Fault-Tolerant Model Predictive Control for Multirotor UAVs

Diget, Emil Lykke; Hasan, Agus; Manoonpong, Poramate

Published in:
2022 American Control Conference, ACC 2022

DOI:
10.23919/ACC53348.2022.9867240

Publication date:
2022

Document version:
Accepted manuscript

Citation for published version (APA):

Go to publication entry in University of Southern Denmark's Research Portal

Terms of use
This work is brought to you by the University of Southern Denmark. Unless otherwise specified it has been shared according to the terms for self-archiving. If no other license is stated, these terms apply:

• You may download this work for personal use only.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.
• You may freely distribute the URL identifying this open access version.

If you believe that this document breaches copyright please contact us providing details and we will investigate your claim.
Please direct all enquiries to puresupport@bib.sdu.dk

Download date: 15. Sep. 2023
Fault-Tolerant Model Predictive Control for Multirotor UAVs

Digite, Emil Lykke; Hasan, Agus; Manoonpong, Poramate

Published in:
2022 American Control Conference, ACC 2022

DOI:
10.23919/ACC53348.2022.9867240

Publication date:
2022

Document version:
Accepted manuscript

Citation for published version (APA):

Go to publication entry in University of Southern Denmark's Research Portal

Terms of use
This work is brought to you by the University of Southern Denmark.
Unless otherwise specified it has been shared according to the terms for self-archiving.
If no other license is stated, these terms apply:

- You may download this work for personal use only.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying this open access version

If you believe that this document breaches copyright please contact us providing details and we will investigate your claim.
Please direct all enquiries to puresupport@bib.sdu.dk

Download date: 19. okt. 2022
Fault-Tolerant Model Predictive Control for Multirotor UAVs

Emil Lykke Diget¹, Agus Hasan² and Poramate Manoonpong³

Abstract—This paper presents a method for advanced fault-tolerant control (FTC) of multirotor unmanned aerial vehicles (UAVs), which includes anomaly detection on sensor measurements, fault estimation on actuators, and a robust model predictive control (MPC). To detect anomalies on the sensor measurements, an Echo State Network is used. System states and faults are estimated using an adaptive extended Kalman filter. The system is further controlled using MPC. The method is tested in numerical simulations with a hexacopter dynamic model. Simulation results show the ability of the FTC to handle failure with different even and uneven actuator faults.

I. INTRODUCTION

For unmanned aerial vehicles (UAVs) or drones, safety is important as sensors and actuators are prone to failure. To prevent drones from falling down and possibly fatally injuring people, countermeasures have to be taken. One way to introduce safety into the platform is by adding algorithms to detect, localize, estimate, and react once the sensors or the actuators fail. This can be done by the use of fault diagnosis (FD) and fault-tolerant control (FTC) techniques.

In [1], Xulin and Yuying presents a fault-tolerant controller using fuzzy logic. They estimate the system states using a Luenberger state observer. The observer estimates the outputs of each rotor and compares it to the desired values from the controller. If the error is too large it marks an actuator fault. Using this knowledge they adapt the control using control allocation. Another method is to estimate the magnitude of the fault. In [2], Zhang presents an adaptive Kalman filter (AKF) to estimate the states and the actuator faults simultaneously for linear time-varying systems. This method is expanded for non-linear systems in [3], where the adaptive extended Kalman filter (AEKF) was applied and the model was linearized. In [4], Hasan presents a more advanced version of the AEKF, called the adaptive eXogenous Kalman filter (AXKF). The advanced version involves a non-linear adaptive observer as a pre-step before the AEKF to add robustness to the method. In [5], Rot et al. uses the AXKF to estimate the magnitude of the actuator faults in a hexacopter.

In [6], Hasan et al. uses the fault estimate to trigger a parachute fail-safe system on a hexacopter.

A way of implementing fault-tolerance in the model predictive control (MPC) is discussed by Camacho et al. in [7], where the actuator faults are detected in a binary fashion. The authors only considers two types of failures: Complete failure and actuator jam. In [8], Maciejowski and Jones show the importance of fault-tolerant control (FTC). They study the case of the fatal crash of El Al Flight 862, which might have been avoided by using FTC. They simulate the airplane and the pilot as two different MPCs. They assume that the airplane MPC have information from an FDI unit which enables them to alter the dynamic model on the fly.

