JOURNAL OF AIR TRANSPORTATION
Vol. 31, No. 1, January–March 2023

Evaluation of Neural Network Verification Methods for Air-to-Air Collision Avoidance

Diego Manzanas Lopez* and Taylor T. Johnson†
Vanderbilt University, Nashville, Tennessee 37235
Stanley Bak‡
Stony Brook University, Stony Brook, New York 11794
Hoang-Dung Tran§
University of Nebraska at Lincoln, Lincoln, Nebraska 68588
and
Kerianne L. Hobbs¶
U.S. Air Force Research Laboratory, Wright-Patterson Air Force Base, Ohio 45433

https://doi.org/10.2514/1.D0255

Neural network approximations have become attractive to compress data for automation and autonomy algorithms for use on storage-limited and processing-limited aerospace hardware. However, unless these neural network approximations can be exhaustively verified to be safe, they cannot be certified for use on aircraft. An example of such systems is the unmanned Airborne Collision Avoidance System (ACAS) Xu, which is a very popular benchmark for open-loop neural network control system verification tools. This paper proposes a new closed-loop extension of this benchmark, which consists of a set of 10 closed-loop properties selected to evaluate the safety of an ownship aircraft in the presence of a co-altitude intruder aircraft. These closed-loop safety properties are used to evaluate five of the 45 neural networks that comprise the ACAS Xu benchmark (corresponding to co-altitude cases) as well as the switching logic between the five neural networks. The combination of nonlinear dynamics and switching between five neural networks is a challenging verification task accomplished with star-set reachability methods in two verification tools. The safety of the ownship aircraft under initial position uncertainty is guaranteed in every scenario proposed.

Nomenclature

COC	=	clear of conflict
SL	=	strong left
SR	=	strong right
v	=	velocity, ft/s
WL	=	weak left
WR	=	weak right
x	=	position of the aircraft in cartesian x direction, ft
$\dot{x}$	=	aircraft velocity in the x direction, ft/s
$\dot{y}$	=	aircraft velocity in the y direction, ft/s
y	=	position of the aircraft in the Cartesian y direction, ft
θ	=	angle to intruder with repect to ownship heading, rad
ρ	=	distance between ownship and intruder, ft
τ	=	time until loss of vertical separation, s
φ	=	third angle planar relative in aircraft geometry, rad
ψ	=	heading of intruder with regard to ownship heading, rad
$\dot{\psi}$	=	time rate of change in heading, rad/s

Subscripts

int	=	intruder
own	=	ownship

I. Introduction

THE use of machine learning in information technology domains has drastically improved automated capabilities like natural language processing [1], image classification [2], and object detection [3]. In the last several years, machine learning has also tackled complex game playing with high-dimensional state spaces, beating world experts in Go [4,5] and StarCraft II [6]. In addition to potentially optimizing the performance of complex systems, neural networks are often much smaller and faster to compute than other optimization algorithms, making them attractive for use on cyber-physical systems (CPSs) like robots, cars, drones, and satellites. For example, neural networks have been used to compress the 2 GB of lookup tables used by the Airborne Collision Avoidance System (ACAS) X to 5 MB of neural network weights [7]. However, verification of machine learning in safety-critical system domains has historically been limited to frameworks that treat neural networks like black boxes. Neural networks (NNs) are not black boxes; they compute deterministic mathematical functions that are increasingly amenable to formal reasoning methods [8,9]. Although many of these methods focus on analyzing the networks in isolation, several recent methods attempted to verify neural network control systems (NNCSs) [10–22].

Despite the recent progress in NNCS verification, developing scalable and nonconservative methods remains a key issue. Due to this growing trend and remaining challenges, several friendly competitions have been recently organized to compare the performance of these verification tools, including those mentioned in Ref. [23] for NN verification and Ref. [24] for NNCS verification. In addition to these competitions, studies by Ivanov et al. [25], Tran et al. [18], and Manzanas Lopez et al. [20] have investigated the limits and capabilities of existing verification methods and tools [10,19], as well as the tradeoffs between scalability and conservativeness. Similar to the

Presented at Paper 2021-0995 at the AIAA SciTech 2021 Forum, Virtual Event, January 11–15 and 19–21, 2021; received 12 March 2021; revision received 22 May 2022; accepted for publication 24 July 2022; published online Open Access 25 October 2022.

*Graduate Student, Electrical Engineering and Computer Science, 1025 16th Avenue South, Suite 102.
†Assistant Professor, Electrical Engineering and Computer Science, 1025 16th Avenue South, Suite 102.
‡Assistant Professor, Computer Science, 100 Nicolls Road.
§Assistant Professor, Computer Science and Engineering.
¶Aerospace Engineer, Autonomy Capability Team (ACT3), 2210 8th Street. Member AIAA.

proposed NNCS in this work, Julian and Kochenderfer [26] and Akintunde et al. [16,17] have formally verified another collision avoidance system for aircraft. Specifically, a closed-loop variant of the simplified version of the vertical avoidance control system in aircraft, which is the vertical collision avoidance system (referred to as VerticalCAS [27]), is verified using reachability analysis and mixed integer linear programming methods, respectively. In addition to the VerticalCAS, Julian and Kochenderfer [28,29] presented the same formal approach to verify the safety of a horizontal collision avoidance system (referred to as HorizontalCAS), which is another advisory system inspired by early prototypes of the ACAS Xu that presents the same input–output space of the ACAS Xu system but with smaller size neural networks. Irfan et al. [21] formally verified the advisory system for small unmanned aircraft [30] (ACAS sXu) via symbolic interval reachability analysis with ReluVal, a NN verification framework, described in Ref. [31]. The research studied a different state space and comprised smaller neural network controllers than the ACAS Xu NNCS presented here. Similarly, Clavière et al. [22] used ReluVal to formally analyze one test case of the ACAS Xu NNCS in combination with a validated simulation approach to approximate the reachable sets of the plant. This paper aims to extend and improve the verification results of these studies to a comprehensive set of benchmarks in which two aircraft are traveling at the same altitude. These benchmarks are evaluated using star-set reachability methods via the neural network verification (NNV) [19] and nnenum (which stands for neural network enumeration) [32] NNCS verification tools, which improve the scalability and overapproximation tightness of the symbolic interval analysis approaches. To the best of our knowledge, without any additional modifications or enhancements to existing NNCS reachability tools, these are the only NNCS tools that are able to verify this benchmark due to the switching behavior of the classification-based controllers. These improvements are especially notable in seven of the 10 test cases where the intruder is approaching the ownship from either side. The symbolic interval analysis methods in Ref. [22] are only able to verify 75% of the area considered in these cases while increasing the verification time between one and two orders of magnitude with respect to other areas. On the other hand, the star-set methods used in this work are able to verify the safety of the entire region considered for each of these test cases while maintaining a similar computation time as the other experiments.

The main contributions of this paper are as follows:

1) The first contributions are a description of the ACAS Xu neural network compression closed-loop benchmark and the definition of 10 closed-loop verification properties.

2) The next contributions are improved scalability and overapproximation tightness in star-set reachability methods by developing a verification approach for neural network closed-loop systems with switching classification-based controllers using star sets.

3) The last contributions are successful verification of five of 45 ACAS Xu neural networks corresponding to cases with no vertical separation, and the switching behavior against all 10 safety properties using two reachability analysis tools: NNV and nnenum.

The rest of the paper is organized as follows. Section II describes the ACAS Xu NN compression system, and Sec. III describe the plant model used as well as the NNCS benchmark. Section IV describes the closed-loop test case generation of safety properties; whereas in Sec. V, the evaluation approaches used are defined. Finally, Sec. VI introduces the results, and Sec. VII provides the concluding remarks and future work plans.

II. ACAS X

The Airborne Collision Avoidance System X is under development to one day replace the Traffic Collision Advisory System II as a midair collision prevention system [33]. ACAS X is designed to be compatible with the Federal Aviation Administration's Next-Generation Air Transportation System, which uses new sensing and navigation technologies coupled with new procedures to better optimize air traffic [33].

There are five variants of ACAS X [33], and this paper provides a collection of 10 closed-loop properties to analyze the safety of a neural network approximation of the ACAS Xu variant:

1) ACAS Xa (active) was designed to provide protection from all tracked aircraft using onboard sensors on large manned aircraft. ACAS Xa issues alerts and vertical advisories to the pilot [34].

2) ACAS Xu (unmanned) was optimized for unmanned aircraft systems (UASs). ACAS Xu issues turn rate advisories to remote pilots [35].

3) ACAS Xo (operation) sends special alerts during operations such as parallel runway approaches [33].

4) ACAS Xp (passive) tracks aircraft for potential collisions, but it does not produce advisories [33].

5) ACAS sXu (small unmanned) was designed for small UASs (SUASs) to avoid collision with manned aircraft, UASs, and other SUASs [30].

Both ACAS Xa and ACAS Xu have a goal to avoid near-midair collisions (NMACs) [36], defined as separation of less than 100 ft vertically and 500 ft horizontally [37], as depicted in Fig. 1.

ACAS X functionality centers on the use of a set of lookup tables generated offline with dynamic programming and Markov decision processes (MDPs) [38]. The input to these lookup tables is a probabilistic state distribution of the relative position and velocity of nearby aircraft approximated from sensor data. The output of the lookup tables is a cost used by ACAS X to decide the optimal advisory to provide to the pilot: no alert, traffic advisory, or a resolution advisory for pilots to maintain or increase safe separation from other aircraft [33].

This paper focuses on analysis of the ACAS Xu variant of ACAS X, to be described briefly in Sec. II.A, and specifically a neural network compression of this ACAS Xu algorithm, to be described in Sec. II.B. The ACAS Xu neural networks receive actual states of the system, rather than probabilistic state distributions from sensor data, and output the most optimal action.

A. ACAS Xu

ACAS Xu, which is the unmanned aircraft version, assigns values to a set of output actions based on a set of input variables as described in Table 1, and depicted in Fig. 2. The first five variables describe two-dimensional considerations, the sixth variable brings the scenario into three dimensions, and the seventh variable promotes advisory selection consistency. In the initial lookup tables, the state variables

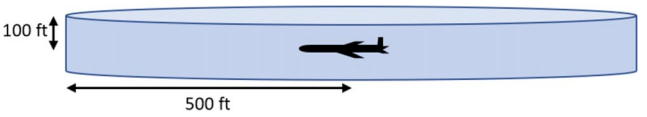


Fig. 1 Near-midair collision cylinder, defined as separation of less than 100 ft vertically and 500 ft horizontally.

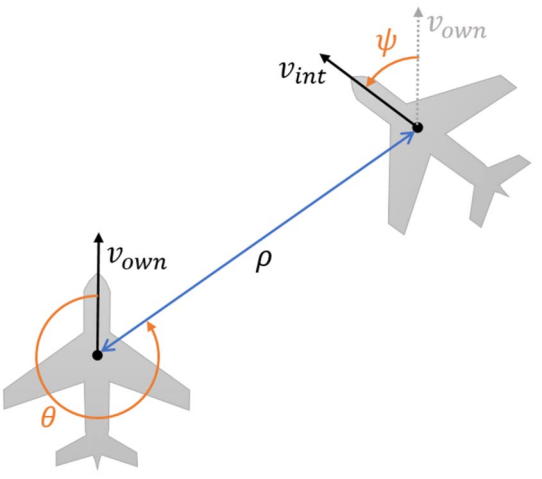


Fig. 2 Diagram of the ACAS Xu physical variables.

Table 1 Input state variables used in ACAS Xu

Variable	Units	Description	Tables	NN
ρ	ft	Distance between ownship and intruder	Yes	Yes
θ	rad	Angle to intruder with regard to ownship heading	Yes	Yes
ψ	rad	Heading of intruder with regard to ownship heading	Yes	Yes
v_{own}	ft/s	Velocity of ownship	Yes	Yes
v_{int}	ft/s	Velocity of intruder	Yes	Yes
τ	s	Time until loss of vertical separation	Yes	N0
a_{prev}	deg/s	Previous advisory	Yes	No

Table 2 ACAS Xu actions (horizontal collision avoidance)

Action	Description
SL	Strong left at 3.0 deg/s
WL	Weak left at 1.5 deg/s
COC	Clear of conflict (do nothing)
WR	Weak right turn at 1.5 deg/s
SR	Strong right turn at 3.0 deg/s

are discretized in a seven-dimensional grid with 120 million points [39] that assign a value to each of five different output action options, resulting in 600 million floating point numbers. When the state of the system falls between discrete points, the nearest neighbor point is used [39]. To deal with sensor uncertainty in the system state, ACAS Xu uses an unscented Kalman filter [40].

B. Neural Network Compression of ACAS Xu

A major challenge to the implementation of ACAS Xu is that the initial lookup tables require hundreds of gigabytes of floating point storage [7]. Using a technique called downsampling, values may be removed from the table in areas where the variation between the values is very small and has been shown to reduce ACAS Xu table size to approximately 2 GB [7]. For ACAS Xa, which limits advisories to vertical maneuvers and has smaller lookup tables to begin with, downsampling was sufficient [41]. However, ACAS Xu's 2 GB size may still be too large for the storage constraints of certified avionics hardware on UASs [28].

Developed in Ref. [7] and evaluated in Ref. [35], 45 separate neural networks were used to compress the lookup table. Each network is denoted $N_{\gamma,\beta}$, where γ corresponds to the index (1 to 5) of a specific value of previous advisory $a_{\text{prev}} \in \{COC, WL, WR, SL, SR\}$ defined in Table 2, and β corresponds to the index (1 to 9) of a specific value of time to loss of vertical separation $\tau \in \{0, 1, 5, 10, 20, 40, 60, 80, 100\}$ seconds. For example, $N_{2,3}$ corresponds to a neural network in which $a_{\text{prev}} = WL$ and $\tau = 5$. Then each of these networks receives inputs for the remaining five state variables (ρ, θ, ψ, v_{own}, and v_{int}) and outputs a value associated with each of the five output variables $\{COC, WL, WR, SL, SR\}$. Several architectures and optimizers were considered and analyzed for the training of all the 45 neural networks. AdaMax, a variant of Adam, was chosen because it proved to learn the fastest without getting stuck in local optima. As for the layer architecture, six hidden layers of 50 neurons each proved to yield the best results while maintaining an efficient computation time [7]. Hence, all the neural networks have five inputs, five outputs, and six hidden layers of 50 neurons each, as depicted in Fig. 3.

