Safe, Real-Time Active Model Discrimination and Fault Diagnosis for Nonlinear Systems via Differentiable Reachability

Xinpei Ni[1], *Student Member, IEEE,* Melkior Ornik[2], *Senior Member, IEEE,* Glen Chou[1], *Member, IEEE,* Samuel Coogan[1], *Senior Member, IEEE,*

***Abstract*—We present a safe, real-time algorithm for active fault diagnosis and model discrimination for uncertain continuous-time nonlinear systems with process and measurement disturbances. Given a finite set of candidate models representing nominal and faulty modes, including actuator and sensor faults, we formulate an output-feedback, time-varying policy optimization problem that (i) handles state-input safety constraints over a finite horizon and (ii) drives the system to produce sampled measurements consistent with at most one model, enabling sound diagnosis. To solve this problem in real time, we develop a tractable approximation using interval over-approximations of reachable state and output sets, and encode diagnosability via a differentiable objective that penalizes overlap between the reachable output sets of possible models. The resulting optimization is solved efficiently online with gradient-based methods using JAX and differentiable reachability primitives. We evaluate our method on sensor and actuator fault diagnosis (up to 11 candidate models) in several nonlinear robotic systems, including a simulated aircraft, a hardware differential-drive robot, quadrupedal navigation, and a simulated quadrotor. Across these case studies, our approach identifies a model discriminating input sequence in under 50 ms, outperforming baselines in discrimination success rate and speed while providing formal soundness guarantees.**

I. Introduction

Faults are inevitable in long-duration robotic deployments. For instance, sensors can drift or saturate, actuators gradually lose effectiveness, and unmodeled dynamics can emerge as hardware wears. In safety-critical settings, delayed or incorrect diagnosis can be catastrophic: controllers that trust biased measurements or command degraded actuators may quickly violate safety constraints or destabilize the system. Thus, robots must not only detect that a fault has occurred, but also isolate *which* fault mode is valid quickly enough to enable reconfiguration or fail-safe behavior.

A standard formalization of fault isolation is *model discrimination* [1]: given a finite set of candidate dynamical models (e.g., a nominal model and multiple fault models), determine which model is consistent with observed behavior. This task is difficult because diagnosis must be performed from noisy, partial measurements under disturbances and uncertainty. Moreover, many distinct candidate models can produce similar measurement signals. For instance, a quadruped with a degraded leg may appear similar to a healthy quadruped with a biased IMU when commanded to walk straight. Control synthesis, however, can simplify diagnosis: for the quadruped, a carefully chosen turning maneuver may generate sensor measurements that disambiguate the failure mode. This motivates *active fault diagnosis* (AFD): rather than passively monitoring outputs, we design control inputs or policies that deliberately excite the system to make the true model identifiable.

However, designing such informative inputs online is challenging. The policy must robustly satisfy state–input safety constraints under *every* candidate fault model and uncertainty realization, while also ensuring that sampled measurements become consistent with *at most one* model so that diagnosis is deterministic. Computing exact reachable sets for general nonlinear systems requires solving the Hamilton-Jacobi partial differential equation. Conversely, statistical and learning-based approaches scale but typically lack formal guarantees on safety and sound diagnosis under worst-case uncertainty.

In this paper, we bridge this gap with a safe, real-time method for active model discrimination and fault diagnosis in uncertain nonlinear systems with process and measurement disturbances. We formulate AFD as an output-feedback, time-varying policy optimization problem that couples robust safety with a diagnosability requirement over sampled outputs. To make the problem tractable, we compute interval over-approximations of reachable *state* and *output* sets and encode discrimination via a differentiable objective that penalizes overlap between the reachable output sets of the possible models. This yields an optimization amenable to efficient gradient-based solution, enabling fast replanning as candidate models are pruned online. Our contributions are:

- An output-feedback policy optimization formulation for safe active fault diagnosis in uncertain nonlinear systems.
- Tractable, differentiable overapproximations of reachable output sets and objective for model discrimination.
- A novel algorithm for model discrimination, with higher success rate from intersection refinements.
- A fast GPU-parallelized implementation enabling online replanning and model pruning.
- Experiments on high-dimensional robotic case studies, including sensor/actuator faults (up to 11 models) on a simulated aircraft, a hardware differential-drive robot, quadrupedal navigation, and a quadrotor in under 50 ms.

[1]X. Ni, G. Chou and S. Coogan are with the Institute of Robotics and Intelligent Machines (IRIM), Georgia Institute of Technology, Atlanta, GA 30332, USA {xni32, chou, sam.coogan}@gatech.edu

[2]Melkior Ornik is with the Department of Aerospace Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA mornik@illinois.edu

Code available at: https://github.com/gtfactslab/Ni_RAL2026

II. Related Work

A. Statistical, Data-Driven, and Discrete Methods

Fault detection and model discrimination are ubiquitous tasks across science and engineering, and have therefore been studied extensively. Early work on model discrimination emerged in the chemistry community in the 1970s, where Bayesian methods were used to probabilistically distinguish among candidate mathematical models of chemical processes [1]. In reliability engineering, fault trees [2] are commonly used for fault detection, but they primarily adopt a discrete-event perspective; in contrast, we consider fault diagnosis for systems with continuous state spaces. Related probabilistic approaches to fault detection and diagnosis have also been explored in the control and robotics communities [3]–[7]. However, these methods are fundamentally statistical and typically do not provide formal guarantees on either fault identifiability or safety. Other approaches leverage deep learning by training classifiers to detect faults [8]–[10]. While such methods can be highly effective within the training distribution, they are largely heuristic and generally lack guarantees on correctness of diagnosis. In contrast, our method ensures safety while performing active fault detection in real time with formal correctness guarantees on detection.

B. Verification and Set-Based Methods

Set-based approaches have been proposed for both passive [11] and active [12] fault diagnosis. These methods compute (over-approximations of) reachable sets under different fault hypotheses and eliminate hypotheses that are inconsistent with the measured outputs. By reasoning over sets, they can rule out fault modes deterministically, rather than probabilistically as in Bayesian formulations (Sec. II-A).

Due to the computational challenges of reachability analysis, most existing set-based techniques are limited to linear or affine dynamics. For example, [12] uses a zonotope-based observer and a mixed-integer program to design active inputs for linear systems, and [13] establishes guaranteed T-detectability for distinguishing affine models. Moreover, [14] considers affine systems under output measurements but requires costly multi-parametric optimization, which precludes real-time use. Several works address nonlinear systems but remain passive: [15], [16] study set-based fault detection for nonlinear dynamics, while [17] provides guaranteed passive discrimination for polynomial dynamics via moment relaxations, at the cost of solving expensive optimization problems that are not amenable to real-time operation. Other nonlinear safety-oriented methods, such as barrier-function approaches [18], [19], can ensure safety under faults but do not directly enable active discrimination; in particular, they typically only passively detect faults or can only detect faults that drive the system toward constraint violation.

