Formal Safety Envelopes for Provably Accurate State Classification by Data-Driven Flight Models

Elkin Cruz-Camacho[*], Ahmad Amer[†], Fotis Kopsaftopoulos[‡], Carlos A. Varela[§]

Rensselaer Polytechnic Institute, Troy, NY, USA

We present a novel approach to formally verify data-driven models for state classification in the domain of aerospace. Dynamic data-driven application systems (DDDAS) extend first-principles models with dynamically sensed data enabling the diagnosis and healing of aircrafts during flight. The intrinsic complexity of these systems makes them prone to spurious errors caused by unseen conditions. Formal (software) verification is a technique that goes beyond unit testing or statistical model checking, which can only guarantee system correctness over a fraction of the system's input space. *Safety envelopes* **bound regions of the system's input space where a formal proof of correctness of state classification holds. We focus on two questions for defining safety envelope boundaries:** (*i*) **does the data follow the model? and** (*ii*) **what is the most likely state of the system given the data? We evaluate safety envelopes with data derived from a wind tunnel experiment tackling the problem of stall detection given piezo-electric sensor measurements over a wing's skin. We define tailored metrics to show the quality of different data-driven models. We encode safety envelopes in the proof assistant Agda, and illustrate their applicability across a variety of input dimensions and Gaussian Process Regression Model (GPRM)-generated data.**

I. Introduction

Aerospace systems are increasingly being used in societal applications from urban air mobility, to bringing packages to customers, to surveying fields of crops, to monitoring wildfires and disaster areas. Yet to be truly autonomous as needed in many of these applications, they lack in terms of self-diagnosis, self-healing, and overall self-awareness. One path forward for aerospace systems to strengthen their resilience is to make them capable of sensing, reasoning, and reacting in real-time, which inevitably means optimal control and decision-making abilities [1]. This is to be aided by access to an unprecedented amount of real-time data from onboard sensors, from which the aeroelastic state, environmental conditions, and structural conditions of aerospace systems [2, 3] can be derived. Smart aerospace systems will be capable of detecting aerodynamic conditions, *e.g.*, stall or flutter, using data from a variety of sources

---

[*]Ph.D. Student, Department of Computer Science, `cruzce@rpi.edu`
[†]Research Engineer at General Electric, `ahmad.amer15@gmail.com`
[‡]Assistant Professor, Department of Mechanical, Aerospace, and Nuclear Engineering, `kopsaf@rpi.edu`
[§]Full Professor, Department of Computer Science, `varelc@rpi.edu`

including piezoelectric sensors, placed on the wings of an aircraft, satellite information, and accurate models of their environment [1, 4]. Dynamic data-driven applications systems (DDDAS) [5] are meant to use this data to create accurate aerodynamic models that can be updated in real-time and be used to determine the aerodynamic performance of flight systems [6].

Since the failure of safety-critical aerospace systems can cause harm to human life, the environment, or property [7], it is imperative to guarantee the correct and safe behaviour of every component in these systems. Statistical model checking [8] has been used in the analysis of statistical systems such as aerospace systems, although it can only guarantee correctness for a finite portion of the input space of the system. Krishnan and Lalithambika [9] present an example of how to use bounded model checking for the analysis of onboard computer code in the Promela language. As models grow and become more complex, the so-called state space explosion becomes a problem for model checking as the time necessary to guarantee any property becomes intractable, thus the need to use statistical and bounded model checking, which explore a portion of the state space. Formal verification goes a step forward and guarantees the correctness of the system specified with it, but only if the system can be fully described. There has been recent work in the formal verification community on complex statistical aerospace systems. The VeriDrone project [10] builds upon existing work using differential dynamic logic [11] to formally verify properties of hybrid systems [12]. Abed et. al [13] formally verify the continuous dynamics that govern the behaviour of uncrewed aerial vehicles (UAVs), for which they formalize the differential equations and dynamics in higher-order logic (HOL). Cohen et. al [14] formally verify the Ellipsoid method used for receding horizon control written in the C language. They modify the algorithm to prevent numerical instability.

We introduce *safety envelopes*, as boundaries in the system's input space where we can formally verify parameterized probabilistic statements on the accuracy of state estimation and classification by data-driven models. In the same manner, flight performance envelopes define a region where it is safe for an aircraft to operate, safety envelopes define regions where a classification is correct according to z-predictability and τ-confidence. z-predictability* captures the statement "the data follows the model", and τ-confidence captures the statement "the state of the system can be accurately determined from the data". The goal of safety envelopes is to reduce Type I and Type II errors when estimating the classification of a value by defining clear boundaries for the state of a system, thus defining safety regions where the system should be constrained to operate. Safety envelopes can only guarantee behavior for stochastic systems that follow the underlying statistical assumptions on the data, *e.g.*, Gaussian distributions. Special runtime programs called *monitors* [15] can analyze real-time data against a safety envelope and determine whether the system is in a distinctly safe state or whether an action should be taken to steer the system away from an unsafe state.
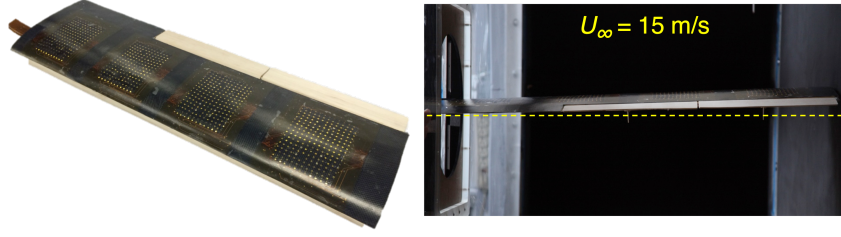
---

*z from the z-score in statistics.

Fig. 1 The composite wing and the wind tunnel setup used to collect data under different flight states. Adapted with permission from Kopsaftopoulos et al. [3]. Copyright 2017 Elsevier Ltda.

The contributions of this paper include:

- Definition of *safety envelopes* as system input regions where probabilistic statements have been formally verified for a statistical data-driven model.

- Presentation of metrics for safety envelopes. Safety envelopes differ from binary classification in which the output of classification is one of three values, not two: positive estimation, negative estimation, and *not* in the safety envelope.

- Formalization, i.e., specification and proof, of four probabilistic properties, and monitor generation using the Agda proof assistant [16] and the Haskell programming language.

- Application of safety envelopes to a safety-critical aviation problem, stall detection, using data from piezoelectric sensors embedded on a wing's skin. Three classes of Gaussian-based models are considered: univariate, bivariate, and univariate extended with artificial data by GPRMs.

The rest of this paper is organized as follows. Section II introduces our data-driven flight model, including wing piezoelectric sensor experiments, data collection, and preprocessing strategy. Section III defines safety envelopes and presents quality metrics for choosing different parameters used in safety envelopes. Section IV presents proofs, an example of how the proofs are formally verified in Agda, and the code generation of runnable code from the theory. Section V contains the evaluation of safety envelopes under three different scenarios: univariate data, bivariate data, and GPRM-generated data, for the problem of stall detection. Finally, Section VI discusses related work, and Section VII concludes including potential future work.

II. The data-driven flight model

The complete experimental assessment and evaluation of this work is based on a prototype composite uncrewed aerial vehicle (UAV) wing with embedded sensing capabilities. The prototype wing was designed, constructed, and tested at Stanford University (Figure 1); for a detailed presentation of the wing, see [2, 3]). The wing design is based on the cambered SG6043 high lift-to-drag ratio airfoil with a 0.86m span, 0.235m chord, and an aspect ratio of 7.32. The wing was outfitted with 32 distributed piezoelectric lead zirconate titanate (PZT) sensors (disc PZT 3.175mm
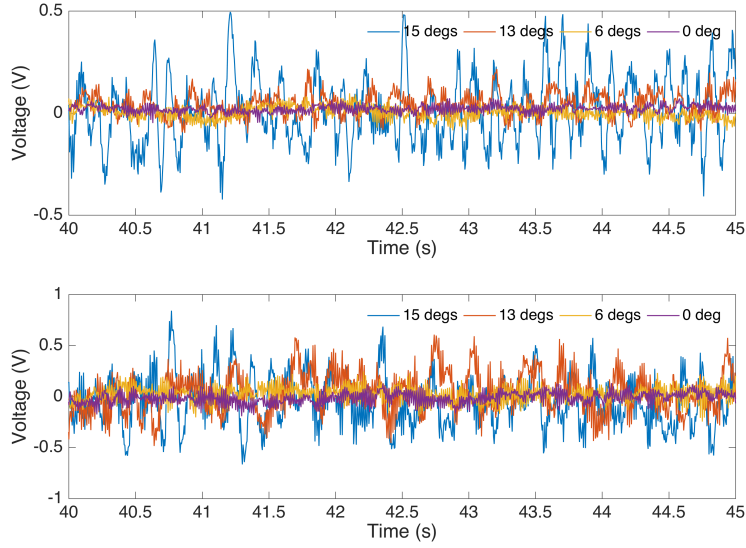
Fig. 2   Indicative signals obtained from a piezoelectric sensor under various angles of attack: (a) freestream velocity U_∞ = 11 m/s (top subplot) and (b) freestream velocity U_∞ = 17 m/s (bottom subplot).

in diameter) and 24 strain gauges to measure its dynamic response. The prototype composite wing was tested in the open-loop low-turbulence wind tunnel facility at Stanford University. A series of wind tunnel experiments was conducted for various angles of attack (AoA) and freestream velocities U_∞. For each AoA, spanning the range from 0 degrees up to 18 degrees with an incremental step of 1 degree, data was sequentially collected for all velocities within the range 9 m/s to 22 m/s (with a step of 1 m/s). The above procedure resulted in a grid of flight state data sets corresponding to 266 different experiments covering the complete range of the considered flight states. For each experiment the vibration response was recorded at different locations on the wing via the embedded piezoelectric sensors (initial sampling frequency f_s = 1000 Hz, initial signal bandwidth 0.1 − 500 Hz). The signals were recorded via a National Instruments X Series 6366 data acquisition module featuring eight 16-bit simultaneously sampled analog-to-digital channels. The initial signals were low-pass filtered (Chebyshev Type II 12th order; cut-off frequency 80 Hz) and sub-sampled to a resulting sampling frequency of f_s = 200 Hz.

In order to investigate the response of the wing under varying AoA and airspeed and determine its flight state, a statistical signal energy analysis was performed for the different sensors. The initial signal of 90 s (N = 91, 000 samples) was split into signal windows of 1 s each. Figure 2 presents indicative piezoelectric signals under different airspeeds and angles of attack. Then, for each signal window the mean value and the standard deviation of the signal energy (time integration of the squared signal V^2 within the time window) were estimated. The goal was to correlate the signal energy in the time domain with the airflow characteristics and aeroelastic properties in order to identify and track appropriate signal features that can be used for the subsequent stall detection of the wing under various flight states. Based on the results of this study [2, 3], it was observed that the vibration data for all the considered states, under
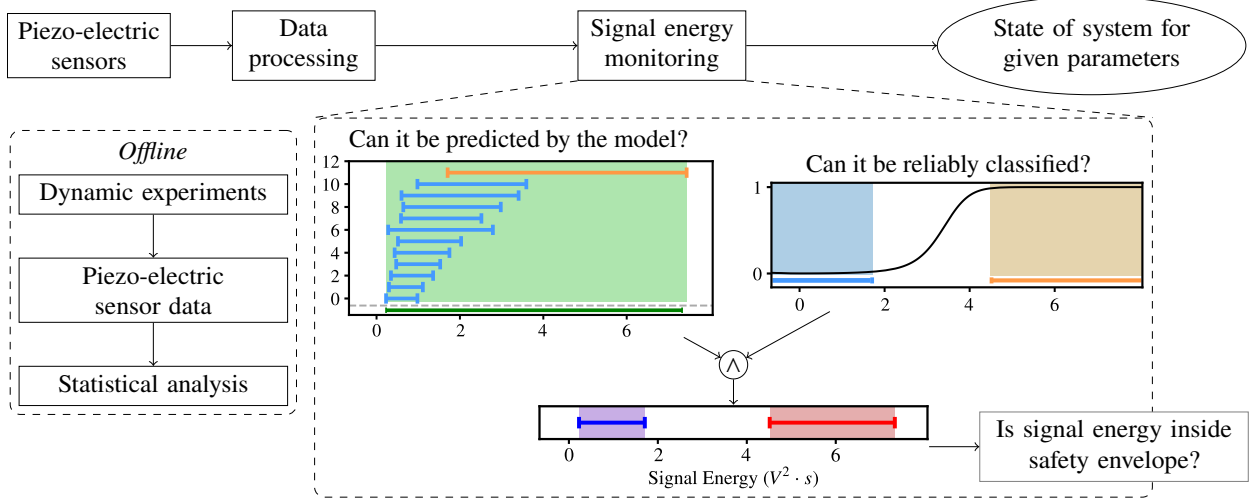
4

Fig. 3   Detection of stall (or its absence) using signal energy safety envelopes.

the aforementioned pre-processing, follow a normal distribution, and thus represent a single flight state with a normal distribution $\mathcal{N}(\mu_\theta, \sigma_\theta^2)$, where μ_θ corresponds to the mean and σ_θ corresponds to the standard deviation of the flight state θ. Therefore, it is possible to compute the multivariate joint normal distributions for the wing sensors under the considered flight states.

Under certain flight states at higher angles of attack, the lift of the wing would decrease below its weight, therefore leading to an aerodynamic stall. The ground truth for the different flight states, i.e., whether the wing exhibits stall or not, was obtained from a series of computational fluid dynamics (CFD) simulations for the same wing design and considered flight states, where the occurrence of stall or no stall was established (for details please see [2, 3]). In addition, during the experiments, the wing base was mounted on a load cell to measure the three forces and three moments (6 degrees of freedom) acting on the wing. The load cell results were in agreement with the CFD-based analysis in indicating the loss of lift, thus the stall of the wing, for the different flight states.

III. Safety Envelopes

Suppose a sensor fails midflight and instead of sounding an alarm, the control system assumes that the sensor is producing accurate data. A human operator or logically redundant system [17] could catch such a mistake and reverse an undesirable action from the control system, but the risk of catastrophic failure is not out of the question—*e.g.*, Air France 447, Tuninter 1153 and Boeing's 737 Max 8 accidents were initiated by such sensor failures [18]. Safety envelopes are dynamic regions of instrument measurements that can be considered correct. If a sensor failed and the data that it produced did not match a model, then the data would be outside of its safety envelope. In case the sensor is producing correct data, the question becomes whether this data should ring an alarm or be used passively by the control system. These scenarios are captured by the following two interlinked questions: is the data predictable by the model?

(z-predictability), and what is the most probable state of the system given the data? (τ-confidence).

The goal of safety envelopes is to determine valid system input regions and their corresponding flight states, *e.g.*, to determine whether a measurement from piezoelectric sensors corresponds to a stall state and to nothing else with very high probability. For this, the values are compared to what a model can sensibly generate (z-predictability) and from which state it is most likely produced (τ-confidence).

A. Univariate Safety Envelopes for Stall Detection

To exemplify safety envelopes, we present the case of stall detection using the signal energy of a single piezoelectric sensor. The input to signal energy safety envelopes is a signal energy $x \in \mathbb{R}$ and its output is one of three classes: stall, no-stall or uncertain. Figure 3 shows how signal energy safety envelopes can be used to detect stall in a live system. In the figure, z-predictability corresponds to the question "can [the signal] be predicted by the model?" and it is represented by the green-colored region. τ-confidence corresponds to the question "can [the signal] be reliably classified?" and it is represented by the light blue and orange regions.

A signal energy model M for stall detection consists of a triple $\langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, C_{stall} \rangle$ where: Θ_{states} are the possible states of the flight system (*e.g.*, all states where angles of attack are natural numbers for an aircraft flying at $15m/s$), $\{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}$ are a family of Gaussian random variables for each state $\theta \in \Theta_{states}$ which encode the distributions of signal energy for each state, and $C_{stall} : \Theta_{states} \rightarrow \{\texttt{stall}, \texttt{no-stall}\}$ is a ground-truth tagging function which determines whether a given flight state is in stall or not. We assume every state is equally likely.

z-predictability determines whether a signal could be likely generated by a model. A signal energy that is not likely to be generated by the model is said to be outside of the safety envelope. Assuming that the signal energy preprocessed from a piezoelectric sensor follows a normal distribution, z-predictability is defined for a signal as:

Definition 1 *Signal energy z-predictability:* *Given a signal energy model $M = \langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, C_{stall} \rangle$, an energy signal x is z-predictable iff there exists a flight state $\theta \in \Theta_{states}$ in the model M such that*

$$\mu_\theta - z\sigma_\theta < x < \mu_\theta + z\sigma_\theta,$$

where μ_θ and σ_θ are the parameters of the normal distribution that describes the signal energy for the flight state θ.

The green regions in the top row of each plot in Fig. 4 illustrate z-predictability given different airspeeds and z parameters.

τ-confidence determines from which state (*stall* or *no-stall*) the signal energy was generated. If the state cannot be determined with enough confidence (τ), then it is said that the signal energy is outside of the safety envelope (and tagged with $\texttt{uncertain}$). For this, we first define[†] a classification function based on a threshold parameter τ:

[†]A helper definition is meant to be used for scaffolding of important definitions (named simply "definition").

Helper definition 1 *Signal energy classification function:* *Given a signal energy model* $M = \langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, C_{stall} \rangle$, *an energy signal x can be classified in one of three categories as:*

$$K_{stall}(M, \tau, x) = \begin{cases} \texttt{stall} & P(\texttt{stall} \mid X = x) \geq \tau \\ \texttt{nostall} & 1 - P(\texttt{stall} \mid X = x) \geq \tau \\ \texttt{uncertain} & \textit{otherwise.} \end{cases}$$

where X is the random variable for the energy signal, $\tau \in (0.5, 1]$, and $P(\texttt{stall} \mid X = x)$ is the conditional probability of stall given $X = x$.

The conditional probability of stall can be computed by the equation $\frac{\sum_{\theta \in \Theta} pdf_\theta(x) P(\texttt{stall=true}|\theta)}{\sum_{\theta \in \Theta} pdf_\theta(x)}$, where $pdf_\theta(x)$ is the probability density function for the distribution $\mathcal{N}(\mu_\theta, \sigma_\theta)$, and the conditional probability $P(\texttt{stall} \mid \theta)$ is determined by the tagging function C_{stall} as 1 if $C_{stall}(\theta) = \texttt{stall}$ and 0 otherwise.

The signature of K_{stall} is $M \times (0.5, 1] \times \mathbb{R} \rightarrow \{\texttt{stall}, \texttt{nostall}, \texttt{uncertain}\}$.

τ is called the threshold of classification and indicates the level of confidence wanted from the classification, or, alternatively, $1 - \tau$ indicates the risk associated with miss-classification [19]. The conditional probability of stall is derived from Bayes' theorem. A step-by-step derivation can be found in Appendix A.

The solid, black curve in the middle row of each plot in Fig. 4 shows the probability of stall for each signal energy value. The left blue regions correspond to the *no stall* class, whereas the right orange regions correspond to the *stall* class, and the unshaded regions in between correspond to the `uncertain` class. Shaded regions are the places where we are confident of the classification with τ certainty.

Definition 2 *Signal energy τ-confidence:* *Given a signal energy model M, and a signal energy $x \in \mathbb{R}$, a classification $K_{stall}(M, \tau, x) = k$ is called τ-confident iff $k \neq \texttt{uncertain}$.*

Safety envelopes are the regions where a signal energy is both z-predictable and τ-confident as exemplified in the following definition.

Definition 3 *Signal energy safety envelopes:* *Given a signal energy model M, $z \in \mathbb{R}^+$, and $\tau \in (0.5, 1]$, a safety envelope $se(M, z, \tau)$ for stall detection is the region $X \in \mathcal{P}(\mathbb{R})$ where the following probabilistic statement holds: for all $x \in X$, x is z-predictable and $K_{stall}(M, \tau, x)$ is τ-confident.*

The shaded regions on the bottom row of each plot in Fig. 4 are the safety envelopes (given z and τ parameters) for model derived by two different fixed airspeeds, and a third model derived from all airspeeds.

Notice that the selection of parameters z and τ influences the size and range of the safety envelopes. A larger z increases the range of z-predictability, and thus data from a larger region of the signal energy space are accepted. A very

large z allows us to accept extremely rare events (outlier data) which includes potentially unsafe data. A larger τ (closer to 1) decreases the region defined by τ-confidence. A τ equal to 1 makes the τ-confident region vacuous. In general, the larger the safety envelopes, the weaker the formal properties associated to them; whereas the smaller the safety envelopes, the stronger the properties we can formally prove about them. The best parameters are application-specific and dependent on the quality of the data and its ability to discriminate between flight classes. We present in subsection III.C several metrics to determine the quality of data-driven models, and thus determine the best safety envelope parameters for a given application.

B. Generalized Safety Envelopes

The concepts that make up signal energy safety envelopes, z-predictability, and τ-confidence, can be easily generalized to multiple input data dimensions. We present one such generalization as models on multivariate-Gaussian distributions. This generalization allows us to use data from multiple correlated inputs such as the signal energy from multiple piezoelectric sensors as shown in Fig. 13. But first, we present some supporting definitions that serve as building blocks for a more rigorous definition of safety envelopes.

A collective-probability model contains all possible states a system can be in. Each state of the system is given by experiment data or by theory and follows a distribution. Each state is assumed to be independent of the others and has some non-zero probability of occurring. As in the case of stall detection, we are often not interested in determining the state of the system (flight state) but rather a condition associated with it (stall). For this, a state is associated with a condition via a tagging function.

