Guaranteed Safe Path and Trajectory Tracking via Reachability Analysis Using Differential Inequalities

Xuejiao Yang[1], Bowen Mu[1], Dillard Robertson[1] and Joseph Scott[1*]

[1]School of Chemical and Biomolecular Engineering, Georgia Institute of Technology, North Ave NW, Atlanta, 30332, GA, USA.

*Corresponding author(s). E-mail(s): joseph.scott@chbe.gatech.edu;
Contributing authors: xyang341@gatech.edu; bowenmu@gatech.edu;
dillardrobertson@gatech.edu;

Abstract

In many automated motion planning systems, vehicles are tasked with tracking a reference path or trajectory that is safe by design. However, due to various uncertainties, real vehicles may deviate from such references, potentially leading to collisions. This paper presents rigorous reachable set bounding methods for rapidly enclosing the set of possible deviations under uncertainty, which is critical information for online safety verification. The proposed approach applies recent advances in the theory of differential inequalities that exploit redundant model equations to achieve sharp bounds using only simple interval calculations. These methods have been shown to produce very sharp bounds at low cost for nonlinear systems in other application domains, but they rely on problem-specific insights to identify appropriate redundant equations, which makes them difficult to generalize and automate. Here, we demonstrate the application of these methods to tracking problems for the first time using three representative case studies. We find that defining redundant equations in terms of Lyapunov-like functions is particularly effective. The results show that this technique can produce effective bounds with computational times that are orders of magnitude less than the planned time horizon, making this a promising approach for online safety verification. This performance, however, comes at the cost of low generalizability, specifically due to the need for problem-specific insights and advantageous problem structure, such as the existence of appropriate Lyapunov-like functions.

Keywords: Reachability analysis, safety verification, collision avoidance, autonomous driving
Categories (2), (3), and (5)

1 Introduction

This paper presents rigorous reachable set bounding methods for verifying the safety of automated vehicles tracking reference paths or trajectories under uncertainty. Tracking is important in automated driving systems for road vehicles, motion planning for autonomous robots, etc. [1]. However, the paths and trajectories computed by such systems, which are safe by design, are not followed exactly due to uncertainties in the vehicle and its environment, which can lead to collisions or other constraint violations. Thus, methods for ensuring safety in real time are essential.

The literature on vehicle safety verification addresses several distinct problems based on how the control inputs are handled. One class of methods assumes the inputs obey a probability distribution modeling the action of human drivers and computes the likelihood of a collision [2, 3]. These methods are

designed to generate warnings for human drivers, not for use in automated systems. The computation of safe inputs is discussed in [2], but safety is only ensured for nominal vehicle dynamics with no uncertainty.

A second class of methods treats the inputs as degrees of freedom and aims to compute inputs that guarantee safe trajectories [4, 5]. This is typically decomposed into the computation of a nominally safe reference path or trajectory followed by synthesis of a robustly safe tracking controller. The reference can be computed by established graph search, optimization, or sampling approaches [1]. In contrast, synthesizing a robustly safe feedback law is a major challenge. This can be done by solving appropriate Hamilton-Jacobi-Isaacs equations, but this is often prohibitive [5, 6]. An alternative is to use robust or tube-based model predictive control [4, 7]. However, this requires either the solution of a robust open-loop optimal control problem at each sampling time or the *a priori* construction of a robustly invariant tube, both of which are prohibitive for nonlinear models. Another approach is to compute a robust control barrier function (R-CBF) [8, 9]. Combined with an appropriate Lyapunov function, an R-CBF can be used to formulate an optimization-based feedback law that ensures safety. However, the required optimization only takes a tractable form for systems with affine dependence on both controls and uncertainties. Moreover, there are no existing methods for computing R-CBFs for general nonlinear systems.

A third class of methods considers the simpler problem of verifying safety for a fixed control input or feedback law [10–14]. This addresses a critical subtask that can be used within larger algorithms for synthesizing safe controllers or motion plans. Some methods compute the probability of safety violations using sampling or stochastic reachable sets [10, 11]. These methods cannot make rigorous safety guarantees. Moreover, sampling is prohibitive with more than a few uncertain quantities. Alternatively, some methods aim to provide rigorous safety guarantees subject to bounded uncertainties using reachability analysis [12–14]. However, computing accurate reachable set enclosures for nonlinear systems is a significant challenge. Most approaches resort to linearized models [4, 12, 13], which does not ensure safety. A rigorous method for nonlinear models is given in [14]. For the example therein, the method verifies the safety of a trajectory $\sim 2\times$ faster than the real vehicle traverses it. While this is promising, there is still a need for more efficient methods. Autonomous vehicles often update their trajectories every few milliseconds [15], so verification on a similar time-scale is desirable.

The methods in this paper fall within the third class above. Given a model, a fixed reference path or trajectory, and a fixed tracking controller, we are interested in computing a rigorous enclosure of the reachable set of the closed-loop system under uncertainty. Effective tracking controllers are available and widely used for many vehicle models, so a method for verifying their safety in real-time is likely to be useful within practical iterative approaches for the harder problem of safe controller synthesis.

Many methods are available for bounding the reachable sets of continuous-time nonlinear systems, but they often exhibit an unworkable compromise between accuracy and efficiency. The method in [16], which was used for safety verification in [14], propagates zonotopic enclosures over discrete time steps using a conservative linearization technique. This is effective in many cases, but the linearization error bound can become conservative. Moreover, high-order zonotopes and/or partitioning may be required to achieve high accuracy, which can lead to high cost. Another class of methods propagates enclosures over discrete time steps by constructing Taylor expansions of the states with respect to time and then bounding the coefficients and remainder term [17]. While early methods used interval arithmetic, contemporary methods achieve much higher accuracy using so-called Taylor model arithmetic [18, 19]. However, high accuracy may require high-order Taylor models, which also comes at high cost. Notably, these costs can be effectively moved offline in some applications by pre-computing reachable sets for a library of motion primitives [20, 21].

A final class of reachability methods is based on the theory of differential inequalities (DI). These methods compute enclosures as the solutions of an auxiliary system of ordinary differential equations (ODEs). The standard DI method computes interval enclosures using an auxiliary system constructed via interval arithmetic [22]. This is very efficient, which is attractive for online verification, but usually yields very conservative bounds. More recent methods replace intervals with polytopes [23], Taylor models [24], or mean-value enclosures [25]. These produce much tighter bounds, but are less efficient. Another category of DI methods aims to use model redundancy to mitigate the conservatism of the standard DI method while largely retaining its speed. These approaches identify constraints that are redundant

with the dynamics (a.k.a. invariants), such as conservation laws, non-negativity of certain states, etc., and exploit them to tighten the bounds continuously as they are propagated forward in time [25–27]. Importantly, this method can be applied to systems that do not satisfy any known invariants by *manufacturing* invariants [27]. This involves embedding the system within a higher-dimensional system that obeys invariants by design (see Section 2). Redundancy-based DI methods have proven to be remarkably effective for many case studies, including systems that naturally satisfy invariants and many that do not [25–27]. However, this approach requires significant problem insight to apply effectively. To date, successful strategies have only been clearly demonstrated for chemical engineering models comprised of dynamic mass and energy balances.

This paper demonstrates the use of redundancy-based DI for vehicle tracking problems for the first time. This is challenging for three primary reasons. First, the models we consider do not naturally obey any invariants. Moreover, compared to mass and energy balance models, it is much more difficult to identify effective manufactured invariants. Second, the presence of a feedback law causes an interval dependency problem that leads to very conservative bounds if not addressed (see Section 2). Finally, the models of interest involve several functions whose interval evaluations are either not well-defined or violate Lipschitz conditions required by DI methods.

To address these issues, we first develop new Lipschitz interval evaluations for several functions. Next, we demonstrate the application of redundancy-based DI for three case studies in detail. In all cases, we address the feedback dependency problem through appropriate coordinate transformations. We then show that manufactured invariants defined in terms of Lyapunov-like functions are highly effective at reducing conservatism. In all cases, we obtain reachability bounds that are much more accurate than standard DI or state-of-the-art zonotope methods with computational times that are orders of magnitude less than the planned time horizons. Although the bounds leave significant room for improvement, they appear accurate enough to support many online safety verification tasks, particularly outdoors. This performance, however, comes at the cost of low generalizability due to the need for problem-specific insights and advantageous problem structure (i.e., the existence of appropriate Lyapunov-like functions).

1.1 Problem Statement

Let $I = [t_0, t_f]$ be a time horizon of interest, let $\mathbf{f}_0 : D_{f0} \subset \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, let $\boldsymbol{\kappa} : D_{\boldsymbol{\kappa}} \subset \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_u}$, and consider the following closed-loop system with input $\mathbf{u}$, disturbance $\mathbf{w}$, and state $\mathbf{x}$:

$$\dot{\mathbf{x}}(t) = \mathbf{f}_0(t, \mathbf{x}(t), \mathbf{w}(t), \mathbf{u}(t)), \text{ a.e. } t \in I, \tag{1a}$$
$$\mathbf{u}(t) = \boldsymbol{\kappa}(t, \mathbf{x}(t), \mathbf{w}(t)), \tag{1b}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{1c}$$

We aim to compute reachability bounds for (1) under a given tracking controller $\boldsymbol{\kappa}$. We assume all states can be measured and allow $\boldsymbol{\kappa}$ to depend on $\mathbf{w}(t)$ in case some disturbances are measured. The dependence of $\boldsymbol{\kappa}$ on the reference path or trajectory is suppressed for brevity. Defining $\mathbf{f}(t, \mathbf{z}, \mathbf{v}) \equiv \mathbf{f}_0(t, \mathbf{z}, \mathbf{v}, \boldsymbol{\kappa}(t, \mathbf{z}, \mathbf{v}))$, (1) is equivalent to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{w}(t)), \text{ a.e. } t \in I, \tag{2a}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{2b}$$

Denote the space of Lebesgue integrable functions $y : I \to \mathbb{R}$ by $L^1(I)$. Let $W \subset \mathbb{R}^{n_w}$ be a compact interval and define the set of admissible disturbances as

$$\mathscr{W} \equiv \{\mathbf{w} \in (L^1(I))^{n_w} : \mathbf{w}(t) \in W \text{ for a.e. } t \in I\}. \tag{3}$$

Let $X_0 \subset \mathbb{R}^{n_x}$ be a compact interval of admissible initial conditions. Let $\mathscr{AC}(I, \mathbb{R}^n)$ be the space of absolutely continuous functions from I into $\mathbb{R}^n$. We assume (2) has a unique solution $\mathbf{x} \in \mathscr{AC}(I, \mathbb{R}^{n_x})$ for every $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathscr{W}$ and denote it by $\mathbf{x}(\cdot; \mathbf{x}_0, \mathbf{w})$ when explicit dependence on $(\mathbf{x}_0, \mathbf{w})$ is necessary for clarity.

Definition 1 The *reachable set* of (2) is defined by

$$\mathrm{Re}(t) \equiv \{\mathbf{x}(t; \mathbf{x}_0, \mathbf{w}) : (\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathscr{W}\},$$

for every $t \in I$. Moreover, functions $\mathbf{x}^L, \mathbf{x}^U : I \to \mathbb{R}^{n_x}$ are called *state bounds* for (2) if $\mathrm{Re}(t) \subset [\mathbf{x}^L(t), \mathbf{x}^U(t)]$, $\forall t \in I$.

Our objective is to compute state bounds for path and trajectory tracking problems of the form (1) with sufficient accuracy and efficiency for rigorous motion planning and real-time safety verification tasks.

Remark 1 Our formulation considers uncertainty in the form of bounded initial conditions $\mathbf{x}_0 \in X_0$ and time-varying disturbances $\mathbf{w}(t) \in \mathscr{W}$. In principle, bounded measurement

errors could be accommodated by appending them to $\mathbf{w}(t)$ (DI methods apply to the general ODEs (2), so there is no restriction on how $\mathbf{w}(t)$ affects the model). However, we assumed perfect measurements because some problem-specific reformulations and simplifications used in the case studies require the controller to act on the true state. This is an important limitation to address in future work. We also do not explicitly account for potential mismatch between the real vehicle dynamics and the idealized model used for verification. Approaches for addressing this using conformant models can be found in [20, 28]. Finally, the important issue of handling uncertainty in the the locations of obstacles during collision checking is not addressed since we focus solely on the problem of bounding the possible vehicle states.

1.2 Notation

For $\mathbf{z}^L, \mathbf{z}^U \in \mathbb{R}^n$, let $Z = [\mathbf{z}^L, \mathbf{z}^U]$ denote the interval $\{\mathbf{z} \in \mathbb{R}^n : \mathbf{z}^L \leq \mathbf{z} \leq \mathbf{z}^U\}$. For $D \subset \mathbb{R}^n$, let $\mathbb{I}D$ denote the set of all intervals Z such that $Z \subset D$. For $\mathbf{h} : D \subset \mathbb{R}^n \to \mathbb{R}^m$, an interval function $H : D_H \subset \mathbb{I}D \to \mathbb{IR}^m$ is an *inclusion function* for $\mathbf{h}$ on D_H if $H(X) \supset \{\mathbf{h}(\mathbf{x}) : \mathbf{x} \in X\}$ for every $X \in D_H$. If $\mathbf{h}$ can be written as a finite composition of basic arithmetic operations such as addition, multiplication, and intrinsic univariate functions ($-x$, x^n, e^x, etc.), then an inclusion function can be readily computed using interval arithmetic [29].

The space of intervals $\mathbb{IR}^n$ is a metric space under the Hausdorff metric $d_H(Z_1, Z_2) = \max\{\|\mathbf{z}_1^L - \mathbf{z}_2^L\|_\infty, \|\mathbf{z}_1^U - \mathbf{z}_2^U\|_\infty\}$ [29]. Thus, following standard metric space definitions, the open ball of radius $\varepsilon > 0$ centered at $X \in \mathbb{IR}^n$ is defined by $B_\varepsilon(X) \equiv \{Z \in \mathbb{IR}^n : d_H(X, Z) < \varepsilon\}$. A set $\mathscr{D} \subset \mathbb{IR}$ is open if for every $Z \in \mathscr{D}$ there exists a $\varepsilon > 0$ such that $B_\varepsilon(Z) \subset \mathscr{D}$. Moreover, a function $F : D_F \subset \mathbb{IR}^n \to \mathbb{IR}^m$ is locally Lipschitz continuous on D_F if for every $Z \in D_F$, there exist constants $M, \varepsilon > 0$ such that $d_H(F(X), F(Y)) \leq M d_H(X, Y)$ for every $X, Y \in B_\varepsilon(Z) \cap D_F$.