Only a small number of papers study set-based *active* model discrimination for *nonlinear* systems. The approach in [20] designs informative inputs for nonlinear dynamics but relies on integer programming and does not scale well with dimension. The method of [21] addresses polynomial and rational dynamics via a computationally expensive bilevel program, and is therefore restricted to small systems and to discriminating between two candidate models. A central difficulty in set-based model discrimination methods is computing reachable *output* sets under output-feedback control, i.e., bounding the set of measurement outputs that can arise under uncertainty, which is needed to certify fault diagnosis and safety under faulty conditions. While there has been progress on reachable-set computation for nonlinear systems under output feedback [22]–[25], these methods remain too slow for real-time deployment or are not differentiable. In contrast to the methods above, in this work, we build on recent advances in efficient reachable *state*-set computation [26] to enable real-time set-based active model discrimination and fault detection for nonlinear systems via fast and differentiable GPU-accelerated reachability analysis.

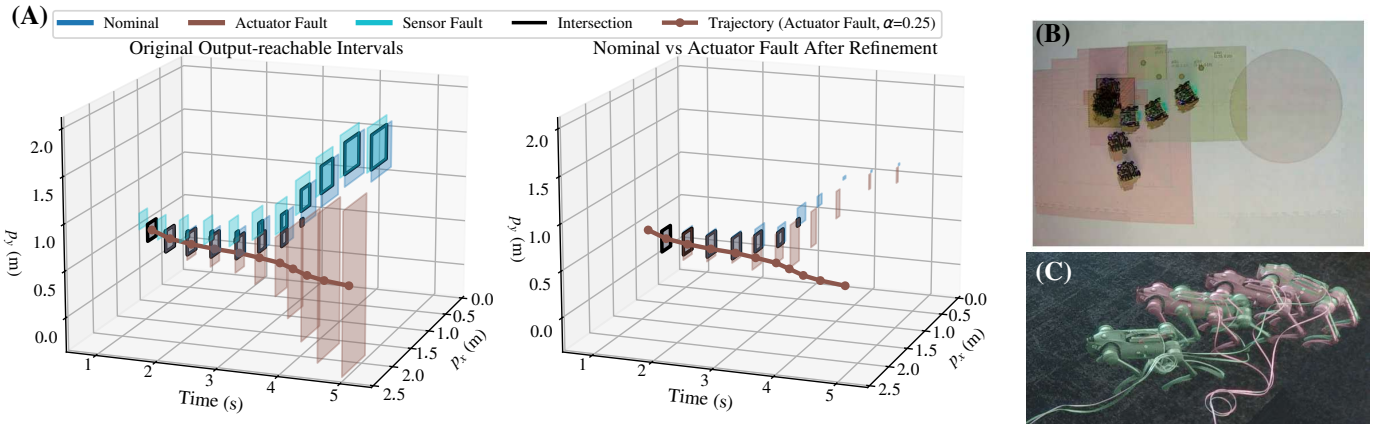


Fig. 1: **(A)** Illustration of output reachable intervals with and without sensor anticipating refinement. Using output-anticipating refinement allows the intersections of output reachable intervals to shrink faster and enables faster diagnosis. **(B)** Top view of a hardware experiment in which a separating output-feedback controller is computed by (10) and applied on a unicycle system. The controller successfully discriminates a sensor fault from an actuator fault in 5 seconds while ensuring that the system's *true position* avoids an obstacle shaded in gray. While the output reachable set for sensor fault (in green) overlaps with the obstacle, this is caused by the sensor fault's output corruption and the robot's *true state* i.e. position remains clear of the obstacle at all times. **(C)** Angled view of a hardware experiment where a separating controller is computed by (9) and applied on a quadruped to diagnose actuator and sensor faults in two separate runs. Frames corresponding to the actuator fault are tinted in red, and frames corresponding to the sensor fault are tinted green. The controller successfully diagnoses the fault in 2 seconds.

III. Problem Formulation

We consider a set of M uncertain nonlinear continuous-time dynamical systems f_m and measurement output maps h_m

$$\dot{x}_m(t) = f_m(x_m(t), u(t), w(t), \theta_a), \tag{1a}$$
$$y_m(t) = h_m(x_m(t), u(t), v(t), \theta_o), \tag{1b}$$

indexed by $m \in \{0, \dots, M-1\}$, with state $x_m \in \mathcal{X}$, control $u \in \mathcal{U}$, process noise $w \in \mathcal{W}$, measurement output $y \in \mathcal{Y}$, and measurement noise $v \in \mathcal{V}$. Here, $f_m : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \times \Theta_a \to \mathcal{X}$ and $h_m : \mathcal{X} \times \mathcal{U} \times \mathcal{V} \times \Theta_o \to \mathcal{Y}$ are differentiable functions. We denote actuator parameters as $\theta_a \in \Theta_a$ and output parameters as $\theta_o \in \Theta_o$. We let (f_0, h_0), i.e., $m = 0$, denote the *nominal* model which defines the state evolution and measurements of the healthy system, with parameters $\theta_a^* \in \Theta_a$ and $\theta_o^* \in \Theta_o$. The remaining models (f_m, h_m), for $m \in \{1, \dots, M-1\}$, denote various fault modes. We denote the set of systems as $\mathcal{M} := \{(f_m, h_m)\}_{m=0}^{M-1}$ and consider the following fault modes:

- *Actuator fault*: where there is a deviation in drift $f_m(x, u, w, \theta_a)$, i.e., $f_m \neq f_0$ and $h_m = h_0$,
- *Sensor fault*: where there is a deviation in the output function $h_m(x, u, v, \theta_o)$, i.e., $f_m = f_0$ and $h_m \neq h_0$,
- *Simultaneous fault*: where there are deviations in both the dynamics and output, i.e., $f_m \neq f_0$ and $h_m \neq h_0$.

Definition 1. *Sampled measurement set.* We assume that over a time interval $[0, T]$, measurements are collected from the system at a discrete set of N_t pre-determined time instants $\hat{\mathcal{T}} := \{t_1, \dots, t_{N_t}\} \subset [0, T]$. Assuming the system is operating under the true model $m^* \in \{0, \dots, M-1\}$, this yields the *sampled measurement set* $\hat{Y} := \{\hat{y}_i := y_{m^*}(t_i)\}_{i=1}^{N_t}$. Let $\hat{\mathcal{Y}}$ denote the set of all possible $\hat{Y}$ that can be realized under different disturbances $w(\cdot)$ and $v(\cdot)$.

Definition 2. *Consistency set.* We define $\mathcal{I}(\hat{Y})$ as the set of models consistent with the sampled measurement set $\hat{Y}$ as

$$\begin{aligned}\mathcal{I}(\hat{Y}) := \{m \in \{0, \dots, M-1\} \mid \exists x(0) \in \mathcal{X}_0, \\ \exists w(\cdot) \in \mathcal{W},\ \exists v(\cdot) \in \mathcal{V}, \exists \theta_a \in \Theta_a, \exists \theta_o \in \Theta_o \\ \text{s.t. } y_m(\hat{t}_i) = \hat{y}_i, \quad \forall i \in \{1, \dots, N_t\}\}.\end{aligned} \tag{2}$$

We consider these *guaranteed* fault diagnosis problems:

Problem 1. *Passive Model Discrimination.* Given $\mathcal{M}$, $\mathcal{W}$, $\mathcal{V}$, a set of possible initial states $\mathcal{X}_0$, and a control input sequence $u(\cdot)$, we wish to identify which model $m \in \mathcal{M}$ is consistent with the system's recorded outputs, or none exists.

