Wing shape estimation with Extended Kalman filtering and KalmanNet neural network of a flexible wing aircraft

Bence Zs. Hadlaczky HADLACZKY.BENCE.ZSOMBOR@SZTAKI.HU
Noémi Friedman FRIEDMAN.NOEMI@SZTAKI.HU
Béla Takarics TAKARICS.BELA@SZTAKI.HU
Bálint Vanek VANEK@SZTAKI.HU
Institute for Computer Science and Control, Eötvös Lóránd Research Network
Kende u. 13-17. 1111, Budapest, Hungary

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

The dynamic behaviour, stability, and the effects of the aerodynamic drag of a large-wingspan aircraft are mainly influenced by the structural flexibility and shape of the wings during flight. Therefore, utilizing a wing shape controller that minimizes the effects of drag can greatly improve the behaviour and fuel consumption of the aircraft. However, such a controller requires the measurement of the dynamics of the wing, more precisely, the modal coordinates which describe the structural and dynamic changes of the wing. For estimating the modal coordinates and reconstructing the wing shape a state observer is necessary because the direct and accurate measurement of these states is not feasible. It is demonstrated in this paper that machine learning-based methods can approach the accuracy of traditional model-based Kalman filtering in wing shape estimation. First, the model-based method Extended Kalman Filtering (EKF) is presented, using a Linear Parameter Varying (LPV) system model. Second, we present a machine learning-based approach based on the new KalmanNet architecture with two different recurrent neural network configurations: one with linear layers and one with one-dimensional convolutional layers. The results are evaluated on the T-Flex aerial demonstrator aircraft and compared using the LPV-based EKF as a reference. It is shown that the learning-based approach provides comparable results to the model-based method while using fewer design parameters.

Keywords: aeroelasticity, structural dynamics, Kalman filtering, recurrent neural network, convolutional neural network, KalmanNet

1. Introduction

In recent years, research and development trends in the aerospace industry placed more emphasis on increasing fuel efficiency since a significant portion of the operating costs of an aircraft comes from fuel consumption. To achieve this objective, using more flexible components, such as highly flexible wings, is a widely investigated solution (Wüstenhagen et al. (2018)). The fuel consumption of the aircraft mainly depends on the aerodynamic drag acting on the wings hence reducing drag with a suitable wing-shape controller results in better fuel economy. However, such a controller requires information about the flexible dynamics of the wing that cannot be measured directly. As a result, a state observer/estimator is required. This paper presents results from the ongoing FliPASED (2019) project. FliPASED aims to extend the control capabilities of the T-Flex aerial demonstrator aircraft by developing drag-reducing control. This motivates the design of a wing shape estimation

algorithm, which is necessary for a drag reduction control system. For this purpose, the present paper provides two different approaches for estimating the states describing the flexible dynamics of the T-Flex.

The well-established solution for designing a state estimator is the extended Kalman filter (EKF) for nonlinear systems, which can be used for inertial estimation of the wing shape (Lustosa et al. (2021)) as well. However, the EKF has two main drawbacks. First, it requires the explicit nonlinear state-space model of the system, which might not be available, or its use is computationally too expensive. The second drawback is that the EKF requires models of the measurement and process noises in terms of their covariance matrices. As a solution to the first issue, the approximation of the full, nonlinear system with a Linear Parameter Varying (LPV) model (Takarics and Vanek (2021)) can be considered, as it can significantly ease the computational burden while being suitable for use with an EKF (Shereen et al. (2016)). However, models of the noise processes are still required.

Machine learning-based approaches have the great advantage that they can be used for inertial odometry (Dugne-Hennequin et al. (2021)), inertial aided navigation (Zhang et al. (2021)), or to obtain state-estimation trajectories with moving horizon estimation when there are unknown parameters present in the system (Muntwiler et al. (2022)). To utilize this advantage, first, we created a 'black box' type neural network architecture to provide state estimates directly. However, we found that this approach could not solve the state estimation problem in our case. As a result, the new KalmanNet architecture (Revach et al. (2022)) was chosen as it retains engineering insight into the physical system. This approach is based on Kalman filtering but uses a Recurrent Neural Network (RNN) to estimate the Kalman gain. As a result, explicit models of the noise processes are not required. The standard KalmanNet architecture uses linear layers and a Gated Recurrent Unit (GRU) to establish correlations between data samples in time. The novelties of this research are the following. First, we apply the KalmanNet architecture to a high-order LPV model of a complex real-life system. This requires a new loss calculation to ensure the stability of the state estimator. There is also a need for the modification of the layer dimensions in order to decrease the computational burden and the implementation of a hyperparameter optimizing algorithm. Secondly, we propose a different neural network architecture for the KalmanNet, which uses one-dimensional (1D) convolutional layers alongside the GRU layer, since 1D convolution can be used for processing time series data (Silva do Monte Lima et al. (2019)) effectively.

