Int. J. Appl. Math. Comput. Sci., 2015, Vol. 25, No. 1, 133–148
DOI: 10.1515/amcs-2015-0010

amcs

FAULT TOLERANT CONTROL USING GAUSSIAN PROCESSES AND MODEL PREDICTIVE CONTROL

XIAOKE YANG [a,*], JAN M. MACIEJOWSKI [a]

[a] Department of Engineering
University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, UK
email: das.xiaoke@hotmail.com, jmm1@cam.ac.uk

Essential ingredients for fault-tolerant control are the ability to represent system behaviour following the occurrence of a fault, and the ability to exploit this representation for deciding control actions. Gaussian processes seem to be very promising candidates for the first of these, and model predictive control has a proven capability for the second. We therefore propose to use the two together to obtain fault-tolerant control functionality. Our proposal is illustrated by several reasonably realistic examples drawn from flight control.

Keywords: fault-tolerant control, Gaussian process, model predictive control, aircraft control, probabilistic modelling.

1. Introduction

The demand for fault-tolerant control (FTC) comes from safety requirements and from economics. In safety-critical applications, there is always some requirement for a safe back-up in case the nominal system fails. A fault-tolerant control system may be a more effective way of providing such a back-up than just physical duplication or triplication of equipment. A topical set of applications arises in automotive control. In this sector there is a rapidly increasing use of control systems for safety-critical functions. The best-known example is that of anti-lock braking systems (ABSs), for which legislation requires some fault-tolerant capability (Anon, 2015). Vehicle manufacturers are currently developing and installing systems for anti-collision braking, automatic steering or automatic lane-changing and overtaking, and one can anticipate that these will also be required to have some fault-tolerant capability.

Another major driver for fault-tolerant control comes from aircraft, including unmanned air vehicles (UAVs). UAVs have many potential non-military applications, some of them requiring operation over densely-inhabited areas such as cities—for example, traffic monitoring. Pilots can overcome many minor faults on an aircraft, and sometimes even major ones. A concern with UAVs operating over inhabited areas is that a relatively minor problem which could be handled easily by a pilot should not cause a UAV to crash. Safe autonomous operation is a key requirement, and fault-tolerant flight control is an essential enabler of that capability.

Economic considerations arise because the costs of lost production due to a fault can be enormous, and the costs of unnecessary energy consumption due to a fault can also be very high. There is therefore a strong incentive to keep manufacturing and production plant operating despite faults—possibly with reduced quality or reduced volume until maintenance can be performed.

Space programmes have requirements for fault-tolerance which arise from their extremely high cost. Although human safety may not be involved (in unmanned programmes), the cost of failure, in both financial and political terms, is so high that mission-critical functions must be fault-tolerant—and this includes control systems. The European Space Agency's *Aurora* programme, for example, aims to explore the outer solar system and beyond, including missions to Mars, culminating in a manned mission. Technologies to support autonomous operation are emphasised in the *Aurora* programme, mainly because distances to Mars and beyond are so large that communications with Earth suffer major delays, so that Earth-originated solutions to unexpected problems would take too long to implement.

Model predictive control (MPC) offers a very

*Corresponding author

attractive and effective framework for fault-tolerant control, providing that one can update the internal model used by MPC when a fault occurs. Detecting and identifying the fault, and representing it in a model-based framework, is a critical problem for FTC. In recent years some machine-learning methods have been shown to be remarkably effective at discovering model-based representations from data. In particular, *Gaussian processes* (Rasmussen and Williams, 2006) have been investigated for control of complex systems, and appear to be very promising in that context (Deisenroth and Rasmussen, 2011).

In this paper we propose to combine the use of Gaussian processes with MPC for the purpose of providing fault-tolerant control functionality. The main body of this paper consists of the following parts. Section 2 gives the rationale of using MPC as a framework for FTC, specifically with Gaussian process models. Section 3 reviews background knowledge on Gaussian processes. Section 4 describes in detail the formulation of MPC with Gaussian processes. Sections 5 and 6 then present two control scenarios and numerical simulation results, followed by a discussion in Section 7 and conclusions in Section 8.

2. MPC as a framework for FTC

Figure 1 shows the components of a typical MPC feedback system (Maciejowski, 2002). At the top, labelled "Setup", are the components that define the controller: a *model* which will be used to generate the predictions needed by the optimiser, an *objective function* which will be minimised by the optimiser, and *constraints* which need to be satisfied by the solution generated by the optimiser. Below this, the block labelled "Online tasks" shows what needs to be executed at each sampling/update interval:

1. Measurements are obtained from sensors located on the plant which is being controlled, and used by an observer to estimate the plant state vector (not always necessary).

2. A specific optimal control problem is defined (which depends on the current (estimated) plant state, using all the ingredients specified in "Setup".

3. This optimal control problem is solved by an optimisation algorithm. Usually this computes a whole sequence of future input commands. Only the first of these is applied to the plant being controlled. Note that the optimisation problem is time-critical; it needs to be solved within the control-update interval.

On the right of Fig. 1, labelled "Real world", is the plant being controlled. Disturbances acting on this plant are shown explicitly, in order to emphasise that the plant never behaves exactly as predicted by the model.

The attraction of MPC for fault-tolerant control is its flexibility. Suppose, for example, that an actuator becomes jammed at a particular value. If this failure is detected, then it can be represented in the MPC setup by constraining the rate of change of that actuator to be zero. If, in addition, the value at which the actuator is jammed is known, then an equality constraint can be added to the problem setup representing this. The optimiser then has the task of discovering an alternative control action—of course it can only succeed if there is at least one alternative actuator that it can use. This capability of representing faults by re-defining constraints is already a valuable feature for fault-tolerant control, since there is an increasing availability of self-validating actuators, such as self-validating valves in the process industries and self-validating control surfaces in aircraft.

A fault may be more complicated than a jammed actuator. For example, a hydraulic leak may change the time constant of an actuator's servo response. Or a heat exchanger may become less efficient due to fouling. Or an aerodynamic lifting surface may become damaged due to fatigue, or become less efficient due to icing. Faults of this kind can be represented by changing the model used by MPC. The fact that the model has an explicit representation in the MPC structure makes it possible to do this. For MPC it does not matter whether the model is a detailed physics-based first-principles model or a black-box model obtained from plant data—but it may be easier to represent a particular fault in a more detailed model.

Further flexibility is offered by the possibility of changing the objective function that is optimised. This is very important for fault-tolerant control. If a severe fault occurs, it is probably not reasonable to try to achieve the same objectives that were achieved when the plant was healthy. Admittedly it is not easy to exploit this aspect of MPC's flexibility, for two reasons: how to change real-world objectives in the event of a fault is an unexplored topic, and then how to change a mathematical objective function to reflect real-world objectives is not straightforward. Nevertheless, the flexibility of changing objectives is a desirable attribute of MPC, even if we do not yet understand how to exploit it properly.

For these reasons, MPC has been advocated as a suitable framework for fault-tolerant control in several publications (Maciejowski, 1997; 1998; Huzmezan and Maciejowski, 1999; Maciejowski and Jones, 2003; Joosten and Maciejowski, 2009). However, much of the potential of MPC depends on being able to represent the effects of the faults, either by changing the constraints, or by changing the model, or both. In particular, finding a suitable new model sufficiently quickly is the major problem faced by fault-tolerant control. This motivates

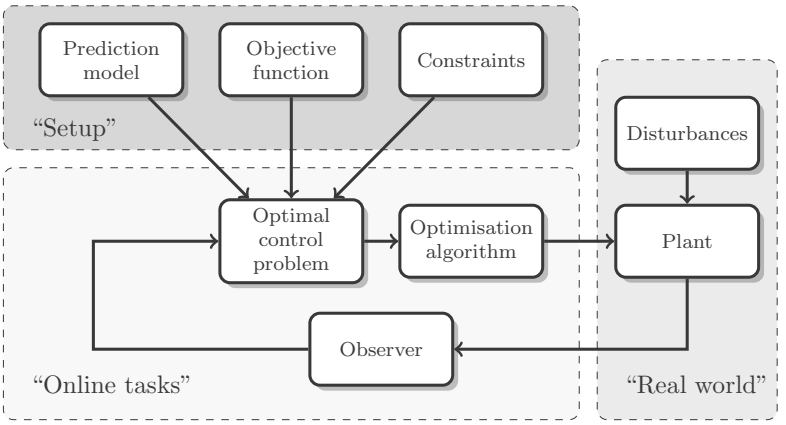


Fig. 1. Components of a model predictive control feedback system.

our interest in using Gaussian processes in the following sections.

Here we add a little more detail about the MPC formulation. A discrete-time dynamic system can be described by the model

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^p$, $\mathbf{u} \in \mathbb{R}^q$ are the states and inputs, respectively, $\mathbf{f}(\cdot,\cdot)$ is a possibly nonlinear vector function, and $\mathbf{w}$ represents a disturbance—or any discrepancy between the model and the actual behaviour. Given a measured or estimated state $\mathbf{x}_k$ at time k, and a set of hypothesised future inputs $\hat{\mathbf{u}}_{k+i|k}$ for $i = 0, 1, \ldots, N-1$, the model can be iterated to give a set of predicted states $\hat{\mathbf{x}}_{k+i|k}$ for $i = 1, 2, \ldots, N$, making some assumptions about the sequence of future disturbances (often assuming that $\mathbf{w}_{k+i} = 0$). The control objectives are then represented by the minimisation of an objective function, most commonly a quadratic function such as (for the case of controlling to an equilibrium $(\mathbf{x}_\infty, \mathbf{u}_\infty)$)

$$J(\mathbf{x}_k, \mathcal{U}_k) = \sum_{i=0}^{N-1} \left[\|\hat{\mathbf{x}}_{k+i|k} - \mathbf{x}_\infty\|^2_{\mathbf{Q}} + \|\hat{\mathbf{u}}_{k+i|k} - \mathbf{u}_\infty\|^2_{\mathbf{R}} \right], \tag{2}$$

where $\mathcal{U}_k = [\hat{\mathbf{u}}_{k|k}, \ldots, \hat{\mathbf{u}}_{k+N-1|k}]$, $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$, together with some state constraints

$$\hat{\mathbf{x}}_{k+i|k} \in \mathbb{X}. \tag{3}$$

Capabilities of the actuators (limits, slew rates, etc.) are represented by some input constraints

$$\hat{\mathbf{u}}_{k+i|k} \in \mathbb{U}. \tag{4}$$

Frequently the objective and/or input constraints are also expressed in terms of control input moves, $\Delta u_k = u_k - u_{k-1}$.

