Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-6 (2022) 285–291

Beta Residuals: Improving Fault-Tolerant Control for Sensory Faults via Bayesian Inference and Precision Learning

Mohamed Baioumy * **William Hartemink** **
Riccardo M.G. Ferrari *** **Nick Hawes** *
* *Oxford Robotics Institute, University of Oxford, United Kingdom. For correspondence: mohamed@robots.ox.ac.uk.*
** *Amazon Web Services, Texas, United States.*
*** *Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands. (e-mail: r.ferrari@tudelft.nl)*

Abstract: Model-based fault-tolerant control (FTC) often consists of two distinct steps: fault detection & isolation (FDI), and fault accommodation. In this work we investigate posing fault-tolerant control as a single Bayesian inference problem. Previous work showed that precision learning allows for stochastic FTC without an explicit fault detection step. While this leads to implicit fault recovery, information on sensor faults is not provided, which may be essential for triggering other impact-mitigation actions. In this paper, we introduce a precision-learning based Bayesian FTC approach and a novel *beta residual* for fault detection. Simulation results are presented, supporting the use of beta residual against competing approaches.

1. INTRODUCTION

The emergence of autonomous robotic systems calls for the introduction of fault-tolerant control architectures that can guarantee safe operation even under faulty conditions. *Fault-Tolerant Control* (FTC) techniques can limit the impact of faults at process, sensor or actuator level (Blanke et al., 2000; Chen and Patton, 1999). FTC approaches typically consists of two steps: fault detection, isolation and identification (FDI); and fault accommodation.

Model-based approaches to FTC proved to be very powerful (Gao et al., 2015), but requires high amounts of knowledge and data for modelling the system and designing FDI residuals and thresholds. This holds especially for cases modelled as stochastic systems, where probabilistic rather than deterministic solutions are preferred.

A notable contribution to solving this problem has been provided by the prolific literature on Bayesian approaches to FDI. Cai et al. (2017) provides an overview of early works which used static Bayesian Networks to model the causal relation between faults and symptoms, and obtain a diagnosis via inference. Later works introduced Dynamic Bayesian Networks (DBN) as a way to encode transient and time varying behaviours, and general spatial and time relations. For instance Lerner et al. (2000) and Verma et al. (2004) use, respectively, a Kalman and a Particle Filter to track possible fault hypotheses, while Codetta-Raiteri and Portinale (2015) illustrate the use of Probabilistic Graphic Models (PGM) and DBNs for spacecraft FDI. In Sun et al. (2020) a Bayesian Recurrent Neural Network is used as a probabilistic model, whose posterior is approximated via a variational approach.

Still, several challenges remain at the forefront of research on this topic. On one side, an integrated Bayesian approach for both FDI and fault recovery is sought. On another, the computational complexity of inference over DBN can be unsuited to embedded implementations on board autonomous robots, when complex models are employed. Finally, Bayesian approaches need adaptation capabilities to cope with time varying systems, for instance due to ageing or other non-fault phenomena.

In order to address the first two issues, several works in the field of robotics and control have recently taken inspiration from the *active inference* framework. In neuroscience, it is regarded as a general theory for perception and action (Friston, 2010) and can be used to understand decision making of biological agents and to build artificial agents. Active inference methods for control have shown promising results in deterministic FTC (Pezzato et al., 2020; Baioumy et al., 2021c), stochastic FTC (Baioumy et al., 2021b), adaptive control for robot manipulators (Buckley et al., 2017; Pezzato et al., 2020; Baioumy et al., 2021a) and state-estimation (Lanillos and Cheng, 2018; Friston et al., 2010).

In this work we address the third challenge as well, by presenting a Bayesian controller for stochastic fault-tolerant control that can track naturally occurring variations and faults via *precision learning*. . In particular, we model the precision (inverse covariance) of each sensor as a random variable and compute on-line its posterior. The *expected precision* acts as a measure of the probability of a sensor being faulty, and its inverse is used to weight a given sensor measurement in the control law. This approach does not require a-priori information about fault characteristics, or an explicitly-defined fault detection criteria and recovery

2405-8963 Copyright © 2022 The Authors. This is an open access article under the CC BY-NC-ND license.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2022.07.143

mechanisms. Still, operators may need an interpretable estimate of the location and probability of a fault. To address this, we introduce a so-called *beta residual* based on the estimation of hyper-parameters of the precision distribution density. A corresponding residual evaluation based on the logistic function is then proposed.

The main contributions of this paper can then be summarised as

(1) a set of controllers performing *sensor* FTC as Bayesian inference by means of precision learning;
(2) a novel approach for extracting an *interpretable* quantity, the beta-residual, representing the controller's Bayesian belief of sensor faultiness.

The rest of the paper is organized as following: Section 2 introduces two types of Bayesian control methods which are used in Section 3 to implement FTC via precision learning. The interpretable beta residual is defined in Section 4, while simulation results and conclusions are presented, respectively, in Section 5 and 6.