This paper presents the LPV-based EKF and the KalmanNet for predicting the modal coordinates and aerodynamic lag states of the nonlinear model of the T-Flex (Wüstenhagen et al. (2018)). The paper is organized as follows. In Section 2, the dynamic model of the T-Flex is presented. Section 3 introduces the LPV model of the original nonlinear system and explains the idea of the LPV-based EKF. In Section 4, the basic structure of the KalmanNet is summarized. Two different recurrent neural network architectures are introduced. The first uses linear layers alongside the GRU cell, while the second uses convolutional layers instead. Section 5 presents the results of the modal coordinate and lag state estimations. The accuracy of the LPV-based EKF and the KalmanNet is compared. Conclusions are drawn in Section 6.

2. T-Flex dynamic model

2.1. Nonlinear model

The model of the T-Flex aircraft is given as a nonlinear state-space system. Details of the modelling and model order reduction are given in Wüstenhagen et al. (2018), Takarics et al. (2018)

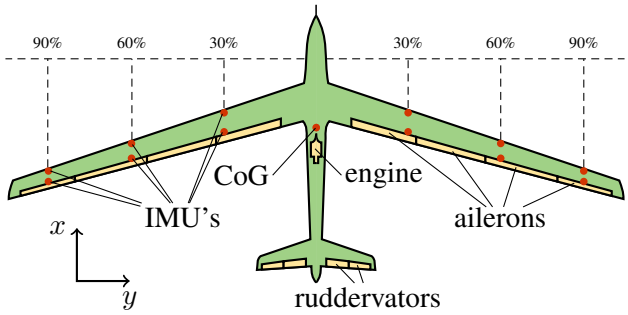

Figure 1: Demonstrator control surfaces and IMU locations

and Meddaikar et al. (2019). The system has 48 states. The state vector $x \in \mathbb{R}^{48}$ consists of rigid body states, states related to the flexible dynamics of the wing and aerodynamics (denoted as x_{flex}), and finally, states that represent the control surface inputs and their first derivatives. The rigid body motion is represented with a 6-DOF model with 12 states: states of translational and angular velocities, position [1], and orientation. The flexible dynamics of the wing are described with $w(x,t) = \sum_{i=1}^{\infty} \Phi_i(x) U_{fi}(t) \approx \sum_{i=1}^{6} \Phi_i(x) U_{fi}(t)$, where $\Phi_i(x)$ denotes the i^{th} mode shape and $U_{fi}(t)$ the i^{th} modal coordinate. So $x_{\text{flex}} \in \mathbb{R}^{14}$ contains the first six modal coordinates, their first derivatives and two lag states: $x_{\text{flex}} = [U_{f1}\, U_{f2}\, U_{f3}\, U_{f4}\, U_{f5}\, U_{f6}\, \dot{U}_{f1}\, \dot{U}_{f2}\, \dot{U}_{f3}\, \dot{U}_{f4}\, \dot{U}_{f5}\, \dot{U}_{f6}\, \text{lag}_1\, \text{lag}_2]^T$. The reason for using only the first six modal coordinates in the model is that these have the largest contribution in describing the flexible behaviour of the wings Meddaikar et al. (2019). The objective of this research is the estimation of the x_{flex}.

The sensors and control surfaces of the system are depicted in Figure 1. The system has 13 inputs, $u \in \mathbb{R}^{13}$, which contains one turbofan engine input (Throttle) and 12 control surface inputs: four-four ailerons (AileronR/L) on each wing and on the V-tail two-two 'ruddervators' (TailR/L).

The output vector $y \in \mathbb{R}^{64}$ of the system has 23 rigid body-related sensors, which provide information about the position (X_E, Y_E, Z_E), orientation (ϕ, θ, ψ), translational (v_N, v_E, v_D) and angular velocity (p, q, r), and acceleration (a_{xB}, a_{yB}, a_{zB}) of the aircraft. Furthermore, the angle of attack (α), sideslip angle (β), air (p_a) and total pressure (p_T), barometric altitude (h_{baro}), indicated (v_{IAS}) and the true airspeed (v_{TAS}) are measured as well. These sensors are located in the center of gravity (CoG) of the fuselage. Each wing of the demonstrator has six-six additional inertial measurement units (IMUs). The IMUs of the leading-edge measure accelerations in the x, y, and z directions, while the IMUs of the trailing edge provide angular velocity data around the x- and y-axis, and acceleration data in the z direction. The exact location of the IMUs can be seen in Figure 1 as well. In addition, the wingtip coordinates are measured with a mono camera preventing acceleration-based estimation errors from diverging in time (Lustosa et al. (2021)). The coordinates of four wingtip points are measured in each direction.

1. For creating the final state-space model, the states of the x and y positions were truncated because, under certain trim conditions, these can cause unstable poles to appear.

