Reliable and Real-Time Anomaly Detection for Safety-Relevant Systems

1st Hagen Heermann
Cyber-Physical Systems Chair
University of Kaiserslautern-Landau
Kaiserslautern, Germany
heermann@informatik.uni-kl.de

2nd Johannes Koch
Cyber-Physical Systems Chair
University of Kaiserslautern-Landau
Kaiserslautern, Germany
johannes.koch@rptu.de

3rd Christoph Grimm
Cyber-Physical Systems Chair
University of Kaiserslautern-Landau
Kaiserslautern, Germany
grimm@informatik.uni-kl.de

***Abstract*—Safety-relevant embedded systems, e.g. in automotive applications, often require redundancy and monitors for anomaly or error detection. This paper presents an approach that permits to detect deviations of a deployed system from the possible behavior of a model. In order to satisfy real-time requirements, we use reachability analysis and represent results by a novel data type Affine Arithmetic Cartesian Decision Diagrams (AACDD). The benefits are demonstrated by the analysis of the comparison with a One Class Support Vector Machine approach on the example of a $\Sigma - \Delta$ modulator.**

***Index Terms*—Hybrid Systems, Runtime Monitoring, Consistency Checking, Reachability Analysis, Anomaly Detection**

I. Introduction

Electronic HW/SW systems are today networked with a digital environment and at the same time tightly interwoven with its physical environment. Such systems are called Cyber-Physical Systems (CPS). CPS introduce new challenges for development, modeling, verification. Particular challenges include functional safety in distributed systems and new applications like predictive maintenance. This often requires the monitoring of deployed systems at run-time in order to detect abnormal behavior.

A particular challenge in this context is to classify observed behavior into regular and abnormal ones considering the following objectives:

- High quality of classification; one wants to find abnormalities with a small number of wrong classification.
- Comprehensive coverage; one wants to detect also abnormalities that are not foreseen by a hazard/safety analysis.
- Real-time classification; if systems have to go into a fail-safe or fail-operational system, the classification must satisfy hard real-time requirements.

We classify approaches to detect such deviations into model-free and model-based ones.

As *model-based approaches* we consider approaches that utilise a model of an underlying error mechanism or a mathematical model of the system behavior. This permits to compare observed and expected values or behavior. Then, one decides whether the system adheres to its intended operation. Prominent examples are based on safety analysis and include runtime verification using Linear Temporal Logic (LTL) and similar approaches. Having a comprehensive and complete analysis of initiating faults for complex systems is a challenge and might result in a limited coverage of abnormalities or errors.

Model-free approaches, on the other hand, rely solely on observing the system without explicit information from e.g. safety/hazard analysis. This falls under the field of anomaly detection, where methods like stochastic approaches, distance-based approaches, and one-class classifiers are employed.

Here, we propose a model-based approach. As illustrated in Fig.1, the proposed approach initially employs a model of system behavior, specifically here but not limited to a hybrid automaton. During development, we use reachability analysis to generate a representation of all possible transitions and, thereby, all possible signal trajectories. For the classification during runtime, we introduce a data structure called Affine Arithmetic Cartesian Decision Diagrams (AACDDs), which encode the system's possible transitions in a compact way. This allows us to, at runtime, determine whether observed measurements are consistent with the reachability analysis results or not (see Sec. III). With runtime we mean here the execution of a physical instance of the system but not limited to. For the deployment and the availability of the measurements we suggest the deployment in a digital twin connected to a with sensors prepared physical twin.

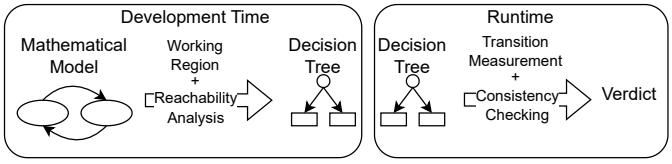

Fig. 1. Overview of the proposed model based anomaly detection process.