2. CONTROL MODELS

In this section, we recall results from prior work to motivate the work proposed in this paper. We first formalize the system dynamics in a problem statement; then, we present two controllers based on Bayesian inference: the unbiased active inference controller (u-AIC) of (Baioumy et al., 2021c,b) and a general Bayesian Controller (BC).

2.1 Problem statement

Let us consider a discrete-time nonlinear system as

$$\begin{cases} \boldsymbol{x}_{t+1} &= f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{q} \\ \boldsymbol{y}_t &= g(\boldsymbol{x}_t) + \boldsymbol{\eta} \end{cases}, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{N_x}$, $\boldsymbol{u}_t \in \mathbb{R}^{N_u}$ and $\boldsymbol{y}_t \in \mathbb{R}^{N_y}$ are, respectively, the system's state, input and output vector. The variables $\boldsymbol{q} \in \mathbb{R}^{N_x}$ and $\boldsymbol{\eta} \in \mathbb{R}^{N_y}$ are zero-mean, Gaussian process and measurement noises, instead. The controller aims to steer the system to a goal state $\boldsymbol{x}_{\text{goal}}$ by applying a control action $\boldsymbol{u}_t$. At any time, unknown to the controller, sensor faults can occur. This includes *hard faults* (complete failure) and *soft faults* (e.g. arbitrary sensor drifts or constant offsets).

2.2 The Unbiased Active Inference Controller

The (u-AIC) uses a generative probabilistic model with the current state $\boldsymbol{x}$, control action $\boldsymbol{u}$ and observations $\boldsymbol{y}$. The system can have multiple observation modalities. For instance, a robot can have joint encoders for position and velocity in addition to a camera. As a running example throughout the paper, we derive the equations for a system with two joint encoders $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}$ and a visual sensor $\boldsymbol{y}^{(v)}$. In this case the joint distribution $p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)})$ is assumed to be factorized as:

$$p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}) = \underbrace{p(\boldsymbol{u}|\boldsymbol{x})}_{control} \underbrace{p(\boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{x})}_{observation\ model} \underbrace{p(\boldsymbol{x})}_{prior} \tag{2}$$

Given the sensor data, we then aim to find the posterior over states and actions $p(\boldsymbol{x}, \boldsymbol{u}|\boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)})$. As is common in variational Bayesian inference, we approximate the posterior with a variational distribution $q(\boldsymbol{x}, \boldsymbol{u})$ and utilize the mean-field assumption ($q(\boldsymbol{x}, \boldsymbol{u}) = q(\boldsymbol{x})q(\boldsymbol{u})$) and the Laplace approximation Fox and Roberts (2012). The posterior over the state $\boldsymbol{x}$ is assumed Gaussian with mean $\boldsymbol{\mu}_x$. The posterior over actions $\boldsymbol{u}$ is also assumed Gaussian with mean $\boldsymbol{\mu}_u$. This results in the expression for *variational free-energy*, a quantity which is then minimized to generate control actions:

$$F = -\ln p(\boldsymbol{\mu}_u, \boldsymbol{\mu}_x, \boldsymbol{y}^{(v)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}) + C. \tag{3}$$

Detailed derivations of the above are available in Baioumy et al. (2021c), while more general information about variational inference is available in Murphy (2012). It then follows, assuming Gaussian distributions, that F can be factorized as in eq. (2) and then expanded to

$$\begin{aligned} F = \frac{1}{2}(&\boldsymbol{\varepsilon}_{y^{(1)}}^\top P_{y^{(1)}} \boldsymbol{\varepsilon}_{y^{(1)}} + \boldsymbol{\varepsilon}_{y^{(2)}}^\top P_{y^{(2)}} \boldsymbol{\varepsilon}_{y^{(2)}} + \boldsymbol{\varepsilon}_{y^{(v)}}^\top P_{y_v} \boldsymbol{\varepsilon}_{y^{(v)}} \\ &+ \boldsymbol{\varepsilon}_x^\top P_x \boldsymbol{\varepsilon}_x + \boldsymbol{\varepsilon}_u^\top P_u \boldsymbol{\varepsilon}_u - \ln |P_u P_{y^{(1)}} P_{y^{(2)}} P_{y^{(v)}} P_x|) + C, \end{aligned} \tag{4}$$

where $\boldsymbol{\varepsilon}_{y^{(1)}}$, $\boldsymbol{\varepsilon}_{y^{(2)}}$, $\boldsymbol{\varepsilon}_{y^{(v)}}$ are the sensor prediction errors of position encoder, velocity encoder, and the visual sensor respectively. Furthermore, $\boldsymbol{\varepsilon}_x$ and $\boldsymbol{\varepsilon}_u$ are the prediction errors for the prior on the state and on the control action.

The prediction error on the state prior $\boldsymbol{\varepsilon}_x = (\boldsymbol{\mu}_x - \hat{\boldsymbol{x}})$ is computed from $\hat{\boldsymbol{x}}$, the prediction of the state, which is a deterministic value. The prediction can be computed via the prediction step of a Kalman filter: an advantage of the (u-AIC) is that an accurate model is not required. In this paper a simple random walk is assumed.