Remark: While it is assumed here that the model set $\mathcal{M}$ contains one and only one correct model $m^* \in \mathcal{M}$, this algorithm will also identify that none of the models in $\mathcal{M}$ is consistent by returning an empty set when the system's behavior is inconsistent with all models in $\mathcal{M}$.

This problem can be solved by Alg. 1. A *separating controller* $u(\cdot)$ for solving Prob. 1 is one that can reduce $\mathcal{M}$ to a singleton, and computing such a *separating controller* requires solving the active fault diagnosis problem:

Problem 2. *Active Fault Diagnosis.* Given $\mathcal{M}$, $\mathcal{W}$, $\mathcal{V}$, and a set of possible initial states $\mathcal{X}_0$, we wish to design a piecewise continuous, time-varying control policy $\pi : [0, T] \times \mathcal{Y} \to \mathcal{U}$ which ensures that 1) any possible resulting sampled measurement set $\hat{\mathcal{Y}}$ is consistent with at most one model in $\mathcal{M}$ and 2) a prescribed set of state-control constraints $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{U}$ encoding, e.g., collision avoidance and actuator limits, is robustly satisfied for all $t \in [0, T]$, *i.e.*, for all $w \in \mathcal{W}$, $v \in \mathcal{V}$, and all models in $\mathcal{M}$. We formalize this problem as the following feasibility problem (3):

Algorithm 1 Fault Diagnosis

Input: Initial model set $\mathcal{M}$, observation intervals $\mathcal{Y}_{t_i}^m$, control sequence $u(\cdot) = \{u_0, u_1, \dots u_{|\hat{\mathcal{T}}|}\}$
Apply u_0 to system
for $i = 1, \dots, |\hat{\mathcal{T}}|$ **do**
  Measure output $y(t_i)$
  for $m \in \mathcal{M}$ **do**
    if $y(t_i) \notin \mathcal{Y}_{t_i}^m$ **then** ▷ $\mathcal{Y}_{t_i}^m := (5)$
      $\mathcal{M} \leftarrow \mathcal{M} \setminus \{m\}$ ▷ Prune inconsistent model
    end if
  end for
  if $|\mathcal{M}| \leq 1$ **then**
    return $\mathcal{M}$ ▷ Diagnosis succeeds
  end if
  Apply u_i to system
end for
return $\mathcal{M}$

$$\begin{aligned}
\text{find} \quad & \pi && \text{(3a)}\\
\text{s.t.} \quad & \dot{x}_m(t) = f_m(x_m(t), \pi(t, y_m(t)), w(t), \theta_a^m), \\
& \qquad \forall t \in [0, T], \quad \forall m \in \{0, \dots, M-1\}, && \text{(3b)}\\
& y_m(t) = h_m(x_m(t), \pi(t, y_m(t)), v(t), \theta_o^m), \\
& \qquad \forall t \in [0, T], \quad \forall m \in \{0, \dots, M-1\}, && \text{(3c)}\\
& x_m(0) \in \mathcal{X}_0, \quad \forall m \in \{0, \dots, M-1\}, && \text{(3d)}\\
& (x_m(t), \pi(t, y_m(t))) \in \mathcal{S}, \quad \forall t \in [0, T], \\
& \forall w(\cdot) \in \mathcal{W}, \forall v(\cdot) \in \mathcal{V}, \forall m \in \{0, \dots, M-1\}, && \text{(3e)}\\
& |\mathcal{I}(\hat{Y})| \leq 1, \quad \forall \hat{Y} \in \hat{\mathcal{Y}}. && \text{(3f)}
\end{aligned}$$

The algorithm for solving (3) must be:

- efficient enough to compute in real time,
- scalable to high-dimensional nonlinear systems, and
- sound; i.e., it identifies that the system is operating under model m only if model m is truly active.

In practice, exactly solving (3) requires exact state and reachable sets which need to be computed by solving a Hamilton-Jacobi partial differential equation [27], which becomes intractable for nonlinear systems with more than about five state dimensions [28]. In what follows, we provide preliminary background in Sec. IV and then describe our approximate solution method for (3) in Sec. V.

IV. PRELIMINARIES

A. State and Output Reachable Sets

Given (f_m, h_m) from (1) and an initial set $\mathcal{X}_0$, the *state reachable set under the controller* $\pi(\cdot)$ at time T is

$$\mathcal{X}_T^m := \{x(T) \in \mathcal{X} \mid x(0) \in \mathcal{X}_0, \dot{x}_m = f_m(x_m, u, w, \theta_a), \\ u(\tau) = \pi(\cdot), w(\tau) \in \mathcal{W}, \theta_a \in \Theta_a, \forall \tau \in [0, T)\}, \tag{4}$$

and the *output reachable set* is its image under h_m,

$$\mathcal{Y}_T^m := \{h_m(x, u, v, \theta_o) \mid x \in \mathcal{X}_T^m, u(\tau) = \pi(\cdot), \\ v \in \mathcal{V}, \theta_o \in \Theta_o\}. \tag{5}$$

B. Interval Analysis and Mixed Monotonicity

An *inclusion function* $\mathsf{F}_m = [\underline{F}_m, \overline{F}_m]$ for f_m satisfies $\underline{F}_m([x], u, [w]) \leq f_m(x, u, w) \leq \overline{F}_m([x], u, [w])$ for all $x \in [x]$, $w \in [w]$. Mixed monotonicity [29] allows us to use inclusion functions to bound an arbitrary nonlinear system's behavior. Mixed monotonicity requires only that f_m decompose into order-preserving and order-reversing parts, which any inclusion function supplies, yielding the embedding system [26], [30]

$$\dot{\underline{x}}_i = \underline{F}_{m,i}(\underline{x}, \overline{x}_{i:\underline{x}}, u, \underline{w}, \overline{w}), \quad \dot{\overline{x}}_i = \overline{F}_{m,i}(\underline{x}_{i:\overline{x}}, \overline{x}, u, \underline{w}, \overline{w}), \tag{6}$$

where $\overline{x}_{i:\underline{x}}$ replaces the ith entry of $\overline{x}$ by $\underline{x}_i$. If $\mathcal{X}_0 \subseteq [\underline{x}(0), \overline{x}(0)]$, then $\mathcal{X}_T^m \subseteq [\underline{x}(T), \overline{x}(T)]$ for all T [29, Thm. 1].

C. Output Reachable Sets and Implementation

Definition 3. *Interval Approximation of Output Reachable Set.* Given $\mathcal{R}_x(T)$ from (6), an inclusion function H_m for h_m, input u, and noise $v \in [v]$, let $\mathcal{R}_y(T, \mathcal{R}_x, u, [v]) := [\underline{H}_m(\mathcal{R}_x(T), u, [v]), \overline{H}_m(\mathcal{R}_x(T), u, [v])]$.