For evaluation, we (1) determine the achieved run-times to assess the applicability in real-time systems, and (2) compare classification results of our model-based approach with a model-free approach (Sec. IV). We focus specifically on the One-Class Support Vector Machine (OCSVM) as a benchmark to compare against the model-based approach. Through this comparison, we highlight the strengths and weaknesses of

This work was partially funded by the BMBF project KI4BoardNet No. 16ME0782

both model-based and model-free approaches in detecting deviations in Cyber-Physical Systems, ultimately contributing to more robust and reliable safety mechanisms.

II. State of the Art

In recent advancements in anomaly detection and runtime verification [1], various model-based approaches have been explored. One common method utilises zonotopes to represent continuous state spaces efficiently, facilitating reachability analysis by computing the Minkowski sum [2], [3]. Althoff et al. developed the CORA approach, leveraging zonotopes to ensure reachset conformity, encompassing all possible state measurements within the model's reachable set [4]. However, zonotopes primarily address continuous states, necessitating additional computation for discrete state verification.

ModelPlex, an alternative runtime verification approach, uses monitors grounded in differential dynamic logic to validate safety properties of trajectories [5]. This method employs logical formulas over real numbers, which complicates direct comparisons with hybrid state spaces. Damm et al. proposed the use of And-Inverter Graphs (AIGs) to describe large discrete state spaces efficiently, simplifying hybrid state space representation by treating linear predicates as single decision variables [6].

In contrast, our approach diverges by employing Affine Arithmetic Cartesian Decision Diagrams (AACDDs) to explicitly model discrete state spaces, representing the state vector with Boolean values at the AACDD leaves. Previous research has also explored Affine Arithmetic Decision Diagrams (AADDs) and Affine Binary Decision Diagrams (BDDAA) for formal verification in Analog Mixed Signal Systems [7], [8], utilising these structures for hierarchical verification processes and bounded value range verification.

While the previously presented methods and functions were based on models, there is also the ability to utilise model-free approaches. Model-free approaches have been extensively explored for anomaly detection. Yehia et al. [9] demonstrated the effectiveness of machine learning algorithms in improving shale gas production data, showing their broader applicability in data analysis contexts, which could be extended to decline curve analysis. Muhr et al. [10] proposed a probabilistic transformation of distance-based outliers, offering a novel perspective but potentially sensitive to distance metrics. Zuo et al. [11] introduced an entropy-based clustering algorithm for subspace outlier detection, which effectively handles high-dimensional data but may struggle with scalability. Schölkopf et al. [12] developed new support vector algorithms, including one-class SVMs, which are robust but computationally intensive. Rashid et al. [13] focused on high-dimensional data using nu-support vector regression, highlighting its effectiveness but complexity. Ding et al. presented extreme learning regression for nu regularization, providing fast learning but potentially less accurate with complex anomalies [14]. Model-free approaches are flexible and adaptable to various types of data without needing a predefined model structure, making them useful for discovering hidden patterns and anomalies. However, they often require large amounts of data and can be computationally intensive, leading to increased processing times and resource requirements. Additionally, model-free methods may need extensive hyperparameter tuning and can struggle with interpretability, making them challenging to implement effectively without significant expertise.

III. Consistency Checking

In this section, we explain our approach to model-based detection. For this purpose, we evaluate the consistency between the reachability of transitions in both the discrete and the continuous domain of a hybrid (mixed discrete/continuous) system. Consistency, in this context, means that a transition is considered consistent if it falls within the over-approximated set of possible transitions. To determine this, we divide the consistency term into three parts: continuous consistency, discrete consistency, and an algorithm that checks the overall consistency.

A. Continuous Consistency

We represent continuous state variables by affine forms. Affine arithmetic allows us to represent and maintain linear dependencies in a symbolic way. They are defined as $\tilde{x} = c + \sum a_i \epsilon_i$ [15], where the $\epsilon_i \in [-1, 1]$ are the so-called noise symbols that are shared allowing for the correlation. The central value $c \in \mathbb{R}$ and $a_i \in \mathbb{R}$ in combination with the noise symbols define then the range of values for x. Specifically, continuous values at a given time t are modeled as uncorrelated affine forms. The parameters of the system are also modeled as uncorrelated affine forms, maintaining the independence of each parameter. We do this to model all possible values that x_t can have within certain bounds and then determine from these possible values the consecutive values for t+1 as we see later.