Finally, the information about the target/goal state is encoded in the control bias distribution $p(\boldsymbol{u}|\boldsymbol{x})$. We choose this distribution to be Gaussian as well with a mean of $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$, which is a function that steers the systems toward the target. This then results in $\boldsymbol{\varepsilon}_u = (\boldsymbol{\mu}_u - f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d))$. The function $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)$ can be *any* controller, for instance a proportional (P) controller: $f^*(\boldsymbol{\mu}_x, \boldsymbol{\mu}_d)) = P(\boldsymbol{\mu}_d - \boldsymbol{\mu}_x)$.

2.3 Bayesian Controller

We now consider a Bayesian controller as in Baioumy et al. (2020), whose generative model is defined as

$$p(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \boldsymbol{y}_t, \boldsymbol{u}_t) \propto \underbrace{p(\boldsymbol{x}_t)}_{\text{Prediction prior}} \underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)}_{\text{Transition model}} \underbrace{p(\boldsymbol{y}_t|\boldsymbol{x}_t)}_{\text{Observation model}} \underbrace{p(\boldsymbol{x}_{t+1})}_{\text{Goal prior}}. \tag{5}$$

The prediction prior $p(\boldsymbol{x}_t)$ is defined the same way as in the u-AIC. The transition model $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$ describes how the agent transitions from a state $\boldsymbol{x}_t$ to a future state $\boldsymbol{x}_{t+1}$ by taking an action $\boldsymbol{u}_t$. The observation model $p(\boldsymbol{y}_t|\boldsymbol{x}_t)$ for a given sensor, defines the (noisy) relationship between the states and observation.

The general approach for this paper works for any choice of distributions. However, for the sake of simplicity, we choose a Gaussian transition model $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; f(\boldsymbol{x}_t, \boldsymbol{u}_t), \Sigma_f)$, where $f(\cdot)$ is a transition function and Σ_f is the uncertainty over the transition. This is

equivalent to assuming $q \sim \mathcal{N}(\mathbf{0}, \Sigma_f)$ in eq. (1). Similarly, the observation model is chosen to be Gaussian as $p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; g(\boldsymbol{x}_t), \Sigma_y)$.

The unusual element in this model is that the goal state is specified by adding a goal prior over a future state $\boldsymbol{x}_{t+1}$, which we implicitly predict to reach. The optimal control action leading to this can thus be computed by Bayesian inference. The goal prior is also chosen to be Gaussian $\mathcal{N}(\boldsymbol{x}_{t+1}; \boldsymbol{x}_{\text{goal}}, \Sigma_{\text{goal}})$. The smaller the value for Σ_{goal} is, the more aggressively the controller will act to reach the target.

Solving this model requires computing the target posterior $p(\boldsymbol{x}_t, \boldsymbol{u}_t|y_t)$. Computing $p(\boldsymbol{x}_t|y_t)$ results in filtering and computing $p(\boldsymbol{u}_t|y_t)$ results in control. Since all our distributions are Gaussian, the negative log-likelihood of this model will consist of least-square terms and logarithms. For our model this would lead to four terms since we have four Gaussian factors in eq. (5). To simplify notation, we define $||x_1 - x_2||^2_\Sigma = (x_1 - x_2)^\top \Sigma^{-1}(x_1 - x_2) + \ln \Sigma^{-1}$. The full negative log-likelihood for the controller is then:

$$\begin{aligned} -\mathcal{L} = &||\boldsymbol{x}_{t+1} - f(\boldsymbol{x}_t, u_t)||^2_{\Sigma_f} + ||\boldsymbol{y}_t - g(\boldsymbol{x}_t)||^2_{\Sigma_g} + \\ &||\boldsymbol{x}_{t+1} - \boldsymbol{x}_{goal}||^2_{\Sigma_g} + ||\boldsymbol{x}_t - \hat{x}||^2_{\Sigma_1} = F. \end{aligned} \tag{6}$$

We can estimate the state x_t and control action u_t by minimizing the negative log-likelihood by using any suitable optimization method. It follows that the result will depend on the prediction prior ($\hat{x}$) and on the observed value from the sensors $\boldsymbol{y}_t$. Balancing these two terms is equivalent to the prediction and measurement update steps of a Kalman filter, as proved in Ho and Lee (1964). The future state x_{t+1} is constrained by the goal state x_{goal} (third term in $-\mathcal{L}$). Thus, to minimize the whole expression, the value for the control input u_t needs to make the system evolve, according to the model $f(\boldsymbol{x}_t, u_t)$, towards the goal state.

3. FAULT-TOLERANT CONTROL USING PRECISION LEARNING

This section introduces the first major contribution of this paper: achieving FTC with precision learning. Two techniquess will be presented: point-based precision learning, as in (Baioumy et al., 2021b); and a Bayesian method, which is another novel contribution of this paper.

