to the missing-value case (where the sampler processes only the relevant rows of the state equation when updating the Kalman filter). All times were measured in MATLAB (R2022a) with the `timeit` function on an Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz (macOS) in single-threaded mode.

References

Carriero, Andrea, Todd E. Clark, Massimiliano Marcellino, and Elmar Mertens (2022), "Addressing COVID-19 outliers in BVARs with stochastic volatility," *Review of Economics and Statistics*, forthcoming.

Carter, C. K., and R. Kohn (1994), "On Gibbs sampling for state space models," *Biometrika*, 81, 541–553.

Chan, Joshua C. C., Aubrey Poon, and Dan Zhu (2023), "High-dimensional conditionally Gaussian state space models with missing data," Papers 2302.03172, arXiv.org.

Cogley, Timothy, and Thomas J. Sargent (2015), "Measuring price-level uncertainty and instability in the United States, 1850–2012," *The Review of Economics and Statistics*, 97, 827–838.

Del Negro, Marco, Domenico Giannone, Marc P. Giannoni, and Andrea Tambalotti (2017), "Safety, liquidity, and the natural rate of interest," *Brookings Papers on Economic Activity*, 48, 235–316.

Del Negro, Marco, Domenico Giannone, Marc P. Giannoni, and Andrea Tambalotti (2019), "Global trends in interest rates," *Journal of International Economics*, 118, 248–262.

Del Negro, Marco, and Giorgio E. Primiceri (2015), "Time varying structural vector autoregressions and monetary policy: A corrigendum," *Review of Economic Studies*, 82, 1342–1345.

Durbin, J., and S.J. Koopman (2002), "A simple and efficient simulation smoother for state space time series analysis," *Biometrika*, 89, 603–615.

Grant, Angelia L., and Joshua C. C. Chan (2017a), "A Bayesian model comparison for trend-cycle decompositions of output," *Journal of Money, Credit and Banking*, 49, 525–552.

——— (2017b), "Reconciling output gaps: Unobserved components model and Hodrick-Prescott filter," *Journal of Economic Dynamics and Control*, 75, 114–121.

Kim, Sangjoon, Neil Shephard, and Siddhartha Chib (1998), "Stochastic volatility: Likelihood inference and comparison with ARCH models," *The Review of Economic Studies*, 65, 361–393.

Mertens, Elmar (2016), "Measuring the level and uncertainty of trend inflation," *The Review of Economics and Statistics*, 98, 950–967.

Omori, Yasuhiro, Siddhartha Chib, Neil Shephard, and Jouchi Nakajima (2007), "Stochastic volatility with leverage: Fast and efficient likelihood inference," *Journal of Econometrics*, 140, 425–449.

Schorfheide, Frank, and Dongho Song (2015), "Real-time forecasting with a mixed-frequency VAR," *Journal of Business & Economic Statistics*, 33, 366–380.