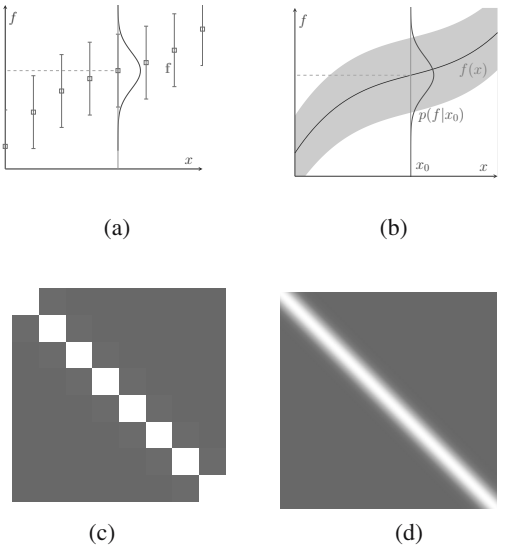
The control input is then found by solving the finite-dimensional constrained optimisation problem

$$\mathcal{U}_k^* = \arg\min_{\mathcal{U}_k} J(\mathbf{x}_k, \mathcal{U}_k) \tag{5}$$

numerically, subject to the constraints (1), (3) and (4). The first element of the sequence $\mathcal{U}_k^*$, namely, $\hat{\mathbf{u}}_{k|k}^*$, is then applied as the input to the plant. This process is repeated at the next step $k+1$, beginning with a new measurement or estimate $\mathbf{x}_{k+1}$. The optimisation problem solved at each step is an open-loop problem, but the repeated application of this process results in closed-loop feedback control, since the problem solved at step k depends on measurements taken at step k. Note that, with the MPC ingredients as defined above, we have defined a time-invariant but nonlinear control law $\kappa(\mathbf{x}_k)$, such that $\hat{\mathbf{u}}_{k|k}^* = \kappa(\mathbf{x}_k)$.

Many variations of this formulation exist, most common being the addition of a separate cost and/or constraint involving the terminal state $\hat{\mathbf{x}}_{k+N|k}$, in order to ensure closed-loop stability. Since the optimisation problem must be solved online in a limited time, it is strongly preferable to choose the details of the formulation such that a convex optimisation problem results. Choosing a linear model in (1), a convex function in (2), and convex constraints in (3) and (4) will yield a convex problem. In particular, choosing the objective function to be quadratic and the constraints to be linear inequalities gives a quadratic programming (QP) problem which can be solved very quickly, if need be using specialised hardware (Hartley *et al.*, 2012; 2014).

Other variations of the MPC formulation include optimisation over feedback policies rather than over open-loop predictions, tightening constraints to allow for known levels of model uncertainties, and 'tube' formulations in which MPC is used to plan a nominal trajectory and conventional feedback is employed to



(a) (b)

(c) (d)

Fig. 2. From a multivariate Gaussian distribution to a Gaussian process: (a) illustrates a multivariate Gaussian distribution over a vector $\mathbf{f}$ of 8 elements, with the error bar indicating the $\pm 2\sigma$ region, (c) depicts its covariance matrix. When the dimension of the vector $\mathbf{f}$ increases to fill the real axis, this multivariate Gaussian distribution approaches to a distribution over functions $f : \mathbb{R} \mapsto \mathbb{R}$, as in the Gaussian process (b), where the solid line indicates the mean function $m : \mathbb{R} \mapsto \mathbb{R}$ of the distribution and the shaded area indicates the $\pm 2\sigma$ uncertainty band. The covariance matrix in (c) becomes refined and approximates a mapping: $k : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$, which is the covariance function (d).

keep the actual system trajectory within some predictable distance from the nominal one. All of these variations are aimed at making MPC more robust to modelling errors and other uncertainties, but they usually involve increased computational complexity (Rawlings and Mayne, 2009).

3. Gaussian processes

A Gaussian process (GP) is a distribution over functions and can be viewed as a generalised multivariate Gaussian distribution over a vector of infinite length. The intuition behind this generalisation is illustrated in Fig. 2, where a Gaussian distribution over vector $\mathbf{f} \in \mathbb{R}^n$, $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, is transformed into a Gaussian process by making vector $\mathbf{f}$ infinitely dense and long, when vector $\mathbf{f}$ essentially becomes a function $f : \mathbb{R} \mapsto \mathbb{R}$. A GP then describes the distribution over such a function f. Formally, a Gaussian process is defined as a collection of random variables, any finite number of which have a consistent joint Gaussian distribution (Åström, 1970; Rasmussen and Williams, 2006). It is worth mentioning that the classical Gaussian process is often defined over the index set of time, leading to a time process. The index set for the Gaussian process hereinafter is generalised to the D-dimensional vector space $\mathbb{R}^D$, which endows the GP with the capability of describing functions defined on a larger space.

Similar to a multivariate Gaussian distribution $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ which is characterised by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$, a Gaussian process $f \sim \mathcal{GP}\left(m(\cdot),\ k(\cdot,\cdot)\right)$ is fully specified by a mean function $m(\cdot)$ and a covariance function $k(\cdot,\cdot)$, defined as

$$m(\mathbf{x}) \triangleq \mathbb{E}_f[f(\mathbf{x})], \tag{6}$$

$$\begin{aligned} k(\mathbf{x},\mathbf{x}') &\triangleq \operatorname{cov}_f[f(\mathbf{x}), f(\mathbf{x}')] \\ &= \mathbb{E}_f\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f(\mathbf{x}') - m(\mathbf{x}')\right)\right], \end{aligned} \tag{7}$$

where $\mathbf{x},\ \mathbf{x}' \in \mathbb{R}^D$ are two arbitrary points. The mean function gives an 'average' shape of the function and the covariance function specifies covariance between any two function values, computed from the corresponding inputs in the index set.

Dynamic system modelling and identification is often posed as a regression problem. The following gives essential background knowledge for regression in a Bayesian framework using Gaussian processes.

3.1. Gaussian process regression.

Gaussian process regression refers to the inference of the underlying function $f(\cdot)$ of a noisy static map

$$y = f(\mathbf{x}) + \epsilon, \tag{8}$$

given a set of input vectors $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \dots & \mathbf{x}_n\end{bmatrix}^\top \in \mathbb{R}^{D\times n}$, output observations $\mathbf{y} = \begin{bmatrix}y_1 & \dots & y_n\end{bmatrix}^\top \in \mathbb{R}^{1\times n}$ and a GP model for $f(\cdot)$. ϵ is a white Gaussian noise term, i.e., $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. In a Bayesian setting, inference of $f(\cdot)$ can be formulated as

$$p(f|\mathbf{X},\mathbf{y},\boldsymbol{\theta}) = \frac{p(\mathbf{y}|f,\mathbf{X},\boldsymbol{\theta})p(f|\boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta})}, \tag{9}$$

where $p(f|\boldsymbol{\theta})$ is the Gaussian process prior and $p(f|\mathbf{X},\mathbf{y},\boldsymbol{\theta})$ is the Gaussian process posterior. $\boldsymbol{\theta}$ is a vector of distribution parameters called hyper-parameters, which will be covered later. A common choice of the GP prior is a zero mean function

$$m(\mathbf{x}) \equiv 0 \tag{10}$$

with a squared-exponential (SE) covariance function

$$k(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^\top \boldsymbol{\Lambda}^{-1}(\mathbf{x}-\mathbf{x}')\right], \tag{11}$$

where $\boldsymbol{\Lambda} = \operatorname{diag}\left(\begin{bmatrix}\lambda_1^2 & \dots & \lambda_D^2\end{bmatrix}\right)$. Then the covariance between the corresponding outputs y and y' can be obtained by appending (11) with $\delta_{\mathbf{x}\mathbf{x}'}\sigma_n^2$ to account for the noise term ϵ in (8), i.e., $\operatorname{cov}[y, y'] = k(\mathbf{x},\mathbf{x}') + \delta_{\mathbf{x}\mathbf{x}'}\sigma_n^2$,

where $\delta_{\mathbf{x}\mathbf{x}'}$ denotes the Kronecker delta symbol defined as $\delta_{\mathbf{x}\mathbf{x}'} = 1$ if $\mathbf{x}$ is the same input point as $\mathbf{x}'$ and 0 otherwise.

The posterior distribution $p(f|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ from (9) can be shown to be another GP (Hall *et al.*, 2012),

$$f|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta} \sim \mathcal{GP}(m_+, k_+), \tag{12}$$

with

$$m_+(\mathbf{x}) = m(\mathbf{x}) + k(\mathbf{x}, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} (\mathbf{y} - m(\mathbf{X})), \tag{13}$$

$$k_+(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} k(\mathbf{X}, \mathbf{x}'), \tag{14}$$

where $m(\mathbf{X}) = \begin{bmatrix} m(\mathbf{x}_1) & \dots & m(\mathbf{x}_n) \end{bmatrix}^\top \in \mathbb{R}^{n \times 1}$, $k(\mathbf{x}, \mathbf{X}) = [k(\mathbf{x}, \mathbf{x}_1) \ \dots \ k(\mathbf{x}, \mathbf{x}_n)] \in \mathbb{R}^{1 \times n}$, $k(\mathbf{X}, \mathbf{x}) = k(\mathbf{x}, \mathbf{X})^\top$, $k(\mathbf{X}, \mathbf{X}) = [k(\mathbf{x}_i, \mathbf{x}_j)]_{n \times n} \in \mathbb{R}^{n \times n}$ and $\mathbf{I}$ is an identity matrix with appropriate dimensions. Figure 3 illustrates functions drawn from the above GP prior and the corresponding GP posterior given data observations.

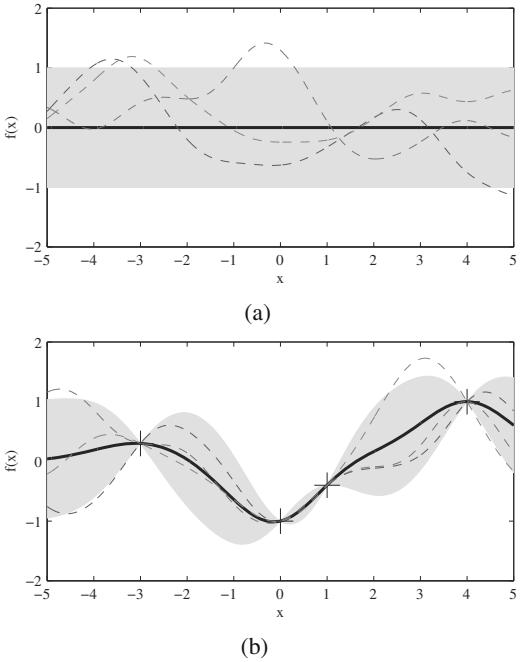


Fig. 3. Functions drawn at random from the GP prior (a) with a zero mean function and an SE covariance function and from the GP posterior (b) given 4 data observations marked by crosses. The solid lines indicate the mean functions and shaded areas give the $\pm 2\sigma$ uncertainty bands.