2.2. LPV model

In our work, we created a Linear Parameter Varying (LPV) approximation of the nonlinear model of the T-Flex (Takarics and Vanek (2021)). It is essentially a point-wise linearization of the nonlinear state space system: linearized at different trim points defined by the scheduling parameters. The scheduling parameters create a multidimensional grid, and a linear, state-space model is assigned to every grid point. As scheduling parameters ($\rho \in \mathbb{R}^2$), the true airspeed (v_{TAS}) and the roll angle (ϕ) outputs were chosen. The grid for the LPV model consists of airspeed values from 35 $\mathrm{m/s}$ to 55 $\mathrm{m/s}$ with a 0.1 $\mathrm{m/s}$ resolution and the roll angles from $0°$ to $45°$ with $1°$ resolution. The state-space equations of the discrete-time LPV system are

$$\begin{aligned} x[k] &= A(\rho[k])x[k-1] + B(\rho[k])u[k], \\ y[k] &= C(\rho[k])x[k] + D(\rho[k])u[k], \end{aligned} \tag{1}$$

where $\rho[k]$ is the time-varying vector of the scheduling parameters at time step k. $A(\rho[k]) \in \mathbb{R}^{48\times48}$, $B(\rho[k]) \in \mathbb{R}^{48\times13}$, $C(\rho[k]) \in \mathbb{R}^{64\times48}$ and $D(\rho[k]) \in \mathbb{R}^{64\times13}$ denotes the parameter-dependent state-space matrices of the LPV system. The state vector is denoted as $x[k]$, the input vector as $u[k]$, while $y[k]$ is the output vector of the system at time step k.

3. Model-based wing shape estimation

Extended Kalman Filtering (EKF) is used as a model-based wing shape estimation approach. The EKF pipeline requires the full, nonlinear state-space description of the system and information about the model noise and observation noise in the form of noise covariance matrices. The discrete-time nonlinear state-space system is of the form:

$$\begin{aligned} x[k] &= \mathrm{f}(x[k-1], u[k]) + w[k], \\ y[k] &= \mathrm{h}(x[k], u[k]) + v[k]. \end{aligned} \tag{2}$$

Here, the nonlinear function $\mathrm{f}(.)$ is called state-transition function, while $\mathrm{h}(.)$ is called state-observation function. The $w[k] \in \mathbb{R}^{48}$ and $v[k] \in \mathbb{R}^{64}$ vectors are the model noise and observation noise vectors, which are described by their covariance matrices $Q \in \mathbb{R}^{48\times48}$ and $R \in \mathbb{R}^{64\times64}$ respectively. Both noises have 0 mean, normal distributions, and the noise vectors at each time step are mutually independent.

The general framework of the EKF consists of two main steps: *prediction* and *update*. In these steps, point-wise linearization is used to approximate the behaviour of the nonlinear system: the Jacobians of the nonlinear state-transition and state-observation functions are calculated to get the linear, state-space matrices $A[k]$, $B[k]$, $C[k]$ and $D[k]$ at each time step. In the *prediction* step, the prior state estimation is calculated using the inputs of the current time step and the estimations from the previous time step with

$$\hat{x}[k|k-1] = \mathrm{f}(\hat{x}[k-1|k-1], u[k]). \tag{3}$$

The prior state estimation covariance $P \in \mathbb{R}^{48\times48}$ is

$$P[k|k-1] = A[k]P[k-1|k-1]A[k]^T + Q. \tag{4}$$

In the *update* step, first, the innovation

$$\tilde{y}[k] = y[k] - \mathrm{h}(\hat{x}[k|k-1], u[k]) \tag{5}$$

is calculated. Then the Kalman gain, $K_G \in \mathbb{R}^{48\times 64}$

$$K_G[k] = P[k|k-1]C[k]^T(C[k]P[k|k-1]C[k]^T + R)^{-1}. \tag{6}$$

With the help of the Kalman gain, the posterior state vector

$$\hat{x}[k|k] = \hat{x}[k|k-1] + K_G[k]\tilde{y}[k], \tag{7}$$

and state prediction covariance

$$P[k|k] = (I - K_G[k]C[k])P[k|k-1] \tag{8}$$

is computed.

To obtain a point-wise linearization we use the LPV model presented in Section 2.2. During simulation, the true airspeed and roll angle are used as the scheduling parameters in the LPV model and are measured at each time step. The resulting $A[k]$, $B[k]$, $C[k]$ and $D[k]$ matrices are used in the calculations of the EKF. Then the EKF conducts the prediction and update steps. To determine Q, both the nonlinear and the LPV models are simulated with doublet inputs on the control surfaces and then the measured outputs and states are compared, and variances of the differences are calculated. The noise variances, R, of the onboard sensors of the T-Flex are specified based on the datasheets of the sensors.