Helper definition 2 *Collective-Probability Model: A* collective-probability model *M is a tuple* $\langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta \rangle$ *where:*

- Θ *is a finite set representing the possible states of the system,*
- Ξ *is an arbitrary set representing the space of measurements from the system,*
- $\{X_\theta\}$ *is a family of random variables, where* $X_\theta : \Xi \to \mathbb{R}$ *for* $\theta \in \Theta$, *one per each possible state,*
- p_Θ *is a probability density function, which represents the probability of the system being in a given state,*
- L_Θ *is a set of labels, which correspond to the final output of the classification system, and*
- $C_\Theta : \Theta \to L_\Theta$ *is the ground-truth tagging function.*

The signal energy model from the previous subsection $\langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, C_{stall} \rangle$ can be extended into the collective-probability model $M_{stall} = \langle \Theta_{states}, \mathbb{R}^+ \cup \{0\}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, p_{states}, \{\texttt{stall}, \texttt{no-stall}\}, C_{stall} \rangle$, where:

- $\Theta = \Theta_{states}$ is a set of flight states (*e.g.*, the flight states for angle of attack of 1 degree and airspeeds between $6m/s$ and $20m/s$),
- $\Xi = \mathbb{R}^+ \cup \{0\}$ is the measurement space for the energy signal,

- $\{X_\theta\} = \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}$ is the family of Gaussian random variables that determine how the signal energy behaves at one given state,

- $p_\Theta = p_{states} = \frac{1}{|\Theta_{states}|}$ is the uniform probability density function that encodes the probability of a state to occur,

- $L_\Theta = \{\texttt{stall}, \texttt{no-stall}\}$ is the set of tags, and

- $C_\Theta = C_{stall}(\theta)$ is the ground-truth tagging function.

Safety envelopes are regions defined by a *probabilistic statement*, but what precisely does probabilistic statement mean?

Helper definition 3 *Probabilistic Statement: Given a collective-probability model $M = \langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta \rangle$, and a parameter space Π, a probabilistic statement S over M is a predicate with parameters $\pi \in \Pi$ and $x \in \Xi$, i.e., $S : M \times \Pi \times \Xi \rightarrow \{\texttt{true}, \texttt{false}\}$.*

Given a signal energy model M_{stall}, let us define an example probabilistic statement $S_{distinct}$ as: $S_{distinct}(M_{stall}, \pi, x) := \bigvee_{\theta \in \Theta} (P(\theta \mid X = x) \geq \pi)$, where $P(\theta \mid X = x)$ is the probability that the aircraft is in the state θ given a signal energy of x, and $\Pi = [0, 1]$ is the confidence threshold. $S_{distinct}$ encodes the question of whether a signal energy x can be used to discriminate a unique flight state that generated it. For example, $S_{distinct}(M_{stall}, 0.99, 3.8)$ corresponds to the predicate of "can the flight state that generated a signal of 3.8 be unequivocally determined with a certainty of 99%".

Next, we define a "region of interest", which is the space under Ξ where a probabilistic statement is true. For example, the region where we can guarantee that the sensor produces adequate data (z-confidence).

Helper definition 4 *Region of interest: Given a collective-probability model $M = \langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta \rangle$, a parameter space Π, and a probabilistic statement S over M, a* region of interest *is the region in Ξ under which S holds with parameters π, i.e., a region of interest is the region defined by $RI(M, S, \pi) = \{x \in \Xi : S(M, \pi, x) = \texttt{true}\}$, one per tag.*

The region of interest for $S_{distinct}$ with parameter $\pi = 0.99$ is a subset of $\mathbb{R}$ for which $S_{distinct}(M_{stall}, 0.99, x)$ is true, i.e., $RI(M_{stall}, S_{distinct}, 0.99) \in \mathcal{P}(\mathbb{R})$.

Figure 5 presents two regions of interest for a collective-probability model with two states (Gaussian distributions). The region on the left (blue line at the bottom) corresponds to the probabilistic statement: "the value falls in the interval $(-1.6, 1)$". The region on the right (red line) corresponds to: "the value falls in the interval $(1.5, 4.6)$". Notice that these example probabilistic statements lack any parameters.

With everything in place, let us define a multidimensional generalization for the signal energy model from the previous subsection (III.A):

Helper definition 5 *Collective-Gaussian Stall Model: A collective-Gaussian stall model is the collective-probability model for stall classification $M_{\textbf{stall}} = \langle \Theta_{states}, \mathbb{R}, \{\mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)\}, p_{states}, \{\texttt{stall}, \texttt{no-stall}\}, C_{stall} \rangle$, where*

- $\Theta = \Theta_{states}$ *is the set of air flight configurations for which there is data,*

- $\Xi = \mathbb{R}$ *is the energy signal space,*

- $X_\theta = \mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$ *is a multivariate-normally distributed random variable for the air flight configuration $\theta \in \Theta_{states}$,*

- $p_\Theta = p_{states}$ *is the probability density function that determines the probability $p_{states}(\theta)$ for an air flight $\theta \in \Theta_{states}$ to occur,*

- $L_\Theta = \{\texttt{stall}, \texttt{no-stall}\}$ *is the set of labels, and*

- $C_\Theta = C_{stall} : \Theta \rightarrow L$ *is a tag function for each flight state.*

With these pieces together we can formally define z-predictability, which encodes how well the data being given adjusts to the (flight) model.

Definition 4 *z-predictability:* *Given* *a* *collective-Gaussian* *stall* *model* *$M_{stall} = \langle \Theta_{states}, \mathcal{R}, \{\mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)\}, p_{states}, \{\texttt{stall}, \texttt{no-stall}\}, C_{stall}\rangle$, and a parameter $z \in \Pi = \mathbb{R}^+$, z-predictability is defined as the probabilistic statement:*

$$S_{z\text{-pred}}(M_{stall}, z, \mathbf{x}) = \exists \theta \in \Theta_{states} : D_M(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta, \mathbf{x}) < z$$

where $D_M(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta, \mathbf{x})$ corresponds to the Mahalanobis distance, and it is equal to $\sqrt{(\boldsymbol{\mu}_\theta - \mathbf{x})^T \boldsymbol{\Sigma}_\theta^{-1} (\boldsymbol{\mu}_\theta - \mathbf{x})}$.

The Mahalanobis distance for univariate Gaussian distributions reduces to the z-score region of the distribution, which is proven in theorem 1.

The soft green region on each plot in Fig. 4 shows the z-predictability region for three different models and two values of z. The first two models contain all flight states corresponding to airspeeds 6 m/s and 20m/s, respectively, and angles of attack in the range $\alpha \in [1, 18]$ and $\alpha \in [1, 12]$, respectively.

Now, we define a generalization for τ-confidence using as input a collective-probability model. We introduce a τ (threshold) dependent classification function:

Helper definition 6 *Conditional Classification Function w.r.t. a Model:* *Given a collective-probability model $M = \langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta\rangle$, the conditional classification function is defined as:*

$$K_{cond}(M, \tau, x) = \begin{cases} \texttt{tag} & P(\texttt{tag} \mid X = x) \geq \tau \\ \texttt{uncertain} & \text{otherwise.} \end{cases}$$

where $\tau \in (0.5, 1]$, $x \in \Xi$, X is the random variable for the values measured on Ξ, and $P(\texttt{tag} \mid X = x)$ is the conditional probability for class $\texttt{tag}$ given x. The signature of K_{cond} is $M \times (0.5, 1] \times \Xi \rightarrow L_\Theta \cup \{\texttt{uncertain}\}$.
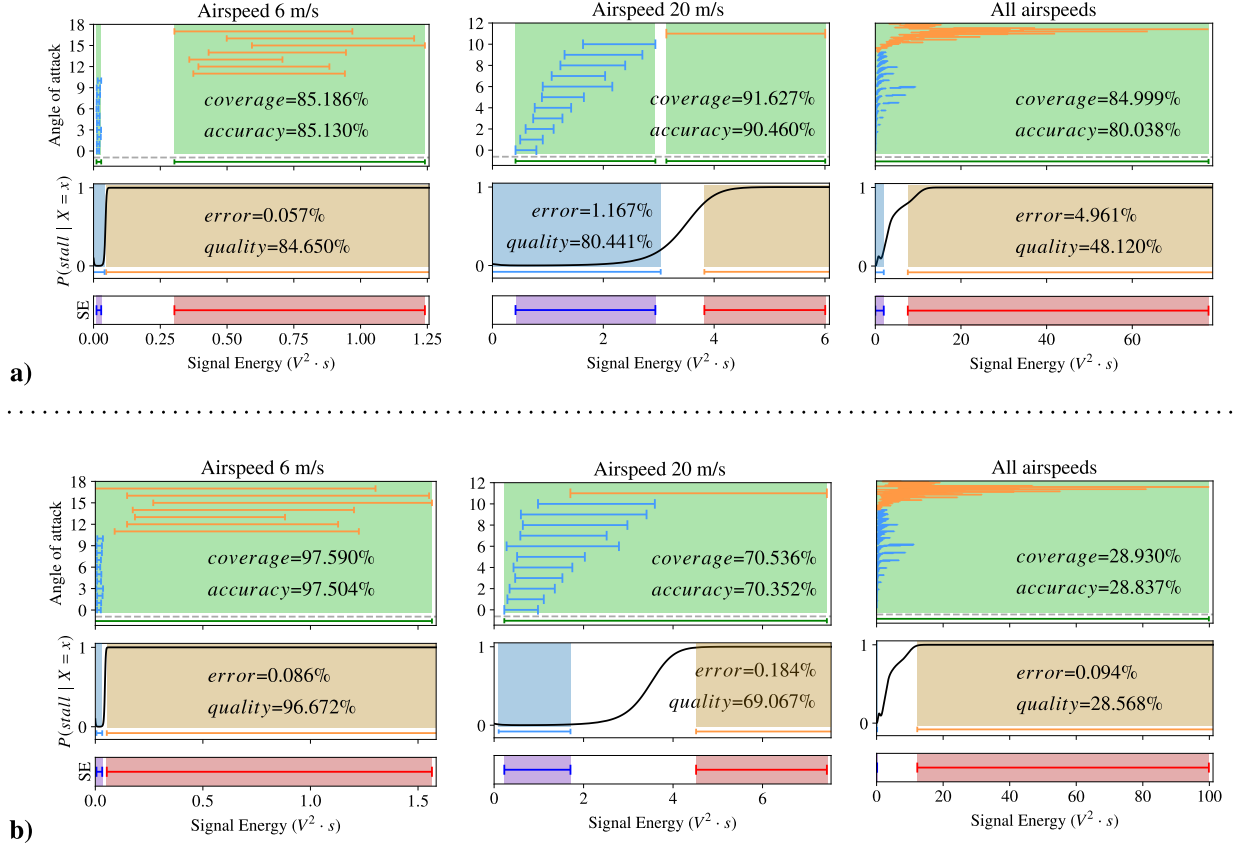
Fig. 4 **Safety envelopes (bottom row) as the intersection of** z**-predictability (top row) and** τ**-confidence (middle row) for airspeed of** $6m/s$**,** $20m/s$ **and all flight states. a)** $z = 1$, $\tau = 80\%$ **b)** $z = 2$, $\tau = 99\%$

The conditional probability $P(\texttt{tag} \mid X = x)$ can be computed by the equation $\frac{\sum_{\theta \in \Theta} pdf_\theta(x) p_\Theta(\theta) P(\texttt{tag}|\theta)}{\sum_{\theta \in \Theta} pdf_\theta(x) p_\Theta(\theta)}$, where $pdf_\theta(x)$ is the probability density function for the distribution X_θ, and the conditional probability $P(\texttt{tag} \mid \theta)$ is determined by the tagging function C_Θ as 1 if $C_\Theta(\theta) = \texttt{tag}$ and 0 otherwise.

The probability of stall (for a univariate collective-Gaussian stall model) can be seen in the middle row of Fig. 4. The black curve corresponds to the probability function $P(\texttt{stall} \mid X = x)$, which indicates the probability of the wing being in a stall condition given a single measurement of the signal energy. A derivation of $P(\texttt{stall} \mid X = x)$ can be found in Appendix A. The classification region can be seen at the bottom of the middle row in Fig. 4, for $\tau = 80\%$ and 99%. The τ-confident region is the union of both colored regions, light blue and orange, where light blue indicates no-stall and orange stall.

We have gathered all that is needed for a multivariate signal-energy safety envelope definition.

Definition 5 *Given a collective-probability model M, and a measurement $x \in \Xi$, a classification $K(M, \tau, x) = k$ is called τ-confident iff $k \neq$ $\texttt{uncertain}$.*

Safety envelopes encode two properties at the same time: whether a value follows a model or can be predicted by it,

and whether a point is likely correctly classified:

Definition 6 *Safety Envelopes Given a collective-Gaussian stall model* $M_{stall} = \langle \Theta_{states}, \mathbb{R}, \{\mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)\}, p_{states}, \{\texttt{stall}, \texttt{no-stall}\}, C_{stall} \rangle$ *and* $(z, \tau) \in \mathbb{R}^+ \times (0.5, 1]$, *a safety envelope SE is the region of interest defined by the probabilistic statements:*

- $\texttt{no-stall}$ *probabilistic statement: "a signal energy value x is z-predictable and the* $\texttt{no-stall}$ *classification is τ-confident"*

$$S_{SE/no\text{-}stall}(M_{stall}, (z, \tau), x) = S_{z\text{-}pred}(M_{stall}, z, x) \wedge K(M_{stall}, \tau, x) = \texttt{no-stall}$$

- $\texttt{stall}$ *probabilistic statement: "a signal energy value x is z-predictable and the* $\texttt{stall}$ *classification is τ-confident"*

$$S_{SE/stall}(M_{stall}, (z, \tau), x) = S_{z\text{-}pred}(M_{stall}, z, x) \wedge K(M_{stall}, \tau, x) = \texttt{stall}$$

The last row of Fig. 4 shows the safety envelopes derived from three different data-driven models with varying z-scores and τ thresholds. For easily separable stall/no-stall conditions, such as $6m/s$, the safety envelope is the same as the region defined by z-predictability; in other cases, the region defined by the τ-confidence reduces the region described by z-predictability or vice versa. Notice that when safety envelopes are applied to a model where all airspeeds and AoAs have been taken into account, the safety envelopes become significantly smaller. This means that it is not possible to assert with high confidence whether a signal energy value entails a stall condition. In the plot on the right of *b)*, the safety envelopes do not include any signal with values from around 1 until 12. In contrast, if we know the airspeed to be $6m/s$, a signal of 1 corresponds likely to a stall, whereas, for an airspeed of $20m/s$, a signal of 1 corresponds likely to no stall.

Safety envelopes can be generalized along other dimensions such as: utilizing a sample of measurements instead of a single measurement; allowing the probability of a state to occur (p_Θ) to change depending on the input (which can be accomplished by defining a p_Θ as a prior, and using $P(\Theta|x)$ inside τ-confidence instead); or, replacing the assumption of normality by defining a custom z-confident process. It is left to the designer of the model to determine the best z-predictability and τ-confidence definitions for their problem.

C. Metrics for Safety Envelopes

In this section, we explore a variety of metrics for safety envelopes that can be used to find the best parameters for a given model and to determine their quality. Metrics allow us to determine the quality of different models and thus compare them. A simple metric is to determine how many data points fall within a region of interest:

Definition 7 *Coverage: Given a collective-probability model* $M = \langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta \rangle$, *a parameter space* Π,

and a probabilistic statement S over M, coverage *is the cumulative probability that falls within the region of interest,* i.e.,

$$coverage(M, S, \pi) = \sum_{\theta \in \Theta} p_{\Theta}(\theta) P\left(x \in RI\left(M, S, \pi\right)\right)$$

with parameters $\pi \in \Pi$.

In Figure 5, an example model with two states and two tags is presented: $M_{eg} = \langle \{\texttt{blue}, \texttt{red}\}, \{X_{\texttt{blue}}, X_{\texttt{red}}\}, p_{eg},$ $\{\texttt{blue}, \texttt{red}\}, C_{eg} \rangle$, where $X_{\texttt{blue}} = \mathcal{N}(0, 1^2)$, $X_{\texttt{red}} = \mathcal{N}(3, 1.4^2)$, and $C_{eg}(x) = x$ the identity function. Two regions of interest are presented for two probabilistic statements: $S_{\texttt{blue}}(M, (y1, y2), x) = y1 < x < y2$ and $S_{\texttt{red}}(M, (z1, z2), x) = z1 < x < z2$. Notice that we need four parameters to define the probabilstic statements and are independent of the data. The example in Fig. 5, showcases the regions for $RI(M, S_{\texttt{blue}}, (-1.6, 1))$ and $RI(M, S_{\texttt{red}}, (1.5, 4.6))$. The coverage of the combined regions $(RI(M, S_{\texttt{blue}}, (-1.6, 1)) \cup RI(M, S_{\texttt{red}}, (1.5, 4.6)))$ is 83.04%.
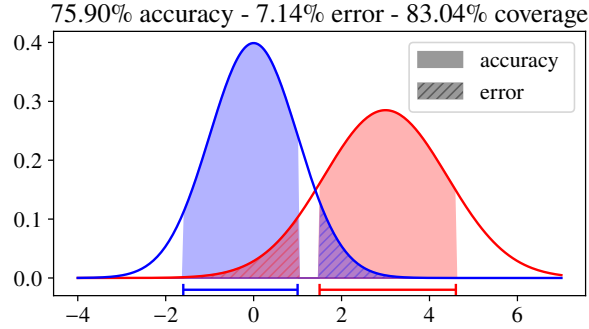
Fig. 5 Representation of accuracy, error, and coverage where the regions of interest are given by the blue and red intervals.

Coverage does not take into account the correct or incorrect classification of a data point. We want a metric that can tell us the quality of the classification. For this, let us analyze from first principles what the possible metrics for safety envelopes are. A metric, in the area of statistical classification, is a value that relates a classification procedure to its performance given some data. A metric is the combination of four possible classification outcomes, namely: false positives, false negatives, true positives, and true negatives[‡]. These outcomes come from the fact that there are two possible classes and two possible estimations. Unfortunately, safety envelopes do not partition the space in only two regions, positive and negative regions, but instead, they partition the space into three regions: positive, negative and "not-inside-safety-envelope". An extended confusion matrix[§] showing all possible six classification outcomes is presented in Table 1. TP stands for true positive, FN stands for false negative.

We would like to find a safety envelope that accepts as few mistakes as possible while covering the largest safe region as possible. In other words, we expect that safety envelopes take:

[‡]A false positive is also known as a Type I error. A false negative is a Type II error.

[§]A confusion matrix is an $n \times n$ matrix that encodes all possible classification outcomes for a classification problem with n classes. Predicted values are assigned to rows and actual values correspond to columns.

Table 1   Extended binary confusion matrix for safety envelopes.

	Positive	Negative
Positive estimation	TP	FP
Negative estimation	FN	TN
Not in SE	Missed P	Missed N

- As few unsafe points as possible: FN + FP / total, *i.e.*, error.

- As many safe points as possible: TP + TN / total, *i.e.*, accuracy.

- As many points as possible: TP + TP + FN + NP / total, *i.e.*, coverage.

Notice that in contrast to machine learning practice, where metrics are computed given a dataset, we are interested in computing the metrics from a model, a collective-probability model.

The expected proportion of safe points, (TN + TP) / total, is captured by:

Definition 8 *Accuracy: Given a collective-probability model $M = \langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta \rangle$, a parameter space Π, and a set of probabilistic statements $\{S_l\}$ over M for each $l \in L_\Theta$,* accuracy *is the cumulative probability that falls within the region of interest and its correctly classified,* i.e.,

$$accuracy(M, \{S_l\}, \pi) = \sum_{l \in L_\Theta} \left(\sum_{\theta \in C_\Theta(\Theta) = l} p_\Theta(\theta) P\left(x \in RI\left(M, S_l, \pi\right)\right)\right)$$

where $C_\Theta(\Theta) = l$ corresponds to the set $\{\theta \in \Theta : C_\Theta(\theta) = l\}$, the set containing all states that are tagged with l.

In Fig. 5, we can see that accuracy corresponds only to the regions correctly classified, red with red and blue with blue. Notice that we can have the same accuracy for different regions of interest (compare Fig. 5 with Fig. 7).