All these fault-tolerant control methods [1]–[8], haven’t considered sensor faults. The sensors of a system are also prone to errors. In [9], Tu et al. develops a method of detecting an IMU attack based on the residual error between sensors readings and the estimated state. To detect anomalies, i.e. deviations from the expected state, machine learning based methods can be applied. Chang et al. [10] apply an Echo State Network (ESN) to detect anomalies, by training the ESN to predict the sensor signal.

In this paper methods that could be used for a fault-tolerant controller that is both tolerant to sensor and actuator faults are presented and applied on a hexacopter model. Thus, this paper proposes an approach to FTC that combines explicit fault estimation using AEKF and sensor anomaly detection using ESN with a fault-tolerant MPC. The paper is working towards providing a framework for safer drone flight where a drone can stay in control despite of faults in the sensors and in the actuators.

II. METHODS

In this section a dynamic model of the hexacopter, as seen in Fig. 1a, is introduced. Afterwards, the AEKF used for state and fault estimation is briefly discussed. Furthermore, the estimated state and fault are used in the MPC to make the system tolerant to faults. At the end, a method of detecting sensor anomalies using an ESN is described. An overview of our proposed framework integrating the AEKF, MPC, and ESN methods for the FTC can be seen in Fig. 1b.

A. Hexacopter Modelling

In this study, the proposed controller is implemented on a hexacopter (Fig. 1a). The state of the drone is defined as:

\[
\mathbf{X} = [\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ \mathbf{x} \ \dot{x} \ \mathbf{y} \ \dot{y} \ \mathbf{z} \ \dot{z}]^\top, \tag{1}
\]

1E.L. Diget is with SDU Robotics, Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Denmark (email: eld@mmmi.sdu.dk).
2A. Hasan is with Department of ICT and Natural Sciences, Norwegian University of Science and Technology, Norway (email: agus.hasan@ntnu.no).
3P. Manoonpong is with Embodied AI and Neurorobotics Lab, SDU Birobotics, Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Denmark and also with Vidyasirimedhi Institute of Science and Technology, Thailand (email: poma@mmmi.sdu.dk).
where \( \phi \), \( \theta \), and \( \psi \) are roll-, pitch-, and yaw angle, respectively, and \( x \), \( y \), and \( z \) are the position in the three axes. \( \dot{} \) denotes the derivative with respect to time. The hexacopter is modelled as presented extensively in literature, \( X_k \) are roll-, pitch-, and yaw angle, respectively.

B. Fault Estimation using AEKF

We use an adaptive extended Kalman filter (AEKF) presented in [3] to estimate the state of the system \( X_k \) and the magnitude of the actuator fault, \( \vartheta_k \). The fault is defined from 0 to 1, where 0 denotes normal operation and 1 denotes complete failure. One fault is defined for one control input. The dynamic model of the hexacopter, can be written in the following standard form [5], [11]:

\[
X_{k+1} = f(X_k, u_k, \vartheta_k) + w_k, \quad Y_k = CX_k + v_k,
\]

where \( u_k = [F_1 \ldots F_6]^\top \) is the input vector, \( f(X_k, u_k, \vartheta_k) \) is the nonlinear model, \( w_k \sim \mathcal{N}(0, Q_{AEKF}) \) is the process noise defined with a covariance of \( Q_{AEKF} \in \mathbb{R}^{12 \times 12} \), \( Y_k \) is the measurement vector, \( C \in \mathbb{R}^{6 \times 12} \) is the measurement matrix, and \( v_k \sim \mathcal{N}(0, R_{AEKF}) \) is the measurement noise defined with a covariance of \( R_{AEKF} \in \mathbb{R}^{9 \times 9} \). In this study all states apart from the linear velocities are measured, thus:

\[
C = \begin{bmatrix}
0_{9 \times 6} & 0_{6 \times 6} \\
0_{6 \times 6} & 0_{9 \times 6}
\end{bmatrix}.
\]

The nonlinear model (2), can be linearized and discretized around the priori state estimate \( \hat{X}_k \), thus obtaining the following discrete-time system:

\[
X_{k+1} = F(X_k)X_k + B_k u_k + E_k \vartheta_k,
\]

where \( E_k = -B_k \text{diag}(\vartheta_k) \) is a matrix that marginalizes the effect of the input \( u_k \) based on the fault \( \vartheta_k \), such that the actual control input to the system becomes \( B_k(I - \text{diag}(\vartheta_k))u_k \). If the drone experiences complete failure on all rotors, then \( \text{diag}(\vartheta_k) = I \), the system input would be zero.