3.1 Point-mass approaches

The observation matrix Σ_y can be updated via gradient descent on $\mathcal{F}$ as: $\dot{\Sigma}_y = -\kappa_\sigma \frac{\partial \mathcal{F}}{\partial \Sigma_y}$. Such update rule has several practical issues. First, in the present case, it would lead to inverting the possibly high dimensional matrix Σ_y. A work around is to directly update its inverse, the *precision* matrix:

$$\dot{\Sigma}_y^{-1} = -\kappa_\sigma \frac{\partial \mathcal{F}}{\partial \Sigma_y^{-1}}. \tag{7}$$

The second issue is that a Σ_y needs to be positive semi-definite. A solution is to perform a re-parameterization with a strictly positive function such as an exponential.

An alternative method is to set a lower bound on the variance, as done in Bogacz (2017).

3.2 Bayesian approaches for one-dimensional problems

Once again consider a one-dimensional problem where observations y are affected by noise generated by a Gaussian with a known mean C and scalar precision ω, $\mathcal{N}(y; C, \omega)$. Given some observation y, we wish to find the posterior $p(\omega|y)$. To do this, we choose the prior over the precision $p(\omega)$ to be a *gamma distribution* defined as:

$$\Gamma(\omega; a, b) = \frac{b^a}{\Gamma(a)} \omega^{a-1} e^{-\omega b}$$

where a and b are the parameters of the distribution and $\Gamma(a) = (a-1)!$ is a factorial function. For example, $\Gamma(5) = 4! = 24$. Now to compute the posterior, we multiply the prior with the Gaussian likelihood model of $p(y|\omega)$ and obtain the posterior which is also a gamma distribution as shown below.

$$\begin{aligned} p(\omega) = \Gamma(\omega; a, b) &\propto \omega^{a-1} e^{-\omega b} \\ p(\omega|y) \propto p(y|\omega)p(\omega) &\propto \omega^{0.5+a-1} e^{-\omega(b + \frac{(y-C)^2}{2})} \\ p(\omega|y) = \Gamma(\omega; a + \frac{1}{2}, b + &\frac{(y-C)^2}{2}) \end{aligned}$$

The last equation shows a simple update rule to modify the belief over the precision for every observation. In the optimization for the state, the following quantities are used: expected precision $\mathbb{E}[\omega] = a/b$, $\text{Mode}[\omega] = (a-1)/b$ and $\text{Var}[\omega] = a/b^2$. Note that while a gamma distribution was assumed for the precision, as is common in Bayesian inference, this analysis may extend to other distributions.

3.3 Bayesian approaches for n-dimensional problems

Using the gamma distribution is limited to one-dimensional problems. For n-dimensional problems, the Wishart distribution is used. Wishart distribution is parameterized by $\mathbf{x}$: a squared positive definite matrix of size p containing random variables, and $\mathbf{V}$ a symmetric positive define matrix of size p as well. Then if $n \geq k$ the Wishart distribution over $\mathbf{x}$ is given by:

$$p(\mathbf{x}) = \frac{1}{2^{nk/2} |\mathbf{V}|^{n/2} \Gamma_k\left(\frac{n}{2}\right)} |\mathbf{x}|^{(n-k-1)/2} e^{-(1/2) tr\left(\mathbf{V}^{-1}\mathbf{x}\right)}$$

where $|\mathbf{x}|$ is the determinant of $\mathbf{x}$ and Γ_k is the multivariate gamma function:

$$\Gamma_k\left(\frac{n}{2}\right) = \pi^{k(k-1)/4} \prod_{j=1}^{k} \Gamma\left(\frac{n}{2} - \frac{j-1}{2}\right).$$

Using the Wishart distribution as a prior can then be done in the same fashion as the gamma distribution.

4. INTERPRETABILITY OF PRECISION LEARNING VIA BETA RESIDUALS

This section introduces the second novel contribution of this paper: using the *beta residual* for fault-detection.

The previous section discussed performing FTC using precision learning. While this technique comes with the benefit of fault recovery without the residual thresholds, fault detection was never explicitly performed. Indications that a fault has occurred or is occurring may be useful for the user of the fault-tolerant controller. Common alternative fault-tolerant schemes explicitly detect faults to trigger recovery, providing the user with the controller's belief of the presence of faults. While precision learning does not yield outputs for explicit fault detection, the precision (inverse covariance) of the sensor model is estimated online at every time-step. This section introduces ways to combine explicit fault detection and isolation with precision learning, including a novel approach called the 'beta residual'.

4.1 Precision learning in conjunction with other methods

A straightforward way to achieve fault detection explicitly is to use existing FDI methods in conjunction with precision learning. For instance, we can learn a probabilistically robust threshold from data offline (when the system is not operating) similar to approaches in Baioumy et al. (2021c). Then online (when the system is in operation) we compute the residual signal as $\|y-\hat{x}\|$, where y is the measurement of the sensor and $\hat{x}$ is the estimate of the state x. In our Bayesian Controllers it holds that: $\|y-\hat{x}\| = \|y-\mu_x\|$.