When considering the system at time $t + 1$, the continuous values are represented as correlated affine forms. This correlation captures the dependencies introduced by the linear continuous dynamics, adhering to the fundamental invariant of affine arithmetic [15]. These correlated affine forms of time $t + 1$ are generated through evaluating the linear dynamics with affine arithmetic's. The goal is to check the consistency of these affine forms with the actual measurements.

To evaluate consistency, we constrain the affine forms to the specific measured values within a given uncertainty range, $\pm\Delta$, accounting for measurement noise and other uncertainties. This process transforms the problem into solving a linear inequality system.

If this inequality system has a solution, it indicates that there exist parameters within the allowed range such that the affine forms representing the continuous variables' value ranges evaluate to the measured values. In such cases, we accept the measurements as consistent with the continuous dynamics and classify them as inliers. Conversely, if no solution exists, the measurement sequence is deemed inconsistent and is classified as an outlier.

An example inequality system is shown in Eq. 1, where $\tilde{x}_t$ and $\tilde{x}_{t+1}$ are the affine forms representing the value ranges of a continuous variable, and m_t and m_{t+1} are the corresponding measurements. This example illustrates how the consistency check is formalized and applied within the context of affine arithmetic. Essentially, we're defining a time-bound flow pipe that encompasses all potential values within specific limits. We then evaluate whether measured values fall within this pipe while considering continuous dynamics.

$$\begin{aligned} m_t - \Delta \leq \tilde{x}_t \leq m_t + \Delta, \\ m_{t+1} - \Delta \leq \tilde{x}_{t+1} \leq m_{t+1} + \Delta \end{aligned} \tag{1}$$

B. Discrete Consistency

We have established how to evaluate the continuous consistency of given measurement tuples. However, since we are dealing with hybrid systems, it is also necessary to assess whether the measurements adhere to the discrete dynamics. In a hybrid system, the discrete state influences the continuous dynamics and, conversely, the continuous state can prompt changes in the discrete state. Let us denote the discrete state as d_t. The evolution of the continuous state x_t depends on the value of d_t, and the transition of d_t is governed by the state of x_t.

For simplicity, we assume that the transition of d_t is determined by linear predicates over x_t (e.g., $x_t \geq 0$). In previous work on pure symbolic simulation [7], the state x_{t+1} has been modeled using an Affine Arithmetic Decision Diagram (AADD) and d_{t+1} using an Affine Binary Decision Diagram (BDD^A). AADDs can model the value range of a continuous variable based on Boolean variables and linear predicates, whereas BDD^{A}s can represent the value of a Boolean variable based on Boolean variables and linear predicates.

Both AADDs and BDD^{A}s are types of decision trees with internal nodes consisting of linear predicates or Boolean variables. The difference lies in their leaf nodes: AADDs have affine forms, while BDD^{A}s have Boolean values. In this paper, we introduce a new type of diagram, the Affine Arithmetic Cartesian Decision Diagram (AACDD). AACDDs share the same internal structure but have leaf nodes containing tuples of affine forms and Boolean values. This structure enables us to define the complete state space at time t+1, incorporating all affine forms and discrete values, along with the linear constraints required to reach that state.

We can check the discrete consistency of a leaf of an AACDD by evaluating whether the set of linear inequalities from the root to the leaf, denoted as χ, has a solution for the given measurements m_t and m_{t+1}. This method ensures that both the continuous and discrete dynamics are accurately represented and verified.

C. Consistency Checking Algorithm

As we have now a method that checks for continuous consistency and discrete consistency, we can define our overall consistency checking algorithm as follows. For every leaf l in the AACDD, that defines the reachable state of a hybrid system from a given set of initial states, set up the following inequality system with given measurements m_t and m_{t+1}.