2 Differential Inequalities

This section introduces the differential inequalities (DI) methods used in this paper. Let $F : D_F \subset \mathbb{I}D_f \to \mathbb{IR}$ be an inclusion function for $\mathbf{f}$ in (2) and denote $[\mathbf{f}^L, \mathbf{f}^U] = F$. The following notation for selecting an individual face of an interval is required.

Definition 2 For every $i \in \{1, \ldots, n_x\}$, define the *face selection operators* $\beta_i^L, \beta_i^U : \mathbb{IR}^{n_x} \to \mathbb{IR}^{n_x}$ by

$$\beta_i^L([\mathbf{z}^L, \mathbf{z}^U]) \equiv \{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U] : z_i = z_i^L\},$$
$$\beta_i^U([\mathbf{z}^L, \mathbf{z}^U]) \equiv \{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U] : z_i = z_i^U\}.$$

The standard DI method originally proposed in [22] computes state bounds $X(t) \equiv [\mathbf{x}^L(t), \mathbf{x}^U(t)]$ as the solutions of the following system of ODEs:

$$\begin{aligned} \dot{x}_i^L(t) &= f_i^L([t,t], \beta_i^L(X(t)), W), \\ \dot{x}_i^U(t) &= f_i^U([t,t], \beta_i^U(X(t)), W), \\ X(t_0) &= X_0. \end{aligned} \tag{4}$$

Briefly, the rationale is as follows. At t_0, we have $\mathbf{x}(t_0) \in [\mathbf{x}^L(t_0), \mathbf{x}^U(t_0)]$. In order for, e.g., $x_i^L(t)$ to remain lower than $x_i(t)$ for $t > t_0$, it is sufficient to require that x_i^L decreases faster than any trajectory x_i with $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathscr{W}$; i.e., $\dot{x}_i^L(t) \leq \dot{x}_i(t)$. However, more careful analysis shows that it is only necessary to have $\dot{x}_i^L(t) \leq \dot{x}_i(t)$ at those $t \in I$ for which $x_i(t) = x_i^L(t)$. This weaker requirement is achieved by bounding f_i over $\beta_i^L(X(t))$ rather than $X(t)$ in (4).

Standard DI is very efficient but often very conservative [26]. One key reason is the *dependency problem*, which refers to the fact that interval arithmetic treats multiple instances of a variable as independent. For example, x_2 appears twice in the ODE $\dot{x}_1 = f_1(\mathbf{x}) = -ax_1x_2 + bx_2x_3$. If F_1 is computed using interval arithmetic, then it will bound the range of f_1 assuming these two instances of x_2 are independent, leading to overestimation. Dependency is not an inherent weakness of DI, but rather a weakness of the kind of inclusion function normally used in DI. Indeed, it can be mitigated using more sophisticated inclusion functions, but this is less efficient. The problem above can also be eliminated by rewriting f_1 as $f_1(\mathbf{x}) = (-ax_1 + bx_3)x_2$. DI can produce substantially different bounds using different expressions for $\mathbf{f}$, even though they are equivalent in real arithmetic. This is important for getting good results from interval methods, but good rearrangements are not always possible.

A more subtle source of conservatism in DI is the *historical dependency problem* [27]. This refers to the fact that even distinct variables such as x_2 and x_3 in the example above are not independent after t_0. Thus, treating these variables as independent when bounding $\mathbf{f}$ also leads to overestimation. Historical dependency is a weakness of DI itself and cannot be resolved by refactoring $\mathbf{f}$ or using better inclusion functions. It can be mitigated by propagating non-interval enclosures because they can capture some of the dependence between states. However, such methods lose much of the speed that is attractive in interval methods.

To address this, several papers have developed efficient interval DI methods that compute tighter bounds

by exploiting *redundant model equations* [25–27]. This refers to any relationships involving the states of a system that are known to be satisfied by all solutions. Common examples include non-negativity of certain states, conservation of mass, energy, or chemical species [26], the unit norm of rotation quaternions in some vehicle models [30], and other invariants. Such relationships are useful because they provide information about the historical dependency between states. Redundancy-based DI methods use these to limit the range of inputs over which f_i must be bounded in (4).

Below, we assume redundant information is available in the form of an *a priori* enclosure G and present the main details of the method in [25]. Subsequently, we discuss how this approach can be applied to systems with no known G using *manufactured invariants*.

Assumption 1 An *a priori* enclosure $G \subset \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w}$ is known such that every solution of (2) with $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathscr{W}$ satisfies $(t, \mathbf{x}(t), \mathbf{w}(t)) \in G$ for all $t \in I$.

The method in [25] makes use of G through a special kind of inclusion function called $\mathscr{R}$. Given t and intervals Z and V, $\mathscr{R}$ computes an interval enclosure of $\mathbf{f}(t, \mathbf{z}, \mathbf{v})$ for all $(\mathbf{z}, \mathbf{v}) \in Z \times V$ such that $(t, \mathbf{z}, \mathbf{v}) \in G$. This is different from F used in standard DI, which computes an enclosure of $\mathbf{f}(t, \mathbf{z}, \mathbf{v})$ for all $(\mathbf{z}, \mathbf{v}) \in Z \times V$. The function $\mathscr{R}$ also needs to satisfy several technical conditions detailed in Assumption 2.

Assumption 2 Let $\mathscr{R} : D_{\mathscr{R}} \subset \mathbb{R} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w} \to \mathbb{IR}^{n_x}$ be an interval function satisfying:

1. For any $(t, Z, V) \in D_{\mathscr{R}}$, the set $\{t\} \times Z \times V$ is contained in the domain of $\mathbf{f}$, D_f, and
$$\mathscr{R}(t, Z, V) \supset \left\{ \mathbf{f}(t, \mathbf{z}, \mathbf{v}) : \begin{array}{l} (\mathbf{z}, \mathbf{v}) \in Z \times V, \\ (t, \mathbf{z}, \mathbf{v}) \in G \end{array} \right\}. \quad (5)$$
2. $D_{\mathscr{R}}$ is open with respect to t and Z. Specifically, for every $(\hat{t}, \hat{Z}, \hat{V}) \in D_{\mathscr{R}}$, there exists $\varepsilon > 0$ such that $(t, Z, \hat{V}) \in D_{\mathscr{R}}$ for every $t \in B_\varepsilon(\hat{t})$ and $Z \in B_\varepsilon(\hat{Z})$.
3. $\mathscr{R}$ is locally Lipschitz continuous with respect to Z, uniformly with respect to t. Specifically, for any $(\hat{t}, \hat{Z}, \hat{V}) \in \mathscr{R}$, there exists $\varepsilon, L > 0$ such that
$$d_H(\mathscr{R}(t, Z, \hat{V}), \mathscr{R}(t, \bar{Z}, \hat{V})) \leq L d_H(Z, \bar{Z}), \quad (6)$$
for every $t \in B_\varepsilon(\hat{t})$ and $Z, \bar{Z} \in B_\varepsilon(\hat{Z})$.

Moreover, let $\mathscr{R}_i = [\mathscr{R}_i^L, \mathscr{R}_i^U]$ be the i^{th} component of $\mathscr{R}$.

Given any such $\mathscr{R}$, state bounds for (2) can be computed using the following corollary from [25].

Corollary 1 *Suppose that* $\mathbf{x}^L, \mathbf{x}^U \in \mathscr{AC}(I, \mathbb{R}^{n_x})$ *are solutions of the following system of ODEs with* $i \in \{1, \ldots, n_x\}$ *and* $X(t) \equiv [\mathbf{x}^L(t), \mathbf{x}^U(t)]$*:*
$$\dot{x}_i^L(t) = \mathscr{R}_i^L(t, \beta_i^L(X(t)), W), \quad (7)$$
$$\dot{x}_i^U(t) = \mathscr{R}_i^U(t, \beta_i^U(X(t)), W),$$
$$X(t_0) = X_0. \quad (8)$$
Then, for every $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathscr{W}$*, the solution* $\mathbf{x}(\cdot; \mathbf{x}_0, \mathbf{w}) \in \mathscr{AC}(I, \mathbb{R}^{n_x})$ *of* (2) *satisfies* $\mathbf{x}(t; \mathbf{x}_0, \mathbf{w}) \in X(t)$ *for all* $t \in I$*.*

Remark 2 Corollary 1 is a simplified version of the result in [25]. In [25], G is a subset of $\mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_x}$ with the assumption that $(t, \mathbf{x}(t), \mathbf{w}(t), \dot{\mathbf{x}}(t)) \in G$. The inclusion property of $\mathscr{R}$ is generalized accordingly. Since none of the redundant relationships defining G in our case studies depend on $\dot{\mathbf{x}}(t)$, this dependence is ommitted for simplicity.

In [25], $\mathscr{R}$ is computed in two steps. Given inputs $(t, Z, V) \in D_{\mathscr{R}}$, Z and V are first refined by eliminating regions that violate $(t, \mathbf{z}, \mathbf{v}) \in G$. This is done using a variant of the interval Krawczyk method [29] called the κ-operator and yields intervals $Z^\dagger$ and $V^\dagger$ satisfying $Z^\dagger \times V^\dagger \supset (Z \times V) \cap \{(\mathbf{z}, \mathbf{v}) : (t, \mathbf{z}, \mathbf{v}) \in G\}$. Next, $Z^\dagger$ and $V^\dagger$ are used to evaluate a standard inclusion function for $\mathbf{f}$; i.e., $\mathscr{R}(t, Z, V) = F([t, t], Z^\dagger, V^\dagger)$.

This provides much tighter bounds than standard DI in many cases, but only applies to systems with known G. To extend it to general systems, Shen and Scott [27] introduced *manufactured invariants*. Assume that $\mathbf{w}(t) = (\mathbf{d}(t), \mathbf{p})$, where $\mathbf{d}(t) \in \mathbb{R}^{n_d}$ is a time-varying disturbance and $\mathbf{p} \in \mathbb{R}^{n_p}$ is a vector of time-invariant uncertain parameters. Shen and Scott's procedure begins by choosing a smooth function $\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_z}$ and defining new state variables by $\mathbf{z}(t; \mathbf{x}_0, \mathbf{w}) \equiv \phi(\mathbf{x}(t; \mathbf{x}_0, \mathbf{w}), \mathbf{p})$. Next, the new states are differentiated to form the augmented system
$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{w}(t)), \quad (9)$$
$$\dot{\mathbf{z}}(t) = \frac{\partial \phi}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{p}) \mathbf{f}(t, \mathbf{x}(t), \mathbf{w}(t)),$$
$$\mathbf{x}(t_0) = \mathbf{x}_0,$$
$$\mathbf{z}(t_0) = \phi(\mathbf{x}_0, \mathbf{p}).$$

If $(\mathbf{x}, \mathbf{z})$ is a solution of (9), then $\mathbf{x}$ is a solution of (2). Therefore, to compute state bounds for (2), it suffices to do so for (9). But, by design, all solutions of (9) satisfy the *manufactured invariants* $\mathbf{z}(t) -$

$\phi(\mathbf{x}(t),\mathbf{p}) = \mathbf{0}$. Thus, Corollary 1 can be applied with $G \equiv \{(t,(\mathbf{x},\mathbf{z}),(\mathbf{d},\mathbf{p})) : \mathbf{z} - \phi(\mathbf{x},\mathbf{p}) = \mathbf{0}\}$. Importantly, ϕ cannot depend on $\mathbf{d}(t)$ since then (9) would involve the derivative of $\mathbf{d}(t)$, which may not be bounded.

This technique has been shown to result in much tighter bounds than standard DI for many problems with no known G. However, this requires careful choice of ϕ. The aim is to choose ϕ such that $\frac{\partial \phi}{\partial \mathbf{x}}\mathbf{f}$ reduces to an expression that does not suffer much from dependency problems. For example, for $\dot{x}_1 = -x_1 - r(\mathbf{w},\mathbf{x})$ and $\dot{x}_2 = -2x_2 + r(\mathbf{w},\mathbf{x})$ with some nonlinear and uncertain term r, a good choice is $z = x_1 + x_2$, which leads to $\dot{z} = -x_1 - 2x_2 = -z - x_2$. When the bounding ODEs (7) are solved for the augmented system, simplifications of this sort can cause the bounds on $\mathbf{z}$ to accumulate conservatism slowly or not at all. In turn, this enables the bounds on $\mathbf{x}$ to be effectively refined using the manufactured invariant during the evaluation of $\mathscr{R}$. See [27] for a more detailed explanation. Although the example above is contrived, similar simplifications can be achieved by simple affine ϕ in many practical examples, and the improvements in bound accuracy are stark [27]. However, most of these examples are drawn from (bio)chemical engineering and the models share some advantageous features. Thus, while this technique is broadly applicable in principle, effective strategies for choosing ϕ have only been demonstrated for a limited class of models.

In this article, we apply redundancy-based DI to path and trajectory tracking problems, which raises several new challenges. First, the systems of interest are closed-loop, so that $\mathbf{f}(t,\mathbf{z},\mathbf{v}) = \mathbf{f}_0(t,\mathbf{z},\mathbf{v},\kappa(t,\mathbf{z},\mathbf{v}))$. This structure ensures that there is a significant interval dependency problem because both $\mathbf{z}$ and $\mathbf{v}$ appear twice. If interval arithmetic is applied to $\mathbf{f}$ in this form, the result will certainly be conservative. Bounds on the range of $\mathbf{f}(t,\cdot,\cdot)$ over some interval $Z \times V$ computed in this way would include all values of $\mathbf{f}_0(t,\mathbf{z},\mathbf{v},\mathbf{u})$ obtained by pairing any $(\mathbf{z},\mathbf{v}) \in Z \times V$ with any input $\mathbf{u} \in \kappa(t,Z,V)$, which completely undermines the desired action of the controller. Second, the systems we consider do not satisfy any known *a priori* enclosures. Thus, it is necessary to manufacture invariants. However, unlike the examples mentioned above, there are no obvious choices of ϕ that lead to desirable simplifications, so new strategies are needed. Finally, the systems we consider involve several functions whose interval evaluations are either not well-defined or violate the Lipschitz property required by Assumption 2, so new operations must be defined.