C. Open-Loop Versus Closed-Loop Verification

Previous research has verified the neural networks using a set of open-loop test points [27,28,35,39]. In other words, for a single instant in time, the neural network is verified to output appropriate scores for a particular set of inputs. The open-loop properties described in Ref. [35] became a popular benchmark for open-loop verification, and they are summarized here in the Appendix B. However, a much more important property that needs to be verified for these neural networks is that, when used in *closed-loop* control, the neural network approximation of the lookup tables will not result in control actions that violate the NMAC safety constraint depicted earlier in Fig. 1. The development and description of a set of closed-loop safety properties are some of the main contributions of this paper.

D. Set-Based Verification Compared to Monte Carlo and Design of Experiments

Set-based verification is a possible alternative to Monte Carlo simulation, which uses thousands or millions of random samples to evaluate performance across the system state space [42]. An alternative approach to Monte Carlo (especially when evaluating individual points is computationally expensive) is to apply the design of experiments to sample the state space in a structured way rather than randomly [43]. Latin Hypercube can be applied to Monte Carlo or the design of experiments approach to use a reduced sample spread evenly across the state space [44]. However, Monte Carlo and the design of experiments are not as comprehensive as set-based neural network verification, which is the focus of this paper.

III. Model

The verification effort in this paper focuses on verifying five of 45 total neural networks corresponding to cases where the loss of vertical separation τ is zero (N_{11}, N_{12}, N_{13}, N_{14}, and N_{15}). This reduces the problem to a two-dimensional co-altitude case. Nonetheless, verification of these neural networks and the switching between them present a significant challenge. Focusing on the co-altitude cases, the nonlinear plant model of the intruder and ownship aircraft are both represented using Dubins aircraft model, described in Eq. (1):

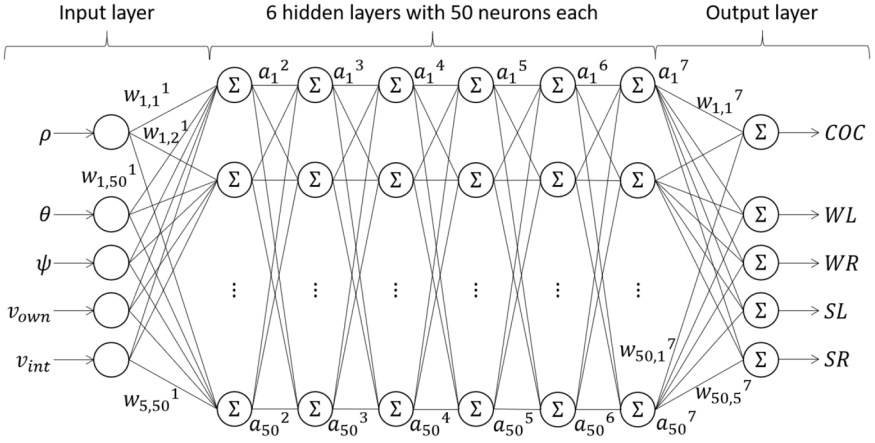


Fig. 3 Depiction of the ACAS Xu neural network.

$$\begin{aligned}\dot{x}_1 &= \dot{x} = v\cos(\psi),\\ \dot{x}_2 &= \dot{y} = v\sin(\psi),\\ \dot{x}_3 &= \dot{\psi} = u\end{aligned} \tag{1}$$

This is a simplified yet reasonable model of the aircraft dynamics at a similar level of abstraction as the inputs to the ACAS Xu NNCS, which represents the aircraft in terms of velocities, headings, and distances. However, the ACAS Xu NN model requires inputs in terms of distance ρ, angle to intruder from ownship θ, and heading of intruder with respect to ownship ψ. Conversion between the states of the Dubins model of each aircraft and the variables used as inputs to the NN is achieved via a set of complex nonlinear functions, defined in Eq. (2):

$$\begin{aligned}\rho &= \sqrt{(x_{\text{int}} - x_{\text{own}})^2 + (y_{\text{int}} - y_{\text{own}})^2}\\ \theta &= 2\tan^{-1}\left(\frac{y_{\text{int}} - y_{\text{own}}}{x_{\text{int}} - x_{\text{own}} + \rho}\right) - \psi_{\text{own}}\\ \psi &= \psi_{\text{int}} - \psi_{\text{own}}\end{aligned} \tag{2}$$

Verification tools are often not able to encode these equations. Thus, these functions are converted to ordinary differential equations (ODEs) by computing their time derivatives. Finally, these and the Dubins models are combined into a single nonlinear model with nine state variables, as described in Eq. (3):

$$\begin{aligned}\dot{x}_1 &= \dot{x}_{\text{own}} = v\cos(\psi_{\text{own}}),\\ \dot{x}_2 &= \dot{y}_{\text{own}} = v\sin(\psi_{\text{own}}),\\ \dot{x}_3 &= \dot{\psi}_{\text{own}} = u,\\ \dot{x}_4 &= \dot{x}_{\text{int}} = v\cos(\psi_{\text{int}}),\\ \dot{x}_5 &= \dot{y}_{\text{int}} = v\sin(\psi_{\text{int}}),\\ \dot{x}_6 &= \dot{\psi}_{\text{int}} = c,\\ \dot{x}_7 &= \dot{\rho} = \frac{(y_{\text{int}} - y_{\text{own}})(\dot{y}_{\text{int}} - \dot{y}_{\text{own}}) + (x_{\text{int}} - x_{\text{own}})(\dot{x}_{\text{int}} - \dot{x}_{\text{own}})}{\sqrt{(x_{\text{int}} - x_{\text{own}})^2 + (y_{\text{int}} - y_{\text{own}})^2}},\\ \dot{x}_8 &= \dot{\theta}\\ &= \frac{2(\dot{y}_{\text{int}} - \dot{y}_{\text{own}})(x_{\text{int}} - x_{\text{own}} + \rho) - 2(y_{\text{int}} - y_{\text{own}})(\dot{x}_{\text{int}} - \dot{x}_{\text{own}} + \dot{\rho})}{(y_{\text{int}} - y_{\text{own}})^2(x_{\text{own}} - x_{\text{int}} + \rho)^2}\\ &\quad - \dot{\psi}_{\text{own}},\\ \dot{x}_9 &= \dot{\psi} = \dot{\psi}_{\text{int}} - \dot{\psi}_{\text{own}}\end{aligned} \tag{3}$$

where c is a user-defined constant control input to aircraft (ft/s). ACAS Xu NN input ranges require that θ and ψ must be in the $[-\pi, \pi]$ interval. Thus, it is necessary to ensure that the angles are always defined in the $[-\pi, \pi]$ range for each control step.

Combining the ACAS Xu neural network system and the nonlinear dynamical model, the ACAS Xu NNCS benchmark is presented in a diagram in Fig. 4. There are four main components or operations highlighted:

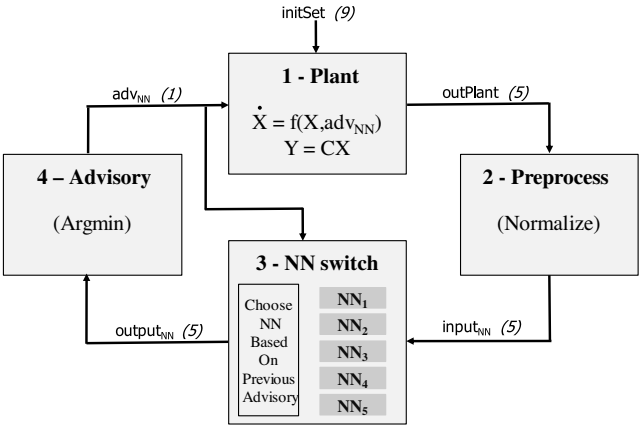


Fig. 4 Diagram of the ACAS Xu neural network closed-loop system.

1) In box 1 is plantReach(). The plant dynamics are simulated based on a specified initial state (initSet) and advisory adv_{NN} for 2 s (sampling time).

2) In box 2 is normalizeNNinputs(). Then, the outputs of the plant (outPlant) are scaled and normalized.

3) In boxes 3 and 4 are NNreach():

a) In box 3, with Reach(), the scaled outputs (input_{NN}) are processed by the corresponding neural network controller, which is selected based on the previous advisory and the loss of vertical separation, as described in Sec. II.B.

b) Box 4, with argminNN(), is based on the five outputs of the neural network output_{NN}, and the turn rate advisory adv_{NN} is computed (argmin) and commanded to the plant.

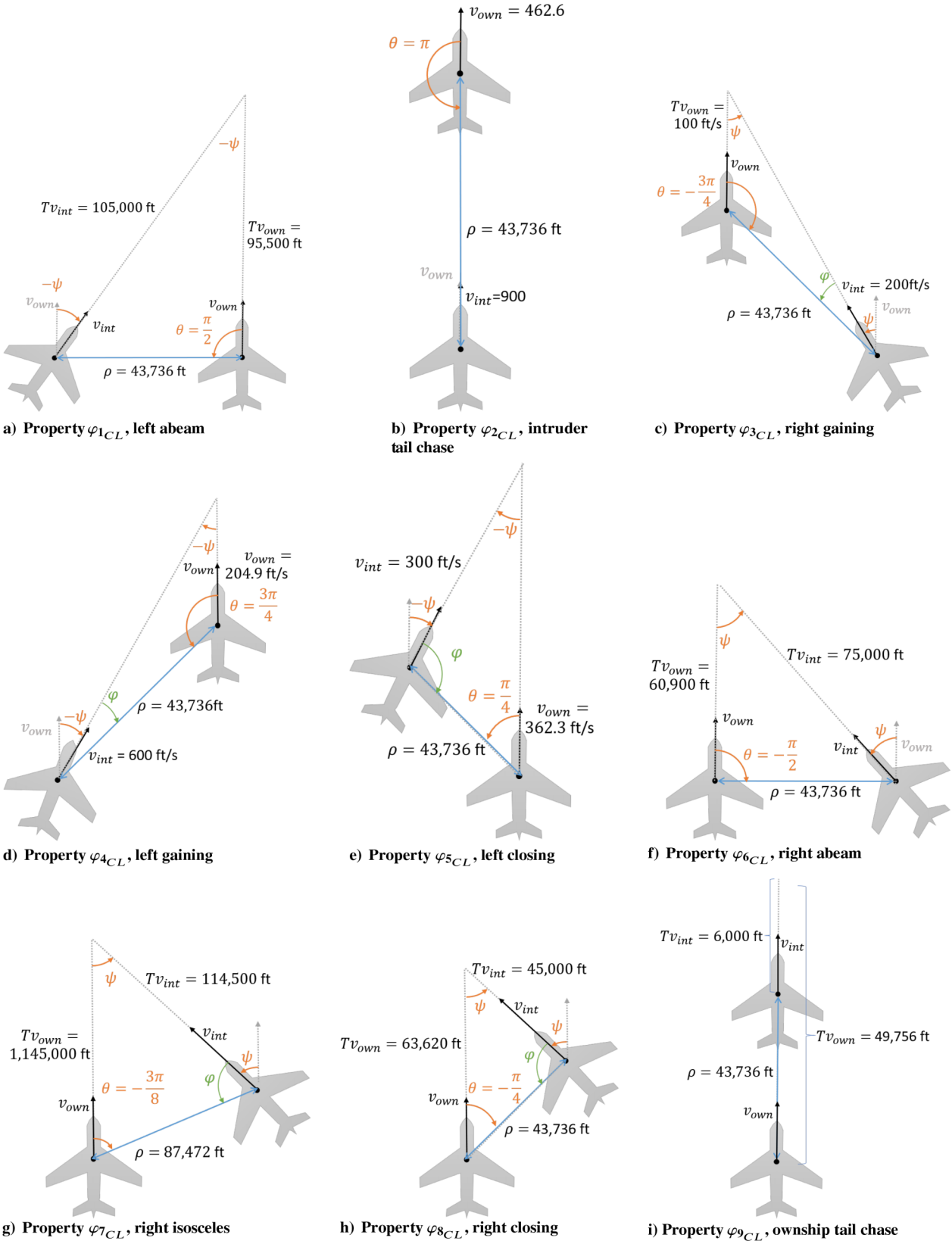
The names of the variables being processed by the next component are defined on top of the arrows, and their respective dimensions are specified in parentheses; i.e., outPlant (5) is a five-dimensional variable output in box 1 and normalized in box 2. It is assumed that the velocities and the flight altitudes of both aircraft are constant. Thus, the ACAS Xu neural network system is able to choose between the five neural networks corresponding to no time to loss of vertical separation ($\tau = 0$). The sampling time of 2 s is selected based on the dynamics of the airplane, for which doing it too often and changing the direction of the trajectory may lead to chattering, whereas doing it too infrequently may lead to crashes.