We compute (6) and $\mathcal{R}_y$ using `immrax`, a parallelizable and differentiable JAX based Python toolbox that facilitates interval analysis and mixed monotone reachability through composable function transforms. Its scalability on GPU, support for just-in-time compilation, and automatic differentiation enables us to compute $\partial \mathcal{R}_y / \partial u$ for (3) in a fast and scalable manner. In prior work [26], `immrax` has been used to compute state reachable sets; we extend it to compute output reachable sets for use here.

Robust safety in (3) is enforced through $\mathcal{R}_x$, and (3f) is relaxed to a cost penalty.

V. METHOD

We now describe our method, which conservatively solves (3). First, we show how to encode the model-discrimination constraint (3f) as a differentiable objective (Sec. V-A). Using a tractable over-approximation of the state and output reachable sets (Sec. IV-C) to obtain an optimization problem that can be implemented efficiently in JAX (Sec. V-B), we formulate an uninformed but functional algorithm for solving separating controllers online (Sec. V-C). We then explore a output-anticipating refinement scheme that reduces conservatism and hence discriminates models faster in Sec. V-D, and discuss its prerequisites for use in Sec. V-E.

A. Discrimination Objective via Output Set Overlap

To efficiently solve (3), one of the core difficulties is in encoding (3f). To encode this, we note that any feasible solution of (3f) guarantees that any realization of π under disturbance is a *separating controller* $u_{\text{sep}} := \{\hat{u}_i := \pi(t_i, y(t_i))\}_{i=1}^{|\hat{\mathcal{T}}|}$ for model set $\mathcal{M}$, i.e., executing u_{sep} ensures that for all $j \in \mathcal{M}$ and $k \in \mathcal{M} \setminus i$, $\mathcal{Y}_j \cap \mathcal{Y}_k = \emptyset$.

For (1), and N_t outputs taken at times $t \in \hat{\mathcal{T}} \subset \mathcal{T} := [0, T]$, denoting the set of all model pairs $q := \{j, k\}$ as $\mathcal{Q} := \{\{j, k\} \mid j, k \in \mathcal{M}, j \neq k\}$, we would like to find a control policy $\pi(t, y(t))$ such that for all pairs $q \in \mathcal{Q}$, there exists a time $t^* \in \hat{\mathcal{T}}$ such that the intersection of the fault cases' corresponding output reachable sets at t under controller $\pi(\cdot)$ is empty; i.e.,

$$\forall q = \{j, k\} \in \mathcal{Q}, \ \exists t_i \in \hat{\mathcal{T}} : \\ \mathcal{Y}_j(t_i, \pi(t, y)) \cap \mathcal{Y}_k(t_i, \pi(t, y)) = \emptyset, \tag{7}$$

while requiring that u satisfies input constraints and that the system satisfies collision avoidance constraints. Since there are cases where there does not exist a separating controller, we include (7) as a term to be minimized in the objective J:

$$J(\pi) := \sum_{q \in \mathcal{Q}} J_q(\pi) \\ J_q(\pi) := \min_{t \in \hat{\mathcal{T}}} \mathsf{Vol}\left(\bigcap_{m \in q} \mathcal{Y}_m^\epsilon(t, \pi(t, y)) \right). \tag{8}$$

where $\mathcal{Y}_m^\epsilon(t, \pi(t, y))$ represents the set $\mathcal{Y}_m(t, \pi(t, y))$ inflated by $\epsilon > 0$, and $\mathcal{Y}_m(t, \pi(t, y)) \subset \mathcal{Y}_m^\epsilon(t, \pi(t, y))$ for all $m \in \mathcal{M}$. We justify the use of (8) below.

Proposition 1. *The controller π is a separating controller for the model set $\mathcal{M}$ if π is a global minimizer of the cost function defined in* (8) *and* $J(\pi) = 0$.

Proof. Notice that if there exists a π such that $J(\pi) = 0$, $J_q(\pi) = 0$ for all $q = \{j, k\}$, and hence there exists a time t when $\mathsf{Vol}(\mathcal{Y}_j^\epsilon(t, \pi(t, y)) \cap \mathcal{Y}_k^\epsilon(t, \pi(t, y))) = 0$. This requires $\mathcal{Y}_j(t, \pi(t, y)) \cap \mathcal{Y}_k(t, \pi(t, y)) = \emptyset$, as $\mathcal{Y}_m \subset \mathcal{Y}_m^\epsilon, \forall m$. □

This proposition shows that any controller that yields zero objective value according to (8) is a separating controller. In particular, this is achieved when all pairs of possible models i, j, where $j \neq i$, have disjoint output reachable sets at some measured time instant $t \in \hat{\mathcal{T}}$. This implies that over the full horizon $[0, T]$, the resulting output sequence is contained only in the output reachable set of a single model.

Note that the volumetric overlap Vol used in (9) can be substituted by many other metrics that can measure separation between two compact sets, such as Hausdorff distance [31], that are not used in this paper. We leave evaluation of these metrics as future work.

B. Differentiable, Parallel Volumetric Overlap in `immrax`

To solve (9) efficiently, we utilize the fact that `immrax` provides differentiable inclusion functions.

While it is difficult to compute volumes of exact reachable sets and their intersections, their interval overapproximations

are much easier to work with. Any two intervals $[a] = [\underline{a}, \overline{a}]$ and $[b] = [\underline{b}, \overline{b}]$ intersect if and only if $\max(\underline{a}, \underline{b}) \leq \min(\overline{a}, \overline{b})$ componentwise. The volume of their intersection is $\mathsf{Vol}([a] \cap [b]) = \Pi_i[\min(\overline{a}, \overline{b}) - \max(\underline{a}, \underline{b})]_i$. Then, given any two *interval overapproximations* of reachable sets, we can substitute them into the previous formula to calculate their intersection in a quick, differentiable, and parallelizable manner using an efficient JAX implementation. See [26] for discussion on conservativeness of these approximations.

This allows us to compute the gradient of the reachable set bounds with respect to the control parameters $\frac{\partial \mathcal{R}_y}{\partial u}$ using JAX's automatic differentiation. We solve (9) to local optimality using a gradient-based optimizer where the output reachable tubes for all $|\mathcal{M}|$ models are computed on GPU. Our optimizer handles input constraints as hard constraints and other constraints are modeled as soft constraints and are only guaranteed to satisfy when cost is reduced to zero. We leave posing safety as hard constraints as future work.

C. Uninformed Optimization for Model Discrimination

Directly implementing the components established so far in JAX, we arrive at an optimization problem that is effective in practice. In particular, it integrates interval reachable-set propagation, the overlap-volume discrimination objective, and robust safety constraints within a single formulation.