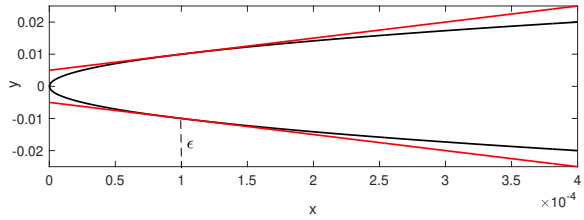

Fig. 1: Multi-valued square root $y = \pm\sqrt{x}$ (black) with lower and upper linearizations at $\varepsilon = 10^{-4}$ (red).

3 Interval Inclusion Functions

This section presents inclusions for several functions that do not appear in standard interval libraries. Each one is designed to be locally Lipschitz so it can be used in the construction of $\mathscr{R}$ satisfying Assumption 2.

The first is the multi-valued square root on $\mathbb{R}_+$. The standard inclusion function is $\pm\sqrt{X} = [-\sqrt{x^U}, +\sqrt{x^U}]$. Unfortunately, this inherits the non-Lipschitz behavior of $\sqrt{x}$ at $x = 0$. To avoid this, we define a weaker inclusion using upper and lower linearizations around $x = \varepsilon > 0$ (see Figure 1). When $x^U \geq \varepsilon$, the inclusion function returns $[-\sqrt{x^U}, +\sqrt{x^U}]$ as usual. However, when $x^U < \varepsilon$, weaker bounds based on the outer linearizations are returned.

Definition 3 For any $\varepsilon > 0$, define $\bar{\sqrt{\ }} : \mathbb{IR}^+ \to \mathbb{IR}$ by

$$\bar{\sqrt{X}} = \begin{cases} \left[-\sqrt{x^U}, \sqrt{x^U}\right] & \text{if } x^U \geq \varepsilon, \\ \left[-\frac{1}{2\sqrt{\varepsilon}}x^U - \frac{\sqrt{\varepsilon}}{2}, \frac{1}{2\sqrt{\varepsilon}}x^U + \frac{\sqrt{\varepsilon}}{2}\right] & \text{if } x^U < \varepsilon. \end{cases}$$

Theorem 1 *If $X \in \mathbb{IR}^+$ and $x \in X$, then $\pm\sqrt{x} \in \bar{\sqrt{X}}$.*

Proof See Subsection 1.1 in Supplementary Information (S.I.) for this paper. □

Theorem 2 *$\bar{\sqrt{\ }}$ is Lipschitz continuous on $\mathbb{IR}^+$.*

Proof See Subsection 1.2 in the S.I. □

Remark 3 Although $\bar{\sqrt{\ }}$ is defined using linear approximations, Theorem 1 ensures that it always produces a valid enclosure. It is approximate only in the sense that it returns

a weaker interval than necessary when $x^U < \varepsilon$. This is the concession we must make to ensure Lipschitz continuity.

Next, we consider arcsin, which is not locally Lipschitz at $x = -1$ and $x = 1$ (Figure 2). In this case, it suffices for our purposes to simply restrict the domain of our inclusion function to intervals contained in $(-1,1)$ and establish local Lipschitz continuity there.

Definition 4 Let $D \equiv (-1,1)$. Define $\overline{\arcsin} : \mathbb{I}D \to \mathbb{IR}$ by

$$\overline{\arcsin}(X) = [\arcsin(x^L), \arcsin(x^U)]. \tag{10}$$

Theorem 3 *For any $X \in \mathbb{I}D$ and $x \in X$, $\arcsin(x) \in \overline{\arcsin}(X)$.*

Proof The result follows immediately from the fact that arcsin is monotonically increasing on $(-1,1)$. □

Theorem 4 $\overline{\arcsin}$ *is locally Lipschitz continuous on $\mathbb{I}D$.*

Proof See Subsection 1.3 in the S.I. □

Next, consider the multi-valued arccos with both positive and negative branches. From Figure 2, it is clear that the inclusion function $\arccos(X) = [-\arccos(x^L), \arccos(x^L)]$ is valid, but it inherits the non-Lipschitz behavior of arccos at -1 and 1. For our purposes, it suffices to simply exclude -1 from the domain of our inclusion function. However, we will need to consider intervals containing 1. Therefore, we propose a weaker inclusion using upper and lower linearizations at some ε close to 1 (Figure 2).

Definition 5 Let $D \equiv (-1,1]$, choose any $\varepsilon \in (0,1)$, and define $\overline{\arccos} : \mathbb{I}D \to \mathbb{IR}$ by

$$\overline{\arccos}(X) = \begin{cases} [-\arccos(x^L), \arccos(x^L)] & \text{if } x^L \le \varepsilon, \\ \left[\frac{1}{\sqrt{1-\varepsilon^2}}(x^L - \varepsilon) - \arccos(\varepsilon), \right. \\ \left. -\frac{1}{\sqrt{1-\varepsilon^2}}(x^L - \varepsilon) + \arccos(\varepsilon)\right] & \text{if } x^L > \varepsilon. \end{cases}$$

Theorem 5 *For any $X \in \mathbb{I}D$ and $x \in X$, $\pm\arccos(x) \in \overline{\arccos}(X)$.*

Proof See Subsection 1.4 in the S.I. □

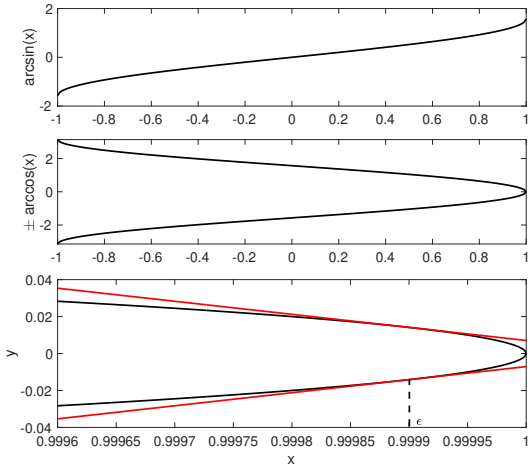

Fig. 2: The arcsin function (top), the multi-valued arccos function on $[-1,1]$ (middle), and the multi-valued arccos function near $x = 1$ with lower and upper linearizations at $\varepsilon = 0.9999$ (red).

Theorem 6 $\overline{\arccos}$ *is locally Lipschitz continuous on $\mathbb{I}D$.*

Proof See Subsection 1.5 in the S.I. □

Finally, we define inclusion functions for the following four trigonometric functions (see Figure 3).

Definition 6 Let $D = \{x \in \mathbb{R} : -\pi/2 < x < \pi/2\}$ and define $h_1, h_2, h_3, h_4 : D \to \mathbb{R}$ by

$$h_1(x) = \begin{cases} \frac{\cos x - 1}{x} & x \neq 0, \\ 0 & x = 0. \end{cases} \tag{11}$$

$$h_2(x) = \begin{cases} \frac{\sin x}{x} & x \neq 0, \\ 1 & x = 0. \end{cases} \tag{12}$$

$$h_3(x) = \begin{cases} \frac{\cos x - 1}{x^2} & x \neq 0, \\ -0.5 & x = 0. \end{cases} \tag{13}$$

$$h_4(x) = \begin{cases} \frac{x\cos(x) - \sin(x)}{x^2} & x \neq 0, \\ 0 & x = 0. \end{cases} \tag{14}$$

In the following definition, $\text{mid}(a,b,c)$ denotes the middle value of its arguments; i.e., $\text{mid}(-10,5,4) = 4$.

Definition 7 Let $D = \{x \in \mathbb{R} : -\pi/2 < x < \pi/2\}$ and define $H_1, H_2, H_3, H_4 : \mathbb{I}D \to \mathbb{IR}$ by

$$H_1(X) = [h_1(x^U), h_1(x^L)], \tag{15}$$

$$H_2(X) = [\min(h_2(x^L), h_2(x^U)), h_2(\text{mid}(x^L, 0, x^U))], \tag{16}$$

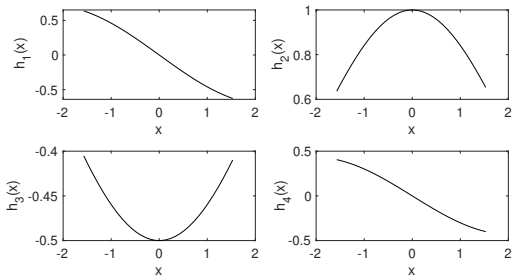

Fig. 3: The functions h_1–h_4 defined in Definition 6

$$H_3(X) = [h_3(\text{mid}(x^L, 0, x^U)), \max(h_3(x^L), h_3(x^U))], \quad (17)$$
$$H_4(X) = [h_4(x^U), h_4(x^L)]. \quad (18)$$

Theorem 7 *For any $X \in \mathbb{I}D$ and $x \in X$, we have $h_1(x) \in H_1(X)$, $h_2(x) \in H_2(X)$, $h_3(x) \in H_3(X)$, and $h_4(x) \in H_4(X)$.*

Proof See Subsection 1.6 in the S.I. □

Theorem 8 *H_1–H_4 are locally Lipschitz continuous on $\mathbb{I}D$.*

Proof See Subsection 1.7 in the S.I. □

Finally, we recall the extended intersection $X \bar{\cap} Z$ from [31], which equals $X \cap Z$ whenever $X \cap Z \neq \emptyset$ and otherwise returns a singleton on the boundary of X.

Definition 8 Define the *extended intersection* $\bar{\cap} : \mathbb{IR}^n \times \mathbb{IR}^n \to \mathbb{IR}^n$ componentwise for all $i \in \{1, \ldots, n\}$ by

$$(X \bar{\cap} Z)_i = \left[\text{mid}\left(x_i^L, x_i^U, z_i^L\right), \text{mid}\left(x_i^L, x_i^U, z_i^U\right)\right]. \quad (19)$$

Theorem 9 *The extended intersection $\bar{\cap}$ is Lipschitz continuous on $\mathbb{IR}^n$ with Lipschitz constant 1.*

Proof See Lemma 2.8 in [32]. □

Remark 4 The operations defined in this section will be used to define $\mathscr{R}$ for the examples in Section 4. Although some of these operations involve if-else statements, this never results in a hybrid reachability problem. Notably, $\mathscr{R}$ only affects the dynamics of the bounds, not the real vehicle. Moreover, these particular if-else statements do not cause discontinuities, as proven in Theorems 2, 4, 6, and 8. Consequently, $\mathscr{R}$ will be locally Lipschitz continuous and the evolution of the bounds via (7) will therefore be purely continuous.

4 Case Studies

In this section, we apply redundancy-based DI (Corollary 1) to obtain reachability bounds for two trajectory tracking problems and one path tracking problem. The first and third examples consider simple unicycle models using different control strategies. This is the simplest model used in the motion planning literature that is not fully actuated (due to its limited turning rate), and is therefore interesting for reachability analysis [1]. The second example considers a full size vehicle model, where both turning rate and acceleration are limited. The open-loop model is the same as in the verification benchmark in [33], but we consider a different control law. All case studies were run on a laptop with a 2.9 GHz Intel Core i7 processor. DI methods were implemented in C++ using the ODE solver CVODE with default settings [34].

4.1 Trajectory Tracking, Unicycle Model

Consider the following model, where (x, y) is the vehicle position (cm), v is the velocity (cm/s), θ is the heading angle (rad), and ω is the heading rate:

$$\begin{aligned} \dot{x} &= v\cos(\theta), \\ \dot{y} &= v\sin(\theta), \\ \dot{\theta} &= \omega. \end{aligned} \quad (20)$$

The reference trajectory $(x_{ref}, y_{ref}, \theta_{ref})$ is the solution of (20) with $\mathbf{x}_{ref,0} = (180.2, 10.34, 3.0)$ and the piecewise control inputs from Table 1. The actuation limit is $v + 26.2\omega \leq 65$ [35], while the maximum value in Table 1 is 48.22. Define

$$\begin{aligned} x_e &= \quad \cos(\theta)(x_{ref} - x) + \sin(\theta)(y_{ref} - y), \\ y_e &= -\sin(\theta)(x_{ref} - x) + \cos(\theta)(y_{ref} - y), \\ \theta_e &= \theta_{ref} - \theta. \end{aligned} \quad (21)$$

We use the tracking control law from [35]:

$$\begin{aligned} \omega &= \omega_{ref} + v_{ref}(k_2 y_e + k_3 \sin(\theta_e)) + d_1, \\ v &= v_{ref}\cos(\theta_e) + k_1 x_e + d_2, \end{aligned} \quad (22)$$

with $k_1 = 10\ \text{s}^{-1}$, $k_2 = 6.4 \times 10^{-3}$ rad/cm^2, and $k_3 = 0.16$ rad/cm. The disturbances d_1 and d_2 are not included in [35] but are assumed to corrupt the inputs here with $d_1(t) \in [-0.1, 0.1]$ and $d_2(t) \in [-1, 1]$. We also assume $\mathbf{x}_0 - \mathbf{x}_{ref,0} \in X_0 = [-5, 5] \times [-5, 5] \times$

Table 1: Reference control inputs for Example 4.1

Time (s)	[0,1]	[1,2]	[2,3]	[3,4]	[4,5]
ω (rad/s)	0.094	-0.680	-1	0.46	1
v (cm/s)	34.6	28.3	22.85	36.17	10.1
Time (s)	[5,6]	[6,7]	[7,8]	[8,9]	[9,10]
ω (rad/s)	-0.915	-0.2955	1.0	0.478	0
v (cm/s)	19.34	31.405	13.131	23.09	8.3

$[-\pi/6, \pi/6]$. In [35], it is shown that the vehicle trajectory converges to the reference when $d_1 = d_2 = 0$. Otherwise, the vehicle may not converge exactly. We aim to compute an enclosure of the real trajectories.