IV. Closed-Loop Verification Test Case Generation

The objective of the closed-loop verification tests is to demonstrate that the ownship and intruder aircraft remain safely separated given a set of initial conditions with uncertainty. The benchmarks presented here are not intended to provide complete coverage of the state space but rather provide a challenge for NNCS verification approaches with a collection of 10 closed-loop properties or test cases, defined in Table 3 and depicted in Fig. 5. The closed-loop

Table 3 ACAS Xu benchmark closed-loop test cases

Test point	Name	v_{int}, ft/s	θ, rad	ρ, ft	v_{own}, ft/s	ψ, rad	x_{int}, ft	y_{int}, ft
$\varphi_{1_{CL}}$	Left abeam	1,050	$\pi/2$	43,736	954.6	−0.4296	−43,736	0
$\varphi_{2_{CL}}$	Intruder tail chase	900	π	43,736	462.6	0	0	−43,736
$\varphi_{3_{CL}}$	Right gaining	200	$-3\pi/4$	43,736	100	0.3617	30,926	−30,926
$\varphi_{4_{CL}}$	Left gaining	600	$3\pi/4$	43,736	204.9	−0.5415	−30,926	−30,926
$\varphi_{5_{CL}}$	Left closing	300	$\pi/4$	43,736	362.3	−0.2379	−30933	30933
$\varphi_{6_{CL}}$	Right abeam	750	$-\pi/2$	43,736	609.3	0.6226	43,736	0
$\varphi_{7_{CL}}$	Right isosceles	1,145	$-3\pi/8$	87,472	1,145.9	0.7835	80,814	33,474
$\varphi_{8_{CL}}$	Right closing	450	$-\pi/4$	43,736	636.2	0.7577	30,926	30,926
$\varphi_{9_{CL}}$	Ownship tail chase	60	0	43,736	497.4	0	0	43,736
$\varphi_{10_{CL}}$	Head-on collision	600	0	120,000	600	π	0	120,000



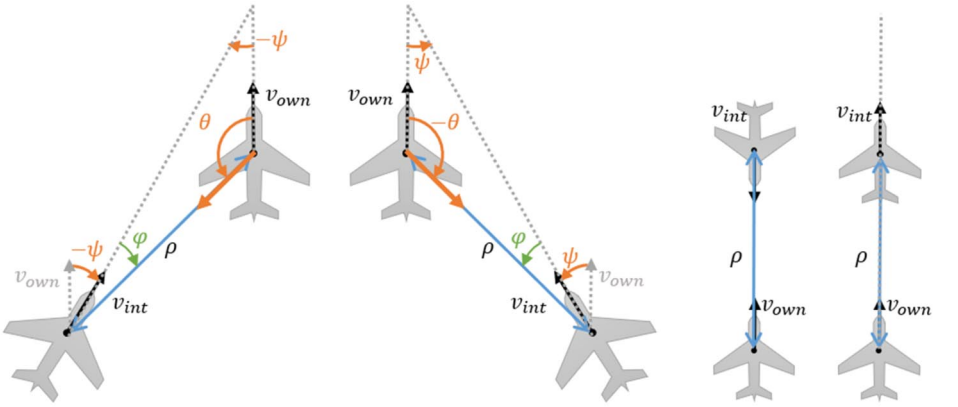
a) Property $\varphi_{1_{CL}}$, left abeam **b) Property $\varphi_{2_{CL}}$, intruder tail chase** **c) Property $\varphi_{3_{CL}}$, right gaining**

d) Property $\varphi_{4_{CL}}$, left gaining **e) Property $\varphi_{5_{CL}}$, left closing** **f) Property $\varphi_{6_{CL}}$, right abeam**

g) Property $\varphi_{7_{CL}}$, right isosceles **h) Property $\varphi_{8_{CL}}$, right closing** **i) Property $\varphi_{9_{CL}}$, ownship tail chase**

Fig. 5 Depiction of the aircraft encounter geometry for the closed-loop ACAS Xu verification properties.

properties presented in this paper are a complement to the open-loop properties: both of which sample portions of the state space. The closed-loop cases specifically sample from a set of cases on a collision course, whereas the open-loop cases do not specifically look at collision cases. A larger sampling outside of the 10 individual test cases is created by adding uncertainty about these points (for example, ± 5000 ft in x and ± 200 ft in y). Then, the set of states reachable from the initial set of states is found using reachability tools such as NNV or nnenum. These benchmarks are not intended to provide complete coverage of the state space but rather demonstrate the capabilities of the star-set-based verification approach (which is a more comprehensive approach with formal guarantees) as an alternative to Monte-Carlo- or design-of-experiments-based simulation analyses.



a) Closing left b) Closing right c) Head-on d) Tail chase

Fig. 6 Closed-loop test case geometry examples.

For the closed-loop benchmarks, the unsafe set is defined by the NMAC cylinder when there is a separation of less than 100 ft vertically or 500 ft horizontally. To scope the testing, the following ranges of variables are used: distances from zero to the maximum turn diameter ($\rho \in [0, 87{,}472]$ ft), the full range of angles to the intruder and heading of the intruder with respect to the ownship ($\theta \in [-\pi, \pi], \psi \in [-\pi, \pi]$), and the full range of reasonable velocities from a slow takeoff velocity to over Mach 1 ($v_{\text{int}} \in [60, 1145]$ ft/s, $v_{\text{own}} \in [100, 1145]$ ft/s). These limits, including different velocity ranges for the ownship and intruder, are inspired by previous work in defining open-loop verification properties [35]. By the law of sines, the variables used to describe the relative aircraft geometry are related with the time to collision T, as described in Eq. (4); and four examples (closing left, closing right, head-on, and tail chase) are depicted in Fig. 6. Any collision test case can be generated by fixing the time to collision T, θ, v_{int}, and ρ, as well as computing the remaining variables, as described in Table 4:

$$\frac{\rho}{\sin\psi} = \frac{v_{\text{own}}T}{\sin(\theta - \pi - \psi)} = \frac{v_{\text{int}}T}{\sin(2\pi - \theta)} \tag{4}$$

To generate a standard set of closed-loop test cases, θ, v_{int}, and ρ were selected from the following discretized sets of values and used to calculate appropriate values of v_{own} and ψ:

$$\theta \in \{-3\pi/4, -\pi/2, -3\pi/8, -\pi/4, 0, \pi/4, \pi/3, 3\pi/4, \pi\} \text{ rad}$$

$$v_{\text{int}} \in \{60, 150, 300, 450, 600, 750, 900, 1050, 1145\} \text{ ft/s}$$

and

$$\rho \in \{10{,}000,\ 43{,}736,\ 87{,}472,\ 120{,}000\} \text{ ft}$$

Using a Latin hypercube sampling of v_{int} and θ for baseline coverage, the test cases in Table 4 may be used to evaluate a range of properties across the state space. To compute the Cartesian coordinates of the intruder, Eqs. (5) and (6) are used. With the exception of the head-on collision test case (case 10), represented by Fig. 6c, each of the test cases are selected so that both aircraft are traveling in roughly the same direction, as depicted in Fig. 5. This is because aircraft are generally separated by at least 1000 ft in altitude when traveling in opposite directions, as commanded by air traffic control or federal regulations. For example, the under Title 14, Section 91.159 of the Code of Federal Regulations instructs pilots to fly at an odd altitude (plus 500 ft for pilots operating under visual flight rules) when traveling east on a heading between 0 and 179 deg, or an even altitude (plus 500 ft for pilots operating under visual flight rules) when traveling west on a heading between 180 and 359 deg [45]. This work did not consider nonstraight trajectories of the ownship and intruder, which would need to be considered in a more comprehensive safety assessment:

$$x_{\text{int}} = -\rho\sin(\theta) \tag{5}$$

$$y_{\text{int}} = \rho\sin(\pi/2 - \theta) \tag{6}$$

V. Analysis Approaches

This section describes the fundamental theory behind star sets used to approximate reachable sets, as well as the closed-loop neural network verification approaches. In terms of complexity, the reachability analysis of nonlinear ODEs is undecidable [46] and the NN reachability problem is nondeterministic polynomial-time (NP) complete [35,47]; thus, the reachability analysis of the NNCS with the nonlinear ODE plant is undecidable.

A. Star-Set Theory

A *star set* [48,49] (or simply star) Θ is a tuple $\langle \kappa, V, P \rangle$ where $\kappa \in \mathbb{R}^n$ is the center vector, $V = \{\mu_1, \mu_2, \cdots, \mu_{\text{m}}\}$ is a set of m basis vectors in $\mathbb{R}^n$, and $P: \mathbb{R}^m \to \{\top, \bot\}$ is a predicate. The set of states represented by the star is given as follows:

$$[\![\Theta]\!] = \{\omega | \omega = \kappa + \sum_{i=1}^{m}(\alpha_i \mu_i) \text{ such that } P(\alpha_1, \cdots, \alpha_m) = \top\} \tag{7}$$

Sometimes, both the tuple Θ and the set of states $[\![\Theta]\!]$ are referred to as Θ. In this work, the predicates are restricted to be a conjunction of linear constraints $P(\alpha) \triangleq H\alpha \le d$, where for p linear constraints $H \in \mathbb{R}^{p \times m}$, α is the vector of m variables (i.e., $\alpha = [\alpha_1, \cdots, \alpha_m]^T$) and $d \in \mathbb{R}^{p \times 1}$. A star is an empty set if, and only if, $P(\alpha)$ is empty.

Any convex polyhedron can be represented as a star set. An affine mapping of a star set $\Theta = \langle \kappa, V, P \rangle$ defined by a mapping matrix W

Table 4 Computing ownship velocity and relative heading from randomly selected angle to intruder, intruder velocity, and distance to the intruder

	Closing left	Closing right	Head-on	Tail chase
θ	$0 < \theta < \pi$	$-\pi < \theta < 0$	$\theta = \pm\pi$	$\theta = 0$
ψ	$-\sin^{-1}\left(\frac{\rho}{Tv_{\text{int}}}\sin(\lvert\theta\rvert)\right)$	$\sin^{-1}\left(\frac{\rho}{Tv_{\text{int}}}\sin(\lvert\theta\rvert)\right)$	0	0 or π
φ (from sum of angles)	$\pi - \lvert\theta\rvert - \lvert\psi\rvert$	$\pi - \lvert\theta\rvert - \lvert\psi\rvert$	— —	— —
v_{own}	$\frac{v_{\text{int}}\sin(\varphi)}{\sin(\lvert\theta\rvert)}$	$\frac{v_{\text{int}}\sin(\varphi)}{\sin(\lvert\theta\rvert)}$	$\frac{Tv_{\text{int}} - \rho}{T}$	$\frac{Tv_{\text{int}} + \rho}{T}$

and an offset vector b is another star set $\Theta' = \langle W\kappa + b, WV, P\rangle$. The intersection of a star set $\Theta = \langle \kappa, V, P\rangle$ with a polyhedron $G \triangleq H_G\alpha \le d_G$ is another star set $\Theta' = \langle \kappa, V, P \wedge P_G\rangle$, where $P_G \triangleq H_G \times V\alpha \le d_G - H_G \times \kappa$.

B. Neural Network Verification Tool

The NNV[**] tool is one of the two verification tools used to analyze the safety and functionality of the ACAS Xu NN system. This tool is evaluated on the 10 closed-loop benchmarks where the safety of the ownship is analyzed with respect to one intruder aircraft. The analysis consists of evaluating, under multiple scenarios and initial state uncertainty, the safety of the ownship aircraft using Algorithm 1, which follows the general computation flow to the one described in Sec. III. For the nonlinear plant reachability analysis, NNV makes use of a CORA [50] integration.

Algorithm 1: ACAS Xu NNCS reachability algorithm in NNV

Result:
Rs: Plant reachable states
Initialize
initSet // Initial state set
a_{prev} // Previous advisory
tF // Number of control steps
safe = True // Safety variable, change to false if NMAC violated
v_{int} // Intruder velocity
v_{own} // Ownship velocity
while *safe*, **do**
  for $(i = 0; i < tF; i = i + 1)$, **do**
    [state_set,outPlant] = plantReach(initSet, a_{prev}) // reachability analysis of the plant
    ρ = extractDist(state_set) // Extract distance set
    if $\rho < 500$, **then**
    safe = False
    end
    $input_{NN}$ = normalizeNNinputs(outPlant) // normalize inputs to the neural network
    adv_{NN} = NNreach(a_{prev}, $input_{NN}$) // reachability analysis of the NN controller; see Algorithm 2
    initSet = state_set
    $a_{prev} = adv_{NN}$
    Rs[i] = state_set
  end
End
return: Rs

The sets are represented using star sets, and the reachability functions compute overapproximations of the reachable sets. Due to the initial uncertainty added to the initial states of the aircraft, it is possible that multiple advisories are issued at each time step, increasing the number of possible trajectories of the ownship. In this case, the reach sets are split and paired with the corresponding advisory to decrease the number of operations and trajectories computed in order to reduce the conservativeness and memory consumption. The workflow of this algorithm implemented in NNV using star-set methods is presented in Algorithm 2. The importance of this algorithm is illustrated with the following example with one initial advisory and one initial state set, where the number of operations is exponentially reduced. For the purpose of this example, it is assumed that at each time step, there are two possible advisories ($output_{NN}$) issued for each plant output ($output_P$) set computed.

1. Example 1

In step 1, there is one $output_P$ set computed based on the initial state and previous advisory. For this $output_P$ set, there are two possible advisories ($output_{NN}$) computed by a NN. In step 2, for each advisory, one $output_P$ is computed based on the initial state, obtaining two sets. For each possible combination of advisories and $output_P$ sets (four), there are two advisories issued, increasing the number of advisories to eight. However, because there is a limit of five maximum unique advisories (*COC, WL, WR, SL, SR*), these five advisories are used for the remainder of the example. In step 3, there are five advisories and two state sets, increasing the number of plant $output_P$ sets to 10 in step 3. By computing all possible combinations between the 10 $output_P$ sets and five advisories, there are a total of 50 NN $output_{NN}$ sets to compute, although the number of advisories computed is limited to five as previously described. In step 4, there are now 10 initial sets and five advisories: out of which, 50 plant $output_P$ sets are computed. Following this procedure, it is observed that the number of $output_P$ sets computed grows by a multiple of five at each future control step: i.e., in step 5, there are 250 plant $output_P$ sets; in step 6, there are 1250 sets; and so on.

2. Example 2

The number of plant $output_P$ sets can be significantly reduced by using Algorithm 2. In step 1, there is one $output_P$ set, for which there are two possible advisories computed by a NN. In step 2, based on the initial state set, for each of the two advisories, the $output_P$ set of the plant is computed, obtaining two sets. By tracking which set corresponds to which advisory, the number of operations is reduced from four to two. For each combination, there are two advisories issued, increasing the number of advisories to four. In step 3, there are four advisories and two $output_P$ sets. For each advisory, the corresponding plant $output_P$ set is computed for a total of four plant $output_P$ sets. Then, for each $output_P$ set, there are two advisories issued for a total of eight advisories. In step 4, the relationship between the advisories and the plant $output_P$ sets computed is one to one; therefore, eight sets are obtained. Following this pattern, it is observed that the number of plant $output_P$ sets increases by two for each control step: i.e., in step 5, there are 16 plant $output_P$ sets; in step 6, there are 32 sets; and so on. If both examples are run for 10 control steps, the total number of $output_P$ sets are reduced from 781,250 to 512.