We now present an optimization problem for active fault diagnosis. We name this formulation the *uninformed algorithm* because it does not explicitly leverage the fact that sensor measurements will be taken along the execution of the control sequence when computing a controller.

$$\min_{u} \quad J(u) := \sum_{q \in \mathcal{Q}} J_q(u) \tag{9a}$$

$$\text{s.t.} \quad J_q(u_i) = \min_{t_i \in \hat{\mathcal{T}}} \mathsf{Vol}\left[\bigcap_{m \in q} \hat{[y]}^m(t_i)\right] \tag{9b}$$

$$[x]^m(t_{i+1}) = \mathcal{R}_x^m(t_{i+1}, [x]^m(t_i), u_i, [w]) \tag{9c}$$

$$[y]^m(t_i) = \mathcal{R}_y^m(t_i, [x]^m(t_i), u_i, [v]) = \mathsf{H}_m([x]^m(t_i), u_i, [v]) \tag{9d}$$

$$\hat{[y]}^m(t_i) = [\underline{y}_m(t_i) - \epsilon, \overline{y}_m(t_i) + \epsilon] \tag{9e}$$

$$\mathcal{R}_x^m(t) \subseteq \mathcal{S}, \quad \forall t \in [0, T], \ \forall m \in \mathcal{M} \tag{9f}$$

$$u(t) \in \mathcal{U}, \quad \forall t \in \mathcal{T} \tag{9g}$$

D. Output-Anticipating Refinement

Starting from (9), we refine the state set used for forward propagation by focusing on ambiguity-relevant states. Noticing that finding a controller satisfying (7) is equivalent to separating each pair at *any* time when an output is measured, we argue that when computing a control sequence, only states that will lead to unseparated outputs require further separation. Specifically, at time t_i, we keep only states whose outputs lie in pairwise intersections of reachable output sets. Assuming a closed-form expression of preimage of output map $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$ exists, we map this set back to a state interval, which is then used to initialize subsequent reachable-set propagation. Fig. 2 provides a visual explanation of this process. To further reduce

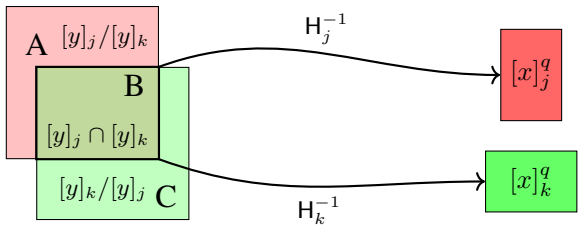


Fig. 2: Illustration of output-anticipating refinement process. Output trajectories that land in only one of the two models' output reachable intervals such as Regions A and C will be instantly discriminated, and only those landing in the intersection (Region B) require further separation. Region B can then be transformed into refined state intervals $[x]_j^q = \mathsf{H}_j^{-1}([y]_q) \cap [x]_j$ and $[x]_k^q = \mathsf{H}_k^{-1}([y]_q) \cap [x]_k$, each model j having a possibly different $[x]_j^q$ for each pair q containing it. The output-anticipating refinement scheme reduces conservatism by conditioning future computation on this fact.

conservatism, we maintain a separate interval for each model pair $q \in \mathcal{Q}$. This pairwise construction directly targets the separation condition in (3f).

The resulting multistep optimization problem is: for any time step $i \in \{0, 1, ..., K\}$, where K denotes the horizon or the number of control steps we wish to solve for, the optimal control u_i is found as:

$$\min_{u_i \in \mathcal{U}} \quad J(u_i) = \sum_{q \in \mathcal{Q}(t_i)} J_q(u_i) \tag{10a}$$

$$\text{s.t.} \quad J_q(u_i) = \min_{t_i \in \hat{\mathcal{T}}} \mathsf{Vol}\left[\bigcap_{m \in q} \hat{[y]}_q^m(t_i)\right] \tag{10b}$$

$$[x]^m(t_{i+1}) = (9c), [y]_q^m(t_i) = \mathsf{H}_m([x]_q^m(t_i), u_i, [v]) \tag{10c}$$

$$\tilde{[y]}_q(t_i) = \bigcap_{m \in q} [y]_q^m(t_i) \tag{10d}$$

$$\hat{[y]}_q^m(t_i) = [\underline{y}_q^m(t_i) - \epsilon, \overline{y}_q^m(t_i) + \epsilon] \tag{10e}$$

$$\tilde{[x]}_q^m(t_i) = \mathsf{H}_m^{-1}(\tilde{[y]}_q(t_i), u_i) \cap [x]_q^m(t_i) \tag{10f}$$

$$[x]_q^m(t_{i+1}) = \mathcal{R}_x^m(t_{i+1} | \tilde{[x]}_q^m(t_i), u_i, [w]) \tag{10g}$$

$$[x]^m(\tau) \subseteq \mathcal{S}, \quad \forall \tau \in [t_i, t_{i+1}], \ \forall m \in \mathcal{M} \tag{10h}$$

$$u(\tau) \in \mathcal{U}, \quad \forall \tau \in [t_i, t_{i+1}] \tag{10i}$$

$$\mathcal{Q}(t_{i+1}) = \{q \in \mathcal{Q}(t_i) \mid J_q(u_i^*) > 0\} \tag{10j}$$

where $[y]_q^m$ is the output reachable set for model m under pair q, and H_m^{-1} inverts model m's output map, satisfying

$$\{x \in \mathcal{X} \mid \exists v \in [v], \exists \theta_o \in \Theta_o^m : h_m(x, u, v, \theta_o) \in [y]\} \subseteq \mathsf{H}_m^{-1}([y], u). \tag{11}$$

This optimization differs from (9) only in (10f), (10d), and (10g), which describe our output-anticipating refinement. We now prove the correctness of this refinement scheme.

Proposition 2. *Let $q \in \mathcal{Q}$ with $m^* \in q$, and suppose $\hat{y}_i \in \tilde{[y]}_q(t_i)$ for all $i \leq \ell$. Then $x(t_i) \in \tilde{[x]}_q^{m^*}(t_i)$ for all $i \leq \ell$ and $x(t_i) \in [x]_q^{m^*}(t_i)$ for all $i \leq \ell + 1$.*

Proof. We show this via induction on time index i. For the base case, $i = 0$, we naturally have $x(0) \in \mathcal{X}_0 \subseteq [x]_q^{m^*}(t_0)$. For the induction step, suppose $x(t_i) \in [x]_q^{m^*}(t_i)$ for some $i \leq \ell$. We first discuss the refinement process. By hypothesis $\hat{y}_i \in \tilde{[y]}_q(t_i)$, and $\hat{y}_i = h_{m^*}(x(t_i), u_i, v(t_i), \theta_o)$ with $v(t_i) \in$

$[v]$ and $\theta_o \in \Theta_o^{m^*}$. Because $x(t_i)$ produces $\hat{y}_i$, it belongs to the left-hand set of (11), so $x(t_i) \in \mathsf{H}_{m^*}^{-1}(\tilde{[y]}_q(t_i), u_i)$. Intersecting with the inductive hypothesis and using (10f) yields $x(t_i) \in \tilde{[x]}_q^{m^*}(t_i)$. For interval propagation, we notice that for $t \in [t_i, t_{i+1}]$, $x(\cdot)$ is a trajectory of f_{m^*} from $x(t_i) \in \tilde{[x]}_q^{m^*}(t_i)$ under u_i and $w(\cdot) \in [w]$. Soundness of $\mathcal{R}_x^{m^*}$ and (10g) give $x(t_{i+1}) \in [x]_q^{m^*}(t_{i+1})$. □

Eqs. (9) and (10) solve for output-feedback laws by folding controller equation and output equation into each model's state and output reachable set propagation, naturally accounting for corrupted measurements in sensor fault scenarios.