The most straightforward approach for bounding x, y, and θ using DI is to apply standard DI directly to (20) with (22). This requires an inclusion function for the closed-loop right-hand sides, which can be computed as follows. Given intervals X, Y, and Θ, intervals X_e, Y_e, and Θ_e are first computed by evaluating (21) in interval arithmetic. Then, bounds on the control inputs V and Ω are computed using (22). Finally, the right-hand sides of (20) are evaluated in interval arithmetic. The result is shown in Figure 4 (green) along with 500 sampled trajectories with $\mathbf{x}_0$, d_1, and d_2 drawn from uniform distributions (gray). Clearly, the bounds are very conservative. This is largely due to a significant dependency problem in the inclusion function described above. Specifically, θ affects the right-hand sides of x and y directly through (20) and again through the control input v using (22) and (21). In real arithmetic, this allows the controller to cancel the systems natural dynamics and impose the desired behavior. However, in interval arithmetic, the θ in the original dynamics is treated as independent from that in the control law, and the bounds explode quickly.

To mitigate this problem, a better approach is to apply standard DI to the dynamics of the errors (21):

$$\begin{aligned} \dot{x}_e &= \omega y_e - v + v_{ref}\cos\theta_e, \\ \dot{y}_e &= -\omega x_e + v_{ref}\sin(\theta_e), \\ \dot{\theta}_e &= \omega_{ref} - \omega. \end{aligned} \tag{23}$$

Plugging in the control law (22) and simplifying gives

$$\begin{aligned} \dot{x}_e = {} & \left(\omega_{ref} + v_{ref}\left(k_2 y_e + k_3 \sin\left(\theta_e\right)\right) + d_1\right) y_e \\ & - k_1 x_e - d_2, \\ \dot{y}_e = {} & -\left(\omega_{ref} + v_{ref}(k_2 y_e + k_3 \sin(\theta_e)) + d_1\right) x_e \\ & + v_{ref}\sin(\theta_e), \\ \dot{\theta}_e = {} & -v_{ref}(k_2 y_e + k_3 \sin(\theta_e)) - d_1. \end{aligned} \tag{24}$$

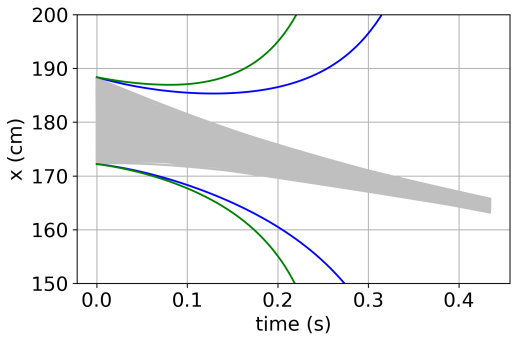

Fig. 4: Example 4.1: Bounds on x produced by applying standard DI to (20) (green) and (24) (blue) with 500 sampled trajectories (gray).

Applying DI to (24) is expected to be more effective because the action of the control law is represented more explicitly in these coordinates and can be better captured by simple interval computations. For example, the nonlinear term $v_{ref}\cos\theta_e$ has been completely cancelled from the right-hand side for x_e. Since this simplification is done analytically, before the use of interval arithmetic, the dependency problem is reduced. Once bounds are computed, they can be mapped back to the original coordinates by interval evaluation of the inverse transformation

$$\begin{aligned} x &= x_{ref} - \cos(\theta_{ref} - \theta_e)x_e + \sin(\theta_{ref} - \theta_e)y_e, \\ y &= y_{ref} - \sin(\theta_{ref} - \theta_e)x_e - \cos(\theta_{ref} - \theta_e)y_e, \\ \theta &= \theta_{ref} - \theta_e. \end{aligned} \tag{25}$$

Figure 4 shows that this approach (blue) yields tighter bounds, but they are still weak and diverge quickly.

To further improve, we now manufacture invariants for (24). The aim is to find a C^1 function ϕ such that $\frac{\partial \phi}{\partial \mathbf{x}}\mathbf{f}$ simplifies to a form that is likely to be bounded accurately using DI. Inspection of (24) shows that this cannot be done using any affine ϕ, unlike most models in [27]. Specifically, although there are common nonlinear terms that would be advantageous to cancel (e.g. $\omega_{ref} + v_{ref}\left(k_2 y_e + k_3 \sin\left(\theta_e\right)\right) + d_1$), they cannot be cancelled out by any linear combination of these ODEs. Therefore, a nonlinear ϕ is needed. For reasons discussed below, an excellent candidate is the Lyapunov function [35]

$$\mathscr{V} = \frac{1}{2}(x_e^2 + y_e^2) + \frac{(1 - \cos(\theta_e))}{k_2}. \tag{26}$$

To use (26) as a manufactured invariant, we define $\mathscr{V}$ as a new state variable and augment (24) with the ODE derived by differentiating $\mathscr{V}$. This ODE benefits from several algebraic simplifications, ultimately leading to

$$\dot{\mathscr{V}} = -k_1 x_e^2 - \frac{v_{ref} k_3 \sin^2(\theta_e)}{k_2} - x_e d_2 + \frac{\sin(\theta_e) d_1}{k_2}. \tag{27}$$

Let $\mathbf{z} \equiv (x_e, y_e, \theta_e, \mathscr{V})$ and $\mathbf{p} = (d_1, d_2)$ be shorthand for generic augmented state and disturbance vectors. Then, by definition, the augmented system consisting of (24) and (27) satisfies Assumption 1 with

$$G = \left\{ (t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^7 : \text{ (26) holds} \right\}. \tag{28}$$

It remains to define $\mathscr{R}$ satisfying Assumption 2. As in [25], this is done in two steps. Given $(t, Z, P) \in D_{\mathscr{R}}$, a refined interval $Z^\dagger \times P^\dagger$ is first computed such that $\{t\} \times Z^\dagger \times P^\dagger$ contains $(\{t\} \times Z \times P) \cap G$. Next, $\mathbf{f}$ is bounded over $\{t\} \times Z^\dagger \times P^\dagger$. In [25], $Z^\dagger \times P^\dagger$ is computed using a variant of the interval Krawczyk method [29]. Here, we use a custom method based on the following rearrangements of (26):

$$\begin{aligned} x_e^2 &= 2\left(\mathscr{V} - \frac{(1-\cos(\theta_e))}{k_2} - \frac{1}{2} y_e^2\right), \\ y_e^2 &= 2\left(\mathscr{V} - \frac{(1-\cos(\theta_e))}{k_2} - \frac{1}{2} x_e^2\right), \\ \cos\theta_e &= 1 - k_2\left(\mathscr{V} - \frac{1}{2}(x_e^2 + y_e^2)\right). \end{aligned} \tag{29}$$

Given any $Z = X_e \times Y_e \times \Theta_e \times V$, the right-hand sides of these equations can be bounded and used to refine X_e, Y_e, and Θ_e. Moreover, this can be done iteratively.

The complete definition of $\mathscr{R}$ is given in Algorithm 1. The refinements are done in the loop, while $\mathbf{f}$ is bounded in lines 13–16. All operations are done using standard interval arithmetic or the operations from Section 3, and we choose $l = 2$. A proof that this algorithm satisfies Assumption 2 is given in the S.I.

Figure 5 compares the results of applying standard DI to (24) (Method (i), blue) and applying redundancy-based DI to the augmented system (24) and (27) with $\mathscr{R}$ from Algorithm 1 (Method (ii), green). The non-smoothness of the bounds in some figures is caused by the piecewise inputs used to generate the reference. While the standard DI bounds rapidly diverge, the redundancy-based DI bounds are much more accurate and diverge slowly if at all. This

Algorithm 1 $\mathscr{R}$ for Example 4.1

1: **function** $\mathscr{R}(t, Z, P)$
2: $(X_e, Y_e, \Theta_e, V) \leftarrow Z$, $(P_1, P_2) \leftarrow P$
3: **for** $i = 1$ to l **do**
4: $V \leftarrow V \cap \left(\frac{1}{2}(X_e^2 + Y_e^2) + \frac{(1-\cos(\Theta_e))}{k_2}\right)$
5: $SQ_{X_e} \leftarrow X_e^2 \cap 2\left(V - \frac{(1-\cos(\Theta_e))}{k_2} - \frac{1}{2}Y_e^2\right)$
6: $X_e \leftarrow X_e \cap \pm\sqrt{SQ_{X_e}}$
7: $SQ_{Y_e} \leftarrow Y_e^2 \cap 2\left(V - \frac{(1-\cos(\Theta_e))}{k_2} - \frac{1}{2}X_e^2\right)$
8: $Y_e \leftarrow Y_e \cap \pm\sqrt{SQ_{Y_e}}$
9: $COS_{\Theta_e} \leftarrow \cos(\Theta_e) \cap \left(1 - k_2(V - \frac{1}{2}Y_e^2 - \frac{1}{2}X_e^2)\right)$
10: $\Theta_e \leftarrow \Theta_e \cap \pm\arccos(COS_{\Theta_e})$
11: **end for**
12: $\Psi \leftarrow (k_2 Y_e + k_3 \sin\Theta_e)$
13: $\Sigma_1 \leftarrow (\omega_{ref} + v_{ref}\Psi + P_1) Y_e - k_1 X_e - P_2$
14: $\Sigma_2 \leftarrow -(\omega_{ref} + v_{ref}\Psi + P_1) X_e + v_{ref}\sin\Theta_e$
15: $\Sigma_3 \leftarrow -v_{ref}(k_2 Y_e + k_3 \sin(\Theta_e)) - P_1$
16: $\Sigma_4 \leftarrow -k_1 X_e^2 - \frac{v_{ref} k_3 \sin^2(\Theta_e)}{k_2} - X_e P_2 + \sin(\Theta_e) P_1 / k_2$
17: **return** $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$
18: **end function**

indicates that using the Lyapunov function as a manufactured invariant is very effective at mitigating the dependency problems discussed in Section 2. Figure 5 also shows the results of bounding (24) using CORA [36], which is a Matlab implementation of state-of-the-art reachability techniques based on conservative linearization (Method (v), yellow). These results are the best obtained after careful tuning of CORA's parameters[1]. Nevertheless, the bounds rapidly diverge, nearly overlapping with standard DI. To improve these bounds, we enabled partitioning in CORA with the max error before splitting set to 30, 30, and 10 for x_e, y_e, and θ_e, and a time step of 0.05s (Method (vi), cyan). This gave tighter bounds, but they could not be extended much beyond $t = 1$ s due to the exponential cost of the partitioning procedure.

Standard DI required only 0.0009s of computation time, but integration was stopped early due to divergence of the bounds. Method (ii) using the Lyapunov function required 0.46s. For context, it took $\sim$ 3s to simulate 3125 real trajectories, which corresponds to a grid with only 5 values for each uncertain variable. With partitioning, CORA required 31s to compute bounds to $t = 1$s and more than 1h to reach $t = 5$s. We conclude that redundancy-based DI offers the best trade-off between accuracy and efficiency, producing

[1] CORA 2022 stable with time step 0.01 s, reduction technique 'girard', 9 Taylor terms, using linearization without advanced linear error compensation, zonotope order 7, tensor order 2.

reasonable bounds with a cost equivalent to sampling about 480 trajectories and $\sim 20\times$ faster than the real travel time for this vehicle.

When $d_1 = d_2 = 0$, $\mathscr{V}$ is decreasing [35] and hence the errors are bounded for all time by the inequality $\frac{1}{2}(x_e^2 + y_e^2) + \frac{(1-\cos(\theta_e))}{k_2} \leq \mathscr{V}(x_{e,0}, y_{e,0}, \theta_{e,0})$. This raises the question of whether the redundancy-based DI bounds are substantially better than what can be inferred from the Lyapunov function alone. This is addressed in the next two examples, but does not apply here because $\mathscr{V}$ may not decrease with nonzero disturbances. This highlights the fact that the validity of these bounds derives solely from DI theory and does not hinge on any special properties of the manufactured invariant. In this example, $\mathscr{V}$ simply serves as a useful redundant equation because of the relatively simple form of (27). This suggests that redundancy-based DI offers a means to exploit 'approximate' (e.g., nominal) Lyapunov functions or invariants to characterize system behavior under uncertainty.

Figure 6 shows the bounds on the original coordinates (x, y, θ) obtained via (25). The accuracy of redundancy-based DI is retained in the original coordinates. Moreover, although these bounds leave some room for improvement, they appear accurate enough to support many motion planning or collision avoidance tasks, particularly for outdoor mobile robots.