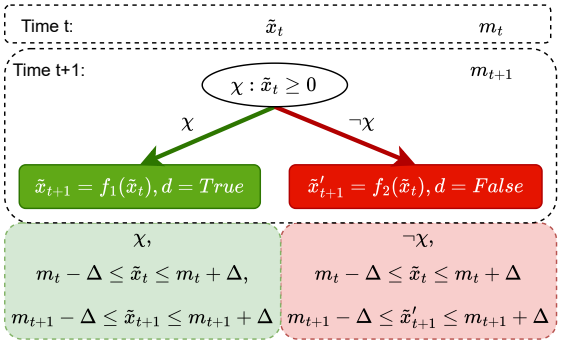

Fig. 2. Example of the structured used to determine the inequality systems to be checked. The model for time step t+1 introduces the variations, the different inequality systems.

$$\begin{aligned} &\chi, \\ m_t - \Delta \leq \tilde{x}_t \leq m_t + \Delta, \\ m_{t+1} - \Delta \leq \tilde{x}_{t+1} \leq m_{t+1} + \Delta \end{aligned} \tag{2}$$

If any of the inequality systems has a solution then accept the measurements as consistent with the model and classify it as an inlier. If no inequality system has a solution, then reject the measurements as inconsistent and as an outlier. We can see a small rudimentary example system in Fig. 2. Here we only have a single continuous variable x and two potential discrete states $d = True$ and $d = False$. Thus we have two potential follow up state spaces and two inequality systems to check. As depicted at the bottom of the figure the linear inequality systems are then dependent on the leaf tuples and the internal constraint χ is added either negated or not.

We recap as follows. The suggested approach begins with the development of a Hybrid-Automaton Model of the system, which is then transformed through symbolic simulation, following a methodology similar to that described in [7]. In the symbolic simulation results, each state variable is represented by either its own AADD or BDD^A. We leverage the AACDD to identify which leaves correspond to each other based on the system's discrete dynamics. To generate the AACDD, we first create tuples by combining leaves from the AADDs with corresponding leaves from the BDD^{A}s. For each tuple, the paths of the contained leaves are joined. If this combined path is consistent, the tuple becomes a leaf of the AACDD, and the path to this leaf in the AACDD is the joined path. Using the resulting AACDD, we can determine specific systems of inequalities. Since these inequality systems always pertain to measurement values, it is essential to maintain the AACDD during runtime to consistently generate these inequality systems as needed.

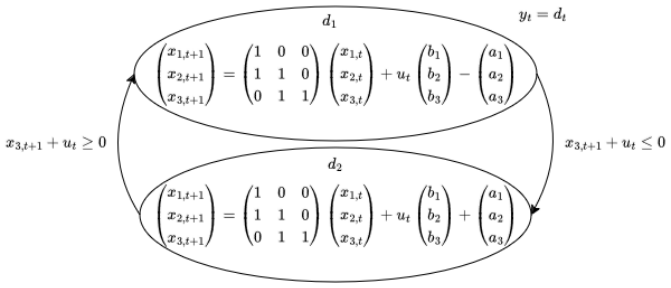

Fig. 3. The hybrid automaton model of the $\Sigma - \Delta$ modulator utilised as reference for the model-based approach as well as for the generation of the data sets.

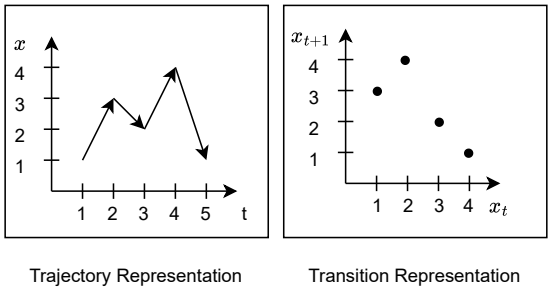

Fig. 4. As depicted the model-based algorithm takes as an input a tuple (x_t, x_{t+1}). In this figure we see a depiction how the trajectories that are generated by e.g. numerical simulation or by actual measured trajectories into a data vector representation. This is a representation of the information that the model-based algorithm takes to define an outlier or an inlier and the same information passed to the model-free approaches.