4. Machine learning-based wing shape estimation

4.1. KalmanNet architecture

The other approach for estimating the flexible states of the T-Flex is to use machine learning. We employ the recently published KalmanNet architecture by (Revach et al. (2022)). The algorithm (or *pipeline*) for the KalmanNet is presented in Figure 2. KalmanNet combines Kalman filtering with a neural network as it uses similar *prediction* and *update* steps, but without computing the state prediction covariance matrix (P). Consequently, the model noise covariance matrix (Q) is not involved. For providing the Kalman gain (Step 4 in Figure 2), a trained Recurrent Neural Network (RNN) is used, thus the observation noise covariance matrix (R) is not involved either. The neural network uses the innovation difference $\Delta y[k] = y[k] - \hat{y}[k|k-1]$, the forward update difference $\Delta\hat{x}[k] = \hat{x}[k-1|k-1] - \hat{x}[k-1|k-2]$, and the roll angle ϕ scheduling parameter as input features. The advantage of the KalmanNet compared to the EKF is that it does not require any information about the model of the noise processes.

The standard Kalman gain predicting neural network presented by Revach et al. (2022) uses a Gated Recurrent Unit (GRU) as the recurrent layer and linear layers with Rectified Linear Units (ReLU) as the activation function. The neural network has a linear layer as the input layer with ReLU activation, followed by the GRU. After the GRU layer, there is another linear layer with ReLU activation, then the linear output layer. As the aircraft model we use is high-dimensional (48 states, 64 outputs), we slightly decreased the dimensions of each layer compared to the original architecture to reduce the computation burden.

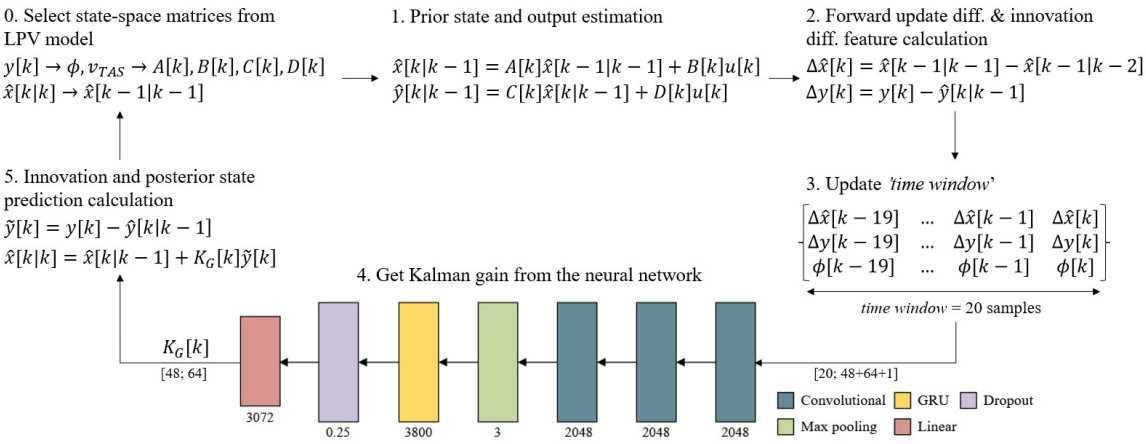

Figure 2: KalmanNet pipeline

Apart from the *linear RNN* architecture presented by Revach et al. (2022), we implement a different neural network loosely based on the work from Zhang et al. (2021). The network architecture still uses a GRU cell, but instead of linear layers, it uses three convolutional blocks at the beginning of the network. A convolutional block consists of a 1D convolutional layer followed by a ReLU activation function. After the ReLU a Batch Normalization layer is used, followed by a Dropout layer with 0.25 dropout probability. The output layer is a linear layer, which provides the Kalman gain matrix. The kernel size for each 1D convolution layer is seven. As the 1D convolutional layer requires a trajectory, or time-window of input features, simply using the forward update difference ($\Delta y[k]$), innovation difference ($\Delta\hat{x}[k]$) and roll angle (ϕ) input features of the current time step is not adequate. Therefore, we use the input features of the current time step and the input features from the previous 19 time steps in the time-window buffer. In Figure 2, the convolutional architecture represents the neural network. The number of features is shown below the convolutional blocks and the pool size below the max pooling layer. The number of units is indicated underneath the GRU and the linear layer. The dropout rate is shown below the dropout layer.

For initializing the layer weights, a standard normal distribution is used. Since the architecture incorporates a discrete-time system, it has a high sensitivity to the initial weight values. Therefore, the standard deviation of the normal distribution for the initialization has to be chosen very small ($5 \cdot 10^{-6}$) to avoid the otherwise highly diverging training process.