Another potentially useful approach for introducing model redundancy is to write the model in multiple coordinate systems simultaneously. This could have advantages if some aspects of the model are more simply represented in the first coordinate system, and others in the second. To try this approach, we augmented (24) and (27) with the closed-loop dynamics in the original coordinates described by (20) with (22). In addition to (26), the states of this augmented system also satisfy the invariants (21) and (25). Therefore, we can define $\mathbf{z} = (x_e, y_e, \theta_e, \mathscr{V}, x, y, \theta)$ and

$$G = \left\{(t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^{10} : (21), (25), (26) \text{ hold}\right\}. \quad (30)$$

To apply the redundancy-based DI using this G, we modified Algorithm 1 to include refinements based on (21) and (25) in the loop after line 10 and compute bounds on the right-hand sides of (20) (which are now included in the augmented system) after line 16. The results are shown in Figures 5–6 (Method (iii), purple). In the error coordinates, the bounds lie entirely behind those of Method (ii). In the original coordinates, a very slight improvement can be seen, e.g., for y just before 4s. We also compared the bounds

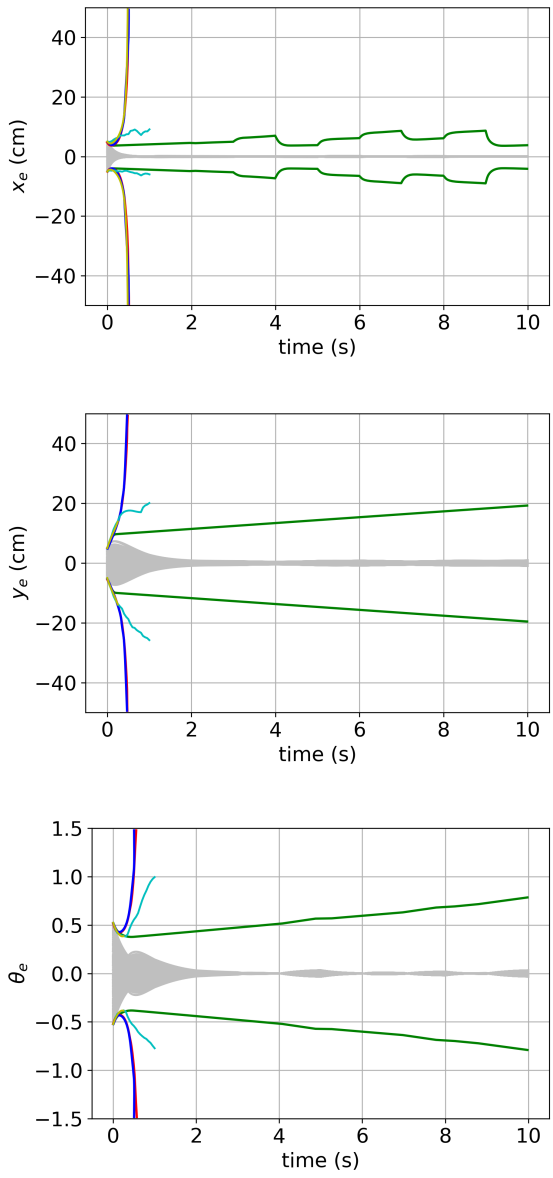

Fig. 5: Example 4.1: Bounds on (x_e, y_e, θ_e) from (i) applying standard DI to (24) (blue), (ii) applying redundancy-based DI to (24) and (27) with invariant (26) (green), (iii) applying redundancy-based DI to (24), (27), and (20) with invariants (21), (25), and (26) (purple), (iv) applying redundancy-based DI to (24) and (20) with invariants (21) and (25) (red), and (v & vi) applying CORA to (24) without and with partitioning (yellow & cyan) with 500 sampled trajectories (gray).

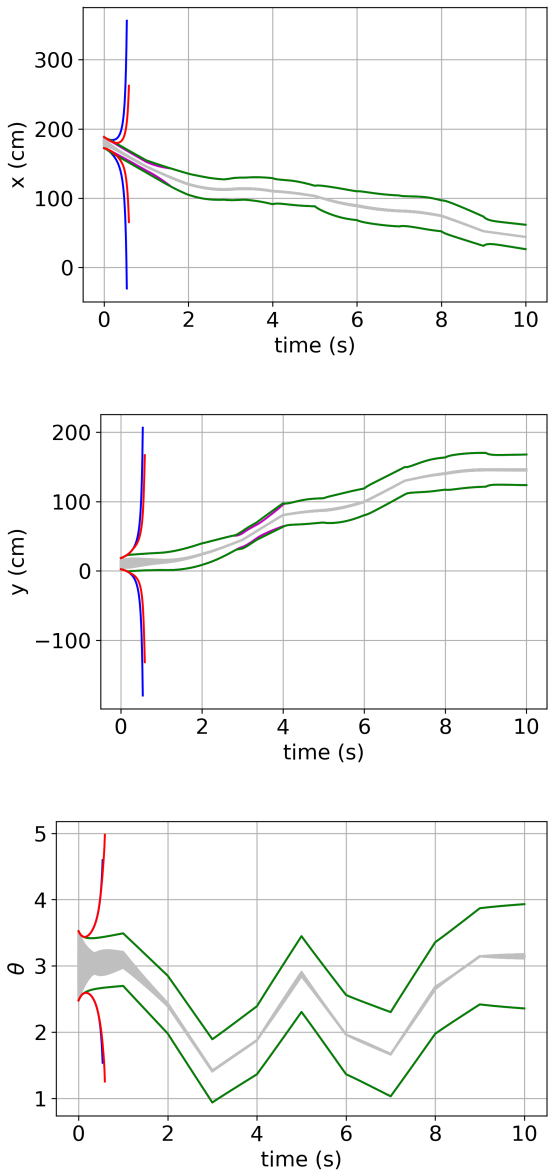

Fig. 6: Example 4.1: Bounds on (x, y, θ) from (i) applying standard DI to (24) (blue), (ii) applying redundancy-based DI to (24) and (27) with invariant (26) (green), (iii) applying redundancy-based DI to (24), (27), and (20) with invariants (21), (25), and (26) (purple), and (iv) applying redundancy-based DI to (24) and (20) with invariants (21) and (25) (red), with 500 sampled trajectories (gray).

obtained by using (21) and (25) without the Lyapunov function (Method (iv), red). This resulted in diverging bounds that are only slightly tighter than those of Method (i) (standard DI). Thus, the redundancy offered by using both the original and error coordinates simultaneously is ineffective at mitigating the dependency problem for this example. It appears that the error coordinates are universally better for interval computations in this case, so that refinements based on (21) and (25) have the effect of using (X_e, Y_e, Θ_e) to tighten (X, Y, Θ), but rarely the reverse.

4.2 Trajectory Tracking, Car Model

Consider the following extended model of a full size autonomous road vehicle of length $l = 2$ m [37]:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= v\frac{\tan\delta}{l}, \\ \dot{\delta} &= u_1, \\ \dot{v} &= \omega_2. \end{aligned} \tag{31}$$

Above, x and y are positions (m), θ and δ are the heading and steering angles (rad), and v is velocity (m/s). The control variables are the steering angle rate u_1 and the acceleration ω_2. The reference trajectory is the solution of the following simplified model from initial conditions $(0\text{ m}, 0\text{ m}, 0\text{ rad}, 10^{-5}\text{ m/s})$ using the piecewise control inputs in Table 2:

$$\begin{aligned} \dot{x}_{ref} &= v_{ref}\cos\theta_{ref}, \\ \dot{y}_{ref} &= v_{ref}\sin\theta_{ref}, \\ \dot{\theta}_{ref} &= \alpha_{ref}, \\ \dot{v}_{ref} &= \omega_2. \end{aligned} \tag{32}$$

The benchmark in [33] limits acceleration for a similar sized car using the same model to 11 m/s^2, while the maximum value in Table 2 is 10 m/s^2.

We apply the tracking control law from [37] based on the global diffeomorphic coordinate transformation

$$\begin{aligned} e_t &= \cos(\theta_{ref})(x - x_{ref}) + \sin(\theta_{ref})(y - y_{ref}), \\ e_n &= -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref}), \\ e_\theta &= \theta - \theta_{ref}, \\ v &= v, \\ \kappa_\delta &= \tan(\delta)/l. \end{aligned} \tag{33}$$

Table 2: Reference control inputs for Example 4.2

Time interval (s)	[0,0.8]	[0.8,1.6]	[1.6,2.4]
α_{ref} (rad/s)	0.1	-2.05	1.16
ω_2 (m/s^2)	10	-4.68	1.59
Time interval (s)	[2.4,3.2]	[3.2,4]	[4,4.8]
α_{ref} (rad/s)	2.3	-0.05	-4.88
ω_2 (m/s^2)	-1.67	-2.9	4.15
Time intervals (s)	[4.8,5.6]	[5.6,6.4]	[6.4,7.2]
α_{ref} (rad/s)	5.47	-0.56	-2.23
ω_2 (m/s^2)	-0.35	-3.63	3.56

Noting that $\dot{\kappa}_\delta = [l^{-1} + l\kappa_\delta^2]u_1$, define the virtual control variable $\omega_1 = [l^{-1} + l\kappa_\delta^2]u_1$. The control law is

$$\begin{aligned} \omega_1 &= -e_\theta v + \dot{\xi} - k_4(\kappa_\delta - \xi), \\ \omega_2 &= \dot{v}_{ref} - k_1 e_t - k_3(v - v_{ref}) + k_2 e_\theta^2 - e_\theta \kappa_{ref}, \end{aligned} \tag{34}$$

where $(k_1, k_2, k_3, k_4) = (2, 3, 1, 10)$, $\kappa_{ref} = \dot{\theta}_{ref}/v_{ref}$, and ξ is defined with h_1–h_4 from Definition 6 by

$$\begin{aligned} \xi &= \kappa_{ref} - k_1[e_t h_1(e_\theta) + e_n h_2(e_\theta)] - k_2 e_\theta, \\ \dot{\xi} &= \dot{\kappa}_{ref} - k_1[\dot{e}_t h_1(e_\theta) - e_t \dot{e}_\theta (h_2(e_\theta) + h_3(e_\theta)) \\ &\quad + \dot{e}_n h_2(e_\theta) + e_n \dot{e}_\theta h_4(e_\theta)] - k_2 \dot{e}_\theta. \end{aligned} \tag{35}$$

We aim to bound the solutions of the closed-loop system (31), (33)–(35) subject to uncertain initial conditions $(e_t, e_n, e_\theta) \in [0,1] \times [0,1] \times [-\pi/6, \pi/6]$. The initial conditions of v and κ_δ are 0. The most straightforward approach is to apply standard DI to (31) with (33)–(35). But, as in Example 4.1, this suffers from major dependency problems. Again, a better approach is to bound the error coordinates used in the feedback law. After some simplification, the error dynamics are

$$\begin{aligned} \dot{e}_t &= v\cos e_\theta - v_{ref}[1 - \kappa_{ref} e_n], \\ \dot{e}_n &= v \sin e_\theta - v_{ref} \kappa_{ref} e_t, \\ \dot{e}_\theta &= v\kappa_\delta - v_{ref}\kappa_{ref}, \\ \dot{\kappa}_\delta &= \omega_1, \\ \dot{v} &= \omega_2. \end{aligned} \tag{36}$$

The results of applying standard DI to (36) with (34) and (35) are show in Figures 7–8 (blue). The bounds are significantly tighter than those obtained by applying DI in the original coordinates (not shown), but still extremely conservative.

We now manufacture invariants for (36). Following Example 4.1, we consider the Lyapunov function

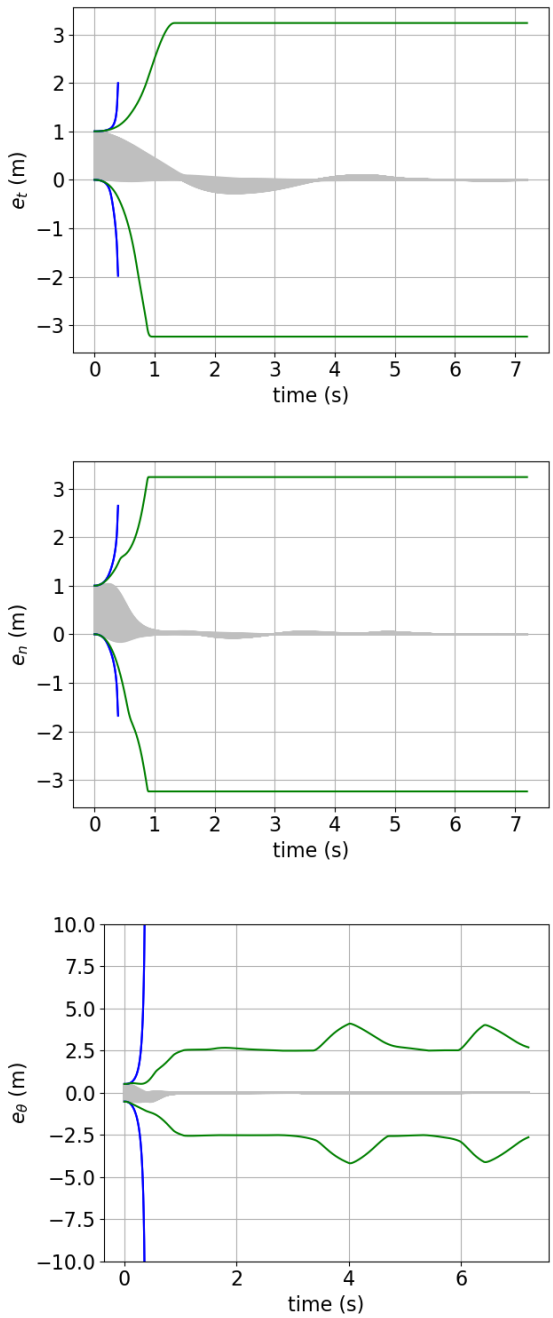

Fig. 7: Example 4.2: Bounds on (e_t, e_n, e_θ) from (i) applying standard DI to (36) (blue) and (ii) applying redundancy-based DI to (36) and (38) with invariants (37) (green) with 500 sampled trajectories (gray).

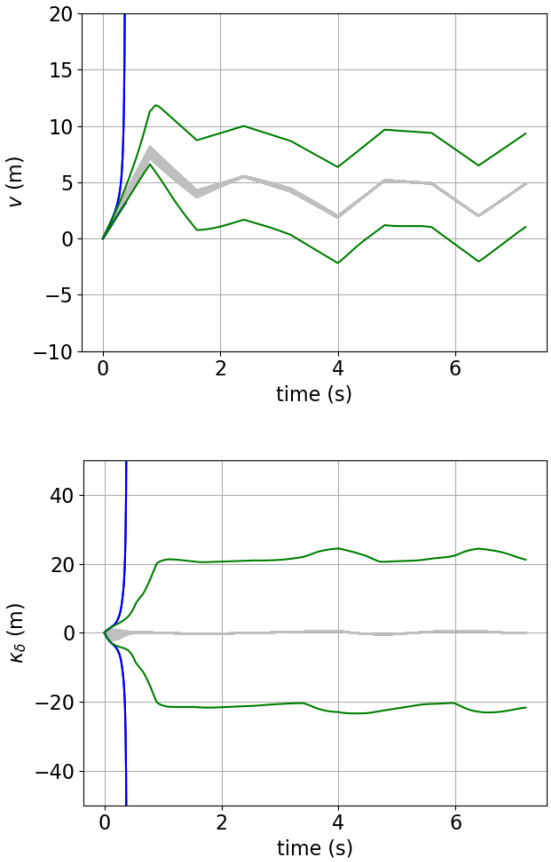

Fig. 8: Example 4.2: Bounds on (v, κ_δ) from (i) applying standard DI to (36) (blue) and (ii) applying redundancy-based DI to (36) and (38) with invariants (37) (green) with 500 sampled trajectories (gray).

and the variable e_δ used in the analysis in [37]:

$$\mathscr{L}_C = [k_1 e_t^2 + k_1 e_n^2 + e_\theta^2 + (v - v_{ref})^2 + e_\delta^2]/2,$$
$$e_\delta = \kappa_\delta - \xi. \tag{37}$$

To use these functions as manufactured invariants, we define $\mathscr{L}_C$ and e_δ as new state variables and augment (36) with their dynamics. After some beneficial simplifications, these are

$$\dot{\mathscr{L}}_C = -v_{ref} k_2 e_\theta^2 - k_3 (v - v_{ref})^2 - k_4 e_\delta^2,$$
$$\dot{e}_\delta = -e_\theta v - k_4 e_\delta. \tag{38}$$

With $\mathbf{z} = (e_t, e_n, e_\theta, k_\delta, v, \mathscr{L}_C, e_\delta)$ and $\mathbf{p} = \emptyset$ denoting generic state and disturbance vectors, the system consisting of (36) and (38) satisfies Assumption 1 with

$$G = \left\{ (t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^9 : (37) \text{ holds} \right\}. \tag{39}$$

We define $\mathscr{R}$ by Algorithm 2 with $l = 2$, which includes several refinements based on (37). A proof that this satisfies Assumption 2 is given in the S.I.