IV. EVALUATION

A. Detection Results

The system under consideration is a $\Sigma - \Delta$ modulator, a analogue to digital converter circuit. This specific $\Sigma - \Delta$ converter is of order three, which correlates to the continuous state variables of the system. The hybrid automaton, the reference model, can be seen in Fig. 3.

In this section we want to compare the presented approach that is fundamentally model-based with one that is model-free. For the comparison of the two different methodologies, model-based as proposed in this paper and model-free, four data sets were created. We will elaborate how they were created as follows.

1) **C (Correct)**: This data set was generated by a numerical simulation as a "perfect world" that strictly adheres to our model. All the parameters were drawn from the set that we initially defined as correct. The parameter ranges defined as correct can be seen in Tab. II. As seen in Fig. 3, these parameters define the actual dynamics of the model. Additionally, the continuous states were initially in the range $[-0.1, 0.1]$. As an input a constant input from the range $[-0.5, 0.5]$ was used. The simulation took 100 time steps and 1000 of these simulation runs. These then were transformed into transition tuples (x_t, x_{t+1}). This process can be seen exemplary in Fig. 4.
2) **SPE (Small Parametric Error)**: The small parametric error data set was created in the same fashion as the data set **C**. The difference is in the parametric range that was used. These ranges can be seen in Tab. III.
3) **LPE (Large Parametric Error)**: The large parametric error data set was created similar to **SPE** and **C** but with the larger parametric errors relative to **C** and **SPE**. The parameter ranges used can be seen in Tab. IV.
4) **N (Noise)**: The final data set takes the data set **C** and adds to every data value some uniformly drawn noise.

Given these four datasets, the different approaches were applied to determine if a given transition belongs to the model (an inlier) or not (an outlier). Before discussing the results, it is important to examine the datasets in more detail. In Table I, the Hausdorff distance between dataset **C** and the others is displayed. The Hausdorff distance is a metric used to measure how far apart two sets of points are. In this case, we see that the datasets are extremely close to each other, with the **LPE** dataset being an exception. This is also visually represented in Fig. 5 for the close datasets and in Fig. 6 for the **LPE** dataset.

TABLE I
IN THIS TABLE THE HAUSDORFF DISTANCE DISTANCE BETWEEN THE DIFFERENT DATA SETS IN RELATION TO THE DATA SET **C** IS GIVEN. FROM THE VALUES WE CAN SEE THAT THE DATA SETS EXCEPT FOR **LPE** ARE EXTREMELY SIMILAR BETWEEN EACH OTHER.

Data Set	Hausdorff distance
SPE	0.354
LPE	15133.139
N	0.122

These metrics suggest a significant overlap between the transitions that make up the trajectories of these different systems. Therefore, it is generally difficult to determine from a single transition whether the observed system is behaving correctly. Such a determination can only be made when an actual outlier is detected, making it challenging to identify small parametric errors. With this context, we can now examine the actual results of the two approaches.

The results of the approaches are presented in Table V. First, we examine the results of the model-based approach proposed in this paper. As expected, all transitions from dataset **C**, which represent the trajectories of the system with correct parameters, are predicted to be inliers. This is significant as it confirms that the proposed approach is an over-approximation of the correct model.

For the **SPE** dataset, only a small number of transitions are detected as outliers. This is due to the fact that most transitions are either equivalent to or very close to those in dataset **C**, as previously discussed and visualized. Consequently, only transitions that do not fit the correct dataset are identified as outliers.