3.2. Training. As in (9), the GP prior and posterior also depend the hyper-parameters $\boldsymbol{\theta} = \begin{bmatrix} \sigma_f & \lambda_1 & \dots & \lambda_D & \sigma_n \end{bmatrix}^\top$. In a fully Bayesian setup, a hyper-prior $p(\boldsymbol{\theta})$ can be placed upon $\boldsymbol{\theta}$ and the hyper-posterior can be inferred by Bayes' rule

$$p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{X})}; \tag{15}$$

then marginalisation over $\boldsymbol{\theta}$ can be performed in (12) to get the the full GP posterior $p(f|\mathbf{X}, \mathbf{y})$. Practically, a point estimate through maximum likelihood estimation (MLE) is often used to avoid the analytical intractability of the integrals and the high computational demands of approximate sampling methods. This leads to the training of a GP by maximising $\log\ p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, i.e.,

$$\max_{\boldsymbol{\theta}} -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{\boldsymbol{\theta}}| - \frac{1}{2}D\log(2\pi), \tag{16}$$

where $\mathbf{K}_{\boldsymbol{\theta}} = k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}$ and the subscript $\boldsymbol{\theta}$ is to emphasis $\mathbf{K}_{\boldsymbol{\theta}}$ is a function of $\boldsymbol{\theta}$. This MLE treatment is clearly not Bayesian, but it is the result of a tradeoff between computational complexity and performance. The optimisation problem (16) in general is not a convex one, thus local optima will be expected, corresponding to particular interpretations to the data with different levels of likelihood.

3.3. Predictions. Since a GP defines a distribution over functions, then, given a test input $\mathbf{x}_* \in \mathbb{R}^D$, a distribution $p(f(\mathbf{x}_*))$ can be predicted by the GP. For deterministic test input $\mathbf{x}^*$, $p(f(\mathbf{x}_*))$ is simply given by

$$f(\mathbf{x}_*) \sim \mathcal{N}(m_+(\mathbf{x}_*), k_+(\mathbf{x}_*, \mathbf{x}_*)). \tag{17}$$

When the test input contains uncertainty (which is often the case in a dynamic system), say $\mathbf{x}_* \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, a marginalisation over $\mathbf{x}_*$ needs to be performed to give the predictive distribution $p(f(\mathbf{x}_*)|\boldsymbol{\mu}, \boldsymbol{\Sigma})$,

$$p(f(\mathbf{x}_*)|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int p(f(\mathbf{x}_*)|\mathbf{x}_*) p(\mathbf{x}_*)\, \mathrm{d}\mathbf{x}_*. \tag{18}$$

The resulting distribution is normally not a Gaussian, but Gaussian approximation through exact moment matching can be performed, as shown in Fig. 4. When the function $f(\cdot)$ is a vector function, say $\mathbf{f}(\mathbf{x}) : \mathbb{R}^D \mapsto \mathbb{R}^E$, E independent GP models can be used, one for each output dimension. In this case the output values from different dimensions will covary under uncertain test inputs, leading to non-zero cross covariances between each output dimension and non-zero cross covariances between the input and the output, i.e.,

$$p(\mathbf{x}_*, \mathbf{f}(\mathbf{x}_*)|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}_{\mathbf{x}_*, \mathbf{f}_*} \\ \boldsymbol{\Sigma}_{\mathbf{x}_*, \mathbf{f}_*}^\top & \boldsymbol{\Sigma}_* \end{bmatrix}\right). \tag{19}$$

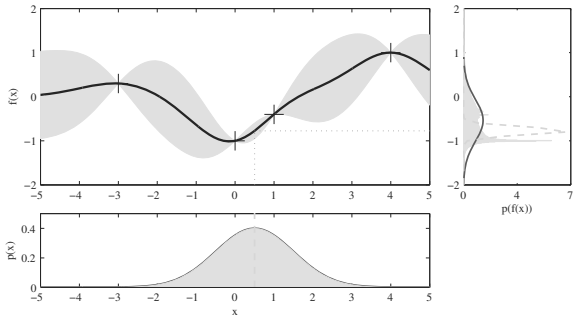


Fig. 4. Prediction with a Gaussian process. The top left figure shows a GP posterior, the bottom figure depicts the distribution of an uncertain test input $\mathbf{x}_*$, the corresponding predicted output $p(f(\mathbf{x}_*))$ is shown in the top right figure: the shaded area represents the distribution obtained from a Monte Carlo approach, the solid line indicates a Gaussian distribution through moment matching and the dashed line represents prediction at a deterministic test input located at the mean of $p(\mathbf{x}_*)$.

Detailed expressions for each element in the mean vector and covariance matrix under the sparse approximation below are listed in Appendix A.

3.4. Gaussian processes for high dimensional problems. One major problem in employing a Gaussian process model is the heavy computational requirement. It is known that, for n-sized training data, predictions at uncertain inputs require $\mathcal{O}(E^2n^2D)$ operations (Deisenroth, 2010), apart from the $\mathcal{O}(n^3)$ complexity of inverting the covariance matrix for which pre-computation is possible. This makes GP models prohibitive for high dimensional (large data size, high input and output dimensions) applications. Two major approaches are used to reduce this computational complexity, namely, sparse approximations and integration of known model information.

3.4.1. Sparse approximations. A sparse GP aims to reduce the n in $\mathcal{O}(E^2n^2D)$, whose computational burden is dominated by matrix operations (e.g., multiplication) involving the $n \times n$ covariance matrix $\mathbf{K}$ as in (13) and (14). Low-rank approximations to matrix $\mathbf{K}$ prove to be an effective solution, among which the fully independent training conditional (FITC) approximation (Snelson and Ghahramani, 2005) is used in this paper. The basic idea of FITC approximation is to introduce a set of $M(M < n)$ inducing or pseudo inputs $\bar{\mathbf{X}} = \begin{bmatrix}\bar{\mathbf{x}}_1 & \bar{\mathbf{x}}_2 & \dots & \bar{\mathbf{x}}_M\end{bmatrix}^\top$ with associated inducing variables or pseudo targets $\bar{\mathbf{f}}$, then integrate out the pseudo targets $\bar{\mathbf{f}}$ and make predictions using the inducing inputs $\bar{\mathbf{X}}$ and the data. With FITC approximation, the computational complexity reduces to $\mathcal{O}(ME)$ for the mean vector and $\mathcal{O}(E^2M^2D)$ for the covariance matrix. Training this approximated sparse GP involves optimisation over the locations of the pseudo-inputs and the hyperparameters, which can be simultaneously done through MLE in (A16).

3.4.2. Integration of explicit model information. Integration of partial model information, on the other hand, aims to reduce the output dimension E and the input dimension D in $\mathcal{O}(E^2n^2D)$. For physical systems, it is common that partial knowledge of the system dynamics is available, such as the relation between position and velocity, or in general states which are related through known relationships. When this is the case, the known relationship can be explicitly used, instead of applying GPs, to reinvent the wheel. For a system of dimension E with F known relations between the states, only $E - F$ GP models need to be trained to make predictions.

For physical systems there are also cases where only a subset of the inputs affect a specific output. This type of model information, although simple, can be effective in reducing the input dimension of GP models (Hall, 2013). It is worth mentioning that a squared exponential (SE) covariance function with an automatic relevance determination form (Rasmussen and Williams, 2006) can also be used to automatically select the relevant input dimensions. Still it is advantageous to incorporate known model information if available, to avoid the extra cost of training a full GP and selecting relevant input channels. In practice, these two types of partial model information are often used together.

4. MPC with Gaussian process models

MPC with Gaussian process models was first proposed by Kocijan *et al.* (2003) and based on that the following gives a detailed formulation when the fact used in fault-tolerant control.

4.1. Prediction model. For dynamic system modelling, Gaussian process state space models can be used. A GP state-space model originates from an ordinary differential equation (ODE) representation of a dynamic system and often applied in a discrete-time setting—state at the next time instant $\mathbf{x}_{k+1}$, state and input at the current time instant $\mathbf{x}_k$ and $\mathbf{u}_k$ are related through a nonlinear function

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \tag{20}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$, $\mathbf{u}_k \in \mathbb{R}^{n_u}$, n_x and n_u are the state and input dimensions, respectively. $\mathbf{w}_k$ is a noise term, $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ is modelled by multiple GPs as

$$f_i \sim \mathcal{GP}(m_i(\mathbf{z}_k), k_i(\mathbf{z}_k, \mathbf{z}'_k)), \quad i = 1, \dots, n_x, \tag{21}$$

where $\mathbf{z}_k = \begin{bmatrix}\mathbf{x}_k^\top & \mathbf{u}_k^\top\end{bmatrix}^\top$. When a prior model, e.g., a nominal model $\mathbf{f}_0(\mathbf{x}_k, \mathbf{u}_k)$, is available, it can be included as a fixed mean function $m_i(\mathbf{z}_k) = f_{0,i}(\mathbf{x}_k, \mathbf{u}_k)$, $i = 1, \ldots, n_x$. The covariance function $k_i(\cdot,\cdot)$ is chosen to be the sum of an SE covariance function and a noise covariance function to account for the process noise. This GP model is advantageous for fault-tolerant control and Fig. 3 gives the intuition: when there is no data observation, or the system is operating in nominal conditions, the system's behaviour is dominantly captured by the prior, which is the nominal model $\mathbf{f}_0$. If a fault occurs and data are subsequently collected, $\mathbf{f}$ will be adapted to the data observations and reflect the new behaviour of the system.

With the GP in (21) as the prediction model in the MPC, given

$$\mathbf{x}_{k+i|k} \sim \mathcal{N}(\bar{\mathbf{x}}_{k+i|k}, \boldsymbol{\Sigma}_{\mathbf{x}_{k+i|k}})$$

where $i = 0, \ldots, N$ and N is the prediction and control horizon, the joint distribution of $\mathbf{x}_{k+i|k}$ and the one-step-ahead state prediction $\mathbf{x}_{k+i+1|k}$ can be written as

$$\begin{aligned} &p(\mathbf{x}_{k+i|k}, \mathbf{x}_{k+i+1|k} | \bar{\mathbf{x}}_{k+i|k}, \boldsymbol{\Sigma}_{\mathbf{x}_{k+i|k}}) \\ &= \mathcal{N}\left(\begin{bmatrix}\bar{\mathbf{x}}_{k+i|k} \\ \bar{\mathbf{x}}_{k+i+1|k}\end{bmatrix}, \right. \\ &\quad \left. \begin{bmatrix}\boldsymbol{\Sigma}_{\mathbf{x}_{k+i|k}} & \boldsymbol{\Sigma}_{\mathbf{x}_{k+i|k}, \mathbf{x}_{k+i+1|k}} \\ \boldsymbol{\Sigma}_{\mathbf{x}_{k+i|k}, \mathbf{x}_{k+i+1|k}}^\top & \boldsymbol{\Sigma}_{\mathbf{x}_{k+i+1|k}}\end{bmatrix}\right), \end{aligned} \tag{22}$$

where $\bar{\mathbf{x}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}}$ represents the mean vector and covariance matrix of variable $\mathbf{x}$. (22) performs the core functionality of the GP model and evaluation of this joint distribution can be done with the FITC approximation techniques in Section 3 while considering the input uncertainty. Specifically, elements in the mean vector $\bar{\mathbf{x}}_{k+i+1|k}$ can be evaluated by (A1) in Appendix A, diagonal elements in the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}_{k+i+1|k}}$ by (A11), off-diagonal elements in $\boldsymbol{\Sigma}_{\mathbf{x}_{k+i+1|k}}$ by (A14) and elements in matrix $\boldsymbol{\Sigma}_{\mathbf{x}_{k+1|k}, \mathbf{x}_{k+i+1|k}}$ by (A13).