Algorithm 2 $\mathscr{R}$ for Example 4.2

function $\mathscr{R}(t, Z, P)$
  $(E_t, E_n, E_\theta, K_\delta, V, L_C, E_\delta) \leftarrow Z$
  for $i = 1$ to l **do**
    $E_v \leftarrow V - v_{ref}$
    $SQ_{E_t} \leftarrow E_t^2 \barwedge \frac{1}{k_1}(2L_C - E_\theta^2 - k_1 E_n^2 - E_v^2 - E_\delta^2)$
    $E_t \leftarrow E_t \barwedge \sqrt{SQ_{E_t}}$
    $SQ_{E_n} \leftarrow E_n^2 \barwedge \frac{1}{k_1}(2L_C - E_\theta^2 - k_1 E_t^2 - E_v^2 - E_\delta^2)$
    $E_n \leftarrow E_n \barwedge \sqrt{SQ_{E_n}}$
    $SQ_{E_\theta} \leftarrow E_\theta^2 \barwedge (2L_C - k_1 E_t^2 - k_1 E_n^2 - E_v^2 - E_\delta^2)$
    $E_\theta \leftarrow E_\theta \barwedge \sqrt{SQ_{E_\theta}}$
    $SQ_{E_v} \leftarrow E_v^2 \barwedge (2L_C - k_1 E_t^2 - k_1 E_n^2 - E_\theta^2 - E_\delta^2)$
    $E_v \leftarrow E_v \barwedge \sqrt{SQ_{E_v}}$
    $SQ_{E_\delta} \leftarrow E_\delta^2 \barwedge (2L_C - k_1 E_t^2 - k_1 E_n^2 - E_v^2 - E_\theta^2)$
    $E_\delta \leftarrow E_\delta \barwedge \sqrt{SQ_{E_\delta}}$
    $\Xi = \kappa_{ref} - k_1[E_t H_1(E_\theta) + E_n H_2(E_\theta)] - k_2 E_\theta$
    $E_\delta \leftarrow E_\delta \barwedge (K_\delta - \Xi)$
    $K_\delta \leftarrow K_\delta \barwedge (E_\delta + \Xi)$
  end for
  $V \leftarrow E_v + v_{ref}$
  $\Sigma_1 \leftarrow V \cos E_\theta - v_{ref}[1 - \kappa_{ref} E_n]$
  $\Sigma_2 \leftarrow V \sin E_\theta - v_{ref} \kappa_{ref} E_t$
  $\Sigma_3 \leftarrow V K_\delta - v_{ref} \kappa_{ref}$
  $\dot{\Xi} \leftarrow \dot{\kappa}_{ref} - k_1[\Sigma_1 H_1(E_\theta) - E_t \Sigma_3 (H_2(E_\theta) + H_3(E_\theta)) + \Sigma_3 H_2(E_\theta) + E_n \Sigma_3 H_4(E_\theta)] - k_2 \Sigma_3$
  $\Sigma_4 \leftarrow -E_\theta V + \dot{\Xi} - k_4 E_\delta$
  $\Sigma_5 \leftarrow \dot{v}_{ref} - k_1 E_t - k_3 E_v + k_2 E_\theta^2 - E_\theta \kappa_{ref}$
  $\Sigma_6 \leftarrow -v_{ref} k_2 E_\theta^2 - k_3 E_v^2 - k_4 E_\delta^2$
  $\Sigma_7 \leftarrow -E_\phi V - k_4 E_\delta$
return $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \Sigma_7)$
end function

Figures 7–8 show the results of applying standard DI to (36) (Method (i), blue) and redundancy-based DI to the augmented system (36) and (38) with $\mathscr{R}$ from Algorithm 2 (Method (ii), green). While standard DI rapidly diverges, redundancy-based DI is much more accurate and does not diverge. Thus, using a Lyapunov function as a manufactured invariant is effective for this example as well. A comparison to CORA was also attempted for this example. To avoid interval division by zero errors in CORA, we had

to replace the functions h_1–h_4 from (6) with polynomial approximations, but this had minimal impact on the trajectories. Unfortunately, CORA still failed in the first time step with the error "Abort analysis due to reachable set explosion!". In terms of computation time, standard DI required only 0.0018s, but the integration was stopped early due to divergence of the bounds. Method (ii) using the Lyapunov function required 0.222s. For comparison, approximating the reachable set by simulating solutions on a grid with 20 points for every uncertain initial condition (i.e., 8000 trajectories) required 14.2s. Thus, Method (ii) produces the most accurate bounds with a cost equivalent to sampling ~ 125 trajectories and $\sim 27\times$ faster than the real travel time for this vehicle.

Since $\mathscr{L}_C$ is decreasing by (38), it already implies that the errors remain within the ellipsoid $[k_1 e_t^2 + k_1 e_n^2 + e_\theta^2 + (v - v_{ref})^2 + e_\delta^2]/2 \leq \mathscr{L}_C(t=0)$. Method (ii) produces bounds on e_t and e_n that are tighter than the bounds implied by this ellipsoid before 0.8s, but equal afterwards, indicating that the Lyapunov function alone is mostly responsible for the improved bounds. In contrast, the Lyapunov ellipsoid only implies that $|e_\theta|$ is bounded by 4.6, which is much weaker than the bound of $\sim$2.5 furnished by Method (ii) over most of the time horizon. Thus, the combination of DI with the Lyapunov function achieves tighter bounds than can be inferred from either method alone.

Figure 9 shows the bounds on the vehicle's position computed by interval evaluation of

$$\begin{aligned} x &= \cos(\theta_{ref})e_t - \sin(\theta_{ref})e_n + x_{ref} \\ y &= \sin(\theta_{ref})e_t + \cos(\theta_{ref})e_n + y_{ref} \\ \theta &= e_\theta + \theta_{ref}. \end{aligned} \tag{40}$$

The use of redundancy in Method (ii) still leads to vastly tighter bounds than standard DI. However, with an error of $\pm$5m in the xy-plane, these bounds still leave significant room for improvement and may be unsuitable for some motion planning tasks. Additional manufactured invariants that would improve the bounds further may exist, but we were unable to find any. As in Example 4.1, we attempted to use the original and error coordinates simultaneously, but this gave no improvement. We conclude that further progress on problems of this complexity will likely still require methods based on more sophisticated set arithmetics.

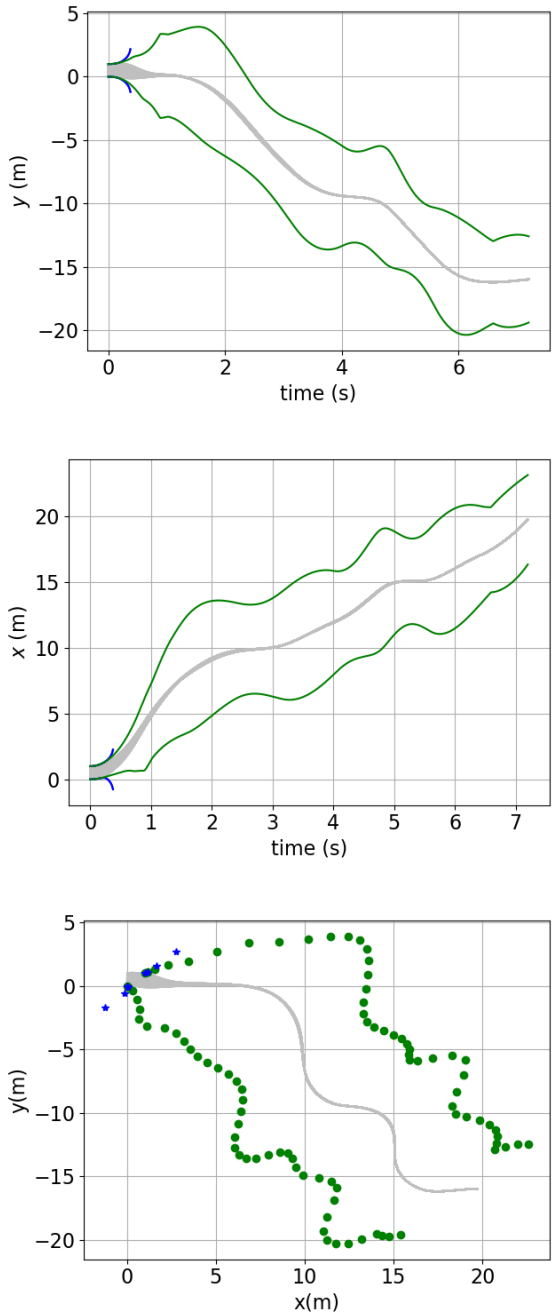

Fig. 9: Example 4.2: Bounds on (x, y) from (i) applying standard DI to (36) (blue) and (ii) applying redundancy-based DI to (36) and (38) with invariants (37) (green) with 500 sampled trajectories (gray).

4.3 Path Tracking, Dubins Car

Consider the model (20) again, but now with velocity as a time-invariant uncertain parameter $v \in V$. The control objective is to track a smooth path $C \equiv \{(x_{ref}(s), y_{ref}(s)) : s \in \mathbb{R}\}$ where $(x_{ref}, y_{ref}) \in \mathscr{C}^2(\mathbb{R}, \mathbb{R}^2)$. Define the tangent to C at s by

$$\mathbf{n}(s) \equiv \left(\frac{\partial x_{ref}}{\partial s}(s), \frac{\partial y_{ref}}{\partial s}(s) \right). \tag{41}$$

We assume the path is parameterized such that $s = 0$ is the desired starting point and $\|\mathbf{n}(s)\|_2 = 1$ for all $s \in \mathbb{R}$. This implies that, for any $s_2 > s_1$, $s_2 - s_1$ is the arclength of the path connecting $(x_{ref}(s_1), y_{ref}(s_1))$ and $(x_{ref}(s_2), y_{ref}(s_2))$. Under this assumption, the *curvature* of C at s is $c(s) = \|\frac{d\mathbf{n}}{ds}(s)\|_2$. We assume that $\bar{c} \equiv \sup_{s\in\mathbb{R}} |c(s)| < +\infty$ and that, for every $(x, y) \in C$, the circle with radius $1/\bar{c}$ that is tangent to C at (x, y) does not contain any points of C in its interior [38].

We use the tracking controller in [38] based on the curvilinear coordinates defined as follows. Consider a single trajectory $(x(t), y(t), \theta(t))$ of (20) corresponding to some $\mathbf{x}_0 \in X_0$, $v \in V$, and $\omega : [t_0, t_f] \to \mathbb{R}$. Assume the distance between $(x(t), y(t))$ and C is less than $1/\bar{c}$ for all $t \in [t_0, t_f]$. Then, the projection of $(x(t), y(t))$ onto C is well-defined and we may define the curvilinear coordinate $s : [t_0, t_f] \to \mathbb{R}$ as

$$s(t) \equiv \operatorname*{argmin}_{\gamma\in\mathbb{R}} \|(x(t), y(t)) - (x_{ref}(\gamma), y_{ref}(\gamma))\|_2. \tag{42}$$

Following [1], define the x and y coordinate errors by

$$\mathbf{r}(t) \equiv (x(t) - x_{ref}(s(t)), y(t) - y_{ref}(s(t))). \tag{43}$$

Moreover, define the tracking error $e : [t_0, t_f] \to \mathbb{R}$ as the perp dot product of $\mathbf{r}$ with $\mathbf{n}$,

$$e(t) \equiv \mathbf{r}(t)^{\perp} \cdot \mathbf{n}(s(t)) = r_x(t) n_y(s(t)) - r_y(t) n_x(s(t)).$$

Finally, define the reference heading angle $\theta_{ref}(s)$ at any point in C as the angle between $\mathbf{n}(s)$ and the positive x-axis, $\theta_{ref}(s) = \operatorname{atan2}(n_y(s), n_x(s))$, and define the tracking angle error $\theta_e : [t_0, t_f] \to \mathbb{R}$ as

$$\theta_e(t) = \theta(t) - \theta_{ref}(s(t)). \tag{44}$$

According to [38], the trajectory $(s(t), e(t), \theta_e(t))$ defined in this way satisfies the following ODEs:

$$\begin{aligned} \dot{s} &= \frac{v\cos(\theta_e)}{1 - c(s)e}, \\ \dot{e} &= v\sin(\theta_e), \\ \dot{\theta}_e &= \omega - \frac{vc(s)\cos(\theta_e)}{1 - c(s)e}. \end{aligned} \tag{45}$$

Following [38], we apply the tracking feedback law

$$\omega = \frac{vc(s)\cos(\theta_e)}{1 - c(s)e} - g_1\theta_e - (g_2 v h_2(\theta_e))\, e, \tag{46}$$

where h_2 is from Definition 6, $g_1 = 5.71\sqrt{v^2 + 0.1}$, and $g_2 = 4$. The reachability problem is to bound the vehicle position at each t under the control law (46). For our initial experiments, let $v \in [5, 6]$ m/s, let $(s_0, e_0, \theta_{e,0}) = (0, 1, \pi/6)$, and let C be a semi-circle with $c(s) = 1/30$ emanating from $(x_0, y_0) = (0, 0)$.