For the **LPE** dataset, many transitions are detected as outliers, as expected from the Hausdorff distance values. However, the visualization shows that some transitions are still

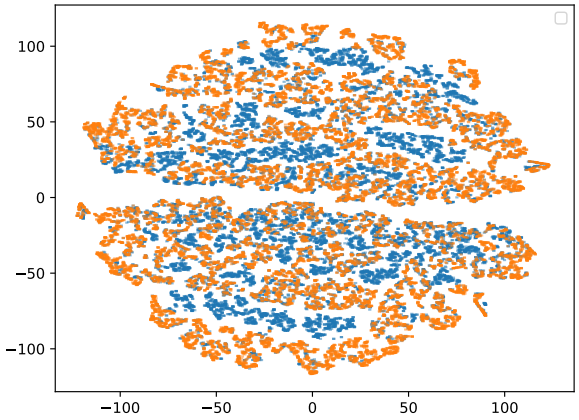

Fig. 5. t-SNE (t-distributed Stochastic Neighbor Embedding) [16], a dimensionality reduction algorithm, plot showing the dimensionality reduction from 6 to 2 for the **C** dataset (blue) and the **SPE** dataset (orange). The original 6 dimensions were from the stacked continuous transitions of time t and t+1. A significant overlap between these datasets is evident, likely due to the shared trajectories in the transitions of **C**. To simplify the visualization, only 20,000 samples from each dataset are shown.

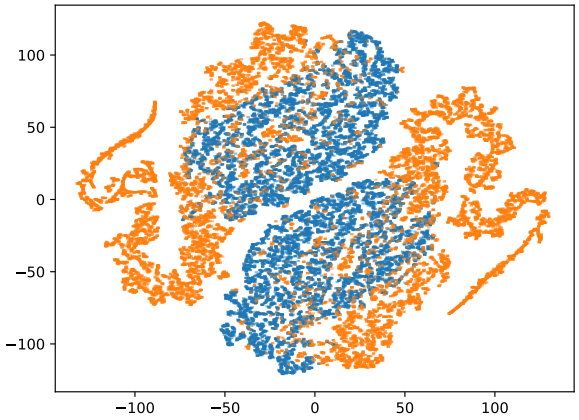

Fig. 6. In this t-SNE plot the blue points represent the correctly estimated $\Sigma - \Delta$ values, while the orange points correspond to the data with large parameter errors.

close enough to the correct possible transitions and are thus evaluated as inliers.

For the dataset **N** with added measurement noise, the issue arises that this noise needs to be accounted for and incorporated into the Δ term to avoid unnecessary and incorrect classifications of outliers due to noise. However, increasing the Δ term reduces the likelihood of detecting transitions from trajectories with small errors, presenting a trade-off between noise tolerance and detection sensitivity.

From the results of the OCSV (One-Class Support Vector Machine), the model-free approach, we can observe that the results are very close to those of the model-based approach.

TABLE II
PARAMETER RANGE OF THE SIGMA DELTA NUMERICAL SIMULATIONS. EVERY EVEN RUN WAS STARTED WITH INITIAL DISCRETE STATE FALSE. ALL OTHER TRUE. DRAWN FROM UNIFORM DISTRIBUTION

Property	Min-Value	Center	Max-Value	Δ
a_1	0.0344	0.0444	0.0544	$\pm - 0.01$
a_2	0.1881	0.2881	0.3881	± 0.1
a_3	0.6997	0.7997	0.8997	± 0.1
b_1	0.0344	0.0444	0.0544	$\pm - 0.01$
b_2	0.1881	0.2881	0.3881	± 0.1
b_3	0.6997	0.7997	0.8997	± 0.1

TABLE III
PARAMETER RANGES FROM WHERE THE PROPERTIES FOR THE ERROR SET ARE DRAWN FROM. THIS IS FOR THE SMALL PARAMETER ERROR DATA SET

Property	L-Min-Value	L-Max-Value	U-Min-Value	U-Max-Value
a_1	0.02	0.0343	0.0545	0.06
a_2	0.1	0.188	0.3882	0.4
a_3	0.6	0.6997	0.8997	0.9
b_1	0.02	0.0343	0.0545	0.06
b_2	0.1	0.188	0.3882	0.4
b_3	0.6	0.6997	0.8997	0.9

However, they exhibit some key issues inherent to model-free approaches. The most significant issue is evident in the results from dataset **C**. Not all correct transitions are classified as inliers. While adjusting specific parameters during the training process of the OCSV can reduce this issue, it cannot be entirely eliminated. Therefore, these approaches fundamentally lack the formal correctness of encompassing all possible correct transitions. Generally, the accuracy of model-free approaches improves with the size of the known correct dataset.