Now fault detection is straightforward to perform: the residual $\|y-\hat{x}\|$ is compared against the learned threshold and a fault is detected once the threshold is exceeded. Fault recovery can happen by triggering precision learning.

This gives us two ways to achieve fault-tolerant control. Precision learning with *implicit fault detection* means that we do not monitor the system and perform precision learning during the whole operation. This means there is no explicit fault detection. Additionally, precision learning with *explicit fault detection* refers to monitoring the systems and only triggering precision learning once a fault is detected. The two methods are compared in the results section and the mean squared error is given in Table 1.

4.2 The beta residual

As discussed, precision learning can work in conjunction with existing fault detection schemes. The controller uses the root mean sqaure error (RMSE) of a Kalman filter $\|y-\hat{x}\|$ as a residual signal and compares it to a learned threshold. We will now consider a different residual based on Bayesian inference, the *beta residual*.

The previous section discusses precision learning as Bayesian inference where instead of computing a point-estimate of the precision, we compute a full distribution. An appropriate choice in this case would be the Gamma distribution, as this is the conjugate prior precision to the Gaussian likelihood noise model with known mean and unknown precision (The Wishart distribution models the precision for the same reason in multi-dimensional cases). This Gamma distribution is parameterised by β and α. Since these parameters encode information about the degree to which a sensor is faulty, one of these parameters may be used as a residual signal. Given that α quantifies the number of observations, it does not correlate with the fault-status of a sensor on its own. Information about the sensor faultiness will be rather encoded in β. The parameter β is inversely proportional to the expected precision:

$$\mathbb{E}[\Gamma(\alpha,\beta)] = \frac{\alpha}{\beta} \tag{8}$$

Now we will discuss how to extract interpretable outputs from the Gamma precision parameter β. Using sensor data labelled with the presence of faults, supervised learning algorithms can be applied to map from β to a probability of a fault. For the sake of demonstration, we use a simple linear classification algorithm, logistic regression Walker and Duncan (1967).

Logistic regression assumes a linear relationship between the log-odds $\ln\frac{p}{1-p}$ and the predictor x: $\ln\frac{p}{1-p} = bx + b_0$, where b and b_0 are model parameters selected to minimize the mean squared error of the classifier. By simple algebraic manipulation, $p = \frac{1}{1+e^{-(bx+b_0)}}$. The sigmoid function, $f(x) = \frac{1}{1+e^{-x}}$ constrains the output of the classifier to always be between 0 and 1. These output prediction are often interpreted as probabilities Walker and Duncan (1967). A logistic regression model can easily be converted to a binary classifier by introducing a threshold on the model's output probability, such as all outputs $p > 0.6$ predicting a fault.

In summary, although precision learning does not explicitly determine the probability of a fault, simulated faults can generate the training data necessary for developing a classifier that uses learned precision statistics to return the probability of a fault. Combined with thresholds on the probability of a fault, the classifier can notify the user of a fault. While precision learning can be triggered using alternative FDI techniques, fault-tolerant control can also be achieved by always using precision learning and extracting interpretable sensor fault detections using beta residuals. This comes with the benefit of having fault classifications align with the controller's belief of the faultiness of sensors: When the sensor is believed to be operating correctly, the inferred precision is high (low variance), while when the sensor is believed to have a fault, the inferred precision will be low (high variance).

5. RESULTS

This section summarizes the results of experimentation in this paper. First, the fault tolerance of the u-AIC with point-mass precision learning is demonstrated. Then, the Bayesian Controller with precision learning as a distribution is shown to outperform alternative model-based FT techniques. Finally, a technique for extracting interpretable outputs from the Bayesian Controller classifying a fault is explained and applied.

5.1 u-AIC with point-mass precision learning

In the first experiment, we apply the methods in Sec. 2.2 on a 2-DOF robotic manipulator. The manipulator has 3 sensor: a position encoder, a velocity encoder and a visual sensor. We use point-based precision learning on the u-AIC.

We test two scenarios: a) precision learning at all times thus performing no explicit fault detection, b) precision learning only when a fault is detected. The first case has no explicit fault detection. In the second case, we use an existing FDI method based on the probabilistically-robust thresholds, then when a fault is detected, precision learning is triggered.

In the simulations, the sensors are injected with zero-mean Gaussian noise. The standard deviation of the noise for encoders is set to $\sigma_q = \sigma_{\dot{q}} = 0.001$, while the one for the camera is set to $\sigma_v = 0.01$. The camera is also affected by barrel distortion with coefficients $K_1 = -1.5e^{-3}$, $K_2 = 5e^{-6}$, $K_3 = 0$ (values are similar to work from Marshall and Lipkin (2014); Piepmeier et al. (2004)). The agent starts in configuration $\boldsymbol{x}_0$, then moves to the targets $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. At $t = 8s$ a fault is injected. The encoder fault is such that the output related to the first joint freezes.