4.2. Training, validation and test data

Training, validation, and test datasets are generated using the high-fidelity nonlinear Simulink model of the T-Flex. The training dataset has four different trajectories that are generated with the help of a baseline controller designed by Ossmann et al. (2019). These four trajectories consist of an oval-shaped track, an '8-shaped' track, a trajectory where the controller only receives roll angle reference signals, and a trajectory where the controller receives altitude and velocity reference signals. To create rich datasets, while having realistic flight conditions, randomized wind gust and turbulence disturbances are used, together with Gaussian sensor noise, based on the flight test results of the T-Flex published by Bartasevicius et al. (2021) for each dataset. The main purpose of applying wind loads is to have disturbances that cannot be incorporated into any covariance matrix and to generate

more realistic trajectories. The initial velocity is set to $42\ \mathrm{m/s}$. The airspeed changes between $39\ \mathrm{m/s}$ and $51\ \mathrm{m/s}$, while the roll angle between $0°$ and $45°$. The barometric altitude also changes between $780\ \mathrm{m}$ and $820\ \mathrm{m}$. The trajectories of the training dataset are split into eight, 96-second long batches. The sampling time is set to $5\ \mathrm{ms}$, which results in 19200-sample long training batches. For training, a single batch is randomly selected from the eight in each epoch. However, validation and testing are conducted using only a trajectory where the aircraft follows the '8-shaped' track.

4.3. Training details

For the two neural network architectures, the training parameters are set with a custom-made hyperparameter optimization algorithm based on *RayTune*. The hyperparameter optimization has 20 runs, each lasting for 25 epochs. The hyperparameter optimization uses the same pipeline as standard training runs otherwise.

The optimizer algorithm is ADAM for both architectures. To avoid overfitting, weight decay is used. The prediction accuracy is calculated with Root Mean Squared Error (RMSE) function. However, although the linearized aircraft model is a stable system, the poles of the system are relatively close to the unstable region. So, a stability criterion is added to the RMSE loss function. It is possible to describe the complex system of the aircraft model joined with the Kalman filter with an error system $e[k+1] = (A[k] - K_G[k]C[k])e[k]$, where $K_G[k]$ is the Kalman gain, $e[k] = x[k] - \hat{x}[k|k]$ is the state prediction difference at time step k. If the state transition matrix of the error system $(A[k] - K_G[k]C[k])$ has any unstable poles, then the whole system is unstable. Hence, the RMSE loss is extended with the distance of the error system poles from the boundary of stability if it is greater than zero, thus making the loss value larger if the computed Kalman gain results in an unstable error system. This is especially useful for the convergence of the training.

An error metric is defined in decibels as $\mathrm{RMSE}_{\mathrm{dB}} = 10\ \lg(\mathrm{RMSE})$, for the sake of convenience during plotting, because the freshly initialized network tends to produce greater errors. This metric is solely used for evaluation and plotting. For optimizing the network weights, the standard RMSE loss value is used during backpropagation.

It is important to mention that the performance of the *linear RNN* architecture proved to be more stable than the *convolutional RNN*, which tends to get stuck in local optima. So, to overcome this issue, a reduction of the learning rate during training is necessary in that case. The threshold is set at $-21\ \mathrm{dB}$ – according to the decibel-based error metric – and the reduction factor is 0.05. The new learning rate is calculated as $\mathrm{lr}^{\mathrm{new}} = \mathrm{factor} \cdot \mathrm{lr}^{\mathrm{old}}$. The learning rate for the *linear RNN* architecture is $3.2 \cdot 10^{-6}$, while the weight decay is $7.5 \cdot 10^{-5}$. In the case of the *convolutional RNN* architecture, the value of the learning rate and the weight decay is $1.5 \cdot 10^{-7}$ and $9.5 \cdot 10^{-5}$ respectively.

5. Results

The performance of the different methods and architectures are evaluated on the 96-second long test dataset presented in Section 4, where the aircraft follows the '8-shaped' track with wind and turbulence disturbances present. Since the main purpose of the observer design is to observe the states describing the flexible dynamics, only the results for these states are presented. The first 4 modal coordinates are plotted, where U_{f1} is the 1^{st} symmetric bending and U_{f2} the 1^{st} asymmetric bending mode. U_{f3} denotes the 1^{st} symmetric torsion mode and U_{f4} is the 1^{st} asymmetric torsion mode. The two aerodynamic lag states are plotted as well.