E. Requirements for Using Output-Anticipating Refinement

The process of computing a preimage of H is NP-hard for general functions h [32]. To compute $\mathsf{H}^{-1}([y](t_i))$, one may employ subpaving refinement algorithms such as SIVIA [31]. For a general h, the time complexity of finding $\mathsf{H}^{-1}([y](t_i))$ in such a fashion is high, thus defeating Algorithm (10)'s purpose of finding a separating controller in real-time speed. Hence, in this paper, (10) is only used if there is a closed form expression for an interval overapproximation $\hat{\mathsf{X}}(t_i) \supseteq \mathsf{H}^{-1}([y](t_i))$. Additionally, if $\hat{\mathsf{X}}(t_i)$ guarantees that $[x](t_i) \subseteq \mathsf{H}^{-1}([y](t_i))$ at all times t_i, (9) should be used. Fortunately, many robotic sensors can be modeled by affine output maps of the form $y = h(x, u, v) = Cx + Du + v$. In this case,we have $x = h^{-1}(y, u, v) = C^{\dagger}(y - Du - v)$, where $C^{\dagger}$ is left-inverse of C, and can be efficiently overapproximated.

VI. Results

We present hardware and simulation experiments of our proposed active model discrimination scheme on four robotic platforms and two synthetic chains used for scaling analysis in Sec. VI-E. Notice that both (9) and (10) are able to handle input and state constraints, and linear output feedback.

Baselines: Very few works are capable of performing active model discrimination on nonlinear systems, and none can formulate output-feedback control policies. To the best of our knowledge, the only method that performs active model discrimination for nonlinear systems and has an open-source implementation is [20]. We use it as a baseline and provide runtime comparisons in Sec. VI-E.

A. Unicycle Model

We first validate our method on a standard Unicycle system [33] governed by the following dynamics:

$$\dot{p}_x = v\cos\phi, \dot{p}_y = v\sin\phi, \dot{v} = u_1 + w_1, \dot{\phi} = u_2 + w_2,$$

where (p_x, p_y) is the robot's position, v is its velocity, ϕ is its heading angle, u_1 and u_2 are control inputs, and w_1, w_2 are disturbances. For this system, only position (p_x, p_y) can be directly observed. We consider two fault scenarios:

- **Actuator Fault**: partial to full loss of turning rate control, modeled as $\dot{\phi} = \theta_a u_2 + w_2$ where $\theta_a \in [0, 1)$ represents the degradation factor.
- **Sensor Fault**: a constant bias is applied to position readings, modeled as $y = [p_x, p_y]^T + \theta_o$ where θ_o represents the sensor bias.

Measurements contain random noise with magnitude less than 0.05m, i.e., $||v||_\infty \leq 0.05$. We compute an open-loop separating controller $\pi(t)$ and a separating output-feedback controller of the form $\pi^*(t, y) = K(t)(y - y_{\text{ref}}) + u_{ff}(t)$, where y_{ref} is a pre-planned straight-line reference trajectory from start to the obstacle, according to (10) and validate it through hardware experiments, whose results are highlighted in Fig. 1(A). Solving for a controller takes $47.2\text{ms} \pm 766\mu\text{s}$.

In the hardware experiments, the robot's position consistently falls within the refined output reachable sets of the true fault model by the end of the trial, successfully distinguishing between the two fault scenarios while avoiding the obstacle, across a total of 14 trials.

We also used both (9) and (10) to solve for a separating controller under 84 combinations of initial state location, initial state interval half-widths ranging from 0.02m to 0.15m, and actuator model uncertainty. (10) is able to solve 60 scenarios, while (9) solves 42. This is because the 0.2m sensor bias renders it impossible to fully separate nominal and sensor fault for initial state uncertainties greater than 0.1m, thus preventing (9) from succeeding when (10) provides better separation using its refinement scheme. The baseline algorithm's MILP solver consistently failed on all configurations. Additionally, we tested synthesized separating controllers in a faithful unicycle robot simulator on 61200 configurations sampled within assumed bounds. Both methods report 100% correct diagnosis on in-bound samples, reflecting our method's soundness guarantee given assumed parameter intervals are not violated.

B. ADMIRE Fixed Wing Aircraft Model

We also validate our proposed method on a simplified version of the ADMIRE fixed wing aircraft Model [34]. We refer readers to [34] for full system dynamics. The mapping $g(x, \theta_a u)$ is nonlinear and thus modeled in the full system as linear interpolation of values contained in multiple aerodynamics datasheets. For ease of implementation, we use a simplified, actuator-linear model with 9 states and 10 inputs, valid around Mach 0.3 and altitude 2000m [35]. This yields a locally linear actuator dynamics of the form $\hat{g} = \bar{B}u$. We impose $||u||_\infty \leq 0.05$ to ensure linearization validity and claim no guarantee outside the linearization region. Hence, we use this formulation:

$$\dot{x} = f(x) + \bar{B}\text{diag}(\theta_a)u, \quad y = x \tag{12}$$

We define the fault set $\mathcal{M} = \{m_0, m_1, \ldots, m_{10}\}$, where m_0 is the nominal case and m_k for $k \in \{1, \ldots, 10\}$ represents a total loss of control authority in the k-th control surface. Specifically, the actuator effectiveness matrix is defined as $\theta_{a,k} = \text{diag}(v)$ where $v_k = 0$ and $v_j = 1$ for $j \neq k$. We assume that an output can be sampled from the system and a new input signal applied every $1s$. Running a GPU-parallelized version of the proposed output-anticipating refinement algorithm computes a 5s long sequence of open-loop control inputs

that fully separates the 11 candidate models, enabling real-time computation of separating inputs on a physical system. The observed increase in runtime compared to Unicycle is due to both the higher dimensionality of the ADMIRE system model and an increase in number of model pairs considered for discrimination.

TABLE I: Output-anticipating refinement (10) discriminates models faster than uninformed method (9).

Number of Unseparated Model Pairs	0s	1s	2s	3s	4s	5s
Uninformed Algorithm (9)	55	17	12	8	5	4
Output-Anticipating Refinement (10)	55	1	1	1	1	0