Without precision updates, the system is not able to reach the target state after the occurrence of the fault. Instead, the robot arm reaches a different configuration to minimise the free-energy, which is built fusing the sensor information from the (faulty) encoders and the (healthy) camera. However, when the faulty encoder is adjusted using precision learning, the agent is able to reach the final configuration. The Mean Squared Error (MSE) between the belief and the true position ($\boldsymbol{\mu}_x - \boldsymbol{x}$) is computed on a sample of test runs and reported in the Table 1. The results are reported for both the joint whose encoder is faulty, and joints with healthy encoders.

	Joints with encoder fault	Joints without encoder fault
No fault-tolerance	0.0036	0.0020
PL with implicit fault detection	5.422 e-5	4.527 e-5
PL with explicit fault detection	6.097 e-5	4.134 e-5

Table 1. Mean Squared Error (MSE) for different methods of fault-tolerant control. PL indicates precision learning

5.2 The Bayesian controller with precision learning

In the second experiment, we compare precision learning on the Bayesian Controller to existing methods. Extensive simulations are performed on a cruise control to showcase the performance. The agent has two identical sensors, one of which suffers form a fault. We consider 3 types of fault: freezing, sensory injection and sensor drift. The agent is tasked with tracking 3 types of trajectories: constant, linear ramp and sinusoidal.

Benchmarks To evaluate the performance of precision learning, two alternative techniques are applied as benchmarks: (Easy and Fast FDI) EF-FDI Berriri et al. (2012) and (State estimation residual FT) SER FT Kommuri et al. (2016).

In EF-FDI, the detection procedure takes advantage of temporal redundancy in sensor observations, and is chosen here as it is representative of methods that are of a low complexity and fast computationally.

$$r_k = |y_t - 2y_{t-1} + y_{t-2}|,\ R_k = r_k + r_{k-1} + r_{k-2} \quad (9)$$

where y_t is the observation. If the statistic R_k exceeds a threshold δ_{EF}, the sensor output is ignored for the expected duration of the fault.

Similarly, SER FT Kommuri et al. (2016), determines faults using analytical and sensor redundancy. A threshold δSER on the residual for state estimation $|y_t - \hat{x}_t|$ is computed. If this threshold is exceeded, the sensor is deemed faulty and the output is ignored.

5.3 Cruise Control System

The Bayesian controller is tasked with tracking a trajectory for a vehicle cruise control problem given by the following system equations:

$$\boldsymbol{x}_{t+1} = (1 - \frac{b}{m}dt)\boldsymbol{x}_t + \frac{dt}{m}\boldsymbol{u}_t + \boldsymbol{q} \quad (10)$$

where $\boldsymbol{x}_t$ is the velocity of the car at time t, b is the drag coefficient, m is the mass of the car, and $\boldsymbol{u}$ is the control action and $\boldsymbol{q}$ is the process noise. Finally, dt is the timestep size. The discrete observation model when no sensor faults are present is given by

$$\boldsymbol{y}_t = \boldsymbol{x}_t + \boldsymbol{\eta} \quad (11)$$

where $\boldsymbol{\eta}$ is the observation noise. The exact parameter values used are provided in Table 2.

Symbol	Name	Value
b	drag coefficient	$5Ns/m$
m	car mass	100 kg
dt	timestep size	0.02 s
$\boldsymbol{q}$	process noise	$\mathcal{N}(0, 0.001m^2s^{-2})$
$\boldsymbol{\eta}$	observation noise	$\mathcal{N}(0, 1m^2s^{-2})$

Table 2. System parameters for the vehicle cruise control problem.

Tracking Performance Applied to the same control problem, each FT technique is tested on 9 different "scenarios" for each FT technique (EF-FDI, No FT, Precision Learning (PL), and SER FT). The scenarios are described below.

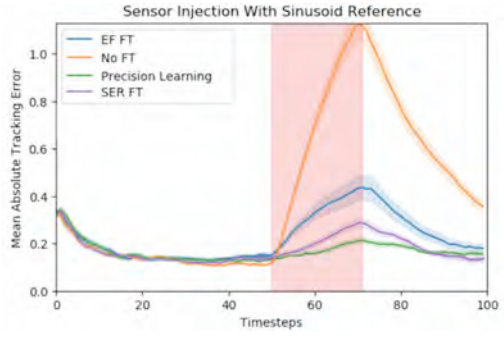

Fig. 1. Mean absolute error in state, where reference is a sinusoid, and the sensor injection is present during the red highlighted period.

To create a quantifiable summary of the relative merit of the different techniques across a range of scenarios, the following procedure was followed. The Bayesian controller is

tasked with a cruise control problem across 900 simulations for each FT technique (EF-FDI, No FT, Precision Learning (PL), and SER FT). The simulations are distributed evenly across sensor fault type (sensor freeze, sensor drift, and sensor injection) and target state trajectory (constant, linear ramp, sinusoidal). Each unique combination of sensor fault type and target state trajectory will be referred to as a 'scenario'. The mean root mean tracking error across 100 experiments for each scenario is compared using heteroskedastic t-tests for difference of means, summarized in Table 3. Each element in the table shows the number of t-tests where the top FT technique's out-performance of the left technique was statistically significant minus the number of t-tests where the reverse was true. Tests were conducted across all three sensor failure types and all three reference paths, for a maximum of 9 in each cell (except for the "Total Merit"). The total Merit entries are the sum of the column, showing total outperformances minus underperformances for a given technique, for a maximum total of 27 (9 per alternative technique). The ensemble mean absolute error across a single scenario, the sensor injection with sinusoid reference is shown in Figure 1.

