Evaluation of Kalman filter for meteorological time series imputation

Simone Massaro

First Supervisor: Prof. Dr. Fabian Sinz
Second Supervisor: Dr. Franziska Koebsch

Master Thesis
Forest and Ecosystem Sciences
Ecosystem Analysis and Modelling

Georg-August Universität Göttigen
February 2023

Contents

1 Introduction

1.1 Eddy Covariance and gaps in meteorological variables

Eddy Covariance (EC) is a state of the art technique for measuring green house gases and energy exchanges between ecosystems and the atmosphere [2]. The technique allows for non-destructive measurements at the ecosystem level with a high temporal resolution. Eddy Covariance data is used for ecological and physiological research of ecosystems, as well as for validation of ecosystem process models and remote sensing observations [21]. The core of Eddy Covariance site is the 3D anemometer and gas analysers, which allows estimating the fluxes of interests (e.g. CO_2, H_2O, CH_4). Beside the fluxes, an Eddy Covariance setups collects measurements of meteorological variables and ecosystem parameters. This additional data provides the context to use and interpret the fluxes measurements.

The acquisition of the meteorological variables can be interrupted by failures in the instruments or power outages, resulting in gaps in the time series [2]. The presence of gaps is a problem for several uses of the EC data.
An important application of EC is the validation of Land Surface Models [3, 15, 5, 19], which are process based model that estimate fluxes using meteorological conditions as input. The errors of Land Surface Models deriving from inaccuracies in the input are comparable to the errors arising from the limitation in the models formulations [33]. This highlights the need of high quality continuous meteorological measurement that reflect that condition at the flux station. Meteorological observations are used as a driver to impute gaps in the fluxes measurements [2], which requires complete meteorological time series. Finally, the presence of gaps affects the calculation of long term averages for meteorological variables.

The described use cases make it necessary to impute the gaps in the meteorological variables. The first approach to reduce the number of gaps in to have redundant instruments on the site, however is this is not always possible and statistical models are used for imputing the gaps [2]. There are three different approaches that can be used to reconstruct missing data: 1) use the *temporal autocorrelation* of the variables, the measurements before and after the gap provide information on the missing data; 2) use *correlation* between different variables, if not all variables are missing the correlation between variables can be used for imputing the missing variable; 3) use *other measurements*, meteorological variables not only measured in EC tower and the data from nearby meteorological stations can also be used for imputation.

1.2 Current Imputation methods

EC post-processing pipeline impute meteorological time series. Arguably the most widely used post-processing pipeline is ONEFlux [22], which is adopted by FLUXENT, the global EC network, ICOS the European network as well as AmeriFlux, the American EC network. ONEFlux uses two different methods for imputing the meteorological data: Marginal Distribution Sampling (MDS) and ERA-Interim (ERA-I). The final gap-filled meteorological product uses either MDS or ERA-I, depending on the quality flag of MDS.

MDS Marginal Distribution Sampling [25] estimates the value of the missing variable by using the observations of the variable from other data points with similar meteorological conditions. The algorithm finds all the similar conditions by taking the observations from a time window around the gap other meteorological variables have similar values in the gap. All the observations of the variables of interest from similar conditions are then averaged to generate the filling value. In case there are not similar meteorological conditions in the starting time window, the size of the time window is progressively increased. If other meteorological variables are also missing, they are not used to find similar conditions.
The algorithm implemented in ONEFlux uses as drivers the incoming shortwave radiation (`SW_IN`), air temperature (`TA`) and Vapour pressure deficit (`VPD`). If `TA` or `VPD` is missing, `SW_IN` is used as the only driver. If all drivers are missing, the mean value at the same of the day is used for gap filling. The starting size of the time window is 7 days.
MDS has a quality flag with 3 possible values (i.e. 1,2,3) that depends on the size of the time window. In ONEFlux MDS is used only if the quality flag is 1, which means that similar conditions are found in a time window smaller than 14 days.

ERA-Interim ERA-Interim is a global meteorological dataset provided by the European Centre for Medium-range Weather Forecast (ECMWF). Weather forecast models are used to reanalyse past observations and produce a continuous and complete dataset for all the globe. The main drawback is the low spatial resolution and temporal resolution, that are respectively 0.7° (roughly 38km) and 3 hours. Moreover, only a subset of the meteoroligical variables are available in ERA-I. ONEFlux reduces the error of the ERA-I data by doing a bias correction with a linear regression and temporally downscaled to match the half-hourly frequency of FLUXNET [29].

Other methods ONEFLux is not the only EC post-processing pipeline which gap fills meteorological data. However, the imputation approaches in other libraries, like REddyProc [30] or OzFlux [16] are very similar. OzFlux implementation differs as it includes data from both ERA-Interim and the Australian Weather Service and for each gaps select the dataset with the smallest error for a window of 90 days around the gap.

Limitations of current methods There are three possible direction to improve the current imputation methods 1) make a better use of temporal autocorrelation of the variables 2) combine different imputation approaches in one prediction 3) provide detailed information on the quality of imputation.

Temporal autocorrelation MDS uses the temporal autocorrelation only in a limited way, as it takes the average of the missing variable across the whole time window and doesn't attribute more weight to the observations closes to the gap, which have the highest correlation with the data in the gap. This is especially relevant for short and medium gaps, which are the majority of gaps in FLUXNET (Appendix figure 11). The bias correction for ERA-Interim doesn't take into consideration the observations around the gap, either. Therefore, there is potential in improving the imputation performance by making a better use of the temporal autocorrelation.

Combination of imputation approaches ONEFlux employs both ERA-I and MDS, but the two methods are used independently, not combined to improve the predictions, but they are used independently. The criteria to select the method to use is only the MDS quality control flags. The information on the missing data from temporal autocorrelation, correlation with other variables and other measurements can be combined to make one more accurate prediction.

Uncertainty A limitation of the current methods is the lack of a robust assessment of the uncertainty of the imputed values. MDS has a quality flag, but it derives from hardcoded values and has only 3 possible values, moreover in the final ONEFlux product the quality flag indicates only which gap filling method has been used. Ideally, each predicted data point has an associated uncertainty, which varies continuously and it is interpretable. In this way, the level of confidence of the model in each prediction is available to the data user. The uncertainty can be used either to discard the data above a custom threshold, which can change depending on the application, or directly included in the downstream calculations.

1.3 Kalman Filter

Imputation of missing values is a topic that has been extensively researched. A wide range of methods have been developed ranging from replacing with the mean to employing deep neural networks [13, 6, 11, 32, 7]. Specifically for meteorological time series there are many different methods [9, 17]. However, imputation in the EC contest has some specific characteristics: the absence of a spatial component (each EC site is modelled separately) and the relatively high number of variables. Moreover, the focus is to impute short and medium gaps (up to 1 week), as almost 99 % of gaps of meteorological variables in FLUXNET are shorter than a week (Appendix figure 11). For this work, we focused into methods that combine all imputation approaches, include interpretable uncertainty and can be applied globally.

There are many families of models that can be suitable for the task. Probabilistic machine learning algorithms are particularly suited, as they directly provide an interpretability uncertainty. Kalman Filter is a probabilistic model that models the evolution of a multivariate latent state over time using a Markov chain. The temporal autocorrelation and the variable correlation are directly considered by the model and is possible to also include ERA-I observations. Other modelling approaches were evaluated: Gaussian Process Factor Analysis and GP-VAE (Gaussian Processes

Variational AutoEncoders). Gaussian Processes Factor Analysis [31] use Gaussian Processes to model a latent variable over time, this is a powerful modelling approach and is fully probabilist. The main limitation is the computational complexity, which in the naive formulation scales cubically with the number of observations. GP-VAE [14] combines Gaussian Processes and deep learning, to provide high quality imputation with uncertainties. Kalman Filter was selected, since it's the simplest model of the one considered that still fulfils all the requirements.

The first goal of this work is to develop an imputation method for meteorological time series in the context of EC that employs a Kalman Filter. The imputation performance of the new method is then evaluated by comparing it with the state-of-the-art methods (ERA-I and MDS) and assessing the behaviour in different scenarios. For simplicity, only data from one EC site, Hainich (Germany), will be used.

2 Methods

2.1 Kalman Filter Theory

Kalman Filter models over time a latent variable x, that represent the state of the system. The state cannot be directly observed, but we can observe meteorological variables y that reflect the state of the system. It is possible to use Kalman Filters to impute missing values, as the value of the state can be updated over time even when there are missing observations. The values of the state are hence available for all time steps, which can be used to then predict the missing observed variables. Kalman Filter is a probabilist machine learning algorithms, so it keeps track of the entire distribution of the latent state $p(x_t)$. The time is considered to be discrete, the state is modelled only at specific times x_t.

In order to be to model the state over time, assumptions on the behaviour of the system are made. There are three key assumptions 1) the states are connected by a Markov chain, which means that the state at time t depends only on the state at time $t-1$ and not the states at previous times $p(x_t|x_{t-1}) = p(x_t|x_{t-1}, x_{t-2}, \ldots, x_0)$ 2) The value of the observed variable depends on the latent state 3) all the relationships are linear and all distributions are Gaussian. Additionally, the mean of the state at time t may depend also on an external control variable. This control variable doesn't depend on the state of the models, but provide information on how the state mean should change. Equations 1 and 2 describe the assumptions on the behaviour of the system:

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; Ax_{t-1} + b + Bc, Q) \tag{1}$$

$$p(y_t|x_t) = \mathcal{N}(y_t; Hx_t + d, R) \tag{2}$$

The probability distributions of the state are computed using Bayesian inference. The computational cost of probabilistic inference can be drastically reduced in this context, as can be performed can be performed using linear algebra operations since all the relations are linear and all distributions are Gaussian.

Kalman Filter is a recursive algorithm (Figure 1), at time t the *predicted state* (x_t^-) is obtained from the previous state (x_{t-1}) and the *control variable* (c_t). Then the state is update using the *observation* (y_t) to obtain the *filtered state*, observations can be partially or totally missing. This is repeated recursively for all time steps. At this point, at each time step, the state x_t depends only on the observations until time t. The *smoothed state* (x_t^s) is the final state that depends also on all the observations after time t. The smoothing phase works by starting from the last time step and recursively updating x_t^s using x_{t+1}^s, x_{t+1} and x_{t+1}^-. Finally, for all the time steps where there is a gap, the *predicted observations*, $\hat{y}_t^g$, are calculated from the state x_t.
The model always considers the probability distribution for the state $p(x_t) = \mathcal{N}(x_t; m_t, P_t)$, for each state at each time step the mean (m_t) and the covariance are (P_t) are stored that are the parameters for a multivariate Gaussian distribution. Similarly, the model predictions are a distribution $p(\hat{y}_t) = \mathcal{N}(\hat{y}_t; \mu_{y_t}, \Sigma_{y_t})$.

2.1.1 Time update

The first step in a Kalman Filter is computing the probability distribution of the predicted state x_t^-, from the state at the previous time step x_{t-1} and the control variable c_t. The predicted state distribution is $p(x_{t-1}) = \mathcal{N}(m_{t-1}, P_{t-1})$. Using equation 1 and the properties of a linear map of Gaussian distributions the following equation can be derived:

$$p(x_t^-) = \mathcal{N}\left(x_t^-; m_t^-, P_t^-\right) \tag{3}$$

$$m_t^- = Am_{t-1} + Bc_t + d \tag{4}$$

$$P_t^- = AP_{t-1}A^T + Q \tag{5}$$

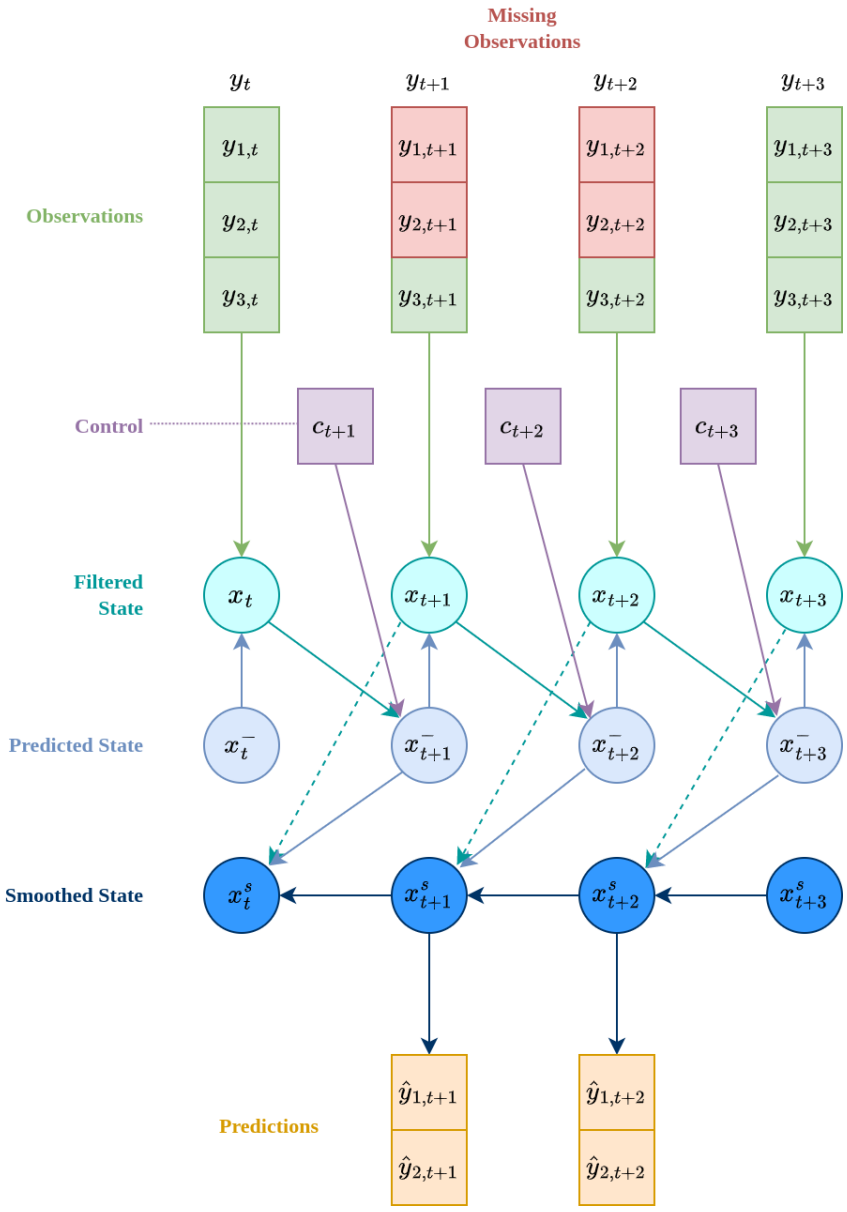

Figure 1: Schematic representation of a Kalman Filter

2.1.2 Measurement update

The predicted probability distribution is the updated to obtain the distribution of the filtered state, using the current observation (y_t). Equation 2 describes the distribution of y_t given x_t, using Bayes theorem is possible to compute the distribution of x_t given an observation y_t.

$$p(x_t|y_t) = \mathcal{N}(x_t; m_t, P_t) \tag{6}$$
$$z_t = Hm_t^- + d \tag{7}$$
$$S_t = HP_t^- H^T + R \tag{8}$$
$$K_t = P_t^- H^T S_t^{-1} \tag{9}$$
$$m_t = m_t^- + K_t(y_t - z_t) \tag{10}$$
$$P_t = (I - K_t H)P_t^- \tag{11}$$

Missing observations The Kalman Filter is robust to missing data and can update the state even though there is missing data. If all the observations at time t are missing, the measurement update step is skipped and the filtered (x_t) is the same of the predicted state (x_t^-). If only some observations in y_t are missing, then a partial measurement step is performed. The vector containing the observations that are not missing at time t, y_t^{ng}, can be expressed as a linear transformation of y_t

$$y_t^{ng} = My_t \tag{12}$$

where M is a mask matrix that is used to select the subset of y_t that is observed. $M \in \mathbb{R}^{n^{ng} \times n}$ and is made of rows which are made of all zeros but for an entry 1 at column corresponding to the of the index of the non-missing observation.
For example, if $y_t = [y_{0,t}, y_{1,t}, y_{2,t}]^T$ and $y_{0,t}$ is the missing observation then

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

using the properties of linear projections of Gaussian distribution we can then derive the distribution $p(y_t^{ng} \mid y_t)$ and from it $p(y_t^{ng} \mid x_t)$

$$p(y_t^{ng}|y_t) = \mathcal{N}\left(y_t^{ng}; M\mu_{y_t}, M\Sigma_{y_t}M^T\right) \tag{14}$$
$$p(y_t^{ng}|x_t) = \mathcal{N}\left(y_t^{ng}; MHx_t + Mb, MRM^T\right) \tag{15}$$

Therefore, it is possible to perform the measurement update step when some observations are missing using a variation of equation 6, where H is replaced by MH, b by Mb and R by MRM^T.