4.2. Target calculation. The steady-state target calculation solves the optimisation problem

$$\min_{\mathbf{x}_\infty, \mathbf{u}_\infty} \ \mathbf{x}_\infty^\top \mathbf{W}_1 \mathbf{x}_\infty + \mathbf{u}_\infty^\top \mathbf{W}_2 \mathbf{u}_\infty, \tag{23}$$

subject to

$$\begin{bmatrix}(\mathbf{A}_k - \mathbf{I}) & \mathbf{B}_k \\ \mathbf{H}_r & \mathbf{0}\end{bmatrix} \begin{bmatrix}\mathbf{x}_\infty \\ \mathbf{u}_\infty\end{bmatrix} = \begin{bmatrix}-\mathbf{d}_k \\ \mathbf{r}\end{bmatrix}, \tag{24}$$

$$\begin{bmatrix}\mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\mathbf{I}\end{bmatrix} \begin{bmatrix}\mathbf{x}_\infty \\ \mathbf{u}_\infty\end{bmatrix} \le \begin{bmatrix}\mathbf{u}_{\max} \\ -\mathbf{u}_{\min}\end{bmatrix}, \tag{25}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are two weighting matrices and $(\mathbf{x}_\infty, \mathbf{u}_\infty)$ denotes the state and input target pair. $\mathbf{H}_r$ is a selection matrix, choosing the dimensions in $\mathbf{x}_\infty$ for which the set point $\mathbf{r}$ is given. $(\mathbf{A}_k,\ \mathbf{B}_k)$ represent the linearised dynamics, for which a linearisation of the GP mean function is used, i.e.,

$$\mathbf{x}_{k+1} = \begin{bmatrix}\mathbf{A}_k & \mathbf{B}_k\end{bmatrix} \begin{bmatrix}\mathbf{x}_k \\ \mathbf{u}_k\end{bmatrix} + \mathbf{d}_k, \tag{26}$$

where

$$\mathbf{A}_k = \frac{\partial \mathbf{m}(\cdot)}{\partial \mathbf{x}}\Big|_{\mathbf{x}_k, \mathbf{u}_k}, \quad \mathbf{B}_k = \frac{\partial \mathbf{m}(\cdot)}{\partial \mathbf{u}}\Big|_{\mathbf{x}_k, \mathbf{u}_k}, \tag{27}$$

$$\mathbf{d}_k = \mathbf{m}\left(\begin{bmatrix}\mathbf{x}_k \\ \mathbf{u}_k\end{bmatrix}\right) - \begin{bmatrix}\mathbf{A}_k & \mathbf{B}_k\end{bmatrix} \begin{bmatrix}\mathbf{x}_k \\ \mathbf{u}_k\end{bmatrix}, \tag{28}$$

$$\mathbf{m}(\cdot) = \begin{bmatrix}m_1(\cdot) & \ldots & m_{n_x}(\cdot)\end{bmatrix}^\top. \tag{29}$$

The target calculation is performed at each time step.

4.3. Cost function and constraints. An uncondensed MPC formulation is adopted with the decision variable $\mathcal{V}_k = [\bar{\mathbf{x}}_{k|k}^\top \ \hat{\mathbf{u}}_{k|k}^\top \ \ldots \ \bar{\mathbf{x}}_{k+N-1|k}^\top \ \hat{\mathbf{u}}_{k+N-1|k}^\top \ \bar{\mathbf{x}}_{k+N|k}^\top]^\top$ and a quadratic cost function is used,

$$\begin{aligned} J(\mathcal{V}_k) = \sum_{i=0}^{N-1} \Big(&\|\bar{\mathbf{x}}_{k+i|k} - \mathbf{x}_\infty\|_{\mathbf{Q}}^2 \\ &+ \|\hat{\mathbf{u}}_{k+i|k} - \mathbf{u}_\infty\|_{\mathbf{R}}^2 + \|\Delta \hat{\mathbf{u}}_{k+i|k}\|_{\mathbf{R}_\Delta}^2 \Big) \\ &+ \left\| \begin{bmatrix}\hat{\mathbf{u}}_{k+N-1|k} - \mathbf{u}_\infty \\ \bar{\mathbf{x}}_{k+N|k} - \mathbf{x}_\infty\end{bmatrix} \right\|_{\mathbf{P}_k}^2, \end{aligned} \tag{30}$$

where weighting matrix $\mathbf{P}_k$ for the terminal cost can either be chosen offline or calculated by solving the algebraic Riccati equation (ARE)

$$\begin{aligned} \mathbf{P}_k = &\mathcal{A}^\top \mathbf{P}_k \mathcal{A} \\ &- (\mathcal{A}^\top \mathbf{P}_k \mathcal{B})(\mathcal{R} + \mathcal{B}^\top \mathbf{P}_k \mathcal{B})^{-1} (\mathcal{B}^\top \mathbf{P}_k \mathcal{A}) + \mathcal{Q}, \end{aligned} \tag{31}$$

where

$$\begin{aligned} \mathcal{A} &= \begin{bmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{B}_k & \mathbf{A}_k\end{bmatrix}, & \mathcal{B} &= \begin{bmatrix}\mathbf{I} \\ \mathbf{B}_k\end{bmatrix}, \\ \mathcal{Q} &= \begin{bmatrix}\mathbf{R} & \\ & \mathbf{Q}\end{bmatrix}, & \mathcal{R} &= \mathbf{R}_\Delta. \end{aligned} \tag{32}$$

Equality constraints of the system's dynamics are applied as

$$\bar{\mathbf{x}}_{k+i+1|k} = \mathbf{m}\left(\begin{bmatrix}\bar{\mathbf{x}}_{k+i|k} \\ \hat{\mathbf{u}}_{k+i|k}\end{bmatrix}\right), \quad i = 0, \ldots, N, \tag{33}$$

where $\bar{\mathbf{x}}_{k+i+1|k}$ is calculated by (22). The initial value $\bar{\mathbf{x}}_{k|k}$ comes from the state measurement/estimate. $\boldsymbol{\Sigma}_{\mathbf{x}_{k|k}}$ can either come from a filter (e.g., Kalman filter) or be set to a small value if the measurement of the state contains little noise.

Input constraints and input slew rate constraints are applied as

$$\begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix} \hat{\mathbf{u}}_{k+i|k} \leq \begin{bmatrix} \mathbf{u}_{\max} \\ -\mathbf{u}_{\min} \end{bmatrix}, \tag{34}$$

$$\begin{bmatrix} -\mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_{k+i-1|k} \\ \hat{\mathbf{u}}_{k+i|k} \end{bmatrix} \leq \begin{bmatrix} \Delta\mathbf{u}_{\max} \\ -\Delta\mathbf{u}_{\min} \end{bmatrix}, \tag{35}$$

where $i = 0, \ldots, N$ and $\hat{\mathbf{u}}_{k-1|k} = \mathbf{u}_{k-1}$ is the previous applied control input. Since the state predictions from a GP model are probabilistic, for the time being we only apply constraints on the mean values of the predicted states as a simplification.

5. Example I: Aircraft longitudinal control

The first example is a fairly simple one, which aims to illustrate the basic functionality of Gaussian process model based MPC in fault-tolerant control.

5.1. System and controller description. Aircraft longitudinal motion has fairly simple dynamics, but still contains considerable input redundancy—typically a trimmable horizontal stabiliser (THS), an elevator, and engine(s). It thus lends itself well for the illustration of fault-tolerant control. In this example we use a linear aircraft model

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{36}$$

where states $\mathbf{x}$ and inputs $\mathbf{u}$ are defined in Tables 1 and 2. The system matrix $\mathbf{A}$ and input matrix $\mathbf{B}$ are

$$\mathbf{A} = \begin{bmatrix} 0.867\,19 & 6.6936 \times 10^{-5} \\ -0.027\,773 & 0.998\,95 \\ 0.201\,46 & -2.1676 \times 10^{-4} \\ 0.2 & 0 \end{bmatrix}$$
$$\begin{bmatrix} -0.190\,95 & 0 \\ 0.892\,64 & -1.9609 \\ 0.883\,79 & 0 \\ 0 & 1 \end{bmatrix}, \tag{37}$$

$$\mathbf{B} = \begin{bmatrix} -3.7758 \times 10^{-3} & -9.0408 \times 10^{-3} \\ 0 & 0 \\ -1.2629 \times 10^{-4} & -3.2794 \times 10^{-4} \\ 0 & 0 \end{bmatrix}. \tag{38}$$

The same model (36) is used as the mean function of the GP model for the system, i.e.

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \tag{39}$$

$$f_i \sim \mathcal{GP}(m_i(\cdot), k_i(\cdot,\cdot)), \quad i = 1, \ldots, 4, \tag{40}$$

$$m_i\left(\begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}\right) = \mathbf{A}(i,:)\mathbf{x}_k + \mathbf{B}(i,:)\mathbf{u}_k, \tag{41}$$

where $\mathbf{A}(i,:)$ represents the i-th row of matrix $\mathbf{A}$ and $\mathbf{B}(i,:)$ represents the i-th row of matrix $\mathbf{B}$.

Table 1. Longitudinal states of the aircraft.

State	Symbol	Unit
Pitch rate	q	rad/s
Air speed	V_{TAS}	m/s
Angle of attack (AoA)	α	rad
Pitch angle	θ	rad

Table 2. Longitudinal inputs and constraints of the aircraft.