The expected proportion of unsafe points, (FN + FP) / total, is captured by:

Definition 9 *Error: Given a collective-probability model $M = \langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta \rangle$, a parameter space Π, and a set of probabilistic statements $\{S_l\}$ over M for each $l \in L_\Theta$,* error *is the cumulative probability that falls within the region of interest and its **incorrectly** classified,* i.e.,

$$error(M, \{S_l\}, \pi) = \sum_{l \in L_\Theta} \left(\sum_{\theta \in C_\Theta(\Theta) \neq l} p_\Theta(\theta) P\left(x \in RI\left(M, S_l, \pi\right)\right)\right)$$

*where $C_\Theta(\Theta) \neq l$ corresponds to the set $\{\theta \in \Theta : C_\Theta(\theta) \neq l\}$, the set containing all states that are **not** tagged with l.*

In Figs. 5-7, error corresponds to the stripped areas. It is clear that $coverage(M, \{S_l\}, \pi) = accuracy(M, \{S_l\}, \pi) + error(M, \{S_l\}, \pi)$. This means that we can have two different regions of interest with the same accuracy but different error, or different combinations of accuracy and error that give rise to the same coverage.
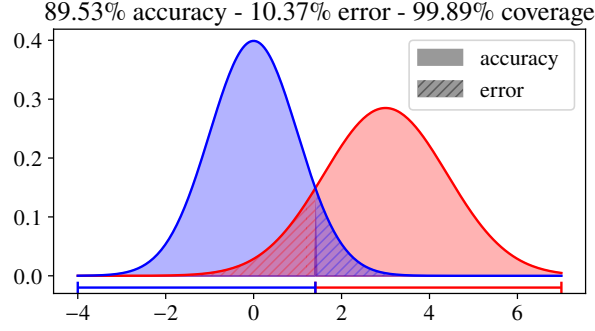
14

Fig. 6 **Representation of accuracy, error, and coverage as in Fig. 5. Larger intervals mean higher accuracy and coverage, but also larger error.**

Notice that high accuracy does not mean small error. It depends on how many points are being discarded by the classification. For this reason, we have to define a metric that encodes our desire for small error and high accuracy, namely:

Definition 10 *Model quality:* *Given a collective-probability model* $M = \langle \Theta, \Xi, \{X_\theta\}, p_\Theta, L_\Theta, C_\Theta \rangle$*, a parameter space* Π*, a set of probabilistic statements* $\{S_l\}$ *over M for each* $l \in L_\Theta$*, and a weight* $w \in \mathbb{R}$*, the* combination of accuracy and error *is*

$$quality(M, \{S_l\}, \pi, w, x) = accuracy(M, \{S_l\}, \pi, x) \times (1 - error(M, \{S_l\}, \pi, x))^w$$

Model quality increases as accuracy do, and it decreases as error increases $(1 - error)$. The w parameter is the weight given to the error. The larger the weight, the costlier error becomes. From observations that can be found in the supplemental material, we have found that an error of $w = 10$ discourages safety envelopes with "too much" error while enforcing a good accuracy.
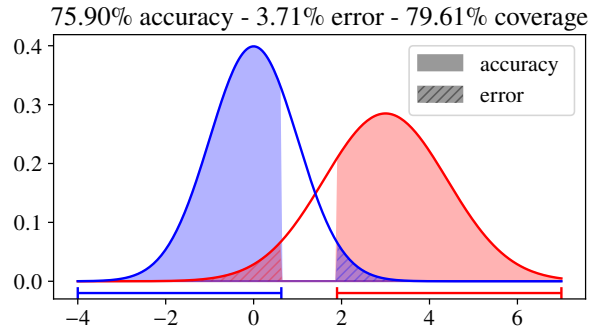
Fig. 7 **Representation of accuracy, error, and coverage as in Fig. 5. Accuracy can be kept as in Fig. 5 while the error is reduced, thanks to a careful tuning of the intervals.**
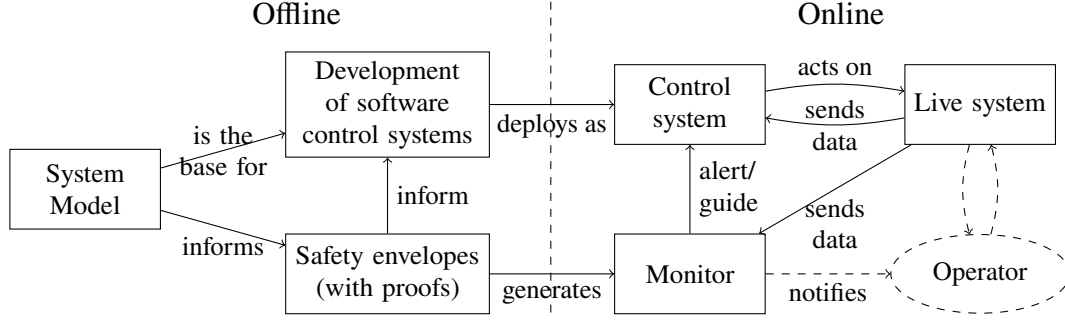
Fig. 8 Safety envelopes and monitors to check for the correct behaviour of a system. Adapted from [20].

IV. Theorems and Formal Proofs

Safety envelopes are to be deployed as an external module to a control system in order to guarantee safe input data. This external module is called a monitor (in runtime verification [15]) and it is to be generated from the safety envelopes in an automatic manner. It is of integral importance that safety envelopes are correctly implemented and guarantee what they are designed for, thus formal proofs of their correct behaviour must accompany them. Figure 8 presents safety envelopes and monitors placement within the production chain and control loop of a control system.

A. Theorems

The following are theorems that we proved mechanically in Agda[¶] as to guarantee the expected behaviour of safety envelopes. The first property to be mechanically proven corresponds to the relationship between z-predictability for univariate normal distributions and the z-score:

Theorem 1 *In the case of univariate normal distributions, the z-predictability condition $D_M(\mu_\theta, \sigma_\theta^2, x) < z$, where D_M is the Mahalanobis distance, reduces to $\mu_\theta - z\sigma_\theta < x < \mu_\theta + z\sigma_\theta$, the prediction interval with a z-score of z.*

Given a signal energy safety envelope, we can determine the connection between z-predictability and signal input as:

Theorem 2 *Given a signal energy model $M_{stall} = \langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, C_{stall} \rangle$, an energy signal $x \in \mathbb{R}$ is z-predictable iff there exist $\theta \in \Theta_{states}$ such that $x \in (\mu_\theta - z\sigma_\theta, \mu_\theta + z\sigma_\theta)$, i.e., x falls within one of the prediction intervals.*

We can prove that a (univariate) signal energy τ-confidence is a special case of τ-confidence:

Theorem 3 *Given a signal energy model $M_{stall} = \langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, C_{stall} \rangle$, and $\tau \in (0.5, 1]$, a classification $K_{stall}(M, \tau, x) = k$ for an observation x is τ-confident iff $P(k \mid x) \geq \tau$.*

As a consequence of Theorems 2 and 3, we can guarantee that signal energy safety envelopes are in fact (general) safety envelopes:

[¶]Full implementation and proofs can be found in the supplementary material to this paper and at `http://wcl.cs.rpi.edu/pilots/fvdddas` (repository name: `safety-envelopes-sentinels`, version 0.1.2.0).

Theorem 4 *Given a signal energy model $M_{stall} = \langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, C_{stall}\rangle$, $\tau \in (0.5, 1]$, and $z \in \mathbb{R}^+$, an energy signal x belongs to safety envelope $RI(M_{stall}, S_{SE/no\text{-}stall} \wedge S_{SE/stall}, (z, \tau))$ iff x is z-predictable and τ-confident.*

B. Formal Proofs and Monitor Generation

We have used the Agda proof assistant to guarantee that the implementation of safety envelopes follows its expected behaviour and the theorems presented before. Additionally, from the Agda code, we can generate verified Haskell code which can be compiled into binary and run separately. Thus we can show a plausible path to implement monitors (see Fig. 8). The following is an excerpt of the formalization where signal energy z-predictability is defined (see definition 1):

```
inside : NormalDist → ℝ → ℝ → Bool
inside nd z x = ((μ - z * σ) <ᵇ x) ∧ (x <ᵇ (μ + z * σ))  where open NormalDist nd using (μ; σ)

z-predictable : Model → ℝ → ℝ → ℝ × Bool
z-predictable M z x = ⟨ x , any (λ nd → inside nd z x) (map (proj₁ ∘ proj₂) (Model.fM M)) ⟩
```

Once safety envelopes have been formally implemented, we can prove properties on them. Such is the case of Theorem 2, which is proven by the following Agda code.

```
-- In words: Given a Model `M` and parameter `z`, `x` is z-predictable iff
-- there exists a pair ⟨α,v⟩ (angle of attack and velocity) such that they are
-- associated to a `nd` (Normal Distribution) and `x` falls within the
-- Predictable Interval
theorem2← : ∀ (M z x)
              → z-predictable M z x ≡ ⟨ x , true ⟩
              → Any (λ{ ⟨ ⟨α,v⟩ , ⟨ nd , p ⟩ ⟩ → x ∈ pi nd z}) (Model.fM M)
theorem2← M z x res≡x,true = any-map (proj₁ ∘ proj₂) (follows-def← M z x res≡x,true)
theorem2→ : ∀ (M z x)
              → Any (λ{ ⟨ ⟨α,v⟩ , ⟨ nd , p ⟩ ⟩ → x ∈ pi nd z}) (Model.fM M)
              → z-predictable M z x ≡ ⟨ x , true ⟩
theorem2→ M z x proofAny = follows-def→ M z x (any-map-rev (proj₁ ∘ proj₂) proofAny)
```

From the Agda formalization, we have generated a *monitor*. A monitor is a computer program whose job is to observe a stream of data to evaluate its consistency and correctness. The interested reader can check the supplemental material, where the full proofs and an extended explanation of the code above can be found. The generated monitor checks when a stream of signal energy measurement, encoded as a floating-point number, is z-predictable. The resulting executable can process a continuous stream of floating-point numbers and outputs a stream of booleans determining the z-predictability of each value. Below a (frankly obscure) piece of Haskell code generated from *z-predictable* on Agda can be seen.

```
name72 = "Avionics.SafetyEnvelopes.z-predictable"
d72 ::
  T24 ->   -- This corresponds to the Model M
  MAlonzo.Code.Avionics.Real.T4 ->   -- real number z
```

```
   MAlonzo.Code.Avionics.Real.T4 ->  -- real number x
   MAlonzo.Code.Agda.Builtin.Sigma.T14  -- a tuple <x, is x z-confident?>
d72 v0 v1 v2
  = coe (d62 (coe (MAlonzo.Code.Data.List.Base.du20 (coe
          (\ v3 -> MAlonzo.Code.Agda.Builtin.Sigma.d28
           (coe (MAlonzo.Code.Agda.Builtin.Sigma.d30 (coe v3)))))
              (coe (d46 (coe v0)))))
          (coe v1) (coe v2))
```

With help of some wrapping functions and code, the function can be called like any other function in Haskell. The implementation and proofs occupy a total of 980 lines in Agda and 130 lines of code in Haskell. From the Agda code, a total of 1160 lines of Haskell code were generated.

V. Experimental Results

A. Signal Energy Safety Envelopes

As explained in subsection II, to evaluate safety envelopes we have constructed multiple models from wind tunnel experiments. Each model is composed of different flight state distributions. We consider three test cases: an easily separable case where only flight states for an airspeed of $6m/s$ are considered, a slightly less separable case with an airspeed of $20m/s$, and a case where we assume no knowledge of airspeed (all airspeeds and AoA flight states are taken into account). The exploration of the optimal τ and z for each of the three cases can be seen in Fig. 9. Note that z is restricted to the range $[0, 0.4]$, in Fig. **c)**, in order to display a readable plot. In all cases, the value of the metrics plateau as z increases outside further out to the right.

As it can be seen in Fig. 9, increasing τ reduces accuracy and error, and thus coverage. The higher z the higher accuracy and error. A more nuanced behaviour can be seen for quality, where there is an optimum for τ but there is no optimum for z, as can be seen in cases **b)** and **c)**. Although, quality increases as z does, there is an important reason not to choose an arbitrarily large z: increasing z allows safety envelopes to accept rare and unlikely values, and so are the chances of accepting an erroneous and possibly unsafe value.

Choosing the parameters carelessly could lead us to unwanted "unsafe" regions. For example, make $\tau = 0.5001$ and $z = 9$. In this case, we would accept a classification that is consistent with both stall and no-stall but that could be generated by either with a high probability. This might not be a big problem if the distributions are many σs apart, but it is a big problem if the model is not highly separable. It makes sense to evade very low values of τ (smaller than 0.65) as well as very small and big values for z.

In Fig. 10, we can see the safety envelopes defined with the optimum z and τ obtained from the exploration shown in Fig. 9. Notice that the error for the large model with all airspeeds is relatively high. This shows that the quality of the signal energy safety envelopes for stall detection decreases when no airspeed information is given. As mentioned before, w forces similar τs for similarly separable data. Even in the case of the model that contains all possible flight states at
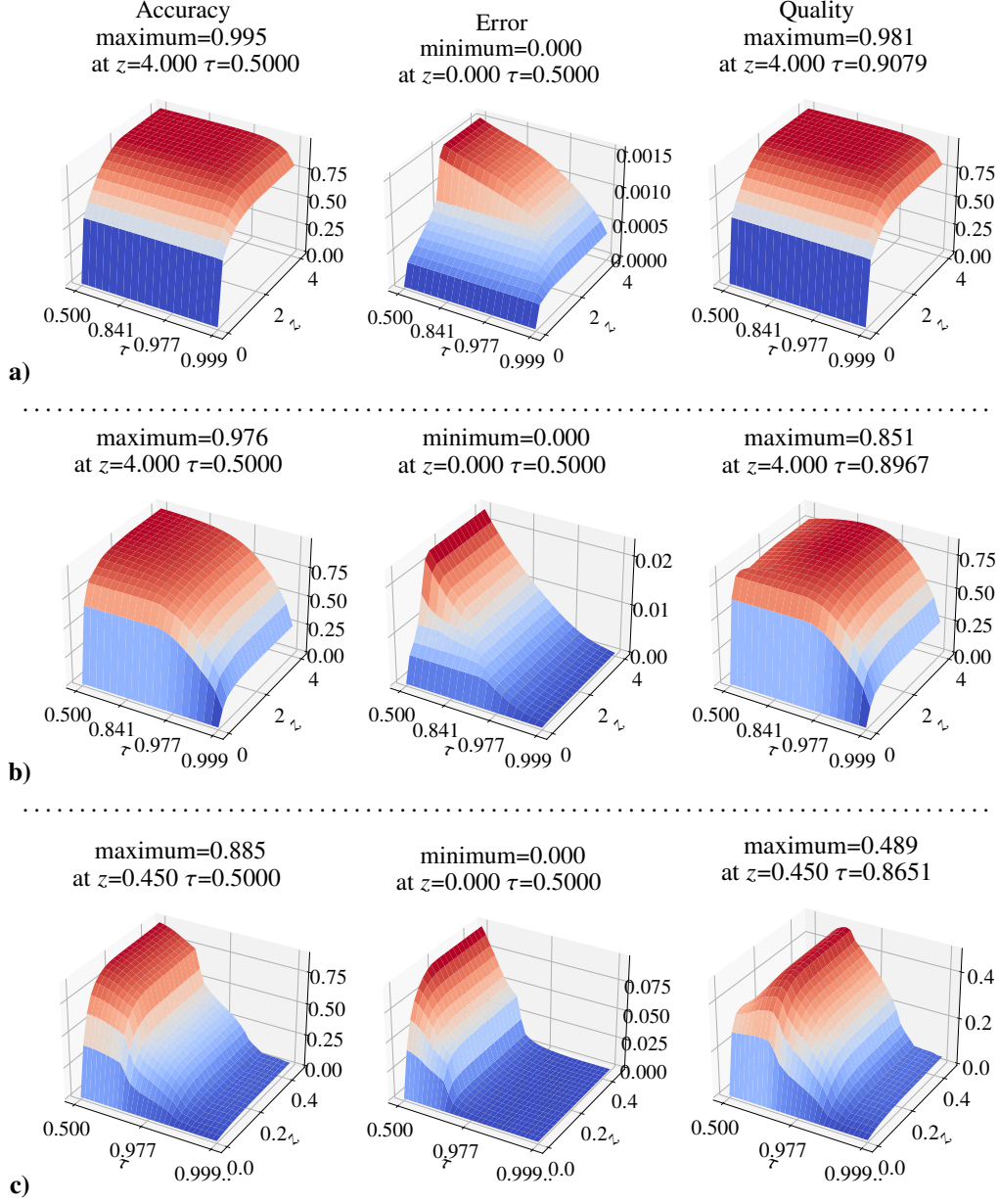
Fig. 9 Optimizing model quality. Exploration of z and τ for the airspeed of a) $6m/s$**, b)** $20m/s$ **and c) all airspeeds, with** $w = 10$**.**

once, the optimal τ stays close to 0.9. What this means is that setting a value for τ on similar models will carry over to, in practice, the same w, or a value close to it. One can just set τ after verifying that the data is separable and one will not need to care much about finding the optimum, which is computationally expensive.

B. GPRMs Applied to Safety Envelopes

GPRMs excel at generalizing a reduced number of observed points into an infinite number of interlinked Gaussian distributions. They are heavily used in aerospace applications as demonstrated by Ahmed et al. [21]. Ahmed et al. made

use of variational heteroscedastic GPRMs, to extend the wind tunnel data, and thus create a larger more refined model. We sample from these variational heteroscedastic GPRMs (VHGPRMs) a hundred Gaussian distributions increasing with artificial data the size of the collective-Gaussian model used to define a safety envelope.

In Figs. 11 and 12, we show a comparison between safety envelopes defined using the distributions computed from wind tunnel experiments and safety envelopes from artificially generated data from GPRMs. With a sufficiently high sample resolution, the z-predictability region becomes a single interval with no gaps even with small values of z. This means that z takes a step back in its influence on the metrics while τ takes full control. There is a significant improvement in the *coverage* and *quality* of the GPRM extended safety envelopes. For a speed of $14m/s$, *coverage* improved from 65.513 to 81.520 and *quality* improved from 63.997 to 77.181, see Fig. 11. A similar improvement can be seen for an airspeed of $17m/s$, see Fig. 11. Preliminary results for other airspeeds, including $6m/s$ and $20m/s$ indicated the same trend of improvements for the metrics. The lack of figures for other airspeeds is due to the nature of GPRM training and the ease with which they overfit, thus producing unnaturalistic results.

The biggest drawback of VHGPRMs is their expensive computing nature and the time-consuming task of finding solid hyperparameters. Even with these disadvantages, VHGPRMs results are hard to match with other techniques. Assuming that VHGPRMs generate artificial distributions as if they were produced by a wind tunnel, any sample we take from them will define a well-behaved safety envelope. With this assumption in place, theorems 1, 2, and 4 apply equally to GPRM synthetic data, where the mean and standard deviation in those expressions can be replaced by the predictive moments of a GPRM. Thus, VHGPRMs add some flexibility in defining safety envelopes, where the moments used for defining them can either come from experimental data, or from properly-trained data-driven VHGPRMs.

In a similar framework, Gaussian Process Classification Models (GPCMs, see [22]), which produce predictive probabilities instead of predictive moments, can be used to "interpolate" the conditional probability for stall in order to allow for "higher resolution" probabilities across the different angles of attack. This approach elegantly allows for the application of theorem 3 onto GPCMs. Thus, with properly-trained GP models (for regression and classification), the concept of safety envelopes can be expanded beyond experimental data using data-driven model-based predictive moments and probabilities.

C. Multivariate Safety Envelopes

The advantage of safety envelopes is their generalizability to multiple dimensions of input data. From the eight available sensors, we chose two sensors with low correlation between them, sensors 1 and 7. We proceeded to compute the mean and covariance matrix for each flight state given the sensors' data. Choosing lowly correlated sensors allows us to show more clearly safety envelopes, as highly correlated sensors show up as lines on the plots. Additional tests revealed no significant difference in the particular selection of sensors concerning the metrics.

Figure 13 shows the safety envelope defined for a bivariate normal distribution-based model. Computing error and
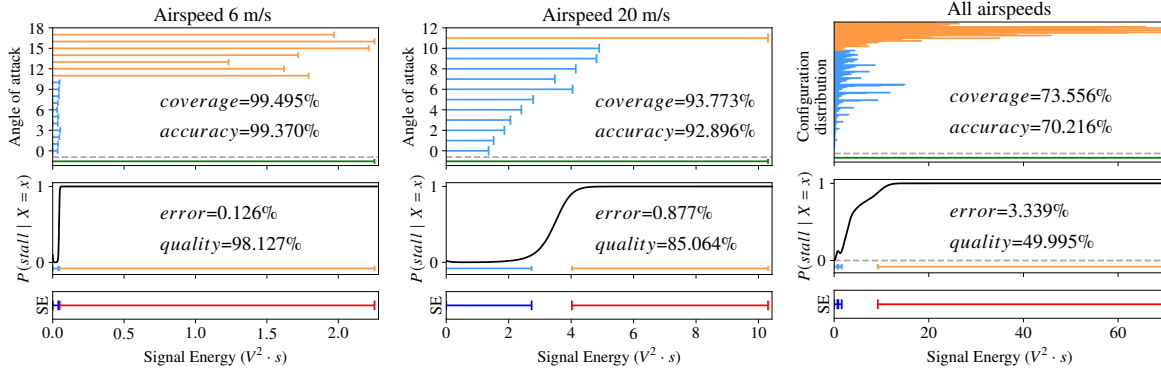
20

Fig. 10   Safety envelopes with optimal parameters z and τ that maximize _quality_ with $w = 10$.
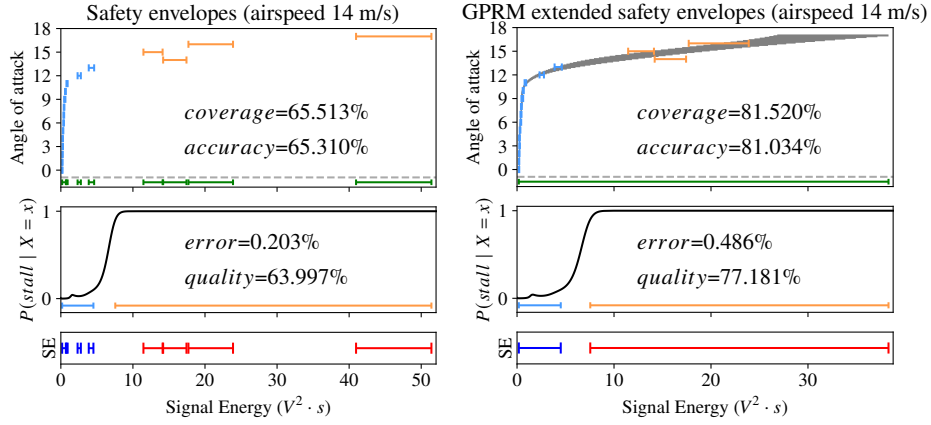

Fig. 11   Safety Envelopes at $14 m/s$ with parameters $z = 0.3$ and $\tau = 0.9$. a) original data, b) GPRM generated data.
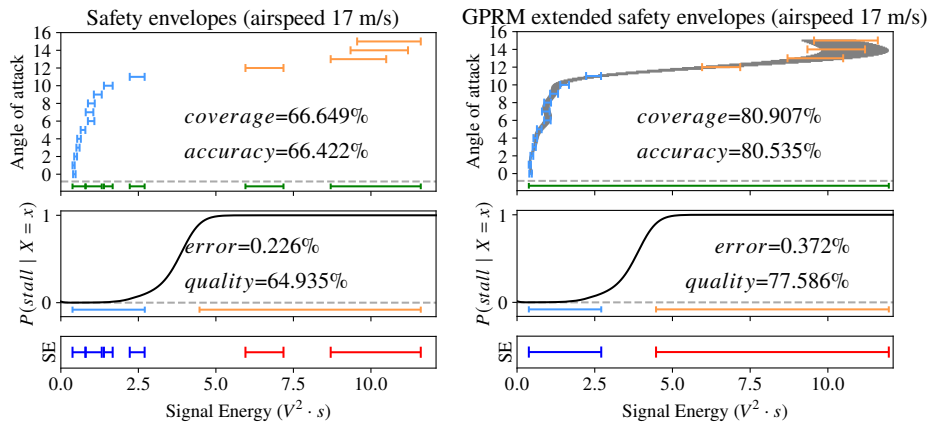

Fig. 12   Safety Envelopes at $17 m/s$ with parameters $z = 0.3$ and $\tau = 0.9$. a) original data, b) GPRM generated data.

accuracy for multivariate-normal distributions required the use of a Monte Carlo simulation as opposed to straightforward computing of the regions from the cumulative distribution function (CDF) as in the univariate models.