2.1.3 Smoothing

In the smoothing step, the filtered state at time t is update using the state $t+1$ corrected by the measurement update. A set of equations for the smoothing pass of a Kalman Filter has been derived by Rauch-Tung-Striebel [24]. They calculate the smoothed state x_t^s from the smoothed,

filtered and predicted state at the successive time step. For the last time step, the smoothed state is set to be equal to the filtered state.

$$p(x_t^s \mid Y) = \mathcal{N}(x_t^s; m_t^s, P_t^s) \tag{16}$$
$$G_t = P_t A^T (P_{t+1}^-)^{-1} \tag{17}$$
$$m_t^s = m_t + G_t(m_{t+1}^s - m_{t+1}^-) \tag{18}$$
$$P_t^s = P_t + G_t(P_{t+1}^s - P_{t+1}^-)G_t^T \tag{19}$$

2.1.4 Predictions

From the state (x_t) it is possible to directly obtain the predictions of the model $\hat{y}_t^g$ by using equation 2 and a mask, define like in equation 12

$$p(\hat{y}_t^g) = \mathcal{N}(\hat{y}_t^g; \mu_{y_t}, \Sigma_{y_t}) \tag{20}$$
$$\mu_{y_t} = MHx_t + Md \tag{21}$$
$$\Sigma_{y_t} = MRM + MHP_t^s H^T M^T \tag{22}$$

2.2 Kalman Filter Implementation

2.2.1 Requirements

Kalman Filter are a widely used algorithm and there are several python libraries that implement Kalman Filter (e.g. `statsmodels`, `pykalman`, `filterpy`). However, no Kalman Filter library was identified which meets all the requirements for this context. It is necessary to support gaps, partial measurements updates, control variables and be a numerically stable implementation. Therefore a custom library for Kalman Filters was developed using the PyTorch library, which has the advantage of automatic differentiation, possibility to use GPUs and better integration with other Machine Learning methods.

2.2.2 Numerical stability

Background The direct implementation of the Kalman filter equations suffer by numerically stability issues [20, 10]. Numerical instability arises from the fact that digital computers store numbers only with a limited number of decimal digits. This results in a loss of information, so that some operations may be incorrectly performed by a computer (e.g. summing a big number and a small number).
For Kalman Filter the components that are most affect by numerical instability are the covariance matrices. To analyse the stability of the operations on these matrices it is relevant to consider the condition number for inversion [20, 18], which describes if the matrix is going to be singular on the numerical representation in the computer. The condition number $k(A)$ is the ratio between the biggest singular value and the smallest. The singular value is $\sigma^2(A) = \lambda(AA^T)$, with $\lambda(A)$ being the eigenvalue of A.

$$k(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} \tag{23}$$

The condition number it's 1 for well-conditioned matrices and tends to infinite for ill-conditioned matrices. As a general rule a matrix cannot be inverted when the reciprocal of the condition number for inversion is close to the machine precision $1/k(A) < \varepsilon$ [20].

Mitigation strategies

Machine precision The simplest to improve the numerical stability is to use higher accuracy in the representation of numbers [10]. Practically, this means to use 64bit floats instead of 32bit floats, which is default in PyTorch.

Matrix decomposition Another way to improve the numerical stability is to reduce the condition number of the state covariance (P). A positive definite matrix has a square root factor, $P^{1/2}$, such as that $P = P^{1/2}(P^{1/2})^T = P^{1/2}(P^{T/2})$. The Cholesky decomposition is an algorithm to find a square root of a matrix, however the Cholesky decomposition calculates only one of possibly many square roots of the matrix.
Utilizing $P^{1/2}$ instead of P doubles the effective numerical resolution of the filter [18] [10] [26]. This is due to the fact that the eigenvalues of $P^{1/2}$ are the square root of the eigenvalues of P, $\lambda(P) = \lambda^2(P^{1/2})$, thus the conditioning number of P is the square of the conditioning number of $P^{1/2}$. Therefore, if in the filter implementation P is never explicitly computed, the numerical stability of the filter is significantly improved. There are several implementations of a Kalman Filter that follow this approach ([23], [8], [4]) and are generally called "square-root filter".

2.2.3 Implementation in PyTorch

There are different approaches for square root filtering. According to [20] the best approach is the UD Filter ([4]), since it has the smallest computational cost. However, the filter is based on the UD factorization and a custom matrix factorization [20] and both of those algorithms cannot be efficiently implemented in PyTorch. The PyTorch function `torch.linalg.ldl_factor` performs an UD factorization, but it's an experimental function and is not differentiable. Moreover, the custom matrix factorization would need to be implemented using scalar operations, which aren't efficient with PyTorch eager execution.
For this reason, a square root filter that propagates Cholesky factors of the covariance matrices is implemented. In this way all the required computations can be expressed in QR factorization, which is a numerically stable method and is a routine implemented in PyTorch.

2.2.4 Time update Square Root Filter

From the equations of the time update step (eq. 3) is possible to derive an algorithm to obtain $P_t^{1/2}$ given $P_{t-1}^{1/2}$, without explicitly computing P_t or P_{t-1}. The equations here described are from [20] eq. 6.60.
Defining

$$W = \begin{bmatrix} AP_{t-1}^{1/2} & Q^{1/2} \end{bmatrix} \tag{24}$$

from equation 3 the following is true:

$$WW^T = P_t \tag{25}$$

$$WW^T = \begin{bmatrix} AP_{t-1}^{1/2} & Q^{1/2} \end{bmatrix} \begin{bmatrix} P_{t-1}^{T/2} A^T \\ Q^{T/2} \end{bmatrix} = AP_{t-1}^{1/2} P_{t-1}^{T/2} A^T + Q^{1/2} Q^{T/2} = AP_{t-1}A^T + Q = P_t \quad (26)$$

The next step is to factorize $W = LU$, where L is a lower triangular matrix and U is an orthogonal matrix, such as that $UU^T = I$. Then $WW^T = LU(LU)^T = LUU^T L^T = LL^T = P_t$. Hence, L is a square root of P_t.
This procedure never explicitly compute P_t and requires only the factorization of a matrix, which is implemented efficiently and in a numerical stable way in the PyTorch `torch.linalg.qr` function.

PyTorch implementation PyTorch doesn't support natively a LU decompositions. It implements the QR factorization: $W = QR$, where Q is an orthogonal matrix and R an upper triangular matrix. This can be easily converted into a LU factorization, as by factorizing W^T then $W^T = QR = (QR)^T = R^T Q^T$ and R^T is a lower triangular matrix.

Summary The steps of the Square Root time update are:

1. let $W = \begin{bmatrix} AP_{t-1}^{1/2} & Q^{1/2} \end{bmatrix}$
2. do a QR factorization $W^T = TR$
3. set $P_t^{1/2} = R^T$

2.2.5 Measurement update Square Root Filter

A similar procedure can be followed for the measurement update step of the filter. The equations here described are from [10].
The starting point is equation 6, for simplicity the time subscripts are omitted in the following equations.
Defining:

$$M = \begin{bmatrix} R^{1/2} & H(P^-)^{1/2} \\ 0 & (P^-)^{1/2} \end{bmatrix} \quad (27)$$

$$V = \begin{bmatrix} S^{1/2} & 0 \\ \bar{K} & P^{1/2} \end{bmatrix} \quad (28)$$

$$\bar{K} = KS^{1/2} \quad (29)$$

the following is true:

$$MM^T = VV^T \quad (30)$$

$$
\begin{aligned}
MM^T &= \begin{bmatrix} R^{1/2} & HP^- \\ 0 & (P^-)^{1/2} \end{bmatrix} \begin{bmatrix} R^{T/2} & 0 \\ (P^-)^{T/2}H^T & (P^-)^{T/2} \end{bmatrix} = \\
&= \begin{bmatrix} R^{1/2}R^{T/2} + H(P^-)^{1/2}(P^-)^{T/2}H^T & HP^-)^{1/2}P^-)^{T/2} \\ (P^-)^{T/2}P^-)^{1/2}H^T & (P^-)^{1/2}P^-)^{T/2} \end{bmatrix} = \\
&= \begin{bmatrix} S & HP^- \\ (P^-)^T H^T & P^- \end{bmatrix}
\end{aligned} \tag{31}
$$

$$
\begin{aligned}
VV^T &= \begin{bmatrix} S^{1/2} & 0 \\ \bar{K} & P^{1/2} \end{bmatrix} \begin{bmatrix} S^{T/2} & \bar{K}^T \\ 0 & P^{T/2} \end{bmatrix} = \begin{bmatrix} S^{1/2}S^{T/2} & S^{1/2}\bar{K}^T \\ \bar{K}S^{T/2} & \bar{K}\bar{K}^T + P^{1/2}P^{T/2} \end{bmatrix} \\
&= \begin{bmatrix} S & S^{1/2}S^{T/2}K^T \\ KS^{1/2}S^{T/2} & KS^{1/2}S^{T/2}K^T + P \end{bmatrix} \\
&= \begin{bmatrix} S & HP^- \\ P^-H^T & KHP + P \end{bmatrix}
\end{aligned} \tag{32}
$$

We can see that all blocks of VV^T are directly equal to MM^T, but the bottom left one, which is equal due to the measurement update for the covariance (equation 11).
Therefore, if we decompose $M = LU$ then $MM^T = LL^T = VV^T$ and the bottom left block of U of size $k \times k$ of L is a square root of P.

Summary The steps of the Square Root measurement update are:

1. let $M = \begin{bmatrix} R^{1/2} & H(P^-)^{1/2} \\ 0 & (P^-)^{1/2} \end{bmatrix}$
2. do a QR factorization of $M^T = TU$
3. $P^{1/2}$ is the bottom left $k \times k$ block of U

2.2.6 Predictions Square Root Filter

The prediction equation for the square root filter are similar to the equations for the time update. defining:

$$
W = \begin{bmatrix} HP_t^{1/2} & R^{1/2} \end{bmatrix} \tag{33}
$$

from equation 20 the following is true:

$$
WW^T = \Sigma_{y_t} \tag{34}
$$

$$
WW^T = \begin{bmatrix} AP_t^{1/2} & R^{1/2} \end{bmatrix} \begin{bmatrix} P_t^{T/2}H^T R^{T/2} \end{bmatrix} = HP_t^{1/2}P_t^{T/2}H^T + R^{1/2}R^{T/2} = HP_tH^T + R = \Sigma_{y_t} \tag{35}
$$

Summary The steps of the Square Root predictions are:

1. let $W = \begin{bmatrix} HP_t^{1/2} & R^{1/2} \end{bmatrix}$
2. do a QR factorization of $W^T = TU$
3. set $\Sigma_{y_t}^{1/2} = U^T$

2.2.7 Smoother Square Root Filter

The available literature for implementing Square Root Smoothing is scarce compared to Square root filter, so no solution has been identified to implement a square root smoother. Therefore, a standard smoother is employed.
Nonetheless, steps were taken to improve the numerical stability of the smoother. The computation in the smoother that is most numerically unstable is the inversion of P_{t+1}^- in equation 17 [20]. The matrix inversion is avoided by using the `torch.cholesky_solve` function. It solves for X the linear system $P_{t+1}^- X = P_t A$, which is equivalent of computing $X = (P_t A^T (P_{t+1}^-)^-1)^T$. This use directly the Cholesky Factor $(P_{t+1}^-)^{1/2}$ to avoid the computation of P_{t+1}^-. A further step to improve the numerical stability is forcing the covariance matrix to be symmetric, by averaging to upper and lower part at after every time step $P_{t,sym}^s = (P_t^s + (P_t^s)^T)/2$, as suggested in [10]. This approach to numerical stability in the smoother is the also applied by the `statsmodels` library [28].

2.3 Kalman Filter Model

2.3.1 Parameters

The Kalman Filter is implemented as PyTorch module, whose parameters are described in Table 1. There is no change over time of the parameters, and the state of the filter is initialized always at the same value from the parameters m_0 and P_0.

Constraint An important aspect for implementing a Kalman Filter in PyTorch is constraining the parameters that represents covariance (Q, R and $P0$) to be positive definite. To achieve this goal the optimizer works on a raw parameter, which is then transformed into a positive definite matrix. The transformation into a positive definite matrix is done by transforming the raw parameter into a lower triangular matrix with a positive diagonal. The diagonal is enforced to be positive by transforming the diagonal of the raw parameter with the softplus function ($x = \log(1 + e^x)$), which is a positive function. In addition a small positive offset 1×10^{-5} is added to the diagonal in order to avoid that the diagonal is close to zero, which would result in a positive semi-definite matrix.
The inverse of the positive definite transformation is implemented, so that parameters can be manually set.
This implementation of the positive definite constraint makes it is that is straightforward to obtain the Cholesky factor of the parameters, which are needed by the Square Root Filter, and at the same time the full parameter, which are needed by the smoother.

Table 1: Parameters of the Kalman Filter Model. n is the number of dimension of the observations, k the number of dimensions of the state, n_{ctr} the number of dimensions of the control variable. Λ is the factor loading matrix of the PCA for the entire dataset.

Parameter name	Notation	Shape	Initial value
State transition matrix	A	$k \times k$	$\begin{bmatrix} I & I \\ 0 & I \end{bmatrix}$
Observation matrix	H	$n \times k$	$\begin{bmatrix} \Lambda^T & 0 \end{bmatrix}$
State transition covariance	Q	$k \times k$	diag(0.1)
Observation covariance	R	$n \times n$	diag(0.01)
State transition offset	d	k	0
Observation offset	b	n	0
Control matrix	B	$k \times n_{ctr}$	$\begin{bmatrix} -I & I \\ 0 & 0 \end{bmatrix}$
Initial state mean	m_0	k	0
Initial state covariance	P_0	$k \times k$	diag(3)

2.3.2 Parameters initialization

The model parameters could be initialized using random values, however this would increase numerical stability issues and increase the training time. Moreover, if the initial parameters are very distant from the optimal ones it is more likely for the optimization algorithm to find a local minimum. The simplicity of the Kalman Filter and the interpretability of its parameters allows to manually initialize the parameters with realist values.

State transition matrix A is initialized using a "local linear trend" model [12]. The idea is that half of the state $x_{l_{t-1}}$ represent the level of the current state and the second half the slope $x_{s_{t-1}}$ of a linear function that describes the rate of change of the state between time steps. The next state level is equal to the current level plus the slope and a random change, while the slope remains constant. The use of a slope allows the model to retain information of several previous states.

Observation Matrix H is initialized by using the transpose of the factor loadings matrix of the principal component analysis of the observed variables. In this way, there isn't a one to one relation between the state and the observation, but one variable in the state contains information about several meteorological variables. This should improve the predictions in the presence of partial gaps. The second part of H is 0 as in a local trend model the observations don't depend on the slope but only the level of the state

Control matrix B is initialized to the difference between the previous observation and the current observation. The number of dimensions of the control is potentially different from the state and the observations, therefore is difficult to find a proper correspondence between the state and the control. The solution found is to initialize the upper part of the matrix to the difference between the control, which roughly correspond to the variables, and then pad the rest with zero to match the correct shape.

Covariance The state transition covariance Q and then observation covariance R are initialized as diagonal matrix with values of 0.1 and 0.01 respectively. This number has been chosen to represent an uncertainty in the state transition that is compatible with the standard deviation of the variables (1 as they are standardized) and a low uncertainty in the observations.

Offsets The observation and state transition offsets are initialized to zero.

Initial state The initial state is set to have as mean zero and as covariance diag(3). The number 3 is an arbitrary number bigger than the state transition covariance, which should represent the high level of uncertainty for the initial state.

2.3.3 Loss Function

The loss function is used to train the model is the negative log likelihood, computed for each data point. At each time step, the model predicts a multivariate normal distribution $p(\hat{y}_t^g)$, which is used to compute the negative log likelihood given the actual observations y_t^g. The negative log likelihoods between different time steps in the same gaps are summed. Then negative log likelihood is averaged between batches.
The actual loss function of the model should be the log likelihood of the joint distribution $p(Y^g)$. However, the analytical form of the joint distribution cannot to easily derived from the Kalman Filter equations. The log likelihood of marginal distributions is instead used, as it is a lower bound to the log likelihood of the joint distribution. Defining $q(x)$ the predicted joint distribution, $p(x)$ the real joint distribution and $q_i(x)$ the marginal distribution at the ith time step

$$q_i(x) = \int q(x_1, ..., x_k) dx_{\neg i} \tag{36}$$

Then, if the family of distribution of $q(x \mid \theta)$, is the same of $\prod_i q_i(x \mid \theta)$, where θ are the model parameters. Then

$$\max_\theta \langle \log q(x \mid \theta) \rangle_{x \sim p(x)} \geq \max_\theta \langle \log \prod_i q_i(x \mid \theta) \rangle_{x \sim p(x)} \tag{37}$$

because $\prod_i q_i(x)$ is more restricted. This means that $q(x)$ fit at least as good as $\prod_i q_i(x)$. For the Kalman Filter $q_i(x)$ is a Gaussian distribution, so $\prod_i q_i(x)$ is also a Gaussian distribution and equation 37 is true.