The AEKF uses an update law to estimate the fault through the following equation:

\[
\hat{\vartheta}_{k+1} = \hat{\vartheta}_k + \Gamma_k(Y_k - CX_k),
\]

where \( \Gamma_k \) is the parameter estimate gain matrix, which is obtained from a recursive algorithm involving some auxiliary variables and a forgetting factor. Details about the AEKF can be found in [2], [3].

C. Fault-Tolerant MPC

Model Predictive Control (MPC) is used in this study. MPC is an optimal control technique. At each time step a cost function is minimized to determine the next control input \( u_k \). Pseudocode for the implementation can be seen in Algorithm 1. The fault estimate \( \hat{\vartheta}_k \) and the reference states \( \vartheta_{k,ref} \) are assumed constant along the prediction horizon.

In this case the yaw angle and the position should be controlled, i.e., \( \gamma = [\psi \ x \ y \ z]^\top \). \( \gamma \) is derived from the state \( X, \gamma = HX \), where \( H \in \mathbb{R}^{4 \times 12} \) is defined as:

\[
H = \begin{bmatrix}
0_{4 \times 4} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}.
\]

The MPC minimizes the following cost function:

\[
\min_{\zeta_k} J_k = \left( \sum_{i=1}^{N_c} \frac{1}{2} \| \gamma_{k+i} - \gamma_{k,ref} \|_Q^2 \right) + \| \epsilon_x(k) \|_\infty \] + \left( \sum_{j=1}^{N_r} \frac{1}{2} \| \Delta u_{k+j-1} \|_{R_{act}}^2 \right) + \| \epsilon_c(k) \|_\infty,
\]

subject to

\[
X_{k+1} = f(X_k, u_k, \vartheta_k), \quad X_0 : \text{initialized}, \quad u_0 : \text{initialized},
\]

\[
\vartheta_k : \text{constant}, \quad \gamma_{k,ref} : \text{constant},
\]

\[
\zeta_k = [u_k \ u_{k+1} \ldots \ u_{k+N_c}]^\top,
\]

\[
\Delta u_k = u_k - u_{k-1},
\]

\[
X_{\text{low}} \leq X \leq X_{\text{high}}, \quad u_{\text{low}} \leq u_k \leq u_{\text{high}}.
\]
\[-\Delta u_{k}^{\text{high}} \leq \Delta u_{k} \leq \Delta u_{k}^{\text{high}}, \quad Q_{\text{MPC}} \succeq 0, \quad R_{\text{MPC}} \succeq 0, \]

where \(\|e\| = \alpha^T \alpha\). \(Q_{\text{MPC}}\) is the penalty matrix on the error in the controlled variable, \((\gamma_{k} - \gamma_{k-1, \text{ref}})\), \(R_{\text{MPC}}\) is the penalty matrix on the input rate \(\Delta u\), \(N_{u}\) is the prediction horizon, and \(N_{c}\) is the control horizon. \(X^\text{low}\) and \(X^\text{high}\) are the bounds on the states, \(u^\text{low}\) and \(u^\text{high}\) are the bounds on the inputs, and \(\Delta u^\text{high}\) is the symmetrical bound on \(\Delta u_{k}\). \(\epsilon_{s}\) and \(\epsilon_{c}\) are soft constraints for \(X_{k}\) and \(\Delta u_{k}\), respectively. The cost should be increased when the constraints are violated. The constraints are written as two functions; \(c_{s}(X_{k})\) for the state constraints and \(c_{c}(\Delta u_{k})\) for the change in input constraints.

A smooth soft constrain on the following form is applied:

\[ \epsilon_{s}(k) = \frac{\ln (1 + \exp(\rho c_{s}(X_{k})))}{\rho}, \]

where \(\rho\) is a tuning parameter that controls how close the softplus-function is to the max-function, [12], and \(I\) is a column vector of ones with the appropriate size. The constraint is similarly defined for \(c_{c}(k)\).