Algorithm 2: Reachability algorithm of switching classification-based controllers: NNreach

Result:
adv_{NN}: Advisories paired with each plant reach set
Inputs
a_{prev} // Previous advisory
$input_{NN}$ // inputs to NNs
N // number of reach sets and previous advisory pairs
$k = 0$
for $(i = 0; i < N; i = i + 1)$, **do**
  $output_{NN}$[i] = Reach($input_{NN}$[a_{prev}[i,1]], a_{prev}[i,0]) // Compute reach set of neural network
  advise[i] = argminNN($output_{NN}$[i]) // Compute advisory based on minimum value of output reach set $output_{NN}$
  for adv in advise[i], **do**
    adv_{NN} [k] = [adv, i] // Track advisories
    k++
  end
End
return: adv_{NN}

C. Neural Network Enumeration Tool

The second verification tool used for analysis is nnenum,[††] which uses star sets to reason over possible neural network outputs given sets of input states. Note that nnenum combines many engineering

[**]Data available online at https://github.com/verivital/nnv [retrieved 3 January 2022].

[††]Data available online at https://github.com/stanleybak/nnenum [retrieved 24 August 2022].

improvements to the base star-set method [51], such as using quick bounds estimates using zonotopes with domain contraction to improve accuracy. This is further combined with an abstraction-refinement approach [32], where star sets first analyze the system with over-approximation; and then they refine by splitting sets on individual neurons only if it is needed to prove the safety property.

In this work, a set of states is first analyzed using the quicker overapproximation mode of nnenum to check whenever multiple advisories are possible. Otherwise, exact analysis is used, which splits the set of states into many smaller states: each with a unique classification. For the plant dynamics, rather than using CORA and the nonlinear differential equations in Eq. (3), nnenum instead performs numerical simulation within each set using the original Dubins car dynamics in Eq. (1) and observations from Eq. (2) in order to produce a local numerical linearization of the observed state variables. Because reachability analysis through linear differential equations is more efficient than reachability with nonlinear equations, the method has better expected scalability. This result is demonstrated in our evaluation section next (Sec. VI), where larger uncertainties in the initial state can be checked with accuracy that visually resembles simulations of the closed-loop system.

VI. Results

The 10 closed-loop test cases depicted on Fig. 5 are evaluated using NNV and nnenum under specified initial uncertainty. For each scenario, initial ownship uncertainties of ± 5000 in the x direction and ± 200 in the y direction are used when evaluated. Although there are some small differences in two out of the 10 cases evaluated between NNV and nnennum ($\varphi_{2_{CL}}$ and $\varphi_{10_{CL}}$), both tools are able to successfully verify the safety of the ownship aircraft with respect to the intruder in all 10 cases. These results are depicted in 10 figures, which consist of five subplots organized as follows: Figs. 5a–5c correspond to nnenum results; and Figs. 5d and 5e correspond to NNV. For each tool, there are zoomed-in (Figs. 5b and 5d) simulation results as well as the reachable sets (Figs. 5c and 5e), whereas Fig. 5a depicts the complete simulation trajectories of the ownship in nnenum.

To illustrate and explain the results from this investigation, two out of the 10 test cases are visualized in this section: right closing $\varphi_{8_{CL}}$ (Fig. 7), and head-on collision $\varphi_{10_{CL}}$ (Fig. 8). The remaining results are found in Figs. 9–16 in Appendix A. This set of results can be summarized as the ownship is always safe, maintaining a much greater distance than the safety distance specified in the NMAC. This is accomplished by turning to the opposite side from which the intruder is approaching, as depicted in Figs. 7 and 9 as well as Figs. 11–15. In the other three scenarios, the ownship is biased to deviate toward the right in order to avoid a collision with the intruder when both are flying in the same path but the opposite direction or the same direction and a different speed. This bias is very evident in Fig. 8, and it is observed in Fig. 16, in which the ownship reachable sets split into two main paths: one toward the right, and one toward the left.

If the ownship is initially located between approximately -2000 ft and $+5000$ in the x axis, the ownship will deviate to the right. If initially located to the left of -2000 ft in the x axis, then the ownship will avoid the collision by turning to the left. The final scenario, which is intruder tail chase $\varphi_{2_{CL}}$, does not present this bias and behaves more as initially predicted. This result is demonstrated in Fig. 10 by both NNV and nnenum, although the tools present slightly different reachable set trajectories. The initial state set is split into two halves at $x = 0$ ft in both simulations and reachable sets. If the ownship is initially located in the positive x axis, it will turn to the right, and vice versa.

The first test case, which is right closing $\varphi_{8_{CL}}$, is illustrated in Fig. 7. It is observed in the simulation plots that when the ownship approaches the intruder's trajectory at approximately $y = 25{,}000$ ft, the controllers start issuing weak and strong left commands to the ownship to maintain a safe distance with the intruder. These decisions are especially evident in the zoomed-in and reach set plots, where the trajectories turn toward the left for a few steps, followed by clear of conflict commands as the ownship safety is maintained. On the other scenario in Fig. 8, the head-on collision $\varphi_{10_{CL}}$ test case demonstrates the ownship's safety when the ownship faces another aircraft in the same path but in the opposite

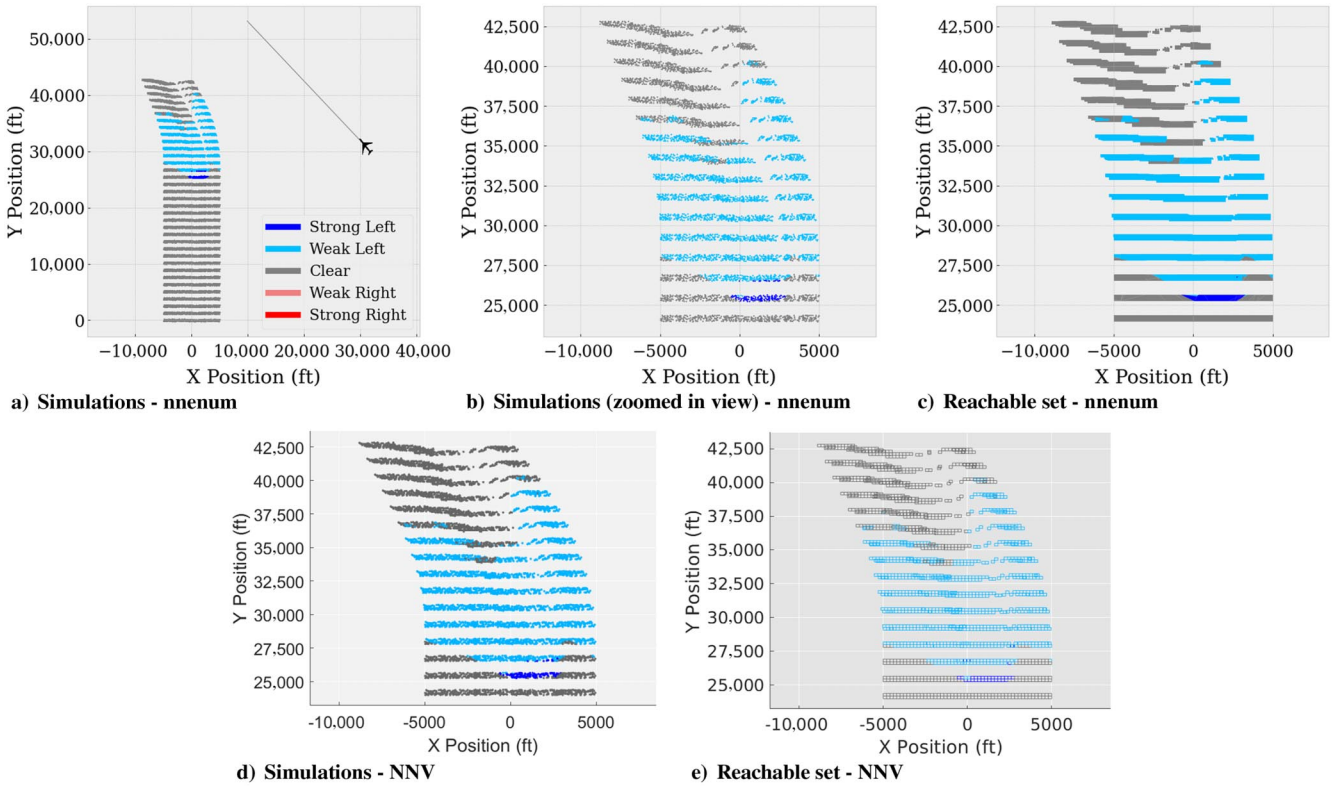


Fig. 7 Right closing $\varphi_{8_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.

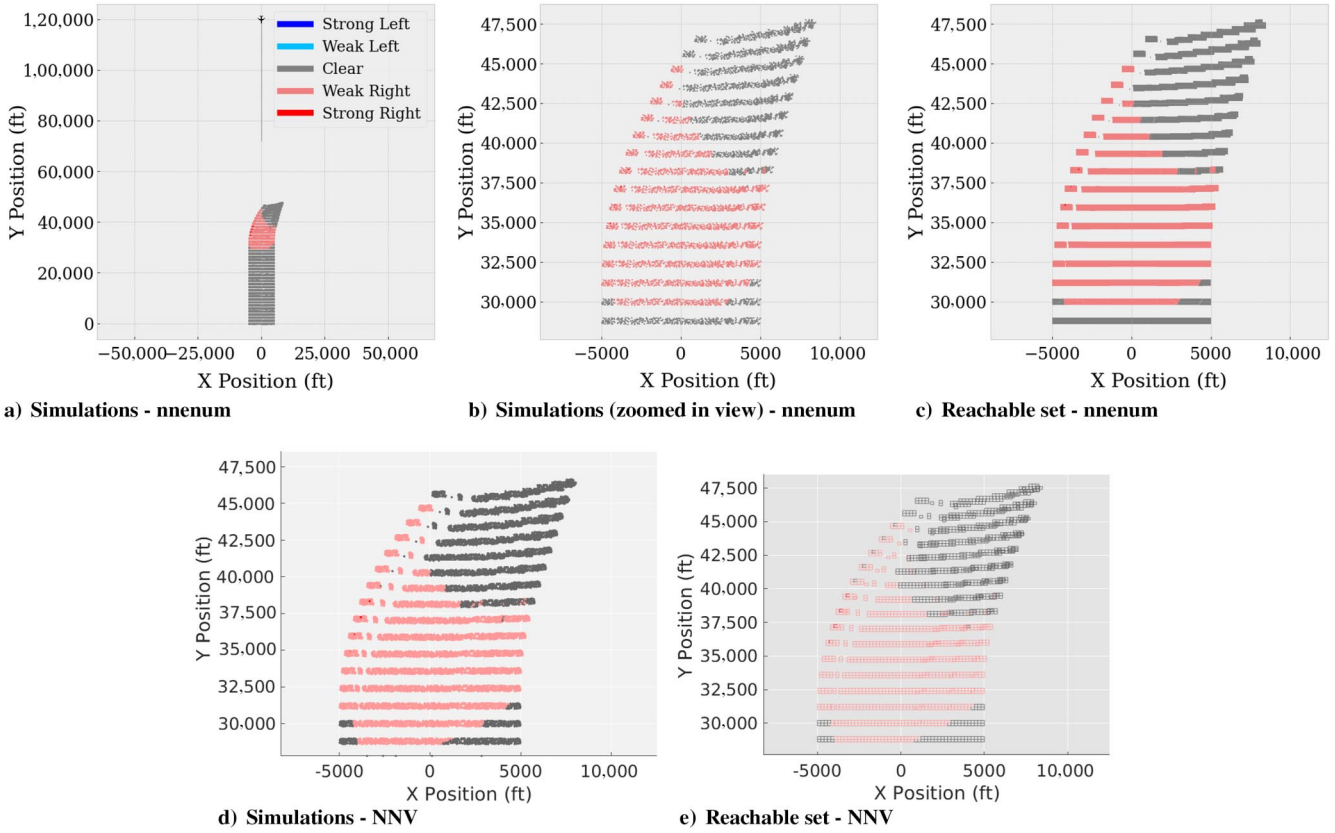


Fig. 8 Head-on collision $\varphi_{10_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.

direction. Both tools show very similar results as the ownship deviates to the right when flying at approximately $y = 30{,}000$ ft, issuing only weak right commands to maintain safety with respect to the intruder.

The similarity between the simulation and verification results observed in these plots is an indication of the reduced overapproximation errors computed by both tools using star-set methods. Despite both tools successfully verifying the safety of the ownship with respect to the intruder aircraft in all 10 cases, there are some main differences between nnenum and NNV, which are discussed in the next section.

VII. Discussion

The results in Sec. VI and in Appendix A demonstrate the capabilities of nnenum and NNV for the verification of real-world unmanned CPSs, where some of the main differences between these tools can also be observed. NNV needs to compute an initial partition on the initial set and then compute the reachable sets for each partition, whereas nnenum is able to compute these partitions as needed at each control step, improving its efficiency with respect to NNV. Another main difference is the nonlinear ODE reachability analysis, where NNV is also capable of computing and displaying the complete continuous-time reachable sets of NNCS; but, for consistency purposes, only the reachable sets of the ownship at each control step are shown for ease of comparison against nnenum. The results also show some of the complexities of deploying and verifying these complex systems. In eight out of 10 cases, NNV and nnenum obtain the same results; but, there are two cases in which there are differences in both the simulations and the reachability analysis. There is a specific step in each scenario in which NNV and nnenum do not issue the same advisory commands. In test case $\varphi_{10_{CL}}$ (Fig. 8), the main difference is in the first reachable set shown, in which nnenum only selects *COC* commands, whereas NNV also has some weak right commands toward the center of the reachable set. The other test case, which is $\varphi_{2_{CL}}$ (Fig. 10), seems similar to the previous test case in the sense that there is one main step in the reachable set computation that differs between NNV and nnenum (step 5) in both sides of the path, which leads to different final paths. To complement the reachability analysis presented and try to better understand the differences between NNV and nnenum, the simulation differences between NNV and nnenum have been analyzed across 1000 randomly initialized simulations; and the results are introduced in Table 5.