2.3.4 Metrics

The main metric used to assess the model performance is the *Root Mean Square Error* (RMSE).

$$RMSE = \sqrt{\frac{\sum_i^n (y_i^g - \hat{y}_i^g)^2}{n}} \tag{38}$$

The advantage of the RMSE is that it can be used also for non-probabilistic methods (e.g. MDS) and that its value has the same physical dimension as the observed variable. The main drawback is that is cannot be used for comparison between variables. For that, the *standardized RMSE* is used, which is the RMSE computed on the standardized variables.

2.3.5 Performance considerations

The iterative nature of the filter, where the current state depends on the previous state, makes it impossible to use PyTorch vectorization across different time steps. This can significantly limit the performance of the filter, especially when executed on GPUs. In order to mitigate this issue, all functions in the Kalman Filter library support batches, so at every time step different data is processed in parallel.

2.4 Data

2.4.1 Data source

The data used to evaluate the performance of Kalman Filters is from the Hainich (Germany) site, which is managed by the University of Göttingen. The source of the data is the FLUXNET 2015 Dataset, which includes measurements with a 30 mins frequency between 2000 and 2012 for Hainich. In total 227952 observations are available. For simplicity, the entire dataset was used for the model training, which includes also gap-filled observations. All the meteorological variables that are gap-filled in the FLUXNET 2015 dataset were selected for the analysis (Table 2).

ERA-Interim The control variables used in the Kalman Filter are the bias corrected and down-scaled ERA-I observations included in the FLUXNET dataset. The variables of interest are also present in ERA-I, except for TS and SWC.

Table 2: Meteorological variables used to evaluate the Kalman Filter imputation. ERA-I column indicates whether the variable is available in the ERA-Interim dataset

Variable Name	Abbreviation	Unit	ERA-I
Air Temperature	**TA**	°C	✓
Incoming Shortwave Radiation	**SW_IN**	W/m^2	✓
Incoming Longwave Radiation	**LW_IN**	W/m^2	✓
Vapour Pressure Deficit	**VPD**	hPa	✓
Wind Speed	**WS**	m/s	✓
Air Pressure	**PA**	hPa	✓
Precipitation	**P**	mm	✓
Soil Temperature	**TS**	°C	✘
Soil Water Content	**SWC**	%	✘

2.4.2 Data preparation pipeline

The dataset needs to be pre-processed by dividing into data blocks, adding an artificial gap and then standardize. The data preparation pipeline takes as input a list of items and outputs the data in a format suitable for training. Each item provides all the information about a gap with the following fields a) `i` the index of the block b) `shift` the shift c) `var_sel` the variables in the gap d) `gap_len` the gap length. The pipeline perform the following steps: 1) split the index of complete data frame from Hainich into blocks of a given length and selects the ith element 2) adds the shift to move the starting point of the data block and select the data from the data frame. For the control variable it also adds the observations with a lag 1, so that at the time t the model has access to the control variable both at time t and $t-1$ 3) creates one continuous artificial gap in the middle of the block for the variables specified in `var_sel` and with a length of `gap_len` 4) convert from Pandas data frame to a PyTorch Tensor 5) Standardize each variable, using the mean (μ_Y) and standard deviation (σ_Y) of the whole dataset.

$$y_t^z = \frac{(y_t - \mu_Y)}{\sigma_Y} \tag{39}$$

After this, the tensors are collated into a batch and potentially moved to the GPU.

2.4.3 Prediction pipeline

The model predicts the mean and the covariance for each time steps for the standardized variables. This needs to be converted back to be scale of the variable to be used for imputation. This operation needs to scale the whole distributions and not only the mean of the prediction. The standardized prediction $\hat{y}_t^z$ is distributed $p(\hat{y}_t^z) = \mathcal{N}\left(\hat{y_t}^z; \mu_{\hat{y}_t}^z, \Sigma_{\hat{y}_t}^z\right)$, the prediction in the original scale $\hat{y}_t$ is distributed $p(\hat{y}_t) = \mathcal{N}\left(\hat{y}_t; \mu_{\hat{y}_t}, \Sigma_{\hat{y}_t}\right)$ and $\Sigma_Y = \text{diag}(\sigma_Y)$.
Then from the inverse of equation 39

$$\hat{y}_t = \Sigma_Y \hat{y}_t^z + \mu_Y \tag{40}$$

Then using the properties of the linear projections of Gaussian distributions

$$p(\hat{y}_t) = \mathcal{N}\left(\hat{y}_t; \Sigma_Y \mu_{\hat{y}_t}^z + \mu_Y, \Sigma_Y \Sigma_{\hat{y}_t}^z \Sigma_Y^T\right) \tag{41}$$

2.5 Model Training

The available data is split between training and validation set, the first 80% of the data points used for training, the remaining 20% for validation. The split is not random, so the validation set doesn't contain periods of time close to the one measured.
The filter is initialized with 9 dimensions for the observations (one for each variable). For the state 18 dimensions, the first 9 dimensions are to match the number of observed variables and the other 9 are for the slope in the local trend model initialization. The control has 14 dimensions, with 7 for the control at time t and the other 7 for the control at time $t-1$.

Generic model The first model to be trained is a generic model, where each data block has a gap in one variable with the length of sampled from a uniform distribution between 12 (6 hours) and 336 (1 week). The missing variable is sampled with equal probability from the list of all variables. The shift is sample from a normal distribution with mean 0 and standard deviation 50. For each block of data in the original data frame 10 different artificial gaps were created, resulting in a total of 2080 data blocks used for training and 520 for validation. The length of the block of data is 446, so that at least 50 observations are available to the model before and after the gap. The batch size is 20. The model was trained for 3 epochs with a learning rate of 1×10^{-3}.

Variable fine-tuning The generic model has been fine tune for each variable, resulting in 9 different models. The training settings are the same for the generic model, expect that the gaps is only in one variable and then number of repetitions for each is 5 (training XXX blocks validation XXX blocks). Each variable, was fine tuned with a different number of epochs depending on the variable: TA 4 epochs, SW_IN 4 epochs, LW_IN2 epochs, VPD 2 epochs, WS 2 epochs, PA 3 epochs, P 0 epochs, TS 4 epochs, SWC 4 epochs. The learning was manually stopped when the training loss started being constant or the validation loss started to increase.

Gap all variables A version of the model was trained with a gap in all variables. The length of the gap was 30 for numerical stability issues. For gaps in all variables with a length of more than 30, the predicted covariance becomes not positive definite, hence is impossible to compute the log likelihood. The model was trained from the beginning for 3 epochs with a learning rate of 1×10^{-3}

No Control The last version of the model is one where the use of the control variables was disabled. The generic model was fine-tuned for 3 epochs with a learning rate of 1×10^{-3}.

2.6 Other methods

2.6.1 MDS

The implementation of the MDS used in the results comparison is from REddyProc ([30]). This package has been used because it provides an R interface, that can be easily integrated with Python. Conversely, ONEFlux implements only a C interface, whose integration in Python is significantly more challenging. The MDS algorithm in REddyProc and ONEFlux are fully equivalent. In detail, the function `REddyProc::sEddyProc_sMDSGapFill` was used, with the defaults setting of using SW_IN, TA an VPD are driver with a tolerance of 2.5 °C, 50 W/m^s and 5 hPa respectively as described in [**reichstein˙separation˙2005**]. The data provided to MDS has a context of at least 90 days around the gap, as required by REddyProc.

2.6.2 ERA

The imputation using ERA-Interim was performed by using the ERA variables available in the FLUXNET dataset without further correction.

2.7 Code Details and Availability

The code for this project has been developed in Python. The main libraries used are PyTorch for the model, FastAI for model training and data preparation, Altair plot plotting and Pandas and Polars

for data analysis. The source code is available at https://github.com/mone27/meteo_imp and the documentation of the library at https://mone27.github.io/meteo_imp/libs. The interactive version of the results ...

3 Results

3.1 Variables Correlation

- 3 variables (WS, PA, P) have low correlation with other variables
- TA is the variables that has the highest correlation with other variables. TS similar to TA
- `SWC` correlated only temperature
- Other variables (`SW_IN` `LW_IN`, `VPD`) have a correlation ranging between .4 and .6 with at least two other variables
- temporal autocorrelation is generally high across variables and decreases over tiem
- `TA`, `TS`, `SW_IN`, `VPD` have daily pattern with higher correlation after 24 hours. in particular `SW_IN` has a negative autocorrelation for a lga of 12 hours but a high for 24 hours
- `WS` and `LW_IN`, `PA` have an autocorrelation the decreases over time
- P has very low temporal autocorrelation

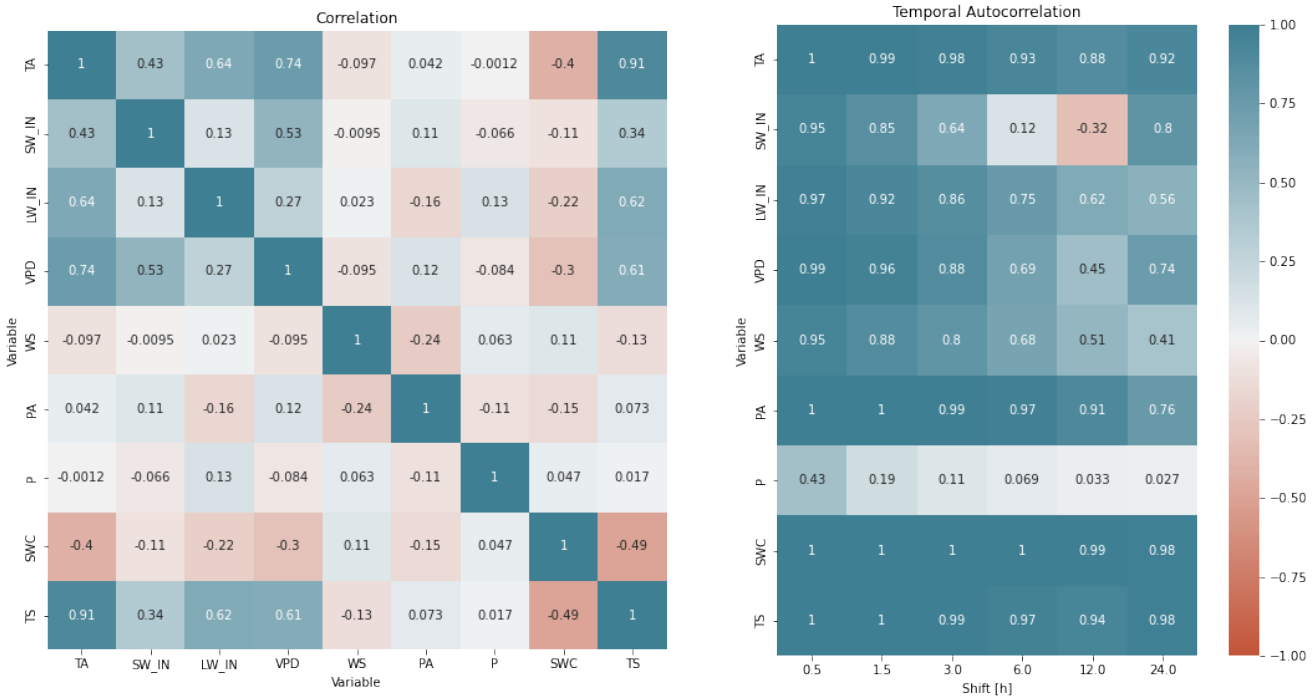

Figure 2: Correlation and Temporal autocorrelation of variables. Subfigure (a) shows the correlation between the meteorological variables. Abbreviations: Air Temperature `TA`, Incoming Shortwave Radiation `SW_IN`, Incoming Longwave Radiation `LW_IN`, Vapour Pressure Deficit `VPD`, Wind Speed `WS`, Air Pressure `PA`, Precipitation `P`, Soil Temperature `TS`, Soil Water Content `SWC`

3.2 Comparison to other imputation methods

- Overall Kalman Filter has a smaller error than ERA that has a smaller error than MDS. Exception is for `P`
- for long gap the relative performance of Kalman Filter is lower
- Kalman Filter has a lower variability of the error. For all variables std is smaller and the max error is smaller
    - `PA`/`TS`/`TA`/`SWC` are the variable with the lowest error and is all comparable with standardized RMSE around .6
    - `WS` is the variable with worse standardized RMSE

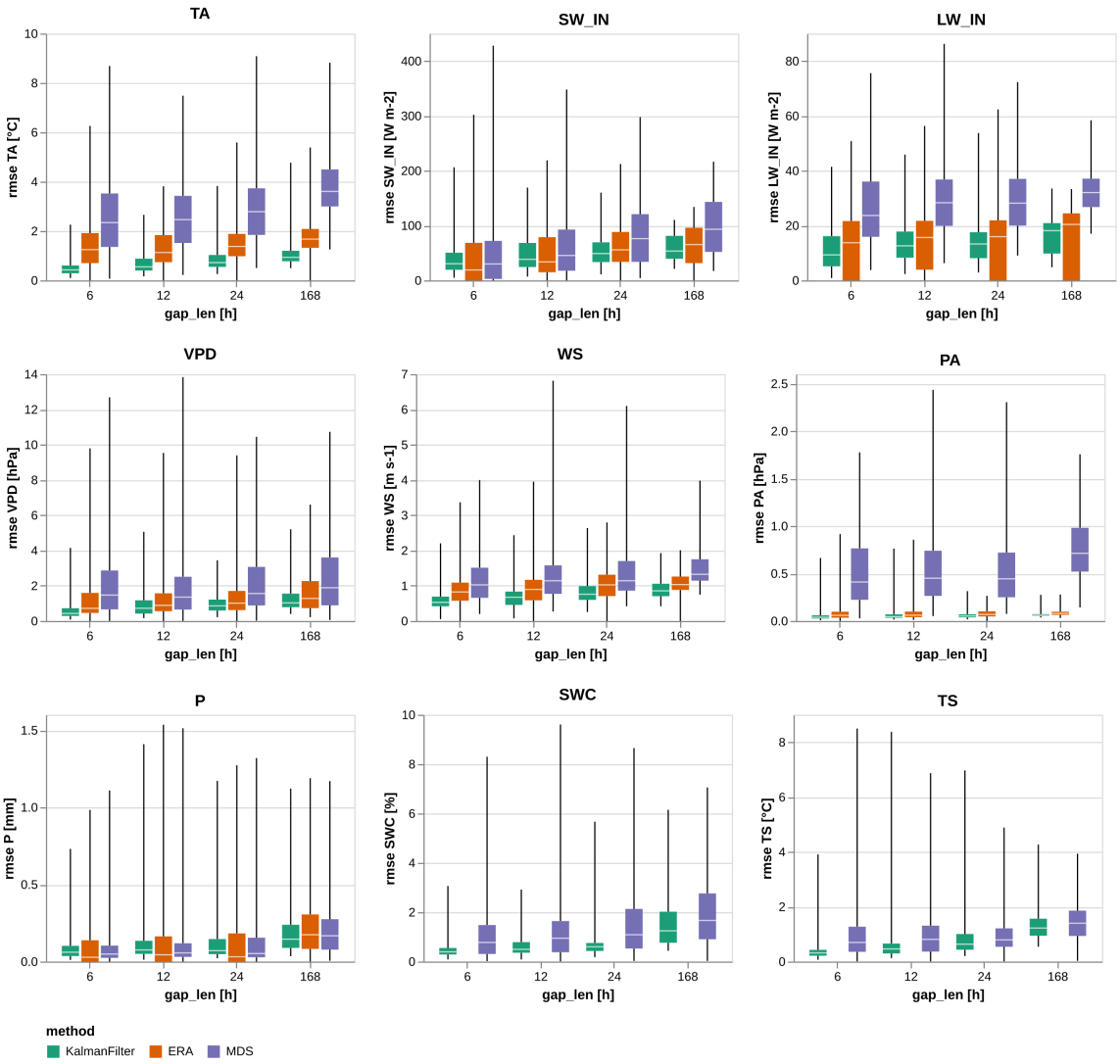

Figure 3: Box plot to compare Root Mean Square Error(RMSE) for each variable between the different methods: Kalman Filter and the state of the art methods ERA and MDS. Each box is delimitated by the first and third quartile, the white mark is the median and then vertical lines extend from the minimum to the maximum. For each variable and each gap length 400 artificial gaps has been made only in the variable of interest and the gap imputed using the different methods. The Kalman Filter model has been fine-tuned to each variable. ERA-I imputation is not available for `TS` and `SWC`