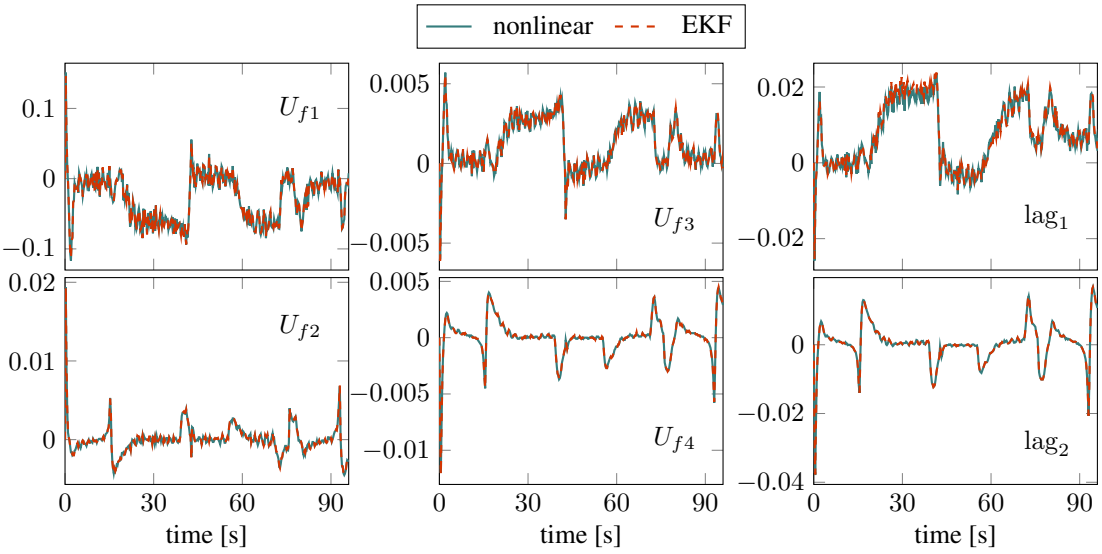

Figure 3: LPV-based EKF results

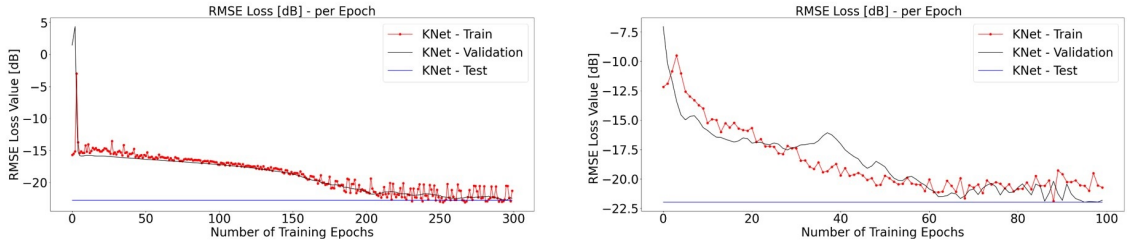

Figure 4: Linear (*left*) and convolutional (*right*) RNN architecture training graphs

5.1. LPV-based EKF

The results of the EKF-based state predictions are shown in Figure 3, where the data with the 'nonlinear' label show the states of the nonlinear model, while the 'EKF' show the states estimated by the filter. The results show that the designed filter accurately predicts the modal coordinates and the aerodynamic lag states even in the presence of wind disturbances whose effects cannot be incorporated into the observation noise matrix. The RMSE of the predictions for the plotted states is $8 \cdot 10^{-4}$ (-30.97 dB). Minor errors occur only during turning manoeuvres in state lag_1. The reason is that the LPV model is still just an approximation of the real, nonlinear system. However, these inaccuracies are inside the error tolerance for this problem.

5.2. KalmanNet

5.2.1. NEURAL NETWORK WITH LINEAR LAYERS

First, the slightly modified original KalmanNet architecture of Revach et al. (2022) – which uses linear layers with the GRU – is trained and evaluated. The training lasted for 300 epochs. Using an Nvidia Tesla V100 GPU with 32GBs of RAM, the whole procedure took 25 hours. The summary

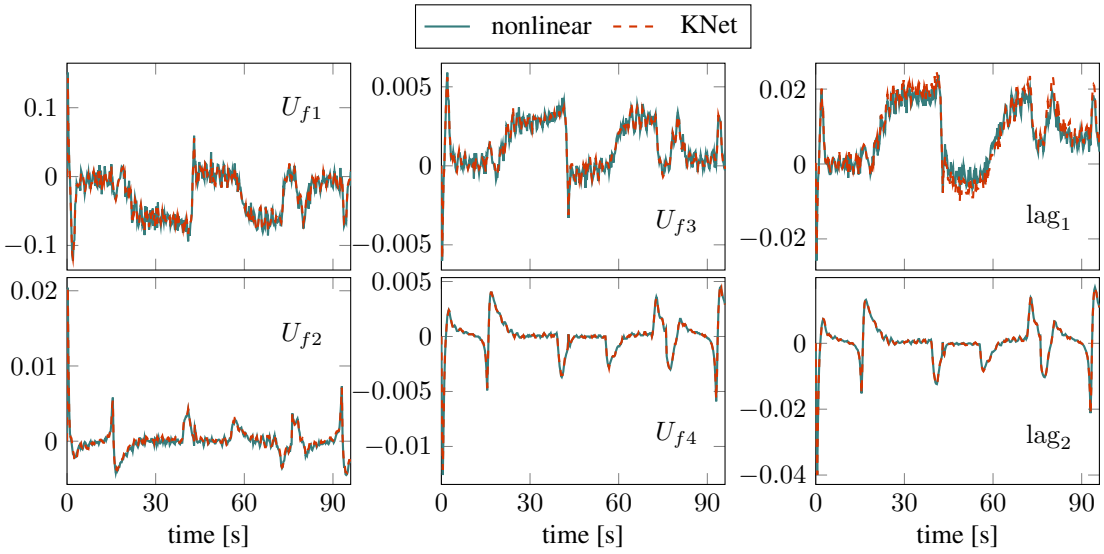

Figure 5: KalmanNet results with linear RNN architecture

of the training is presented in Figure 4 (*left*). The decibel-based metric presented in Section 4.3 is used for the plotting.