It is observed that even in the test cases in which NNV and nnenum return the same results, there are still some minor differences between the two. Some possible sources of error are the different solvers used for the computation of the states of the plant, or a slightly different implementation on the plant model, among others. NNV computes the reachability analysis of the plant model using all nine state variables described in Eq. (3); but, nnenum makes use of only the first six variables, and then it uses those to compute the remaining three (7 to 9) a posteriori. These differences in computation may lead to small computation differences between the two, which accumulate through time and eventually lead to slightly different final paths, although both tools successfully verify the safety of the ownship in every case. Another main difference between NNV and nnenum is the different reachability schemes used, which lead to significant differences in the computation time. In average, nnenum is able to compute the reachable set of each closed-loop property in 172 s, whereas NNV completes each test case in 140,819 s, which is approximately 39 h. The main reasons for the difference in computation time are the different partitioning routines and the plant reachability techniques. When compared to other validation techniques such as Monte Carlo simulations, these reachability techniques can prove and guarantee the safety of the ownship under the whole initial uncertainty of the ownship aircraft, whereas Monte Carlo simulations can only provide probable guarantees, which are determined by the number of simulations run. In terms of computation time, nnenum is equivalent to running 0.13 simulations per square foot of the initial set of the ownship, whereas NNV is equivalent to running 114 simulations per square foot. In this aspect,

Table 5 Simulation differences between NNV and nnenum across all 10 closed-loop properties[a]

	Closed-loop properties									
	$\varphi_{1_{CL}}$	$\varphi_{2_{CL}}$	$\varphi_{3_{CL}}$	$\varphi_{4_{CL}}$	$\varphi_{5_{CL}}$	$\varphi_{6_{CL}}$	$\varphi_{7_{CL}}$	$\varphi_{8_{CL}}$	$\varphi_{9_{CL}}$	$\varphi_{10_{CL}}$
	x_{own}									
Mean	0.24	243.29	$9.9e-11$	$7.4e-10$	$5.6e-11$	$1.4e-09$	$2.3e-9$	$2.1e-9$	$9.5e-10$	71.86
Standard deviation	0.039	355.89	$1.5e-10$	$5.6e-10$	$2.0e-10$	$1.5e-9$	$3.3e-9$	$2.8e-9$	$1.0e-9$	105.41
Maximum	0.95	$1.1e+3$	$3.7e-10$	$1.3e-9$	$8.8e-10$	$3.7e-9$	$8.9e-9$	$6.2e-9$	$2.3e-9$	270.77
	y_{own}									
Mean	0.0562	51.58	$4.8e-11$	$1.9e-10$	$1.3e-9$	$2.9e-10$	$7.3e-10$	$1.1e-10$	$1.2e-10$	14.54
Standard deviation	0.098	75.94	$7.7e-11$	$1.7e-10$	$1.5e-9$	$3.8e-10$	$1.3e-9$	$1.5e-10$	$1.3e-10$	24.42
Maximum	0.26	241.61	$2.7e-10$	$4.2e-10$	$4.7e-9$	$1.0e-9$	$3.9e-9$	$4.6e-10$	$3.4e-10$	68.43
	ψ_{own}									
Mean	$1.2e-5$	0.039	$8.3e-16$	$6.3e-15$	$3.2e-16$	$2.6e-15$	$1.6e-15$	$2.0e-15$	$2.3e-15$	0.0069
Standard deviation	$3.2e-5$	0.048	$1.6e-15$	$4.9e-15$	$1.1e-15$	$2.8e-15$	$2.6e-15$	$3.0e-15$	$2.6e-15$	0.011
Maximum	$1.0e-4$	0.11	$5.7e-15$	$1.2e-14$	$4.9e-15$	$8.6e-15$	$8.5e-15$	$7.7e-15$	$5.7e-15$	0.025

[a]For each experiment, 1000 random simulations were sampled. Note that x_{own} and y_{own} values are in feet and ψ_{own} are in radians; they correspond to the average values of the absolute difference between the recorded variables in NNV and nnenum.

nnenum is preferable over Monte Carlo simulations and NNV due to its efficient computation of the reachable sets of this ACAS Xu closed-loop benchmark.

VIII. Conclusions

This paper introduced a set of 10 new closed-loop verification properties and developed and applied a closed-loop verification approach to verify both the output of NNCS and the switching behavior between neural network controllers. The approach was demonstrated using the NNV and nnenum tools on five of 45 ACAS Xu neural networks, with switching between all five corresponding to co-altitude collision cases. Both tools show reachable states of an ownship aircraft with modified Dubins dynamics on a collision course with an intruder aircraft. The reachability computation considers initial state uncertainty and ownship control inputs provided by five switched NNs. The verification objective was to show that the switch NNCS resulted in collision-free courses, and even a large initial uncertainty. This study showed that it is possible to do a comprehensive safety verification analysis of a complex air collision advisory system for aircraft traveling in straight co-altitude paths, but further research is needed to determine if climbing or descending flights can also be proven safe using these tools. In the tool demonstration, nnenum showed a tighter and faster computation of the overapproximation of its reachability methods, as observed by the similarity between the simulation and reachable plots, whereas NNV is able to present the complete continuous-time reachable sets of the ownship trajectories. A summary of the main outcomes of the current methods can be summarized as follows:

1) For an initial assessment, a small amount of Monte Carlo simulations may be sufficient (in combination with other testing routines) to check the proper behavior of the CPS and its implementation.

2) In safety-critical applications, Monte Carlo simulations cannot provide formal guarantees, which are vital for safety-critical systems such as autonomous aircraft. In this case, formal verification tools such as nnenum and NNV would be needed.

3) NNV and nnenum can both provide formal guarantees on the safety of the aircraft using reachability analysis for neural networks as well as nonlinear ODEs.

4) Also, nnenum has proven able compute the reachable sets of the NNCS much faster than NNV, by about four orders of magnitude faster, largely due to its efficient partitioning scheme and the nonlinear ODE reachability method.

Appendix A: Closed-Loop Verification Results

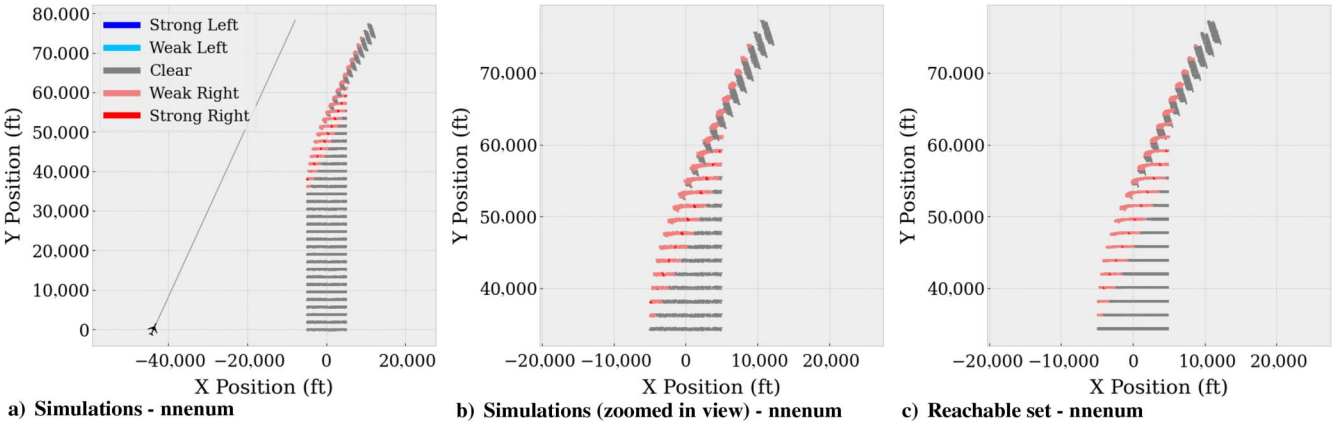
The ACAS Xu NN compression closed-loop verification benchmark presents a total of 10 properties described in Sec. IV, specified in Table 3, and illustrated in Fig. 5. In Sec. VI, a summary of the verification results for all 10 properties was presented, but only properties $\varphi_{8_{CL}}$ and $\varphi_{10_{CL}}$ were depicted. In this appendix, the remainder of the verification results are illustrated in Figs. 9–16.

Appendix B: Open- Verification Properties

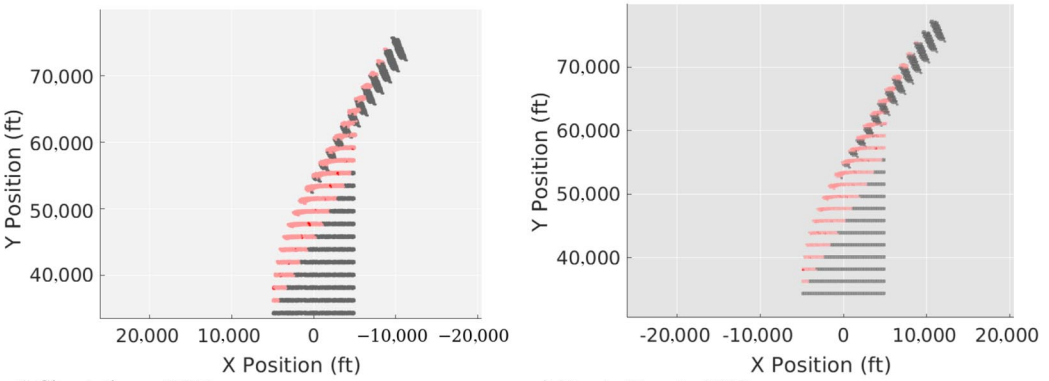
An ACAS Xu verification benchmark proposed 10 open-loop properties of the neural network approximation. These properties are summarized in this appendix with visual depictions of the geometry in Fig. B1 to provide the reader with more intuition into how these neural networks have been verified in the past and how this compares to the closed-loop test cases. It is worth noting that these neural networks assign a value to each of the five possible outputs with the convention that the *lowest* output is the best choice (in other neural network designs, the highest input may be the best choice).

The open loop properties are defined in Table 6, and can be described as follows. Property φ_1 may select a value of ρ (approximately 10.5 miles away) that is larger than the maximum turning radius of an aircraft going at 1145 ft/s (43,736). This maximum velocity of 1145 ft/s is approximately 780 mph, which as a conservative upper bound is much larger than what a commercial aircraft would typically fly (Mach 1 is 761 mph at sea level, decreasing to 678 mph at a 30,000 ft typical commercial aircraft cruise altitude). The 60 ft/s (40 mph) velocity of the intruder in this property is a conservative lowest velocity for a fixed-wing aircraft. Although the selection of variables seems reasonable, this first property is not necessarily a good property because it specifies a value of the output that is an artifact of the design choices for the neural network rather than an ordinal ranking of the desired outcome for the aircraft encounter. In other words, a better property might specify that the selected action produced by the neural network should be clear of conflict. Property φ_2 is a slightly better variation of φ_1 because it tests that clear of conflict is not the last choice of the neural network for a subset of the conditions of φ_1.

Properties φ_3 and φ_4 check that the neural network does not output clear of conflict in cases where the intruder is ahead of the ownship and a collision is imminent. Property φ_3 checks when the two aircraft are approaching for a head-on collision within less than 1 s (unless there is sufficient vertical separation). Property φ_4 checks when the ownship is directly behind and closing on the

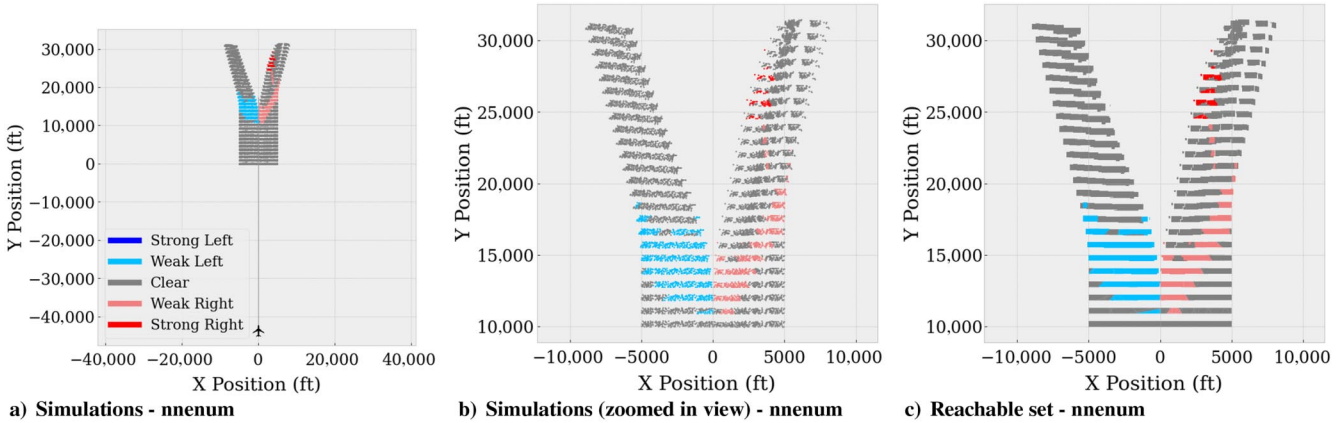


a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum

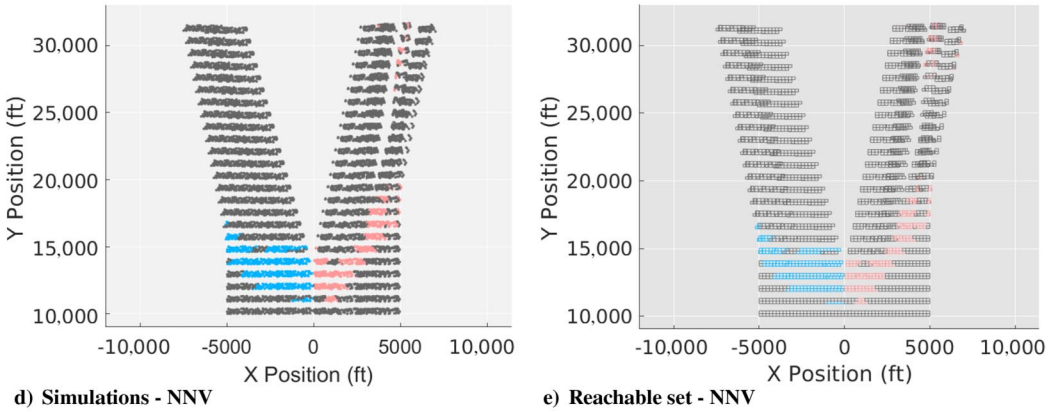


d) Simulations - NNV e) Reachable set - NNV

Fig. 9 Left abeam $\varphi_{1_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.