Table 3: RMSE Comparison imputation methods. The best method for each gap length is highligthed in bold

		KalmanFilter		ERA		MDS	
	RMSE	mean	std	mean	std	mean	std
Variable	Gap [h]						
TA [C]	6	**0.499**	0.297	1.353	1.024	2.667	1.893
	12	**0.692**	0.411	1.396	0.874	2.907	1.657
	24	**0.851**	0.458	1.510	0.778	2.879	1.520
	168	**1.088**	0.439	1.767	0.613	3.706	1.212
SW_IN [W/m^2]	6	**46.643**	43.146	52.648	68.819	67.157	88.813
	12	**51.466**	35.352	56.909	50.644	75.928	74.866
	24	**58.631**	31.420	66.977	42.979	88.078	61.912
	168	**60.990**	24.732	68.715	34.253	105.824	52.659
LW_IN [W/m^2]	6	**10.641**	7.930	13.459	12.386	26.929	14.703
	12	**13.469**	8.445	15.175	12.364	29.168	14.040
	24	**14.884**	8.151	15.399	12.359	28.622	11.842
	168	17.269	7.142	**16.722**	11.178	32.562	9.168
VPD [hPa]	6	**0.593**	0.439	1.330	1.570	2.138	2.216
	12	**0.847**	0.580	1.263	1.250	2.042	2.001
	24	**0.942**	0.570	1.295	1.086	2.049	1.853
	168	**1.260**	0.622	1.630	1.070	2.627	1.873
WS [m/s]	6	**0.582**	0.301	0.917	0.559	1.131	0.712
	12	**0.715**	0.320	0.964	0.529	1.232	0.757
	24	**0.819**	0.393	0.999	0.484	1.293	0.755
	168	**0.925**	0.310	1.064	0.328	1.503	0.574
PA [hPa]	6	**0.050**	0.066	0.081	0.098	0.517	0.408
	12	**0.058**	0.060	0.081	0.073	0.607	0.445
	24	**0.056**	0.024	0.076	0.042	0.567	0.409
	168	**0.070**	0.066	0.088	0.070	0.789	0.386
P [mm]	6	0.117	0.244	0.122	0.293	**0.117**	0.280
	12	0.122	0.220	0.123	0.259	**0.117**	0.237
	24	**0.151**	0.232	0.155	0.268	0.153	0.262
	168	**0.197**	0.177	0.222	0.199	0.215	0.197
SWC [%]	6	**0.507**	0.382	nan	nan	1.273	1.461
	12	**0.615**	0.469	nan	nan	1.304	1.456
	24	**0.727**	0.530	nan	nan	1.300	1.171
	168	**1.522**	1.037	nan	nan	1.869	1.476
TS [C]	6	**0.394**	0.323	nan	nan	1.044	0.999
	12	**0.612**	0.496	nan	nan	1.054	0.972
	24	**0.897**	0.879	nan	nan	1.086	0.877
	168	**1.447**	0.844	nan	nan	1.534	0.913

- `TA`: KF good performance and improvement over ERA. For long gaps follows ERA but often better. For reference, the accuracy of the thermometer installed at Hainich [1] is 0.1 C [27].
- `SW_IN`: KF performance at night is not good, there are negative values up to -50 and then a lot of variation even though should be constant at 0 and the control variable is correct. During the day methods are comparable with often Kalman Filter having the best performance
- `LW_IN` KF compare performance ERA. ERA has a much wider range of errors, for some gaps is very good for others is quite worse. MDS is quite bad basically predicting a constant value and the come drastic changes. LW_IN has a "blocky" behaviour that is correctly predicted by ERA
- `VPD`. KF is better especially for short gaps. MDS not very good miss variation at night (probably due to no change in SW_IN)
- WS has the worse standarized RMSE, a lot of short term variability that no model captures. KF still best results
- `PA` MDS significantly worse than other methods
- P KF worse performance than other methods, but still similar performance
- SWC KF has on average a better performance than MDS, especially for short gaps. However, the time series of KF are not very good. The variation in the predicted time series is often much higher than the actual one (Figure 6 14 16). ERA is not available
- TS KF has on average a better performance than MDS, especially for short gaps. However, the time series of KF are not very good. The variation in the predicted time series is often much higher than the actual one when there is a small variation in the variable(Figure 6 gap length 6h and 12h) conversely is much better when there is variation (Figure 14 16). the KF models correctly the increase and decrease in soil temperature even though if the values may not be accurate. ERA is not available

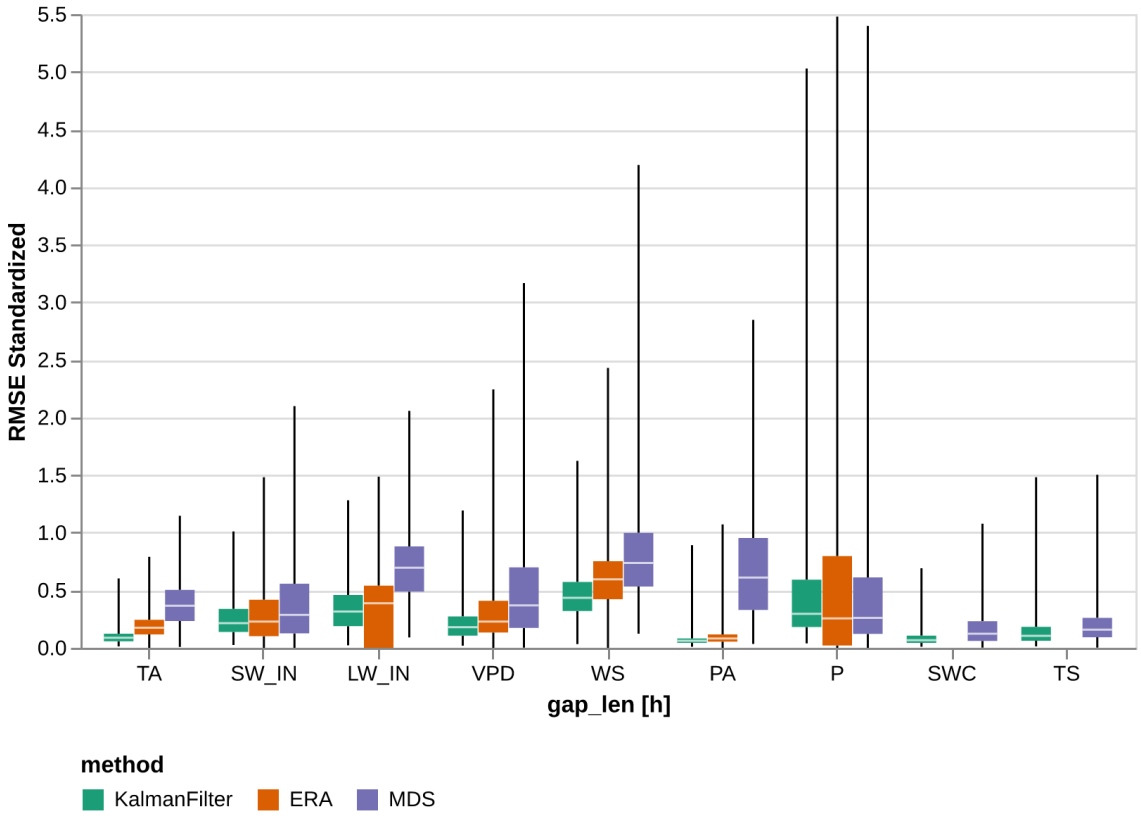

Figure 4: Box plot to compare Standardized Root Mean Square Error(RMSE) for each variable between the different methods: Kalman Filter and the state of the art methods ERA and MDS. The same data from figure 4 has been aggregated for all gap lengths

Table 4: Comparison of imputation methods using Standardized RMSE. The best method for each gap length is highligthed in bold

		KalmanFilter		ERA		MDS	
	RMSE	mean	std	mean	std	mean	std
Variable	Gap [h]						
TA	6	**0.063**	0.037	0.171	0.129	0.337	0.239
	12	**0.087**	0.052	0.176	0.110	0.367	0.209
	24	**0.107**	0.058	0.191	0.098	0.363	0.192
	168	**0.137**	0.055	0.223	0.077	0.468	0.153
SW_IN	6	**0.229**	0.211	0.258	0.337	0.329	0.435
	12	**0.252**	0.173	0.279	0.248	0.372	0.367
	24	**0.287**	0.154	0.328	0.211	0.432	0.303
	168	**0.299**	0.121	0.337	0.168	0.519	0.258
LW_IN	6	**0.254**	0.189	0.321	0.295	0.642	0.350
	12	**0.321**	0.201	0.362	0.295	0.695	0.335
	24	**0.355**	0.194	0.367	0.295	0.682	0.282
	168	0.412	0.170	**0.399**	0.266	0.776	0.219
VPD	6	**0.136**	0.100	0.304	0.360	0.489	0.507
	12	**0.194**	0.133	0.289	0.286	0.467	0.458
	24	**0.216**	0.130	0.296	0.248	0.469	0.424
	168	**0.288**	0.142	0.373	0.245	0.601	0.429
WS	6	**0.358**	0.185	0.564	0.344	0.696	0.438
	12	**0.440**	0.197	0.593	0.325	0.758	0.465
	24	**0.504**	0.242	0.615	0.297	0.795	0.465
	168	**0.569**	0.191	0.655	0.202	0.924	0.353
PA	6	**0.058**	0.077	0.095	0.114	0.605	0.477
	12	**0.068**	0.070	0.094	0.085	0.710	0.520
	24	**0.066**	0.029	0.089	0.049	0.663	0.478
	168	**0.082**	0.077	0.103	0.082	0.923	0.451
P	6	0.419	0.872	0.435	1.046	**0.418**	0.999
	12	0.435	0.784	0.438	0.924	**0.418**	0.847
	24	**0.537**	0.827	0.553	0.955	0.545	0.935
	168	**0.701**	0.632	0.793	0.711	0.766	0.704
SWC	6	**0.057**	0.043	nan	nan	0.143	0.164
	12	**0.069**	0.053	nan	nan	0.146	0.163
	24	**0.082**	0.060	nan	nan	0.146	0.131
	168	**0.171**	0.116	nan	nan	0.210	0.166
TS	6	**0.070**	0.057	nan	nan	0.184	0.177
	12	**0.108**	0.088	nan	nan	0.186	0.172
	24	**0.159**	0.155	nan	nan	0.192	0.155
	168	**0.256**	0.149	nan	nan	0.271	0.161

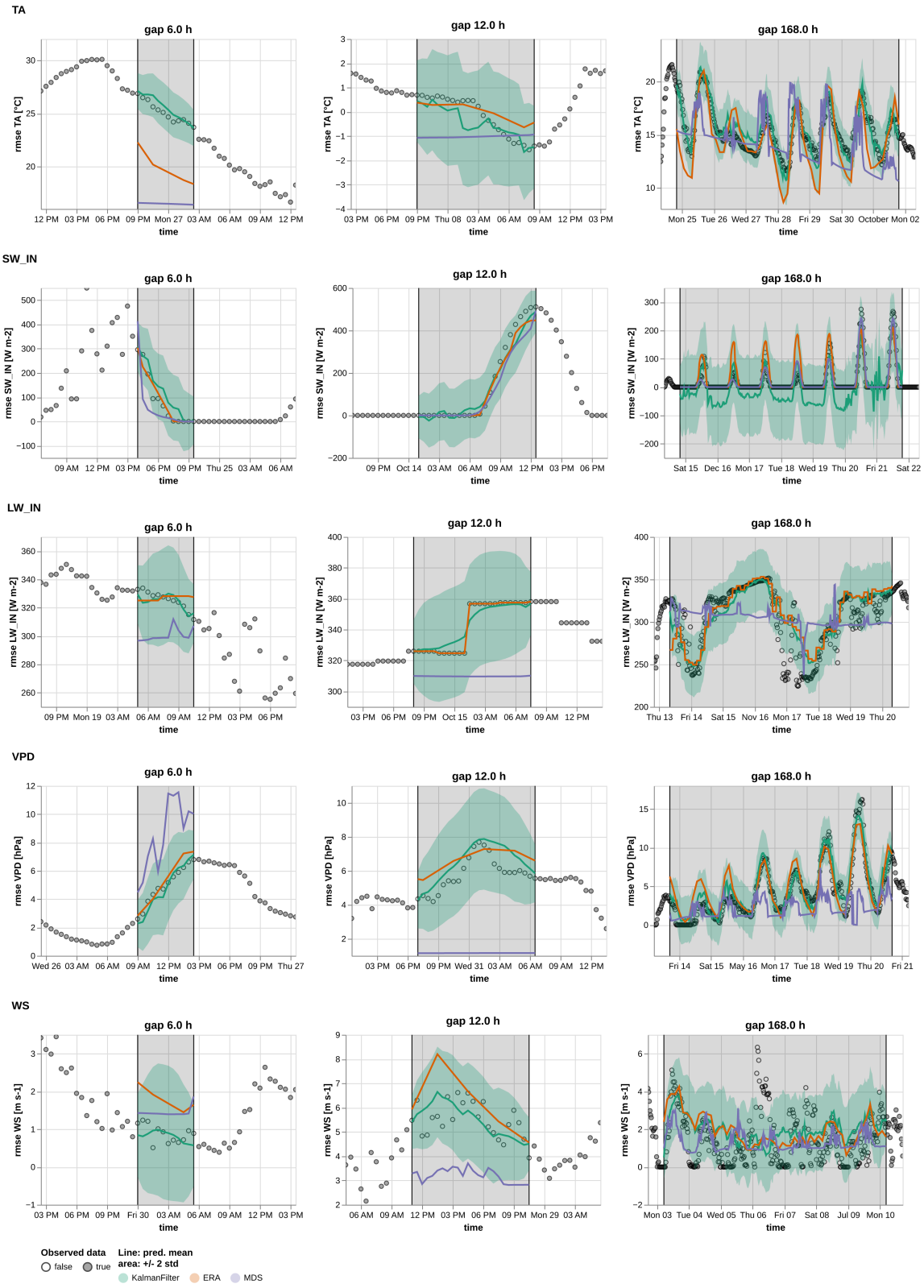

Figure 5: Example time series for `TA`, `SW_IN`, `LW_IN`, `VPD`, `WS`. For each variable, 3 random artificial gap (length 6 hours, 12 hours, 1 week) are imputed using the three methods: Kalman Filter (green), ERA-I (orange), MDS (purple). For the Kalman Filter the shared area show the uncertainty of the prediction $+/-2\sigma$. The vertical black line are the start and the beginning of the gap.