FT Technique	EF-FDI	No FT	PL	SER FT
EF-FDI	0	-6	8	8
No FT	6	0	8	8
PL	-8	-8	0	-5
SER FT	-8	-8	5	0
Total Merit	-10	-22	21	11

Table 3. Comparison for different FTC methods with the relative merit reported.

5.4 Interpretability using Bayesian residual

In the third experiment, we compare the novel 'beta residual' to the more common State Estimation Residual (equivalent to the RMSE of the Kalman filter).

In the procedure, 1000 sensor faults were generated, with a mean-time-to-fault of 2.8 s, and mean-time-to-recovery of 0.4 s for a total of approximately 160,000 simulation timesteps. The occurrences of the faults and recoveries are distributed assuming a constant probability of failure. The faults are therefore Poisson-distributed in their time-to-occurence and in length. When a fault is triggered in the simulation, a random selection of one of the 3 faults is selected. The model assumes that only one fault at a time occurs with the same sensor. Faults are allowed to occur simultaneously at both of the sensors (although this is not common). The learned precision parameters and the SER residual are saved together with the binary fault truth values. The results are split into 80% training set and 20% test set. Logistic regression is trained on the training set and evaluated on the test set.

The test set Receiver Operating Characteristic (ROC) is shown in Table 4. The ROC-AUC is the are under the ROC. It is a common measure of the performance of binary classifiers that return probabilities rather than classes. For the controller with the given FT strategy on the left, the model input is used to predict the 0-1 faultiness of the sensor with logistic regression. The ROC-AUC, an assessment of the classification power of the model.

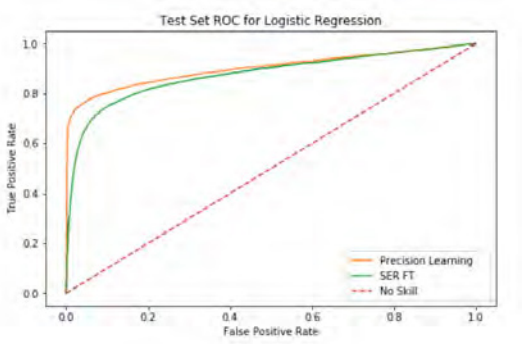

Fig. 2. The Receiver Operating Characteristic (ROC) shows the trade-off between true positive rate and false positive rate for all possible thresholds. The diagonal line in red is the ROC of a hypothetical model that guesses randomly. EF-FFT is not shown for comparison because its residual for fault detection corresponds with the onset of faults, not the presence of faults, rendering it unsuitable for classification using logistic regression.

A brief explanation of the ROC as a metric follows. In order to evaluate a classifier, accuracy is an insufficient metric, due to imbalances in the classes (there are more non-faulty cases than faulty cases). In the case of binary classification, both the true positive rate and the false positive rate should be considered. However, in order to evaluate the logistic regression model as a classifier, a probability threshold for classification must be selected. The ROC evaluates the model at all possible thresholds: Each point on the curve represents a threshold with an associated false positive rate (x-axis) and true positive rate (y-axis). The area under the curve measures the fit of the logistic regression model: The closer to 1, the better the classifier. A random guesser has an ROC-AUC (Area Under Curve) of 0.5, labelled as "No Skill" in Figure 2.

These results show that the parameter beta can be effectively used as a residual signal. A stochastic threshold was learned using logistic regression, however, other supervised models may be used.

FT Technique	Input	AUC
SER FT	$\|\|y - \hat{x}\|\|$	0.873
Precision Learning	β	0.907

Table 4. ROC-AUC for fault detection with different residuals.

6. DISCUSSION AND CONCLUSION

This paper compares a set of fault-tolerant controllers based on Bayesian inference and on a novel approach: precision learning. All controllers perform state-estimation, control, fault detection and recovery as a single inference procedure. Sensor faults are modelled as changes in the covariance/precision of the sensor model. Thus by learning the precision online, the agent can achieve fault-tolerant control. In addition, by modeling the control problem as Bayesian inference, standard methods using non-Gaussian distributions and non-linear systems can be leveraged. The results show how controllers based on precision learning outperform existing approaches on a variety of tasks. This

includes different types of fault (sensor freeze, sensor injection and drift), while tracking different trajectories (ramp, step-response, and sinusoid).