Input	Symbol	Feasible region	Slew rate (deg/s)
Elevator	δ_{E}	[−25, 15] deg	[−37, 37]
THS	δ_{THS}	[−10.4, 4.6] deg	[−0.236, 0.236]

The control objective is the tracking of the pitch angle θ. For MPC, the horizon is $N = 20$ steps and the sampling/update interval is $T_s = 0.2$ s, namely, the prediction horizon duration is 4 s. No target calculation is performed and formulation of MPC is the same as in Section 4 with weighting matrices in the cost function (30) chosen as

$$\mathbf{Q} = \mathrm{diag}\left(\begin{bmatrix} 34.38 & 0 & 0 & 103.13 \end{bmatrix}\right), \tag{42}$$

$$\mathbf{R} = \mathbf{0}, \quad \mathbf{R}_\Delta = \mathrm{diag}\left(\begin{bmatrix} 0.02 & 10 \end{bmatrix}\right), \quad \mathbf{P}_k = \mathbf{0}. \tag{43}$$

5.2. Fault scenario and results. The fault scenario investigated here is 'stuck elevators', which refers to the condition that the elevator is jammed at the position where the fault happens and does not respond to control commands. This is a typical scenario to exemplify the 'daisy-chaining' property (Maciejowski, 1998) since the THS—as compared with the elevators—is much less frequently used for attitude control. It is also suitable for demonstrating how performance could be improved with the assistance of a GP model to learn the behaviour of the faulty system.

Simulations are completed on two MPC instances, one with a nominal model and a disturbance observer to implement the daisy-chaining property, the other with a GP model. Reference for the pitch angle changes from 0 to 5 degrees when the simulation starts and the fault happens shortly afterwards at 0.04 s. Simulation results are shown in Figs. 5 to 7. In Fig. 5, the first plot depicts the pitch angle responses. It can be seen that responses from both controllers suffer from oscillations, due to the slow movement of the THS constrained by its maximum slew rate. Both controllers manage to suppress the oscillation while GP-based MPC is more effective and the oscillation dies out more quickly.

Figure 6 shows the control commands to the actuators. The actual movement of the stuck elevator is also shown by the dotted line in the first plot. It can be seen that after the fault happens the elevator command from nominal MPC quickly saturates as a result of the disturbance observation. During this process, the THS is

virtually motionless due to the relatively large weighting on it. After the elevator command saturates, nominal MPC starts to actively command the THS at its maximum speed. As the pitch angle gets closer to the setpoint, a reduction in the control effort is needed. This happens first in the elevator command which moves backwards and saturates in its lower limit, after which the THS starts to move. This pattern introduces oscillations in the elevator command. In contrast, the elevator command from the GP-based controller reveals a different profile: it does not saturate as quickly as that from nominal MPC, indicating that the learning of the GP is going on. Although it also experiences the traverse between the upper and lower limit of the elevator, what follows the traverse indicates an identification of the fault in the elevator, since effective commands are issued to the THS to stop the oscillation while commands to the elevator settle to a nearly constant value.

Figure 7 compares the mean function of the prior and posterior GP (at the end of simulation) modelling the dynamics of the first state $f_1(\mathbf{x}, \mathbf{u})$ to illustrate the learning of the faulty actuator behaviour from the GP. In the plots the surfaces are projections of the mean functions into the elevator-THS plane with states fixed at time 0 and the end of simulation, respectively. The solid lines are the contours of the surfaces. Figure 7(a) shows the prior mapping at the beginning of the simulation, which sketches the linear nominal model. Figure 7(b) shows the posterior mapping at the end of simulation. It can be seen that the posterior mapping has fairly flat contours in the elevator input dimension, which means that changes in this dimension will not significantly affect the state, an indication of failure in the elevator. The optimiser in MPC thus gains not much reward by manipulating this actuator, resulting in the slow-changing elevator commands in Fig. 6.

5.3. Wind gust. In the previous simulation, wind is not considered and a wind speed of 0 m/s is assumed. In this part, a wind gust model is included to test the robustness of the controller towards wind disturbances. In flight control, wind is usually described by its speed in the aircraft body reference frame $\mathbf{W} = \begin{bmatrix} w_x & w_y & w_z \end{bmatrix}^\top$ and the inclusion of the wind changes the wind-related aircraft states, e.g., airspeed, the angle of attack and the sideslip angle. Denote the airspeed vector as $\mathbf{V}_{\text{air}}$ in the aircraft body frame; then

$$\mathbf{V}_{\text{air}} = [V\cos\alpha\cos\beta - w_x \; V\sin\beta - w_y \; V\sin\alpha\cos\beta - w_z]^\top,$$

where V, α, β represent the airspeed, angle of attack and sideslip angle in the wind-free condition, respectively. The angle of attack α_{air} and airspeed V_{air} in wind

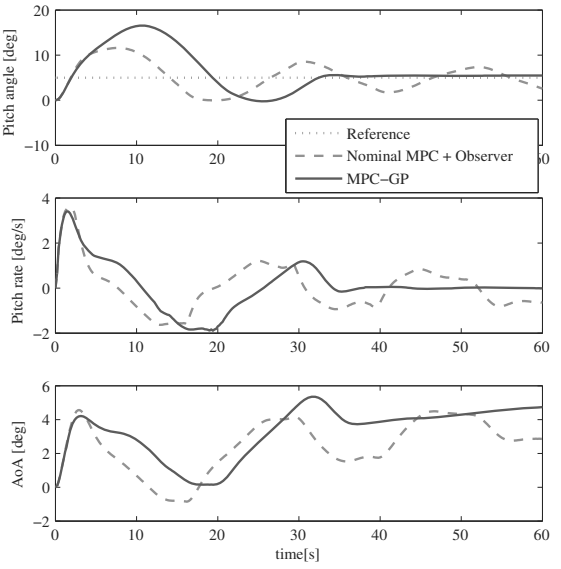


Fig. 5. Responses of some selected longitudinal states, with comparisons between nominal MPC with a disturbance observer and GP-based MPC.

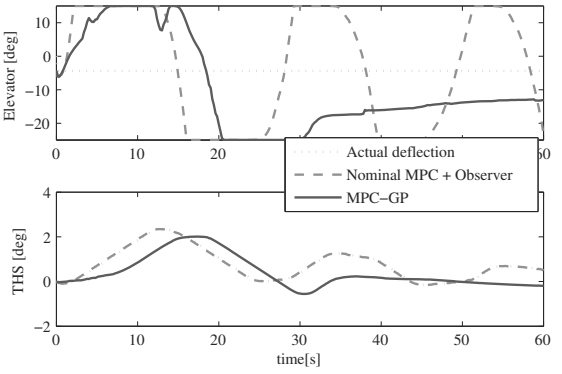


Fig. 6. Commands from MPC and the actual control surface deflections, with comparisons between nominal MPC with a disturbance observer and GP-based MPC.

conditions are

$$\alpha_{\text{air}} = \text{atan2}(V\sin\alpha\cos\beta - w_z, V\cos\alpha\cos\beta - w_x)$$

and

$$V_{\text{air}} = \sqrt{\mathbf{V}_{\text{air}}^\top \mathbf{V}_{\text{air}}}.$$

These two relations show that the wind-affected states V_{air} and α_{air} are functions of the original states V and α and the wind speed $\mathbf{W}$ (β is assumed to be 0 for longitudinal control). The wind gust is modelled as the standard '1-cosine' shaped curve as in the military specification MIL-F8785C (Anon, 1980). For example, wind speed in

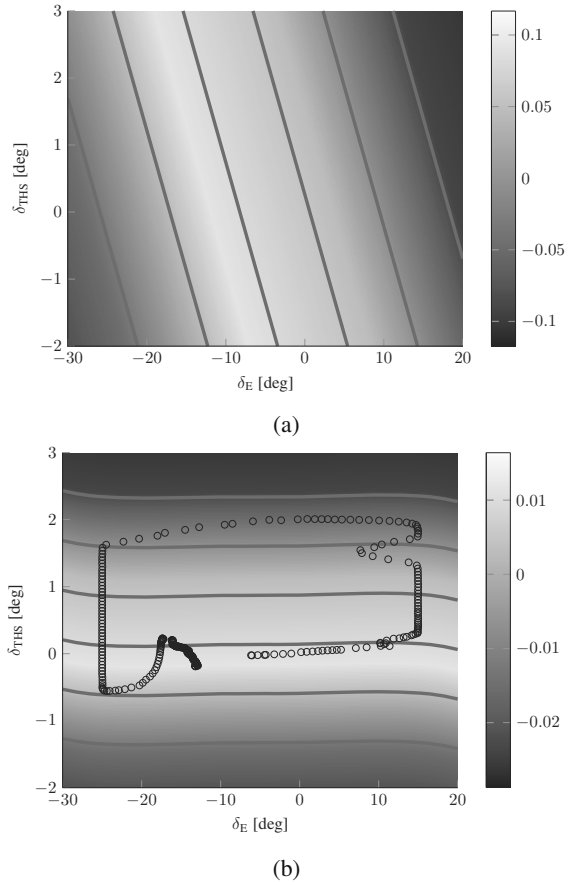


Fig. 7. Prior (a) and posterior (b) mean function of the GP modelling $f_1(\mathbf{x}, \mathbf{u})$ in the 'stuck elevators' fault case. The surface is a projection of the mean function into the elevator-THS plane. The solid lines are contours of the surface. Circles indicate the location of the data accumulated.

the x axis is described by

$$w_x = \begin{cases} 0, & x < 0, \\ \dfrac{V_m}{2}\left(1 - \cos\left(\dfrac{\pi x}{d_m}\right)\right), & 0 \le x \le d_m, \\ V_m, & x > d_m, \end{cases} \tag{44}$$

where V_m is the gust amplitude, d_m is the gust length, x is the distance travelled by the aircraft. w_y, w_z have the corresponding expressions.

A simulation of the 'stuck elevators' fault under the wind gust condition is carried out with $d_{m,x} = 120\,\mathrm{m}$, $V_{m,x} = 5.5\,\mathrm{m/s}$, $d_{m,z} = 80\,\mathrm{m}$ and $V_{m,z} = 3.0\,\mathrm{m/s}$. Results are compared with responses under the same controller in the wind-free condition and are shown in Figs. 8 and 9. It can be seen that, under wind gust conditions, the controller still achieves fairly good control of the aircraft attitude and no significant differences from the wind-free condition can be observed. The explanation

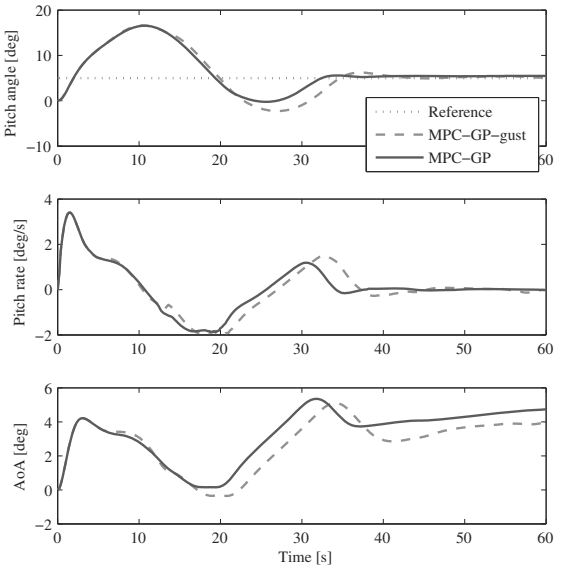


Fig. 8. Selected longitudinal states of the aircraft in the 'stuck elevators' fault case, with comparisons between wind-free and gust conditions.