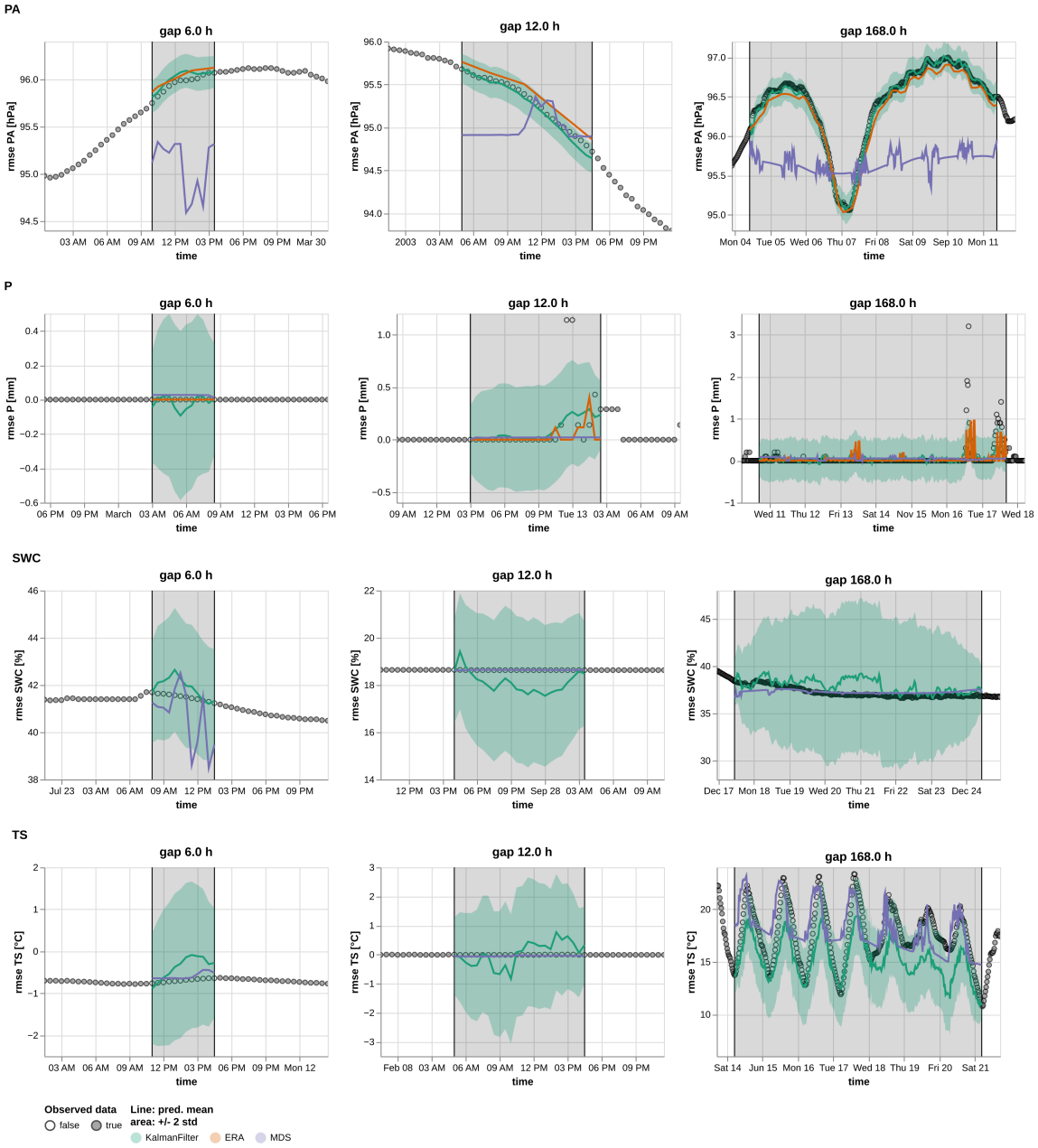

Figure 6: Example time series for `TA`, `SW_IN`, `LW_IN`, `VPD`, `WS`. For each variable, 3 random artificial gap (length 6 hours, 12 hours, 1 week) are imputed using the three methods: Kalman Filter (green), ERA-I (orange), MDS (purple). For the Kalman Filter the shared area show the uncertainty of the prediction $+/-2\sigma$. The vertical black line are the start and the beginning of the gap.

3.3 Analysis Kalman Filter

3.3.1 Gap Length

- short gaps have a smaller error
- then there is a steep increase for gaps up to 24 hours
- after 24 hours the is no or little change into the error
- the variability on of the error between gaps (std of RMSE) follows a similar partner
- P and SWC are an exception from this pattern, with both the mean and std oft the error increasing for gaps longer than a day

Table 5: RMSE Comparison Kalman filter for different gap lengths

Variable	RMSE	1	3	6	12	24	48	72	168
TA	mean	0.289	0.371	0.519	0.651	1.105	1.037	1.055	1.096
SW_IN	mean	25.258	38.323	41.271	49.211	54.404	61.842	62.757	62.953
LW_IN	mean	6.076	8.091	12.706	16.450	15.547	17.341	16.377	16.828
VPD	mean	0.295	0.410	0.669	0.890	1.118	1.143	1.313	1.132
WS	mean	0.348	0.494	0.599	0.684	0.802	0.941	0.924	0.903
PA	mean	0.026	0.044	0.049	0.055	0.055	0.068	0.064	0.065
P	mean	0.110	0.084	0.135	0.121	0.149	0.169	0.174	0.180
SWC	mean	0.261	0.437	0.498	0.628	0.710	0.871	1.108	1.447
TS	mean	0.231	0.276	0.402	0.658	0.820	1.066	1.159	1.433

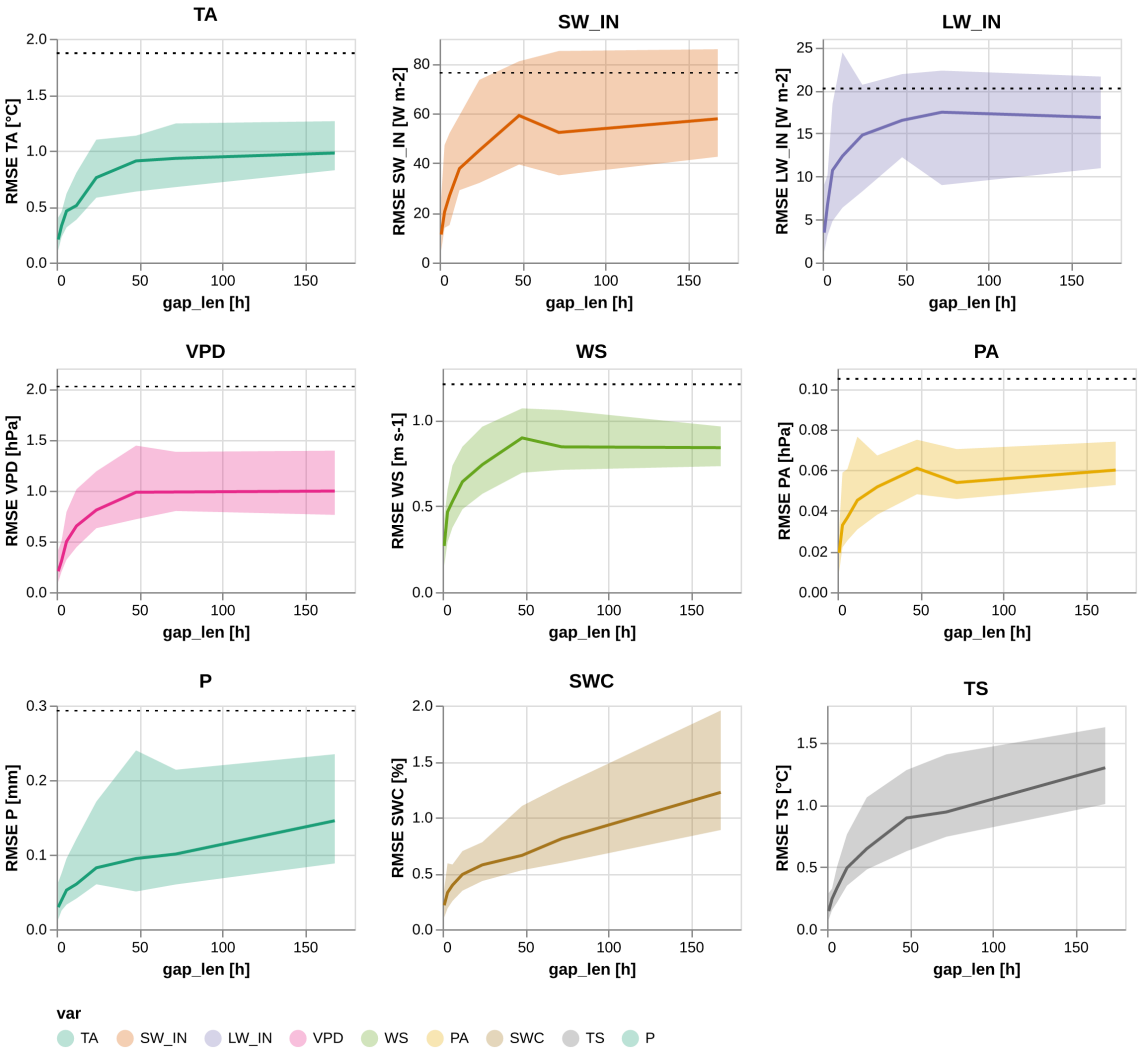

Figure 7: Gap len

3.3.2 Variable correlation

- the max gap length is 15 because after that the model crashed due to numerical stability issues
- compare a specialized model with gap in only one var and a generic model trained with gaps in all variables
- Overall for short gaps variable correlation doesn't help much and actually is worse
- for `TA` presence other variables helps to reduce error and variability of error
- for `SW_IN` worse with other variables for gap of 15 hours
- for `VPD` and `WS` other variable help reducing both mean and std for RMSE
- for `P` other variables increase error
- `SWC` the RMSE is higher when there are other variables in the gap
- for `TS` when there are other var err higher for short gaps but then comparable

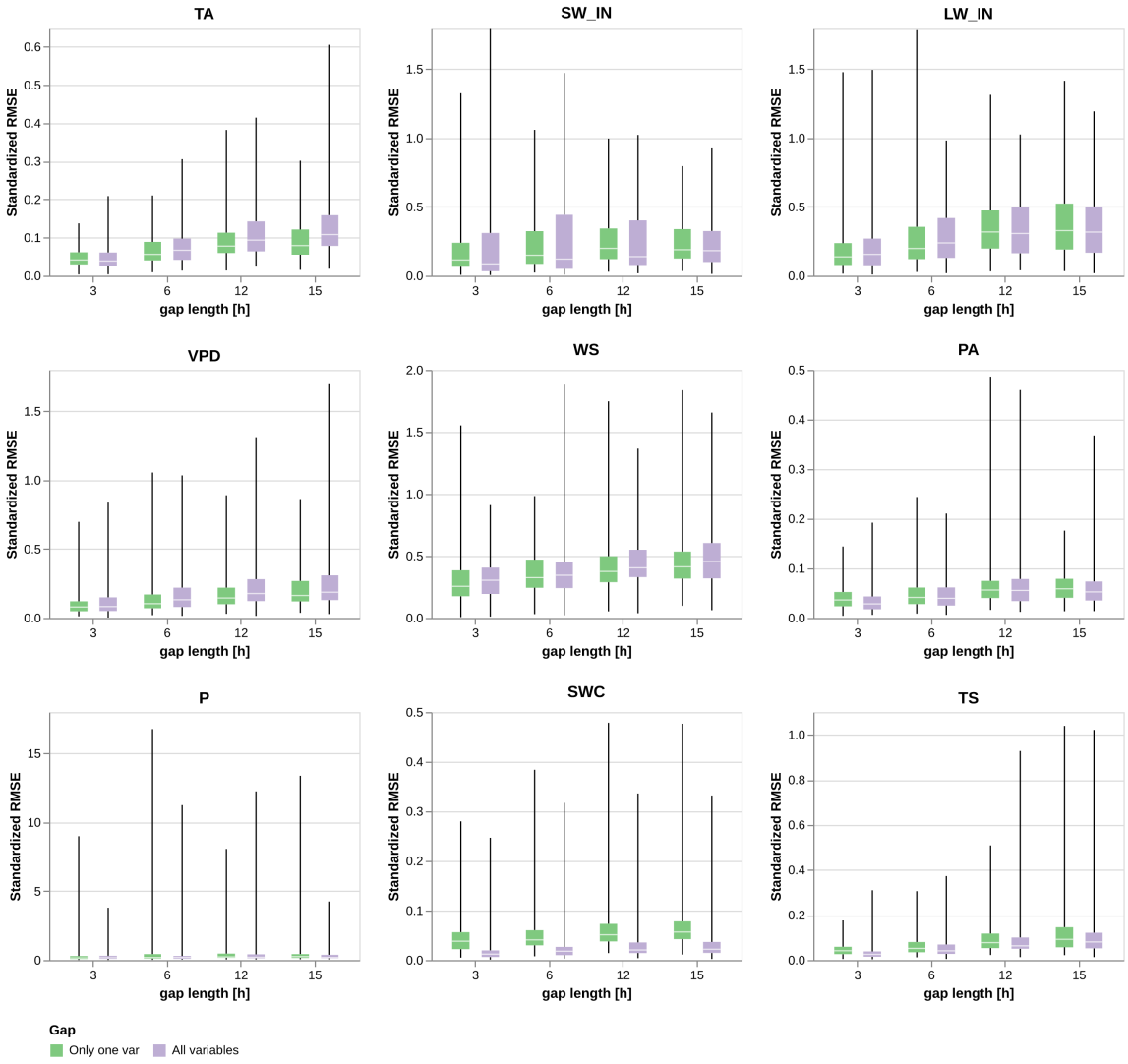

Figure 8: Gap single var

Variable	Gap RMSE Standardized Gap [h]	Only one var mean	Only one var std	All variables mean	All variables std	diff.
TA [C]	3	0.048	0.026	**0.047**	0.032	0.001
	6	**0.067**	0.037	0.080	0.054	0.013
	12	**0.090**	0.048	0.115	0.071	0.024
	15	**0.098**	0.057	0.130	0.082	0.033
SW_IN [W/m^2]	3	**0.205**	0.230	0.229	0.296	0.024
	6	**0.236**	0.215	0.282	0.314	0.047
	12	0.259	0.197	**0.259**	0.250	0.000
	15	0.259	0.180	**0.237**	0.191	0.022
LW_IN [W/m^2]	3	**0.201**	0.197	0.214	0.195	0.013
	6	**0.259**	0.202	0.283	0.187	0.025
	12	0.356	0.225	**0.353**	0.220	0.003
	15	0.365	0.232	**0.360**	0.218	0.005
VPD [hPa]	3	**0.103**	0.086	0.122	0.123	0.019
	6	**0.138**	0.123	0.178	0.156	0.041
	12	**0.175**	0.117	0.220	0.167	0.045
	15	**0.206**	0.132	0.256	0.207	0.050
WS [m/s]	3	**0.291**	0.178	0.317	0.160	0.026
	6	**0.372**	0.186	0.386	0.253	0.014
	12	**0.430**	0.241	0.464	0.220	0.034
	15	**0.453**	0.208	0.499	0.258	0.046
PA [hPa]	3	0.040	0.024	**0.034**	0.023	0.006
	6	0.051	0.034	**0.048**	0.032	0.003
	12	**0.064**	0.047	0.065	0.051	0.001
	15	0.063	0.030	**0.062**	0.040	0.002
P [mm]	3	0.369	0.803	**0.296**	0.408	0.073
	6	0.624	1.487	**0.379**	0.964	0.245
	12	**0.456**	0.759	0.517	1.119	0.061
	15	0.566	1.222	**0.431**	0.675	0.135
SWC [%]	3	0.044	0.034	**0.020**	0.032	0.024
	6	0.055	0.049	**0.027**	0.038	0.028
	12	0.066	0.053	**0.034**	0.042	0.032
	15	0.071	0.055	**0.038**	0.048	0.033
TS [C]	3	0.048	0.030	**0.033**	0.031	0.015
	6	0.068	0.051	**0.055**	0.047	0.013
	12	**0.099**	0.073	0.100	0.125	0.001
	15	0.121	0.113	**0.118**	0.131	0.003

3.3.3 Control variable

- generic model with control vs generic model without control
- for short gaps the use of the control is increasing the error
- for longer gaps the control is usually helping
- for `SWC` and `TA` using the control is slightly worse
- for `PA` the error without control is much higher around 5 times bigger for 1 week long gaps.
- for `P` control is helping a bit

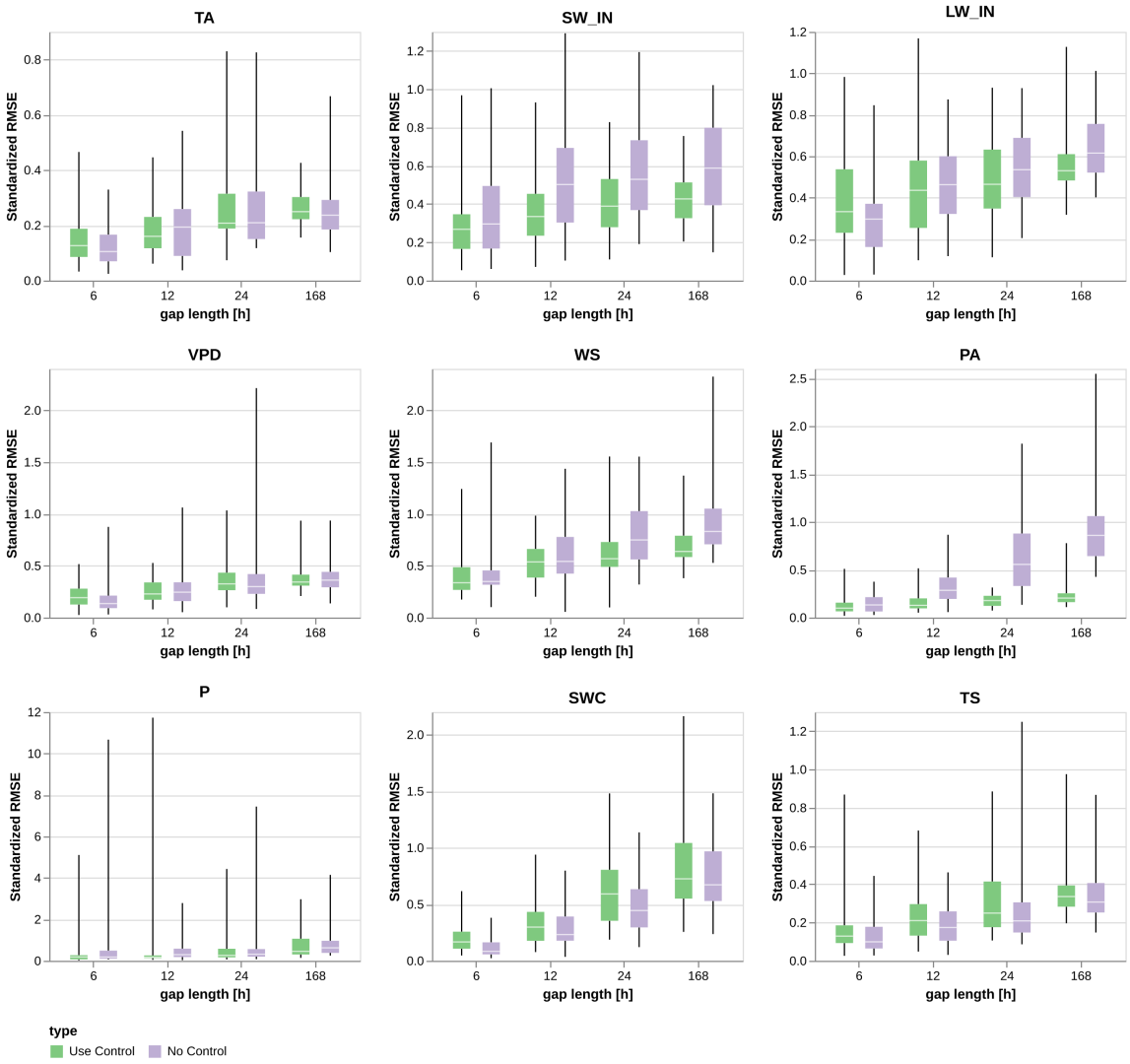

Figure 9: Comparison between generic model with control variable (green) and generic model without control variable (purple). 50 samples for each variable and each gap length.