Another major difference between the approaches, which cannot be directly deduced from the results, is the dependency of model-free approaches on accurately determining certain parameters. In contrast, the model-based approach presented here can derive all necessary parameters from system specifications and by determining specific upper and lower bounds on expected noise.

B. Runtime Results

The simplex solver utilised to demonstrate feasibility has a potential exponential worst-case behavior. Generally, it operates in polynomial time relative to the input size of the inequality system [17]. In our worst-case scenario, we must

TABLE IV
PARAMETER RANGES FROM WHERE THE PROPERTIES FOR THE ERROR SET ARE DRAWN FROM. THIS IS FOR THE LARGE PARAMETER ERROR DATA SET

Property	L-Min-Value	L-Max-Value	U-Min-Value	U-Max-Value
a_1	0.001	0.0343	0.0545	0.5
a_2	0.01	0.188	0.3882	0.7
a_3	0.01	0.6997	0.8997	1.6
b_1	0.001	0.0343	0.0545	0.5
b_2	0.01	0.188	0.3882	0.7
b_3	0.01	0.6997	0.8997	1.6

TABLE V
INLIER/OUTLIER PREDICTION RESULTS FOR THE DIFFERENT DATA SETS. THE Δ USED FOR THE MODEL-BASED RESULTS WAS 0.001.

Method	Data Set	Number Data Vectors	Inliers	Outliers
model-based	**C**	100000	100000	0
model-based	**SPE**	100000	98552	1448
model-based	**LPE**	100000	25431	74569
model-based	**N**	100000	97615	2385
model-free	**C**	100000	90000	10000
model-free	**SPE**	100000	84831	15169
model-free	**LPE**	100000	38738	61262
model-free	**N**	100000	86867	13133

TABLE VI
RUDIMENTARY TIMING RESULTS FOR THE DIFFERENT DATA SETS.

Method	Data Set	Num. Data Vectors	Overall	Per
model-based	**C**	100000	5.561s	55.61μs
model-based	**SPE**	100000	5.607s	56.07μs
model-based	**LPE**	100000	9.375s	93.75μs
model-based	**N**	100000	5.76s	57.6μs
model-free	**C**	100000	25.578s	256.0μs
model-free	**SPE**	100000	25.589s	259.0μs
model-free	**LPE**	100000	25.353s	254.0μs
model-free	**N**	100000	25.574s	256.0μs

check one inequality system for each leaf. Consequently, the potential upper worst-case for the algorithm is exponential in the number of continuous variables and discrete states of the system being considered. However, as previously mentioned, this is not typically the case on average. Moreover, since the inequality systems are independent, we can parallelize the process to determine the results. For this specific application, the results are promising, as shown in Table VI. The observed increase in runtime for the **LPE** dataset can be attributed to the detection of numerous outliers. When an outlier is detected, every inequality system must be checked for a solution, while for inliers, the algorithm can terminate early.

V. CONCLUSION

This paper introduced a novel model-based approach for anomaly detection, utilising decision trees to represent possible transitions within a system. Our approach was evaluated against a one-class SVM on various datasets. Key findings demonstrated that the model-based approach can establish a core set of accepted transitions, providing a clear and interpretable framework that the one-class SVM lacks. This method also proved to be reliable and real-time capable, with an upper limit on the time required to determine results, ensuring consistent performance. Timing experiments further underscored the efficiency of the model-based approach, showing promising results in comparison to the model-free one-class SVM. These advantages highlight the potential of our method for applications requiring both accuracy and real-time processing capabilities. Future research should focus on extending the approach to check longer trajectories instead of just individual transitions. Additionally, a broader set of datasets and models should be explored to validate the generalizability of our findings. Examining different types of failures and conducting an in-depth analysis of the explainability of detected errors will further enhance the robustness and applicability of our model-based approach. This work lays the foundation for more reliable and interpretable anomaly detection techniques, capable of delivering real-time insights across various domains.