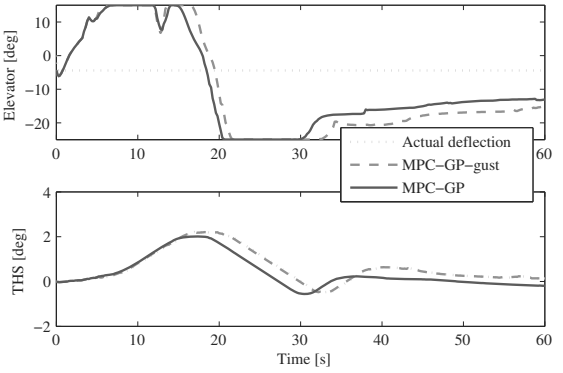


Fig. 9. Commands and actual inputs of the aircraft in the 'stuck elevators' fault case, with comparisons between wind-free and gust conditions.

of this is that gust disturbance to the system is in the form of $\Delta\mathbf{f}(\mathbf{x}, \mathbf{u})$, a static function of the states and inputs. This can be treated as a modified $\mathbf{f}$ in (20), thus can be well captured by the GP model through the input output data. Another fault scenario that falls into this category is the loss of efficiency (LOS) fault in an actuator, for which either the additive or the multiplicative form can be covered by this changed input to state mapping.

A profiling of the execution speed of the controller is also done. In this example the dimensions of the GP problem were, using the notation of Section 3, $D = 6$, $E = 4$, $M = 50$, $n = 300$. The solution of one MPC problem and the training of the Gaussian process each took about 0.6 s on average, using a 2.7 GHz Intel Core i5 processor with 8 GB of 1333 MHz DDR3 memory,

along with the IPOPT optimisation library (Wächter and Biegler, 2006) being called from MATLAB.

6. Example II: GARTEUR RECOVER benchmark

6.1. System and controller description. The GARTEUR RECOVER benchmark model is a 'high-fidelity nonlinear aircraft and fault model for a large transport aircraft', based on the flight data reconstruction, analysis and modelling of the Flight 1862 accident case (Edwards *et al.*, 2010). Two fault scenarios from the benchmark are investigated to evaluate the proposed FTC scheme, namely, 'stuck elevators' and 'rudder runaway'. These faults are assumed to be unanticipated and no FDI information is assumed either, as the GP model in the controller has the capability of learning the effects of the faults and adapting itself to the actual faulty system.

Aircraft states including the engine states are chosen as $\mathbf{x} = [\, p \;\; q \;\; r \;\; V_{\mathrm{TAS}} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \mathrm{EPR1} \;\; \mathrm{EPR2} \;\; \mathrm{EPR3} \;\; \mathrm{EPR4} \,]^\top$, where p, q and r are the angular rates, V_{TAS} is the true airspeed. α and β are the angle of attack and sideslip angle, respectively. ϕ and θ are the roll and pitch angles, respectively. EPR1 to 4 are the engine pressure ratios of the 4 engines. Inputs are defined in Table 3 and illustrated in Fig. 10 (Hartley *et al.*, 2014). The aircraft is trimmed with level flight at an altitude of $h = 600$ m and air speed $V_{\mathrm{TAS}} = 133.8$ m/s. A linearised model around this trim point is obtained and used as the mean function of the GP model. MPC takes a sampling/update interval of $T_s = 0.2$ s and the prediction/control horizon $N = 5$. Target calculation as in Section 4.2 is performed, with the weighting matrices chosen as

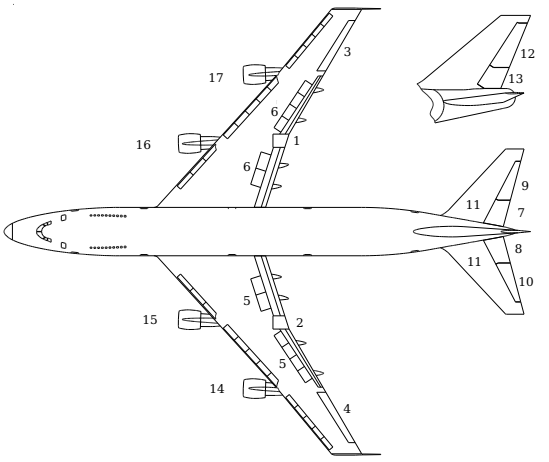
$$\mathbf{W}_1 = 1000 \times \mathbf{I}_{12}, \quad \mathbf{W}_2 = 1000 \times \mathbf{I}_{17}. \tag{45}$$

The selection matrix $\mathbf{H}_r \in \mathbb{R}^{3\times 12}$ is a sparse matrix with $\mathbf{H}_r(1,4) = \mathbf{H}_r(2,7) = \mathbf{H}_r(3,8) = 1$ and 0 for other elements. $\mathbf{u}_{\max}$ and $\mathbf{u}_{\min}$ are as shown in Table 3. The cost function in (30) is used and the values of the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ are adopted from Hartley *et al.* (2014) as

$$\begin{aligned}\mathbf{Q} = \mathrm{diag}\,([&7000 \;\; 1200 \;\; 1400 \;\; 8 \;\; 1200 \;\; 2400 \;\; 4800 \\ &4800 \;\; 0.005 \;\; 0.005 \;\; 0.005 \;\; 0.005])\,, \end{aligned} \tag{46}$$

$$\begin{aligned}\mathbf{R} = \mathrm{diag}\,([&0.002 \;\; 0.002 \;\; 0.002 \;\; 0.002 \;\; 0.003 \;\; 0.003 \\ &0.002 \;\; 0.002 \;\; 0.002 \;\; 0.002 \;\; 21 \;\; 0.05 \\ &0.05 \;\; 3 \;\; 3 \;\; 3 \;\; 3])\,. \end{aligned} \tag{47}$$

Matrix $\mathbf{R}_\Delta = 0.01\mathbf{R}$, and $\mathbf{P}_k$ is calculated according to (31). Dimensions of the GP are, using the notation from Section 3, $D_{\max} = 29$, $E = 12$, $M = 100$, $n = 300$. Notation $D_{\max}$ is used to indicate the maximum number of inputs due to the reduction of inputs in Section 3.4.2 coming into play, and a 12×29 binary masking matrix $\mathbf{G}$ is used to tell which inputs are eliminated (represented by 0 elements). Elements in $\mathbf{G}$ are determined from standard aerodynamics and flight dynamics relations, and are listed in Appendix B.



1	Right Inboard Aileron (RIA)	2	Left Inboard Aileron (LIA)
3	Right Outboard Aileron (ROA)	4	Left Outboard Aileron (LOA)
5	Left Spoiler Panel Array (LSP)	6	Right Spoiler Panel Array (RSP)
7	Right Inboard Elevator (RIE)	8	Left Inboard Elevator (LIE)
9	Right Outboard Elevator (ROE)	10	Left Outboard Elevator (LOE)
11	Trimmable Horizontal Stabiliser (THS)	12	Upper Rudder (UR)
13	Lower Rudder (LR)	14-17	Engines #1 to #4 (EPR)

Fig. 10. Sketch of Boeing 747-200 with numbers indicating inputs used in the case study (source: Boeing).

Table 3. Inputs for the aircraft model.

ID	Input	Feasible region	Slew rate
1,2	RIA, LIA	[−20,20] deg	[−40,45] deg/s
3,4	ROA, LOA	[−25,15] deg	[−45,55] deg/s
5,6	LSP, RSP	[0,45] deg	[−75,75] deg/s
7,8	RIE, LIE	[−23,17] deg	[−37,37] deg/s
9,10	ROE, LOE	[−23,17] deg	[−37,37] deg/s
11	THS	[−12,3] deg	[−0.5,0.5] deg/s
12,13	UR, LR	[−25,25] deg	[−50, 50] deg/s
14-17	EPR 1-4	[0.94,1.62]	[−1, 1]

The control objective is trajectory tracking, for which the reference is a piece-wise linear one consisting of a period of straight and level flight, followed by a 90 degrees change in heading and by a 200 m descent, followed by a 10 m/s deceleration, as shown in Fig. 11. Simulation results of the nominal case are also presented in this figure, which shows not much difference between nominal MPC and MPC with a GP model in this case.

6.2. Fault tolerant control simulation results.

6.2.1. Stuck elevators. The 'stuck elevators' fault is characterised by the condition that all elevator surfaces are stuck at 3-degree downward offsets from the trim

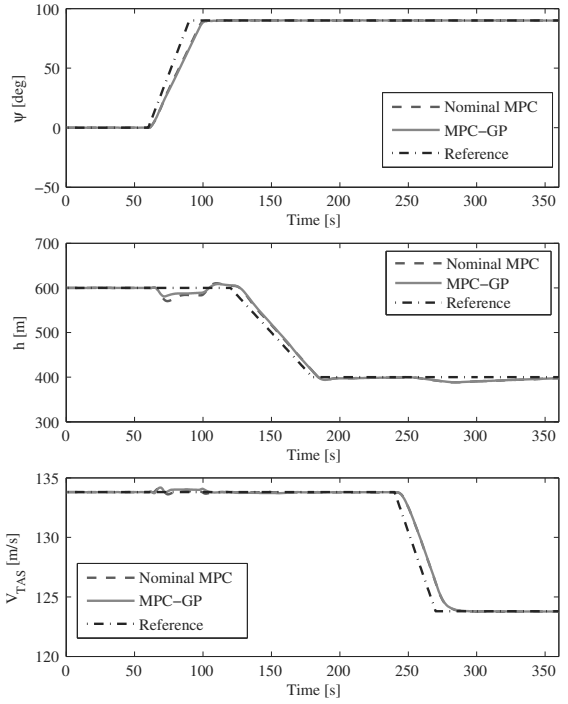


Fig. 11. Reference trajectories tracking in the nominal condition and comparison between nominal MPC and MPC with a GP model.