Error and quality improve in the multivariate case comparing it to the univariate case, with $error = 0.047\%$ and $quality = 91.361\%$ (see Fig. 13 **b**)), than for the univariate case with $error = 0.877\%$ and $quality = 85.064\%$ (see Fig. 10). Safety envelopes perform better—with less error, and higher model quality—for the case of $20m/s$, even at non-optimized values of z and τ. The difference is heightened when all flight states are considered, where all metrics improve: $accuracy = 78.959\%$, $error = 0.994$ and $quality = 71.449\%$ for the multivariate case against $accuracy = 70.216\%$, $error = 3.339$ and $quality = 49.995\%$ for the univariate case.
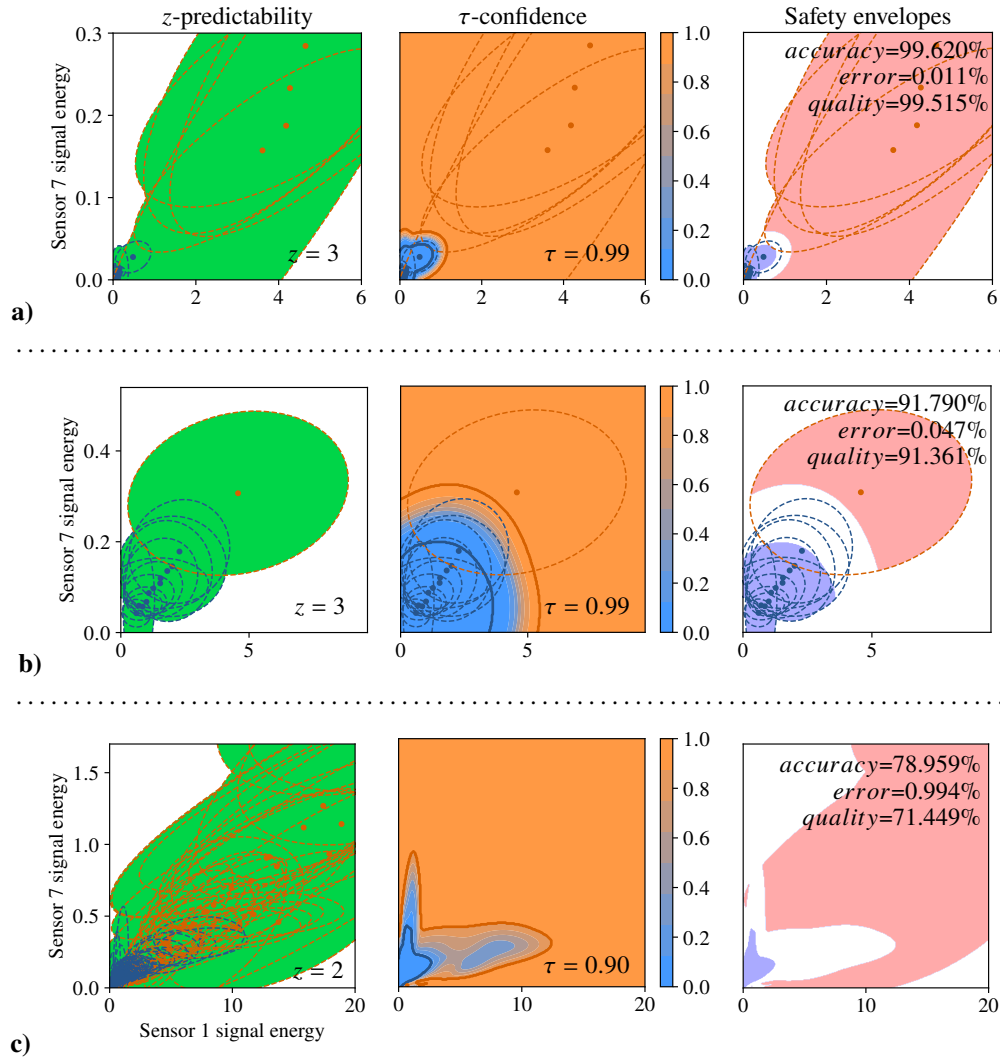
Fig. 13   Multivariate safety envelopes. Distribution means are dots and orange lines (ellipses) are the confidence regions. a) $6m/s$ **b)** $20m/s$ **c) all flight states.**

VI. Related Work

Jackson et. al [23] define a novel concept to determine when a Markov process is safe. From a dataset, they construct a Markov process which they restrict given a parameter ϵ. ϵ is used similarly to safety envelopes z, namely, it restricts safe behaviour to a region where the behaviour is likely to occur and discards anything else outside this region as possibly problematic. Specifically, they check whether the current state of a Markov model is within the range of the possible things the model should do, parameterized by ϵ. Safety envelopes take a step further and include a classification step, τ-confidence, which is firmly grounded on Bayes' rule.

HOL and Isabelle are interactive proof assistants with a rich history of proofs from discrete and continuous probability theory [24–27]. These libraries implement measure theory, discrete, continuous and normal random variables, and many other fundamental theorems on probability theory like the central limit theorem, all of them built from the bottom-up in a robust verifiable environment. In our work, we followed a top-down methodology where we assume the correct formalization of well-known real number theory, and probability theory. In this way, we differ from previous work by implementing more complex structures than what could be done in a limited time with a bottom-up approach. Agda, as opposed to HOL and Isabelle, is a programming language and proof assistant built on top of a constructive theory [28]. It is possible to write code and create an executable with very little extra work in Agda as opposed to HOL and Isabelle. We were able to produce a monitor from the formalization on about 90 lines of Haskell code and additional 130 lines of Agda. Copilot [29] and PILOTS [17, 30] have presented strategies to detect and recover from faulty data streams due to: hardware errors in airplane systems, and dynamic data-driven applications systems (DDDAS), respectively. Those systems do not yet incorporate formal verification and depend therefore on the quality of the software implementation, testing environment, and robustness of the programming language.

Veridrone [31], and other Coq initiatives (*e.g.,* [32]) have incorporated formal verification into working systems to formally prove properties like maximum speed restrictions and correct behaviour according to a specification. These approaches follow a similar framework to ours, namely, the use of software verification techniques for the creation of verified pieces of code, but are different in their end goal, the implementation of verified control systems. Safety envelopes do not steer a system in a specific direction, rather they are meant as a warning system and input for the control system. Another approach for the verification of control systems is DryVR [33], in which a system, defined as a labeled, directed acyclic graph, is determined to behave safely (or not) with the help of simulation trajectories and a blacklist of unsafe states. Although safety envelopes are not an approach to building control systems, they can be used to find unsafe states, which can be later used as blacklists in systems like DryVR.

The concept of a region restricted by parameters where an aircraft can operate safely appears in the literature time and time again. Such is the case of Jeannin et al. [34], who define "safe regions" for an aircraft to operate where no collisions are expected assuming correct behaviour, or Paul et al. [35] with the concept of "correctness envelopes", which determine the conditions for a system to follow some rigid properties. Safety envelopes can be used on their own,

but they can shine when incorporated into larger frameworks of control systems such as the simplex architecture [36]. The simplex architecture's goal is to allow for safe control upgrades of complex control systems. A "safety region" in the simplex architecture delimits the region where a system can be controlled. If the experimental controller gets close to the boundary of the safe region, a robust simpler controller takes over. Safety envelopes could be used to define a tighter region within the hand-written safety region of simplex architecture or as a replacement.

Breese et. al [20] presented the idea of *formal safety envelopes* from which this work sprung into life. They propose a first-order logic-based definition for safety envelopes in which only one parameter is necessary, z, not two, z and τ. Cruz-Camacho et. al [37] extended safety envelopes to encompass both predictability and classification, which results in higher tunability, thanks to the extra parameter τ. Based on the work of Breese et. al, Paul et. al [35] proposed a metric for safety envelopes using preprocessed data as an input. This paper combines and extends all of these prior works with a detailed, justified, and generalized definition of safety envelopes, in particular, we extended Cruz-Camacho et. al's safety envelopes to multivariate distributions and GPRM-generated data from Ahmed et. al [21].

VII. Conclusions

We presented a novel, formally verified concept for classification given a statistical model for one or multiple real numbered data inputs. *Safety envelopes* encode two conditions a safe classification must have: z-predictability, whether an input value is consistent with a model; and, τ-confidence to quantify confidence in a classification. Four metrics to compare different models and parameters of safety envelopes were given: coverage, accuracy, error, and quality. Metrics are fundamental to finding the proper parameters a safety envelope should have. A formalization of safety envelopes in Agda was presented, and with it, four formal proofs that tie z-predictability and τ-confidence with any input value. Formally verified Haskell code was generated from the Agda formalization, and from it, an executable was produced to process a stream of data. We explored how to integrate GPRMs into safety envelopes and their results showcase the extensibility of safety envelopes to use synthetic data. Safety envelopes were shown to work seamlessly with one as well as two input value dimensions, *i.e.*, with models constructed out of univariate or bivariate normal distributions.

Future work includes: extending safety envelopes to correct faulty inputs where physical redundancy is available as in the case of multiple sensor inputs; finding the minimum number of flight states needed to construct a good model to reduce the number of physical experiments necessary to perform; studying the impact of better-informed priors in the quality of the models; and, finding all possible sources of numerical instability which would make floating-point numbers a bad fit as approximations for real numbers.

Acknowledgments

for his input on metrics for safety envelopes.

A. Conditional probability deduction

In this appendix, we present a derivation for the equation to compute the conditional probability $P(\texttt{tag} \mid X = x)$, which was given as:

$$P(\texttt{tag} \mid X = x) = \frac{\sum_{\theta \in \Theta} pdf_\theta(x) p_\Theta(\theta) P(\texttt{tag} \mid \theta)}{\sum_{\theta \in \Theta} pdf_\theta(x) p_\Theta(\theta)} \tag{1}$$

This is possible thanks to Bayes' rule for classification. Namely, we can rewrite the expression as:

$$P(\texttt{tag} \mid X = x) = \frac{f_{X|\texttt{tag}}(x) P(\texttt{tag})}{f_X(x)} \tag{2}$$

The marginal probability $f_X(x)$ can be computed as

$$f_X(x) = \sum_{\theta \in \Theta} f_{X,\theta}(x) = \sum_{\theta \in \Theta} f_{X|\Theta=\theta}(x) p_\Theta(\theta) \tag{3}$$

where $f_{X|\Theta=\theta}(x)$ is the probability density function for the state θ, *i.e.*, $f_{X|\Theta=\theta}(x) = pdf_\theta(x)$ with parameters from X_θ. Note: X is the same as the space Ξ, where x lies, while X_θ is the random variable associated with the state θ.

The conditional probability $f_{X|\texttt{tag}}(x)$ is computed in a similar manner as the marginal probability. First, we apply the law of total probability and then Bayes' rule again:

$$f_{X|\texttt{tag}}(x) = \sum_{\theta \in \Theta} pdf_\theta(x) P(\Theta = \theta \mid \texttt{tag}) = \sum_{\theta \in \Theta} pdf_\theta(x) \frac{p_\Theta(\theta) P(\texttt{tag} \mid \Theta = \theta)}{P(\texttt{tag})} \tag{4}$$

where $P(\texttt{tag} \mid \Theta = \theta)$ is the probability that a specific configuration (flight state) to be tagged with $\texttt{tag}$ (*e.g.*, to produce stall). This probability is either 0 or 1 and it is given by expert judgment.

With the marginal 3 and conditional probabilities 4 in place, we can rewrite equation 2 as:

$$P(stall \mid X = x) = \frac{\sum_{\theta \in \Theta} pdf_\theta(x) p_\Theta(\theta) P(stall \mid \theta)}{\sum_{\theta \in \Theta} pdf_\theta(x) p_\Theta(\theta)} \tag{5}$$

which is the expression shown previously in Equation 1.

References

[1] Kopsaftopoulos, F., and Chang, F.-K., "A Dynamic Data-Driven Stochastic State-Awareness Framework for the Next Generation of Bio-inspired Fly-by-Feel Aerospace Vehicles," *Handbook of Dynamic Data Driven Applications Systems*, Springer, 2018, pp. 697–721.

[2] Kopsaftopoulos, F., Nardari, R., Li, Y.-H., and Chang, F.-K., "Data-driven State Awareness for Fly-by-feel Aerial Vehicles: Experimental Assessment of a Non-parametric Probabilistic Stall Detection Approach," *Structural Health Monitoring 2017*, DEStech Publications, Inc., 2017, pp. 1596–1604.

[3] Kopsaftopoulos, F., Nardari, R., Li, Y.-H., and Chang, F.-K., "A stochastic global identification framework for aerospace structures operating under varying flight states," *Mechanical Systems and Signal Processing*, Vol. 98, 2018, pp. 425–447. https://doi.org/https://doi.org/10.1016/j.ymssp.2017.05.001, URL https://www.sciencedirect.com/science/article/pii/S0888327017302467.

[4] Kopsaftopoulos, F., "Data-driven stochastic identification for fly-by-feel aerospace structures: Critical assessment of non-parametric and parametric approaches," *AIAA Scitech 2019 Forum*, 2019, p. 1534.

[5] Darema, F., "Dynamic data driven applications systems: A new paradigm for application simulations and measurements," *International Conference on Computational Science*, Springer, 2004, pp. 662–669.

[6] Paul, S., Hole, F., Zytek, A., and Varela, C. A., "Flight Trajectory Planning for Fixed Wing Aircraft in Loss of Thrust Emergencies," *Dynamic Data-Driven Application Systems (InfoSymbiotics/DDDAS 2017)*, Cambridge, MA, 2017.

[7] Srivatanakul, T., "Security analysis with deviational techniques," PhD Thesis, University of York, York, UK, Apr. 2005.

[8] Agha, G., and Palmskog, K., "A Survey of Statistical Model Checking," *ACM Transactions on Modeling and Computer Simulation*, Vol. 28, No. 1, 2018, pp. 6:1–6:39.

[9] Krishnan, R., and Lalithambika, V. R., "Modeling and Validating Launch Vehicle Onboard Software Using the SPIN Model Checker," *Journal of Aerospace Information Systems*, Vol. 17, No. 12, 2020, pp. 695–699. https://doi.org/10.2514/1.I010876.

[10] Malecha, G., Ricketts, D., Alvarez, M. M., and Lerner, S., "Towards foundational verification of cyber-physical systems," *2016 Science of Security for Cyber-Physical Systems Workshop (SOSCYPS)*, IEEE, 2016, pp. 1–5.

[11] Platzer, A., "Differential Dynamic Logic for Hybrid Systems," *Journal of Automated Reasoning*, Vol. 41, No. 2, 2008, pp. 143–189.

[12] Ghorbal, K., Jeannin, J.-B., Zawadzki, E., Platzer, A., Gordon, G. J., and Capell, P., "Hybrid Theorem Proving of Aerospace Systems: Applications and Challenges," *Journal of Aerospace Information Systems*, Vol. 11, No. 10, 2014, pp. 702–713.

[13] Abed, S., Rashid, A., and Hasan, O., "Formal Analysis of Unmanned Aerial Vehicles Using Higher-Order-Logic Theorem Proving," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 481–495. https://doi.org/10.2514/1.I010730.

[14] Cohen, R., Feron, E., and Garoche, P.-L., "Verification and Validation of Convex Optimization Algorithms for Model Predictive Control," *Journal of Aerospace Information Systems*, Vol. 17, No. 5, 2020, pp. 257–270. https://doi.org/10.2514/1.I010686.

[15] Falcone, Y., Havelund, K., and Reger, G., "A Tutorial on Runtime Verification," *Engineering Dependable Software Systems*, 2013, pp. 141–175.

[16] Norell, U., "Towards a practical programming language based on dependent type theory," PhD Thesis, Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, Sep. 2007.

[17] Imai, S., Blasch, E., Galli, A., Zhu, W., Lee, F., and Varela, C. A., "Airplane Flight Safety Using Error-Tolerant Data Stream Processing," *IEEE Aerospace and Electronics Systems Magazine*, Vol. 32, No. 4, 2017, pp. 4–17.

[18] Imai, S., Hole, F., and Varela, C. A., "Self-Healing Data Streams Using Multiple Models of Analytical Redundancy," *The 38th AIAA/IEEE Digital Avionics Systems Conference (DASC 2019)*, San Diego, CA, 2019. URL http://wcl.cs.rpi.edu/papers/ DASC2019_imai.pdf.

[19] Alpaydin, E., *Introduction to Machine Learning*, 3rd ed., The MIT Press, Cambridge, Mass, 2014.

[20] Breese, S., Kopsaftopoulos, F., and Varela, C., "Towards Proving Runtime Properties of Data-Driven Systems Using Safety Envelopes," *The 12th International Workshop on Structural Health Monitoring*, Stanford, CA, 2019.

[21] Ahmed, S., Amer, A., Varela, C., and Kopsaftopoulos, F., "Data-Driven State Awareness for Fly-by-Feel Aerial Vehicles via Adaptive Time Series and Gaussian Process Regression Models," *Dynamic Data-Driven Applications Systems (InfoSymbiotics/DDDAS 2020)*, 2020.

[22] Amer, A., and Kopsaftopoulos, F. P., "Towards Unified Probabilistic Rotorcraft Damage Detection and Quantification via Non-parametric Time Series and Gaussian Process Regression Models," *Proceedings of the Vertical Flight Society 76th Annual Forum & Technology Display*, Virginia Beach, VA, USA, 2020.

[23] Jackson, J., Laurenti, L., Frew, E., and Lahijanian, M., "Safety Verification of Unknown Dynamical Systems via Gaussian Process Regression," *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 860–866. https://doi.org/10.1109/ CDC42340.2020.9303814.

[24] Hurd, J., "Formal verification of probabilistic algorithms," Tech. Rep. UCAM-CL-TR-566, University of Cambridge, Computer Laboratory, May 2003.

[25] Hasan, O., and Tahar, S., "Probabilistic analysis of wireless systems using theorem proving," *Electronic Notes in Theoretical Computer Science*, Vol. 242, No. 2, 2009, pp. 43–58.

[26] Qasim, M., Hasan, O., Elleuch, M., and Tahar, S., "Formalization of Normal Random Variables in HOL," *Intelligent Computer Mathematics*, edited by M. Kohlhase, M. Johansson, B. Miller, L. de Moura, and F. Tompa, Springer International Publishing, Cham, 2016, pp. 44–59.

[27] Avigad, J., Hölzl, J., and Serafin, L., "A Formally Verified Proof of the Central Limit Theorem," *Journal of Automated Reasoning*, Vol. 59, No. 4, 2017, pp. 389–423.

[28] Luo, Z., *Computation and reasoning: a type theory for computer science*, Oxford University Press, Inc., USA, 1994.

[29] Pike, L., Wegmann, N., Niller, S., and Goodloe, A., "Copilot: monitoring embedded systems," *Innovations in Systems and Software Engineering*, Vol. 9, No. 4, 2013, pp. 235–255.

[30] Chen, S., Imai, S., Zhu, W., and Varela, C. A., "Towards Learning Spatio-Temporal Data Stream Relationships for Failure Detection in Avionics," *Handbook of Dynamic Data Driven Applications Systems*, edited by E. Blasch, S. Ravela, and A. Aved, Springer International Publishing, Cham, 2018, pp. 97–121.

[31] Ricketts, D., Malecha, G., Alvarez, M. M., Gowda, V., and Lerner, S., "Towards verification of hybrid systems in a foundational proof assistant," *2015 ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2015, pp. 248–257.

[32] Anand, A., and Knepper, R., "ROSCoq: Robots Powered by Constructive Reals," *Interactive Theorem Proving*, Vol. 9236, edited by C. Urban and X. Zhang, Springer International Publishing, Cham, 2015, pp. 34–50.

[33] Fan, C., Qi, B., Mitra, S., and Viswanathan, M., "DryVR: Data-Driven Verification and Compositional Reasoning for Automotive Systems," *Computer Aided Verification*, Vol. 10426, edited by R. Majumdar and V. Kunčak, Springer International Publishing, Cham, 2017, pp. 441–461. https://doi.org/10.1007/978-3-319-63387-9_22.