C. Quadrupedal Navigation

We extend this framework to model potential actuator and sensor failures in quadrupedal navigation. The dynamics of a Unitree Go2 quadruped controlled using its high-level SDK can be modeled as a planar rigid body with augmented state $x = [p_x, p_y, \tilde{p}_x, \tilde{p}_y, \phi]^T$ and control $u \in \mathbb{R}^3$, where p_x, p_y denote true horizontal and lateral positions in meters, v_x, v_y denote commanded body-frame velocities in m/s, ϕ denotes yaw angle in radians, and ω is yaw rate in rad/s. Additionally, the states $\tilde{p}_x, \tilde{p}_y, \tilde{v}_y$ denote odometer measurements of horizontal and lateral positions and velocity, which come with hardware noise. Its dynamics can be expressed as:

$$\begin{aligned}
\dot{p}_x &= v_x \cos\phi - v_y \sin\phi, \quad \dot{p}_y = v_x \sin\phi + v_y \cos\phi \\
\dot{\tilde{p}}_x &= v_x \cos\phi - \tilde{v}_y \sin\phi, \quad \dot{\tilde{p}}_y = v_x \sin\phi + \tilde{v}_y \cos\phi \\
\tilde{v}_y &= (1-\omega^2)v_y + \omega^2\nu, \quad \dot{\phi} = \omega, \quad y = \theta_o x \\
[v_x, v_y, \omega] &= (\text{diag}(\theta_a)u)^T \\
\theta_o^{\text{nominal}} &= [\mathbf{I}_{2\times2}\ \mathbf{0}_{2\times2}\ \mathbf{0}_{2\times1}], \theta_o^{\text{fault}} = [\mathbf{0}_{2\times2}\ \mathbf{I}_{2\times2}\ \mathbf{0}_{2\times1}] \\
\theta_a &\in [\underline{\theta_a}, \overline{\theta_a}], \nu \in [\underline{\nu}, \overline{\nu}], \theta_a^* = \mathbb{1}_3, \theta_o^* = \mathbb{1}_2
\end{aligned}$$

The system's actuator fault is represented by modifying the actuator outputs: a loss-of-efficiency fault scales the commanded input by a factor $\theta_a \in [\underline{\theta_a}, \overline{\theta_a}]$, For sensor fault, this simplified model captures our observation that the quadruped's onboard odometer drifts over time due to inaccuracies in v_y measurements, whose magnitude depends on the magnitude of yaw rate. We use $\tilde{v}_y = (1-\omega^2)v_y + \omega^2\nu$ to model this inaccuracy in measured lateral velocities, which is excited only when a turn is commanded to the robot.

To demonstrate our proposed algorithm's ability to handle constraints, we add a collision avoidance constraint to the optimization problem. Assuming there exists a static obstacle at a position $p_o = [-1, 0]$m, we use a circular constraint function $c(p) := \|p - p_o\|^2 - r_o^2$, where $p = [p_x, p_y]^\top$, and penalize intersection in the optimization. Because the collision avoidance constraint is imposed on p_x, p_y, which are not observed in sensor fault, (9) must be used in place of (10). Solving (9) with this collision avoidance constraint in place of (9f) yields a controller that avoids the obstacle whenever the penalty is driven to zero.

D. Quadrotor Actuator Fault Diagnosis

We consider a Crazyflie quadrotor following the standard 12-D quadrotor dynamical model [36] with realistic physical parameters and 5 candidate models including a nominal model and four single-channel actuator fault models (thrust, roll, pitch, and yaw moment, respectively, each with $\alpha \in [0.5, 0.9]$ in the faulted channel). At a 10 Hz control frequency, our algorithm synthesizes a 0.5 s (5-segment) separating input under which all 5 candidates' reachable output sets are successfully separated.

We also solve for a constant output-feedback control policy of the form $u(t_i) = u_{ff} + Ky(t_i)$ for the quadrotor using (10), where u_{ff} and K are fixed throughout the horizon. At a 50 Hz sampling rate, the controller successfully discriminates the fault within 0.12 s.

TABLE II: Runtime comparison between our method and baseline.

Task/Run Time	Ours	Baseline
Unicycle Open Loop	3.84ms	solver fails
Unicycle Output Feedback with Collision Avoidance	47.2ms	not supported
ADMIRE Open Loop	45.3ms	fails in 301s
Go2 Open Loop	40.2ms	fails in 22.6s
Quadrotor Open Loop	19.3ms	succeeds in 124s
Quadrotor Output Feedback	38.4ms	not supported

E. Runtime Analysis

We compare the runtimes of our implementation against the mixed integer program baseline in [20] across the four robotic systems with different sets of constraints. The runtimes reported for our method use (10) except for Go2 open loop, where the need to use unmeasured states to satisfy collision avoidance constraint requires the use of (9). Runtimes reported in Table II are from a laptop with Intel i7-13700HX, 32GB RAM, and an RTX4070 (8GB VRAM).

When setting up the baseline, we use the same dynamics and set of candidate models for each case. We do not provide a comparison for the case of output feedback, as the baseline cannot compute feedback controllers. The baseline fails to find a solution for all systems except quadrotor.

We use the two systems below to analyze how our algorithms' runtimes scale as the system's state dimensionality and number of candidate models change:

Integrator Chain: N-th order integrator chain follows $\dot{x}_n = x_{n+1}, n = 1, \ldots, N-1; \quad \dot{x}_N = \theta_a u \quad y = x + \theta_o$ where the nominal model has $\theta_a = 1$, actuator fault has $\theta_a \in [\alpha_{\text{lo}}, \alpha_{\text{hi}}]$ and sensor fault is modelled as a nonzero θ_0.

Nonlinear Chain: a 10-dimensional nonlinear system with $\dot{x}_i = a_i x_i - b_i x_i^3 + \theta_{a,i} u_i$. $a_i, b_i > 0$ are fixed per-channel.

In both systems, actuator faults enter multiplicatively on the input channel(s) through θ_a, while sensor faults are modeled separately at the output level as an affine map $y = \beta x + \xi$ with a predefined, known gain β and bias ξ. The integrator-chain experiment for dimensionality scaling was run on the aforementioned laptop, and the model-count and horizon sweeps below on a remote server with a single V100 GPU, 16GB VRAM and 80GB of RAM. Runtime grows mildly with state dimension, from 2.3ms to 4.5ms with (9) and 2.5ms to 10.2ms with (10) as N goes from 2 to 100 with a fixed 4-model candidate set and a fixed 4-step horizon, with peak VRAM reaching 9.6MB and 55.8MB respectively. Host RAM stays around 0.8GB over the whole sweep thanks to JAX's `vmap`.

We tested runtime scaling with respect to number of models on a 10-dimensional nonlinear chain system with a sweep from 3 to 21 candidate models. The results show faster growth: post-compilation run time ranges from 1.5ms to 7.6ms with (9) and 1.6ms to 36.5ms with (10), with VRAM usage reaching 27 and 415MB respectively. We additionally performed AMD on the 10-dimensional nonlinear chain system with horizons from 1 to 50 steps, whose run time ranges from 0.5ms to 28.1ms with (9) and 0.4ms to 137.7ms with (10), with peak VRAM reaching 45.2MB and 360.0MB, respectively. Overall, one evaluation of (9) costs $O(|\hat{\mathcal{T}}||\mathcal{M}|[nC_f + |\mathcal{M}|p])$ time and $O(|\mathcal{M}||\hat{\mathcal{T}}|n)$ memory, with C_f the cost of evaluating f_m, and (10) adds one factor of $|\mathcal{M}|$ to both by propagating one interval per model pair. These serial worst cases are loose in practice, as `vmap` makes $|\mathcal{M}|^2$ a count of work rather than depth. This confirms runtime and memory usage growth is dominated by number of candidate models and planning horizon rather than state dimensionality. In practice, it is not advisable to use too many planning steps, and while $\hat{\mathcal{T}}$ can be chosen at hardware limit, choosing a higher sampling rate increases both diagnosis success rate and runtime.