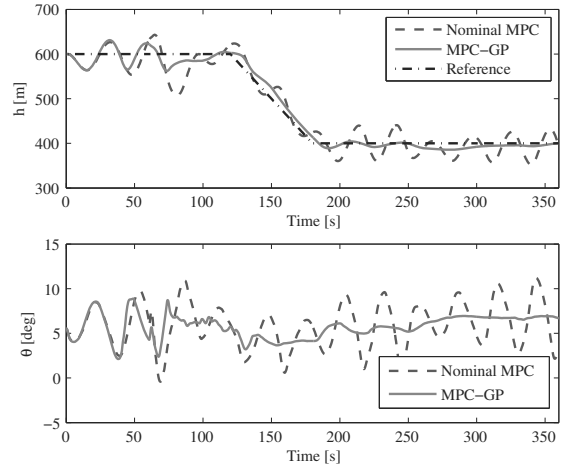


Fig. 12. Altitude and pitch angle responses in the 'stuck elevators' fault case, with comparison between nominal MPC and MPC with a GP model.

position. Both nominal MPC and GP-based MPC achieve satisfactory trajectory tracking in this fault scenario. Since this fault mainly affects the longitudinal dynamics, Figs. 12 and 13 illustrate only the significantly affected states and inputs, with comparisons between the two controllers.

As can be seen in Fig. 12, considerable oscillations occur in both altitude and pitch angle responses under nominal MPC. This is due to the controller being unaware

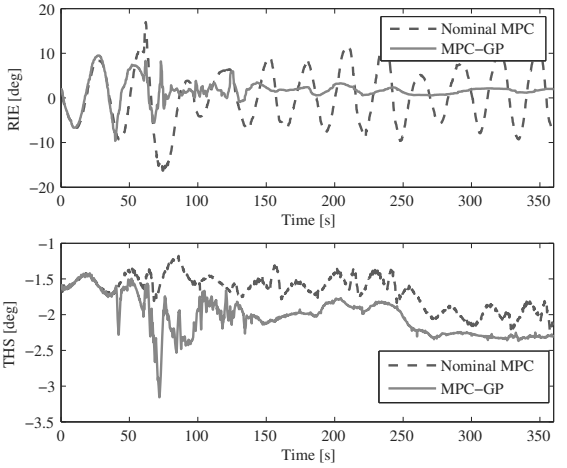


Fig. 13. Comparison of longitudinal input commands in the 'stuck elevators' fault case between nominal MPC and MPC with a GP model. 'RIE' stands for the right inboard elevator.

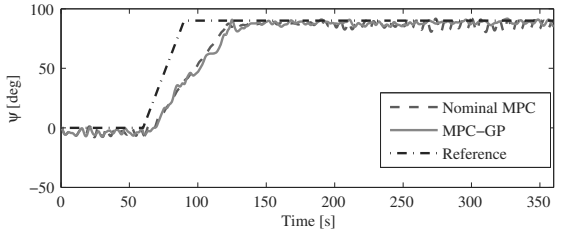


Fig. 14. Yaw angle response in the 'rudder runaway' fault case, with comparison between nominal MPC and MPC with a GP model.

of the fault of the elevator which is still being actively used by nominal MPC, as shown in Fig. 13. When a GP model is engaged at 40 s, MPC with that model gradually identifies the ineffectiveness of the elevators and thus reduces the corresponding command, but increases the command to the THS. This reflects a similar functionality of GP as revealed by the example in Section 5.

6.3. Rudder runaway. The 'rudder runaway' fault describes a condition in which all rudder surfaces move quickly to a limit and create significant roll and yaw moments. In this case both nominal MPC and GP-based MPC achieve good tracking performances. But improvements are still obtained by using the GP model: as can be seen in Fig. 15, oscillations occur in the rudder commands from the nominal controller; when the GP model is engaged at 40 s, this oscillatory command is effectively suppressed as a result of learning the ineffectiveness of the rudders (this also reduces the oscillations in the yaw angle responses from nominal MPC as depicted in Fig. 14).

Other fault scenarios from the benchmark will not be

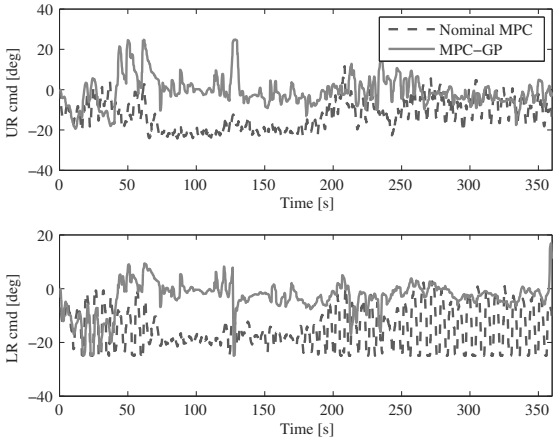


Fig. 15. Comparison of rudder commands in the 'rudder runaway' fault case between nominal MPC and MPC with a GP model. 'UR' and 'LR' stand for the upper and the lower rudder, respectively.

covered here. Some preliminary simulation results can be found in the work of Maciejowski and Yang (2013).

7. Discussion

In our opinion, control is the relatively easy part of fault-tolerant control, *if* one knows what fault(s) has occurred. The harder part is the 'fault detection and identification' part. For control purposes it is not really necessary to know the detailed nature of the fault; it is enough to know the new input/output behaviour of the plant (or input/state behaviour if unmeasured states are to be controlled). Learning this new behaviour is the classical task of system identification. Most methods of system identification produce linear models, and work best in rather simple statistical environments. But when a fault occurs which is severe enough to require the activation of fault-tolerant control—that is, when the inherent robustness of a feedback controller is insufficient—then it is quite likely that the plant being controlled will be driven away from its previous operating condition, and will thus exhibit significantly nonlinear behaviour. Furthermore, neither the control inputs acting on it nor disturbances are likely to resemble stationary random processes. Hence standard methods of system identification are very likely not to work. Furthermore, such methods usually require quite large amounts of data to be collected before they give reliable new models.

On the other hand, using Gaussian processes in the manner outlined in Section 4 yields nonlinear dynamic models, in general. Furthermore, Gaussian processes have been known to give useful models from remarkably little data. This is reinforced in Section 5, in which a useful GP model was learned from about 25 s of (simulated) data, sampled at 0.2 s, namely, from about 125 input-output observations.

Our proposal of combining Gaussian processes with MPC involves two potential computational bottlenecks: solving the optimisation required for MPC, and training the GP model on data and extracting predictions from it. The times reported for these tasks at the end of Section 5 are too long for real-time execution on an aircraft. But these times have been obtained without any serious effort at speeding up the computations either by algorithmic refinements or by deploying a suitable hardware platform. In the light of experience reported, for example, by Hartley *et al.* (2012; 2014), one can be rather confident that the execution times can fairly easily be reduced to an extent that makes it feasible to use these methods, even on relatively fast systems such as aircraft.

A long-standing problem in the field of fault-tolerant control is: What kind of academic results can we expect to get? It seems unreasonable to expect that we can obtain proofs that our methods work, following the occurrence of faults, which almost by definition are unknown in advance. Also, in a sense it seems to be unnecessary to obtain mathematical proofs of success. After all, human operators such as pilots take actions to try to compensate for faults without having such proofs. They start with a very strong belief that if they do nothing then disaster will occur. Presumably they take actions because they believe that these will reduce the probability of disaster. Bayesian inference and reasoning give a theoretical framework which offers a chance of replicating and formalising the decision-making methodology of human operators—with the added benefit of eliminating inconsistency from the process. It has proved to be a very powerful means of taking prior probabilities, which can come from various sources such as mathematical models and subjective beliefs, combining them with observations of data, and computing the resulting posterior probabilities. One attraction of using Gaussian processes, beyond those we have already mentioned, is that they fit within this Bayesian framework (though, of course, they are not the only tools that do so).

8. Conclusions

In this paper we proposed that the use of Gaussian processes in combination with MPC is potentially a very powerful approach to fault-tolerant control. MPC itself is a promising approach to fault-tolerant control because of its flexibility, as argued in Section 2. Gaussian processes have proved themselves to provide very powerful modelling tools in other applications (Deisenroth and Rasmussen, 2011). In the examples of Sections 5 and 6 we reinforced that evidence using examples of fault-tolerant flight control. It also appears that the proposed approach can be made feasible for real-time implementation, with some further development effort.

Of course, it needs hardly be said that there are still many unanswered questions before the approach that we propose is really proven to be effective for fault-tolerant control. An incomplete list of such questions is as follows:

- How much data are needed to obtain a useful Gaussian process model (or any other kind of model) and how long will it take to collect these data?
- Should we deliberately perturb the plant in order to get better data for the modelling? (This is the classic 'dual control' question.)
- How can we detect that the new model we are building is better (for the purpose of control) than the old one we already had?
- How should we combine the unstructured information we get from a Gaussian process with the highly structured information we may have—such as 'the rudder is jammed at 3 degrees'? (For a very preliminary start to answer this, see the work of Hall *et al.* (2012).)
- How can we hope to validate and verify a system based on our proposed approach?

Despite this formidable list, we remain optimistic that the combination of Gaussian processes and MPC will indeed prove to be a powerful approach to the problem of fault-tolerant control.

Acknowledgment

This research was supported by EU Framework Programme 7, project 314544, *RECONFIGURE: Reconfiguration of Control in Flight for Integral Global Upset Recovery*, as well as the China Scholarship Council and the Cambridge Overseas Trust.

We have benefited from codes provided by Carl Rasmussen as well as discussions with Carl Rasmussen, Ed Hartley and Joe Hall.

This paper was previously presented at the 2013 Conference on *Control and Fault-Tolerant Systems (SysTol)*.

References

Anon (1980). Military specification, flying qualities of piloted airplanes, MIL-F-8785C.

Anon (2015). Federal motor vehicle safety standards, Standard no. 135: Light vehicle brake systems, *Technical report*, National Highway Traffic Safety Administration, Washington, DC, `http://www.access.gpo.gov/nara/cfr/waisidx_08/49cfr571_08.html`.

Åström, K.J. (1970). *Introduction to Stochastic Control Theory*, Academic Press, New York, NY.

Deisenroth, M. (2010). *Efficient Reinforcement Learning Using Gaussian Processes*, KIT Scientific Publishing, Karlsruhe.

Deisenroth, M. and Rasmussen, C. (2011). PILCO: A model based and data-efficient approach to policy search, *Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA*, pp. 465–472.

Edwards, C., Lombaerts, T. and Smaili, H. (2010). *Fault Tolerant Flight Control*, Lecture Notes in Control and Information Sciences, Vol. 399, Springer-Verlag, Berlin/Heidelberg.

Hall, J. (2013). *Machine Learning for Control: Incorporating Prior Knowledge*, Ph.D. thesis, University of Cambridge, Cambridge.

Hall, J., Rasmussen, C. and Maciejowski, J. (2012). Modelling and control of nonlinear systems using Gaussian processes with partial model information, *Proceedings of the IEEE 51st Annual Conference on Decision and Control, Maui, HI, USA*, pp. 5266–5271.