[34] Jeannin, J.-B., Ghorbal, K., Kouskoulas, Y., Gardner, R., Schmidt, A., Zawadzki, E., and Platzer, A., "A Formally Verified Hybrid System for the Next-Generation Airborne Collision Avoidance System," *Tools and Algorithms for the Construction and Analysis of Systems*, Vol. 9035, edited by C. Baier and C. Tinelli, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 21–36. https://doi.org/10.1007/978-3-662-46681-0_2.

[35] Paul, S., Kopsaftopoulos, F., Patterson, S., and Varela, C. A., "Towards Formal Correctness Envelopes for Dynamic Data-Driven Aerospace Systems," *Handbook of Dynamic Data-Driven Application Systems*, edited by F. Darema and E. Blasch, Springer, 2020. Preprint. To appear.

[36] Seto, D., Krogh, B., Sha, L., and Chutinan, A., "The Simplex Architecture for Safe Online Control System Upgrades," *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, IEEE, Philadelphia, PA, USA, 1998, pp. 3504–3508 vol.6. https://doi.org/10.1109/ACC.1998.703255.

[37] Cruz-Camacho, E., Paul, S., Kopsaftopoulos, F., and Varela, C. A., "Towards Provably Correct Probabilistic Flight Systems," *Dynamic Data Driven Application Systems*, edited by F. Darema, E. Blasch, S. Ravela, and A. Aved, Springer International Publishing, Cham, 2020, pp. 236–244.

Supplementary Material - Formal Safety Envelopes for Provably Accurate State Classification by Data-Driven Flight Models

This document contains supplementary material that could not be fitted in the main article and it is presented for the interested reader on the code aspects of the work or some developments adjacent to the work. Section 1 presents the Bayes theorem as it used in the appendix of the paper. Section 2 is a an introduction to VHGPRMs which are used to generate artificial data for safety envelopes. Section 3 contains some observations relating the weight w and its impact on the metrics. Section 4 contains the Agda code containing all the proofs and supporting libraries. Section 5 contains all the code written in Haskell used to interface with the Agda code and create an actual executable.

Contents

1 Bayes Theorem Application for Classification

Given a sample space X from which data can be sampled, a set of K labels or classes that are assigned to each value in the space, and assuming that each class is associated with a probability distribution, we can determine the probability of a value belonging to one of the classes by applying Bayes' rule [1, pp. 53, eq. 3.5]:

$$P\big(C = C_i \mid X = x\big) = \frac{f_{X|C=C_i}(x)P\big(C = C_i\big)}{f(x)} = \frac{f_{X|C=C_i}(x)P\big(C = C_i\big)}{\sum_{k=1}^{K} f_{X|C=C_k}(x)P\big(C = C_k\big)} \tag{1}$$

where C_i is the class i (out of K classes), $f_{X|C=C_i}(x)$ is the conditional probability of $X = x$ given the class C_i (also called *class likelihood*), and $P\big(C = C_i\big)$ is the probability for the class C_i to occur (thus $\sum_{i=1}^{K} P\big(C = C_i\big) = 1$). Because a value must belong to only one class, the summation of all probabilities for a single value must be one, i.e., $\sum_{i=1} P\big(C = C_i \mid X = x\big) = 1$.

The procedure to classify a value x consists on finding $\arg\max_{1 \le i \le K} P\big(C = C_i \mid X = x\big)$.

2 Variational Heteroscedastic GPRMs

Gaussian process regression models (GPRMs) [2] are standard tools in data-driven applications such as in fly-by-feel with the purpose of generating and extending finite observations of a phenomenon into a family of infinite and interlinked normal distributions.

Given a training data set $\mathcal{D}$ containing n inputs-observation pairs $\{\mathbf{x}_i \in \mathbb{R}^D, y_i \in \mathbb{R}, \ i = 1, 2, 3, \dots, n\}$, a *standard (homoscedastic) GPRM* can be formulated as follows:

$$y = f(\mathbf{x}) + \epsilon \tag{2}$$

where a GP prior with mean $m(\mathbf{x})$ and covariance $k(\mathbf{x}, \mathbf{x}')$ is placed on the latent function $f(\mathbf{x})$, and an independent, identically-distributed (*iid*), zero-mean Gaussian prior with variance σ_n^2 is placed on the noise term ϵ, that is:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \quad \epsilon \sim iid \ \mathcal{N}(0, \sigma_n^2) \tag{3}$$

As is common in the GPRM literature, $m(\mathbf{x})$ is set to zero, and the squared exponential covariance function (kernel) is used for the latent function GP, owing to its ability to monotonically decrease as input values go farther from each other, which allows for similar latent function values for close input points, and vice versa:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Lambda^{-1} (\mathbf{x} - \mathbf{x}')\right) \tag{4}$$

In equation 4, σ_0^2 is the output variance, and Λ^{-1} is the inverse of a diagonal matrix of the characteristic input length scales corresponding to each dimension (D i.e each covariate) in the input data. For a single-input dimension (i.e. $D = 1$), the entries along the diagonal of Λ^{-1} will be identical; otherwise, there will be a separate input length scale for every covariate in the training input data.

Training. Training of the GPRM involves optimizing the hyperparameters ($\theta \equiv \sigma_0^2, \Lambda, \sigma_n^2$), which is typically done *via* Type II Maximum Likelihood [2, Chapter 5, pp. 109], whereas the marginal likelihood (evidence) of the training observations is maximized (or its negative log is minimized for reasons related to computational stability). That is, the following expression is minimized with respect to θ:

$$-\log p(\mathbf{y}|X, \theta) = -\log \mathcal{N}(\mathbf{y}|\mathbf{0}, K_{XX} + \sigma_n^2 \mathbb{I}) \tag{5a}$$

$$= -\frac{1}{2}\mathbf{y}^T(K_{XX} + \sigma_n^2 \mathbb{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K_{XX} + \sigma_n^2 \mathbb{I}| - \frac{n}{2}\log 2\pi \tag{5b}$$

In the expression above, K_{AB} denotes $K(A, B)$ (covariance matrix), and $\mathbb{I}$ the identity matrix

Prediction. Prediction can be done by assuming joint Gaussian distribution between the training observations ($\mathbf{y}$), and a test observation (y_* - to be predicted) at the set of test inputs ($\mathbf{x}_*$) as follows:

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} = \mathcal{N}\left[\mathbf{0}, \begin{matrix} K_{XX} + \sigma_n^2 \mathbb{I} & \mathbf{k}_{X\mathbf{x}_*} \\ \mathbf{k}_{\mathbf{x}_* X} & k_{\mathbf{x}_* \mathbf{x}_*} + \sigma_n^2 \mathbb{I} \end{matrix}\right] \tag{6}$$

where $\mathbf{k}_{X\mathbf{x}_*}$ is the vector of covariances between X and $\mathbf{x}_*$. By invoking the properties of multivariate Gaussian distributions [3], the predictive distribution over y_* can be defined as follows:

$$p(y_*|\mathbf{x}_*, X, \mathbf{y}) = \mathcal{N}(\mathbb{E}\{y_*\}, \mathbb{V}\{y_*\}) \tag{7a}$$

$$\mathbb{E}\{y_*\} = \mathbf{k}_{\mathbf{x}_* X}(K_{XX} + \sigma_n^2 \mathbb{I})^{-1}\mathbf{y} \tag{7b}$$

$$\mathbb{V}\{y_*\} = k_{\mathbf{x}_* \mathbf{x}_*} - \mathbf{k}_{\mathbf{x}_* X}(K_{XX} + \sigma_n^2 \mathbb{I})^{-1}\mathbf{k}_{X\mathbf{x}_*} + \sigma_n^2 \tag{7c}$$

such that $\mathbb{E}\{y_*\}$ and $\mathbb{V}\{y_*\}$ are the predictive mean and variance, respectively, at the set of test inputs.

Variational Heteroscedastic Gaussian Process Regression Models (VHGPRMs)

One of the inherent drawbacks of using standard (homoscedastic) GPRMs is the assumption of a fixed noise variance throughout the input space, which, in many real-life applications, is impractical. Thus, several modifications have been put forward to allow for the noise variance to vary with the input (that is, an input-dependent noise variance) [4, 3]. This is to say that the GPRM formulation in Equation 2 would become:

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

with the noise prior defined as

$$\epsilon \sim \mathcal{N}(0, r(\mathbf{x})) \tag{9}$$

This new formulation takes the name of *variational heteroscedastic GPRMs (VHGPRMs)* and has many advantages over GPRMs, such as the variance of the test observation can be predicted along with the test observation itself analytically as noted by Rogers et al. [3]. Although, VHGPRMs allow more flexibility they are also harder to train as no analytical form exists for the integral at the prediction stage.

One of the most common strategies is to treat the input-dependent noise as a GP itself, which was first put forward by Goldberg *et al.* [5], that is

$$r(\mathbf{x}) = \exp(g(\mathbf{x})) \tag{10a}$$

$$g(\mathbf{x}) \sim \mathcal{GP}(\mu_0, k_g(\mathbf{x}, \mathbf{x}')) \tag{10b}$$

such that μ_0 and $k_g(\mathbf{x}, \mathbf{x}')$ are the mean and covariance for the GP prior on the noise variance function, respectively, where an exponential function is used in order to ensure that the noise variance stays positive [6]. It is worth noting that the subscript g was introduced to differentiate between the covariance function of the noise GP and that of the noise-free process. At this point, it is useful to introduce shorthand notations for the covariance functions as follows:

$$K_j(X, X) \equiv K_j$$
$$K_j(\mathbf{x}_* X) \equiv K_{j*}$$
$$K_j(\mathbf{x}_* \mathbf{x}_*) \equiv K_{j**}$$

Such that j can be f or g. Although the formulation in equation 10a provides a better treatment of data with hetersocedastic noise, the added complexity results in making the marginal likelihood and the predictive distribution over unknown observations not analytically-tractable. One of the proposed approaches to approximate them was put forward by Lazaro-Gredilla and Titsias [6], which is based on variational approximations. Briefly, their approach is based on approximating $p(f, g|D)$ by $q(f)q(g)$ *via* the minimization of the Kullback-Leibler divergence between them, where the $q(f)$ and $q(g)$ are the variational probability densities (arbitrary density functions) over sets of f and g, respectively. The resulting marginal variational (MV) bound (M) becomes:

$$M(, \Sigma) = \log \mathcal{N}(\mathbf{y}|\mathbf{0}, K_f + R) - \frac{1}{4} \text{tr}(\Sigma) - \text{KL}(\mathcal{N}(g|, \Sigma)||\mathcal{N}(g|\mu_0 \mathbf{1}, K_g)) \tag{11}$$

In equation 11, the mean and covariance Σ come from the restricting $q(g)$ to be $\mathcal{N}(g|, \Sigma)$, the KL($\cdot$) term is the Kullback Leibler divergence between the GP prior on g and the aforementioned restriction on $q(g)$, tr($\cdot$) denotes the trace of the enclosed matrix, $\mathbf{1}$ indicates a vector of ones, and R is a diagonal matrix with elements:

$$R_{ii} = \exp(_i - \frac{\Sigma_{ii}}{2}) \tag{12}$$

where i is as defined before.

Training. In equation 11, the number of free parameters to be determined becomes n+n(n+1)/2, which would make the training process much more computationally exhaustive. Thus, Lazaro-Gredilla and Titsias [6] proposed a reparametrization of and Σ at the maxima of the marginal variational bound into the following:

$$= K_g \left(\Lambda - \frac{1}{2} \mathbb{I} \right) \mathbf{1} + \mu_0 \mathbf{1}, \qquad \Sigma^{-1} = K_g^{-1} + \Lambda \tag{13}$$

such that Λ is a positive semi-definite diagonal matrix of the variational parameters (to be determined through optimization).

Prediction. Based on the formulation presented by Lazaro-Gredilla and Titsias [6], the predictive distribution for a new test point y_* can be simplified to:

$$q(y_*) = \int\int p(y_*|f_*, g_*)q(f_*)q(g_*)df_* dg_*$$
$$= \int \mathcal{N}(y_*|a_*, c_*^2 + \exp(g_*))\mathcal{N}(g_*|\mu_*, \sigma_*^2)dg_* \tag{14}$$

with:

$$a_* = \mathbf{k}_{f*}(K_f + R)^{-1}\mathbf{y} \tag{15a}$$

$$c_*^2 = k_{f**} - \mathbf{k}_{f*}^T(K_f + R)^{-1}\mathbf{k}_{f*} \tag{15b}$$

$$\mu_* = \mathbf{k}_{g*}^T(\Lambda - \frac{1}{2}\mathbb{I})\mathbf{1} + \mu_0 \tag{15c}$$

$$\sigma_*^2 = k_{g**} - \mathbf{k}_{g*}^T(K_g + \Lambda^{-1})^{-1}\mathbf{k}_{g*} \tag{15d}$$

3

Although the integration above does not have an analytical solution, the first two moments of the predictive distribution can be calculated analytically as follows:

$$\mathbb{E}\{y_*|\mathbf{x}_*, \mathscr{D}\} = a_* \tag{16a}$$

$$\mathbb{V}\{y_*|\mathbf{x}_*, \mathscr{D}\} = c_*^2 + \exp(\mu_* + \frac{\sigma_*^2}{2}) \tag{16b}$$

Thus, under the assumption of a Gaussian predictive distribution over the unknown test observation y_*, which is not necessarily true, the variance of the test observation can be predicted along with the test observation itself, which allows for a more flexible model compared to standard GPRMs [3].

$$p(y_*|X, \mathbf{y}, \mathbf{x}_*) = p(f_*|X, \mathbf{y}, \mathbf{x}_*) = int \, p(f_*|X, \mathbf{x}_*, \mathbf{f})p(\mathbf{f}|X, \mathbf{y}) \tag{17a}$$

$$p(\mathbf{f}|X, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)}{p(\mathbf{y}|X)} \tag{17b}$$

$$\sigma(f(\mathbf{x})) = p(y = +1|\mathbf{x}) = \frac{1}{1 + \exp(-f(\mathbf{x}))} \tag{17c}$$

3   Observations on w

We present a series of observations regarding the connection of τ and z with coverage, error, accuracy and quality as metrics.

Observation 1 *Relation between coverage and z.* *Given a collective-Gaussian stall model $M_{stall} = \langle \Theta_{states}, \{\mathcal{N}(\mu_\theta, \sigma_\theta^2)\}, P_{states}, L_{stall}, C_{stall} \rangle$, where $\mathcal{N}(\mu_\theta, \sigma_\theta^2)$ are univariate normal distributions, and $\tau = 0.5$, the area covered by the prediction interval defined by a z-score of z is smaller than or equal to the coverage of M_{stall} with the same z. Equivalently, $pred(\mathcal{N}(\mu, \sigma^2), z) \leq coverage(M_{stall}, S_{SE/no\text{-}stall}, S_{SE/stall}, (z, 0.5), x)$ for any M_{stall}, where $pred(\mathcal{N}(\mu, \sigma^2), z) = cdf_Z(z) - cdf_Z(-z)$ with cdf_Z as the cumulative distribution function for the standard normal distribution.*

This relation comes from the fact that the region defined by z-predictability is the union of multiple prediction intervals and thus the z-predictability region covers a bigger region for each distribution. The inequality is strict unless all distributions have the same parameters μ and σ^2.

Notice that for very large values of z, the difference between coverage and the prediction interval defined by z shrinks to a negligible value. Assuming a very large z, we can analyze the effect that τ has on the safety envelope and the metrics as if z played no role. This leads us to:

Observation 2 *Connection between τ and w for model quality.* *Given a model M_{stall} and a weight error of w, w determines a unique optimum τ which maximizes the model quality metric. As the value of w increases so will the optimal τ. Conversely, a value of τ will force a weight w for which τ is the optimum. The higher the τ, the more the error will be weighted and thus the more we will care about the error.*

To show some evidence for the observation, we analyze a simple M_{stall} model. Let us assume a model composed of only two normal distributions, each tagged with blue and red, and parameters $\sigma_{red}^2 = \sigma_{blue}^2 = 1$ and $\mu_{red} = -\mu_{blue}$, respectively. Under these conditions, a μ of 1 means that the two identical distributions are separated by 2σ. We can analytically compute all the metrics as: $accuracy = cdf_{blue}(x_{blue})$, $error = cdf_{red}(x_{blue})$ and $quality = accuracy \times (1 - error)^w$.

where w is the weight parameter for the error, and x_{blue} is the point to the right of the blue distribution where the probability of blue given the input is blue with probability τ. It can be found that $x_{blue} = \frac{1}{2\mu}(ln(1 - \tau) - ln(\tau))$. The objective is to find the optimal τ for a given w, which means computing the derivative of *quality* and finding its roots:

$$\frac{d}{d\tau}cdf_{blue}(x_{blue})(1 - cdf_{red}(x_{blue}))^w = 0$$

$$pdf_{blue}(x_{blue})(1 - cdf_{red}(x_{blue}))^w = cdf_{blue}(x_{blue})w(1 - cdf_{red}(x_{blue}))^{w-1}pdf_{red}(x_{blue})$$

It is impossible to compute analytically the roots of this expression. We have computed it numerically for a range of different w. Figure 1 shows the correlation between a value of w and the optimal τ that it forces for different separations. Notice that a μ of 0.1 means that the red distribution is a normal distribution centered in 0.1 and the blue distribution is centered in -0.1 (a 0.2σ separation), i.e., it
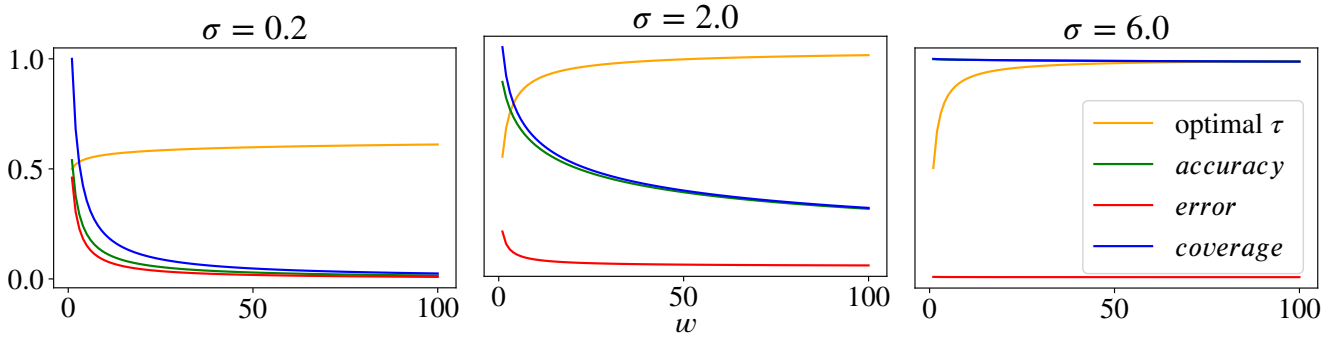
Figure 1: τ, accuracy, error, and coverage as functions of $w(1 \leq w)$ for models with differing data separation. $\sigma = 0.2$ means a separation of 0.2σ between guassian distributions.

is very difficult to discern where a point could come from any of the two distributions. The accuracy and error are in the same order of magnitude and close to 50% when $\tau = 0.5$.

Notice that w does not force a specific error (with the same w, we can arrive at different error values depending on the model). The behaviour of τ for $\sigma = 2$ and $\sigma = 6$ is very similar, but the error is considerably different. What w does is to define a τ where the models are somewhat separable or clearly separable.

With $\tau = 0.5$, accuracy and error are the highest, which leads us to:

Observation 3 *The error when $\tau = 0.5$ **and z is large gives a good estimation of the inherent quality of a model.** When $\tau = 0.5$, we obtain the best classical Bayesian classifier possible (nothing lies outside the safety envelope region). After computing the error we are in two possible conditions: if the error is large, i.e., close to 0.5, the model is not good, and it is difficult to have a good safety envelope; and, if the error is small the model is a good model and has safety envelopes of quality.*

Figure 2 shows how the error can give a fair evaluation of how separable the data is. For distributions that are close to each other, thus not highly separable, the error is high. The error almost fades away with highly separable distributions, thus good models.
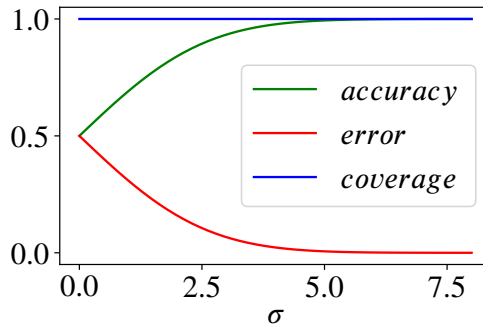
Figure 2: Error and accuracy as functions of σ (separation between identical gaussian distribution) when $\tau = 0.5$.

As a rule of thumb, after setting $\tau = 0.5$ and computing the error, we can make the following judgment in the quality of a model: if the error is small, close to 0, then the data is separable; if on the contrary, it is close to 0.5, it is highly non-separable, a bad data-driven model.

4 Agda Code

Agda allows for Unicode characters to be part of the names of functions and expressions. This flexibility means that Agda source code looks quite pretty in the console, but it might make it difficult for someone to type it out. Some care must be taken if one wants to copy the code present in this section into a plain text document. The code is presented for those wishing to have some guidance on each module and its purpose.