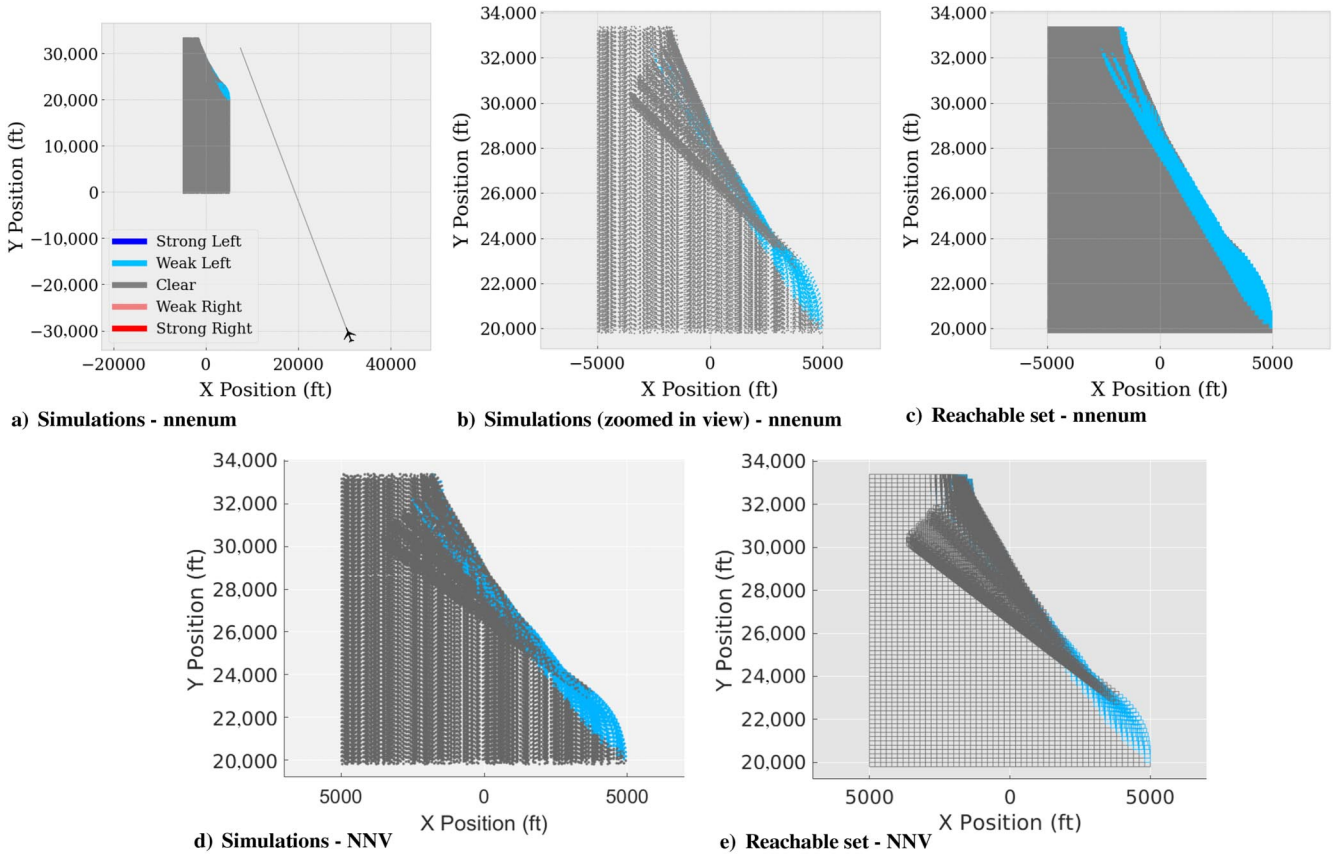


a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum



d) Simulations - NNV e) Reachable set - NNV

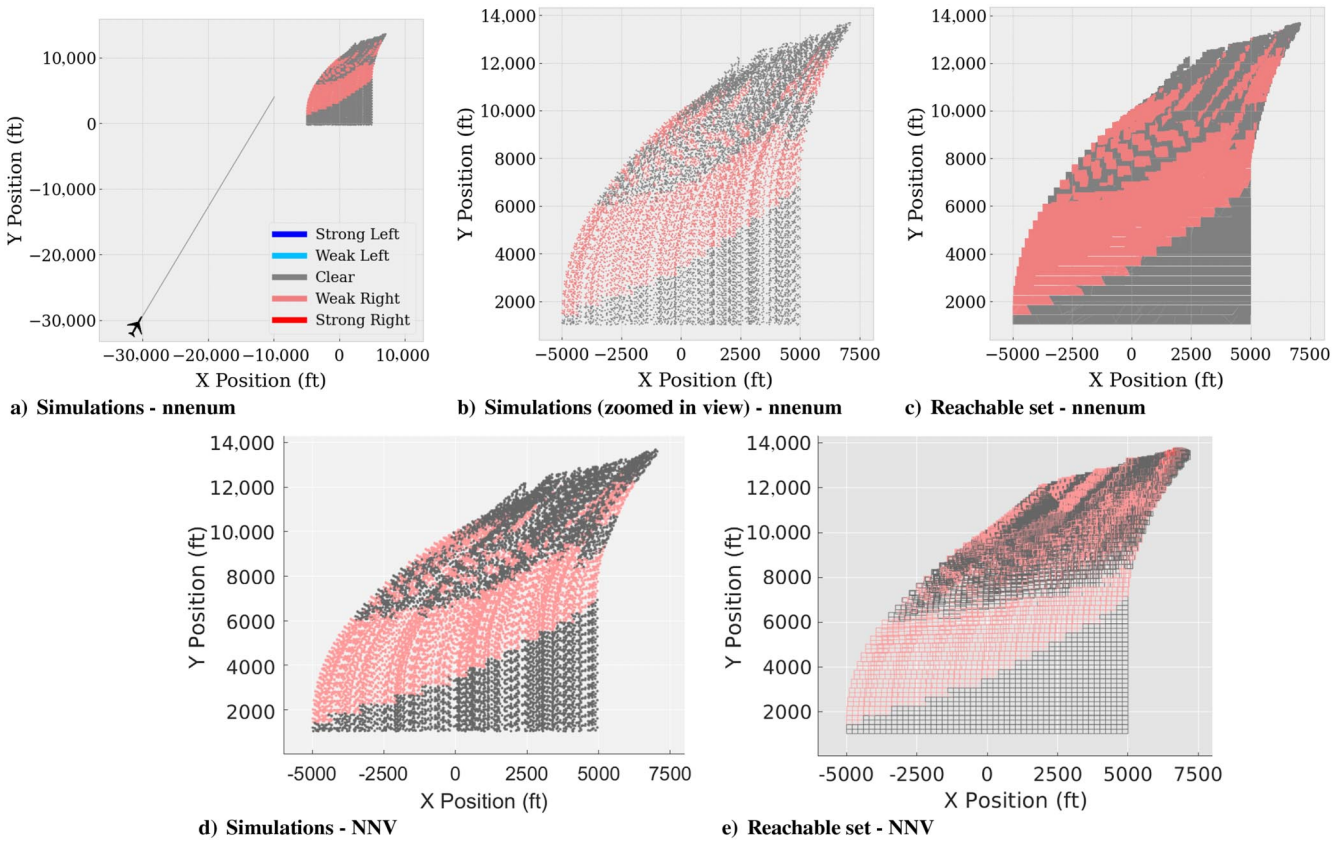
Fig. 10 Intruder tail chase $\varphi_{2_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.



a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum

d) Simulations - NNV e) Reachable set - NNV

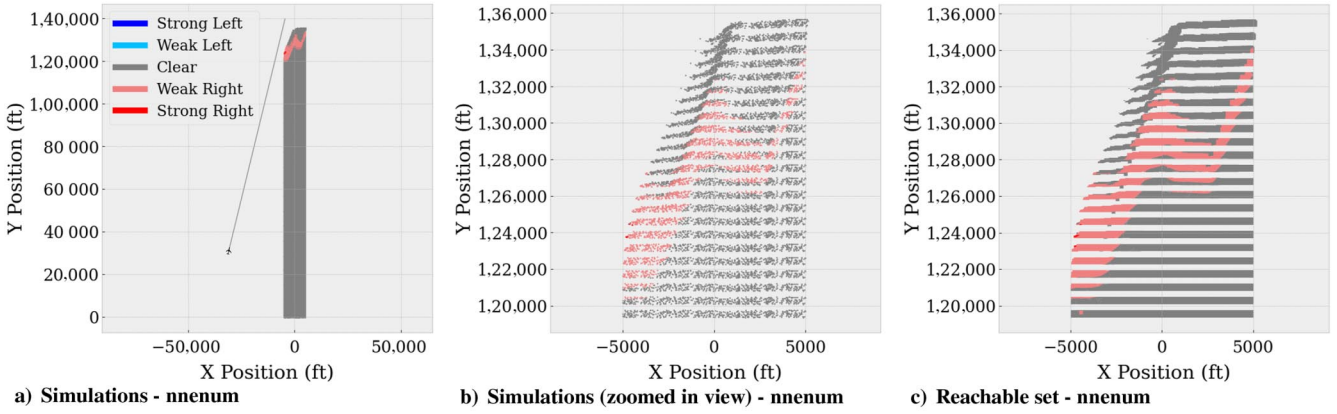
Fig. 11 Right gaining $\varphi_{3_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.



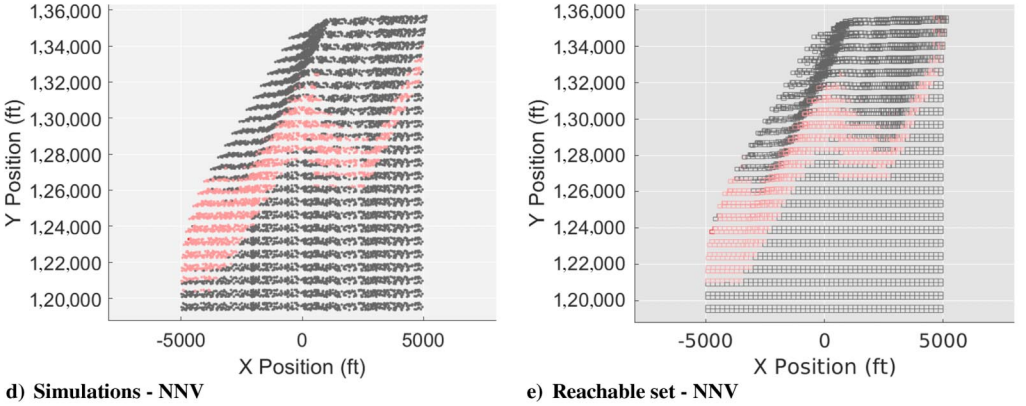
a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum

d) Simulations - NNV e) Reachable set - NNV

Fig. 12 Left gaining $\varphi_{4_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.

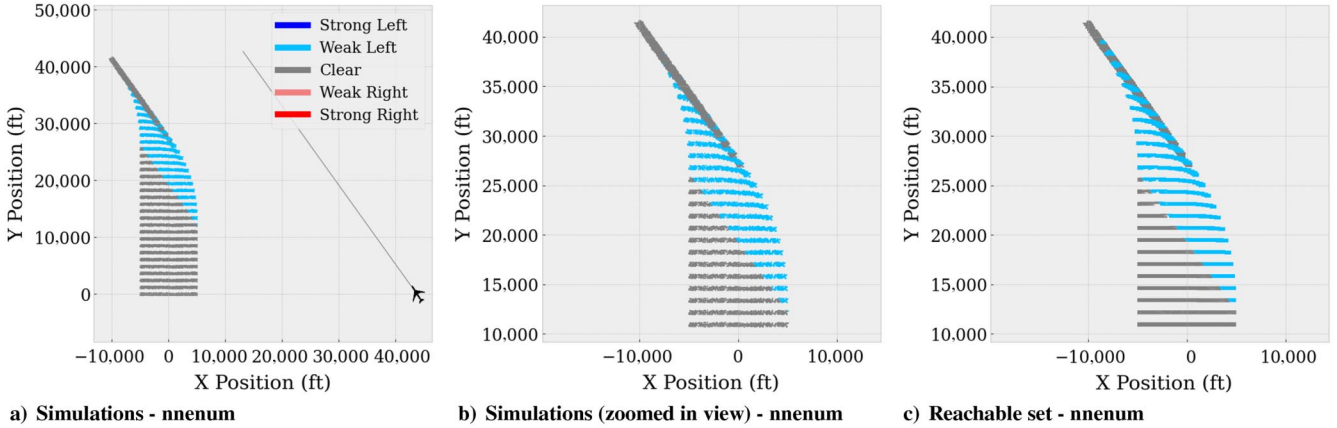


a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum

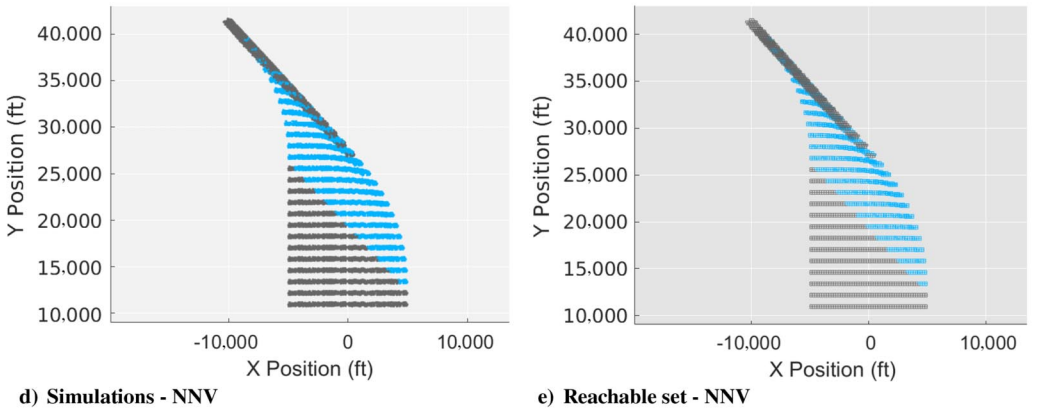


d) Simulations - NNV e) Reachable set - NNV

Fig. 13 Left closing $\varphi_{5_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.