To get a scalar that represents \(\epsilon_{s}\) and \(\epsilon_{c}\), the \(L^\infty\)-norm or maximum norm is applied, defined as:

\[ ||v||_{\infty} = \max \{|v_1|, |v_2|, \ldots, |v_n|\}, \]

where \(v\) is a placeholder vector, and the subscripted \(v_1, v_2, \ldots, v_n\) are the vector elements. The state constraint function and the constraints on the input change are defined as follows:

\[ c_{s}(X_{k}) = \begin{bmatrix} |\phi| - \phi_{\text{bound}} \\ |\theta| - \theta_{\text{bound}} \\ |\dot{z}| - 3\theta_{\text{bound}} \end{bmatrix}, \quad c_{c}(\Delta u_{k}) = \Delta u_{k}^\text{low} - \Delta u_{k}^\text{high}, \]

which defines the constraints on the roll angle, \(\phi\), the pitch angle, \(\theta\), and the linear velocities \(\dot{z}\), \(g\), and \(\ddot{z}\). \(\Delta u_{k}^\text{low}\) denotes the element wise absolute value of \(\Delta u_{k}\). Under normal operation the elements of \(c_{s}(X_{k})\) are negative. If the constraint are violated, it will be positive.

The constraints are only applied to the first state and the first input change, respectively, as it’s only the first input that is applied to the system. In the implementation of the cost function the system is simulated from the initial state \(N_{p}\) steps forward, from time \(k\) to \(k + N_{p}\) using the dynamic model (2). The cost function is minimized for \(N_{e}\) control inputs. \(N_{e}\) should be lower than \(N_{p}\). By having a longer \(N_{p}\), the effect of the last control input is allowed to settle in the cost function.

D. Measurement Anomaly Detection using ESN

To detect anomalies in the sensor data, a system based on the one presented in [10]. For this study we have in total nine sensor signals; three angles, three angular rates, and three positions. To get the network inputs, we normalise the sensor signals as follows; the angles are divided by \(\pi/2\), the angular rates by \(\pi\) and the positions by 100.

The signals are grouped as follows: (roll, roll rate, pitch-position), (pitch, pitch rate, x-position), and (yaw, yaw rate, z-position). One network for each signal group is used. Each ESN consists of three layers; input, hidden, and output, with sizes \(N_{u}\), \(N_{e}\), and \(N_{w}\), respectively. As depicted in Fig. 2 the connections between the layers have weight matrices \(W_{in} \in \mathbb{R}^{N_{i} \times N_{x}}\), \(W_{hid} \in \mathbb{R}^{N_{x} \times N_{y}}\), and \(W_{out} \in \mathbb{R}^{N_{y} \times N_{z}}\), where \(W_{res}\) and \(W_{out}\) are randomly initialized with values from a uniform distribution of \([-0.4: 0.4]\) and \([-1: 1]\), respectively, both with a sparsity of 10\%, i.e. 90% of the elements are zero. \(W_{res}\) is resized to a spectral radius \(< 1\), specifically 0.99 to fulfill the echo state property. The following equations are the update rule and the readout equation, respectively, for the ESN:

\[ A_{k+1} = (1 - \lambda_{\text{ESN}}) A_{k} + \lambda_{\text{ESN}} \tanh(W_{in}M_{k+1} + W_{hid}A_{k}), \]

\[ N_{k+1} = W_{out} A_{k}, \]

where \(A_{k} \in \mathbb{R}^{N_{x} \times 1}\) is the reservoir state, \(\lambda_{\text{ESN}}\) is the leaking rate, \(M_{k} \in \mathbb{R}^{N_{x} \times 1}\) is the sensor measurement vector, and \(N_{k} \in \mathbb{R}^{N_{z} \times 1}\) is the output vector of the network.

The ESN is used to predict the next measurement of the data. Thus in the training sequence the network is presented with the same signals as inputs and outputs, with the output signals being shifted one step forwards in time. Due to the input and output signals being the same, \(N_{w} = N_{y} = 3\). In this paper \(N_{x} = 50\). Recursive least squares (RLS) is used to train the network as presented in [13] with a learning rate \(\lambda_{\text{RLS}} = 0.9999995\).