4.1 `Avionics/SafetyEnvelopes/Properties.agda`

This document contains the four proofs presented in the main article on safety envelopes. This file depends on all the others under `Avionics/` except for `ExtInterface.agda`.

```
module Avionics.SafetyEnvelopes.Properties where

open import Data.Bool using (Bool; true; false; _∧_; T)
open import Data.Empty using (⊥; ⊥-elim)
open import Data.List as List using (List; []; _∷_; any)
open import Data.List.Relation.Unary.Any as Any using (Any; here; there; satisfied)
open import Data.Maybe using (Maybe; just; nothing; is-just; _»=_)
open import Data.Product as Prod using (∃-syntax; _×_; proj₁; proj₂) renaming (_,_ to ⟨_,_⟩)
open import Data.Sum using (_∪_; inj₁; inj₂)
open import Function using (_∘_)
open import Relation.Binary.PropositionalEquality as Eq using (_≡_; refl; cong; cong₂; inspect; [_]; sym; trans)
open Eq.≡-Reasoning using (begin_; _≡⟨⟩_; _≡˘⟨_⟩_; _≡⟨_⟩_; _∎)
open import Relation.Nullary using (Dec; yes; no; ¬_)
open import Relation.Nullary.Decidable using (toWitness; fromWitness)
open import Relation.Unary using (_∈_)

open import Avionics.Bool using (≡→T; T∧→×; ×→T∧; lem∧)
open import Avionics.List using (≡→any; any-map; any-map-rev; any→≡)
open import Avionics.Real
    using (ℝ; _+_; _-_; -_; _*_; _÷_; _^_; _<ᵇ_; _≤_; _<_; _<?_; _≤?_; _≢0;
           0ℝ; 1ℝ; 2ℝ; 1/2; abs; 1/_; _^2; √_; fromℕ;
           double-neg;
           ⟨0,∞⟩; [0,∞);
           <-transˡ; 2>0; ⟨0,∞⟩→0<; 0<→⟨0,∞⟩; >0→≢0; >0→≥0;
           0≟0≡yes0≡0;
           abs<x→<x∧-x<; neg-involutive; neg-distrib-+; neg-mono-<->; neg-def; m-m≡0;
           +-comm; +-assoc; *-comm; *-assoc; +-monoˡ-<; m÷n<o≡m<o*n; m<o÷n≡m*n<o; neg-distribˡ-*;
           √x^2≡absx; x*x≡x^2; x^2*y^2≡⟨xy⟩^2; 1/x^2≡⟨1/x⟩^2)
open import Avionics.Probability using (NormalDist; Dist)
open import Avionics.SafetyEnvelopes
    using (inside; inside'; mahalanobis1; z-predictable'; P[_|X=_]_; classify''; classify; M→pbs;
           StallClasses; Stall; NoStall; Uncertain;
           no-uncertain;
           safety-envelope; z-predictable; Model; τ-confident;
           Stall≡1-NoStall; NoStall≡1-Stall; ≤p→¬≤1-p; ≤1-p→¬≤p
           )

open NormalDist using (σ; μ)

--<ᵇ→< : ∀ {x y} → T (x <ᵇ y) → x < y
--<ᵇ→< = toWitness

-- Preliminary defitinions

-- `pi` is the prediction interval for the z score, i.e.,
-- pi(N (μ, σ), z) = [μ − zσ, μ + zσ]
pi : NormalDist → ℝ → ℝ → Set
pi nd z x =     (μ nd) - z * (σ nd) < x
            × x < (μ nd) + z * (σ nd)

extractDists : Model → List NormalDist
extractDists M = List.map (proj₁ ∘ proj₂) (Model.fM M)
```

```
--------------------------- Starting point - Theorem 1 ---------------------------
-- Theorem 1 says:
-- In the case of univariate normal distributions, the z-predictability condition
-- > mahalanobis(μ, x, σ²⁻¹) < z
-- reduces to
-- > μ - zσ < x <ᵇ (μ + zσ)
-- the prediction interval with a z-score of z
z-pred-interval≡mahalanobis<z : ∀ (nd z x)
                                    → inside nd z x ≡ inside' nd z x
z-pred-interval≡mahalanobis<z nd z x =
  begin
    inside nd z x
  ≡⟨⟩
    (u - z * s <ᵇ x) ∧ (x <ᵇ (u + z * s))
  ≡⟨ cong (λ e → ((u - z * s) <ᵇ x) ∧ (x <ᵇ e)) (
    begin
      u + z * s
    ≡˘⟨ cong₂ (_+_) (neg-involutive _) (neg-involutive _) ⟩
      -(- u) + -(-(z * s))
    ≡˘⟨ neg-distrib-+ _ _ ⟩
      -((- u) + (- (z * s)))
    ≡⟨⟩
      -(- u - z * s)
    ∎
  ) ⟩
    (u - z * s <ᵇ x) ∧ (x <ᵇ -(- u - z * s))
  ≡⟨ cong (λ e → (e <ᵇ x) ∧ (x <ᵇ -(- u - z * s))) (
    begin
      u - z * s
    ≡˘⟨ cong (_+ (- (z * s))) (neg-involutive _) ⟩
      -(- u) + (- (z * s))
    ≡˘⟨ neg-distrib-+ _ _ ⟩
      -(- u + z * s)
    ∎
  ) ⟩
    (-(- u + z * s) <ᵇ x) ∧ (x <ᵇ -(- u - z * s))
  ≡˘⟨ cong (λ e → (-(- u + z * s) <ᵇ e) ∧ (e <ᵇ -(- u - z * s))) (neg-involutive _) ⟩
    (-(- u + z * s) <ᵇ -(- x)) ∧ (-(- x) <ᵇ -(- u - z * s))
  ≡˘⟨ cong ((-(- u + z * s) <ᵇ -(- x)) ∧_) (neg-mono-<-> _ _) ⟩
    (-(- u + z * s) <ᵇ -(- x)) ∧ (- u - z * s <ᵇ - x)
  ≡˘⟨ cong (_∧ (- u - z * s <ᵇ - x)) (neg-mono-<-> _ _) ⟩
    (- x <ᵇ - u + z * s) ∧ (- u - z * s <ᵇ - x)
  ≡⟨ cong (_∧ (- u - z * s <ᵇ - x)) (
    begin
      - x <ᵇ - u + z * s
    ≡˘⟨ cong (_<ᵇ - u + z * s) (neg-def _) ⟩
      0ℝ - x <ᵇ - u + z * s
    ≡˘⟨ cong (λ e → e - x <ᵇ - u + z * s) (trans (+-comm _ _) (m-m≡0 _)) ⟩
      (- u + u) - x <ᵇ - u + z * s
    ≡˘⟨ cong (_<ᵇ - u + z * s) (+-assoc _ _ _) ⟩
      (- u) + (u - x) <ᵇ - u + z * s
    ≡˘⟨ +-monoˡ-< _ _ _ ⟩
```

7

$u - x <^{b} z * s$

∎

) ⟩

$(u - x <^{b} z * s) \wedge (- u - z * s <^{b} - x)$

≡⟨ cong $((u - x <^{b} z * s) \wedge\_)$ (

begin

$- u - z * s <^{b} - x$

≡˘⟨ cong $(- u - z * s <^{b}\_)$ (neg-def $\_$) ⟩

$- u - z * s <^{b} 0ℝ - x$

≡˘⟨ cong $(\lambda e \to - u - z * s <^{b} e - x)$ (trans (+-comm $\_$ $\_$) (m-m≡0 $\_$)) ⟩

$- u - z * s <^{b} (- u + u) - x$

≡˘⟨ cong $(- u - z * s <^{b}\_)$ (+-assoc $\_$ $\_$ $\_$) ⟩

$- u - z * s <^{b} - u + (u - x)$

≡˘⟨ +-mono^{l}-< $\_$ $\_$ $\_$ ⟩

$- (z * s) <^{b} u - x$

∎

) ⟩

$(u - x <^{b} z * s) \wedge (- (z * s) <^{b} u - x)$

≡˘⟨ cong $((u - x <^{b} z * s) \wedge\_)$ (

begin

$- z <^{b} (u - x) \div s$

≡⟨ m<o÷n≡m*n<o $\_$ $\_$ $\_$ ⟩

$(- z) * s <^{b} (u - x)$

≡˘⟨ cong $(\_<^{b} (u - x))$ (neg-distrib^{l}-* $\_$ $\_$) ⟩

$- (z * s) <^{b} u - x$

∎

) ⟩

$(u - x <^{b} z * s) \wedge (- z <^{b} (u - x) \div s)$

≡˘⟨ cong $(\_\wedge (- z <^{b} (u - x) \div s))$ (m÷n<o≡m<o*n $\_$ $\_$ $\_$) ⟩

$((u - x) \div s <^{b} z) \wedge (- z <^{b} (u - x) \div s)$

≡˘⟨ abs<x→<x∧-x< ⟩

abs $((u - x) \div s) <^{b} z$

≡˘⟨ cong $(\_<^{b} z)$ (√x^2≡absx $\_$) ⟩

$√(((u - x) \div s)\hat{\ }2) <^{b} z$

≡⟨⟩

$√(((u - x) * (1/ s))\hat{\ }2) <^{b} z$

≡˘⟨ cong $(\lambda e \to √ e <^{b} z)$ (

begin

$(u - x) * (1/ (s * s)) * (u - x)$

≡⟨ *-comm $\_$ $\_$ ⟩

$(u - x) * ((u - x) * (1/ (s * s)))$

≡⟨ *-assoc $\_$ $\_$ $\_$ ⟩

$(u - x) * (u - x) * (1/ (s * s))$

≡⟨ cong $(\lambda e \to (u - x) * (u - x) * (1/ e))$ (x*x≡x^2 $\_$) ⟩

$(u - x) * (u - x) * (1/ (s \hat{\ }2))$

≡⟨ cong $((u - x) * (u - x) *\_)$ (1/x^2≡(1/x)^2 $\_$) ⟩

$(u - x) * (u - x) * (1/ s) \hat{\ }2$

≡⟨ cong $(\_* (1/ s) \hat{\ }2)$ (x*x≡x^2 $\_$) ⟩

$(u - x)\hat{\ }2 * (1/ s) \hat{\ }2$

≡⟨ x^2*y^2≡(xy)^2 $\_$ $\_$ ⟩

$((u - x) * (1/ s))\hat{\ }2$

∎

$)\rangle$

$\quad \sqrt{((u - x) * (1/ (s * s)) * (u - x))} <^b z$

$\equiv\langle\rangle$

$\quad$ mahalanobis1 u x $(1/ (s * s)) <^b z$

$\equiv\langle\rangle$

$\quad$ inside' $nd\ z\ x$

■

$\quad$ where u $= \mu\ nd$

$\quad\quad\quad$ s $= \sigma\ nd$

```
---- ########### Theorem 1 END ###########

--------------------------- Starting point - Theorem 2 ---------------------------
-- Proof of Theorem 2 (paper)
--
-- In words, it says that:
-- The energy signal x is z-predictable iff there exist ⟨α, v⟩ s.t.
-- M(⟨α, v⟩)1 = di and x ∈ pi(di , z).
--
-- Notice that `Any (λ nd → x ∈ pi nd z) nds` translates to:
-- there exists nd such that `nd ∈ nds` and `x ∈ pi(nd, z)`
```

follows-def←' : ∀ $(nds\ z\ x)$

$\quad\quad\quad\quad\quad$ → z-predictable' $nds\ z\ x \equiv \langle\ x\ ,$ true $\rangle$

$\quad\quad\quad\quad\quad$ → Any $(\lambda\ nd → x ∈$ pi $nd\ z)\ nds$

follows-def←' $nds\ z\ x\ res{\equiv}x,true$ = Any-x∈pi

$\quad$ where

$\quad\quad$ res≡true = cong $\mathsf{proj}_2\ res{\equiv}x,true$

```
   -- the first `toWitness` takes a result `(μ nd - z * σ nd) <ᵇ x` (a
   -- boolean) and produces a proof of the type `(μ nd) - z * (σ nd) < x`
   -- assuming we have provided an operator `<?`
```

$\quad\quad$ toWitness' = $\lambda\ nd →$ Prod.map (toWitness $\{Q = (\mu\ nd - z * \sigma\ nd) <?\ x\})$

$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad$ (toWitness $\{Q = x <? (\mu\ nd + z * \sigma\ nd)\})$

```
   -- We find the value for which `inside z x` becomes true in the list `nds`
```

$\quad\quad$ Any-bool = ≡→any $(\lambda\ nd →$ inside $nd\ z\ x)\ nds$ res≡true

```
   -- Converting the boolean proof into a proof at the type level
```

$\quad\quad$ Any-x∈pi = Any.map $(\lambda\ \{nd\} →$ toWitness' $nd ∘$ T∧→×) Any-bool

```
-- forward proof
```

follows-def→' : ∀ $(nds\ z\ x)$

$\quad\quad\quad\quad\quad$ → Any $(\lambda\ nd → x ∈$ pi $nd\ z)\ nds$

$\quad\quad\quad\quad\quad$ → z-predictable' $nds\ z\ x \equiv \langle\ x\ ,$ true $\rangle$

follows-def→' $nds\ z\ x\ any[x{∈}pi{-}z]nds$ = let

```
   -- converts a tuple of `(μ nd) - z * (σ nd) < x , x < (μ nd + z * σ nd)`
   -- (a proof) into a boolean
```

$\quad$ *fromWitness'* $nd = \lambda\{\langle\ \mu{-}z\sigma{<}x\ ,\ x{<}\mu{+}z\sigma\ \rangle →$

$\quad\quad$ ×→T∧ $\langle$ (fromWitness $\{Q = (\mu\ nd - z * \sigma\ nd) <?\ x\}\ \mu{-}z\sigma{<}x)$

$\quad\quad\quad\quad\quad\quad$, (fromWitness $\{Q = x <? (\mu\ nd + z * \sigma\ nd)\}\ x{<}\mu{+}z\sigma)$

$\quad\quad\quad\quad\quad\quad\rangle\}$

```
   -- Converting from a proof on `_<_` to a proof on `_<ᵇ_`
```

$\quad$ *any[inside]nds* = (Any.map $(\lambda\ \{nd\} → $ *fromWitness'* $nd)\ any[x{∈}pi{-}z]nds)$

```
   -- Transforming the prove from Any into equality (_≡_)
```

$\quad$ *z-pred-x≡⟨x,true⟩* = any→≡ $(\lambda\ nd →$ inside $nd\ z\ x)\ nds$ *any[inside]nds*

```
     -- Extending the result from a single Bool value to a pair `ℝ × Bool`
```

9

```
    in cong (⟨ x ,_⟩) z-pred-x≡⟨x,true⟩

-- From the prove above we can obtain the value `nd` and its prove `x ∈ pi nd z`
-- Note: An element of ∃[ nd ] (x ∈ pi nd z) is a tuple of the form ⟨ nd , proof ⟩
--prop2← : ∀ (nds z x)
--  → z-predictable' nds z x ≡ ⟨ x , true ⟩
--  → ∃[ nd ] (x ∈ pi nd z)
--prop2← nds z x res≡x,true = satisfied (follows-def←' nds z x res≡x,true)

-- This proofs is telling us that `z-predictable` follows from the definition
follows-def← : ∀ (M z x)
              → z-predictable M z x ≡ ⟨ x , true ⟩
              → Any (λ nd → x ∈ pi nd z) (extractDists M)
follows-def← M z x res≡x,true = follows-def←' (extractDists M) z x res≡x,true

follows-def→ : ∀ (M z x)
              → Any (λ nd → x ∈ pi nd z) (extractDists M)
              → z-predictable M z x ≡ ⟨ x , true ⟩
follows-def→ M z x Any[x∈pi-nd-z]M = follows-def→' (extractDists M) z x Any[x∈pi-nd-z]M

-- ############ FINAL RESULT - Theorem 2 ############

-- In words: Given a Model `M` and parameter `z`, if `x` is z-predictable, then
-- there exists θ (a flight state) such that they are associated to a `nd`
-- (Normal Distribution) and `x` falls withing the Predictable Interval
theorem2← : ∀ (M z x)
              → z-predictable M z x ≡ ⟨ x , true ⟩
              → Any (λ{⟨ θ , ⟨ nd , p ⟩ ⟩ → x ∈ pi nd z}) (Model.fM M)
theorem2← M z x res≡x,true = any-map (proj₁ ∘ proj₂) (follows-def← M z x res≡x,true)

-- The reverse of theorem2←
theorem2→ : ∀ (M z x)
              → Any (λ{⟨ θ , ⟨ nd , p ⟩ ⟩ → x ∈ pi nd z}) (Model.fM M)
              → z-predictable M z x ≡ ⟨ x , true ⟩
theorem2→ M z x Any[θ→x∈pi-nd-z]M = follows-def→ M z x (any-map-rev (proj₁ ∘ proj₂) Any[θ→x∈pi-nd-z]M)

-- ################ Theorem 2 END ################

---------------------------- Starting point - Theorem 3 ----------------------------
lem← : ∀ (pbs τ x k)
    → classify'' pbs τ x ≡ k
    → k ≡ Uncertain ∪ ∃[ p ] ((P[ k |X= x ] pbs ≡ just p) × (τ ≤ p))
lem← pbs τ x k _ with P[ Stall |X= x ] pbs | inspect (P[ Stall |X=_] pbs) x
lem← _ _ _ Uncertain _ | nothing | [ P[k|X=x]≡nothing ] = inj₁ refl
lem← _ τ _ _         _ | just p    | [ _ ] with τ ≤? p | τ ≤? (1ℝ - p)
lem← _ _ _ Stall     _ | just p    | [ P[k|X=x]≡justp ] | yes τ≤p | no ¬τ≤1-p = inj₂ ⟨ p , ⟨ P[k|X=x]≡justp , τ≤p ⟩ ⟩
lem← _ _ _ NoStall _ | just p    | [ P[k|X=x]≡justp ] | no ¬τ≤p | yes τ≤1-p =
  let P[NoStall|X=x]≡just1-p = Stall≡1-NoStall P[k|X=x]≡justp
  in inj₂ ⟨ 1ℝ - p , ⟨ P[NoStall|X=x]≡just1-p , τ≤1-p ⟩ ⟩
lem← _ _ _ Uncertain _ | _        | _ | _ | _ = inj₁ refl

lem→' : ∀ (pbs τ x p)
    -- This line is asking for the main assumptions for the code to work properly:
    -- * 0.5 < τ ≤ 1
    -- * 0 ≤ p ≤ 1
    → (1/2 < τ × τ ≤ 1ℝ) × (0ℝ ≤ p × p ≤ 1ℝ)
    → (P[ Stall |X= x ] pbs) ≡ just p
    → τ ≤ (1ℝ - p)
```