Table 6: RMSE Comparison Kalman filter with and without control variable. The best result for each for each gap length is highligthed in bold

Variable	type RMSE Standardized Gap [h]	Use Control mean	Use Control std	No Control mean	No Control std	diff.
TA [C]	6	0.158	0.107	**0.118**	0.066	0.040
	12	**0.185**	0.097	0.194	0.110	0.008
	24	**0.254**	0.133	0.274	0.164	0.021
	168	0.267	0.065	**0.249**	0.104	0.018
SW_IN [W/m^2]	6	**0.289**	0.179	0.347	0.217	0.058
	12	**0.345**	0.177	0.524	0.285	0.179
	24	**0.408**	0.167	0.570	0.267	0.162
	168	**0.435**	0.124	0.597	0.232	0.161
LW_IN [W/m^2]	6	0.382	0.222	**0.300**	0.179	0.082
	12	**0.449**	0.228	0.465	0.183	0.016
	24	**0.487**	0.190	0.534	0.185	0.047
	168	**0.569**	0.141	0.643	0.147	0.074
VPD [hPa]	6	0.212	0.122	**0.202**	0.198	0.009
	12	**0.259**	0.126	0.270	0.172	0.011
	24	**0.373**	0.184	0.383	0.332	0.009
	168	**0.378**	0.127	0.396	0.155	0.018
WS [m/s]	6	**0.404**	0.210	0.425	0.255	0.021
	12	**0.544**	0.194	0.616	0.290	0.071
	24	**0.617**	0.230	0.817	0.328	0.200
	168	**0.695**	0.208	0.900	0.293	0.206
PA [hPa]	6	**0.123**	0.090	0.146	0.087	0.024
	12	**0.164**	0.105	0.333	0.206	0.170
	24	**0.179**	0.065	0.655	0.387	0.476
	168	**0.224**	0.110	0.928	0.422	0.704
P [mm]	6	**0.440**	0.888	0.635	1.561	0.195
	12	0.648	1.766	**0.543**	0.627	0.105
	24	**0.501**	0.687	0.767	1.462	0.266
	168	**0.793**	0.707	0.822	0.751	0.029
SWC [%]	6	0.201	0.127	**0.122**	0.090	0.078
	12	0.345	0.206	**0.286**	0.164	0.058
	24	0.627	0.312	**0.495**	0.252	0.131
	168	0.812	0.390	**0.741**	0.308	0.071
TS [C]	6	0.168	0.160	**0.132**	0.090	0.035
	12	0.230	0.139	**0.197**	0.116	0.033
	24	0.314	0.187	**0.264**	0.197	0.051
	168	0.358	0.136	**0.348**	0.166	0.010

3.3.4 Variable fine tuning

- fine-tuning the model is improving the performance across all variables
- biggest increase is in `SWC` and `TA`, which much smaller error especially for long gap. Variability of RMSE also decreases

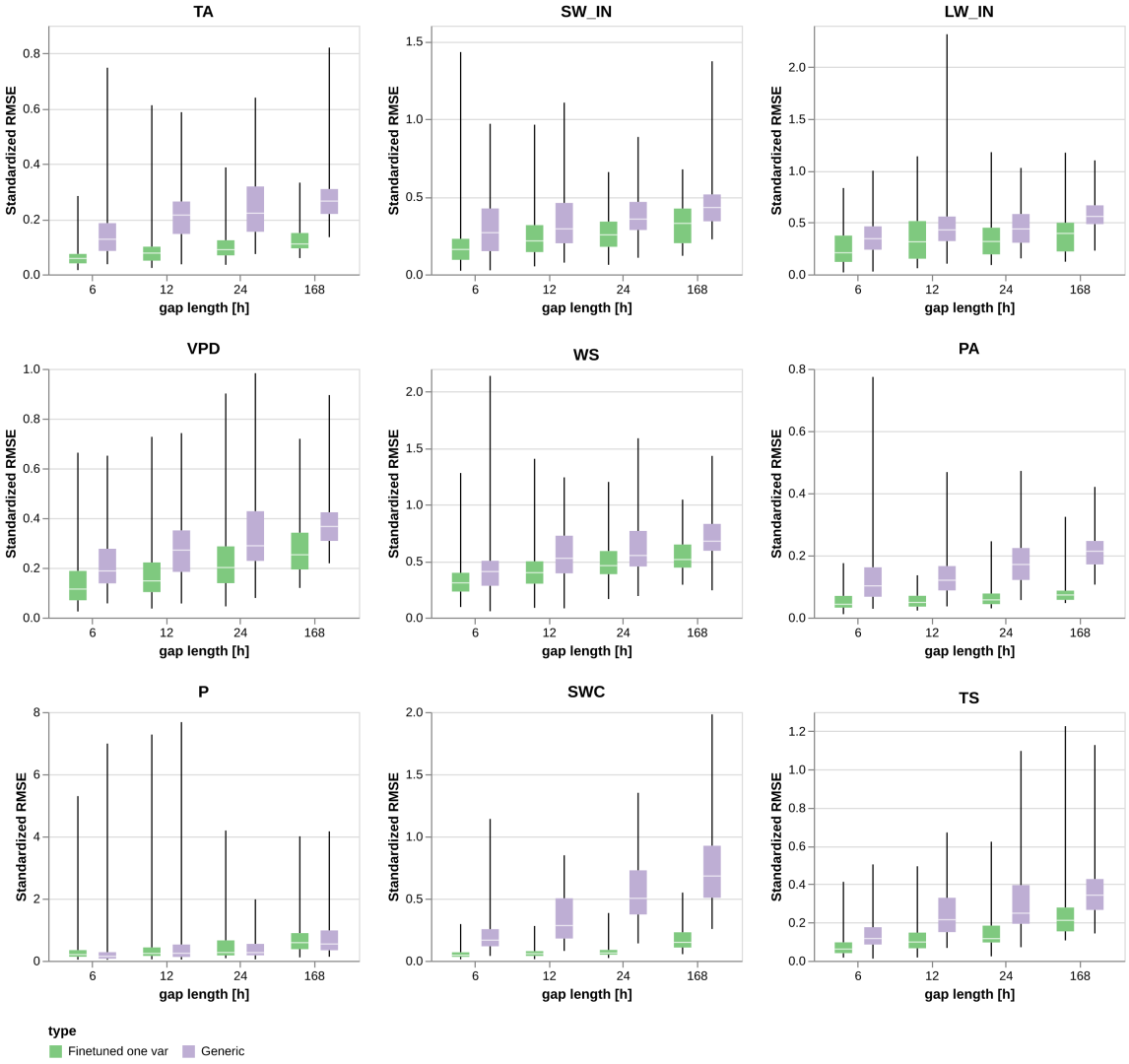

Figure 10: Comparison model fine-tuned for each variable (green) with generic model (purple). 100 samples for each variable and gap length

Variable	type RMSE Standardized Gap [h]	Generic mean	 std	Finetuned one var mean	 std	 diff.
TA [C]	6	0.158	0.118	**0.065**	0.041	-0.093
	12	0.225	0.113	**0.090**	0.072	-0.136
	24	0.241	0.112	**0.110**	0.068	-0.131
	168	0.277	0.103	**0.129**	0.052	-0.148
SW_IN [W/m^2]	6	0.315	0.229	**0.223**	0.235	-0.091
	12	0.381	0.242	**0.255**	0.159	-0.125
	24	0.386	0.158	**0.280**	0.140	-0.106
	168	0.454	0.167	**0.326**	0.123	-0.127
LW_IN [W/m^2]	6	0.371	0.191	**0.269**	0.192	-0.102
	12	0.488	0.292	**0.370**	0.258	-0.118
	24	0.465	0.190	**0.360**	0.216	-0.105
	168	0.575	0.141	**0.384**	0.181	-0.191
VPD [hPa]	6	0.222	0.124	**0.147**	0.112	-0.075
	12	0.291	0.138	**0.189**	0.138	-0.102
	24	0.340	0.172	**0.246**	0.177	-0.094
	168	0.381	0.117	**0.281**	0.125	-0.101
WS [m/s]	6	0.485	0.345	**0.340**	0.194	-0.145
	12	0.554	0.230	**0.430**	0.212	-0.124
	24	0.620	0.254	**0.489**	0.176	-0.131
	168	0.709	0.194	**0.548**	0.162	-0.161
PA [hPa]	6	0.135	0.112	**0.053**	0.031	-0.082
	12	0.145	0.084	**0.056**	0.027	-0.089
	24	0.183	0.089	**0.066**	0.035	-0.117
	168	0.213	0.055	**0.078**	0.034	-0.135
P [mm]	6	0.462	1.060	**0.456**	0.780	-0.006
	12	**0.500**	0.888	0.501	0.909	0.000
	24	**0.408**	0.357	0.531	0.637	0.123
	168	0.761	0.637	**0.753**	0.613	-0.008
SWC [%]	6	0.206	0.150	**0.054**	0.036	-0.152
	12	0.346	0.203	**0.064**	0.043	-0.282
	24	0.568	0.276	**0.078**	0.055	-0.490
	168	0.787	0.379	**0.178**	0.094	-0.609
TS [C]	6	0.140	0.088	**0.076**	0.054	-0.064
	12	0.245	0.135	**0.124**	0.096	-0.121
	24	0.307	0.168	**0.156**	0.109	-0.152
	168	0.374	0.166	**0.242**	0.141	-0.132

4 Discussion

[still missing]

4.1 Future steps

4.1.1 data

- train model for different sites and see difference between sites
- use ERA-5 Land data

4.1.2 Gap filling quality assessment

- can train with more realistic gaps properties (length/other variables missing/time of day)
- better analysis of gaps in fluxnet

4.1.3 Model improvement

- numerical stability
- make a model to predict the parameters of the Kalman Filter depending on the conditions
- use Neural Network to process control variable
- training and understand variables that are weird
- include uncertainty of observations in the model
- high variability ERA data error between sites

5 Conclusions

- Kalman Filter from preliminary results have the potential to improve imputation
- lower variability
- on all tested variables smaller or comparable error than other methods, but P
- provide interpretable uncertainty
- still work needs to be done for model developement and understand the filter performance in different conditions
- robust assessment of performance in real life conditions and test different sites

References

References

[1] *Associated ICOS Ecosystem Station Labelling Report - Hainich*. 2020. URL: https://data.icos-cp.eu/objects/_tFsWRgQcO7FkfvOqOOqIC8H (visited on 02/18/2023).

[2] Marc Aubinet, Timo Vesala, and Dario Papale, eds. *Eddy Covariance: A Practical Guide to Measurement and Data Analysis*. Dordrecht: Springer Netherlands, 2012. ISBN: 978-94-007-2350-4 978-94-007-2351-1. DOI: 10.1007/978-94-007-2351-1. URL: https://link.springer.com/10.1007/978-94-007-2351-1 (visited on 02/10/2023).

[3] M. Balzarolo et al. "Evaluating the Potential of Large-Scale Simulations to Predict Carbon Fluxes of Terrestrial Ecosystems over a European Eddy Covariance Network". In: *Biogeosciences* 11.10 (May 20, 2014), pp. 2661–2678. ISSN: 1726-4170. DOI: 10.5194/bg-11-2661-2014. URL: https://bg.copernicus.org/articles/11/2661/2014/ (visited on 02/10/2023).

[4] Gerald J. Bierman and Catherine L. Thornton. "Numerical Comparison of Kalman Filter Algorithms: Orbit Determination Case Study". In: *Automatica* 13.1 (Jan. 1, 1977), pp. 23–35. ISSN: 0005-1098. DOI: 10.1016/0005-1098(77)90006-1. URL: https://www.sciencedirect.com/science/article/pii/0005109877900061 (visited on 01/12/2023).

[5] Gordon B. Bonan et al. "Improving Canopy Processes in the Community Land Model Version 4 (CLM4) Using Global Flux Fields Empirically Inferred from FLUXNET Data". In: *Journal of Geophysical Research: Biogeosciences* 116.G2 (2011). ISSN: 2156-2202. DOI: 10.1029/2010JG001593. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2010JG001593 (visited on 02/10/2023).

[6] Stef van Buuren and Karin Groothuis-Oudshoorn. "Mice: Multivariate Imputation by Chained Equations in R". In: *Journal of Statistical Software* 45 (Dec. 12, 2011), pp. 1–67. ISSN: 1548-7660. DOI: 10.18637/jss.v045.i03. URL: https://doi.org/10.18637/jss.v045.i03 (visited on 02/18/2023).

[7] Wei Cao et al. "BRITS: Bidirectional Recurrent Imputation for Time Series". In: (), p. 11.

[8] NEAL A. CARLSON. "Fast Triangular Formulation of the Square Root Filter." In: *AIAA Journal* 11.9 (1973), pp. 1259–1265. ISSN: 0001-1452. DOI: 10.2514/3.6907. URL: https://doi.org/10.2514/3.6907 (visited on 02/15/2023).

[9] Rafaela Lisboa Costa et al. "Gap Filling and Quality Control Applied to Meteorological Variables Measured in the Northeast Region of Brazil". In: *Atmosphere* 12.10 (10 Oct. 2021), p. 1278. ISSN: 2073-4433. DOI: 10.3390/atmos12101278. URL: https://www.mdpi.com/2073-4433/12/10/1278 (visited on 05/14/2022).

[10] Dan Simon. *Optimal State Estimation Kalman, H and Nonlinear Approaches*. 2006. ISBN: 100-47 1-70858-5.

[11] Wenjie Du, David Cote, and Yan Liu. *SAITS: Self-Attention-based Imputation for Time Series*. May 9, 2022. arXiv: 2202.08516 [cs]. URL: http://arxiv.org/abs/2202.08516 (visited on 05/14/2022).

[12] James Durbin and Siem Jan Koopman. *Time Series Analysis by State Space Methods*. Mar. 2, 2012. URL: https://doi.org/10.1093/acprof:oso/9780199641178.001.0001.

[13] Chenguang Fang and Chen Wang. "Time Series Data Imputation: A Survey on Deep Learning Approaches". Nov. 23, 2020. arXiv: 2011.11347 [cs]. URL: http://arxiv.org/abs/2011.11347 (visited on 05/12/2022).

[14] Vincent Fortuin et al. *GP-VAE: Deep Probabilistic Time Series Imputation*. Feb. 20, 2020. DOI: 10.48550/arXiv.1907.04155. arXiv: 1907.04155 [cs, stat]. URL: http://arxiv.org/abs/1907.04155 (visited on 07/28/2022).