To find anomalies, the prediction error is calculated as \(\delta_{k} = M_{k} - N_{k}\). Peak recognition is used to find peaks in the prediction error as they represent anomalies [10]. Under normal operation the prediction error should be close to zero, and the anomaly detector should just pass the measurement \(Y_{k}\) along with it’s normal covariance \(R_{\text{AEKF}}\) to the AEKF. If there is an anomaly, the corresponding covariance is changed to a very high value, e.g., \(1 \times 10^{8}\) such that the specific measurement is mostly ignored by the AEKF.

Fig. 2. The ESN used in this study act as a non-linear predictor with embedded temporal memory. In this setup three measurements, \(M_{k}\), are used as inputs and a prediction of the three next measurements, \(N_{k}\), are the outputs. The solid lines, \(W_{in}\) and \(W_{res}\), are randomly initialized and the dotted lines, \(W_{out}\), are trained. Three networks are used; one for roll, pitch and yaw angles, one for the three angular rates, and one for the positions.
### III. Numerical Simulations

In this section numerical simulations are performed to test the presented methods. In subsection III-A, the fault tolerance of the AEKF- and EKF-based MPC is compared. To show the AEKF-based MPC can function under an uneven (asymmetric) fault, the method is tested in subsection III-B. The ESN-based anomaly detection system is tested in two experiments, subsection III-C and subsection III-D. First the signal prediction ability of the ESN is compared with a simple predictor, a moving average low-pass filter (LPF) with an increasing level of noise. Second the anomaly detection abilities of the ESN-based anomaly detector is reviewed and compared with an LPF-based one. The parameters for the different parts of the system can be found in Table I. The optimization problem is solved using the software package NLopt by Johnson [14] using the SLSQP algorithm.

#### A. Fault Tolerance Analysis

This simulation aims to compare the fault-handling of the proposed fault-tolerant MPC compared with an ordinary MPC with EKF as the Estimator, see Fig. 1b. To do this several simulations were performed. The fault on all motors were increased evenly (symmetrically) starting from 0.00 until the drone couldn't fly in steps of 0.01. The fault was introduced after 5 s, and the system was simulated for 25 s. For each fault value the simulation was run five times and the root mean square error (RMSE) was calculated. The mean and standard deviation of this was calculated. The experiment was conducted for both the AEKF and the EKF.

The system was simulated to follow a Gerono lemniscate, which is a curve forming a figure eight. The parametric equations for the control reference are:

\[
g_{k,ref}(t) = \begin{bmatrix} x_{k,ref}(t) \\ y_{k,ref}(t) \\ z_{k,ref}(t) \end{bmatrix},
\]

\[
g_{k-1,ref}(t) = \begin{bmatrix} x_{k-1,ref}(t) \\ y_{k-1,ref}(t) \\ z_{k-1,ref}(t) \end{bmatrix},
\]

\[
y_{k,ref}(t) = 5 \sin \frac{\pi t}{3} - \frac{1}{3}.
\]

For position coordinates the RMSE was defined as:

\[
\text{RMSE} = \sqrt{\frac{\|\Delta x_k\|^2 + \|\Delta y_k\|^2 + \|\Delta z_k\|^2}{N}}.
\]

Where the fault causes the drone to deviate from the course, later to follow it again. The EKF-based MPC can fly with a small fault level, but it can’t reach the reference properly, see c) and e) in Fig. 3.

#### B. Asymmetric Fault

In this case three asymmetric faults were introduced at three different times:

\[
\Phi = \begin{cases} 
0.0 & \text{if } t \in [0;10] \\
0.1 & \text{if } t \in [10;40] \\
0.2 & \text{if } t \in [40;70] \\
0.3 & \text{if } t \in [70;100] 
\end{cases}.
\]

This experiment used the same parameters as presented in the earlier experiment, subsection III-A. The estimated faults can be seen in Fig. 4. After the three changes at 10s, 40s, and 70s, major spikes are going both up and down in the fault estimate. Though the estimate converged quickly to the true value, in around 10s.