10

```agda
                      → classify″ pbs τ x ≡ NoStall
lem→' pbs _ x _ _ _ _ _ with P[ Stall |X= x ] pbs
lem→' _ τ _ _ _ _ _ _ | just p with τ ≤? p | τ ≤? (1ℝ - p)
lem→' _ _ _ _ _ _ _ _ | just p | no _    | yes _ = refl
lem→' _ _ _ _ _ refl τ≤1-p | just p | _ | no ¬τ≤1-p = ⊥-elim (¬τ≤1-p τ≤1-p)
lem→' _ _ _ _ assumpts refl τ≤1-p | just p | yes τ≤p | yes _ = ⊥-elim (¬τ≤p τ≤p)
  where
    1/2<τ = proj₁ (proj₁ assumpts)
    τ≤1 = proj₂ (proj₁ assumpts)
    0≤p = proj₁ (proj₂ assumpts)
    p≤1 = proj₂ (proj₂ assumpts)
    ¬τ≤p = ≤1-p→¬≤p 1/2<τ τ≤1 0≤p p≤1 τ≤1-p

τ≤p→τ≤1-⟨1-p⟩ : ∀ τ p → τ ≤ p → τ ≤ 1ℝ - (1ℝ - p)
τ≤p→τ≤1-⟨1-p⟩ τ p τ≤p rewrite double-neg p 1ℝ = τ≤p

lem→ : ∀ (pbs τ x k)
        → (1/2 < τ × τ ≤ 1ℝ)
        → ∃[ p ] (((P[ k |X= x ] pbs) ≡ just p) × (τ ≤ p))
        → classify″ pbs τ x ≡ k
lem→ pbs _ x Stall _ _ with P[ Stall |X= x ] pbs
lem→ _ τ _ _ _     _ | just p with τ ≤? p | τ ≤? (1ℝ - p)
lem→ _ _ _ _ _ _     _ | just p | yes _            | no _ = refl
lem→ _ _ _ _ _ _     ⟨ _ , ⟨ refl , τ≤p ⟩ ⟩ | just p | no ¬τ≤p | _      = ⊥-elim (¬τ≤p τ≤p)
lem→ _ _ _ _ 1/2<τ≤1 ⟨ _ , ⟨ refl , τ≤p ⟩ ⟩ | just p | yes _ | yes τ≤1-p = ⊥-elim (¬τ≤1-p τ≤1-p)
  where 1/2<τ = proj₁ 1/2<τ≤1
        τ≤1 = proj₂ 1/2<τ≤1
        postulate 0≤p : 0ℝ ≤ p
                  p≤1 : p ≤ 1ℝ
        ¬τ≤1-p = ≤p→¬≤1-p 1/2<τ τ≤1 0≤p p≤1 τ≤p
lem→ pbs τ x NoStall 1/2<τ≤1 ⟨ p , ⟨ P[k|X=x]≡justp , τ≤p ⟩ ⟩ = let
    P[S|X=x]≡just1-p = NoStall≡1-Stall P[k|X=x]≡justp
    τ≤1-⟨1-p⟩ = τ≤p→τ≤1-⟨1-p⟩ τ p τ≤p
    assumptions = ⟨ 1/2<τ≤1 , 0≤p≤1 ⟩
  in lem→' pbs τ x (1ℝ - p) assumptions P[S|X=x]≡just1-p τ≤1-⟨1-p⟩
  where postulate 0≤p≤1 : (0ℝ ≤ 1ℝ - p × 1ℝ - p ≤ 1ℝ)

prop3M-prior← : ∀ (M τ x k)
                → classify M τ x ≡ k
                → k ≡ Uncertain ∪ ∃[ p ] (((P[ k |X= x ] (M→pbs M)) ≡ just p) × (τ ≤ p))
prop3M-prior← M = lem← (M→pbs M)

prop3M-prior←' : ∀ (M τ x k)
                → k ≡ Stall ∪ k ≡ NoStall
                → classify M τ x ≡ k
                → ∃[ p ] (((P[ k |X= x ] (M→pbs M)) ≡ just p) × (τ ≤ p))
prop3M-prior←' M τ x k _ cMτx≡k with prop3M-prior← M τ x k cMτx≡k
prop3M-prior←' _ _ _ Stall (inj₁ _) _ | inj₂ P[k|X=x]≥τ = P[k|X=x]≥τ
prop3M-prior←' _ _ _ NoStall _ _ | inj₂ P[k|X=x]≥τ = P[k|X=x]≥τ

prop3M-prior→ : ∀ (M τ x k)
                → (1/2 < τ × τ ≤ 1ℝ)
                → ∃[ p ] (((P[ k |X= x ] (M→pbs M)) ≡ just p) × (τ ≤ p))
                → classify M τ x ≡ k
prop3M-prior→ M τ x k 1/2<τ≤1 = lem→ (M→pbs M) τ x k 1/2<τ≤1

prop3M← : ∀ (M τ x)
```

```
          → τ-confident M τ x ≡ true
          → ∃[ k ] ((classify M τ x ≡ k) × (k ≡ Stall ∪ k ≡ NoStall))
prop3M← M τ x τconf≡true with classify M τ x
... | Stall      = ⟨ Stall , ⟨ refl , inj₁ refl ⟩ ⟩
... | NoStall = ⟨ NoStall , ⟨ refl , inj₂ refl ⟩ ⟩

prop3M→ : ∀ (M τ x k)
          → k ≡ Stall ∪ k ≡ NoStall
          → classify M τ x ≡ k
          → τ-confident M τ x ≡ true
prop3M→ M τ x Stall (inj₁ k≡Stall) cMτx≡k = cong no-uncertain cMτx≡k
prop3M→ M τ x NoStall (inj₂ k≡NoStall) cMτx≡k = cong no-uncertain cMτx≡k

---- ############ FINAL RESULT - Theorem 3 ############
-- Theorem 3 says:
-- a classification k is τ-confident iff τ ≤ P[ k | X = x ]
theorem3← : ∀ (M τ x)
          → τ-confident M τ x ≡ true
          → ∃[ k ] (∃[ p ] (((P[ k |X= x ] (M→pbs M)) ≡ just p) × (τ ≤ p))) -- which means: τ ≤ P[ k | X = x ]
theorem3← M τ x τconf≡true = let -- prop3M-prior← M τ x k (prop3M← M τ x k τconf≡true)
  ⟨ k , ⟨ cMτx≡k , k≢Uncertain ⟩ ⟩ = prop3M← M τ x τconf≡true
  in ⟨ k , prop3M-prior←' M τ x k k≢Uncertain cMτx≡k ⟩

theorem3→ : ∀ (M τ x k)
          → (1/2 < τ × τ ≤ 1ℝ)
          → k ≡ Stall ∪ k ≡ NoStall
          → ∃[ p ] (((P[ k |X= x ] (M→pbs M)) ≡ just p) × (τ ≤ p)) -- which means: τ ≤ P[ k | X = x ]
          → τ-confident M τ x ≡ true
theorem3→ M τ x k 1/2<τ≤1 k≢Uncertain ⟨p,⟩ =
    prop3M→ M τ x k k≢Uncertain (prop3M-prior→ M τ x k 1/2<τ≤1 ⟨p,⟩)
---- ############ Theorem 3 END ############

-------------------------- Starting point - Theorem 4 --------------------------
-- The final theorem is more a corolary. It follows from Theorem 2 and 3
prop4M← : ∀ (M z τ x)
          → safety-envelope M z τ x ≡ true
          → (Any (λ{⟨ θ , ⟨ nd , p ⟩ ⟩ → x ∈ pi nd z}) (Model.fM M))
             × ∃[ k ] (∃[ p ] (((P[ k |X= x ] (M→pbs M)) ≡ just p) × (τ ≤ p)))
prop4M← M z τ x seM≡true = let
    -- Taking from the safety-envelope definition its components
    ⟨ left , τ-conf ⟩ =
             lem∧ {a = proj₂ (z-predictable M z x)}
                  {b = τ-confident M τ x}
                  seM≡true
    z-pred-x≡⟨x,true⟩ = cong (⟨ x ,_⟩) left
  in ⟨ theorem1← M z x z-pred-x≡⟨x,true⟩ , theorem2← M τ x τ-conf ⟩

prop4M→ : ∀ (M z τ x k)
          → (1/2 < τ × τ ≤ 1ℝ)
          → k ≡ Stall ∪ k ≡ NoStall
          → (Any (λ{⟨ θ , ⟨ nd , p ⟩ ⟩ → x ∈ pi nd z}) (Model.fM M))
             × ∃[ p ] (((P[ k |X= x ] (M→pbs M)) ≡ just p) × (τ ≤ p))
          → safety-envelope M z τ x ≡ true
prop4M→ M z τ x k 1/2<τ≤1 k≢Uncertain ⟨ Any[θ→x∈pi-nd-z]M , ⟨p,⟩ ⟩ = let
  z-pred≡⟨x,true⟩ = theorem1→ M z x Any[θ→x∈pi-nd-z]M
  τ-conf          = theorem2→ M τ x k 1/2<τ≤1 k≢Uncertain ⟨p,⟩
  in cong₂ (_∧_) (cong proj₂ z-pred≡⟨x,true⟩) τ-conf
---- ############ Theorem 4 END ############
```

12

4.2 `Avionics/SafetyEnvelopes/ExtInterface.agda`

This file makes two functions available to Haskell `zPredictable` and `sampleZPredictable`, which implement z-predictability for one measurement (presented in the paper) and z-predictability for a sample of measurements, respectively.

```
module Avionics.SafetyEnvelopes.ExtInterface where

open import Data.Bool using (Bool)
open import Data.Float using (Float)
open import Data.List using (List; map)
open import Data.Maybe using (Maybe; just; nothing)
open import Data.Product using (_×_; _,_)

open import Avionics.Probability using (Dist; NormalDist; ND)
open import Avionics.Real renaming (fromFloat to ff; toFloat to tf)
open import Avionics.SafetyEnvelopes using (z-predictable'; sample-z-predictable)

open import ExtInterface.Data.Maybe using (just; nothing) renaming (Maybe to ExtMaybe)
open import ExtInterface.Data.Product as Ext using (⟨_,_⟩)

fromFloats-z-predictable : List (Float Ext.× Float) → Float → Float → ExtMaybe (Float Ext.× Bool)
fromFloats-z-predictable means×stds z x =
    let
      ndists = map (λ{⟨ mean , std ⟩ → ND (ff mean) (ff std)}) means×stds
      (m , b) = z-predictable' ndists (ff z) (ff x)
    in
      just ⟨ tf m , b ⟩
{-# COMPILE GHC fromFloats-z-predictable as zPredictable #-}

fromFloats-sample-z-predictable :
    List (Float Ext.× Float)
    → Float → Float → List Float → ExtMaybe (Float Ext.× Float Ext.× Bool)
fromFloats-sample-z-predictable means×stds zμ zσ xs =
    let
      ndists = map (λ{⟨ mean , std ⟩ → ND (ff mean) (ff std)}) means×stds
    in
      return (sample-z-predictable ndists (ff zμ) (ff zσ) (map ff xs))
  where
    return : Maybe (ℝ × ℝ × Bool) → ExtMaybe (Float Ext.× Float Ext.× Bool)
    return nothing = nothing
    return (just (m' , v' , b)) = just ⟨ tf m' , ⟨ tf v' , b ⟩ ⟩
{-# COMPILE GHC fromFloats-sample-z-predictable as sampleZPredictable #-}
```

4.3 `Avionics/Bool.agda`

This file contains some basic tools/functions and properties for booleans and their operations. They are used to "operate" with booleans at the type level.

```
module Avionics.Bool where

open import Data.Bool using (Bool; true; false; _∧_; T)
open import Data.Unit using (T; tt)
open import Data.Product using (_×_; _,_)
open import Relation.Binary.PropositionalEquality using (_≡_; refl; inspect; [_])

--open import Avionics.Product using (_×_; ⟨_,_⟩)

--TODO: Replace with T⟺≡ from standard library
```

```agda
≡→T : ∀ {b : Bool} → b ≡ true → T b
≡→T refl = tt

T→≡ : ∀ {b : Bool} → T b → b ≡ true
T→≡ {true} tt = refl

T∧→× : ∀ {x y} → T (x ∧ y) → (T x) × (T y)
T∧→× {true} {true} tt = tt , tt
--TODO: Find a way to extract the function below from `T-∧` (standard library)
--T∧→× {x} {y} = ? -- Equivalence.to (T-∧ {x} {y})

×→T∧ : ∀ {x y} → (T x) × (T y) → T (x ∧ y)
×→T∧ {true} {true} (tt , tt) = tt


lem∧ : {a b : Bool} → a ∧ b ≡ true → a ≡ true × b ≡ true
lem∧ {true} {true} refl = refl , refl

∧≡true→×≡ : ∀ {A B : Set} {f : A → Bool} {g : B → Bool}
              (n : A) (m : B)
            → f n ∧ g m ≡ true
            → f n ≡ true × g m ≡ true
∧≡true→×≡ {f = f} {g = g} n m fn∧gm≡true = lem∧ {f n} {g m} fn∧gm≡true
```

4.4 `Avionics/List.agda`

This file contains basic properties of lists not present in the official library and might be used by any other library.

```agda
module Avionics.List where

open import Data.Bool using (Bool; true; false; T)
open import Data.List as List using (List; []; _::_; any)
open import Data.List.Relation.Unary.Any using (Any; here; there)
open import Function using (_∘_)
open import Relation.Binary.PropositionalEquality using (_≡_; inspect; [_]; refl)

open import Avionics.Bool using (≡→T)

≡→any : ∀ {a} {A : Set a} (f) (ns : List A)
          → any f ns ≡ true
          → Any (T ∘ f) ns
          --→ Any (λ x → T (f x)) ns
≡→any f [] ()
≡→any f (n :: ns) any-f-⟨n::ns⟩≡true with f n | inspect f n
... | true | [ fn≡t ] = here (≡→T fn≡t)
... | false | _ = there (≡→any f ns any-f-⟨n::ns⟩≡true)

any→≡ : ∀ {a} {A : Set a} (f) (ns : List A)
          → Any (T ∘ f) ns
          → any f ns ≡ true
any→≡ f (n :: _) (here _) with f n
... | true = refl -- or: T→≡ [*proof*from*here*]
any→≡ f (n :: ns) (there Any[T∘f]ns) with f n
... | true = refl
... | false = any→≡ f ns Any[T∘f]ns


any-map : ∀ {A B : Set} {p : B → Set} {ls : List A}
```

```
            (f : A → B)
          → Any p (List.map f ls)
          → Any (p ∘ f) ls
--any-map {ls = []} _ ()
any-map {ls = l :: ls} f (here pb) = here pb
any-map {ls = l :: ls} f (there pb) = there (any-map f pb)

any-map-rev : ∀ {A B : Set} {p : B → Set} {ls : List A}
            (f : A → B)
          → Any (p ∘ f) ls
          → Any p (List.map f ls)
any-map-rev {ls = l :: ls} f (here pb) = here pb
any-map-rev {ls = l :: ls} f (there pb) = there (any-map-rev f pb)
```

4.5 `Avionics/Probability.agda`

This file contains a skeleton of the required probability theory needed for safety envelopes to work. The symbol ? indicates the holes missing in the formalization.

```
{-# OPTIONS --allow-unsolved-metas #-}

module Avionics.Probability where

--open import Data.Fin using (Fin; fromℕ<)
open import Data.Nat using (ℕ; zero; suc)
open import Data.Product using (_×_; _,_)
open import Data.Vec using (Vec; lookup)
open import Relation.Binary.PropositionalEquality using (refl)
open import Relation.Unary using (_∈_)

open import Avionics.Real using (
    ℝ; _+_; _-_; _*_; _÷_; _^_; √_; 1/_; _^2;
    -1/2; π; e; 2ℝ; 1ℝ; 0ℝ;
    ⟨0,∞⟩; [0,∞⟩; [0,1])

record Dist (Input : Set) : Set where
  field
    pdf : Input → ℝ
    cdf : Input → ℝ
    pdf→[0,∞⟩ : ∀ x → pdf x ∈ [0,∞⟩
    cdf→[0,1] : ∀ x → cdf x ∈ [0,1]
    --intpdf≡cdf  : int pdf ≡ cdf
    --intpdf[-∞,∞]≡1ℝ  : int pdf [ -∞ , ∞ ] ≡ 1ℝ

record NormalDist : Set where
  constructor ND
  field
    μ : ℝ
    σ : ℝ
    --.0<σ  : 0ℝ <? σ

  dist : Dist ℝ
  dist = record
    {
      pdf = pdf
    ; cdf = ?
    ; pdf→[0,∞⟩ = ?
```

```agda
      ; cdf→[0,1] = ?
      }
    where
      √2π = (√ (2ℝ * π))
      1/⟨σ√2π⟩ = (1/ (σ * √2π))

      pdf : ℝ → ℝ
      pdf x = 1/⟨σ√2π⟩ * e ^ (-1/2 * (⟨x-μ⟩÷σ ^2))
        where
          ⟨x-μ⟩÷σ = ((x - μ) ÷ σ)

-- Bivariate Normal Distribution
-- Representation taken from:  https://upload.wikimedia.org/wikipedia/commons/a/a2/
                                Cumulative_function_n_dimensional_Gaussians_12.2013.pdf
record BiNormalDist : Set where
  constructor ND
  field
    μ₁ : ℝ
    μ₂ : ℝ
    σ₁ : ℝ
    σ₂ : ℝ
    ρ : ℝ

  μ : ℝ × ℝ
  μ = μ₁ , μ₂

  Σ : ℝ × ℝ × ℝ × ℝ
  Σ = σ₁ , (ρ * σ₁ * σ₂) , (ρ * σ₁ * σ₂) , σ₂

  Σ-¹ : ℝ × ℝ × ℝ × ℝ
  Σ-¹ = ( 1/1-ρ² * (1ℝ ÷ σ₁)
        , 1/1-ρ² * -ρ/σ₁σ₂
        , 1/1-ρ² * -ρ/σ₁σ₂
        , 1/1-ρ² * (1ℝ ÷ σ₂))
    where 1/1-ρ² = 1ℝ ÷ (1ℝ - ρ * ρ)
          -ρ/σ₁σ₂ = (0ℝ - ρ) ÷ (σ₁ * σ₂)

  dist : Dist ℝ
  dist = record
    {
      pdf = ?
    ; cdf = ?
    ; pdf→[0,∞⟩ = ?
    ; cdf→[0,1] = ?
    }

Mat : ℕ → Set
Mat n = Vec (Vec ℝ n) n

-- Multivariate Normal Distribution
record MultiNormal : Set where
  constructor MultiND
  field
    n : ℕ
    μ : Vec ℝ n
    Σ : Mat n

  dist : Dist (Vec ℝ n)
```

16

```
  dist = record
    {
      pdf = ?
    ; cdf = ?
    ; pdf→[0,∞⟩ = ?
    ; cdf→[0,1] = ?
    }

--num : Vec ℕ 2 → ℕ
--num vec = lookup vec (fromℕ< {0} _)

--num2 : MultiNormal → ℝ
--num2 mn = lookup μ (fromℕ< {0} _)
-- where open MultiNormal mn using (n; μ; Σ) --; dist)
-- --open Dist dist using (pdf)
```

4.6 `Avionics/Real.agda`

This file presents all the real number theory necessary to build up to safety envelopes. Each postulate is an axiom, and it's taken as a true statement but they cannot be executed. We approximate real numbers with floating-point numbers and for this, each operation in real numbers is implemented as its floating-point representation.

```
module Avionics.Real where

open import Algebra.Definitions using (LeftIdentity; RightIdentity; Commutative)
open import Data.Bool using (Bool; _∧_)
open import Data.Float using (Float)
open import Data.Maybe using (Maybe; just; nothing)
open import Data.Nat using (ℕ)
open import Level using (0ℓ; _⊔_) renaming (suc to lsuc)
open import Relation.Binary using (Decidable; _Preserves_→_)
open import Relation.Binary.Definitions using (Transitive; Trans)
open import Relation.Binary.PropositionalEquality using (_≡_; refl)
open import Relation.Nullary using (Dec; yes; no)
open import Relation.Nullary.Decidable using (False; ⌊_⌋)
open import Relation.Unary using (Pred; _∈_)

infix 4 _<_ _≤_ _<ᵇ_ _≤ᵇ_
infixl 6 _+_ _-_
infixl 7 _*_

infix 4 _≟_

postulate
  ℝ : Set
  -- TODO: `fromFloat` should return `Maybe ℝ`
  fromFloat : Float → ℝ
  toFloat : ℝ → Float
  fromℕ : ℕ → ℝ

  _+_ _*_ _^_ : ℝ → ℝ → ℝ
  -_ abs _^2 : ℝ → ℝ
  e π 0ℝ 1ℝ -1/2 1/2 2ℝ : ℝ

  -- This was inspired on how the standard library handles things.
  -- See: https://plfa.github.io/Decidable/
  _<_ _≤_ : ℝ → ℝ → Set
```

```
  _≟_ : Decidable {A = ℝ} _≡_
  _≤?_ : (m n : ℝ) → Dec (m ≤ n)
  _<?_ : (m n : ℝ) → Dec (m < n)

_<ᵇ__≤ᵇ__≡ᵇ_ : ℝ → ℝ → Bool
p <ᵇ q = ⌊ p <? q ⌋
p ≤ᵇ q = ⌊ p ≤? q ⌋
p ≡ᵇ q = ⌊ p ≟ q ⌋

_≢0 : ℝ → Set
p ≢0 = False (p ≟ 0ℝ)

Subset : Set → Set _
Subset A = Pred A 0ℓ

-- Dangerous definitions, but necessary!
postulate
  ⟨0,∞⟩ [0,∞) [0,1] : Subset ℝ

  1/_ : (p : ℝ) → ℝ
  √_ : (x : ℝ) → ℝ

_÷_ : (p q : ℝ) → ℝ
(p ÷ q) = p * (1/ q)

-- Dangerous definitions, but necessary!
postulate
  m÷n<o≡m<o*n : ∀ m n o → (m ÷ n <ᵇ o) ≡ (m <ᵇ o * n)
  m<o÷n≡m*n<o : ∀ m n o → (m <ᵇ o ÷ n) ≡ (m * n <ᵇ o)

_-_ : ℝ → ℝ → ℝ
p - q = p + (- q)

postulate
  double-neg : ∀ (x y : ℝ) → y - (y - x) ≡ x
  neg-involutive : ∀ x → -(- x) ≡ x
  neg-distrib-+ : ∀ m n → - (m + n) ≡ (- m) + (- n)
  neg-def : ∀ m → 0ℝ - m ≡ - m
  m-m≡0 : ∀ m → m - m ≡ 0ℝ
  neg-distribˡ-* : ∀ x y → - (x * y) ≡ (- x) * y
  √x^2≡absx : ∀ x → √ (x ^2) ≡ abs x

  >0→≢0 : ∀ {x : ℝ} → x ∈ ⟨0,∞⟩ → x ≢0
  >0→≥0 : ∀ {x : ℝ} → x ∈ ⟨0,∞⟩ → x ∈ [0,∞)
  >0*>0→>0 : ∀ {p q : ℝ} → p ∈ ⟨0,∞⟩ → q ∈ ⟨0,∞⟩ → (p * q) ∈ ⟨0,∞⟩
  ≢0*≢0→≢0 : ∀ {p q : ℝ} → p ≢0 → q ≢0 → (p * q) ≢0

  2>0 : 2ℝ ∈ ⟨0,∞⟩
  π>0 : π ∈ ⟨0,∞⟩

  e^x>0 : (x : ℝ) → (e ^ x) ∈ ⟨0,∞⟩
  x*x≡x^2 : ∀ x → x * x ≡ x ^2
  x^2*y^2≡⟨xy⟩^2 : ∀ x y → (x ^2) * (y ^2) ≡ (x * y)^2
  1/x^2≡⟨1/x⟩^2 : ∀ x → 1/ (x ^2) ≡ (1/ x)^2
  --√q≥0 : (q : ℝ) → (0≤q : q ∈ [0,∞)) → (√ q) {0≤q} ∈ [0,∞)
  --q>0→√q>0 : {q : ℝ} → (0<q : q ∈ ⟨0,∞⟩) → (√ q) {>0→≥0 0<q} ∈ ⟨0,∞⟩
```

$0\leq\rightarrow[0,\infty) : \{n : \mathbb{R}\} \rightarrow 0\mathbb{R} \leq n \rightarrow n \in [0,\infty)$
$0<\rightarrow\langle0,\infty\rangle : \{n : \mathbb{R}\} \rightarrow 0\mathbb{R} < n \rightarrow n \in \langle0,\infty\rangle$
$[0,\infty)\rightarrow0\leq : \{n : \mathbb{R}\} \rightarrow n \in [0,\infty) \rightarrow 0\mathbb{R} \leq n$
$\langle0,\infty\rangle\rightarrow0< : \{n : \mathbb{R}\} \rightarrow n \in \langle0,\infty\rangle \rightarrow 0\mathbb{R} < n$