A complication that did not occur in the trajectory tracking examples is that there is no closed-form expression for computing (s, e, θ_e) from (x, y, θ). Specifically, s is determined implicitly from (42), and e and θ_e depend on s. Thus, the closed-loop system cannot be written in the original coordinates in a form amenable to bounding calculations. The most straightforward approach in this case is to directly bound the error coordinates using (45)–(46). This is advisable in any case because it potentially reduces the dependency problem as in Examples 4.1 and 4.2. However, converting these error bounds back to the original coordinates is also nontrivial and requires special care.

To bound the error coordinates, the simplest approach is to apply standard DI to (45)–(46). The dependency problem is significantly reduced by first substituting the control law into (45) and analytically cancelling $\frac{vc(s)\cos(\theta_e)}{1-c(s)e}$ to obtain

$$\begin{aligned} \dot{s} &= \frac{v\cos(\theta_e)}{1 - c(s)e}, \\ \dot{e} &= v\sin(\theta_e), \\ \dot{\theta}_e &= -g_1\theta_e - g_2 v h_2(\theta_e) e. \end{aligned} \tag{47}$$

However, bounding (47) still produces weak bounds that start to diverge after 0.4s (Figure 10, Method (i)).

To improve these bounds, we manufacture an invariant based on the Lyapunov function [38]

$$\mathscr{L} = \frac{1}{2}(e^2 + (1/g_2)\theta_e^2). \tag{48}$$

We define $\mathscr{L}$ as a new state, augment (45) with

$$\dot{\mathscr{L}} = -\frac{g_1}{g_2}\theta_e^2, \tag{49}$$

and define $\mathscr{R}$ as Algorithm 3 with $l = 2$ (see S.I. for proof of Assumption 2). Figure 10 shows that the resulting bounds (Method (ii)) are much more accurate than standard DI and do not diverge. Figure 10 also shows the bounds that result if the custom refinement steps in lines 3–9 of Algorithm 3 are replaced with the general-purpose κ-operator from [25] (Method (iii)). The custom refinements clearly lead to significantly tighter bounds.

Algorithm 3 $\mathscr{R}$ for Example 4.3

1: **function** $\mathscr{R}(t,Z,P)$
2: $(S \times E \times \Theta_e \times L) \leftarrow Z$ and $V \leftarrow P$
3: **for** $i = 1$ to l **do**
4: $L \leftarrow L \barwedge [\frac{1}{2}(E^2 + (1/g_2)\Theta_e^2)]$
5: $SQ_E \leftarrow E^2 \barwedge (2L - \frac{1}{g_2}\Theta_e^2)$
6: $E \leftarrow E \barwedge \bar{\sqrt{SQ_E}}$
7: $SQ_{\Theta_e} \leftarrow \Theta_e^2 \barwedge (g_2(2L - E^2))$
8: $\Theta_e \leftarrow \Theta_e \barwedge \bar{\sqrt{SQ_{\Theta_e}}}$
9: **end for**
10: $\Sigma_1 \leftarrow \frac{V\cos(\Theta_e)}{1-\bar{c}E}$
11: $\Sigma_2 \leftarrow V\sin(\Theta_e)$
12: $\Sigma_3 \leftarrow -5.71\sqrt{V^2+0.1}\Theta_e - g_2 V H_2(\Theta_e)E,$
13: $\Sigma_4 \leftarrow -\frac{5.71\sqrt{V^2+0.1}}{g_2} SQ_{\Theta_e}$
14: **return** $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$
15: **end function**

We now propose additional manufactured invariants to further improve the bounds. Defining $\phi_1 = v\cos(\theta_e)$ and $\phi_2 = v\sin(\theta_e)$ leads to the system

$$\begin{aligned}
\dot{s} &= \frac{\phi_1}{1 - c(s)e}, \\
\dot{e} &= \phi_2, \\
\dot{\theta}_e &= -g_1\theta_e - g_2 e v h_2(\theta_e), \\
\dot{\mathscr{L}} &= -\frac{g_1}{g_2}\theta_e^2, \\
\dot{\phi}_1 &= -v\sin(\theta_e)\dot{\theta}_e, \\
\dot{\phi}_2 &= v\cos(\theta_e)\dot{\theta}_e.
\end{aligned} \tag{50}$$

The solutions of this system satisfy (48) and

$$\begin{aligned}
\phi_1 &= v\cos(\theta_e), \\
\phi_2 &= v\sin(\theta_e), \\
v^2 &= \phi_1^2 + \phi_2^2, \\
\frac{\phi_2}{\phi_1} &= \tan(\theta_e).
\end{aligned} \tag{51}$$

To use all of these invariants, we modified $\mathscr{R}$ by adding refinements based on (51) as described in Algorithm 4, where $\bar{\sqrt{}}$, $\bar{\arcsin}$, and $\bar{\arccos}$ are defined in Section 3 (see S.I. for proof of Assumption 2). The resulting bounds are shown as Method (iv) in Figure 10. Adding ϕ_1 and ϕ_2 and the corresponding invariants (51) leads to even further improvement and provides the tightest bounds among all methods considered.

Algorithm 4 $\mathscr{R}$ for Example 4.3 with additional manufactured invariants

1: **function** $\mathscr{R}(t,Z,P)$
2: $(S \times E \times \Theta_e \times L \times \Phi_1 \times \Phi_2) \leftarrow Z$ and $V \leftarrow P$
3: **for** $i = 1$ to l **do**
4: Execute lines 4–8 in Algorithm 3
5: $\Phi_1 \leftarrow V\cos(\Theta_e) \barwedge \Phi_1$
6: $\Phi_2 \leftarrow \Phi_2 \barwedge V\sin(\Theta_e)$
7: $SQ_V \leftarrow V^2 \barwedge (\Phi_1^2 + \Phi_2^2)$
8: $V \leftarrow V \barwedge \bar{\sqrt{SQ_V}}$
9: $SQ_{\Phi_1} \leftarrow \Phi_1^2 \barwedge (V^2 - \Phi_2^2)$
10: $\Phi_1 \leftarrow \Phi_1 \barwedge \bar{\sqrt{SQ_{\Phi_1}}}$
11: $SQ_{\Phi_2} \leftarrow \Phi_2^2 \barwedge (V^2 - \Phi_1^2)$
12: $\Phi_2 \leftarrow \Phi_2 \barwedge \bar{\sqrt{SQ_{\Phi_2}}}$
13: $COS_{\Theta_e} \leftarrow \cos\Theta_e \barwedge (\Phi_1/V)$
14: $\Theta_e \leftarrow \Theta_e \barwedge \bar{\arccos}(COS_{\Theta_e})$
15: $SIN_{\Theta_e} \leftarrow \sin\Theta_e \barwedge (\Phi_2/V)$
16: $\Theta_e \leftarrow \Theta_e \barwedge \bar{\arcsin}(SIN_{\Theta_e})$
17: $TAN_{\Theta_e} \leftarrow \tan\Theta_e \barwedge (\Phi_2/\Phi_1)$
18: $\Theta_e \leftarrow \Theta_e \barwedge \arctan(TAN_{\Theta_e})$
19: **end for**
20: $\Sigma_1 \leftarrow \frac{\Phi_1}{1-\bar{c}E}$
21: $\Sigma_2 \leftarrow \Phi_2$
22: $\Sigma_3 \leftarrow -5.71\sqrt{V^2+0.1}\Theta_e - g_2 V H_2(\Theta_e)E,$
23: $\Sigma_4 \leftarrow -\frac{5.71\sqrt{V^2+0.1}}{g_2} SQ_{\Theta_e}$
24: $\Sigma_5 \leftarrow -V\sin(\Theta_e)\Sigma_3$
25: $\Sigma_6 \leftarrow V\cos(\Theta_e)\Sigma_3$
26: **return** $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6)$
27: **end function**

The Lyapunov function (48) already implies $\frac{1}{2}(e^2 + (1/g_2)\theta_e^2) \le V(e_0, \theta_{e,0})$. From this, it is easily shown that $|e|$ and $|\theta_e|$ are bounded by 1.04 m and 2.07

rad. However, these bounds are significantly weaker than those provided by Methods (ii) and (iv).

Although Method (iv) produces effective bounds on (s,e,θ_e), the goal is ultimately to bound the position (x,y). This is challenging because there is no simple relation giving (x,y) in terms of (s,e,θ_e) that can be evaluated in interval arithmetic. Unlike trajectory tracking, the error coordinates here do not simply specify deviations from a fixed reference point at each t. Rather, the interval $S(t)$ contains a range of s values at each t that indicate different "closest" points along the reference path, while $E(t)$ contains possible deviations in the directions normal to the path.

To address this, we now derive a method for mapping $S(t)$ and $E(t)$ into a sharp non-interval enclosure of $(x(t),y(t))$. Consider a single trajectory $(x(t),y(t))$ corresponding to some $v \in V$ and $\mathbf{x}_0 \in X_0$. From the definition of $\mathbf{r}(t)$, we have that $(x(t),y(t)) = (x_{ref}(s(t)),y_{ref}(s(t)))+\mathbf{r}(t)$. From the optimality conditions of (42), it can be shown that $\mathbf{r}(t)$ is always orthogonal to $\mathbf{n}(s(t))$. Moreover, using standard properties of the perp dot product defining $e(t)$, we can infer that $\mathbf{r}(t)$ points -90° relative to $\mathbf{n}(s(t))$ when $e(t) > 0$ and $+90^\circ$ relative to $\mathbf{n}(s(t))$ when $e(t) < 0$. Combining these facts, we conclude that

$$\mathbf{r}(t) = e(t)\begin{bmatrix} \sin(\theta_{ref}(s(t))) \\ -\cos(\theta_{ref}(s(t))) \end{bmatrix}, \tag{52}$$

where the vector on the right is the unit vector with angle $\theta_{ref}(s(t)) - \frac{\pi}{2}$. Therefore, for every $t \in I$,

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \in \left\{ \begin{bmatrix} x_{ref}(s) \\ y_{ref}(s) \end{bmatrix} + e\begin{bmatrix} \sin(\theta_{ref}(s)) \\ -\cos(\theta_{ref}(s)) \end{bmatrix} : \begin{array}{c} s \in S(t) \\ e \in E(t) \end{array} \right\}.$$

This set can be visualized in the (x,y)-plane as follows. For a fixed t, we first choose a grid of points $s \in S(t)$. Then, for each s, we calculate $x_{ref}(s)$, $y_{ref}(s)$, and $\theta_{ref}(s)$ and plot the line segment described by varying e above. The resulting enclosure is shown for several t values in Figure 11 (purple) using $S(t)$ and $E(t)$ from Method (iv). For these results, we used a longer path with $c(s) = 1/30$ for $s \in [0,80]$ m and $c(s) = -1/30$ for $s \in [80,160]$ m and let the initial conditions for e and θ_e be uncertain with $(e_0,\theta_{e,0}) \in [0.8,1] \times [\pi/12,\pi/6]$. Tight enclosures of (x,y) are obtained that remain stable over a long time horizon. However, they become elongated at later times, indicating that deviations normal to the path are tightly bounded, while the position along the path is uncertain. This is expected because v is uncertain.

For path tracking problems, one may be uninterested in the time dependence of the bounds and only concerned with bounding the deviations from the path at each s. This is awkward with the bounds derived above because evaluating the maximum deviation at s requires checking $E(t)$ for every t such that $s \in S(t)$. A simpler approach is to compute bounds as functions of s rather than t. Note that $\dot{s} = \frac{ds}{dt} > 0$ provided that $\theta_e \in (-\frac{\pi}{2},\frac{\pi}{2})$ and $|e| < \frac{1}{\bar{c}}$. Therefore, we may apply a change of variables to (47) to obtain

$$\begin{aligned} \frac{de}{ds} &= \tan\theta_e(1-c(s)e), \\ \frac{d\theta_e}{ds} &= (1-c(s)e)\left(\frac{-g_1\theta_e}{v\cos\theta_e} - \frac{g_2h_2(\theta_e)e}{\cos\theta_e}\right). \end{aligned} \tag{53}$$

Moreover, differentiating (48) gives

$$\frac{d\mathscr{L}}{ds} = -\frac{g_1\theta_e^2(1-c(s)e)}{g_2v\cos(\theta_e)}. \tag{54}$$

Algorithm 5 $\mathscr{R}$ for Example 4.3 using (53)

function $\mathscr{R}(t,Z,P)$
  $(E \times \Theta_e \times L) \leftarrow Z$ and $V \leftarrow P$
  for $i = 1$ to l **do**
    Apply lines 4–8 in Algorithm 3
  end for
  $\Sigma_1 \leftarrow \tan(\Theta_e)(1-\bar{c}E)$
  $\Sigma_2 \leftarrow (1-\bar{c}E)\left(\frac{-5.71\sqrt{V^2+0.1}\Theta_e}{V\cos\Theta_e} - \frac{g_2H_2(\Theta_e)E}{\cos\Theta_e}\right)$
  $\Sigma_3 \leftarrow -(1-\bar{c}E)\frac{5.71\sqrt{V^2+0.1}\Theta_e^2}{g_2V\cos\Theta_e}$
return $\Sigma \leftarrow (\Sigma_1,\Sigma_2,\Sigma_3)$
end function

Figure 11 shows the bounds computed by applying standard DI to (53) and redundancy-based DI to (53) and (54) with invariant (48). The latter methods uses Algorithm 5, which is a straightforward modification of Algorithm 3. The proof that Algorithm 5 satisfies Assumption 2 is analogous to that for Algorithm 3 and is omitted for brevity. Both methods provide bounds on e as a function of s, which can be used to enclose the vehicle position $(x(s),y(s))$ at each s as

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} \in \left\{ \begin{bmatrix} x_{ref}(s) \\ y_{ref}(s) \end{bmatrix} + e\begin{bmatrix} \sin(\theta_{ref}(s)) \\ -\cos(\theta_{ref}(s)) \end{bmatrix} : e \in E(s) \right\}.$$

This is a line segment for each s that can be visualized by plotting its right-most and left-most points (relative

to $\mathbf{n}(s)$) as functions of s. These are given by

$$\begin{aligned} x^{l/r}(s) &= \quad e^{L/U}(s)\sin(\theta_{ref}(s)) + x_{ref}(s), \\ y^{l/r}(s) &= -e^{L/U}(s)\cos(\theta_{ref}(s)) + y_{ref}(s), \end{aligned} \tag{55}$$

where (x^l, y^l) and (x^r, y^r) are the left and right-most points and $[e^L, e^U] = E$.