#### C. Signal Prediction with Gaussian Noise

We tested how robust the proposed ESN can predict the sensor signal when facing different noise levels. The ESN is trained to predict the roll angle, the roll angular rate, and the \(y\)-position using data from the simulations from the prior experiments with the parameters given in the two bottom parts of Table I. The purpose of this experiment was to compare how well the ESN predicts a signal compared to a traditional predictor, a moving average with window size 10.
The result of the fault tolerance analysis. On the left a graph shows the RMSE with respect to a specific even fault for the AEKF-based MPC and the EKF-based MPC, respectively. Six green crosses marks specific fault levels. 3D positions corresponding to these levels can be seen on the right. a) shows the trajectory when there is no fault. The two cases are equal, and the drone follows the trajectory nicely. b) and c) shows 0.05 fault in the AEKF and the EKF case, respectively. It can be seen that the AEKF handles the actuator fault, and the drone quickly comes back to the reference. The EKF-based MPC can handle the error, but it doesn’t reach the reference signal. In d), the AEKF, and e), the EKF, the fault is 0.10. Here, in the AEKF case the drone dips a bit after the fault, but it gets back on track. The EKF-based MPC can’t handle the fault, and instead slides along the ground plane. f) shows a fault 0.40, which the AEKF can handle, though it slides a bit on the ground plane. The AEKF can handle up to around 0.45 fault, after which the controller can’t find a suitable solution. An animation of the experiment results can be seen on [http://www.manoonpong.com/FTC/SupplementaryVideo1.mp4](http://www.manoonpong.com/FTC/SupplementaryVideo1.mp4).

White noise was added to the input signal, and the prediction was compared with the target. The white noise was scaled according to the range of the original input signal, such that 100% white noise corresponds to $\sim N(0,1)$ range. The noise was from 0% to 100% in steps of 1%, and each experiment was repeated five times. To compare how well they predict the target, a normalized root mean square error (NRMSE) was computed for each of the three signals:

$$NRMSE = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - d_i)^2}}{d_{\text{max}} - d_{\text{min}}},$$

where $y$ is the vector of predictions, $d$ is the vector of targets, $N$ is the number of samples, $d_{\text{max}}$ is the maximum value of the target signal, and $d_{\text{min}}$ is the minimum value of the target signal.

The result of the experiment can be seen in Fig. 5. In the roll- and the $y$-signal the ESN predicts the signal with a lower NRMSE compared to the LPF. These signals also moves...
Fig. 6. The result of the anomaly detection experiment detecting noise spikes. The heatmap shows the spike altitude as a factor of the signal standard deviation ($\sigma$) along the horizontal axis, and the period of the spike occurrences along the vertical axis. Each square describe the MCC of that specific case. a) The result of the proposed ESN-based anomaly detector. b) The result of the simple LPF-based anomaly detector.

D. Anomaly Detection with Sensor Faults

This experiment tests the effectiveness of the proposed anomaly detection scheme. The ESN used in this experiment was the same as the one in the last experiment. It was compared with a identical scheme except with a moving average filter with window size 10 as the predictor. Noise spikes were added to the data with a period ranging from 10 to 100 samples in steps of 10. To create a noisy measurement a factor of the signal standard deviation is added to the measurement, following:

$$m_k = \bar{m}_k + \beta \sigma,$$

where $m$ is a measurement, $\bar{m}$ is the noisy measurement, $\beta$ is the spike amplitude, and $\sigma$ is the signal standard deviation, following [10]. $\beta$ is ranging from 1–10 in steps of 1. The Matthews correlation coefficient is calculated using the confusion matrix for each result:

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}.$$  

where TP is true positives, FP is false positives, TN is true negatives, and FN is false negatives. The MCC ranges from the worst value −1 to the best value 1.

The result is presented as heatmaps in Fig. 6, where yellow is the best result, and purple is the worst. The columns are the roll, roll rate, and the $y$ respectively. For both cases performance worsens when the amplitude of the spikes are larger and they are closer together. Comparing the columns directly, the ESN-based anomaly detector has a better result with a larger section of the heatmaps having higher values compared to the LPF.

IV. CONCLUSIONS

This paper presents a fault-tolerant control framework with sensor anomaly detection using ESN and actuator fault estimation using AEKF and MPC. Through a fault tolerance analysis it was shown that the AEKF-based MPC gave superior control compared to using a conventional EKF-based MPC, and it could even adapt to different motor fault levels throughout the simulation. The ESN was compared to the LTF with respect to their signal prediction and anomaly detection performance. The ESN-based anomaly detection gave better results compared to the LPF-based solution. Further research should focus on bringing the framework to a physical drone platform to test it in the real world.

ACKNOWLEDGMENT

This paper is partially funded by Equinor’s gift professorship in system dynamics for development of digital twin at NTNU.

REFERENCES