Hartley, E., Jerez, J., Suardi, A., Maciejowski, J., Kerrigan, E. and Constantinides, G. (2012). Predictive control of a Boeing 747 aircraft using an FPGA, *Proceedings of the IFAC NMPC'12 Conference, Noordwijkerhout, The Netherlands*, pp. 80–85.

Hartley, E.N., Jerez, J.L., Suardi, A., Maciejowski, J.M., Kerrigan, E.C. and Constantinides, G.A. (2014). Predictive control using an FPGA with application to aircraft control, *IEEE Transactions on Control Systems Technology* **22**(3): 1006–1017.

Huzmezan, M. and Maciejowski, J. (1999). Reconfigurable flight control during actuator failures using predictive control, *14th IFAC World Congress, Beijing, China*, pp. 301–306.

Joosten, D. and Maciejowski, J. (2009). Model predictive controller design for fault-tolerant flight control purposes based upon an existing output feedback controller, *Proceedings of the 7th IFAC Safeprocess Symposium, Barcelona, Spain*, pp. 253–258.

Kocijan, J., Murray-Smith, R., Rasmussen, C. and Likar, B. (2003). Predictive control with Gaussian process models, *Proceedings of IEEE Region 8 EUROCON 2003: Computer as a Tool, Ljubljana, Slovenia*, Vol. 1, pp. 352–356.

Maciejowski, J.M. (1998). The implicit daisy-chaining property of constrained predictive control, *Applied Mathematics and Computer Science* **8**(4): 695–711.

Maciejowski, J.M. (1999). Modelling and predictive control: Enabling technologies for reconfiguration, *Annual Reviews in Control* **23**(1): 13–23.

Maciejowski, J. (2002). *Predictive Control with Constraints*, Prentice-Hall, Harlow.

Maciejowski, J. and Jones, C. (2003). Fault-tolerant flight control case study: Flight 1862, *Proceedings of the 5th IFAC Safeprocess Symposium, Washington, DC, USA*, pp. 265–276.

Maciejowski, J. and Yang, X. (2013). Fault tolerant control using Gaussian processes and model predictive control, *Proceedings of the 2nd International Conference on Control and Fault-Tolerant Systems, Nice, France*, pp. 1–12.

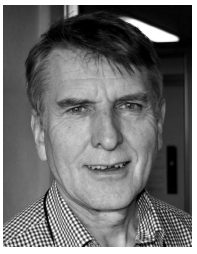
Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA.

Rawlings, J. and Mayne, D. (2009). *Model Predictive Control: Theory and Design*, Nob Hill Publishing, Madison, WI.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs, *NIPS 2005, Vancouver, Canada*.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Mathematical Programming* **106**(1): 25–57.

Xiaoke Yang received the B.Eng. and M.Eng. degrees in automatic control from Beihang University, Beijing, China, in 2007 and 2010, respectively, and the Ph.D degree in control engineering from the University of Cambridge, UK. His present research interest is fault tolerant control with predictive control and Bayesian nonparametric modelling approaches.



Jan M. Maciejowski received the B.Sc. degree in automatic control from Sussex University, Sussex, UK, in 1971, and the Ph.D. degree in control engineering from Cambridge University, UK, in 1978. He was a systems engineer with Marconi Space and Defence Systems Ltd., Frimley, Surrey, UK, from 1971 to 1974, working mostly on attitude control of spacecraft and high altitude balloon platforms. Since 1981, he has been with the University of Cambridge, where he is currently a professor of control engineering and the head of the Information Engineering Division. He was the president of the European Union Control Association from 2003 to 2005 and the president of the Institute of Measurement and Control in 2002. His current research interests include the theory and applications of predictive control, and its application to fault-tolerant control, in system identification, and in the control of autonomous systems. Prof. Maciejowski was a recipient of the Honeywell International Medal from InstMC in 2008.

Appendix A
GP with FITC approximation

For a GP model with the mean function (10) and the covariance function (11), given test input $\mathbf{x}_* \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \in \mathbb{R}^D$, the distribution of the output from the underlying function is $p(f_*) = \mathcal{N}(\mu_*, \sigma_*^2)$. Detailed expressions are listed as follows if FITC approximation is applied through inducing inputs $\bar{\mathbf{X}} = \begin{bmatrix}\bar{\mathbf{x}}_1 & \dots & \bar{\mathbf{x}}_M\end{bmatrix}^\top \in \mathbb{R}^{D\times M}$. Note the bar notation $\bar{\mathbf{x}}$ here represents the inducing inputs, instead of the mean value as used in Section 4.

The mean value of the output is

$$\mu_* = \boldsymbol{\beta}^\top \mathbf{q}, \tag{A1}$$

where $\mathbf{q} \in \mathbb{R}^{M\times 1}$ with

$$q_i = \sigma_f^2 |\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1} + \mathbf{I})|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\bar{\mathbf{x}}_i - \boldsymbol{\mu})^\top (\boldsymbol{\Sigma} + \boldsymbol{\Lambda})^{-1}(\bar{\mathbf{x}}_i - \boldsymbol{\mu})\right], \tag{A2}$$

and $\boldsymbol{\beta} \in \mathbb{R}^{M\times 1}$ with

$$\boldsymbol{\beta} = \mathbf{B}^{-1}\mathbf{K}_{Mn}\boldsymbol{\Gamma}^{-1}\mathbf{y}, \tag{A3}$$

where

$$\boldsymbol{\Gamma} = \operatorname{diag}(\mathbf{K}_{nn} - \mathbf{Q}_{nn}) + \sigma_n^2\mathbf{I}, \tag{A4}$$

$$\mathbf{Q}_{nn} = \mathbf{K}_{nM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{Mn}, \tag{A5}$$

$$\mathbf{B} = \mathbf{K}_{MM} + \mathbf{K}_{Mn}\boldsymbol{\Gamma}^{-1}\mathbf{K}_{nM}, \tag{A6}$$

and

$$\mathbf{K}_{nn} \in \mathbb{R}^{n\times n}, \quad [\mathbf{K}_{nn}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j), \tag{A7}$$

$$\mathbf{K}_{MM} \in \mathbb{R}^{M\times M}, \quad [\mathbf{K}_{MM}]_{i,j} = k(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j), \tag{A8}$$

$$\mathbf{K}_{Mn} \in \mathbb{R}^{M\times n}, \quad [\mathbf{K}_{Mn}]_{i,j} = k(\bar{\mathbf{x}}_i, \mathbf{x}_j), \tag{A9}$$

$$\mathbf{K}_{nM} = \mathbf{K}_{Mn}^\top . \tag{A10}$$

The variance of the output is

$$\sigma_*^2 = \sigma_f^2 - \operatorname{tr}\left[\left(\mathbf{K}_{MM}^{-1} - \mathbf{B}^{-1}\right)\tilde{\mathbf{Q}}\right] + \boldsymbol{\beta}^\top\tilde{\mathbf{Q}}\boldsymbol{\beta} - \mu_*^2, \tag{A11}$$

where $\tilde{\mathbf{Q}} \in \mathbb{R}^{M\times M}$ and

$$\tilde{Q}_{ij} = \frac{k(\bar{\mathbf{x}}_i, \boldsymbol{\mu})k(\bar{\mathbf{x}}_j, \boldsymbol{\mu})}{|2\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1} + \mathbf{I}|^{\frac{1}{2}}} \times \exp\left[(\mathbf{z}_{ij} - \boldsymbol{\mu})^\top(\boldsymbol{\Sigma} + \frac{1}{2}\boldsymbol{\Lambda})^{-1}\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}(\mathbf{z}_{ij} - \boldsymbol{\mu})\right], \tag{A12}$$

where $\mathbf{z}_{ij} = \frac{1}{2}(\bar{\mathbf{x}}_i + \bar{\mathbf{x}}_j)$.

The covariance between input and output is

$$\operatorname{cov}[\mathbf{x}_*, f(\mathbf{x}_*)|\boldsymbol{\mu}, \boldsymbol{\Sigma}] = \sum_{i=1}^{M}\beta_i q_i(\boldsymbol{\Sigma} + \boldsymbol{\Lambda})^{-1}(\bar{\mathbf{x}}_i - \boldsymbol{\mu}). \tag{A13}$$

When there are multiple outputs, covariance between different output dimensions should also be considered, e.g., for outputs f_a and f_b:

$$\operatorname{cov}\left[f_a^*, f_b^* | \boldsymbol{\mu}, \boldsymbol{\Sigma}\right] = \boldsymbol{\beta}_a^\top\tilde{\mathbf{Q}}\boldsymbol{\beta}_b - \mu_a^*\mu_b^*, \tag{A14}$$

where $\tilde{\mathbf{Q}} \in \mathbb{R}^{M\times M}$ and

$$\tilde{Q}_{ij} = \frac{k_a(\bar{\mathbf{x}}_i, \boldsymbol{\mu})k_b(\bar{\mathbf{x}}_j, \boldsymbol{\mu})}{|\boldsymbol{\Sigma}(\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1}) + \mathbf{I}|^{\frac{1}{2}}} \times \exp\left[\frac{1}{2}\mathbf{z}_{ij}^\top\left(\boldsymbol{\Sigma}(\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1}) + \mathbf{I}\right)^{-1}\boldsymbol{\Sigma}\mathbf{z}_{ij}\right], \tag{A15}$$

with $\mathbf{z}_{ij} = \mathbf{\Lambda}_a^{-1}(\bar{\mathbf{x}}_i - \boldsymbol{\mu}) + \mathbf{\Lambda}_b^{-1}(\bar{\mathbf{x}}_j - \boldsymbol{\mu})$.

The training of the GP model with FITC approximation is done by maximising the following likelihood:

$$\begin{aligned}&\log p(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}})\\ &\quad = -\frac{1}{2}\log|\mathbf{Q}_{nn} + \mathbf{\Gamma}|\\ &\qquad - \frac{1}{2}\mathbf{y}^\top(\mathbf{Q}_{nn} + \mathbf{\Gamma})^{-1}\mathbf{y} - \frac{n}{2}\log(2\pi). \qquad \text{(A16)}\end{aligned}$$

Appendix B
Supplementary information for Example II

Input masking matrix $\mathbf{G}$ for the GP model:

$\mathbf{G}(:, 1:14) =$

$$\begin{bmatrix}
1&1&1&1&0&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&0&0\\
1&0&1&1&1&1&1&1&1&1&1&1&0&0\\
1&1&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0
\end{bmatrix},$$

$\mathbf{G}(:, 15:29) =$

$$\begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
0&0&1&1&1&1&1&1&1&0&0&0&0&0&0\\
0&0&1&1&1&1&1&1&1&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}.$$

Received: 31 January 2014
Revised: 10 June 2014