Standard DI does not produce effective bounds as expected. However, the redundancy-based DI bounds are very tight and offer a slight improvement over Method (iv). Moreover, the cost is only 0.016s. This is much more efficient than, e.g., sampling 1000 trajectories (> 0.35s) and orders of magnitude faster than the real vehicle traverses the path (> 25s for all $v \in V$).

5 Conclusion

Calculating accurate reachable set enclosures is a critical step towards safe motion planning. In this paper, advanced differential inequalities (DI) methods were applied to compute such enclosures for three representative case studies. Naïve application of the standard DI method, which uses only basic interval computations, produced extremely weak bounds. This was caused by significant interval dependency, largely due to the embedded feedback laws. In contrast, redundancy-based DI methods produced greatly improved bounds. With the possible exception of Example 4.2, these bounds appear tight enough to support safe motion planning tasks, and the computational cost was acceptable for online use.

Redundancy-based DI also outperformed the conservative linearization approaches in CORA for our examples. Since CORA uses zonotope calculations that are usually superior to interval arithmetic, this provides further evidence for the effectiveness of invariants. Future developments enabling the use of invariants within conservative linearization techniques could be very impactful, possibly yielding methods with still higher accuracy.

Applying redundancy-based DI required significant problem-specific insights and should not be considered an effective general-purpose solution in its current form. Nevertheless, some strategies were effective across all case studies. First, it was critical to formulate the system in coordinates that made the action of the feedback law at least partially explicit (i.e., through term cancellations or other simplifications). Applying DI to the dynamics of the error coordinates on which the feedback law is based was generally advantageous. For path tracking, it was also better to use the arclength coordinate as the independent variable rather than time. Second, it was critical to manufacture effective nonlinear invariants. The case studies repeatedly showed that using Lyapunov functions as invariants is highly effective. This is likely because the simplified forms of their time derivatives express the action of the feedback law in an explicit manner that can be captured by interval computations. A key drawback of this strategy is that, for many models, Lyapunov functions either don't exist or only remain valid in a small region. However, as shown in Example 4.1, DI does not require true Lyapunov functions and can still make effective use of approximate ones. In the third example, additional invariants also led to significant improvements. In contrast, attempts to use multiple coordinate systems to generate invariants were not effective. Third, we found that custom refinement algorithms were superior to the general-purpose algorithm in [25].

Although our results are generally positive, there is substantial room for improvement in both accuracy and cost. The development of more effective manufactured invariants by researchers with deeper expertise in vehicle dynamics and the underlying geometry could be extremely beneficial. Combining the redundancy-based approach with methods based on more advanced set representations could also be effective, provided costs can be kept low [16, 23–25]. Finally, further optimization of the refinement algorithms could lead to substantial cost reductions.

References

[1] B. Paden, M. Cáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.*, 1(1):33–55, 2016.

[2] A. Broadhurst, S. Baker, and T. Kanade. Monte Carlo road safety reasoning. In *IEEE Intell. Veh. Symp.*, pages 319–324, 2005.

[3] A. Eidehall and L. Petersson. Statistical threat assessment for general road scenes using Monte Carlo sampling. *IEEE Trans. Intell. Transp. Syst.*, 9(1):137–147, 2008.

[4] Y. Zhou and J.S. Baras. Reachable set approach to collision avoidance for UAVs. In *IEEE Conf. Decis. Control*, pages 5947–5952, 2015.

[5] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C.J. Tomlin. A classification-based approach for approximate reachability. In *Int. Conf. Robot. Autom.*, pages 7697–7704, 2019.

[6] S. Kleff and N. Li. Robust motion planning in dynamic environments based on sampled-data Hamilton–Jacobi reachability. *Robotica*, 38(12):2151–2172, 2020.

[7] M. Obayashi and G. Takano. Real-time autonomous car motion planning using NMPC with approximated problem considering traffic environment. *IFAC-Pap.*, 51(20):279–286, 2018.

[8] A.D. Ames, X. Xu, J.W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Automat. Contr.*, 62(8):3861–3876, 2016.

[9] M. Jankovic. Robust control barrier functions for constrained stabilization of nonlinear systems. *Automatica*, 96:359–367, 2018.

[10] M. Prandini and J. Hu. Application of reachability analysis for stochastic hybrid systems to aircraft conflict prediction. In *IEEE Conf. Decis. Control*, pages 4036–4041, 2008.

[11] M. Althoff, O. Stursberg, and M. Buss. Model-based probabilistic collision detection in autonomous driving. *IEEE Trans. Intell. Transp. Syst.*, 10(2):299–310, 2009.

[12] P. Falcone, M. Ali, and J. Sjoberg. Predictive threat assessment via reachability analysis and set invariance theory. *IEEE Trans. Intell. Transp. Syst.*, 12(4):1352–1361, 2011.

[13] R. Kianfar, P. Falcone, and J. Fredriksson. Safety verification of automated driving systems. *IEEE Intell. Transp. Syst. Mag.*, 5(4):73–86, 2013.

[14] M. Althoff and J.M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Trans. Robot.*, 30(4):903–918, 2014.

[15] S.J. Anderson, S.C. Peters, T.E. Pilutti, and K. Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *Int. J. Veh. Auton. Syst.*, 8(2-4):190–216, 2010.

[16] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *IEEE Conf. Decis. Control*, pages 4042–4048, 2008.

[17] N.S. Nedialkov, K.R. Jackson, and G.F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Appl. Math. Comput.*, 105(1):21–68, 1999.

[18] Y. Lin and M.A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Appl. Numer. Math.*, 57(10):1145–1162, 2007.

[19] B. Houska, M.E. Villanueva, and B. Chachuat. Stable set-valued integration of nonlinear dynamic systems using affine set-parameterizations. *SIAM J. Numer. Anal.*, 53(5):2307–2328, 2015.

[20] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff. Ensuring drivability of planned motions using formal methods. In *IEEE IEEE Trans. Intell. Transp. Syst.*, pages 1–8, 2017.

[21] D. Heß, M. Althoff, and T. Sattel. Formal verification of maneuver automata for parameterized motion primitives. In *IEEE Int. Conf. Intell. Robots Syst.*, pages 1474–1481, 2014.

[22] G.W. Harrison. Dynamic models with uncertain parameters. In *Proceedings of the first international conference on mathematical modeling*, volume 1, pages 295–304, 1997.

[23] S. M. Harwood and P. I. Barton. Efficient polyhedral enclosures for the reachable set of nonlinear control systems. *Math. Control Signals Syst.*, 28(1):8, 2016.

[24] B. Chachuat and M. Villanueva. Bounding the solutions of parametric ODEs: When Taylor models meet differential inequalities. In *Comput. Aided Chem. Eng.*, volume 30, pages 1307–1311. 2012.

[25] K. Shen and J.K. Scott. Exploiting nonlinear invariants and path constraints to achieve tighter reachable set enclosures using differential inequalities. *Math. Control Signals Syst.*, pages 1–27, 2020.

[26] J.K. Scott and P.I. Barton. Bounds on the reachable sets of nonlinear control systems. *Automatica*, 49(1):93–100, 2013.

[27] K. Shen and J.K. Scott. Rapid and accurate reachability analysis for nonlinear dynamic systems by exploiting model redundancy. *Comput. Chem. Eng.*, 106:596–608, 2017.

[28] M. Althoff and J.M. Dolan. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *Am. Control Conf.*, pages 3559–3566. IEEE, 2012.

[29] A. Neumaier. *Interval Methods for Systems of Equations*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1991.

[30] B.O.S. Teixeira, J. Chandrasekar, L.A.B. Torres, L.A. Aguirre, and D.S. Bernstein. State estimation for linear and non-linear equality-constrained systems. *Int. J. Contr.*, 82(5):918–936, 2009.

[31] J.K. Scott. *Reachability analysis and deterministic global optimization of differential-algebraic systems*. PhD thesis, Massachusetts Institute of Technology, 2012.

[32] P.I. Barton J.K. Scott. Interval bounds on the solutions of semi-explicit index-one DAEs. Part 2: computation. *Numer. Math.*, 125(1):27–60, 2013.

[33] N. Kochdumper, P. Gassert, and M. Althoff. Verification of collision avoidance for common-road traffic scenarios. In *ARCH@ ADHS*, pages 184–194, 2021.

[34] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31(3):363–396, 2005.

[35] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 384–389, 1990.

[36] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, page 120–151, 2015.

[37] M. Werling, L. Groll, and G. Bretthauer. Invariant trajectory tracking with a full-size autonomous road vehicle. *IEEE Trans. Robot.*, 26(4):758–765, 2010.

[38] C. Samson. Path following and time-varying feedback stabilization of a wheeled mobile robot. In *Proc. Int. Conf. Adv. Robot.*, volume 13, pages 1–14, 1992.

Statements and Declarations

Funding. This material is based upon work supported by the National Science Foundation under grant no. 1949748.

Competing Interests. The authors have no relevant financial or non-financial interests to disclose.

Author Contributions. J. Scott and X. Yang contributed to the study conception and design, algorithm development, and theoretical analyses. X. Yang led the software development and simulations with assistance from B. Mu. Comparisons to CORA were run by D. Robertson. The first draft was written by X. Yang and edited by J. Scott. All authors read and approved the final manuscript.

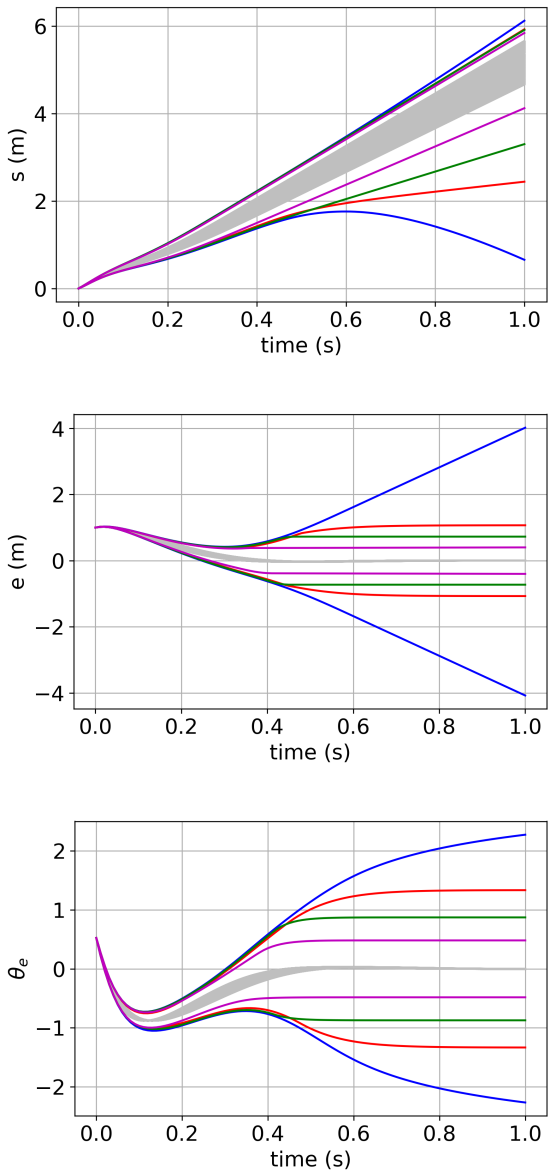

Fig. 10: Example 4.3: Bounds on (s, e, θ_e) from (i) applying standard DI to (47) (blue), (ii&iii) applying redundancy-based DI to (47) and (49) with invariant (48) using Algorithm 3 (green) and the κ-operator [25] (red), and (iv) applying redundancy-based DI to (50) with invariants (51) and (48) using Algorithm 4 (purple), along with 500 sampled trajectories (grey).

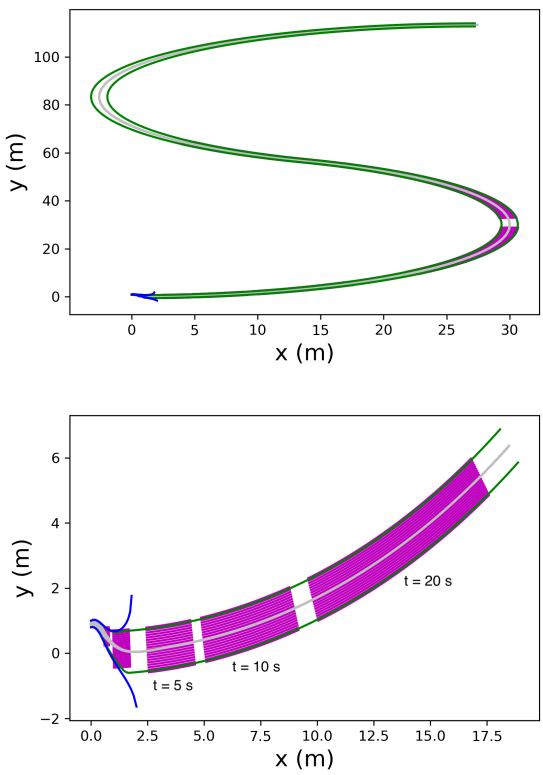

Fig. 11: Example 4.3: Position bounds for the entire path (top) and the beginning of the path (bottom) computed by applying redundancy-based DI to (50) with invariants (51) and (48) (purple shaded), applying standard DI to (53) (blue), and applying redundancy-based DI to (53) and (54) with invariant (48) (green), along with 500 sampled trajectories (grey).