$m{\leq}n{\rightarrow}m{-}p{\leq}n{-}p : \{m\ n\ p : \mathbb{R}\} \rightarrow m \leq n \rightarrow m - p \leq n - p$
$m{<}n{\rightarrow}m{-}p{<}n{-}p : \{m\ n\ p : \mathbb{R}\} \rightarrow m < n \rightarrow m - p < n - p$

```
-- trans-≤ reduces to: {i j k : ℝ} → i ≤ j → j ≤ k → i ≤ k
```
trans-$\leq$: Transitive $\_\leq\_$
$<$-trans^l : Trans $\_<\_\ \_\leq\_\ \_<\_$

```
--+-identityˡ : LeftIdentity 0ℝ _+_
--+-identityʳ : RightIdentity 0ℝ _+_
```
$+$-identity^l : $\forall x \rightarrow 0\mathbb{R} + x \equiv x$
$+$-identity^r : $\forall x \rightarrow x + 0\mathbb{R} \equiv x$
```
--+-comm : Commutative _+_
```
$+$-comm : $\forall m\ n \rightarrow m + n \equiv n + m$
```
--+-assoc : Associative _+_
```
$+$-assoc : $\forall m\ n\ o \rightarrow m + (n + o) \equiv (m + n) + o$

$*$-comm : $\forall m\ n \rightarrow m * n \equiv n * m$
$*$-assoc : $\forall m\ n\ o \rightarrow m * (n * o) \equiv (m * n) * o$

```
--0ℝ ≟_
```
$0\stackrel{?}{=}0{\equiv}yes0{\equiv}0 : (0\mathbb{R} \stackrel{?}{=} 0\mathbb{R}) \equiv$ yes refl

```
-- TODO: These properties should be written for _<_ instead of _<ᵇ_
```
abs$<$x$\rightarrow<$x$\wedge$-x$<$: $\forall \{u\ x\} \rightarrow ($abs $u <^b x) \equiv ((u <^b x) \wedge (- x <^b u))$
```
--neg-mono-<-> : -_ Preserves _<_ → (λ a b -> b < a)
```
neg-mono-$<$-$>$: $\forall m\ n \rightarrow (m <^b n) \equiv (- n <^b - m)$
```
--+-monoˡ-< : ∀ n → (_+ n) Preserves _<_ → _<_
```
$+$-mono^l-$<$: $\forall n\ o\ p \rightarrow (o <^b p) \equiv (n + o <^b n + p)$

```
-- One of the weakest points in the whole library architecture!!!
-- This is wrong, really wrong, but useful
{-# COMPILE GHC ℝ = type Double #-}
{-# COMPILE GHC fromFloat = \x -> x #-}
{-# COMPILE GHC toFloat = \x -> x #-}
{-# COMPILE GHC fromℕ = fromIntegral #-}

{-# COMPILE GHC _<ᵇ_ = (<) #-}
{-# COMPILE GHC _≤ᵇ_ = (<=) #-}
{-# COMPILE GHC _≡ᵇ_ = (==) #-}

{-# COMPILE GHC _+_ = (+) #-}
{-# COMPILE GHC _-_ = (-) #-}
{-# COMPILE GHC _*_ = (*) #-}
{-# COMPILE GHC _^_ = (**) #-}

{-# COMPILE GHC _^2 = (**2) #-}

{-# COMPILE GHC e = 2.7182818284590452353 #-}
{-# COMPILE GHC π = 3.1415926535897932384 #-}
{-# COMPILE GHC 0ℝ = 0 #-}
{-# COMPILE GHC 1ℝ = 1 #-}
```

```
{-# COMPILE GHC 2ℝ = 2 #-}
{-# COMPILE GHC -1/2 = -1/2 #-}

-- REAAALY CAREFUL WITH THIS!
-- TODO: Add some runtime checking to this. Fail hard if divisor is zero
{-# COMPILE GHC 1/_ = \x -> (1/x) #-}
{-# COMPILE GHC _÷_ = \x y -> (x/y) #-}
{-# COMPILE GHC √_ = \x -> sqrt x #-}
```

4.7 `Avionics/SafetyEnvelopes.agda`

This file implements signal energy safety envelopes and the Mahalanobis operation for one and two variable dimensions.

```
module Avionics.SafetyEnvelopes where

open import Data.Bool using (Bool; true; false; _∧_; _∨_)
open import Data.List using (List; []; _∷_; any; map; foldl; length)
open import Data.List.Relation.Unary.Any as Any using (Any)
open import Data.List.Relation.Unary.All as All using (All)
open import Data.Maybe using (Maybe; just; nothing; is-just; _»=_)
open import Data.Nat using (ℕ; zero; suc)
open import Data.Product using (_×_; proj₁; proj₂) renaming (_,_ to ⟨_,_⟩)
open import Function using (_∘_)
open import Relation.Binary.PropositionalEquality
    using (_≡_; _≢_; refl; cong; subst; sym)
open import Relation.Unary using (_∈_)
open import Relation.Nullary using (yes; no; ¬_)
open import Relation.Nullary.Decidable using (fromWitnessFalse)

open import Avionics.Real
    using (ℝ; _+_; _-_; _*_; _÷_; _^_; _<ᵇ_; _≤ᵇ_; _≤_; _<_; _<?_; _≤?_; _≟_;
           1/_;
           0ℝ; 1ℝ; 2ℝ; 1/2; _^2; √_; fromℕ)
--open import Avionics.Product using (_×_; ⟨_,_⟩; proj₁; proj₂)
open import Avionics.Probability using (Dist; NormalDist; ND; BiNormalDist)

sum : List ℝ → ℝ
sum = foldl _+_ 0ℝ

inside : NormalDist → ℝ → ℝ → Bool
inside nd z x = ((μ - z * σ) <ᵇ x) ∧ (x <ᵇ (μ + z * σ))
  where open NormalDist nd using (μ; σ)

mahalanobis1 : ℝ → ℝ → ℝ → ℝ
mahalanobis1 u v IV = √((u - v) * IV * (u - v))

inside' : NormalDist → ℝ → ℝ → Bool
inside' nd z x = mahalanobis1 μ x σ²⁻¹ <ᵇ z
  where open NormalDist nd using (μ; σ)
        σ²⁻¹ = 1/ (σ * σ)

mahalanobis2 : (ℝ × ℝ) → (ℝ × ℝ) → (ℝ × ℝ × ℝ × ℝ) → ℝ
--mahalanobis2 u v VI = ...
mahalanobis2 ⟨ u₁ , u₂ ⟩ ⟨ v₁ , v₂ ⟩
          ⟨ iv₁₁ , ⟨ iv₁₂ , ⟨ iv₂₁ , iv₂₂ ⟩ ⟩ ⟩ = √ ⟨u-v⟩IV⟨u-v⟩'
  where x₁ = u₁ - v₁
```

```
        x₂ = u₂ - v₂
        ⟨u-v⟩IV⟨u-v⟩' = (x₁ * x₁ * iv₁₁ + x₁ * x₂ * iv₁₂
                        + x₁ * x₂ * iv₂₁ + x₂ * x₂ * iv₂₂)
```

insideBiv : BiNormalDist → ℝ → (ℝ × ℝ) → Bool
insideBiv bnd z x = mahalanobis2 $μ$ x $Σ$⁻¹ <ᵇ z
  where open BiNormalDist bnd using ($μ$; $Σ$⁻¹)

FlightState = (ℝ × ℝ)

record Model : Set where
  field
    -- Flight states
    SM : List FlightState
    -- Map from flight states to Normal Distributions
    fM : List (FlightState × (NormalDist × ℝ))
    -- Every flight state must be represented in the map fM
    .fMisMap₁ : All ($λ$ $θ$ → Any ($λ$ $θ,ND$ → proj₁ $θ,ND$ ≡ $θ$) fM) SM
    .fMisMap₂ : All ($λ$ $θ,ND$ → Any ($λ$ $θ$ → proj₁ $θ,ND$ ≡ $θ$) SM) fM
    .lenSM>0 : length SM ≢ 0
    --.lenfM>0 : length fM ≢ 0 -- this is the result of the bijection above and .lenSM>0

z-predictable' : List NormalDist → ℝ → ℝ → ℝ × Bool
z-predictable' nds z x = ⟨ x , any ($λ$ nd → inside nd z x) nds ⟩

z-predictable : Model → ℝ → ℝ → ℝ × Bool
z-predictable M z x = z-predictable' (map (proj₁ ∘ proj₂) (Model.fM M)) z x

--

sample-z-predictable : List NormalDist → ℝ → ℝ → List ℝ → Maybe (ℝ × ℝ × Bool)
sample-z-predictable nds $zμ$ $zσ$ [] = nothing
sample-z-predictable nds $zμ$ $zσ$ (_ ∷ []) = nothing
sample-z-predictable nds $zμ$ $zσ$ xs@(_ ∷ _ ∷ _) = just ⟨ mean , ⟨ var_est , any inside" nds ⟩ ⟩
  where
    n = fromℕ (length xs)

    mean = (sum xs ÷ n)
    var_est = (sum (map ($λ${x →(x - mean)^2}) xs) ÷ (n - 1ℝ))

    inside" : NormalDist → Bool
    inside" nd = (($μ$ - $zμ$ * $σ$) <ᵇ mean) ∧ (mean <ᵇ ($μ$ + $zμ$ * $σ$))
        ∧ ($σ$^2 - $zσ$ * std[$σ$^2] <ᵇ var) ∧ (var <ᵇ $σ$^2 + $zσ$ * std[$σ$^2])
      where open NormalDist nd using ($μ$; $σ$)
        $σ$^2 = $σ$ ^2
        --Var[$σ$^2] = 2 * ($σ$^2)^2 / n
        std[$σ$^2] = (√2ℝ) * ($σ$^2 ÷ (√n))

        -- Notice that the estimated variance here is computed assuming `$μ$`
        -- it's the mean of the distribution. This is so that Cramer-Rao
        -- lower bound can be applied to it
        var = (sum (map ($λ${x →(x - $μ$)^2}) xs) ÷ n)

nonneg-cf : ℝ → ℝ × Bool
nonneg-cf x = ⟨ x , 0ℝ ≤ᵇ x ⟩

data StallClasses : Set where
  Stall NoStall Uncertain : StallClasses

```
P[stall]f⟨_|stall⟩_ : ℝ → List (ℝ × ℝ × Dist ℝ) → ℝ
P[stall]f⟨ x |stall⟩ pbs = sum (map unpack pbs)
  where
    unpack : ℝ × ℝ × Dist ℝ → ℝ
    unpack ⟨ P[θ] , ⟨ P[stall|θ] , dist ⟩ ⟩ = pdf x * P[θ] * P[stall|θ]
      where open Dist dist using (pdf)

f⟨_⟩_ : ℝ → List (ℝ × ℝ × Dist ℝ) → ℝ
f⟨ x ⟩ pbs = sum (map unpack pbs)
  where
    unpack : ℝ × ℝ × Dist ℝ → ℝ
    unpack ⟨ P[θ] , ⟨ _ , dist ⟩ ⟩ = pdf x * P[θ]
      where open Dist dist using (pdf)

-- There should be a proof showing that the resulting value will always be in the interval [0,1]
P[_|X=_]_ : StallClasses → ℝ → List (ℝ × ℝ × Dist ℝ) → Maybe ℝ
P[ Stall |X= x ] pbs with f⟨ x ⟩ pbs ≟ 0ℝ
... | yes _ = nothing
... | no _   = just (((P[stall]f⟨ x |stall⟩ pbs) ÷ (f⟨ x ⟩ pbs)))
P[ NoStall |X= x ] pbs with f⟨ x ⟩ pbs ≟ 0ℝ
... | yes _ = nothing
... | no _   = just (1ℝ - ((P[stall]f⟨ x |stall⟩ pbs) ÷ (f⟨ x ⟩ pbs)))
P[ Uncertain |X= _ ] _ = nothing

postulate
  -- TODO: Find out how to prove this!
  -- It probably requires to prove that P[Stall|X=x] + P[NoStall|X=x] ≡ 1
  Stall≡1-NoStall : ∀ {x pbs p}
                  → P[ Stall |X= x ] pbs ≡ just p
                  → P[ NoStall |X= x ] pbs ≡ just (1ℝ - p)
  NoStall≡1-Stall : ∀ {x pbs p}
                  → P[ NoStall |X= x ] pbs ≡ just p
                  → P[ Stall |X= x ] pbs ≡ just (1ℝ - p)

  -- Main assumptions
  -- * 0.5 < τ ≤ 1
  -- * 0 ≤ p ≤ 1
  -- Of course, these assumptions are never explicit in the code. But note
  -- that, only the first assumption can be broken by an user bad input. The
  -- second assumption stems for probability theory, yet not proven but
  -- should be true
  ≤p→¬≤1-p : ∀ {τ p}
           -- This first line are the assumptions. From them, the rest should follow
           → 1/2 < τ → τ ≤ 1ℝ -- 0.5 < τ ≤ 1
           → 0ℝ ≤ p → p ≤ 1ℝ -- 0 ≤ p ≤ 1
           → τ ≤ p
           → ¬ τ ≤ (1ℝ - p)
  ≤1-p→¬≤p : ∀ {τ p}
           → 1/2 < τ → τ ≤ 1ℝ -- 0.5 < τ ≤ 1
           → 0ℝ ≤ p → p ≤ 1ℝ -- 0 ≤ p ≤ 1
           → τ ≤ (1ℝ - p)
           → ¬ τ ≤ p


classify'' : List (ℝ × ℝ × Dist ℝ) → ℝ → ℝ → StallClasses
```

22

```
classify″ pbs τ x with P[ Stall |X= x ] pbs
...      | nothing = Uncertain
...      | just p with τ ≤? p | τ ≤? (1ℝ - p)
... | yes _ | no _ = Stall
... | no _ | yes _ = NoStall
... | _    | _     = Uncertain -- the only missing case is `no _ | no _`,
                                   `yes _ | yes _` is not possible

M→pbs : Model → List (ℝ × ℝ × Dist ℝ)
M→pbs M = map convert (Model.fM M)
  where
    n = fromℕ (length (Model.fM M))

    convert : (ℝ × ℝ) × (NormalDist × ℝ) → ℝ × ℝ × Dist ℝ
    convert ⟨ _ , ⟨ nd , P[stall|c] ⟩ ⟩ = ⟨ 1/ n , ⟨ P[stall|c] , dist ⟩ ⟩
      where open NormalDist nd using (dist)

classify : Model → ℝ → ℝ → StallClasses
classify M = classify″ (M→pbs M)

no-uncertain : StallClasses → Bool
no-uncertain Uncertain = false
no-uncertain _ = true

τ-confident : Model → ℝ → ℝ → Bool
τ-confident M τ = no-uncertain ∘ classify M τ

safety-envelope : Model → ℝ → ℝ → ℝ → Bool
safety-envelope M z τ x = proj₂ (z-predictable M z x)
                          ∧ τ-confident M τ x
```

4.8 ExtInterface/Data/Maybe.agda

This file implements the datatype Maybe which mirrors Haskell's datatype by the same name.

```
module ExtInterface.Data.Maybe where

open import Function using (_∘_)
open import Level using (Level)

private
  variable
    a b c : Level
    A : Set a
    B : Set b
    C : Set c

data Maybe (A : Set) : Set where
  nothing : Maybe A
  just : (x : A) → Maybe A

maybe : ∀ {A : Set} {B : Maybe A → Set} →
        ((x : A) → B (just x)) → B nothing → (x : Maybe A) → B x
maybe j n (just x) = j x
maybe j n nothing = n

map : (A → B) → Maybe A → Maybe B
map f = maybe (just ∘ f) nothing
```

```
-- Monad: bind

infixl 1 _»=_
_»=_ : Maybe A → (A → Maybe B) → Maybe B
nothing »= f = nothing
just a    »= f = f a

{-# COMPILE GHC Maybe = data Maybe (Nothing | Just) #-}
```

4.9 `ExtInterface/Data/Product.agda`

This file is used instead of the official "product" library in Agda because the official cannot be converted into the "tuple" datatype in Haskell. This file allows for a cleaner interface between Agda and Haskell.

```
module ExtInterface.Data.Product where

infixr 4 ⟨_,_⟩
infixr 2 _×_

data _×_ (A B : Set) : Set where
  ⟨_,_⟩ : A → B → A × B

{-# COMPILE GHC _×_ = data (,) ((,)) #-} -- Yeah, kinda abstract

proj₁ : ∀ {A B : Set} → A × B → A
proj₁ ⟨ x , y ⟩ = x

proj₂ : ∀ {A B : Set} → A × B → B
proj₂ ⟨ x , y ⟩ = y

map : ∀ {A B C D : Set}
      → (A → C) → (B → D) → A × B → C × D
map f g ⟨ x , y ⟩ = ⟨ f x , g y ⟩

map₁ : ∀ {A B C : Set}
      → (A → C) → A × B → C × B
map₁ f = map f (λ x → x)

map₂ : ∀ {A B D : Set}
      → (B → D) → A × B → A × D
map₂ g = map (λ x → x) g
```

5 Haskell Code

The Haskell code needed for running the sentinel/monitor is minimal. It loads the functionality implemented in Agda and uses it as a simple function.

5.1 `src/Main.hs`

This file showcases how the (already loaded with the data) z-predictability function can be used within Haskell. For the univariate case, single dimension, the program can take a stream of input floating-point numbers from the input command line and outputs immediately the result to the screen (is the value z-predictable).

```haskell
module Main where

import Pipes
import qualified Pipes.Prelude as P
import Data.List (tails)
import Control.Monad (forM)
import System.IO.Error (catchIOError)
import System.Environment (getArgs)

import SafetyEnvelopes (checkZPredictable, checkSampleZPredictable)

main :: IO ()
main = do
  args <- getArgs
  case args of
    ["single"] -> main_single
    ["sample"] -> main_sample
    _ -> putStrLn "You need to supply a parameter to check data. Either `single` or `sample`"


main_single :: IO ()
main_single = do
  let z = 4.0
      airspeed_i = 1
      c = checkZPredictable airspeed_i z

  runEffect $ P.stdinLn
          >-> P.map (\x-> show . c $ read x)
          >-> P.stdoutLn

main_sample :: IO ()
main_sample = do
  let z = 4.0
      mul_var = 4.0
      airspeed_i = 2
      sample_n = 10

  lines <- readLines
  let samples = takeWhile ((==sample_n) . length) $ (take sample_n) <$> tails (map read lines)
  forM samples $ \sample-> do
    print $ checkSampleZPredictable airspeed_i z mul_var sample
  return ()

readLines :: IO [String]
readLines = do
  line <- catchIOError (Just <$> getLine) (\e -> return Nothing)
  case line of
    Just l -> (l:) <$> readLines
    Nothing -> return []
```

5.2 `app/SafetyEnvelopes.hs`

This file defines functions that wrap around the Agda exported definitions. The models parameters are contained on the variables `means` and `stds`.

```haskell
module SafetyEnvelopes
    ( checkZPredictable
    , checkSampleZPredictable
```

```haskell
  ) where

import Data.Maybe (fromMaybe)

import MAlonzo.Code.Avionics.SafetyEnvelopes.ExtInterface (zPredictable, sampleZPredictable)

checkZPredictable :: Int -> Double -> Double -> (Double, Bool)
checkZPredictable n mul x = fromMaybe (-1, False) $ check_mean_cf n mul x

checkSampleZPredictable :: Int -> Double -> Double -> [Double] -> (Double, (Double, Bool))
checkSampleZPredictable n m_mean m_var = fromMaybe (-1, (-1, False)) . check_sample_cf n m_mean m_var

check_all :: Double -> Double -> Maybe (Double, Bool)
check_all mul = zPredictable (zip (concat means) (concat stds)) mul

check_mean_cf :: Int -> Double -> Double -> Maybe (Double, Bool)
check_mean_cf n mul x = do
  means_ <- means `at` n
  stds_  <- stds `at` n
  zPredictable (zip means_ stds_) mul x

check_sample_cf :: Int -> Double -> Double -> [Double] -> Maybe (Double, (Double, Bool))
check_sample_cf n mul_mean mul_var xs = do
  means_ <- means `at` n
  stds_  <- stds `at` n
  sampleZPredictable (zip means_ stds_) mul_mean mul_var xs

at :: [a] -> Int -> Maybe a
xs `at` i
  | i < 0 = Nothing
  | otherwise = helper xs i
    where helper :: [a] -> Int -> Maybe a
          helper (x:_) 0 = Just x
          helper (_:xs) i = helper xs (i-1)
          helper []   i = Nothing

--- Many lines for each, `means` and `stds`, with data for Gaussian distributions
means = [[...], ...]

stds = [[...], ...]
```

References

[1] Alpaydin, E., *Introduction to Machine Learning*, The MIT Press, Cambridge, Mass, 3rd ed., 2014.

[2] Rasmussen, C. E. and Williams, C. K. I., editors, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[3] Rogers, T., Gardner, P., Dervilis, N., Worden, K., A.E., M., Papatheou, E., and Cross, E., "Probabilistic modelling of wind turbine power curves with application of heteroscedastic Gaussian process regression," *Renewable Energy*, Vol. 148, 2020, pp. 1124–1136.

[4] Lázaro-Gredilla, M., Titsias, M. K., Verrelst, J., and Camps-Valls, G., "Estimation of vegetation chlorophyll content with variational heteroscedastic Gaussian processes," *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*, 2013, pp. 3010–3013.

[5] Goldberg, P. W., Williams, C. K., and Bishop, C. M., "Regression with input-dependent noise: a Gaussian process treatment," *Advances in Neural Information Processing Systems*, Vol. 10, 1998, pp. 1124–1136.

[6] Lázaro-Gredilla, M. and Titsias, M. K., "Variational heteroscedastic Gaussian process regression," *Proceedings of the 28th International Conference on Machine Learning*, 2011, pp. 841—848.