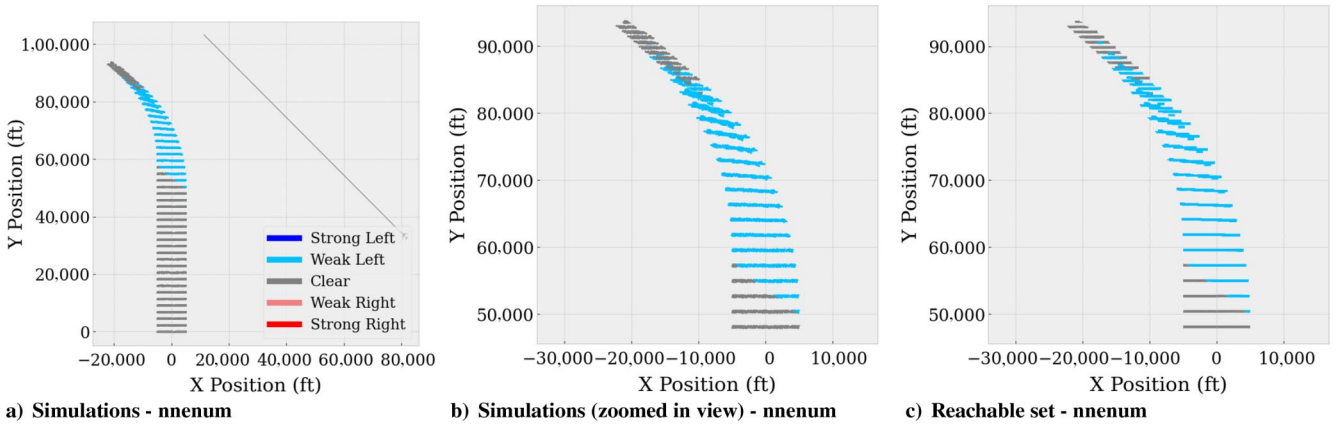


a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum

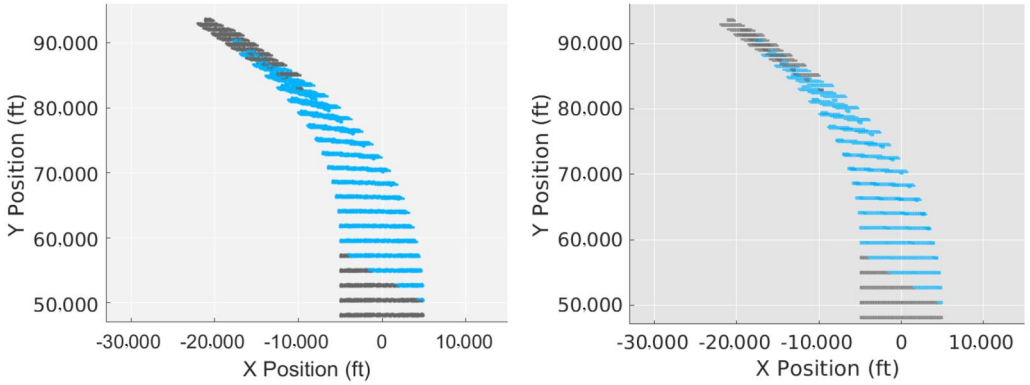


d) Simulations - NNV e) Reachable set - NNV

Fig. 14 Right abeam $\varphi_{6_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.

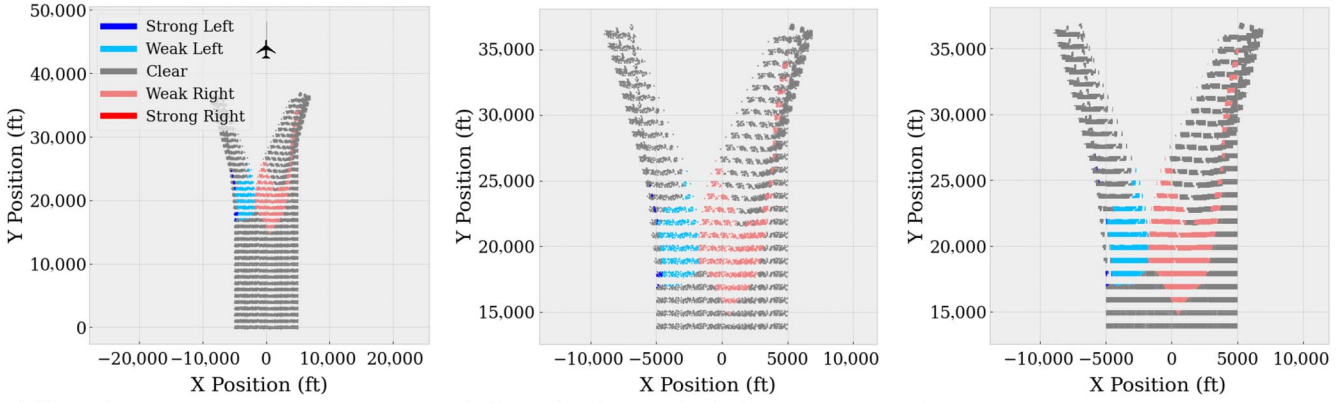


a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum

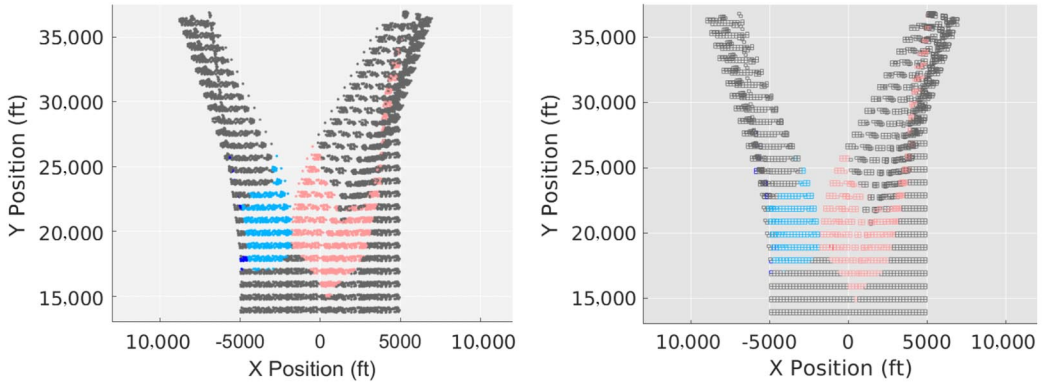


d) Simulations - NNV e) Reachable set - NNV

Fig. 15 Right isosceles $\varphi_{7_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.

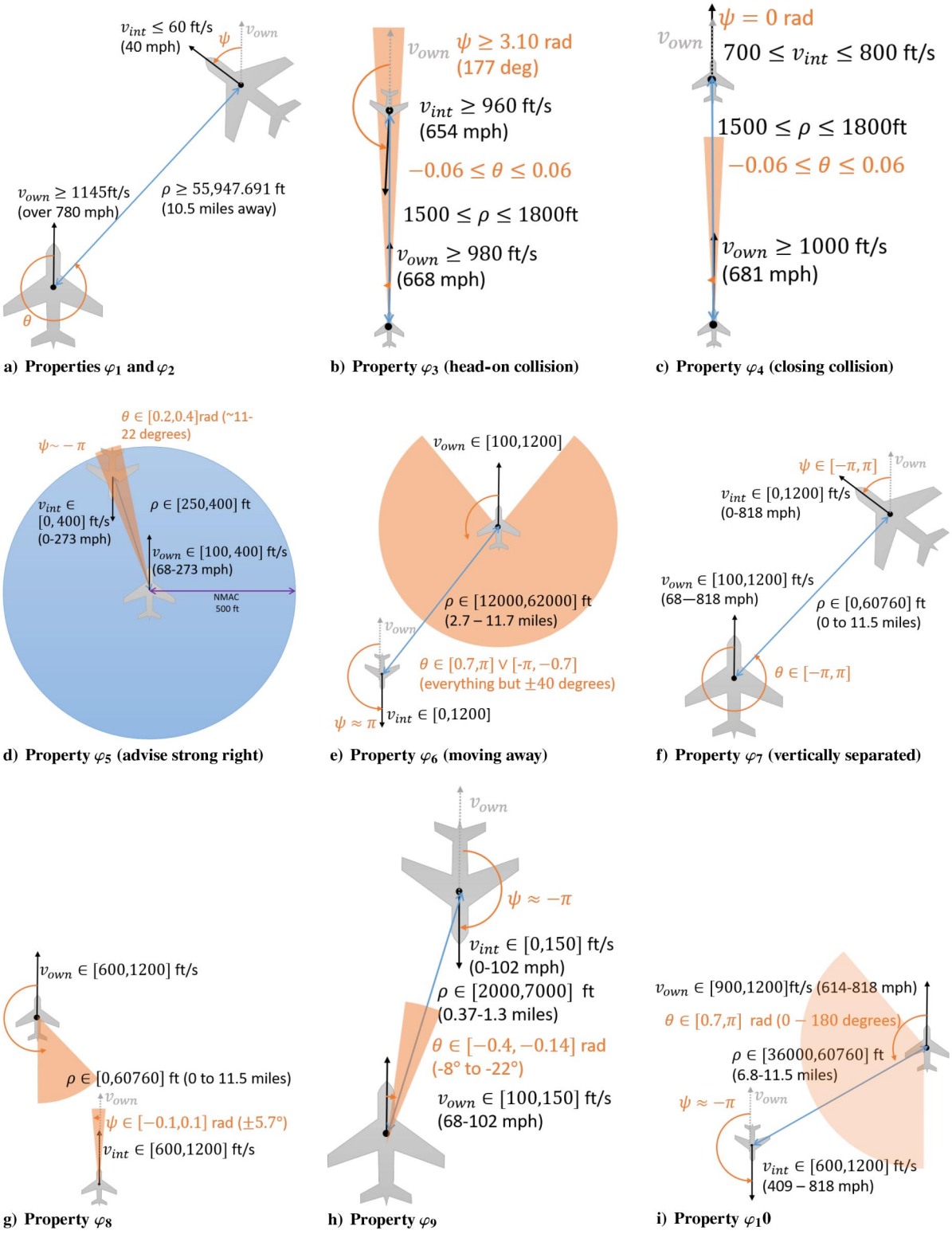


a) Simulations - nnenum b) Simulations (zoomed in view) - nnenum c) Reachable set - nnenum



d) Simulations - NNV e) Reachable set - NNV

Fig. 16 Ownship tail chase $\varphi_{9_{CL}}$ with initial ownship uncertainty $x_0, y_0 = [\pm 5000, \pm 200]$.



a) Properties φ_1 and φ_2 **b) Property φ_3 (head-on collision)** **c) Property φ_4 (closing collision)**

d) Property φ_5 (advise strong right) **e) Property φ_6 (moving away)** **f) Property φ_7 (vertically separated)**

g) Property φ_8 **h) Property φ_9** **i) Property φ_10**

Fig. B1 Depiction of the aircraft encounter geometry for the open-loop ACAS Xu verification properties.

intruder at a rate of 200–300 ft/s, which will result in a collision in less than 9 s.

The remaining six properties check a wide range of conditions. Property φ_5 checks that the neural network favors the strong right advisory if the intruder is passing very close to the ownship's left. Property φ_6 checks that the neural network outputs clear of conflict if the intruder is sufficiently far away behind and moving away from the ownship. Property φ_7 checks that the neural network does not output a strong left or right advisory when vertical separation is large ($\tau = 100$ s) and the previous advisory was clear of conflict. Property φ_8 checks that the neural network outputs weak left or clear of conflict as the minimal score when the previous advisory was weak left, there is 100 s until a loss of vertical separation, and the intruder is behind and to the right of the ownship. Property φ_9 checks that the neural network outputs a strong left signal in a potential head-on collision where the intruder is passing to the right. Property φ_{10} checks that the neural network outputs clear of conflict if the intruder aircraft is far away and moving away.

Table 6 ACAS Xu benchmark open-loop properties

	ρ, ft	θ, rad	ψ, rad	v_{own}, ft/s	v_{int}, ft/s	Networks	Output
φ_1	≥55,948	×	×	≥1145	≤60	All	*COC* ≤1500
φ_2	≥55,948	×	×	≥1145	≤60	$N_{x\geq 2,y}$	*COC* not maximum
φ_3	∈ [1500, 1800]	∈ [−0.06, 0.06]	≥3.10	≥980	≥960	$\neq N_{1,y\geq 7}$	*COC* not minimum
φ_4	∈ [1500, 1800]	∈ [−0.06, 0.06]	0	≥1000	∈ [700, 800]	$\neq N_{1,y\geq 7}$	*COC* not minimum
φ_5	∈ [250, 500]	∈ [0.2, 0.4]	$\approx -\pi$	∈ [100, 400]	∈ [0, 400]	$N_{1,1}$	*SR* minimum
φ_6	∈ [12,000, 62,000]	∈ [0.7, π] ∨∈ [$-pi$, −0.7]	$\approx\pi$	∈ [100, 1200]	∈ [0, 1200]	$N_{1,1}$	*COC* minimum
φ_7	∈ [0, 60,760]	∈ [$-\pi$, π]	∈ [$-\pi$, π]	∈ [100, 1200]	∈ [0, 1200]	$N_{1,9}$	*SL*,*SR* minimum
φ_8	∈ [0, 60,60]	∈ [$-\pi$, -0.75π]	∈ [−0.1, 0.1]	∈ [600, 1200]	∈ [600, 1200]	$N_{2,9}$	*WL* ∨ *COC*
φ_9	∈ [2000, 7000]	∈ [−0.4, −0.14]	$\approx -\pi$	∈ [100, 150]	∈ [0, 140]	$N_{3,3}$	*SR* minimum
φ_{10}	∈ [36,000, 60,760]	∈ [0.7, π]	$\approx -\pi$	∈ [900, 1200]	∈ [600, 1200]	$N_{4,5}$	*COC* minimum

Acknowledgments

This material is based upon work supported by the U.S. Air Force Office of Scientific Research (AFOSR) under award numbers FA9550-19-1-0288, FA9550-18-1-0122, FA9550-22-1-0019, FA9550-21-1-0121, and FA9550-22-1-0450; the National Science Foundation (NSF) Established Program to Stimulate Competitive Research (EPSCoR) First Award under grant numbers FMitF 1918450, 1910017, and 2028001; the U.S. Defense Advanced Research Projects Agency (DARPA) Assured Autonomy program through contract number FA8750-18-C-0089; and the Office of Naval Research (ONR) under award number N00014-22-1-2156. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFOSR, DARPA, NSF, or ONR.

References

[1] Indurkhya, N., and Damerau, F. J., *Handbook of Natural Language Processing*, Vol. 2, CRC Press, Boca Raton, FL, 2010.