VII. Conclusion

This paper presented a safe, real-time framework for active model discrimination and fault diagnosis in uncertain nonlinear systems. By combining interval reachable-set over-approximations with a differentiable overlap-based objective, the proposed method computes informative control inputs that improve fault separability while explicitly enforcing safety constraints. Across simulation and hardware case studies, the approach demonstrated fast runtimes and reliable discrimination performance for both sensor and actuator fault scenarios. A key contribution is the output-anticipating refinement strategy (10), which reduces conservatism by propagating ambiguity-relevant state intervals. A natural next step is to explore this idea by allowing a different controller to be assigned to each model pair, rather than optimizing a shared control law across all pairs.

References

[1] P. Reilly, "Statistical methods in model discrimination," *The Canadian journal of chemical engineering*, vol. 48, no. 2, pp. 168–173, 1970.

[2] A. H. de Andrade Melani and G. F. M. de Souza, "Obtaining fault trees through sysml diagrams: A mbse approach for reliability analysis," in *2020 Annual reliability and maintainability symposium*. IEEE, 2020.

[3] F. Xu, Y. Wan, Y. Wang, and V. Puig, "Observer-based asymptotic active fault diagnosis: Separate and joint design of observer gain and input," *Automatica*, vol. 183, p. 112548, 2026.

[4] M. G. Patan and I. Ustoglu, "Robust augmented state extended kalman filter for actuator/sensor fault detection and isolation in quadrotor unmanned aerial vehicles," *Journal of Dynamic Systems, Measurement, and Control*, vol. 148, no. 1, p. 011001, 2026.

[5] E. Khalastchi and M. Kalech, "A sensor-based approach for fault detection and diagnosis for robotic systems," *Autonomous Robots*, vol. 42, no. 6, pp. 1231–1248, 2018.

[6] R. Golombek, S. Wrede, M. Hanheide, and M. Heckmann, "Online data-driven fault detection for robotic systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[7] S. Streif, F. Petzke, A. Mesbah, R. Findeisen, and R. D. Braatz, "Optimal experimental design for probabilistic model discrimination using polynomial chaos," *IFAC Proceedings Volumes*, vol. 47, no. 3, 2014.

[8] K. Garg, C. Dawson, K. Xu, M. Ornik, and C. Fan, "Model-free neural fault detection and isolation for safe control," *IEEE Control Systems Letters*, vol. 7, pp. 3169–3174, 2023.

[9] J.-H. Park and D. E. Chang, "Data-driven fault detection and isolation of system with only state measurements and control inputs using neural networks," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2021, pp. 108–112.

[10] B. Wang, Y. Shen, and Y. Zhang, "Active fault-tolerant control for a quadrotor helicopter against actuator faults and model uncertainties," *Aerospace Science and Technology*, vol. 99, p. 105745, 2020.

[11] P. Rosa and C. Silvestre, "Fault detection and isolation of lpv systems using set-valued observers: An application to a fixed-wing aircraft," *Control Engineering Practice*, vol. 21, no. 3, pp. 242–252, 2013.

[12] J. K. Scott, R. Findeisen, R. D. Braatz, and D. M. Raimondo, "Input design for guaranteed fault diagnosis using zonotopes," *Automatica*, vol. 50, no. 6, pp. 1580–1589, 2014.

[13] F. Harirchi and N. Ozay, "Guaranteed model-based fault detection in cyber–physical systems: A model invalidation approach," *Automatica*, vol. 93, pp. 476–488, 2018.

[14] G. R. Marseglia and D. M. Raimondo, "Active fault diagnosis: A multi-parametric approach," *Automatica*, vol. 79, pp. 223–230, 2017.

[15] B. Mu and J. Scott, "Set-based fault diagnosis for uncertain nonlinear systems," *Computers & Chemical Engineering*, vol. 180, 2024.

[16] B. Rego, J. Scott, D. Raimondo, M. Terra, and G. Raffo, "Zeta: a library for zonotope-based estimation and fault diagnosis of discrete-time systems," in *IEEE Conference on Decision and Control*, 2025.

[17] F. Harirchi, Z. Luo, and N. Ozay, "Model (in) validation and fault detection for systems with polynomial state-space models," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 1017–1023.

[18] L. Ballotta, A. Peruffo, R. Ferrari, and M. Mazo Jr, "Fault detection via output-based barrier functions," *European Journal of Control*, 2025.

[19] H. Zhang, Z. Li, and A. Clark, "Safe control for nonlinear systems under faults and attacks via control barrier functions," *IEEE Transactions on Automatic Control*, 2025.

[20] K. Singh, Y. Ding, N. Ozay, and S. Z. Yong, "Input design for nonlinear model discrimination via affine abstraction," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 175–180, 2018.

[21] N. Rudolph, S. Streif, and R. Findeisen, "Set-based experiment design for model discrimination using bilevel optimization," *IFAC-PapersOnLine*, vol. 49, no. 26, pp. 295–299, 2016.

[22] S. Dean, A. Taylor, R. Cosner, B. Recht, and A. Ames, "Guaranteeing safety of learned perception modules via measurement-robust control barrier functions," in *Conference on Robot Learning*, 2021.

[23] G. Chou and R. Tedrake, "Synthesizing stable reduced-order visuomotor policies for nonlinear systems via sums-of-squares optimization," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, 2023.

[24] G. Chou, N. Ozay, and D. Berenson, "Safe output feedback motion planning from images via learned perception modules and contraction theory," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 349–367.

[25] A. P. Leeman, S. Zhan, M. N. Zeilinger, and G. Chou, "Vision-sls: Safe perception-based control from learned visual representations via system level synthesis," *arXiv preprint arXiv:2604.24894*, 2026.

[26] A. Harapanahalli, S. Jafarpour, and S. Coogan, "immrax: A parallelizable and differentiable toolbox for interval analysis and mixed monotone reachability in jax," *IFAC-PapersOnLine*, vol. 58, no. 11, 2024.

[27] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," 2017. [Online]. Available: https://arxiv.org/abs/1709.07523

[28] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," *CoRR*, vol. abs/2011.02082, 2020. [Online]. Available: https://arxiv.org/abs/2011.02082

[29] S. Coogan, "Mixed monotonicity for reachability and safety in dynamical systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 5074–5085.

[30] M. Abate and S. Coogan, "Computing robustly forward invariant sets for mixed-monotone systems," in *Conf. on Dec. Ctrl. (CDC)*, 2020.

[31] L. Jaulin and E. Walter, "Set inversion via interval analysis for nonlinear bounded-error estimation," *Automatica*, vol. 29, no. 4, 1993.

[32] O. Didrit, L. Jaulin, M. Kieffer, and E. Walter, "Applied interval analysis," *Springer-Verlag*, 2001.

[33] S. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[34] L. Forssell and U. Nilsson, *ADMIRE the aero-data model in a research environment version 4.0, model description*, 2005.

[35] J.-B. Bouvier and M. Ornik, "Resilience of linear systems to partial loss of control authority," *Automatica*, vol. 152, p. 110985, 2023.

[36] F. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.