The trained model is evaluated on the same dataset as the LPV-based EKF. The results for the first four modal coordinates and the two aerodynamic lag states can be seen in Figure 5. From the results, it can be seen that the neural network managed to produce comparable results with the EKF in the case of U_{f1}, U_{f2}, U_{f3}, U_{f4} and lag_2. In the prediction of lag_1 however, a larger error is present between 40 s – 60 s, where the aircraft conducts a climb from 782 m to 803 m with heavier acceleration. The RMSE value of the predictions is $1.2 \cdot 10^{-3}$ (-29.1 dB).

5.2.2. NEURAL NETWORK WITH CONVOLUTIONAL LAYERS

Second, the proposed network architecture with convolutional layers is implemented, trained and evaluated. In this case, the duration of the training was 100 epochs that took 15 hours to complete using the V100 GPU. The 1D convolution expects a time series as an input, and the 20-sample long time window is used, which equals to 0.1 s trajectory with the 5 ms sampling time. Unfortunately, it was not possible to use a larger window size, as we ran out of GPU memory after a few training epochs (and training with CPU was not feasible, due to its slow execution speed). The training graph is shown in Figure 4 (*right*) with loss values in decibels.

Testing is done with the same dataset as in the previous approaches. The results for the first four modal coordinates and the two aerodynamic lag states are presented in Figure 6. The results indicate the following: the network manages to give very similar predictions as the LPV-based filter. In the case of lag_1 there are still larger errors present however, the prediction error in the 40 s – 60 s interval is smaller than in the case of the emphlinear RNN architecture (Figure 5). In this case, the RMSE is $1.29 \cdot 10^{-3}$ (-28.89 dB) for the first four modal coordinates and the two lag states.

Important to mention that both KalmanNet configurations managed to provide predictions with less noise and smaller error than the EKF in the case of the derivatives of the modal coordinates. The RMSE values for the derivative states when using the *linear* and the *convolutional* architectures

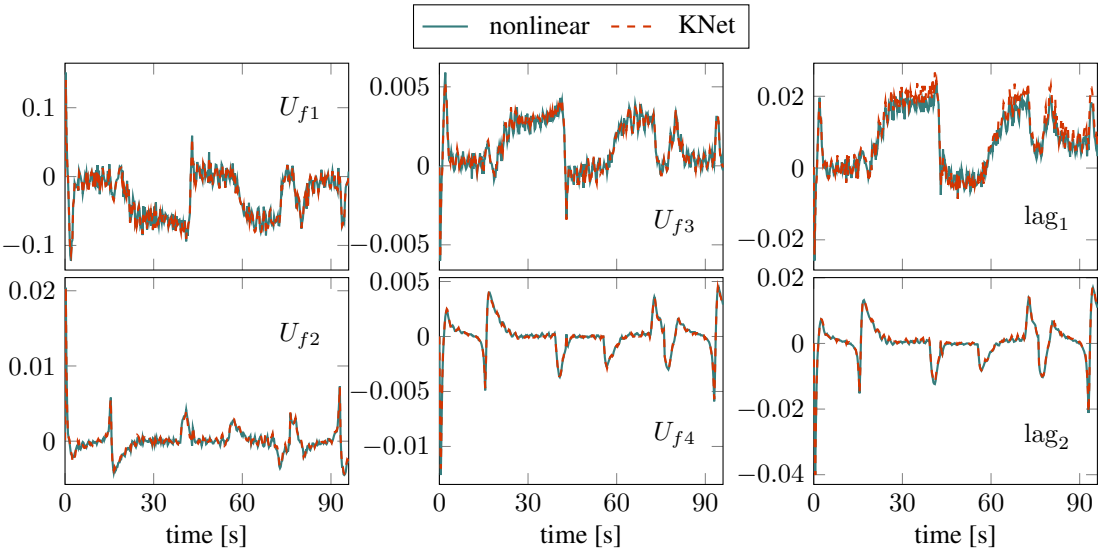

Figure 6: KalmanNet results with convolutional RNN architecture

are 0.008 (-20.97 dB) and 0.010 (-20.00 dB) respectively. When using the EKF, it is 0.0183 (-17.37 dB). As part of the disturbances comes from the effects of the wind gust load, these cannot be incorporated into either the observation noise matrix or the model noise matrix, so the EKF is incapable of smoothing them. However, the KalmanNet does not rely on information about the noise models, so it manages to smooth out the effects of the wind-caused disturbances for a certain degree in the estimations of the derivative states.