[15] Andrew D. Friend et al. "FLUXNET and Modelling the Global Carbon Cycle". In: *Global Change Biology* 13.3 (2007), pp. 610–633. ISSN: 1365-2486. DOI: 10.1111/j.1365-2486.2006.01223.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2486.2006.01223.x (visited on 02/10/2023).

[16] Peter Isaac et al. "OzFlux Data: Network Integration from Collection to Curation". In: *Biogeosciences* 14.12 (June 19, 2017), pp. 2903–2928. ISSN: 1726-4170. DOI: 10.5194/bg-14-2903-2017. URL: https://bg.copernicus.org/articles/14/2903/2017/ (visited on 05/12/2022).

[17] Xin Jing et al. "A Multi-imputation Method to Deal With Hydro-Meteorological Missing Values by Integrating Chain Equations and Random Forest". In: *Water Resources Management* 36.4 (Mar. 1, 2022), pp. 1159–1173. ISSN: 1573-1650. DOI: 10.1007/s11269-021-03037-5. URL: https://doi.org/10.1007/s11269-021-03037-5 (visited on 02/18/2023).

[18] P. Kaminski, A. Bryson, and S. Schmidt. "Discrete Square Root Filtering: A Survey of Current Techniques". In: *IEEE Transactions on Automatic Control* 16.6 (Dec. 1971), pp. 727–736. ISSN: 1558-2523. DOI: 10.1109/TAC.1971.1099816.

[19] K. Kramer et al. "Evaluation of Six Process-Based Forest Growth Models Using Eddy-Covariance Measurements of CO2 and H2O Fluxes at Six Forest Sites in Europe". In: *Global Change Biology* 8.3 (2002), pp. 213–230. ISSN: 1365-2486. DOI: 10.1046/j.1365-2486.2002.00471.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-2486.2002.00471.x (visited on 01/18/2023).

[20] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB, Second Edition*. 2001. ISBN: 0-471-26638-8.

[21] Dario Papale. "Ideas and Perspectives: Enhancing the Impact of the FLUXNET Network of Eddy Covariance Sites". In: *Biogeosciences* 17.22 (Nov. 17, 2020), pp. 5587–5598. ISSN: 1726-4189. DOI: 10.5194/bg-17-5587-2020. URL: https://bg.copernicus.org/articles/17/5587/2020/ (visited on 01/18/2023).

[22] Gilberto Pastorello et al. "The FLUXNET2015 Dataset and the ONEFlux Processing Pipeline for Eddy Covariance Data". In: *Scientific Data* 7.1 (1 July 9, 2020), p. 225. ISSN: 2052-4463. DOI: 10.1038/s41597-020-0534-3. URL: https://www.nature.com/articles/s41597-020-0534-3 (visited on 05/12/2022).

[23] JAMES POTTER and ROBERT STERN. "STATISTICAL FILTERING OF SPACE NAVIGATION MEASUREMENTS". In: *Guidance and Control Conference*. American Institute of Aeronautics and Astronautics, 1963. DOI: 10.2514/6.1963-333. URL: https://arc.aiaa.org/doi/abs/10.2514/6.1963-333 (visited on 01/13/2023).

[24] H. E. RAUCH, F. TUNG, and C. T. STRIEBEL. "Maximum Likelihood Estimates of Linear Dynamic Systems". In: *AIAA Journal* 3.8 (1965), pp. 1445–1450. ISSN: 0001-1452. DOI: 10.2514/3.3166. URL: https://doi.org/10.2514/3.3166 (visited on 02/19/2023).

[25] Markus Reichstein et al. "On the Separation of Net Ecosystem Exchange into Assimilation and Ecosystem Respiration: Review and Improved Algorithm". In: *Global Change Biology* 11.9 (2005), pp. 1424–1439. ISSN: 1365-2486. DOI: 10.1111/j.1365-2486.2005.001002.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2486.2005.001002.x (visited on 05/12/2022).

[26] Mark G. Rutten. "Square-Root Unscented Filtering and Smoothing". In: *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing. Apr. 2013, pp. 294–299. DOI: 10.1109/ISSNIP.2013.6529805.

[27] *Specification - Vaisala HMP3 General Purpose Humidity and Temperature Probe*. URL: https://docs.vaisala.com/v/u/B211826EN-C/en-US (visited on 02/18/2023).

[28] *Statsmodels.Tsa.Statespace.Kalman_filter.KalmanFilter — Statsmodels*. URL: https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.kalman_filter.KalmanFilter.html#statsmodels.tsa.statespace.kalman_filter.KalmanFilter (visited on 02/19/2023).

[29] N. Vuichard and D. Papale. "Filling the Gaps in Meteorological Continuous Data Measured at FLUXNET Sites with ERA-Interim Reanalysis". In: *Earth System Science Data* 7.2 (July 13, 2015), pp. 157–171. ISSN: 1866-3508. DOI: 10.5194/essd-7-157-2015. URL: https://essd.copernicus.org/articles/7/157/2015/ (visited on 05/11/2022).

[30] Thomas Wutzler et al. "Basic and Extensible Post-Processing of Eddy Covariance Flux Data with REddyProc". In: *Biogeosciences* 15.16 (Aug. 23, 2018), pp. 5015–5030. ISSN: 1726-4170. DOI: 10.5194/bg-15-5015-2018. URL: https://bg.copernicus.org/articles/15/5015/2018/ (visited on 05/15/2022).

[31] Byron M Yu et al. "Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity". In: *Advances in Neural Information Processing Systems*. Vol. 21. Curran Associates, Inc., 2008. URL: https://proceedings.neurips.cc/paper/2008/hash/ad972f10e0800b49d76fed33a21f6698-Abstract.html (visited on 07/07/2022).

[32] Yifan Zhang and Peter J. Thorburn. "A Dual-Head Attention Model for Time Series Data Imputation". In: *Computers and Electronics in Agriculture* 189 (Oct. 1, 2021), p. 106377. ISSN: 0168-1699. DOI: 10.1016/j.compag.2021.106377. URL: https://www.sciencedirect.com/science/article/pii/S016816992100394X (visited on 06/15/2022).

[33] Y. Zhao et al. "How Errors on Meteorological Variables Impact Simulated Ecosystem Fluxes: A Case Study for Six French Sites". In: *Biogeosciences* 9.7 (July 11, 2012), pp. 2537–2564. ISSN: 1726-4170. DOI: 10.5194/bg-9-2537-2012. URL: https://bg.copernicus.org/articles/9/2537/2012/ (visited on 02/13/2023).

A Additional Results

The entire FLUXNET 2015 dataset was used to compute the distribution of gap lengths across the all the sites for each variable. A gap was definite when the QC flag of the variable is different from 0 or the data itself is missing. Figure 11 shows the complete distribution of the gaps, while figure 12 focuses only on gaps shorter than a week.