A key limitation of this approach is the lack of an explicit fault detection step. To mitigate this issue, one could use standard fault detection approaches in conjunction with precision learning. We demonstrate that this yield satisfactory performance. Additionally, we introduce a novel *beta residual*, which shows improved performance compared the commonly used residuals such as the root mean square error (RMSE) of a Kalman filter. Additionally, it is pointed out how the beta residual can be interpreted as an hyperparameter which represents the trust in a sensor at a specific time instant. Future work will further investigate the use of the beta residual for fault-detection, including its fault detectability properties.

REFERENCES

Baioumy, M., Duckworth, P., Lacerda, B., and Hawes, N. (2021a). Active inference for integrated state-estimation, control, and learning. In *Proc of IEEE Int. Conf. on robotics and automation (ICRA)*.

Baioumy, M., Mattamala, M., and Hawes, N. (2020). Variational inference for predictive and reactive controllers. In *ICRA 2020 Workshop on New advances in Brain-inspired Perception, Interaction and Learning*. Paris, France.

Baioumy, M., Pezzato, C., Corbato, C.H., Hawes, N., and Ferrari, R. (2021b). Towards stochastic fault-tolerant control usingprecision learning and active inference. In *Int. Workshop on Active Inference*.

Baioumy, M., Pezzato, C., Ferrari, R., Corbato, C.H., and Hawes, N. (2021c). Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference. In *European Control Conf. (ECC)*.

Berriri, H., Naouar, M.W., and Slama-Belkhodja, I. (2012). Easy and fast sensor fault detection and isolation algorithm for electrical drives. *IEEE Trans. on Power Electronics*, 27(2), 490–499. doi:10.1109/TPEL.2011.2140333.

Blanke, M., Frei, W.C., Kraus, F., Patton, J.R., and Staroswiecki, M. (2000). What is fault-tolerant control? *IFAC Proceedings Volumes*, 33(11), 41–52.

Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76, 198–211.

Buckley, C.L., Kim, C.S., McGregor, S., and Seth, A.K. (2017). The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81, 55–79.

Cai, B., Huang, L., and Xie, M. (2017). Bayesian networks in fault diagnosis. *IEEE Trans. Ind. Inf.*, 13(5), 2227–2240.

Chen, J. and Patton, R.J. (1999). *Robust model-based fault diagnosis for dynamic systems*. Springer Science & Business Media, LLC.

Codetta-Raiteri, D. and Portinale, L. (2015). Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. *IEEE Trans. Syst. Man Cybern.*, 45(1), 13–24.

Fox, C.W. and Roberts, S.J. (2012). A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2), 85–95.

Friston, K.J. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2), 27–138.

Friston, K., Stephan, K., Li, B., and Daunizeau, J. (2010). Generalised filtering. *Mathematical Problems in Engineering*.

Gao, Z., Cecati, C., and Ding, S.X. (2015). A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Trans. on industrial electronics*, 62(6), 3757–3767.

Ho, Y. and Lee, R. (1964). A bayesian approach to problems in stochastic estimation and control. *IEEE Trans. on automatic control*, 9(4), 333–339.

Kommuri, S.K., Defoort, M., Karimi, H.R., and Veluvolu, K.C. (2016). A robust observer-based sensor fault-tolerant control for pmsm in electric vehicles. *IEEE Trans. on Industrial Electronics*, 63(12), 7671–7681.

Lanillos, P. and Cheng, G. (2018). Adaptive robot body learning and estimation through predictive coding. In *IROS*.

Lerner, U., Parr, R., Koller, D., Biswas, G., and Others (2000). Bayesian fault detection and diagnosis in dynamic systems. In *AAAI/IAAI*, 531–537. aaai.org.

Marshall, M. and Lipkin, H. (2014). Kalman filtering visual servoing control law. In *IEEE Procs. of Int. Conf. on Mechatronics and Automation*.

Murphy, K.P. (2012). *Machine learning: a probabilistic perspective*. MIT Press.

Pezzato, C., Ferrari, R., and Corbato, C.H. (2020). A novel adaptive controller for robot manipulators based on active inference. *IEEE Robotics and Automation Letters*, 5(2), 2973–2980.

Pezzato, C., Baioumy, M., Corbato, C.H., Hawes, N., Wisse, M., and Ferrari, R. (2020). Active inference for fault tolerant control of robot manipulators with sensory faults. In Springer (ed.), *1st Int. Workshop on Active Inference, ECML PKDD*, volume 1326 of *Communications in Computer and Information Science*.

Piepmeier, J., McMurray, G., and Lipkin, H. (2004). Uncalibrated dynamic visual servoing. In *IEEE Trans. on Robotics and Automation*, volume 20, 143–147.

Sun, W., Paiva, A.R.C., Xu, P., Sundaram, A., and Braatz, R.D. (2020). Fault detection and identification using Bayesian recurrent neural networks. *Comput. Chem. Eng.*, 141, 106991.

Verma, V., Gordon, G., Simmons, R., and Thrun, S. (2004). Real-time fault diagnosis [robot fault diagnosis]. *IEEE Robot. Autom. Mag.*, 11(2), 56–66.

Walker, S.H. and Duncan, D.B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2), 167–179.