6. Conclusion

In this paper, we propose a model-based and a machine learning-based approach to estimate the states describing the flexible dynamics of the flexible aircraft called T-Flex. The model-based approach uses an LPV-based EKF, while the machine learning-based solution utilizes the KalmanNet architecture with two different neural network setups. We showed that the EKF-based estimator is able to predict the flexible and aerodynamic lag states. The neural network-based approach is also capable of estimating the above-mentioned states. However, in the case of lag_1 larger errors are present than in the predictions of the LPV-based EKF. Comparing the two neural networks it can be concluded that both provide relatively similar accuracy and get close to the accuracy of the LPV-based EKF. The training of the *linear RNN* architecture proved to be more stable, but it requires roughly double the training time as the *convolutional RNN* and has higher memory consumption. The long-term goal is to test both architectures using experimental flight data and incorporate them into the FCC of the T-Flex, thus making it possible to design a wing shape controller to minimize aerodynamic drag during flights.

Acknowledgments

The research leading to these results is part of the FLiPASED project. This project has received funding from the Horizon 2020 research and innovation programme of the European Union under grant agreement No 815058.

The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Autonomous Systems National Laboratory Program.

Special thanks to our colleague, Bálint Patartics for his practical suggestions and constructive advice.

References

Julius Bartasevicius, Sebastian J Koeberle, Daniel Teubl, Christian Roessler, and Mirko Hornung. Flight testing of 65kg t-flex subscale demonstrator. In *32nd Congress of the International Council of the Aeronautical Sciences*, pages 1–16. ICAS, 2021.

Quentin Arnaud Dugne-Hennequin, Hideaki Uchiyama, and João Paulo Silva Do Monte Lima. Understanding the behavior of data-driven inertial odometry with kinematics-mimicking deep neural network. *IEEE Access*, 9:36589–36619, 2021.

FliPASED. Flight Phase Adaptive Aero-Servo-Elastic Aircraft Design Methods. Horizon 2020 research and innovation programme of the European Union, grant agreement No 815058., 2019. URL https://flipased.eu/. (Accessed on 03. 2022.).

Leandro R Lustosa, Ilya Kolmanovsky, Carlos ES Cesnik, and Fabio Vetrano. Aided inertial estimation of wing shape. *Journal of Guidance, Control, and Dynamics*, 44(2):210–219, 2021.

Yasser M Meddaikar, Johannes Dillinger, Thomas Klimmek, Wolf Krueger, Matthias Wuestenhagen, Thiemo M Kier, Andreas Hermanutz, Mirko Hornung, Vladyslav Rozov, Christian Breitsamter, et al. Aircraft aeroservoelastic modelling of the flexop unmanned flying demonstrator. In *AIAA scitech 2019 forum*, page 1815, 2019.

Simon Muntwiler, Kim P Wabersich, and Melanie N Zeilinger. Learning-based moving horizon estimation through differentiable convex optimization layers. In *Learning for Dynamics and Control Conference*, pages 153–165. PMLR, 2022.

Daniel Ossmann, Tamas Luspay, and Balint Vanek. Baseline flight control system design for an unmanned flutter demonstrator. In *2019 IEEE Aerospace Conference*, pages 1–10. IEEE, 2019.

Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

Muhammad K Shereen, Muhammad I Khan, Naeem Khan, and Wasi Ullah. By the design and implementation of modified kalman filter for lpv systems. *International Journal of Engineering Works*, 3(4):26–31, 2016.

João Paulo Silva do Monte Lima, Hideaki Uchiyama, and Rin-ichiro Taniguchi. End-to-end learning framework for imu-based 6-dof odometry. *Sensors*, 19(17):3777, 2019.

Béla Takarics and Bálint Vanek. Robust control design for the flexop demonstrator aircraft via tensor product models. *Asian Journal of Control*, 23(3):1290–1300, 2021.

Béla Takarics, Bálint Vanek, Aditya Kotikalpudi, and Peter Seiler. Flight control oriented bottom-up nonlinear modeling of aeroelastic vehicles. In *2018 IEEE aerospace conference*, pages 1–10. IEEE, 2018.

Matthias Wüstenhagen, Thiemo Kier, Yasser M Meddaikar, Manuel Pusch, Daniel Ossmann, and Andreas Hermanutz. Aeroservoelastic modeling and analysis of a highly flexible flutter demonstrator. In *2018 atmospheric flight mechanics conference*, page 3150, 2018.

Ming Zhang, Mingming Zhang, Yiming Chen, and Mingyang Li. Imu data processing for inertial aided navigation: A recurrent neural network based approach. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3992–3998. IEEE, 2021.