REFERENCES

[1] O. Maler, "Some thoughts on runtime verification," in *Runtime Verification: 16th International Conference, RV 2016, Madrid, Spain, September 23–30, 2016, Proceedings 7*, pp. 3–14, Springer, 2016. doi: 10.1007/978-3-319-46982-9_1.

[2] P. McMullen, "On zonotopes," *Transactions of the American Mathematical Society*, vol. 159, pp. 91–109, 1971. doi: 10.1090/S0002-9947-1971-0279689-2.

[3] A. Girard and C. Le Guernic, "Zonotope/hyperplane intersection for hybrid systems reachability analysis," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 215–228, Springer, 2008. doi:10.1007/978-3-540-78929-1_16.

[4] M. Althoff, "Checking and establishing reachset conformance in cora 2023," in *Proc. of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2023. doi: 10.29007/5v1g.

[5] S. Mitsch and A. Platzer, "Modelplex: Verified runtime validation of verified cyber-physical system models," *Formal Methods in System Design*, vol. 49, pp. 33–74, 2016. doi: 10.1007/s10703-016-0241-z.

[6] W. Damm, S. Disch, H. Hungar, J. Pang, F. Pigorsch, C. Scholl, U. Waldmann, and B. Wirtz, "Automatic verification of hybrid systems with large discrete state space," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 276–291, Springer, 2006. doi: 10.1007/11901914_22.

[7] C. Zivkovic, C. Grimm, M. Olbrich, O. Scharf, and E. Barke, "Hierarchical verification of ams systems with affine arithmetic decision diagrams," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1785–1798, 2018. doi: 10.1109/TCAD.2018.2864238.

[8] H. Heermann and C. Grimm, "Runtime verification of hybrid systems with affine arithmetic decision diagrams," in *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, ITG-Fb. 309: MBMV 2023, 2023. isbn: 978-3-8007-6065-7.

[9] T. Yehia, A. Wahba, S. Mostafa, and O. Mahmoud, "Suitability of different machine learning outlier detection algorithms to improve shale gas production data for effective decline curve analysis," *Energies*, vol. 15, no. 23, 2022.

[10] D. Muhr, M. Affenzeller, and J. Küng, "A probabilistic transformation of distance-based outliers," *Machine Learning and Knowledge Extraction*, vol. 5, no. 3, pp. 782–802, 2023.

[11] Z. Zuo, Z. Li, P. Cheng, and J. Zhao, "A novel subspace outlier detection method by entropy-based clustering algorithm," *Scientific Reports*, vol. 13, p. 15331, Sept. 2023.

[12] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New Support Vector Algorithms," *Neural Computation*, vol. 12, pp. 1207–1245, 05 2000.

[13] W. D. Abdullah Mohammed Rashid, Habshah Midi and J. Arasan, "Detection of outliers in high-dimensional data using nu-support vector regression," *Journal of Applied Statistics*, vol. 49, no. 10, pp. 2550–2569, 2022. PMID: 35757042.

[14] J. L. Xiao-Jian Ding, Fan Yang and J. Cao, "Extreme learning regression for nu regularization," *Applied Artificial Intelligence*, vol. 34, no. 5, pp. 378–395, 2020.

[15] J. Stolfi and L. H. de Figueiredo, "An introduction to affine arithmetic," *Trends in Computational and Applied Mathematics*, vol. 4, no. 3, pp. 297–312, 2003. doi: 10.5540/tema.2003.04.03.0297.

[16] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione, "Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets," *Nature Communications*, vol. 10, p. 5415, Nov. 2019.

[17] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, p. 385–463, may 2004.