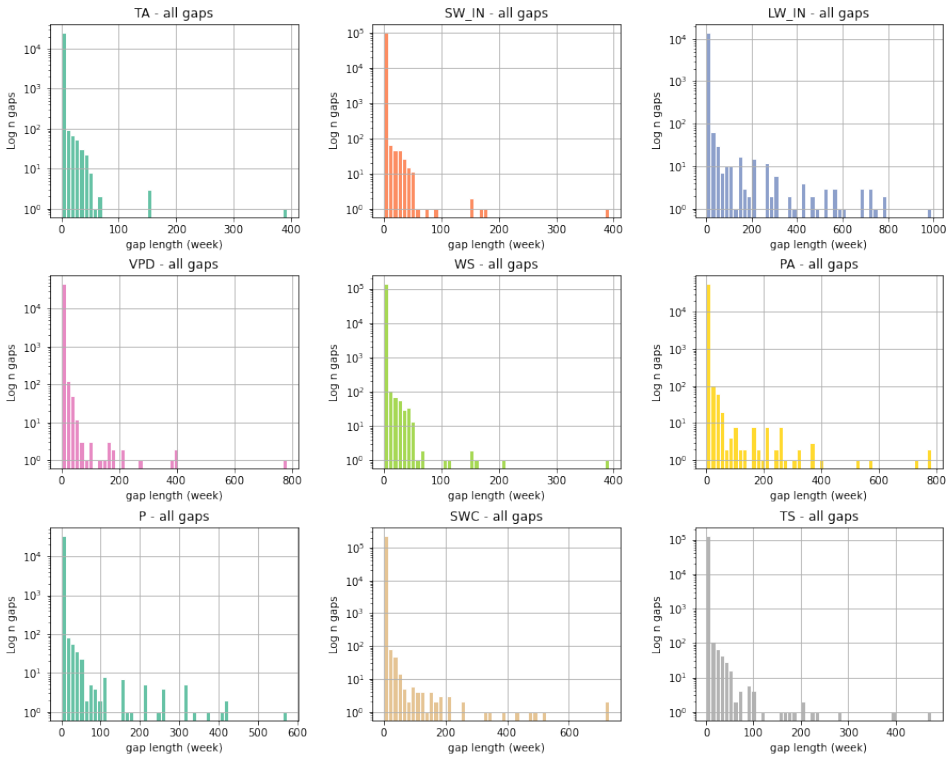

Figure 11: FLUXNET gap len

A.1 Gap length distribution in FLUXNET

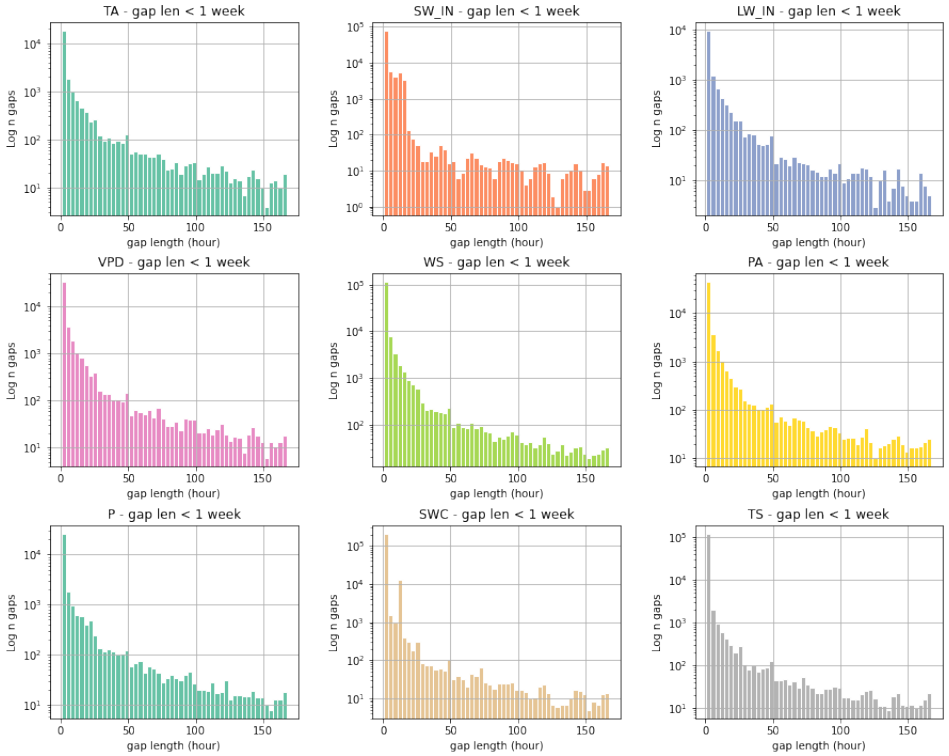

Figure 12: FLUXNET gap len less than a week

A.2 Additional Timeseries

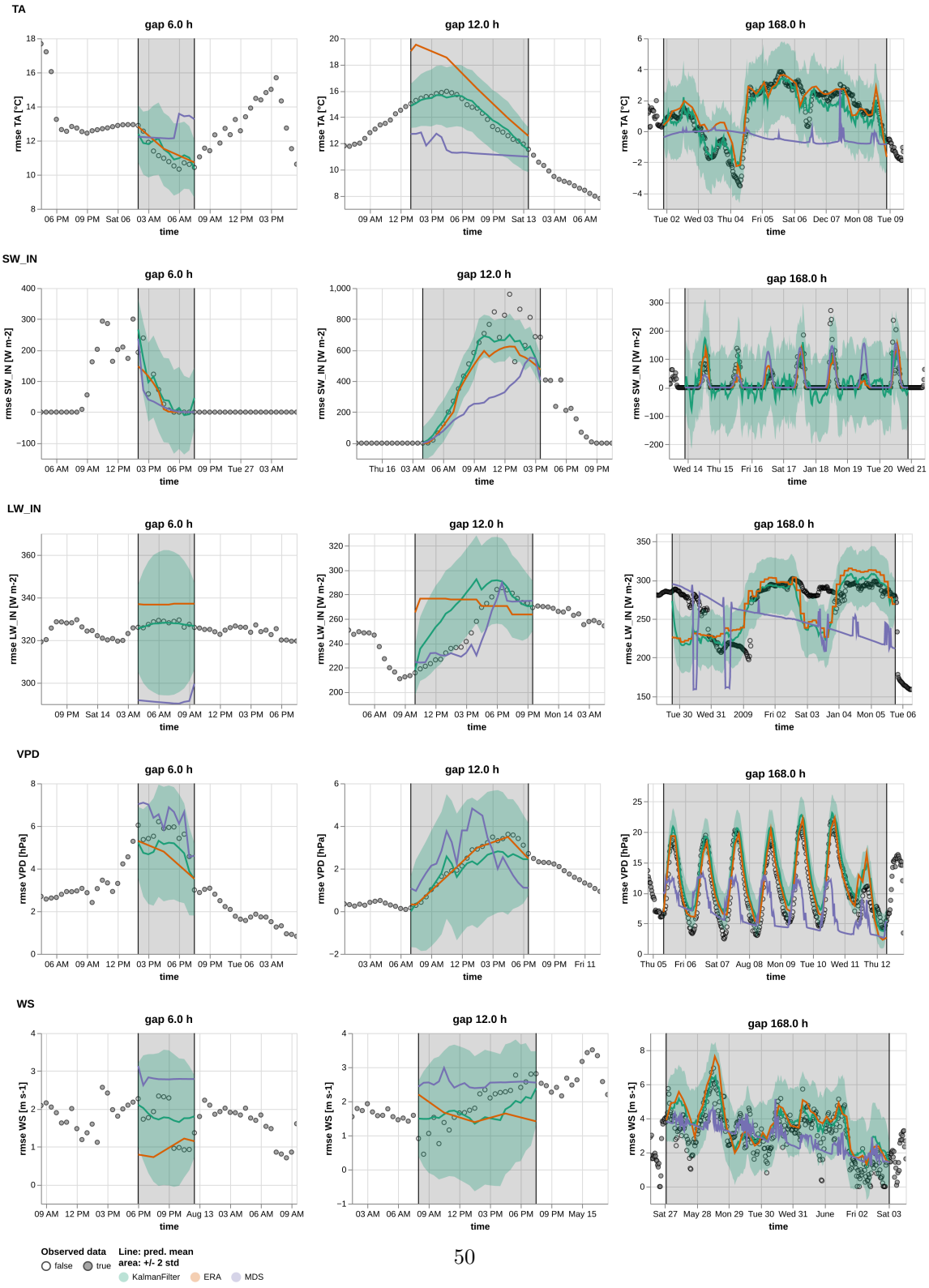

Figure 13: Timeseries 1

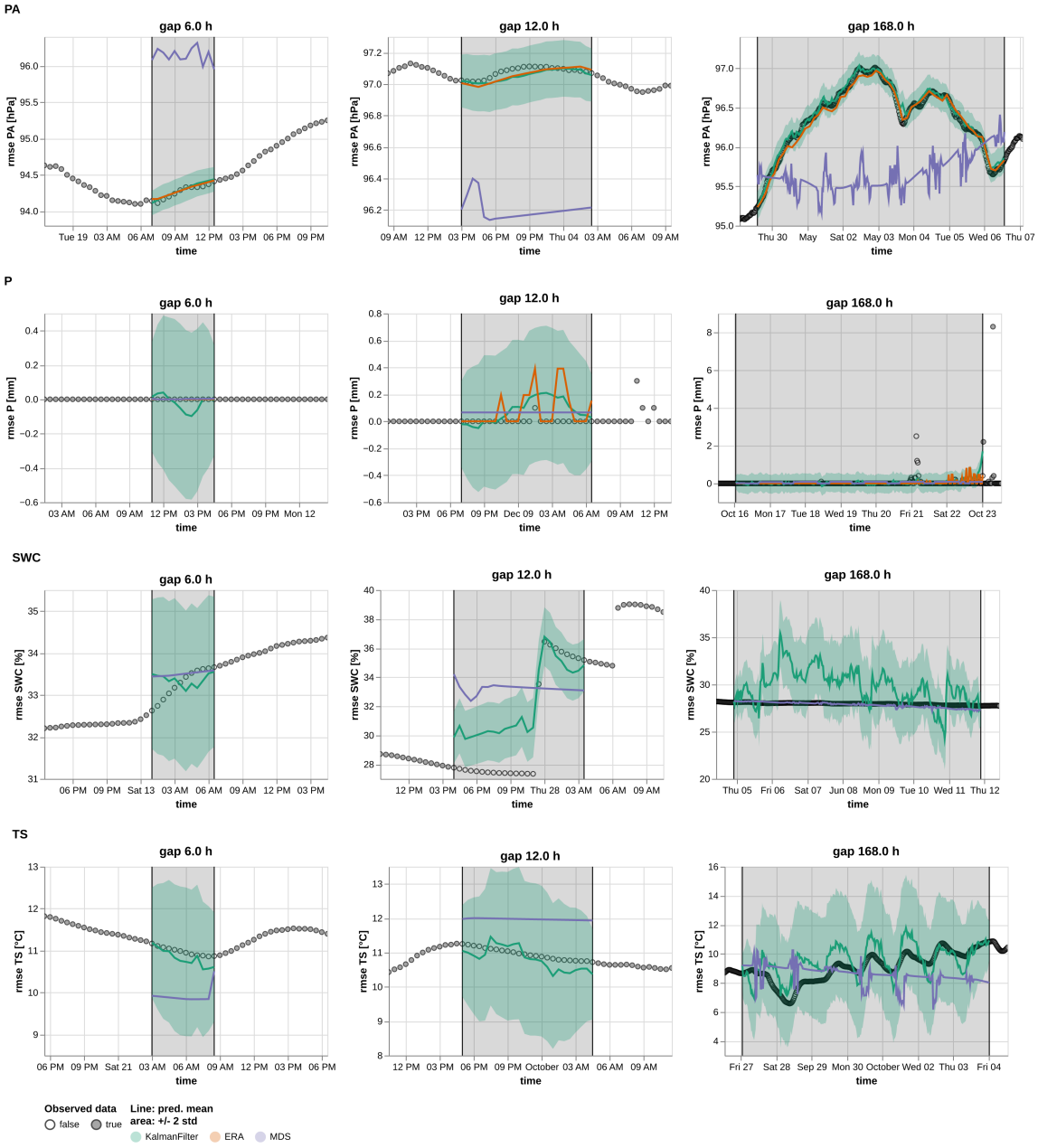

Figure 14: Timeseries 1

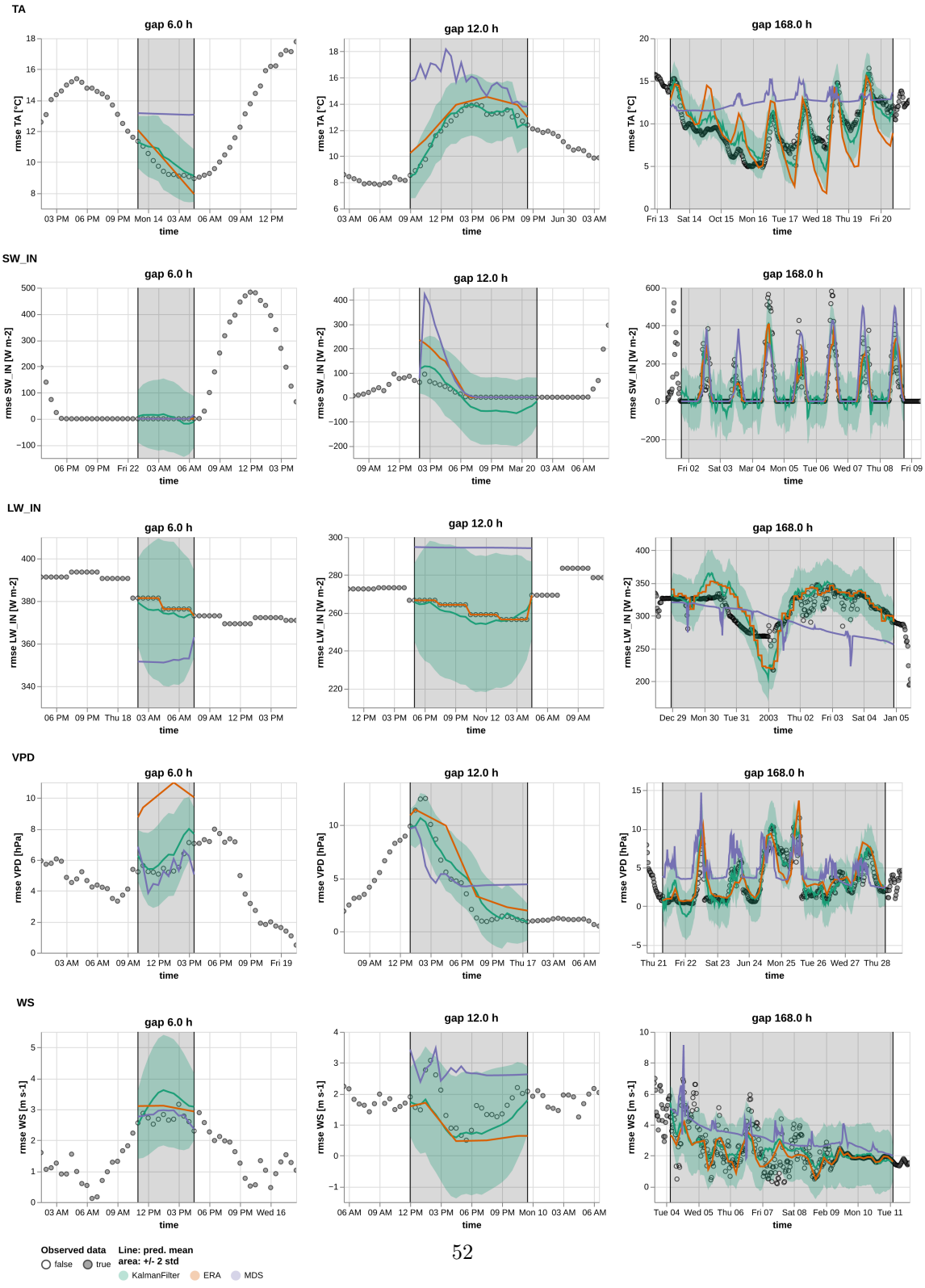

Figure 15: Timeseries 1

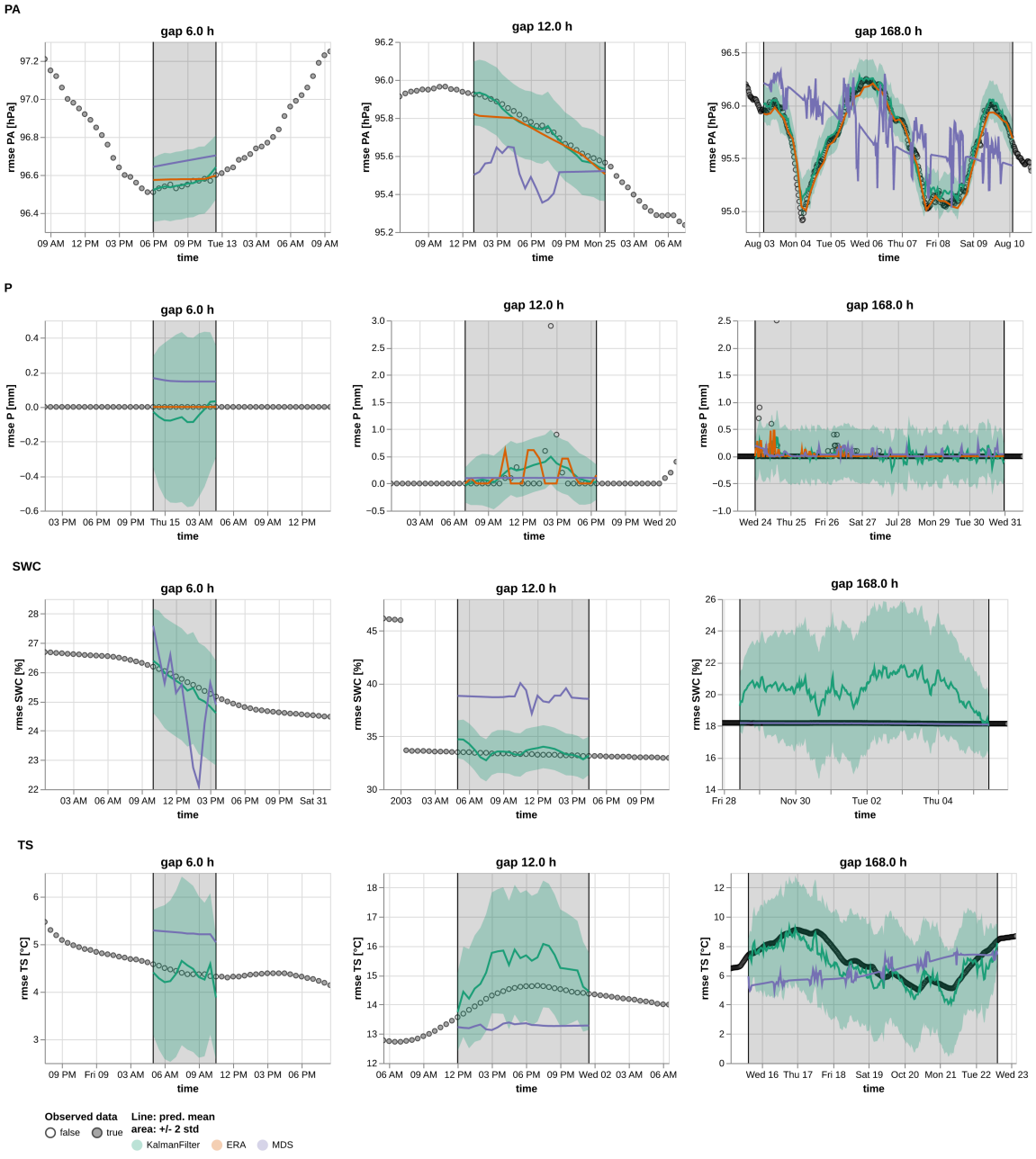

Figure 16: Timeseries 1

B Comparison between Standard and Square Root Kalman Filter

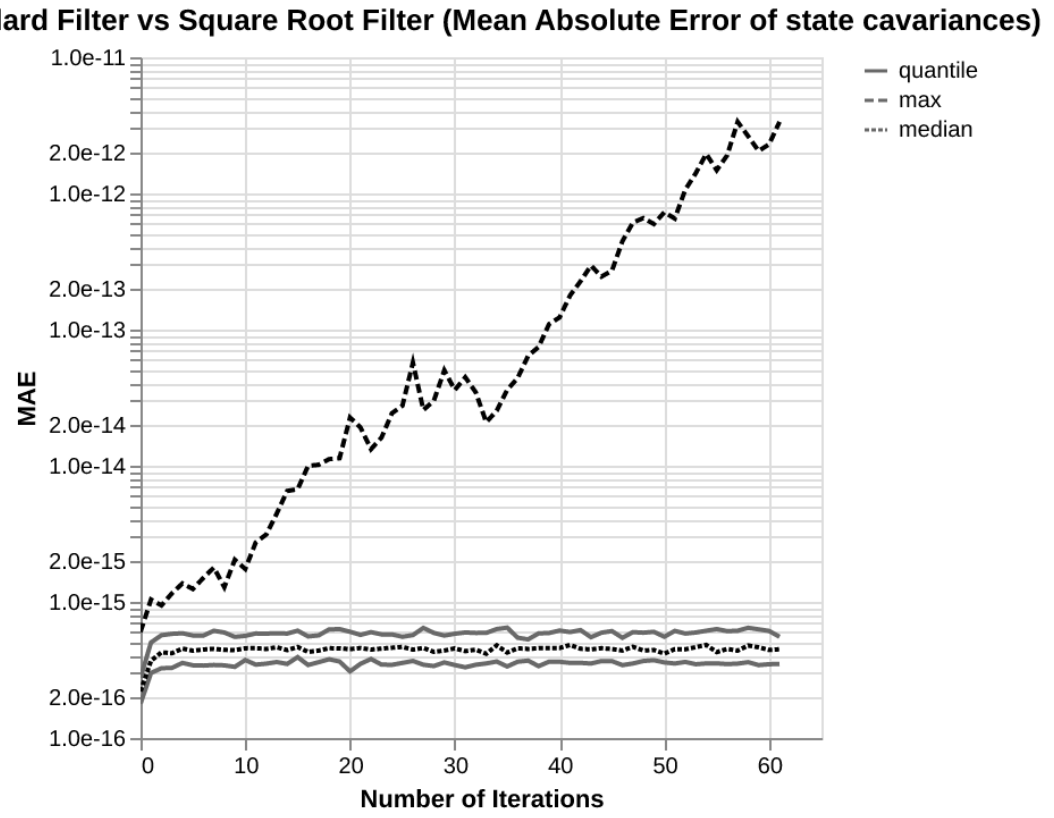

Figure 17: Numerical stability comparison between standard Kalman Filter implementation and Square Root Filter. For 100 times the filter has been initialized with random parameters (drawn from a uniform distribution range 0-1) and then filtered 100 observations. At each filter iteration the Mean Absolute Error (MAE) was calculated between the state covariance from the standard filter and the square root filter. The plot shows the median, 1 and 3 quartile and the maximum of the MAE across the 100 samples.

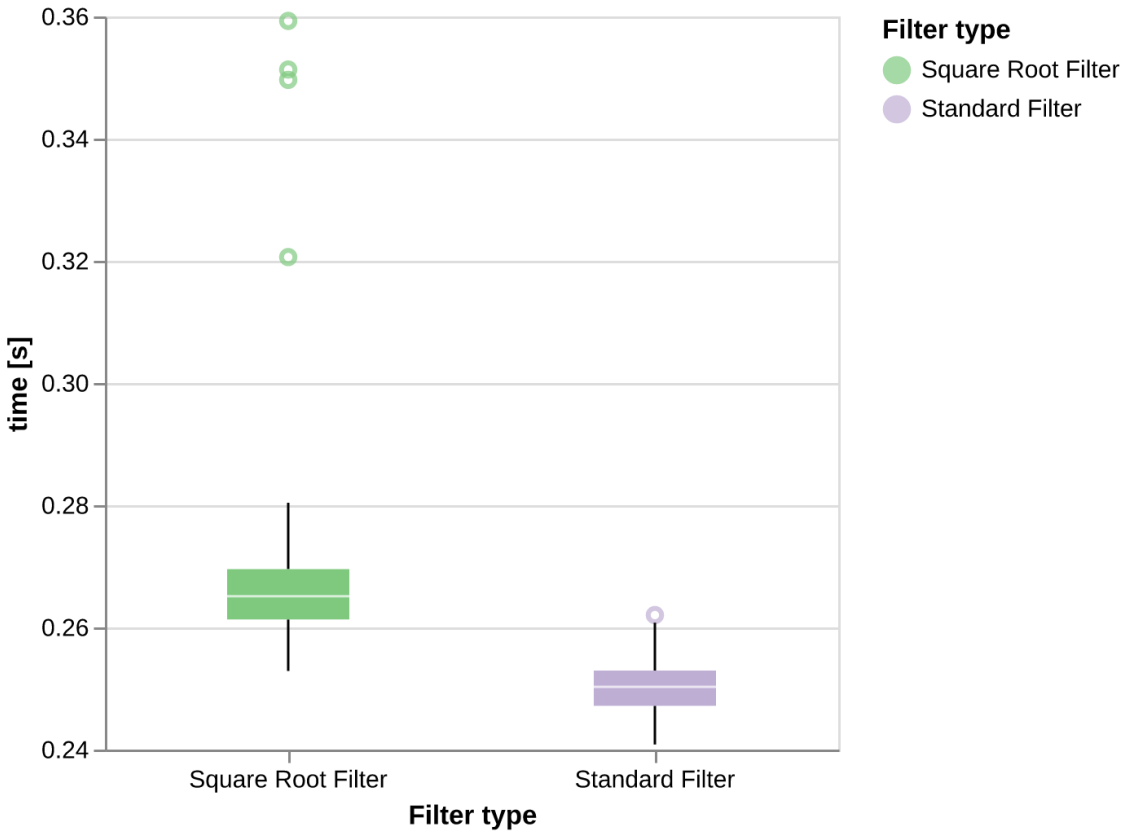

Figure 18: Performance comparison between standard Kalman Filter implementation and Square Root Filter. 100 samples with the following settings: Number of observations: 100, dimension observations 4, dimension state: 3, dimension control: 3, batch size: 5. Data and parameters are randomly generated.