[2] Tuia, D., Volpi, M., Copa, L., Kanevski, M., and Munoz-Mari, J., "A Survey of Active Learning Algorithms for Supervised Remote Sensing Image Classification," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 5, No. 3, 2011, pp. 606–617. https://doi.org/10.1109/JSTSP.2011.2139193

[3] Han, J., Zhang, D., Cheng, G., Liu, N., and Xu, D., "Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection: A Survey," *IEEE Signal Processing Magazine*, Vol. 35, No. 1, 2018, pp. 84–100. https://doi.org/10.1109/MSP.2017.2749125

[4] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M., "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, Vol. 529, No. 7587, 2016, pp. 484–489. https://doi.org/10.1038/nature24270

[5] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., and Bolton, A., "Mastering the Game of Go Without Human Knowledge," *Nature*, Vol. 550, No. 7676, 2017, pp. 354–359.

[6] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., and Georgiev, P., "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning," *Nature*, Vol. 575, No. 7782, 2019, pp. 350–354. https://doi.org/10.1038/s41586-019-1724-z

[7] Julian, K. D., Lopez, J., Brush, J. S., Owen, M. P., and Kochenderfer, M. J., "Policy Compression for Aircraft Collision Avoidance Systems," *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Inst. of Electrical and Electronics Engineers, New York, 2016, pp. 1–10. https://doi.org/10.1109/DASC.2016.7778091

[8] Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., and Kochenderfer, M. J., "Algorithms for Verifying Deep Neural Networks," *Foundations and Trends in Optimization*, Vol. 4, Nos. 3–4, 2021, pp. 244–404. https://doi.org/10.1561/2400000035

[9] Xiang, W., Musau, P., Wild, A. A., Lopez, D. M., Hamilton, N., Yang, X., Rosenfeld, J. A., and Johnson, T. T., "Verification for Machine Learning, Autonomy, and Neural Networks Survey," Preprint, Submitted 3 Oct. 2018, https://arxiv.org/abs/1810.01989.

[10] Ivanov, R., Weimer, J., Alur, R., Pappas, G. J., and Lee, I., "Verisig: Verifying Safety Properties of Hybrid Systems with Neural Network Controllers," *Proceedings of the 22nd Association for Computing Machinery (ACM) International Conference on Hybrid Systems: Computation and Control*, Assoc. for Computing Machinery, New York, 2019, pp. 169–178. https://doi.org/10.1145/3302504.3311806

[11] Dutta, S., Chen, X., and Sankaranarayanan, S., "Reachability Analysis for Neural Feedback Systems Using Regressive Polynomial Rule Inference," *Proceedings of the 22nd Association for Computing Machinery (ACM) International Conference on Hybrid Systems: Computation and Control*, Assoc. for Computing Machinery, New York, 2019, pp. 157–168. https://doi.org/10.1145/3302504.3311807

[12] Sun, X., Khedr, H., and Shoukry, Y., "Formal Verification of Neural Network Controlled Autonomous Systems," *Proceedings of the 22nd Association for Computing Machinery (ACM) International Conference on Hybrid Systems: Computation and Control*, Assoc. for Computing Machinery, New York, 2019, p. 147–156. https://doi.org/10.1145/3302504.3311802

[13] Xiang, W., Manzanas Lopez, D., Musau, P., and Johnson, T. T., "Reachable Set Estimation and Verification for Neural Network Models of Nonlinear Dynamic Systems," *Safe, Autonomous and Intelligent Vehicles*, edited by H. Yu, X. Li, R. M. Murray, S. Ramesh, and C. J. Tomlin, Springer International, Cham, Switzerland, 2019, pp. 123–144. https://doi.org/10.1007/978-3-319-97301-2_7

[14] Xiang, W., Tran, H.-D., Rosenfeld, J. A., and Johnson, T. T., "Reachable Set Estimation and Verification for a Class of Piecewise Linear Systems with Neural Network Controllers," *2018 Annuak American Control Conference (ACC)*, Inst. of Electrical and Electronics Engineers, New York, 2018, pp. 1574–1579. https://doi.org/10.23919/ACC.2018.8431048

[15] Fan, J., Huang, C., Chen, X., Li, W., and Zhu, Q., "ReachNN*: A Tool for Reachability Analysis of Neural-Network Controlled Systems," *Automated Technology for Verification and Analysis*, edited by D. V. Hung, and O. Sokolsky, Springer International, Cham, Switzerland, 2020, pp. 537–542. https://doi.org/10.1007/978-3-030-59152-6_30

[16] Akintunde, M. E., Botoeva, E., Kouvaros, P., and Lomuscio, A., "Formal Verification of Neural Agents in Non-Deterministic Environments," *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, Assoc. for Computing Machinery, New York, 2020, pp. 25–33. https://doi.org/10.1007/s10458-021-09529-3

[17] Akintunde, M. E., Botoeva, E., Kouvaros, P., and Lomuscio, A., "Verifying Strategic Abilities of Neural-symbolic Multi-agent Systems," *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, edited by D. Calvanese, E. Erdem, and M. Thielscher, IJCAI Press, 2020, pp. 22–32. https://doi.org/10.24963/kr.2020/3

[18] Tran, H.-D., Cai, F., Diego, M. L., Musau, P., Johnson, T. T., and Koutsoukos, X., "Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control," *ACM Transactions on Embedded Computer Systems*, Vol. 18, Vol. 18, No. 5, Oct. 2019, Paper 105. https://doi.org/10.1145/3358230

[19] Tran, H.-D., Yang, X., Manzanas Lopez, D., Musau, P., Nguyen, L. V., Xiang, W., Bak, S., and Johnson, T. T., "NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems," *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I*, Springer-Verlag, Berlin, 2020, pp. 3–17. https://doi.org/10.1007/978-3-030-53288-8_1

[20] Manzanas Lopez, D., Musau, P., Hamilton, N., Tran, H.-D., and Jonhson, T. T., "Case Study: Safety Verification of an Unmanned Underwater Vehicle," *2020 Institute of Electrical and Electronics Engineers (IEEE) Security and Privacy Workshops (SPW)*, Inst. of Electrical and Electronics Engineers, New York, 2020, pp. 189–195. https://doi.org/10.1109/SPW50608.2020.00047

[21] Irfan, A., Julian, K. D., Wu, H., Barrett, C., Kochenderfer, M. J., Meng, B., and Lopez, J., "Towards Verification of Neural Networks for Small Unmanned Aircraft Collision Avoidance," *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, Inst. of Electrical and Electronics Engineers, New York, 2020, pp. 1–10. https://doi.org/10.1109/DASC50938.2020.9256616

[22] Clavière, A., Asselin, E., Garion, C., and Pagetti, C., "Safety Verification of Neural Network Controlled Systems," *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Inst. of Electrical and Electronics Engineers, New York, 2021, pp. 47–54. https://doi.org/10.1109/DSN-W52860.2021.00019

[23] Bak, S., Liu, C., and Johnson, T. T., "The Second International Verification of Neural Networks Competition (VNN-COMP 2021): Summary and Results," Preprint, submitted 31 Aug. 2021, https://arxiv.org/abs/2109.00498.

[24] Johnson, T. T., Lopez, D. M., Benet, L., Forets, M., Guadalupe, S., Schilling, C., Ivanov, R., Carpenter, T. J., Weimer, J., and Lee, I., "ARCH-COMP21 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants," *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, edited by G. Frehse, and M. Althoff, Vol. 80, EPiC Series in Computing, EPiC, July 2021, pp. 90–119. https://doi.org/10.29007/kfk9

[25] Ivanov, R., Carpenter, T. J., Weimer, J., Alur, R., Pappas, G. J., and Lee, I., "Case Study: Verifying the Safety of an Autonomous Racing Car with a Neural Network Controller," *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control (HSCC '20)*, Assoc. for Computing Machinery, New York, 2020. https://doi.org/10.1145/3365365.3382216

[26] Julian, K. D., and Kochenderfer, M. J., "A Reachability Method for Verifying Dynamical Systems with Deep Neural Network Controllers," Preprint, Submitted 1 March 2019, https://arxiv.org/abs/1903.00520.

[27] Julian, K. D., Sharma, S., Jeannin, J.-B., and Kochenderfer, M. J., "Verifying Aircraft Collision Avoidance Neural Networks Through Linear Approximations of Safe Regions," *AIAA Spring Symposium*, AIAA, Reston, VA, 2019. https://doi.org/10.48550/ARXIV.1903.00762

[28] Julian, K. D., and Kochenderfer, M. J., "Guaranteeing Safety for Neural Network-Based Aircraft Collision Avoidance Systems," *IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, Inst. of Electrical and Electronics Engineers, New York, 2019, pp. 1–10. https://doi.org/10.1109/DASC43569.2019.9081748.

[29] Julian, K. D., and Kochenderfer, M. J., "Reachability Analysis for Neural Network Aircraft Collision Avoidance Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 6, 2021, pp. 1132–1142. https://doi.org/10.2514/1.G005233

[30] Alvarez, L. E., Jessen, I., Owen, M. P., Silbermann, J., and Wood, P., "ACAS sXu: Robust Decentralized Detect and Avoid for Small Unmanned Aircraft Systems," *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, Inst. of Electrical and Electronics Engineers, New York, 2019, pp. 1–9. https://doi.org/10.1109/DASC43569.2019.9081631

[31] Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S., "Formal Security Analysis of Neural Networks Using Symbolic Intervals," *Proceedings of the 27th USENIX Conference on Security Symposium*, USENIX Assoc., New York, 2018, p. 1599–1614.

[32] Bak, S., "nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement," *NASA Formal Methods Symposium*, Springer, Berlin, 2021, pp. 19–36.

[33] Olson, W. A., "Airborne Collision Avoidance System x," Lincoln Lab., Massachusetts Inst. of Technology, 2015.

[34] Kochenderfer, M. J., Amato, C., Chowdhary, G., How, J. P., Reynolds, H. J. D., Thornton, J. R., Torres-Carrasquillo, P. A., Ure, N. K., and Vian, J., "Decision Making Under Uncertainty: Theory and Application," *Optimized Airborne Collision Avoidance*, MIT Press, Cambridge, MA, 2015, pp. 249–276.

[35] Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J., "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks," *International Conference on Computer Aided Verification*, Springer, Berlin, 2017, pp. 97–117. https://doi.org/10.1007/978-3-319-63387-9_5

[36] Marston, M., and Baca, G., "ACAS-Xu Initial Self-Separation Flight Tests," NASA, March 2015, https://ntrs.nasa.gov/api/citations/20150008347/downloads/20150008347.pdf.

[37] Holland, J. E., Kochenderfer, M. J., and Olson, W. A., "Optimizing the Next Generation Collision Avoidance System for Safe, Suitable, and Acceptable Operational Performance," *Air Traffic Control Quarterly*, Vol. 21, No. 3, 2013, pp. 275–297. https://doi.org/10.2514/atcq.21.3.275

[38] Kochenderfer, M. J., and Chryssanthacopoulos, J., "Robust Airborne Collision Avoidance Through Dynamic Programming," Lincoln Lab., Massachusetts Inst. of Technology Project Report ATC-371, Vol. 130, Cambridge, MA, 2011.

[39] Julian, K. D., Kochenderfer, M. J., and Owen, M. P., "Deep Neural Network Compression for Aircraft Collision Avoidance Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 3, 2019, pp. 598–608. https://doi.org/10.2514/1.G003724

[40] Julier, S., and Uhlmann, J., "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, Vol. 92, No. 3, 2004, pp. 401–422. https://doi.org/10.1109/JPROC.2003.823141

[41] Kochenderfer, M. J., and Monath, N., "Compression of Optimal Value Functions for Markov Decision Processes," *2013 Data Compression Conference*, IEEE Publ., Piscataway, NJ, 2013, p. 501. https://doi.org/10.1109/DCC.2013.81

[42] Raychaudhuri, S., "Introduction to Monte Carlo Simulation," *2008 Winter Simulation Conference*, Inst. of Electrical and Electronics Engineers, New York, 2008, pp. 91–100. https://doi.org/10.1109/WSC.2008.4736059

[43] Antony, J., *Design of Experiments for Engineers and Scientists*, Elsevier, New York, 2014.

[44] Viana, F. A., "A Tutorial on Latin Hypercube Design of Experiments," *Quality and Reliability Engineering International*, Vol. 32, No. 5, 2016, pp. 1975–1985. https://doi.org/10.1002/qre.1924

[45] "VFR Cruising Altitude or Flight Level," *General Operating and Flight Rules*, Title 14, Code of Federal Regulations 91.159, 2003.

[46] Hainry, E., "Reachability in Linear Dynamical Systems," *Logic and Theory of Algorithms*, edited by A. Beckmann, C. Dimitracopoulos, and B. Löwe, Springer, Berlin, 2008, pp. 241–250. https://doi.org/10.1007/978-3-540-69407-6_28

[47] Ruan, W., Huang, X., and Kwiatkowska, M., "Reachability Analysis of Deep Neural Networks with Provable Guarantees," *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, edited by J. Lang, International Joint Conferences on Artificial Intelligence, July 2018, pp. 2651–2659. https://doi.org/10.24963/ijcai.2018/368

[48] Bak, S., and Duggirala, P. S., "Simulation-Equivalent Reachability of Large Linear Systems with Inputs," *Computer Aided Verification*, edited by R. Majumdar, and V. Kunčak, Springer International, Cham, Switzerland, 2017, pp. 401–420. https://doi.org/10.1007/978-3-319-63387-9_20

[49] Tran, H.-D., Manzanas Lopez, D., Musau, P., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T., "Star-Based Reachability Analysis of Deep Neural Networks," *Formal Methods—The Next 30 Years: Third World Congress, FM 2019*, Springer-Verlag, Berlin, 2019, p. 670–686. https://doi.org/10.1007/978-3-030-30942-8_39

[50] Althoff, M., "An Introduction to CORA 2015," *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, Vol. 34, EPiC Series in Computing, edited by G. Frehse, and M. Althoff, EPiC, Dec. 2015, pp. 120–151. https://doi.org/10.29007/zbkv

[51] Bak, S., Tran, H.-D., Hobbs, K., and Johnson, T. T., "Improved Geometric Path Enumeration for Verifying ReLU Neural Networks," *32nd International Conference on Computer-Aided Verification (CAV)*, 2020.

H. Idris
Associate Editor